UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE SOFTWARE PARA A MONITORAÇÃO DE DESEMPENHO DE REDES, UTILIZANDO O RMON

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

RICARDO HENRIQUE REKOWSKY

BLUMENAU, NOVEMBRO/1999

1999/2-33

PROTÓTIPO DE SOFTWARE PARA A MONITORAÇÃO DE DESEMPENHO DE REDES, UTILIZANDO O RMON

RICARDO HENRIQUE REKOWSKY

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. Antônio Carlos Tavares

_____

Prof. Miguel A. Wisintainer

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais e familiares, pela vida, pela sabedoria, pela educação, pelo incentivo ao estudo e à pesquisa, e principalmente, pelo carinho e compreensão durante mais esta etapa vencida.

Aos meus amigos, que sempre me motivaram para seguir em frente, me apoiaram nas minhas decisões e fizeram com que eu acreditasse mais em mim do que nunca.

Ao meu professor e orientador Sérgio Stringari, pelo incentivo à pesquisa, por sua dedicação e preocupação pelo fundamento do trabalho e conclusão do mesmo, incentivando me sempre para a obtenção deste sucesso.

A todos aqueles que de alguma forma me ajudaram.

E a Deus, pois sem Ele, não haveria nada neste mundo.

RESUMO

Este Trabalho de Conclusão de Curso (TCC) apresenta um estudo sobre gerência de redes, enfocando a técnica *Remote Monitoring* (RMON) de análise de desempenho a partir do protocolo *Simple Network Management Protocol* (SNMP) em redes locais de computadores. Apresenta também, o desenvolvimento de um protótipo de software de monitoração de desempenho para redes locais de computadores para o sistema operacional Windows NT.

ABSTRACT

This paper demonstrates a study about network management, emphasizing the Remote Monitoring (RMON) technique which provides perfomance analysis based on the Simple Network Management Protocol (SNMP) on local area networks. Furthermore, it shows the implementation of a network management prototype for the local area network operational system Windows NT.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

API - *Apllication Program Interface*

ASN.1 - *Abstract Sintax Notation.One*

CMIP - *Commom Management Information Protocol*

GUI - *Graphical User Interface*

IETF - *Internet Engineering Task Force*

IP - *Internet Protocol*

ISO - *International Organization for Standardization*

LAN - *Local Area Network*

MAN - *Metropolitan Area Network*

MIB - *Management Information Base*

OSI - *Open System Interconection*

PDU - *Protocol Data Unit*

RFC - *Request for Comments*

RMON – *Remote Monitoring*

SMI - *Structure of Management Information*

SNMP - *Simple Network Management Protocol*

TCP - *Transmission Control Protocol*

WAN - *Wide Area Network*

1 INTRODUÇÃO

Ao longo dos anos, a informática tem difundido cada vez mais e tem obtido uma constante melhoria a nível de tecnologia e acessibilidade, mostrando-se indispensável nas tarefas mais simples até as mais complexas dentro de empresas.

O uso de computadores pessoais, denominados de PCs (*Personal Computers*), tem se tornado cada vez mais freqüente, suprindo a necessidade de executar tarefas cada vez mais rápidas e precisas. Os computadores utilizam programas para controlar as diversas atividades executadas no âmbito operacional, podendo haver diversos destes computadores em um único estabelecimento [VEL86].

A necessidade de compartilhar recursos computacionais, como hardware, software e dados, entre diversos computadores e o compartilhamento de informações entre usuários distintos dentro da empresa, fez necessária a interligação destes computadores, através de técnicas denominadas de redes de computadores [CAM97] [MOU86] [TAN94] [THO97].

Cada vez mais as redes de computadores proliferam-se em ambientes com dois ou mais computadores, onde vários usuários podem fazer parte da mesma rede, utilizando-a para o compartilhamento de informações, de periféricos ligados a elas e, também, para armazenar cópias de segurança através da rede. As redes de computadores estabelecem a interligação destes componentes e agilizam a execução de tarefas que antes poderiam levar mais tempo a ser executadas por falta de equipamento, ou até, por causa de algum eventual tempo de espera para que uma tarefa pudesse ser executada [HAM86].

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 influenciar no desempenho da rede, mas quando dezenas de usuários usufruem destes recursos poderá haver uma queda de desempenho significativa através da sobrecarga da rede (*overhead*), prejudicando os serviços e a operacionabilidade da rede [HAM86] [MOU86].

Gerenciar uma rede de computadores significa tratar de condições anormais e problemas relacionados a atividades a serem inicializadas, em andamento ou em término para

melhor utilização da rede. Muitas vezes o gerenciamento pode ser confundido com gerenciamento de falhas. Existem diversos fatores gerenciáveis em uma rede de computadores, como controle de acesso a rede, gerenciamento de configuração e gerenciamento de desempenho, entre outros [BRI94] [HEL92] [MOU86].

Para se obter uma arquitetura de gerenciamento que possa atender todos estes quesitos mencionados acima e demais fatores gerenciáveis em uma rede e que tenha as características de integração, simplicidade, segurança e flexibilidade fez com que a *International Organization for Standartization* (ISO) apresentasse uma especificação padronizada de arquitetura de gerenciamento de rede OSI [BRI94].

Através da arquitetura OSI de gerenciamento define-se uma estrutura de gerenciamento, os componentes de gerenciamento, os serviços e o protocolo para a troca de informações de gerenciamento. A forma de tratamento dos dados e a apresentação dos resultados a um operador não fazem parte do escopo de padronização OSI por serem consideradas questões locais [BRI94] [HEL92].

Como parte do gerenciamento de rede OSI há o gerenciamento de desempenho que tem como objetivo principal, a avaliação de desempenho da rede e adequação dos meios de comunicação utilizados pelos usuários às suas reais necessidades [HAM86] [HEL92] [BRI94].

Para a avaliação de desempenho da rede deve-se estabelecer critérios de avaliação apropriados. Estes critérios podem ser, por exemplo, a média de *bits* que trafega pela rede dentro de um certo intervalo de tempo ou o atraso na transferência de informações [HEL92] [HAM86] [MAR94].

O protocolo *Simple Network Managment Protocol* (SNMP) foi projetado para permitir que o software da estação de gerenciamento de rede comunicasse-se com os agentes nos dispositivos gerenciados. Para confirmar que o tráfego devido ao gerenciamento de rede é mínimo, o SNMP é implementado utilizando abordagem cliente/servidor assíncrona. Isso significa que uma entidade SNMP (estação de gerenciamento ou dispositivo gerenciado) não precisa esperar por uma resposta depois de enviar uma mensagem. Ele pode enviar outra mensagem se necessário, ou continuar com suas funções predefinidas [SOA95].

O *Remote Monitoring* (RMON) utiliza o protocolo SNMP de gerência de redes para monitorar as operações básicas de redes Ethernet e Token Ring. O primeiro padrão RMON, RFC 1757, define dois grupos específicos Ethernet e sete outros grupos que se aplicam tanto a Ethernet como a Token Ring. O segundo padrão, RFC 1513, define extensões Token Ring para RMON. Com esses dois primeiros padrões, o RMON criou uma base para futuras extensões da *Management Information Base* (MIB). Hoje, o padrão inclui 13 grupos Ethernet e Token Ring que contribuem para uma meta comum: permitir o monitoramento de todas LANs com agentes baseados em RMON independentes de qualquer marca ou fornecedor [MAR94].

O RMON permite monitorar uma rede, mesmo que algum segmento dela não esteja funcionando corretamente, isto é, não possua conexão com as demais estações conectadas à rede. Para este monitoramento é necessário observar-se a implementação deste protocolo em vários pontos da rede para permitir um gerenciamento *off-line*, ou seja, para quando estações não estiverem conectadas à rede possa-se coletar informações a seu respeito, mesmo que estas informações não sejam enviadas no momento para a estação gerente. Uma estação pode, também, apenas enviar as informações à estação gerente apenas quando houver alguma anormalidade, diminuindo assim a existência de uma sobrecarga na rede.

Este Trabalho de Conclusão de Curso visa o desenvolvimento de um protótipo de software para monitorar o desempenho de rede de computadores, pretendendo identificar que tipos de pacotes (entrada/saída) estão sendo mais utilizados e a visualização através de gráficos, podendo, assim, auxiliar na monitoração do desempenho da rede.

Para a especificação do protótipo utilizou-se a técnica de fluxogramação, representando graficamente o funcionamento do protótipo do sistema implementado [GAN83].

Como ferramenta para a implementação foi utilizado o ambiente de programação Delphi versão 4.0 [CAN98].

1.1 MOTIVAÇÃO

O fascínio e a curiosidade sobre a área de Redes de Computadores, em especial a de Gerência de Redes foi o principal incentivo para a realização deste Trabalho de Conclusão de

Curso. Ajudar a solucionar problemas de desempenho nas redes de computadores, com o intuito de utilizá-lo como uma ferramenta auxiliar na administração de redes de computadores.

1.2 ÁREA

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

1.3 PROBLEMA

O grande crescimento do uso de microcomputadores proporciona uma sobrecarga nas redes em ambientes de trabalho. A rede tende-se a tornar-se mais lenta devido ao uso desapropriado de seus recursos fazendo com que o desempenho seja comprometido.

1.4 OBJETIVOS DO TRABALHO

Os objetivos principais do trabalho são especificar e implementar um protótipo de software para a monitoração de desempenho de redes locais de computadores com o Windows NT, utilizando a técnica de análise de desempenho RMON.

1.5 ORGANIZAÇÃO DO TRABALHO

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

O capítulo 1 introduz o contexto geral do trabalho divididos nos segmentos: introdução, motivação, área, problema, objetivos do trabalho.

O capítulo 2 enfoca as redes de computadores, os tipos e arquiteturas de redes mais relevantes para este TCC.

O capítulo 3 fundamenta o tópico gerenciamento de redes subdividido em considerações, conceitos e funcionamento sobre modelos e protocolos de gerenciamento de redes.

No capítulo 4 é descrito o monitoramento remoto, o RMON, seus fundamentos e descrição dos grupos RMON.

O capítulo 5 é dedicado exclusivamente as informações referentes ao desenvolvimento do protótipo, contendo detalhes da especificação (técnicas utilizadas), implementação (ambiente de programação e sistema operacional utilizado) e um detalhamento sobre o funcionamento do protótipo desenvolvido.

O capítulo 6 descreve as conclusões obtidas neste TCC e sugestões para extensões e trabalhos futuros a serem realizados.

Para finalizar este TCC, apresenta-se 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 muito repentinamente. Os equipamentos estão ficando cada vez mais potentes e de auxílio quase que indispensável para as empresas. Cada vez mais utiliza-se computadores para agilizar tarefas rotineiras através de ferramentas mais sofisticadas. Para muitas empresas surgiu a necessidade de que os computadores estivessem interligados de alguma forma aos demais computadores existentes dentro da corporação, para que alguns recursos como impressoras e unidades de *backup*, por exemplo, estivessem disponíveis para todos os usuários e não fossem mais privilégio de apenas alguns.

Uma grande parte dos usuários de redes visualizam a rede como um grande dor de cabeça. Muitos não entendem porque precisam dela e se perguntam por que não podem ter o equipamento desejado em suas máquinas. A falta de treinamento destes usuários da rede, também pode levar ao uso inadequado da rede, sobrecarregando-a e 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 [CAM97] [SZT99].

Com freqüência, o usuário não percebe que uma rede pode ser complexa e exige que todas as aplicações estejam disponíveis na rede no momento que desejar, para assim, agilizar sua tarefa e obter as informações mais rápidas a qualquer momento [SZT99].

A rede não está isenta à falhas ou erros inesperados. As dúvidas e problemas do dia-a-dia de usuários de rede, fez com que surgissem alternativas para resolver tais problemas.

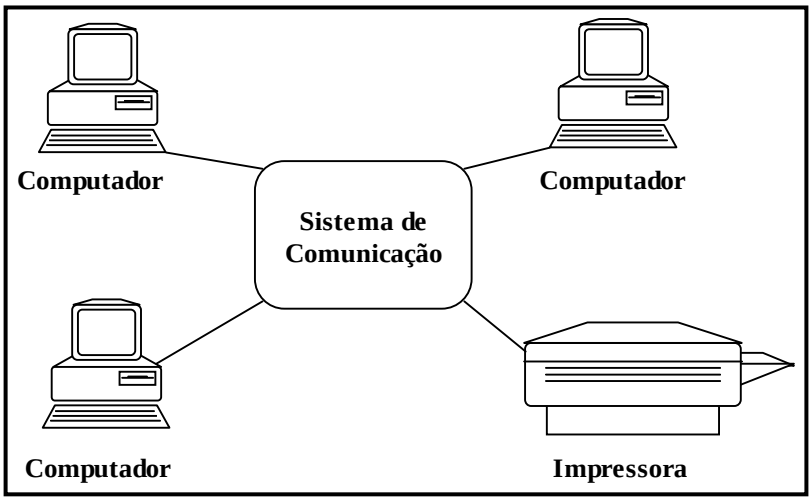
Através de ferramentas auxiliares, os administradores de rede procuram prevenir e solucionar problemas existentes. Ao longo deste trabalho será demonstrado uma das formas com a qual isto é possível.

2.1 HISTÓRICO DAS REDES DE COMPUTADORES

Visando o compartilhamento de recursos e demais periféricos surgiu a rede de computadores e os sistemas em rede.

Segundo [SOA95], uma rede de computadores deve ser capaz de trocar informações e compartilhar periféricos entre estações de trabalho interligadas através de um sistema de comunicação, conforme a fig. 1.

FIGURA 1: Rede de Computadores.



2.2 TIPOS DE REDES

Segundo [CAM97], [KEE95], [SOA95], [TAN94] e [THO97], existem atualmente basicamente três tipos de tecnologias disponíveis para redes de computadores:

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

As LANs emergiram para possibilitar o compartilhamento de recursos (tanto de *hardware* como *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 [SOA95] [TAN94].

Segundo [SOA95] e [TAN94], 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 estão, segundo [SOA95], [TAN94] e [THO97]:

- Abrangência geográfica limitada (distâncias menores que 25 km);
- Altas taxas de transmissão;
- Baixas taxas de erro;
- Possuem pequenos atrasos de transmissão;
- Facilidade de interconexão entre redes distintas (preservação das redes originais);
- Geralmente são redes privadas.

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 [SOA95] [TAN94].

Como a mais nova das três tecnologias de redes aqui apresentadas, as MANs surgiram com o objetivo de viabilizar o compartilhamento de recursos, tanto de hardware como de software, entre redes [SOA95] [TAN94].

c) Redes Geograficamente Distribuídas (*Wide Area Networks* – WANs)

As WANs são as pioneiras entre as redes de computadores. Elas 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”, segundo [SOA95].

As WANs possuem um custo de comunicação elevado, por isso a taxa de transmissão é baixa, mesmo que em alguns *links* cheguem hoje a velocidades 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 [CAM97].

2.3 ARQUITETURAS DE REDES

Através da evolução de vários projetos de redes, tentando alcançar uma forma estruturada de visualização e representação das diversas arquiteturas de redes, procurou-se estruturar a rede como um conjunto de camadas hierárquicas, divididas em suas funções e serviços oferecidos.

Segundo [SOA95], “a arquitetura de rede é formada por níveis, interfaces e protocolos, onde cada nível oferece um conjunto de serviços ao nível superior, utilizando funções realizadas no próprio nível e serviços disponíveis nos níveis inferiores”.

A padronização das diversas arquiteturas de redes foi estabelecida pela *International Organization for Standartization* (ISO) que definiu o modelo de referência para sistemas abertos de interconexão, propondo uma estrutura com sete níveis como referência para a arquitetura dos protocolos de redes de computadores [SOA95] [TAN94].

A existência de diversos tipos de redes dificultava a interconexão de redes. Para solucionar este problema e obter a interconexão entre os diversos tipos redes utilizou-se a arquitetura Internet, baseada no protocolo *Transmission Control Protocol / Internet Protocol* (Protocolo de controle de transmissão/Protocolo Internet - TCP/IP) [SOA95] [TAN94].

A seguir são demonstradas um exemplo de arquitetura de camada, o *Ethernet* e o método de acesso *Token Ring*.

a) Ethernet

A rede *Ethernet* é a mais conhecida entre as atualmente utilizadas, e, está no mercado há mais tempo do que as outras tecnologias de rede. A redução dos preços e uma relativa alta velocidade de transmissão de dados aumentaram a utilização da *Ethernet*. Ela pode ser utilizada com a topologia de barramento (Coaxial) ou Estrela (Par trançado com HUB).

Neste tipo de rede, cada PC “escuta” o tráfego na rede e se não escutar nada, eles transmitem as informações. Se dois clientes transmitirem informações ao mesmo tempo, eles são alertados sobre a possibilidade de colisão, param a transmissão e esperam um período aleatório para tentar transmitir novamente. Este método é conhecido como *Carrier Sense*

Multiple Access with Collision Detection (Acesso Múltiplo com Detecção de Portadora/Detecção de Colisão - CSMA/CD) [CAM97] [SOA95] [TAN94].

b) Token Ring

O método de acesso de *token ring* (passagem de permissão) utiliza um método circular para determinar qual estação tem permissão para transmitir. O *token ring* opera na topologia em anel e garante que todas as estações da rede tenham chance de transmitir dados. Ele alcança esse objetivo utilizando o padrão conhecido como *token* ou permissão.

Em uma rede *token ring*, o computador monitora a rede até reconhecer o padrão denominado de permissão. Ao perceber a transmissão, o computador envia um pacote de dados. Este pacote de dados trafega pelo anel e o destinatário recebe a passagem. Quando o pacote retornar ao transmissor ele passa o *token* para a próxima estação. Este processo repete-se infinitamente. Os tempos necessários são medidos em frações de segundos [CAM97] [SOA95] [TAN94].

Mais informações sobre arquiteturas de rede podem ser encontradas em [CAM97] [SOA95] [TAN94].

3 GERENCIAMENTO DE REDES

Com a utilização mais freqüente de microcomputadores e com o aumento de seu uso em ambientes de trabalho corporativos e escritórios, por exemplo, observou-se a necessidade de compartilhar os diversos recursos disponíveis na empresa, como impressoras, *drives* de CD-ROM e discos rígidos, entre outros, de tal forma que, tornassem-se comunitários para todos os usuários daquele ambiente de trabalho. Para tanto, precisava-se pensar em uma maneira organizada de distribuir os recursos para que eles pudessem ser aproveitados da melhor maneira possível. Ao longo deste capítulo será apresentada a evolução da gerência de redes e suas conseqüências.

3.1 CONSIDERAÇÕES SOBRE GERÊNCIA DE REDES

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 mais 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 [RAB99] [SOA95].

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. Algumas características principais, que devem ser consideradas para a implementação de um sistema do inteligente, serão apresentadas ao decorrer deste trabalho [RAB99].

Atualmente existem diversos conceitos relacionados ao gerenciamento de redes. Para entender melhor seu significado é preciso um profundo estudo sobre o assunto.

Segundo [HEL92], 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.

O gerenciamento de redes fornece mecanismos para monitoração, controle e coordenação da utilização de recursos em um ambiente *Open Systems Interconnection* (OSI) e define padrões de protocolo OSI para troca de informações entre estes recursos, segundo [BRI94], [RIG96] e [SOA95].

Segundo [RNT91], “o gerenciamento de redes é o conjunto de funções que visa promover a produtividade da planta e dos recursos disponíveis e integrar, de forma organizada, as funções de operação, administração e manutenção de todos os elementos da rede e dos serviços de telecomunicações”.

Os sistemas de gerenciamento atuais são baseados na interação de vários componentes da rede. A seguir podem ser vistos os principais componentes [TAN94] [ZAK88]:

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

b) *Agentes*: são “pequenos” programas que residem nos elementos da rede que devem ser gerenciados. Os agentes coletam e armazenam diversas informações de gerenciamento. Mais sobre agentes ver item 3.6.5;

c) *Objetos gerenciados*: são quaisquer elementos que possam ser gerenciáveis;

d) *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;

e) *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;

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

g) *Interfaces de programas aplicativos* (*Application Program Interface – API)*: através das APIs é possível interagir com o sistema operacional utilizado;

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

Apesar dos conceitos serem distintos existem alguns aspectos em comum entre eles. Os autores destes conceitos são unânimes quando mencionam que o aumento da complexidade das redes exige que seu gerenciamento seja feito de maneira simples, através de ferramentas de gerenciamento amigáveis e modernas, para o usuário. Além disso, mencionam a lentidão em novos produtos e ferramentas para o âmbito das redes de computadores.

O gerenciamento de redes e a necessidade de organização dos conceitos e estruturas resultaram em dois modelos básicos de gerenciamento de redes, conforme podem ser vistos nos itens a seguir.

3.2 O 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 – ISSO) 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 [MAF99] [SZT96].

Nele define-se um modelo funcional onde cada área tem um conjunto de funções, que ao serem implementadas, que serão utilizadas para gerenciar a rede.

No gerenciamento OSI existem cinco áreas funcionais, que quando implementadas podem ser utilizadas para o gerenciamento da rede. Estas áreas funcionais são mostradas a seguir [MAF99] [SZT96] [WIE97]:

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

b) *Gerência de desempenho*: utiliza vazão e taxa de erros para representação e cálculos dos dados;

c) *Gerência de falhas*: analisa o 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.

Relacionando o modelo de gerenciamento OSI, quanto à sua estrutura, ele ainda é dividido em três tipos principais [BRI93] [SZT96]:

a) *Gerenciamento de sistemas*: sendo executado na camada de aplicação, precisa de apoio das setes 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*: é utilizada no gerenciamento de uma única instância de comunicação em uma camada e utiliza-se do protocolo normal da camada para troca de informações e não precisa de um protocolo especial para poder realizar estas trocas de informações.

Relacionado o protocolo de gerenciamento OSI e a MIB, verifica-se que a MIB guarda as informações transferidas ou modificadas pelo uso de protocolos de gerenciamento OSI. Estas 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 o agente e os objetos gerenciados situam-se no mesmo ambiente local [MAF99] [SZT96].

Existem três tipos de hierarquias de gerenciamento utilizadas pelos Sistemas de Gerenciamento OSI: Herança, Nomeação e Registro. Estes três tipos serão demonstrados a seguir.

a) Hierarquia de Herança

A hierarquia de herança, também denominada de hierarquia de classe, está relacionada com as propriedades associadas aos objetos, descritas através de seus atributos, comportamento, pacotes condicionais, operações e notificações. Nesta hierarquia define-se o conceito de classes de objeto hierarquizadas às quais pertencem objetos com propriedades similares [LIM99] [MAF99].

b) Hierarquia de Nomeação

A hierarquia de nomeação, também chamada de hierarquia de *containment*, descreve o relacionamento entre instâncias de objetos e seus respectivos nomes.

Nesta hierarquia, define-se um relacionamento "estar contido em" aplicado aos objetos. Cada objetos de uma classe específica pode conter objetos de uma mesma classe e/ou de classes distintas. Um objeto gerenciado que contém outro objeto é denominado de *superior* e o objeto que é contido é denominado de *subordinado*. Um objeto gerenciado está contido dentro de um e somente um objeto gerenciado superior [LIM99] [MAF99].

Este relacionamento de *containment* pode ser utilizado para modelar hierarquias de partes do mundo real, como por exemplo, módulos, submódulos e componentes eletrônicos, ou ainda, hierarquias organizacionais, como por exemplo, diretório, arquivos, registros e campos [LIM99] [MAF99].

Desta forma entende-se que o objeto gerenciado existe somente se o seu objeto superior existir (dependência direta) e que todo objeto gerenciado tem um nome que é derivado de um relacionamento de *containment* [LIM99] [MAF99].

c) Hierarquia de Registro

A hierarquia de registro é utilizada para a identificação de objetos, independentemente das hierarquias de heranças e nomeação. Esta hierarquia é especificada segundo as regras estabelecidas pela notação *Abstract Syntax Notation* (Notação de sintaxe abstata - ASN.1) para árvore de registros utilizada 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 [LIM99] [MAF99].

Ainda como parte do modelo de gerenciamento OSI, há o protocolo de informação de gerência de comunicação *(Communications 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 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. Estes comandos são similares aos comandos especificados no protocolo SNMP, mas com a diferença de que o CMIP possui um conjunto superior e maior de comandos em relação ao SNMP. O CMIP foi projetado para abranger um nível de capacidade e de detalhes maior do que o SNMP [BRI94] [TAN94].

Todas as informações obtidas através dos agentes são armazenadas na Base de Informações de Gerenciamento (*Management Information Base* - MIB), bem como as informações sobre o funcionamento dos *hosts* e demais componentes. A MIB é exemplificada e demonstrada no item 3.6.2 ao decorrer deste trabalho.

Maiores informações a respeito do CMIP podem ser obtidas em.

Mais informações e detalhamento mais aprofundado sobre o modelo de referência OSI para gerenciamento de redes e sobre o CMIP podem ser obtidas em [BRI93], [BRI94], [HEL92], [LIM99], [MAF99], [SOA95], [SZT96] e [TAN94].

3.3 O PROTOCOLO SNMP

Segundo [HEL92], inicialmente, o protocolo *Simple Network Management Protocol* (Protocolo de Gerenciamento de Redes Simples – SNMP) foi desenvolvido por quatro pessoas como um veículo (meio) de gerenciamento de grandes sistemas interconectados durante o ano de 1987. Ao final de 1988 começavam a aparecer os primeiros produtos utilizando o protocolo SNMP. Desde então, várias empresas anunciaram o suporte ao SNMP inclusive a IBM com o seu módulo *Netview*® que possibilita monitorar e controlar produtos compatíveis com o SNMP.

O protocolo *Simple Network Management Protocol* (Protocolo de gerenciamento de redes simples - SNMP) é o padrão de gerenciamento de redes reconhecido pela comunidade internacional, no tocante a sua funcionalidade. Ele faz parte integrante do conjunto do protocolo *Transmission Control Protocol / Internet Protocol* (Protocolo de Controle de transmissão/Protocolo Internet - TCP/IP) e foi originalmente desenvolvido com a finalidade de monitorar , achar e resolver problemas de roteadores e *bridges*. Roteadores e *bridges* são dispositivos de *hardware* e *software* utilizados para ligar diferentes segmentos em uma rede. O SNMP opera na camada de aplicação da arquitetura Internet TCP/IP [CAMP97] [SOA95] [TAN94].

3.3.1 FUNCIONAMENTO DO SNMP

Segundo [BRI94], [HEL92], [MEI99], [RFC1157], [SOA95], e [SZT96], o modelo de referência SNMP consiste de quatro principais componentes:

a) Interconexão de redes;

b) Protocolos de rede;

c) Aplicação gerente;

d) Aplicação agente.

Maiores informações referente aos itens a) e b) podem ser obtidos nos itens 3.0 e seguintes e, ainda, nos livros [BRI94], [HEL92], [MEI99], [SOA95], e [SZT99].

Os itens c) e d) serão descritos a seguir.

O SNMP fornece habilidade para monitorar e informar o *status* de comunicação entre:

a) Computadores executando Windows 95/98/NT, por exemplo;

b) Servidores LAN;

c) Roteadores ou *gateways*;

d) Minicomputadores ou computadores *mainframe*;

e) Servidores de terminal (*Terminal Servers*);

f) Ligando *hubs* (*Wiring hubs*).

O SNMP utiliza uma arquitetura distribuída de gerenciamento de sistemas e agentes.

O serviço SNMP envia uma informação para um ou mais *hosts* quando um *host* solicitar ou quando um evento significativo ocorrer, por exemplo, quando um *host* está pesquisando o espaço em disco rígido.

As implementações fundamentais do SNMP permitem monitorar e isolar falhas. Em contrapartida as aplicações mais sofisticadas permitem gerenciar o desempenho e a configuração da rede. Em geral, estas aplicações incorporam menus e alarmes codificados em cores. O detalhamento desta funções será feito no item 3.3.2 a seguir.

3.3.2 DESCRIÇÃO DO SNMP

Segundo [BRI94], [HEL92] e [TAN94] classificam-se os elementos de rede como monitor (cliente ou gerente) e agentes (servidores). O primeiro é responsável pela monitoração e o controle dos *gateways* e *hosts*, que correspondem aos agentes.

De acordo com [BRI94] e [MEI99], o protocolo SNMP é baseado na técnica conhecida como "busca-armazenamento" (*fetch-store*), isto é, todas as operações previstas para este protocolo são derivadas de operações básicas de busca e armazenamento. Estas operações básicas corresponde a:

a) *get-request:* leitura do valor de uma variável;

b) *get-next-request:* leitura do valor da próxima variável;

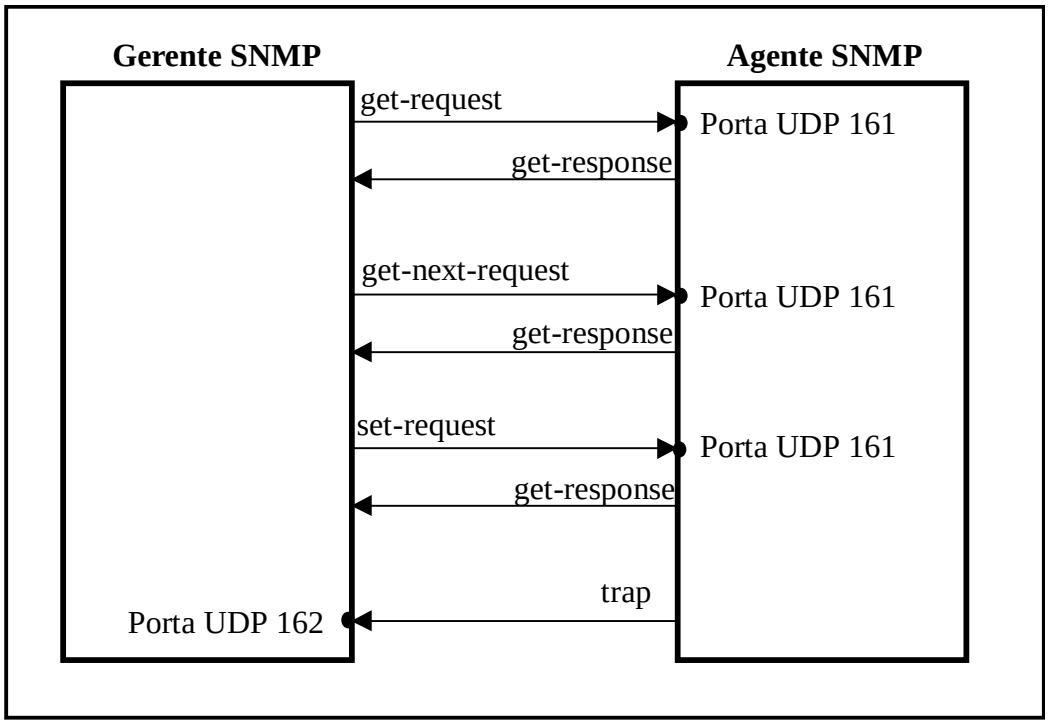
c) *get-response:* resposta à operação de leitura (get-request on get-next-request);

d) *set-request:* gravação do valor de uma variável;

e) *trap:* notificação da ocorrência de um evento específico.

O *trap* permite aos servidores SNMP enviar informações aos clientes sempre que ocorrer algum tipo de evento pré-definido que possa informar a ocorrência de alterações nos objetos, como por exemplo, erros, falhas ou quaisquer operações anormais. Veja na figura 2 a seguir a representação gráfica das operações do protocolo SNMP.

FIGURA 2: Resumo das operações SNMP.



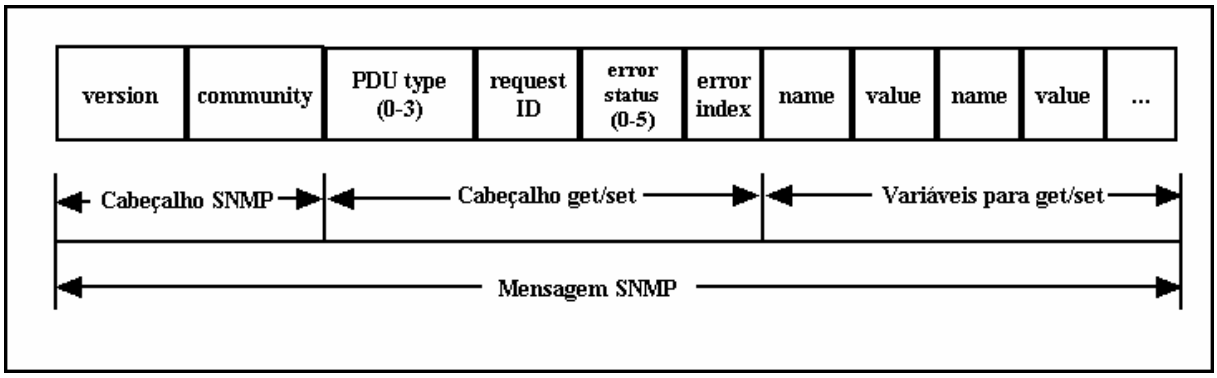
Segundo [BRI94] e [MEI99], as mensagens deste protocolo não possuem campos fixos e são especificadas na notação *Abstract Syntax Notation* 1 (ASN.1 – ver item 3.6.4 para mais detalhes) para a mensagem e a resposta. Elas consistem de três partes principais:

a) **Versão de protocolo (Cabeçalho SNMP):** Identificador da comunidade SNMP, que é utilizado para permitir o acesso à objetos gerenciados através de um servidor SNMP por meio de um cliente;

b) **Área de dados:** Para cada uma das operações citadas acima é definido um tipo específico de mensagem de protocolo, isto é, um tipo de *Protocol Data Unit* (PDU). Cada PDU é constituída por requisição de um cliente ou por uma resposta de um pedido.

Os agentes SNMP executam operações elementares, como estabelecer e obter valores das variáveis. O programa que analisa, manipula, combina ou aplica algum algoritmo sobre os dados que devem residir no monitor (gerente). Na fig. 3, abaixo, é apresentado o formato para mensagens SNMP.

FIGURA 3: Formato para mensagens SNMP.



Explicação das variáveis da fig. 3.

a) *Version*: versão SNMP utilizada para criar a mensgem;

b) *Community*: definição textual que contem o nome da comunidade para a mensagem. É utilizada freqüentemente como uma senha entre a estação gerente e a agente. Por padrão a variável recebe o valor público (*public)*;

c) *PDU Type*: Os valores possíveis para este tipo de mensagem são *GetRequest*, *GetRequest*, *GetNextRequest*, *SetRequest*, e *Response* para a versão 1 do SNMP. A versão 2 também podem ter as variáveis *GetBulkRequest* e *InformRequest*;

d) *RequestID*: – Uma identificação única associada ao gerente para relacionar requerimentos e respostas;

e) *ErrorStatus*: Representa o *status* do erro da mensagem. É somente utilizado em mensagens de resposta. Para os valores *GetRequest*, *GetNextRequest*, *GetBulkRequest*, *InformRequest* e *SetRequest* este campo deve ser ignorado;

f) *ErrorIndex*: Indica qual foi a provável variável que originou a falha de uma requisição. Utilizada somente para mensagens de reposta. Para os valores *GetRequest*, *GetNextRequest*, *GetBulkRequest*, *InformRequest* e *SetRequest* este campo deve ser ignorado;

g) Variáveis para *get/set*: Uma seqüência de pares de nome-valores (name-value) representando os dados que estão sendo requeridos ou respondidos. Na requisição os valores são ignorados e na resposta eles são definidos.

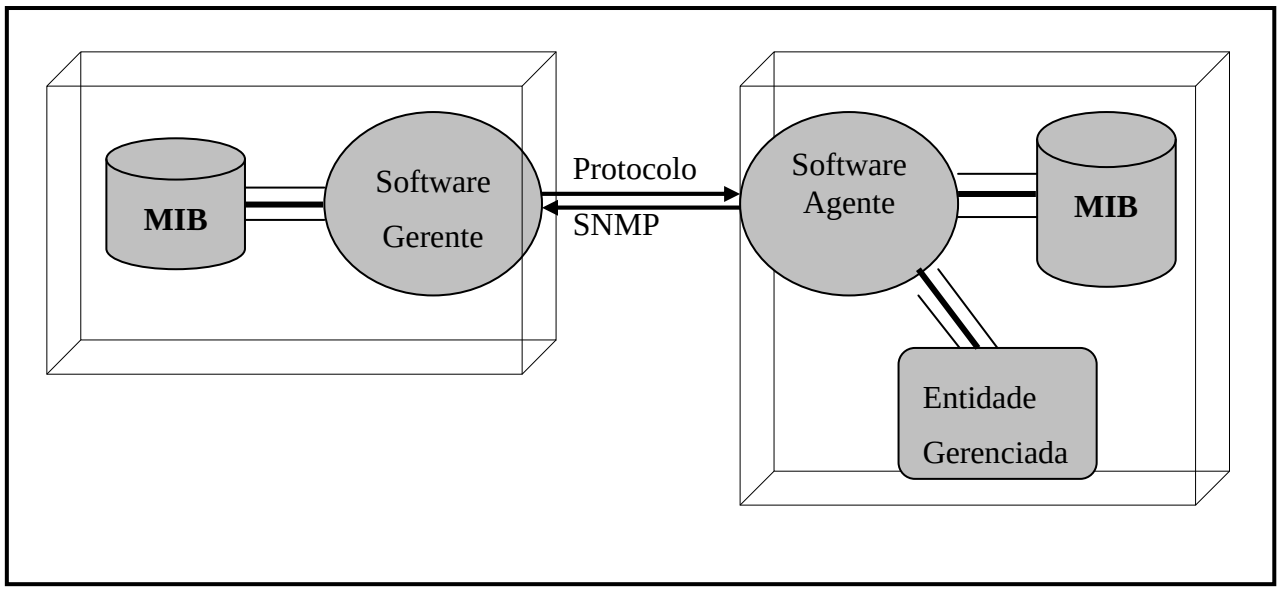
3.4 COMPARATIVO DOS PROTOCOLOS SNMP E CMIP

Esses dois modelos possuem, basicamente, seus objetivos em comum: transferência de informações entre sistemas de gerenciamento de redes de tal forma que os agentes possam obter as informações mais importantes para o gerenciamento junto aos objetos gerenciados. Entende-se por objeto gerenciado a representação de um recurso, seja um sistema hospedeiro (estação de trabalho), *gateways* ou equipamentos que servem para transmissão, como o modem por exemplo. Já os gerentes processam as informações obtidas pelos agentes em busca de detecção de presença de falhas no funcionamento de qualquer um de seus componentes, como por exemplo nos *hosts* [BRI94] [SOA95].

3.5 GERENTES, MIB, ESTRUTURAS DE DADOS E AGENTES

A arquitetura de gerência de rede em sistemas abertos introduz o conceito de gerente e agente, os quais trocam informações entre si a respeito dos objetos gerenciados, através dos protocolos específicos de cada arquitetura e de uma estrutura de dados denominada MIB. Esta relação entre gerente, agente e MIB, é mostrada esquematicamente na fig. 4 [BRI94], [HEL92], [MEI99], [SOA95] e [TAN94]:

FIGURA 4: Esquema simplificado da relação Gerente, Agente e MIB



As palavras gerentes, agentes e MIB já foram mencionadas ao longo deste trabalho. Para entender estas expressões melhor, segue um detalhamento a seu respeito a seguir, baseado nos autores dos livros [BRI94], [HEL92], [MEI99], [SOA95] e [TAN94].

3.5.1 GERENTES

Os gerentes (ou monitores) são programas de gerenciamento SNMP. Estes gerentes coletam diversos dados sobre os segmentos a serem monitorados em uma rede tornando estas informações disponíveis para o administrador da rede através de textos, de visualização gráfica ou através de interfaces orientado a objeto. O programa gerente envia mensagens para a rede em busca de informações. Essas mensagens são recebidas pelo agente no *host*, que inicializa as operações *get*, *get-next* e *set*. O programa gerente aguarda as mensagens provenientes do agente que contém os resultados das operações, e mostra-as ao módulo gerenciador ou salva os dados em um arquivo especificado ou em uma base de dados.

Segundo [MEI99], uma aplicação (programa) gerente poderá ser constituída de quatro componentes principais:

a) Operações de gerenciamento;

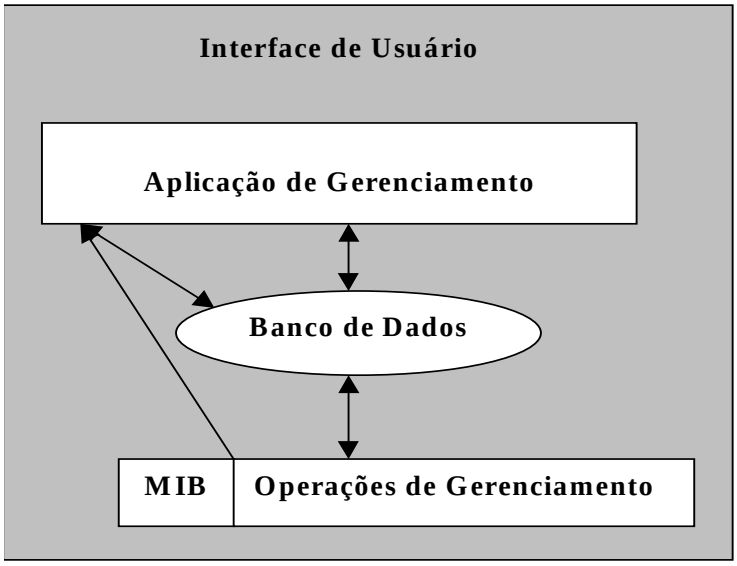
b) *Management Information Base* (MIB – Base de Informações de Gerenciamento);

c) Banco de Dados;

d) Interface com o usuário.

O gerenciamento de várias estações de trabalho é uma tarefa que requer intensos recursos de máquinas por causa do grande consumo de processamento por parte das aplicações gerentes. Por este motivo, as implementações normalmente ocorrem em uma estação de trabalho utilizando o ambiente Windows NT ou UNIX com bastante capacidade de memória RAM (*Random Access Memory* – Memória de Acesso Randômico), espaço de armazenamento e dispositivos de *backup* seguros. Segue a visualização do esquema simplificado do módulo gerente conforme a fig. 5.

FIGURA 5: Principais componentes de uma aplicação gerente



3.5.2 MIB

Segundo [MEI99] e [SZT99], a *Management Information Base* (MIB – Base de Informações de Gerenciamento) é um conjunto de objetos que podem ser gerenciados e seus atributos. A MIB é o elemento de ligação entre o *software* gerente, situado na estação de gerenciamento e agente, em um sistema de gerência, onde estão localizadas as estruturas de dados através das quais as informações podem ser passadas entre eles, utilizando o protocolo SNMP. A MIB tem como objetivo descrever os objetos que podem ser gerenciados em uma

rede e é uma estrutura que se constitui na visão que o gerente tem dos objetos gerenciáveis dos elementos ou segmentos disponíveis rede.

Basicamente, são quatro os tipos de MIBs: MIB I, MIB II, MIB experimental e MIB privada. Os fundamentos a seguir baseiam-se em [MEI99] e [SZT99].

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, que introduziu novas informações além daquelas encontradas na MIB I. Através das MIBs do tipo I e II é possível obter informações como o tipo e o *status* da interface (Ethernet, Token-Ring), número de pacotes transmitidos, número de pacotes com erros, informações sobre protocolo de transmissões, 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. Estes grupos são mostrados nas tabelas a seguir, sendo a tabela 1 referente a MIB-I e a tabela 2 referente a MIB-II, segundo [COM98] e [SZT99]:

TABELA 1: Categorias de informações na MIB-I.

Categoria da MIB	**Objetos contidos para**
System (sistema)	Informações básicas sobre o sistema operacional ou referentes ao *host*
Interfaces	Interfaces de redes
at (conv. end.)	Conversão (interpretação/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 na MIB-II.

Categoria da MIB	Objetos contidos para
System (sistema)	Informações básicas sobre o sistema operacional ou referentes ao *host*
Interfaces	Interfaces de redes
at (conv. End.)	Conversão (interpretação/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)
Transmiss	Transmissão média-específica
Snmp	Aplicações SNMP

As MIBs experimentais são aquelas, como diz o nome, que encontram-se em fase de testes e experimentação, com a perspectiva de serem adicionadas ao padrão e que, genericamente, fornecem características mais específicas e precisas sobre a tecnologia dos meios de transmissão e equipamentos empregados [MEI99] [SZT99].

As MIBs privadas são específicas dos equipamentos gerenciados, possibilitando que detalhes peculiares a um determinado equipamento possam ser obtidos para uma análise futura de seus dados. Assim, é possível obter-se informações sobre colisões, configuração, *swap* de portas, só para citar algumas delas, de um *hub*. Também é possível realizar testes, reinicializar ou desabilitar uma ou mais portas do *hub* através de MIBs proprietárias [MEI99] [SZT99].

A construção das MIBs segue as especificações contidas nos documentos denominados *Structure of Management Information* (Informações de Gerenciamento de Estruturas – SMI) para a versão 1 do SNMP, para o SNMPv2 e agora mais recentemente para o SNMPv3. Estas especificações impõem um grau de simplicidade elevado na descrição das MIBs para que a implementação destas MIBs tornem-se cada vez mais simples de serem realizadas [COM98] [MEI99] [SZT99].

A unidade básica de acesso em uma MIB SNMP é representada por um atributo simples, que é denominado de objeto, definido por um tipo e pelas operações que podem ser realizadas sobre ele. A unidade básica de acesso é uma classe de objeto, definida por um conjunto de atributos e de operações. Embora nas MIBs SNMP não exista a definição de classes, os objetos são organizados em grupos que representam uma determinada funcionalidade de um elemento de rede, podendo em cada grupo existir tabelas de objetos limitados, de no máximo, duas dimensões [COM98] [SZT99].

Estas categorias são utilizadas quando é mantida a definição da MIB para obter-se uma interoperabilidade entre diversos *softwares* clientes de gerenciamento de rede, mesmo que as versões disponíveis nos roteadores ou nos segmentos monitorados sejam diferentes. Para os casos de atualização da MIB, os programas clientes podem mandar mensagens alertando ao administrados que estas novas definições ainda não estão definidas neste protocolo afim de deixar o administrador ciente até qual posição a MIB está correspondendo às necessidades de uso [COM98].

Segundo [COM98], as variáveis pertencentes à MIB são (tabela 3):

TABELA 3: Variáveis pertencentes à MIB.

Variáveis da MIB	**Categoria**	**Descrição**
SysUpTime	Sistema	Tempo desde o último reinicio
IfNumber	Interfaces	Número de interfaces de rede
IfMtu	Interfaces	MTU para uma interface especial
IpDefaultTTL	Ip	Valor que o IP utiliza em campo de tempo de vida
IpInReceives	Ip	Número de datagramas recebidos
IpForwDatagrams	Ip	Número de datagramas encaminhados
IpOutNoRoutes	Ip	Número de falhas do roteamento
IpReasmOKs	Ip	Número de datagramas remontados
IpFragOKs	Ip	Número de datagramas fragmentados
IpRoutingTable	Ip	Tabela de roteamento IP
IcmpInEchos	Icmp	Número de solicitações Echo ICMP recebidas
TcpRtoEchos	Tcp	Tempo mínimo de retransmissão permitido pelo TCP

TABELA 3 (Continuação): Variáveis pertencentes à MIB.

TcpMaxConn	Tcp	Conexões máximas de TCP permitidas
TcpInSegs	Tcp	Número de segmentos recebidos pela TCP
UdpInDatagrams	Udp	Número de datagramas do UDP recebidos
EgpInMsgs	Egp	Número de mensagens EGP recebidas

O mapeamento para a resposta da consulta por parte do monitor acontece pelo *software* agente, que interpreta a variável MIB até a estrutura dos dados [COM98].

3.5.3 ESTRUTURA DAS INFORMAÇÕES DE GERÊNCIA

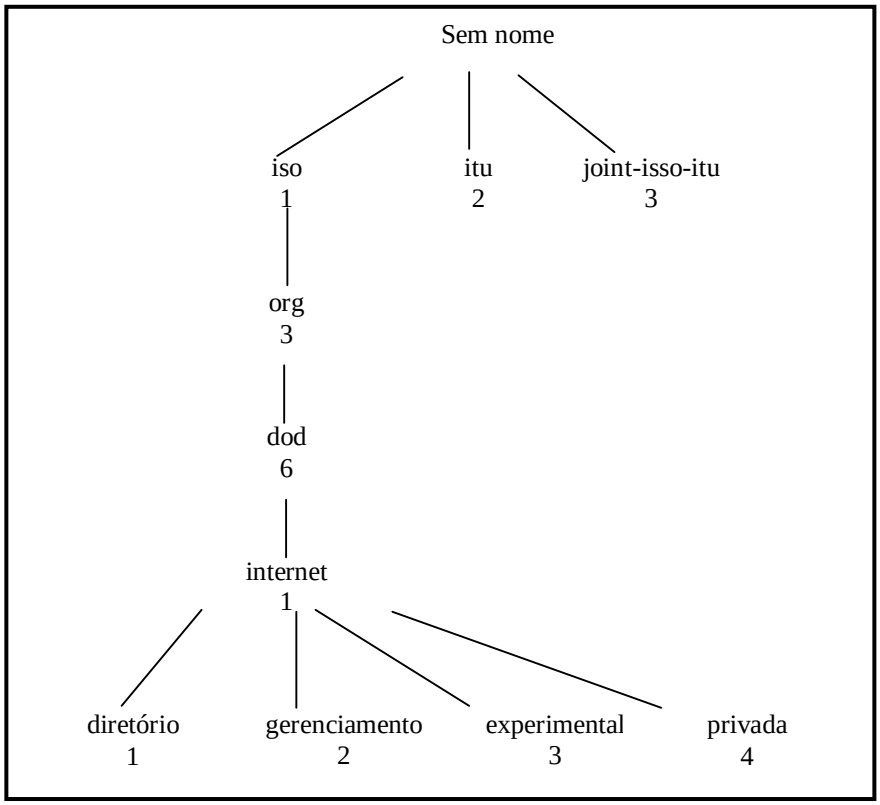
Até agora pode-se observar que a MIB especifica variáveis de gerenciamento de redes e seus significados. Mas a MIB não é o único padrão responsável pela especificação de um conjunto de regras para a definição e identificação de variáveis MIB. Existe um outro padrão que auxilia nesta especificação. Este outro padrão é conhecido como *Structure Management Information* (Informações de Gerenciamento de Estruturas - SMI) [COM98].

O SMI restringe os tipos de variáveis permitidas na MIB e especifica regras para a definição de tipos de variáveis. A consideração mais relevante é que as regras da SMI descrevem como a MIB efetua uma consulta em tabelas de valores [COM98].

Uma MIB pode ser descrita como uma árvore para a representação de objetos definidos. Os diversos níveis da árvore são compostos pelos itens de dados. Identificadores de objetos identificam ou nomeiam unicamente os objetos da MIB na árvore. Estes identificadores de objetos são organizados hierarquicamente com um dígito específico pré-definido pelo protocolo da MIB, associado por diferentes organizações [COM98] [RFC1271].

A árvore MIB pode ser bastante extensa em tamanho devido as ramificações serem tanto de origem experimentais como privadas. As ramificações de uma árvore MIB podem ser definidas por qualquer fornecedor para habilitar a instância de objetos. A raiz da árvore não possui nome definido, mas dispões de três nodos descendentes diretos. Um exemplo da estrutura básica da MIB pode ser vista na fig. 6 a seguir.

FIGURA 6: Exemplo de variável MIB



Mais informações a respeito da MIB e sua estrutura pode ser encontrada em [COM98], [MEI99], [SZT99] e [RFC1271].

3.5.4 DEFINIÇÕES DE VARIÁVEIS MIB SEGUNDO A ASN.1

O padrão SMI define que todas as variáveis MIB devem ser especificadas segundo a *Abstract Syntax Notation.1* (Notação de Sintaxe Abstrata - ASN.1 – leia-se A – S – N "ponto" um). A ASN.1 surgiu de dois focos principais: de pessoas que leram documentos e definiram notações para objetos e através de codificação da mesma informação contida em protocolos de comunicação. Por estes pretextos, a ASN.1 é considerada uma linguagem formal. Em ambos os casos citados anteriormente, as notações devem obrigatoriamente ser seguidas com o intuito de não deixar existir qualquer tipo de ambigüidade para a representação e significado do objeto. A definição dos objetos é feita somente através de números inteiros. Tais definições devem ser seguidas rigorosamente para que as diversas implementações sirvam para quaisquer tipos de computadores, já que atualmente existem diversos tipos e marcas de computadores disponíveis e em uso no mercado de trabalho. Através destas diretivas base, a

notação de objetos segundo a ASN.1 ajuda a simplificar a implementação de protocolos de gerenciamento de redes e possibilitar a interoperabilidade entre dispositivos gerenciáveis [COM98] [SZT99].

Um estudo mais aprofundado sobre a ASN.1 e notação de variáveis MIB poderá ser realizado em [COM98] e [SZT99].

3.5.5 AGENTES

Segundo [BRI94], [TAN94] e [MEI99], os agentes são pequenos aplicativos situados nas estações ou segmentos a serem gerenciados, que "observam" todo o fluxo na segmento da rede e armazena-os para estatísticas e fornecem uma interface para as MIBs e objetos gerenciados instalados no computador. Os programas gerenciados SNMP enviam pedidos para os computadores na rede. O programa agente no computador recebe estes pedidos e processa-os retirando as informações desejadas das MIBs. O agente, então, retorna a informação requisitada para o programa gerenciador SNMP, que foi o responsável pela requisição da informação.

Segundo [BRI94] e [MEI99], o código de um agente é constituído de pelo menos três funções básicas:

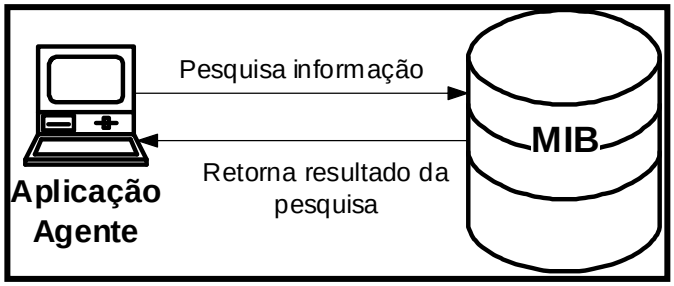
a) um núcleo para implementação e tratamento dos protocolos de gerência;

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.

Através desta informações os gerentes obtém os dados desejados dos segmentos que estão sendo monitorados. A comunicação entre o gerente e o agente acontece através do protocolo SNMP, como mencionado anteriormente. Veja a seguir na fig. 7, o esquema simplificado da relação entre uma aplicação gerente e a MIB para cada entidade a ser gerenciada.

FIGURA 7: Relação aplicação gerente e MIB.



4 MONITORAMENTO REMOTO

O protocolo SNMP visto anteriormente não é o protocolo mais adequado para ambientes de redes corporativas constituídas de várias redes locais conectadas a longa distância. Redes que possuem longas distâncias podem sofrer em desempenho se tiverem seu espaço de banda ocupados por grandes quantidades de informações de gerenciamento, e começam a tornarem-se lentas ao longo de sua extensão. Para solucionar este problema surgiu, então, o *Remote Monitoring* (RMON – Monitoramento Remoto) [MEI99] [ROS99] [SZT99] [RFC1271].

O primeiro documento RMON a ser publicado foi o *Requests for Comments* (RFC - Requisição para comentários especificados pela *Internet Engineering Task Force* - IETF) 1271 em Novembro de 1991, com seu foco principal voltado para a Ethernet (ver item 2.3). A partir de 1993 o *RMON Working Group* adicionou extensões do *Token Ring* (ver item 2.3) junto ao documento RFC 1513. O *RMON Working Group* é um grupo de pesquisa e análise com ênfase no monitoramento remoto [MEI99] [ROS99] [SZT99] [RFC1271].

Os primeiros produtos, utilizando o RMON, começaram a ser desenvolvidos por fornecedores independentes de produtos de monitoração de LANs, como por exemplo, a empresa AXON [ROS99].

Através destas primeiras implementações surgiu a MIB (explicada anteriormente no item 3.5.2) do RMON em dezembro de 1994 que foi especificada com o documento RFC 1757, que trata da MIB do RMON atualizando o RFC 1217, fazendo com que o documento RFC 1217 tornasse-se desnecessário e obsoleto [ROS99].

Através destas MIBs os gerentes de rede podem coletar informações de diversos segmentos remotos para a solução de problemas e gerência de desempenho. As informações de uma MIB do RMON, segundo[MEI99] [ROS99] [SZT99] [RFC1271], devem incluir:

a) Estatísticas do histórico de tráfego, bem como do tráfego atual para os segmentos de rede, para um *host* específico no segmento e entre *hosts*;

b) Um alerta versátil e mecanismos de eventos para ajuste dos *thresholds* e notificar mudanças no comportamento da rede ao gerente;

c) Filtros flexíveis e captura de pacotes que podem ser utilizadas para obter-se um analisador de protocolo e distribuído.

O monitoramento remoto (*Remote Monitoring* – RMON) é visto comumente como uma alternativa de baixo custo para analisadores e monitores de redes tentando, assim, baixar os custos de manutenção da rede [ROS99].

O padrão RMON possibilita ao monitor observar diversas redes ou pontos dela a partir de um local único, centralizado, podendo achar uma solução conveniente para eventuais problemas de maneira mais simples. Mas isso não significa que o RMON pode fornecer todas as soluções para o monitor. Em muitos casos o monitor pode verificar um determinado segmento de rede e retirá-lo da rede remotamente. Já no ponto prático o monitor da rede pode resolver determinados problemas sem necessariamente precisar deslocar-se até o local aonde a rede se encontra, economizando assim tempo e diminuindo o custo de manutenção e/ou reparo [MEI99] [ROS99] [SZT99] [RFC1271].

Através do monitoramento remoto o monitor pode executar diversos testes a partir de um único computador remoto através de dispositivos de monitoramento distribuídos. Desta forma, o monitor pode obter estatísticas do desempenho da rede e melhorá-lo, estabilizá-lo e prever as tendências para a rede em observação. No protocolo RMON ainda fica definido que dispositivos diversos como estações de trabalho, *switches* e hubs, entre outros, ficam atribuídos ao monitor remoto. Cada dispositivo ou objeto RMON deve ajudar na coleta, análise, filtragem de informações de gerenciamento da rede e passar somente eventuais erros ocorridos à estação monitora. Através das informações de gerenciamento provenientes das estações monitoradas, a estação monitora pode tentar solucionar falhas ocorridas na rede [MEI99] [ROS99] [RFC1271].

O protocolo deve, segundo [MEI99], [ROS99], [SZT99] e [RFC1271],visar principalmente:

a) a redução da quantidade de informações trocadas entre o segmento local gerenciado e a estação monitora conectada a uma rede local remota;

b) a possibilidade de gerenciamento contínuo de segmentos de redes locais, mesmo quando não houver comunicação temporária entre o objeto RMON e a estação monitora, devido a uma interrupção temporária;

c) permitir o gerenciamento pró-ativo da rede, diagnosticando e registrando eventos que auxiliem na detecção de mau funcionamento nos diversos segmentos da rede e ajudar na prevenção der falhas que possam vir a interromper sua operação normal;

d) detectar, registrar e informar à estação gerente condições de erro e eventos significativos da rede;

e) enviar informações de gerenciamento para múltiplas estações gerentes, permitindo, em casos de situações críticas de operação da rede gerenciada, que a causa da falha ou mau funcionamento dos segmentos da rede possam ser diagnosticados por mais de uma estação monitora.

Após a análise do RMON ainda faltaria verificar o que realmente caracteriza um gerenciador de desempenho.

Segundo [EMB86], [MEI99], [ROB99], [ROS99], [SZT99] e [RFC1271], os principais passos para a implementação de um gerenciador de desempenho de rede são:

a) Identificação de quais são as necessidades de desempenho;

b) Definição do intervalo de tempo para novas requisições e quais dados deverão ser obtidos;

c) Captura e manutenção das medidas e métricas dos banco de dados;

d) Agrupamento de dados sobre o desempenho para ferramentas de modelagem;

e) Definição dos relatórios: estabelecimento dos valores máximos permitidos (*thresholds*), taxas de alarmes e granularidade;

f) Análise do desempenho, incluindo:
   - Análise do tráfego da rede;
   - Análise das tendências;

- Análise dos protocolos;
- Análise do fluxo de dados (*Troughput*);
- Análise de utilização;
- Recomendações referentes à mudanças e atualizações.

Para melhor utilização de um *software* de gerenciamento de desempenho deve-se coletar dados referentes ao uso e desempenho para diversos segmentos isolados e componentes da rede, assim como, possibilitar uma análise mais profunda dos segmentos que estão sendo monitorados.

4.1 OBJETIVOS DO MONITORAMENTO REMOTO

A especificação RMON é a definição de uma MIB, denominada de RMON-MIB. O principal objetivo desta MIB é a definição de padrões de monitoramento e interfaces para a comunicação entre agentes/gerentes SNMP.

O monitoramento remoto serve para auxiliar o administrador da rede, entre outras funções, como já mencionado anteriormente. A seguir estão demonstrados cinco principais objetivos do monitormanento remoto, de acordo com [MEI99], [SZT99], [RFC1271] e [RFC1513]:

a) Operação *Offline*

Às vezes existem condições onde uma estação gerenciadora não está em constante contato com seus dispositivos monitorados, por causa de uma falha na comunicação na rede entre a estação gerenciadora e o agente.

Por esta razão, esta MIB permite à um agente ser configurado para desempenhar diagnósticos e coletar estatísticas continuamente, mesmo quando a comunicação com a estação gerenciadora não for possível ou eficiente. O agente pode, desta forma, tentar informar a estação gerenciadora quando uma condição anormal ocorrer. Por conseguinte, mesmo em circunstâncias onde a comunicação entre a estação gerenciadora e o agente não for contínua, erros, desempenho e informações a respeito de configurações podem ser constantemente acumuladas e comunicadas à estação gerenciadora convenientemente e eficientemente;

b) Monitoramento Pró-Ativo

Dado os recursos disponíveis no agente é útil executar diagnósticos constantemente e criar um registro (*log*) do desempenho da rede. O monitor sempre deverá estar disponível após qualquer tipo de falha. Ele pode informar à estação gerenciadora sobre a falha e pode também armazenar informações estatísticas em forma de histórico sobre os erros. Este histórico poderá ser chamado pela estação gerenciadora na tentativa de desempenhar diagnósticos futuros para a causa do problema.

Além deste dois principais objetivos seguem ainda conforme [MEI99], [SZT99], [RFC1271] e [RFC1513]:

c) Detecção do Problema e Relatórios

O agente pode ser configurado para reconhecer condições anormais, erros mais comuns e verificar constantemente por eles. Quando uma destas condições ocorrer o evento pode ser registrado e a estação gerenciadora poderá ser informada de várias formas distintas;

d) Valor Adicional ao Dado

Pelo fato de um dispositivo de monitoramento remoto representar um recurso dedicado exclusivamente às funções de gerenciamento de rede por causa que ele se situa diretamente no segmento da rede a ser monitorado, o dispositivo de gerenciamento remoto de rede tem a possibilidade de acionar valores significativos aos dados que ele obtiver. Por instância, realçando estes *hosts* na rede que geram o maior tráfego de erros, os agentes podem fornecer precisamente à estação gerenciadora as informações que ela precisa para conseguir resolver um conjunto de problemas;

e) Múltiplos Gerentes

Uma organização pode ter múltiplas estações gerenciadoras para diferentes unidades da organização para diferentes funções (por exemplo, as áreas de engenharia e produção) e na tentativa de prover a recuperação de desastres. Por causa que ambientes com várias estações gerenciadoras podem existir, o dispositivo gerenciador tem que conseguir lidar concorrentemente com mais de uma estação gerenciadora.

4.2 CONSIDERAÇÕES SOBRE O RMON

Os padrões do RMON definem um sistema de coleta de dados de duas entidades conforme especificados nos documentos RFC 1757 e RFC 1513 [RFC1271] [RFC1513].

No RF 1757 é especificado um monitor ou também denominado de agente de verificação que situa-se em cada segmento da rede monitorada. Como parte complementar ao RFC 1757, o documento RFC 1513 especifica a aplicação de gerenciamento que executa em uma estação central como ponto central do sistema de monitoramento da rede [SZT99] [RFC1271].

Para obter-se dados referente as estações que estão sendo monitoradas, o aplicativo gerenciador, baseado no RMON, recolhe informações a partir de agentes (ver item 4.11 para mais detalhes sobre agentes) situados nas estações monitoradas, e envia estas informações para a estação gerenciadora através de requisição destes dados por parte da própria estação gerenciadora ou através de eventos específicos [SZT99] [RFC1271].

O número de agentes possíveis para o monitoramento das estações pode ser igual ao número de estações existentes na rede. Não necessariamente o número de agentes precisa ser igual ao número de estações na rede. Em casos em que a rede seja muito grande pode-se obter dados apenas dos principais pontos a serem monitorados da rede a partir de agentes situados nestes pontos que foram considerados de maior relevância para a análise [MEI99] [SZT99] [RFC1271].

Estes padrões descritos do RMON são definidos como MIBs e uma estrutura de dados SNMP contendo dados estatísticos acessíveis através de comandos SNMP. Nas MIBs as diversas tabelas de estatística são denominadas de grupos. A coleta dos dados é obtida a partir destas tabelas. Além disso, as tabelas podem ser manipuladas por agentes independentemente do volume de dados transmitidos através rede e de processos pendentes ou não terminados [MEI99] [SZT99].

Existem diversas formas de visualização dos dados armazenados. Para exemplificar como estas formas de visualização podem ser diferentes, seguem alguns exemplos. Os dados podem ser representados de forma gráfica, por exemplo, através de linhas coloridas que deslocam-se entre uma estação de origem e outra de destino. Outra forma de visualização é a amostragem do resumo em diversos segmentos da rede [MEI99] [SZT99] [RFC1271].

4.3 GRUPOS DE OBJETOS ESPECIFICADOS NA RMON-MIB 1271

Segundo [MEI99], [SZT99] e [RFC1271], os principais grupos de objetos do padrão RMON são:

a) *Statistics* (Estatísticas): contém informações para quantificar a eficácia operacional da rede. Neste grupo são armazenadas as estatísticas fundamentais para cada subrede que está sendo monitorada. Atualmente, este grupo contém objetos definidos para interfaces *Ethernet*. Extensões futuras podem incluir outros tipos de redes. Este grupo fornece estatísticas de *Cyclical Redudancy Check* (CRC), colisões e tamanhos de pacotes diferentes;

b) *History* (Histórico): provê dados estatísticos de grupos estatísticos para a análise dos diversos segmentos existentes na rede;

c) *Host Table* (Tabela de *Hosts*): coleta informações a respeito de cada sistema servidor no segmento analisado. O monitor adquire novos conhecimentos sobre novos *hosts* da rede observando a origem e o destino na camada de acesso de controle (*Media Access Control* - MAC) dos pacotes. Para cada *Host* conhecido é mantido um conjunto de estatísticas;

d) *HostTopN*: este grupo reduz a sobrecarga de rede ordenando as *Host Tables* de acordo com a atividade e apresentando somente o *host* que tiver o maior índice de atividade em cada segmento existente. Este grupo é utilizado para manter estatísticas sobre o conjunto de *hosts* existentes em uma subrede que incia em uma lista baseada em algum parâmetro. Mas o que isso significa exatamente? Imagine que, por exemplo, uma lista pode ser mantida para 5 *hosts* que transmitem a maior parte dos dados durante um dia. As estatísticas geradas são provenientes dos dados do grupo *Host*. O conjunto de estatísticas para um objeto do grupo *Host* em uma subrede, coletados durante uma amostragem, são referenciados como um relatório. Assim, cada um destes relatórios mostra os resultados para apenas uma variável, e esta variável representa o total de trocas que ocorreram em um objeto do grupo *Host* em intervalo específico;

e) *Traffic Matrix* (Matriz de fluxo): emite relatórios de erros ocorridos entre pares de *hosts* e relatórios referente ao fluxo de dados. Este grupo também é utilizado para gravar informações sobre o tráfego existente entre pares de *hosts* em uma subrede. A informação obtida é armazenada em forma de uma matriz. Através deste método é mais simples de recuperar informações específicas, tais como, por exemplo, quais dispositivos usufruem mais de recursos de um servidor;

f) *Alarms* (Alarmes): relata mudanças ocorridas nas características da rede. Toda informação é baseada no valores máximos permitidos (*threshold*) para todas as variáveis MIB de interesse. Este grupo é utilizado para a definição de um conjunto de entradas para a monitoração do desempenho da rede. Caso um limite pré-estabelecido for alcançado, é gerado um alarme e enviado para a estação central de monitoração. Existem diversos motivos para que possa ocorrer o disparo de um alarme. Se, por exemplo, houver mais que 200 erros *Cyclical Redudancy Check* (CRC) em um período de tempo de amostragem específico, um alarme pode ser gerado;

g) *Events* (Eventos): registra os eventos ocorridos. As informações são baseadas em *thresholds* que servem para iniciar a captura de dados ou a contagem de inicialização de registros para o possível isolamento de pontos específicos da rede. Neste grupo encontram-se as definições para os eventos que possam ocorrer. Quando uma condição localizada na MIB é encontrada um evento é disparado, e um evento pode disparar uma ação definida em outra parte da MIB. Um evento também pode levar uma informação a ser registrada (logada) neste grupo;

h) *Filters* (Filtros): são pacotes que correspondem a definições que permitem aos gerentes de redes isolar certos tipos de tráfico na rede. O filtro de dados permite ao monitor gravar pacotes observados como parte de um modelo de *bit* desta porção de pacotes juntos. O filtro de s*tatus* permite ao monitor gravar pacotes observados em parte desses *status* (válido, ou algum tipo de erro, entre outros). Ambos os filtros podem ser combinados através de operações lógicas AND/OR (e/ou) para aplicar um teste nos pacotes recebidos. Os vários pacotes que são recebidos e seguem um após o outro são chamados de canal. Um canal pode ser configurado para gerar um evento quando um pacote passa por um canal e o canal está

habilitado. Os pacotes que passam pelo canal podem ser capturados se o mecanismo for definido no grupo *Packet Capture*;

i) *Packet Capture* (Captura de pacotes): copia os pacotes contidos na relação de correspondências dos filtros.

4.4 DESCRIÇÃO DOS OBJETOS DAS TABELAS CLASSIFICADAS POR GRUPOS SEGUNDO O RFC 1271

Os objetos e suas definições foram baseadas no documento RFC 1271, que especifica a MIB do RMON. A seguir seguem as definições dos objetos classificados nas tabelas em seus respectivos grupos:

a) O Grupo *Statistics* (Estatística)

Ele consiste de uma tabela com uma entrada para cada interface monitorada. As estatísticas são guardadas na forma de contadores e quando ocorrer uma nova entrada e esta for válida, esta entrada é criada com o valor inicial em 0.

Neste grupo as funções da tabela de controle e de dados são combinadas. As variáveis desta tabela que podem ser alteradas (*read-write*) são *etherStatsDataSource*, *etherStatsOwner*, *etherStatsStatus*. Dessa maneira, uma estação gerente pode requisitar que o monitor junte estatísticas de uma ou mais dessas interfaces;

b) O Grupo *History* (Histórico)

Este grupo é composto de duas tabelas. A primeira, *historyControlTable*, especifica a interface e os detalhes da função. Já a segunda tabela, *etherHistoryTable*, armazena os dados obtidos para poder gerar o histórico.

As variáveis da tabela *historyControlTable* pode ser vista na tabela 4 a seguir.

TABELA 4: Definição das variáveis da tabela *historyControlTable*.

Campo da Tabela	Descrição
HistoryControlIndex	Um número inteiro para a identificação de um registro. Ele também pode ser utilizado para a identificação de registros correspondentes na tabela de dados.
HistoryControlDataSource	Identifica a interface e de qual subrede são os dados.
HistoryControlBucketsReq	Especifica o número de intervalos de amostragem requisitados até que os dados sejam armazenados na parte da tabela de dados associada à eles.
HistoryControlInterval	Define o intervalo em segundos para que os dados sejam amostrados para cada *bucket*. O valor pode ter uma variação de 1 à 3600, o valor padrão é de 1800.

A tabela *etherHistoryTable* possui as seguintes variáveis, conforme a tabela 5.

TABELA 5: Definição das variáveis da tabela *etherHistoryTable*.

Campo da Tabela	Descrição
EtherHistoryIndex	Define o índice relacionada à entrada.
EtherHistorySampleIndex	Identifica o índice que mostra o que essa entrada representa, entre todas as outras amostragens existentes, associadas com o mesmo registro da tabela *historyControlTable*.
EtherHistoryIntervalStart	Armazena o valor de *sysUpTime* (proveniente do grupo *system* da MIB II) do início do intervalo em que a amostra foi avaliada.

As duas tabelas se relacionam através de cada registro da tabela *historyControlTable* possuírem um único valor de *historyControlIndex*. Não é possível que dois registros tenham a mesma combinação de valores de *historyControlDataSource* e *historyControlInterval*. Isto significa que para uma subrede, mais de um processo de amostragem pode ser feito, mas para que isso possa ser realizado cada variável terá um período de amostragem diferente;

Nota: Quando utiliza-se os valores padrão, a aplicação gerente obterá uma amostra a cada 1800 segundos, que totaliza 30 minutos. Cada amostra é armazenada em um registro de *etherHistoryTable* e só os 50 registros mais recentes são retidos.

c) O Grupo *HostTable*

Este grupo é composto de três tabelas. A primeira tabela é de Controle e as outras duas são de Dados. A tabela de controle, denominada de *hostControlTable*, possui a seguinte estrutura, conforme a tabela 6. As outras duas tabelas serão mostradas na seqüência desta tabela.

TABELA 6: Definição das variáveis da tabela *hostControlTable*.

Campo da Tabela	Descrição
HostControlIndex	Especifica um inteiro que identifica um registro. Cada registro na tabela se referencia a uma única subrede do monitor. O mesmo valor é utilizado para identificar registros correspondentes na tabela *hostTable* e na tabela *hostTimeTable*.
HostControlDataSource	Identifica a interface e a subrede que deu origem aos dados.
HostControlTableSize	Indica o número de registros na tabela *hostTable* que são associadas a este registro.
HostControlLastDeleteTime	Define o valor de *sysUpTime* correspondendo à última vez que uma entrada foi apagada da porção da *hostTable* associada com este registro. Caso não tenha sido eliminado nenhum registro o valor será o valor padrão 0.
HostControlEntry	Define uma entrada de controle para cada *host* que for adicionado em uma nova pesquisa.
HostControlOwner	Controla quem é proprietário do *host* em questão.
HostControlStatus	Verifica qual o estado do *host*. Especifica se um determinado *host* está ou não ativo.

As tabelas *hostControlTable* e *hostTable* estão diretamente relacionadas.

Para cada subrede especificada através de um registro na *hostControlTable*, a *hostTable* contém um registro para cada endereço na camada de controle de acesso (MAC) revelada na subrede.

Cada registro da *hostTable* possui estatísticas de dados sobre o *host* correspondente. A tabela 7 demonstra a estrutura da tabela *hostTable*:

TABELA 7: Definição das variáveis da tabela *hostTable*.

Campo da Tabela	Descrição
HostAddress	Indica o endereço MAC do host.
HostCreationOrder	Indica um índice que define a seqüência relativa de criação dos hosts capturados por uma entrada *hostControlEntry* particular.
HostIndex	Especifica o conjunto de estatísticas pertencentes a entrada.
HostEntry	Define uma entrada de controle para cada *host* que for adicionado em uma nova pesquisa.
HostInPkts	O número de pacotes transmitidos com sucesso para o endereço do *Host* atual desde que ele foi adicionado ao *hostTable*.
HostOutPkts	O número de pacotes enviados com sucesso ou não a partir do endereço do *Host* atual desde que ele foi adicionado ao *hostTable*.
HostInOctets	O número de octetos transmitidos para este endereço desde que foi adicionado à *hostTable* (exceto os *framing bits*, mas incluindo os octetos FCS), com exceção para os octetos em pacotes que não foram transmitidos com sucesso.
HostOutOctets	O número de octetos transmitidos para este endereço desde que foi adicionado à *hostTable* (exceto os *framing bits*, mas incluindo os octetos FCS), incluindo os octetos em pacotes que não foram transmitidos com sucesso.
HostOutErrors	O número de pacotes que foram transmitidos por este endereço, inclusive os pacotes que foram transmitido com erros, desde que o *host* foi adicionado à *hostTable*.
HostOutBroadcastPkts	O número de endereços de pacotes com transmissão bem sucedidas por este endereço que foram direcionadas para o endereço *broadcast* desde que este *host* foi adicionada à *hostTable*.
HostOutMulticastPkts	O número de pacotes com transmissão bem sucedidas transmitidos por este endereço que foram direcionados para um endereço *multicast* desde que este *host* foi incluído na *hostTable*. Não inclui pacotes direcionados ao endereço *broadcast*.

A terceira tabela, *hostTimeTable* auxilia a coleta de dados para estatísticas de cada *host*. A tabela 8 demonstra as variáveis desta tabela a seguir.

TABELA 8: Definição das variáveis da tabela *hostTimeTable*.

Campo da Tabela	Descrição
HostTimeAddress	Especifica o endereço do *host*.
HostTimeCreationOrder	Indica um índice que define a seqüência relativa de criação dos *hosts* capturados.
HostTimeIndex	Indica o índice de uma entrada de controle para cada *host* que for adicionado em uma nova pesquisa.
HostTimeEntry	Define uma entrada de controle para cada *host* que for adicionado em uma nova pesquisa.
HostTimeInPkts	O número de pacotes transmitidos com sucesso para o endereço do *Host* atual desde que ele foi adicionado ao *hostTimeTable.*
HostTimeOutPkts	O número de pacotes enviados com sucesso ou não a partir do endereço do *Host* atual desde que ele foi adicionado ao *hostTimeTable.*
HostTimeInOctets	O número de octetos transmitidos para este endereço desde que foi adicionado à *hostTimeTable* (exceto os *framing* bits, mas incluindo os octetos FCS), com exceção para os octetos em pacotes que não foram transmitindo com sucesso.
HostTimeOutOctets	O número de octetos transmitidos para este endereço desde que foi adicionado à *hostTimeTable* (exceto os *framing* bits, mas incluindo os octetos FCS), incluindo os octetos em pacotes que não foram transmitindo com sucesso.
HostTimeOutErrors	O número de pacotes que foram transmitidos por este endereço, inclusive os pacotes que foram transmitido com erros, desde que o *host* foi adicionado à *hostTimeTable*.
HostTimeOutBroadcastPkts	O número de endereços de pacotes com transmissão bem sucedidas por este endereço que foram direcionadas para o endereço *broadcast* desde que este *host* foi adicionada à *hostTable*.
HostTimeOutMulticastPkts	O número de pacotes com transmissão bem sucedidas transmitidos por este endereço que foram direcionados para um endereço *multicast* desde que este *host* foi incluído na *hostTable*. Não inclui pacotes direcionados ao endereço *broadcast.*

d) O Grupo *HostTopN*

Neste grupo são definidas uma tabela de controle e uma tabela de dados. A tabela de controle é denominada de *hostTopNControlTable* e a tabela de dados de *hostTopNTable* Confira na tabela 9 as variáveis na tabela de controle *hostTopNControlTable*.

TABELA 9: Definição das variáveis da tabela *hostTopNControlTable*.

Campo da Tabela	Descrição
HostTopNControlIndex	Identificador de registros. Cada registro contido na tabela define um relatório preparado para uma determinada interface.
HostTopNHostIndex	Este valor é resultante da junção dos valores das variáveis *hostControlIndex* e *hostIndex*. Especifica uma subrede.
HostTopNRateBase	Especifica uma entre sete variáveis de hostTable. Essas variáveis são todas do tipo inteiro. Essas variáveis podem ser: HostTopNInPkts(1),HostTopNOutPkts(2), HostTopNInOctets(3), HostTopNOutOctets(4), HostTopNOutErrors(5), HostTopNOutBroadcastPkts(6) e HostTopNOutMulticastPkts(7).
HostTopNTimeRemaining	Especifica o número de Segundos para o intervalo de amostragem para o relatório que está sendo coletado.
HostTopNDuration	Define o intervalo de amostragem, em segundos, para o relatório.
HostTopNRequestedSize	Define o número máximo de *hosts* requisitados pela tabela *TopN*.
HostTopNGrantedSize	Define o número máximo de hosts na tabela *TopN*.
HostTopNStartTime	Especifica o valor da variável *sysUpTime* quando o relatório *TopN* foi apresentado pela última vez.

Na tabela 10, a seguir, a definição das variáveis para a tabela *hostTopNTable*.

TABELA 10: Definição das variáveis da tabela *hostTopNTable*.

Campo da Tabela	Descrição
HostTopNReport	Contém o relatório onde esta entrada faz parte. O relatório identificado por este índice é o mesmo relatório identificado pelo mesmo valor do índice de *hostTopNControlIndex*.
HostTopNIndex	Um índice que identifica um registro entre todos registros associados com este relatório.
HostTopNAddress	O endereço MAC deste *host*.
HostTopNRate	A quantidade de trocas efetuadas na variável selecionada durante um intervalo de amostragem específico.

f) O grupo *Traffic Matrix* (Matriz de Fluxo)

Este grupo é composto de três tabelas. A tabela de controle, denominada de *matrixControlTable*, e duas tabelas de dados, denominadas de *matrixSDTable* e *matrixDSTable*.

Veja os campos da tabela de controle *matrixControlTable* na tabela 11.

TABELA 11: Definição das variáveis da tabela *matrixControlTable*.

Campo da Tabela	Descrição
MatrixControlIndex	Identifica um registro na tabela e define uma função que detecta comunicações em uma interface particular, bem como, estatísticas locais sobre estas comunicações em duas tabelas de dados.
MatrixControlDataSource	Identifica a interface e a subrede que deu origem aos dados.
MatrixControlTableSize	Define o número de registros na tabela *matrixSDTable* que estão associadas com o registro. Define também o número de registros na tabela *matrixDSTable*.
MatrixControlLastDeleteTime	Define o valor de *sysUpTime* correspondendo à última vez que uma entrada foi apagada. Caso não tenha sido eliminado nenhum registro o valor será o valor padrão 0.

Na tabela 12 é mostrada a tabela *matrixSdTable* que armazena estatísticas sobre o tráfego entre um *host* de origem e seus destinos.

TABELA 12: Definição das variáveis da tabela *MatrixSdTable*.

Campo da Tabela	Descrição
MatrixSdSourceAddress	Endereço de origem da MAC.
MatrixSdDestAddress	Endereço de destino da MAC.
MatrixSdIndex	É um conjunto de estatísticas coletadas. O conjunto identificado por um valor é o mesmo conjunto identificado pelo mesmo valor de *matrixCounterIndex*.

A tabela *matrixSDTable* é indexada primeiro por *matrixSDIndex*, depois pelo endereço de origem e depois pelo endereço de destino. A tabela *matrixDsTable* é indexada primeiro por *matrixDsIndex*, depois pelo destino e pela origem.

Toda vez que o monitor detectar uma nova comunicação entre um dos *hosts* listados na tabela *matrixControlTable*, ele gera dois novos registros nas duas tabelas de dados. Caso o limite especificado na variável *matrixControlTableSize* for alcançado, o monitor apaga os registros mais recentes como for necessário;

g) O Grupo *Alarms* (Alarmes)

O grupo *Alarms* é caracterizado pela tabela *alarmTable*, armazenando cada entrada em uma variável a ser monitorada, na tabela, conforme mostrado na tabela 13 a seguir.

TABELA 13: Definição das variáveis da tabela *alarmTable*.

Campo da Tabela	Descrição
AlarmIndex	Número inteiro que identifica um registro na tabela.
AlarmInterval	Intervalo em segundos para amostragem dos dados.
AlarmVariable	Identificador para uma variável na MIB local.
AlarmSampleType	Método de cálculo do valor a ser comparado. Se o valor for (1) absoluto, então o valor da variável selecionada será comparado com as entradas. Se o valor for (2) delta, então o valor da variável selecionada do último exemplo é subtraida do valor atual e a diferença comparada com as entradas.
AlarmValue	Valor de estatísticas durante o último período de amostragem.
AlarmStartupAlarm	Tem o valor (1) risingAlarm, (2) fallingAlarm ou (3) risingOrFallingAlarm.

TABELA 13 (Continuação): Definição das variáveis da tabela *alarmTable*.

AlarmRisingThreshold	Um *threshold* para uso estatístico de amostras. Quando uma amostra for maior ou igual a este *threshold*, e o valor da amostra no último intervalo especificado foi menor que este *threshold*, um evento único será gerado. Um evento único também será gerado, se a primeira amostra depois desta entrada tornar-se válida, e for maior ou igual a este *threshold* e os objetos associados *alarmStartupAlarm* for igual a *risingAlarm*(1) ou *risingOrFallingAlarm*(3). Depois que um evento crescente for gerado, não será gerado nenhum outro evento até que o valor da amostra seja inferior a este *threshold* e alcance o valor de *alarmFallingThreshold*. Este objeto não poderá ser modificado se o valor do objeto *alarmStatus* for igual a *valid*(1).
AlarmFallingThreshold	Um *threshold* para uso estatístico de amostras. Quando uma amostra for maior ou igual a este *threshold*, e o valor da amostra no último intervalo especificado foi menor que este *threshold*, um evento único será gerado. Um evento único também será gerado, se a primeira amostra depois desta entrada tornar-se válida, e for maior ou igual a este *threshold* e os objetos associados *alarmStartupAlarm* for igual a falling*Alarm*(1) ou *risingOrFallingAlarm*(3). Depois que um evento crescente for gerado, não será gerado nenhum outro evento até que o valor da amostra seja inferior a este *threshold* e alcance o valor de *alarmRisingThreshold*. Este objeto não poderá ser modificado se o valor do objeto *alarmStatus* for igual a *valid*(1).
AlarmRisingEventIndex	O índice de *EventEntry* que é utilizado quando cruza-se um *threshold* crescente. O *eventEntry* identificado por um valor específico deste índice é o mesmo que foi identificado pelo objeto *eventIndex.* Se não houver uma entrada correspondente na tabela *eventTable*, então não existem associações. Em específico se este valor for zero, nenhum evento associado será gerado, já que o valor 0 não é um índice de evento válido. Este objeto não poderá ser modificado caso o objeto associado *alarmStatus* for igual a valid(1).
AlarmFallingEventIndex	O índice de *EventEntry* que é utilizado quando cruza-se um *threshold* crescente. O *eventEntry* identificado por um valor específico deste índice é o mesmo que foi identificado pelo objeto *eventIndex.* Se não houver uma entrada correspondente na tabela *eventTable*, então não existem associações. Em específico se este valor for zero, nenhum evento associado será gerado, já que o valor 0 não é um índice de evento válido. Este objeto não poderá ser modificado caso o objeto associado *alarmStatus* for igual a valid(1).

h) O Gupo *Events* (Eventos)

Este grupo é constituído de uma tabela de Controle e uma de Dados. A tabela de Controle, denominada de *eventTable*, possui as definições para os eventos. Cada registro dessa tabela contém parâmetros que descrevem um determinado evento para ser gerado quando algumas condições específicas são alcançadas. As variáveis contidas na tabela *eventTable* são mostradas na tabela 14 a seguir.

TABELA 14: Definição das variáveis da tabela *eventTable*.

Campo da Tabela	Descrição
EventIndex	Identificador de registros na tabela.
EventDescription	Descrição textual do evento.
EventType	Pode ter um dos valores *none* (1), *log* (2), *SNMP-trap* (3) ou *log-and-trap* (4). No caso do *log*, é feita uma entrada na tabela *logTable* para cada evento ocorrido. No caso do *SNMP-trap*, uma mensagem SNMP é enviada para uma ou mais estações gerente para cada evento ocorrido.
EventCommunity	Se uma mensagem SNMP é enviada, o objeto especifica a comunidade de estações gerente para recebe-la.
EventLastTimeSent	O valor de *sysUpTime* no momento que este entrada gerou pela última vez um evento.

Caso um evento tenha sido registrado, serão geradas entradas na tabela *logTable* associada, que inclui as variáveis mostradas na tabela 15 a seguir.

TABELA 15: Definição das variáveis da tabela *logTable*.

Campo da Tabela	Descrição
LogEventIndex	Identifica qual evento gerou uma entrada específica no registro.
LogIndex	Um índice que identifica um registro entre todas outras entradas associadas com o mesmo tipo de evento.
LogTime	O valor de sysUpTime quando esta entrada foi gerada.
LogDescription	Descrição do evento que ativou a entrada no registro.

As condições para o acontecimento dos eventos são definidas em outros grupos do RMON.

i) O Grupo *Filter* (Filtros)

Este grupo é constituído de duas tabelas denominadas de *channelTable*, que define um canal e junto com este canal estão um ou mais registros na segunda tabela, *filterTable*, que define os filtros associados à tabela. Na tabela 16 a seguir está definida a estrutura da tabela *channelTable.*

TABELA 16: Definição das variáveis da tabela *channelTable*.

Campo da Tabela	Descrição
ChannelIndex	Define um número inteiro que identifica um canal.
ChannelIfIndex	Identifica a interface monitor e a subrede, da qual os filtros são utilizados para aprovar dados neste canal.
ChannelAcceptType	Controla a ação dos filtros associados. Os valores podem ser: *acceptMatched*(1) e *acceptFailed*(2).
ChannelDataControl	Quando o valor for 1, então os dados, *status* e eventos trafegam por este canal. Se o valor for 0, eles não trafegam.
ChannelTurnOnEventIndex	Identifica um evento pré-definido para trocar o valor da variável *channelDataControl* associada do estado OFF para ON quando um evento é gerado. O valor desta variável identifica um objeto indexado por *eventIndex* no grupo *Event*. Não existem associações, se não existirem eventos.
ChannelTurnOffEventIndex	Identifica um evento pré-definido para trocar o valor da variável *channelDataControl* associada do estado ON para OFF quando um evento é gerado. O valor desta variável identifica um objeto indexado por *eventIndex* no grupo *Event*. Não existem associações, se não existirem eventos.
ChannelEventIndex	Identifica um objeto indexado por *eventIndex* no grupo *Event* e identifica também a geração de um evento pré-definido Quando o *channelDataControl* associado está ligado e um pacote está unido.
ChannelEventStatus	Representa o evento *status* desse canal. Se o canal estiver configurado para gerar eventos, pacotes são juntados. Assim, o valor pode ser visto conforme a seguir. Quando o valor for *eventReady*(1), um simples evento será gerado para um pacote unido, após mudando este objeto para (2). No estado *eventFired*(2) não são gerados eventos. Este permite a estação gerente responder à notificação de um evento e então re-habilitar o objeto. Quando o valor for *eventAlwaysReady*(3), todos pacotes unidos geram um evento.

TABELA 16 (Continuação): Definição das variáveis da tabela *channelTable*.

ChannelMatches	Especifica um contador que registra o número de pacotes unidos e é atualizado quando o *channelDataControl* for colocado para OFF.
ChannelDescription	Descrição em forma de texto do canal.

A estrutura da tabela *filterTable* é mostrada na tabela 17.

TABELA 17: Definição das variáveis da tabela *filterTable*.

Campo da Tabela	Descrição
FilterIndex	Identifica um filtro de dados e um de *status* que serão aplicados nos pacotes recebidos pela interface.
FilterChannelIndex	Identifica índice do canal.
FilterPktDataOffSet	Deslocamento a partir do início de cada pacote onde uma mistura de dados foi tentada.
FilterPktData	Os dados que devem ser unidos com o pacote de entrada (Modelo de bits a ser testado).
FilterPktDataMask	Máscara que será aplicada ao processo (Bits relevantes a o teste).
FilterPktDataNotMask	Máscara inversa.
FilterPktStatusMask	Máscara que será aplicada no *status* do processo.
FilterPktStatusNotMask	Máscara inversa.
FilterPktStatus	O *status* que será unido com o pacote de entrada.

j) O Grupo *Packet Capture* (Captura de Pacotes)

Este grupo é constituído de duas tabelas. A primeira, *bufferControlTable,* especifica os detalhes da função do *buffer.* A segunda, *captureBufferTable* faz o *buffer* dos dados.

Cada registro contido na tabela *bufferControlTable* define um *buffer* que é utilizado para capturar e armazenar pacotes de um canal. Confira as variáveis para a tabela *bufferControlTable* na tabela 18 abaixo.

TABELA 18: Definição das variáveis da tabela *bufferControlTable*.

Campo da Tabela	Descrição
BufferControlIndex	Identifica um registro na tabela. O mesmo valor é utilizado para identificar registros correspondentes na tabela *captureBufferTable*.
BufferControlChannelIndex	Identifica qual canal originou os pacotes para este registro.
BufferControlFullStatus	Pode ter dois valores. Caso o valor for *spaceAvaliable*(1), o *buffer* possui espaço para aceitar novos pacotes. Se o valor for *full* (2), significa que para aceitar novos pacotes, o *buffer* depende do valor de *bufferControlFullAction*.
BufferControlFullAcion	Pode ter dois valores. Caso o valor for *lockWhenFull*(1), o *buffer* não poderá mais aceitar pacotes depois que estiver cheio. Caso o valor for *wrapWhenFull*(2), o *buffer* comporta-se como uma espécie de *buffer* circular, isto é, depois de cheio apaga os pacotes mais antigos para liberar espaços para pacotes mais recentes.
BufferControlCapSliceSize	Define o número máximo de octetos que cada pacote pode possuir, iniciando com o começo de cada pacote, que será armazenado no *buffer* capturado. Caso valor for 0, o *buffer* armazenará a quantidade de octetos que for possível. O valor padrão estabelecido é de 100 octetos.
BufferControlDownLoadSliceSize	Define o número máximo de cada pacote no buffer que será retornado em uma entrada SNMP.
BufferControlDownLoadOffset	Define o deslocamento para cada pacote do primeiro octeto no *buffer* que será retornado através de uma entrada SNMP.
BufferControlMaxOctetsReq	Especifica o tamanho para o *buffer* requisitado em octetos. O valor da variável de -1 faz com que o buffer seja o maior o possível.
BufferControlMaxOctetsGra	Especifica o tamanho máximo do *buffer* concedido em octetos. Este é o número de octetos que poderá ser armazenado.
BufferControlCapturedPkts	Mostra o número de pacotes que estão capturados no *buffer* até o momento atual.
BufferControlTurnOnTime	Armazena o valor de *sysUpTime* quando o *buffer* foi inicializado pela primeira vez.

A tabela de dados, denominada de *captureBufferTable*, armazena um registro para cada pacote capturado. A estrutura desta tabela pode ser vista na tabela 19 a seguir.

TABELA 19: Definição das variáveis da tabela *bufferControlTable*.

Campo da Tabela	Descrição
CaptureBufferControlIndex	Indica o registro do *buffer* com o qual um pacote está diretamente associado.
CaptureBufferIndex	Especifica um índice que identifica um pacote entre todos outros pacotes associados com o mesmo *buffer*.
CaptureBufferPacketID	Demonstra um índice que descreve a ordem de pacotes recebidos em subrede (interface).
CaptureBufferPacketData	Indica o pacote de dados armazenado atualmente.
CaptureBufferPacketLength	Mostra o tamanho do pacote recebido. Não necessariamente significa que o capote tenha que ter sido recebido por inteiro, mas pode também ser apenas uma parte do pacote, que está atualmente armazenada.
CaptureBufferPacketTime	Especifica o tempo em milisegundos que já ocorreram desde que o *buffer* tornou-se ativo até que o pacote foi capturado.
CaptureBufferPacketStatus	Indica o *status* de erro para o pacote.

5 DESENVOLVIMENTO DO PROTÓTIPO

O protótipo desenvolvido neste Trabalho de Conclusão de Curso visa uma análise de dados (amostras) obtidas a partir de agentes (clientes) sobre segmentos monitorados da rede, visualizados a partir de monitores, no âmbito do gerenciamento de redes, sendo que a comunicação entre o cliente e monitor acontece através do protocolo SNMP para troca de dados e fornecimento de informações, no ambiente Windows NT.

Neste capítulo são apresentadas as técnicas e ferramentas utilizadas durante o desenvolvimento do protótipo, bem como, para especificação e implementação do protótipo proposto.

Para a implementação do protótipo proposto foi utilizado o grupo de componentes *SNMP Tool for Delphi 4.0* da empresa *Dart Communications*, que possibilita o uso do protocolo SNMP através de seus componentes. Ainda para a implementação foi utilizado o ambiente de programação *Borland Delphi 4.0* da empresa *Inprise Corporation*, que baseia-se na linguagem de programação *Object Oriented Pascal* (OOP). Para a visualização do conteúdo das MIBs foi utilizado o programa *MIB Browser* da empresa *MG-SOFT Corporation*.

5.1 ESPECIFICAÇÃO DO PROTÓTIPO

Para a especificação do protótipo foi considerada a programação baseada em eventos em ambientes gráficos livres.

Na representação do funcionamento dos procedimentos e funções do protótipo foram consideradas as linguagens baseadas em eventos em ambientes gráficos livres, utilizando fluxogramas para especificação de funções e processos.

As linguagens gráficas possuem sua representação através de uma simbologia gráfica em conjunto com cadeias de símbolos alfanuméricos ou especiais. As informações e ações a serem tomadas podem ser vistas com maior rapidez e clareza através de sua representação detalhada e precisa. Ainda, o entendimento dos processos pode ser visualizado passo a passo [MEN89].

Nas linguagens gráficas livres a simbologia gráfica é arbitrária, ou seja, não está baseada em princípios [MEN89].

5.2 APRESENTAÇÃO DO PROTÓTIPO

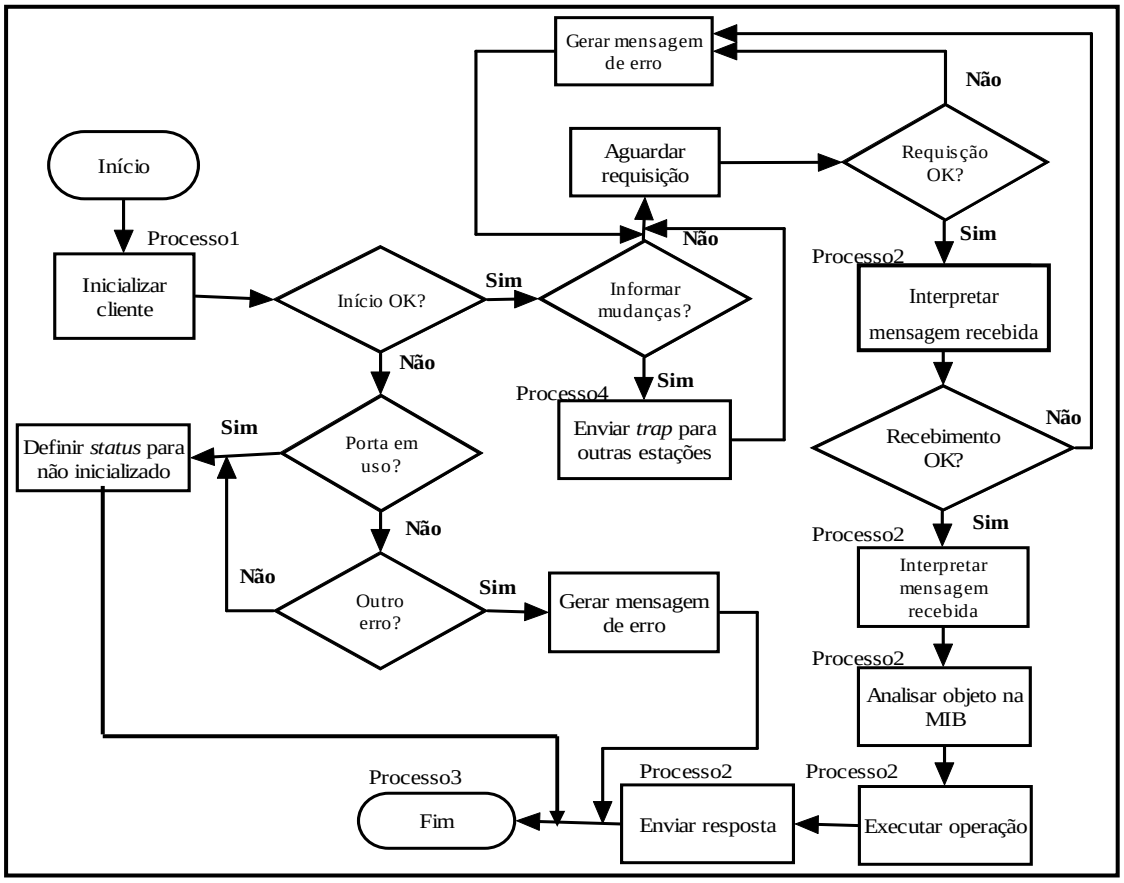
O protótipo desenvolvido é composto de dois módulos: o módulo cliente e o módulo monitor.

A seguir segue a visualização geral do protótipo divido em seus dois módulos principais, bem como, o detalhamento de seus principais processos e funções.

1) Módulo cliente

A visualização do protótipo mostra o esquema simplificado de funcionamento do protótipo do módulo cliente (fig.8).

FIGURA 8: Visualização geral do módulo cliente.

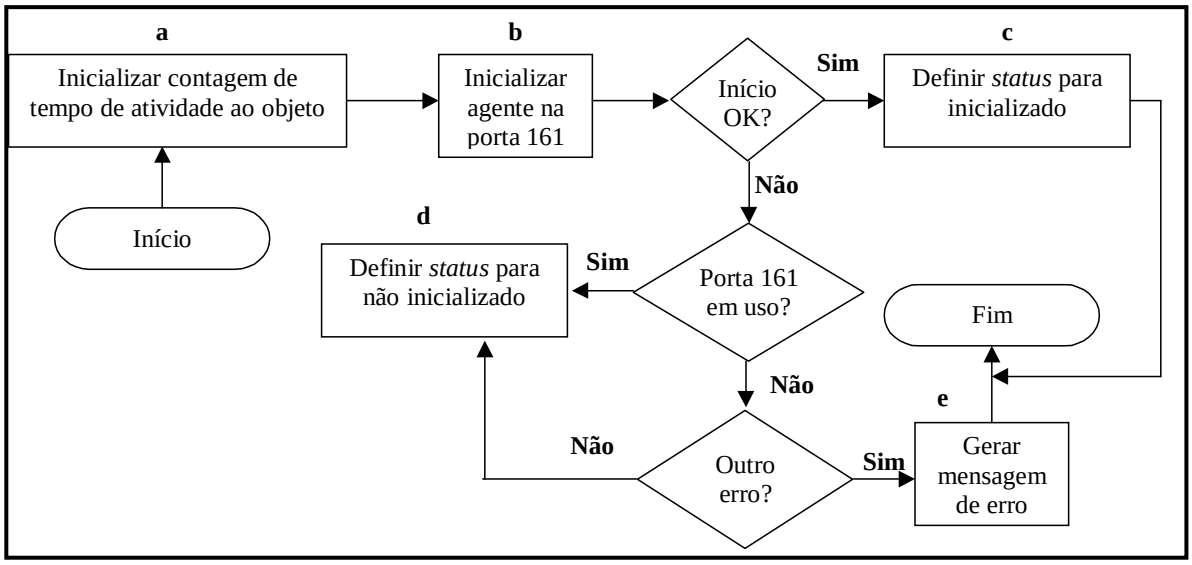


Para melhor entendimento dos processos seguem as descrições dos principais processos existentes no módulo cliente. As descrições estão ilustradas nas figs. 9, 10, 11, 12, 13 e 14.

a) Inicialização do Agente

Esta descrição tem por finalidade mostrar a inicialização do módulo cliente (processo 1 do esquema de visualização geral do módulo cliente). A fig. 9 representa este processo.

FIGURA 9: Inicialização do agente.



Descrição dos processos:

Processo a: A contagem do tempo é inicializada quando o agente for inicializado (contagem de tempo para o agente ativo).

Processo b: O agente é inicializado na porta 161 (porta padrão para os clientes do protocolo SNMP).

Processo c: O *status* do agente é dado como inicializado (ativo), isto é, ele fica "escutando" na porta 161, esperando requisições/informações a seu respeito.

Processo d: Caso a porta 161 estiver em uso por um aplicativo, podendo este até ser uma outra instância do mesmo agente, seu *status* é definido como não

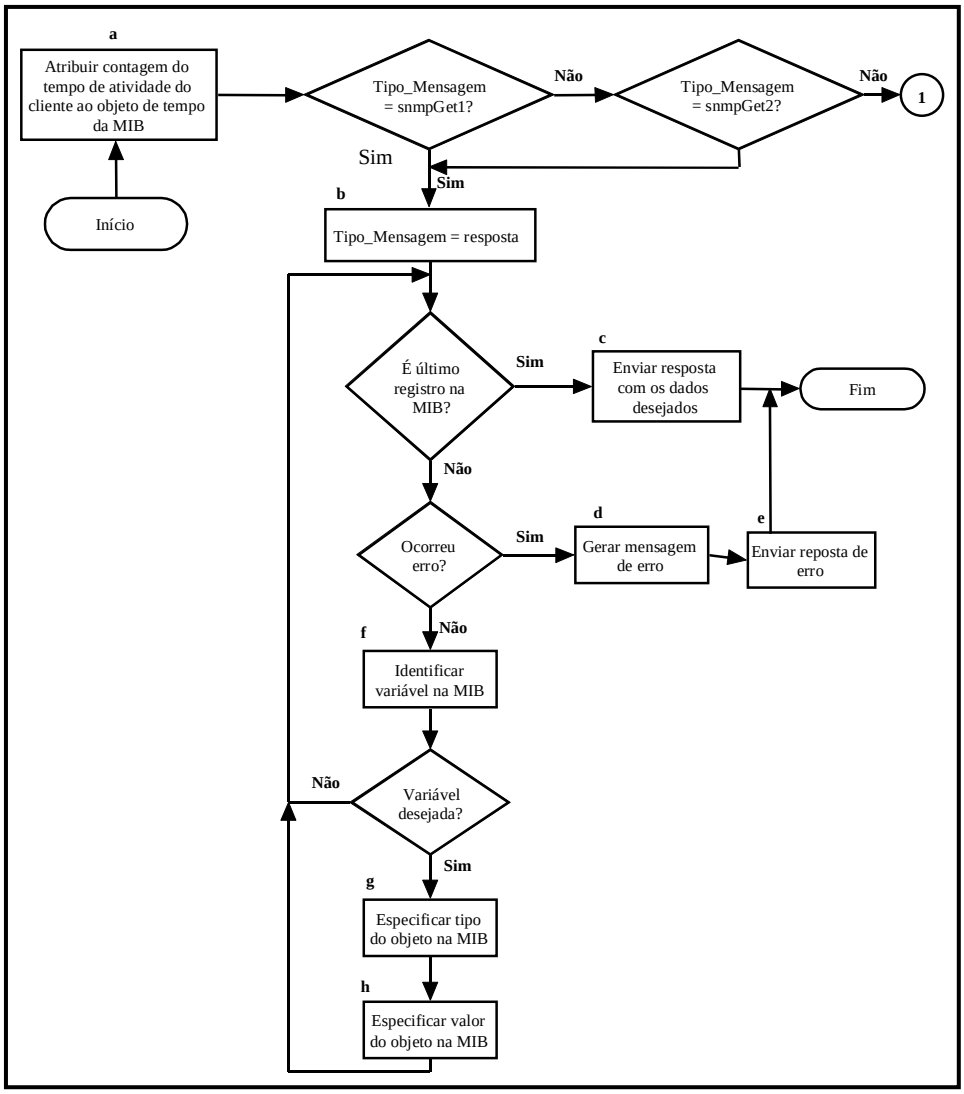
inicializado (desativado) e o agente não está em modo de espera para acesso via monitor.

Processo e: É gerado uma mensagem do erro ocorrido e mostrado o mesmo em console.

b) Interpretar Mensagem Recebida

Esta descrição é equivalente a: Interpretar Mensagem Recebida, Analisar Objeto na MIB, Executar operação, Gerar erro e enviar resposta do fluxograma genérico (processo 2 do esquema de visualização geral do módulo cliente). A representação desta descrição é representada pelas figs. 10, 11 e 12, mostradas a seguir.

FIGURA 10: Interpretação da mensagem como *snmpGet1* e *snmpGet2*.



Descrição dos processos:

Processo a: É inicializada a contagem do tempo de atividade do cliente. Tempo máximo de atividade permitido é de aproximadamente 50dias. Após este período a contagem deve ser reinicializada.

Processo b: O tipo da mensagem é atribuído como *snmpResponse* (resposta) para poder enviar o retorno dos dados requisitados.

Processo c: Os dados requisitados são enviados para o módulo monitor responsável pela estação ou segmento monitorado.

Processo d: O erro ocorrido é interceptado e identificado para que esta ocorrência possa ser mostrada.

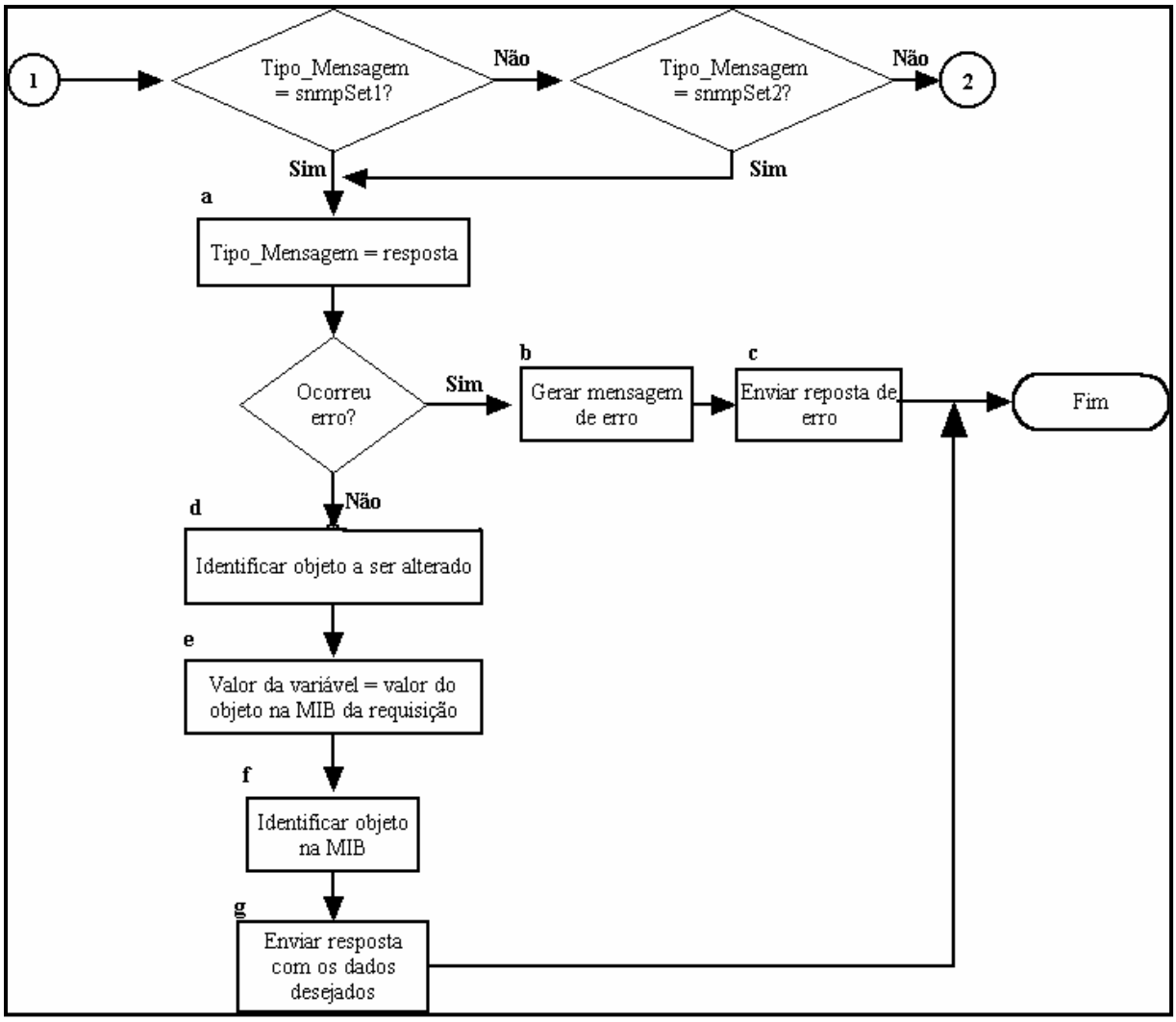
Processo e: O erro encontrado em *d* é mostrado.

Processo f: O objeto a ser identificado é comparada através de sua identificação (OID) na MIB do módulo cliente, para ver se este objeto está contido na MIB e possa assim ser monitorado (gerenciado).

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

Processo h: É feita a atribuição do valor do objeto no segmento monitorado.

FIGURA 11: Interpretação da mensagem como *snmpSet1* e *snmpset2*.



Descrição dos processos:

Processo a: O tipo da mensagem é atribuído como *snmpResponse* (resposta) para poder enviar o retorno dos dados requisitados.

Processo b: O erro ocorrido é interceptado e identificado para que esta ocorrência possa ser mostrada.

Processo c: O erro encontrado em *b* é mostrado.

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

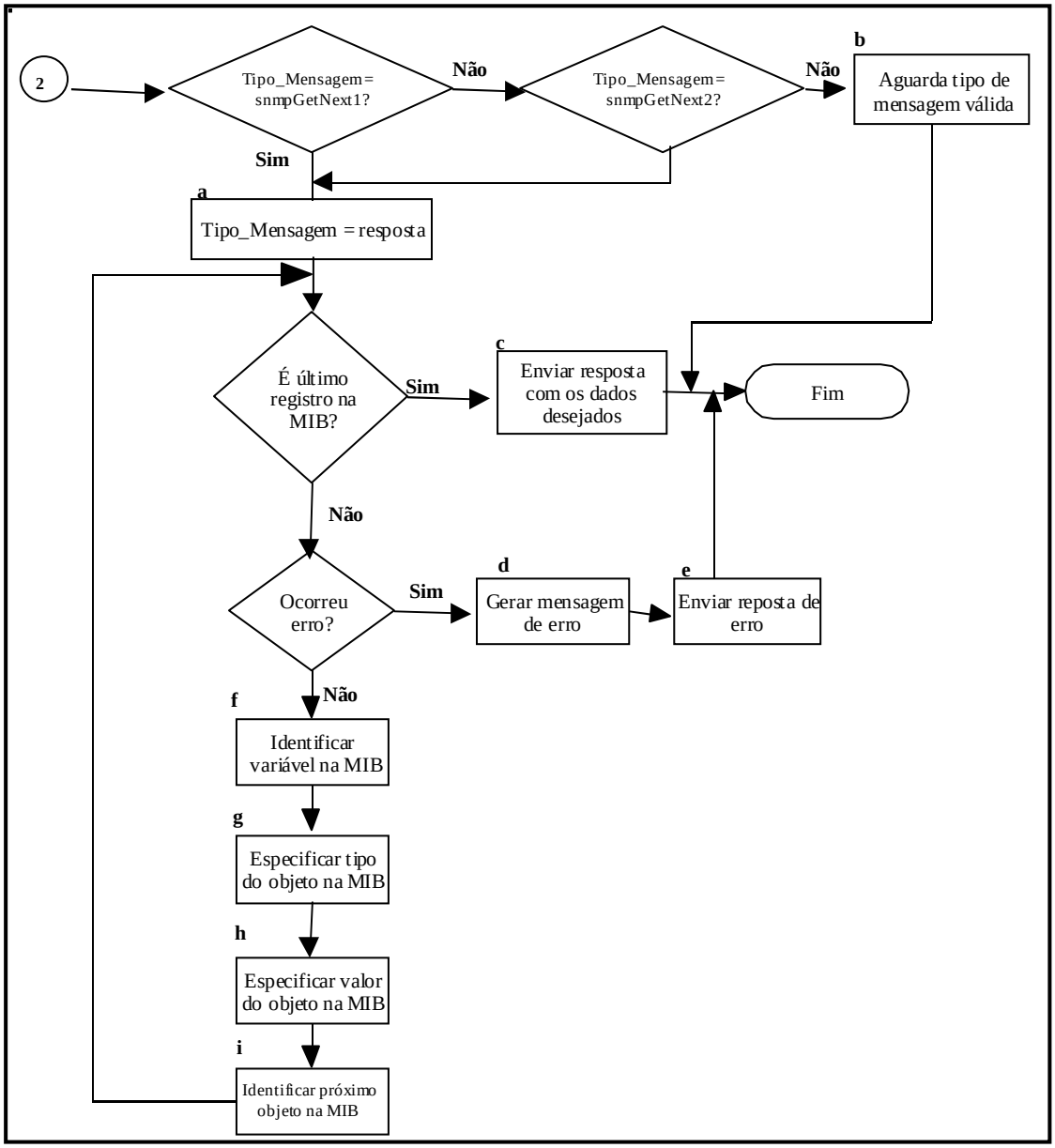
Processo e: A valor do objeto local recebe o valor do objeto que foi alterado no modulo monitor (se houver permissão de alteração).

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

Processo g: Os dados requisitados são enviados para o módulo monitor responsável pela estação ou segmento monitorado.

Para complementar as duas figuras anteriores (figs. 10 e 11) segue a fig. 12.

FIGURA 12: Interpretação da mensagem como *snmpGetNext1* e *snmpGetNext2.*



Descrição dos processos:

Processo a: O tipo da mensagem é atribuído como *snmpResponse* (resposta) para poder enviar o retorno dos dados requisitados.

Processo b: Não executa nenhuma operação. Aguarda tipo de mensagem válida.

Processo c: Os dados requisitados são enviados para o módulo monitor responsável pela estação ou segmento monitorado.

Processo d: O erro ocorrido é interceptado e identificado para que esta ocorrência possa ser mostrada.

Processo e: O erro encontrado em *d* é mostrado.

Processo f: O objeto a ser identificado é comparada através de sua identificação (OID) na MIB do módulo cliente, para ver se este objeto está contido na MIB e possa assim ser monitorado (gerenciado).

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

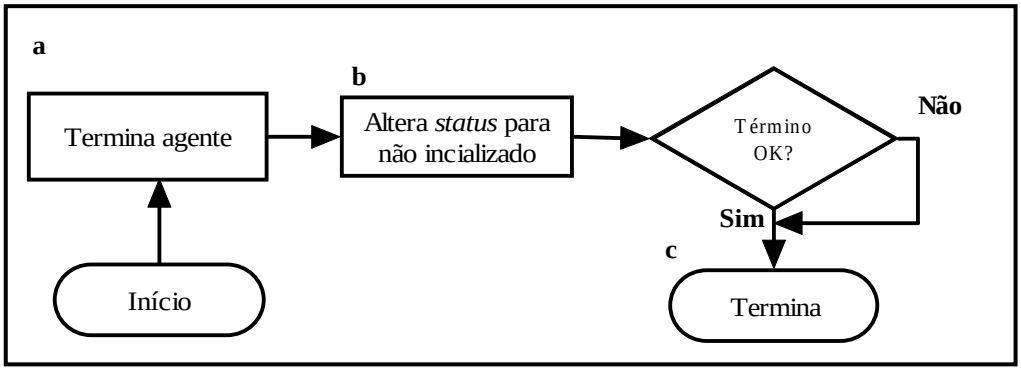
Processo h: É feita a atribuição do valor do objeto no segmento monitorado.

Processo i: É atribuído o valor do próximo objeto a ser identificado na MIB. A identificação ocorre através de sua identificação (OID) na MIB do módulo cliente, para ver se este objeto está contido na MIB e possa assim ser monitorado (gerenciado).

c) Finalizar programa (módulo cliente)

Esta parte da especificação demonstra o término do programa através da fig. 13 (processo 3 do esquema de visualização geral do módulo cliente).

FIGURA 13: Término do módulo cliente



Descrição dos Processos:

Processo a: Termina agente na porta 161.

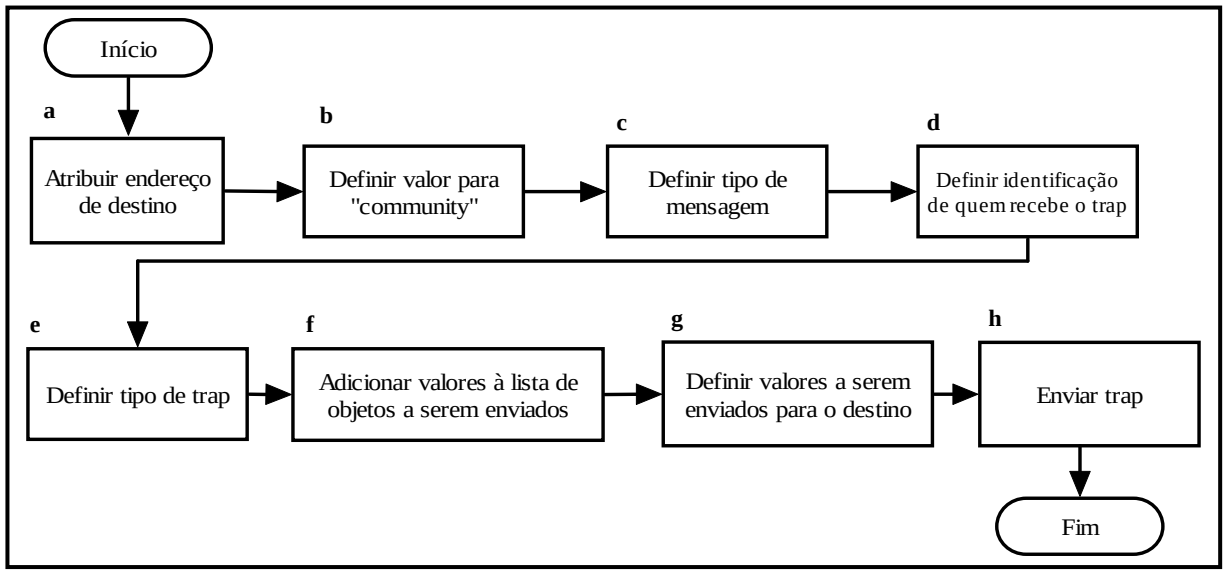
Processo b: Altera o *status* agente de inicializado para não inicializado.

Processo c: Encerrar e fechar o programa.

d) Enviar Trap

Através dos *traps* o módulo pode enviar notificações avisando que ocorreu alguma alteração ou outro evento em seu módulo. Como o *trap* é composto pode ser visto na fig. 14 (processo 4 do esquema de visualização geral do módulo cliente).

FIGURA 14: Envio de *Trap*.



Descrição dos Processos:

Processo a: Especificar o endereço de destino para o envio do *trap*.

Processo b: Definir o valor para quem deve ter permissão de ver/receber a mensagem. Por padrão (*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 quem é dono da mensagem enviada, isto é, quem será o real receptor da mensagem.

Processo e: Especificar o tipo de mensagem genérica de *trap*.

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

Processo g: Os objetos especificados em *f* recebem a atribuição de seus valores.

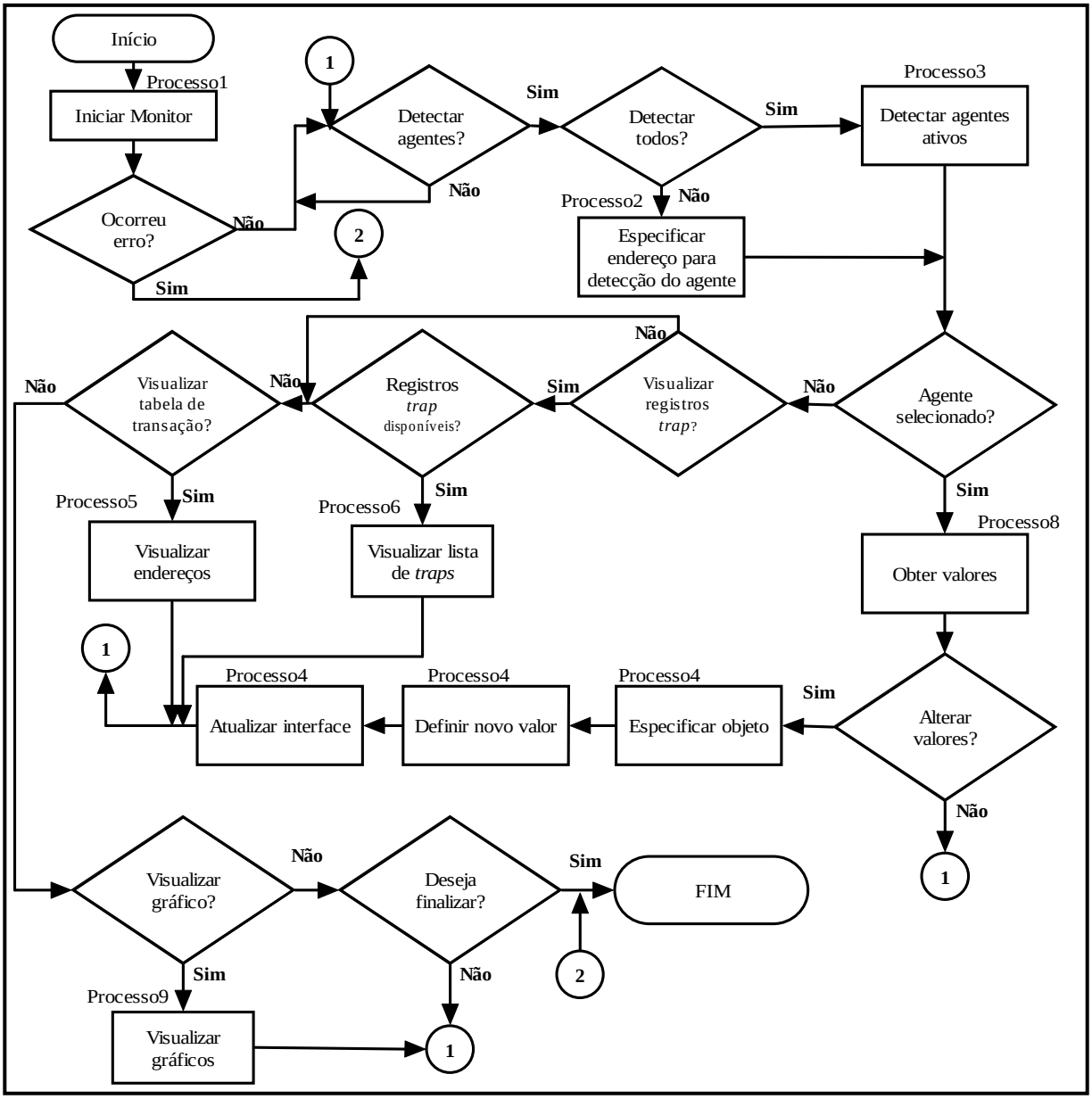
Processo h: Envio do *trap* para seu destino com os objetos e valores desejados.

A seguir segue a descrição do módulo monitor e seus principais processos.

2) Módulo Monitor

A visualização geral do módulo monitor pode ser vista na fig. 15 a seguir.

FIGURA 15: Visualização geral do módulo monitor.

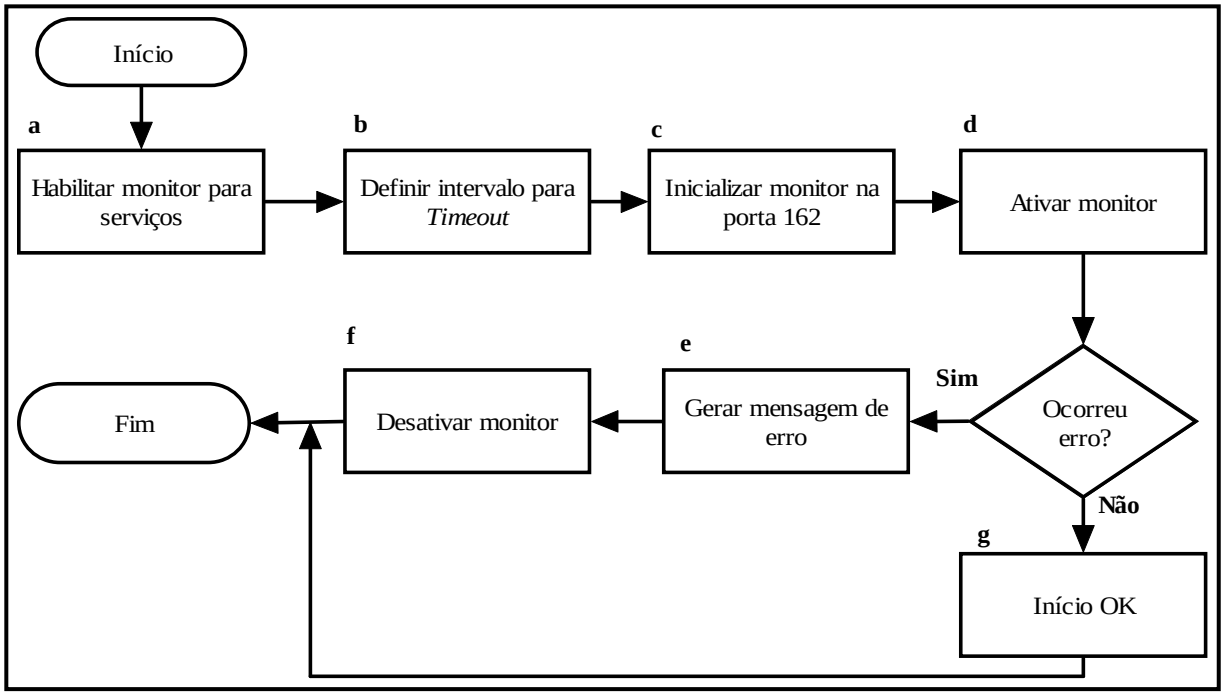


Para melhor entendimento dos processos seguem as descrições dos principais processos existentes no módulo cliente. As descrições estão ilustradas nas figs. 16, 17, 18, 19, 20, 21 e 22.

a) Inicialização do Monitor

A inicialização do monitor (processo 1 do esquema de visualização geral) ocorre conforme descrito na fig. 16.

FIGURA 16: Inicialização do Monitor



Descrição dos processos:

Processo a: Definir o estado do monitor para *trabalhando* para possibilitar as operações provenientes de envio, busca, entre outras.

Processo b: Definir o intervalo para que seja interrompida a operação em casos que não houver resposta, ou quando há uma demora muito grande para resposta.

Processo c: Iniciar o monitor. A porta padrão para monitores é a porta 162.

Processo d: Definir o estado do monitor para *ativo* para possibilitar o monitoramento.

Processo e: O erro ocorrido é interceptado e identificado para que esta ocorrência possa ser mostrada.

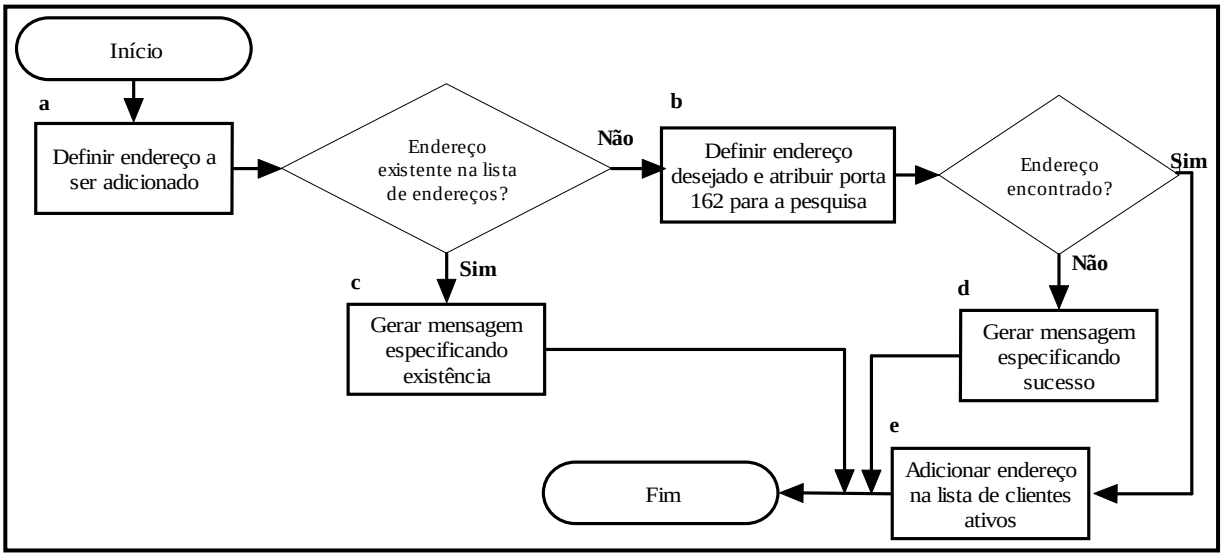
Processo f: Definir o estado do monitor para *desativado* para possibilitar o monitoramento.

Processo g: Inicio foi efetuado corretamente.

b) Adicionar cliente específico

Como um cliente específico é adicionado ao monitor pode ser visto na fig. 17 (processo 2 do esquema de visualização geral):

FIGURA 17: Adicionar cliente específico.



Descrição dos processos

Processo a: Especificar o endereço da rede ou segmento que deve ser adicionado à lista de endereços de clientes ativos.

Processo b: Uso do endereço especificado em *a* para realizar a busca na rede.

Processo c: Caso o endereço especificado já exista na lista, informar a existência do endereço na lista.

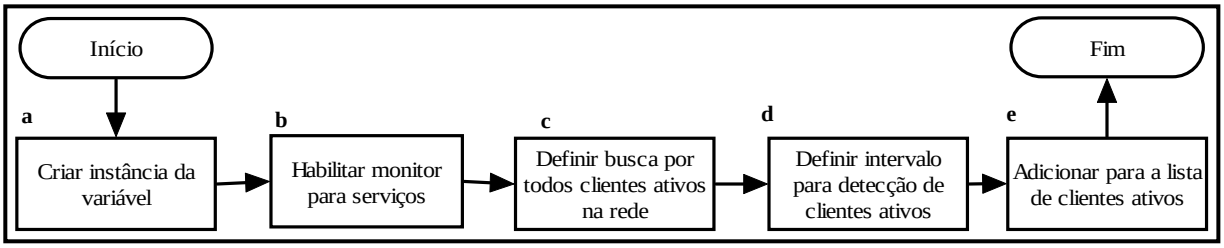
Processo d: Informar o resultado da inclusão do endereço especificado na lista de clientes.

Processo e: Adicionar e mostrar o endereço especificado na lista de clientes ativos.

c) Adicionar múltiplos clientes

Como vários clientes são adicionados ao monitor a partir de uma única busca pode ser visto na fig. 18 (processo 3 do esquema de visualização geral):

FIGURA 18: Adicionar múltiplos clientes.



Descrição dos processos

Processo a: Criar nova instância de uma variável para adicionar múltiplos clientes.

Processo b: Definir o estado do monitor para *trabalhando* para possibilitar as operações provenientes de envio, busca, entre outras.

Processo c: Caso o cliente estiver ativo, objetivar a busca e detecção do cliente.

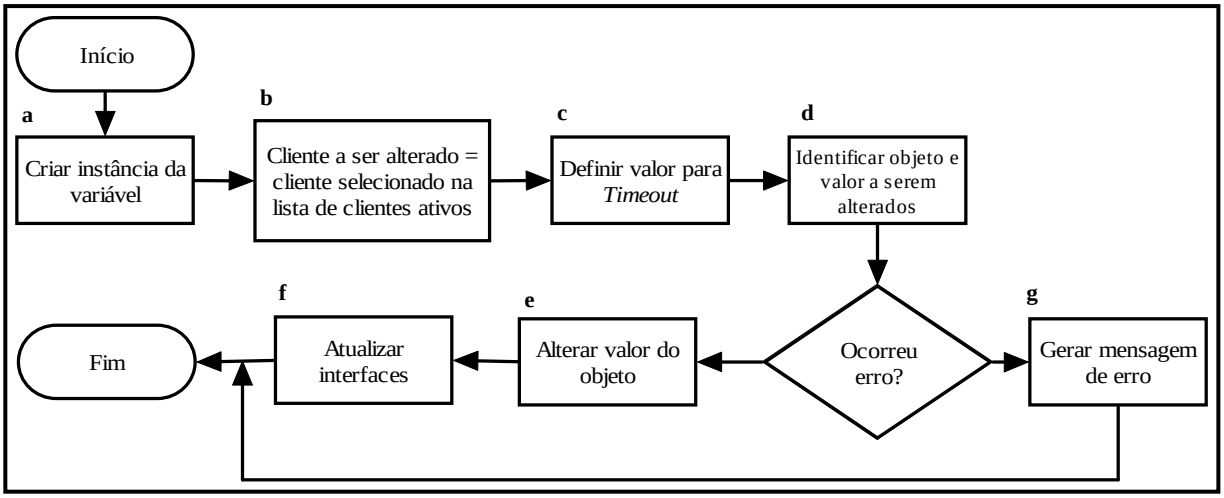
Processo d: Definir intervalo de espera, caso não haja resposta ou não seja possível detectar um cliente ou o objeto a ser alterado.

Processo e: Quando encontrado, adicionar o cliente à lista de clientes ativos.

d) Alterar valor do cliente selecionado (detalhado)

Como alterar o valor de um objeto em um cliente pode ser visto na fig. 19 (processo 4 do esquema de visualização geral).

FIGURA 19: Alterar valor do cliente selecionado.



Descrição dos processos

Processo a: Criar nova instância de uma variável para adicionar múltiplos clientes.

Processo b: O endereço do cliente a ter seu valor alterado deve ser igual ao endereço do cliente selecionado na lista de clientes ativos.

Processo c: Definir intervalo de espera, caso não haja resposta ou não seja possível detectar um cliente.

Processo d: Especificar qual objeto deve ter seu valor alterado e o valor correspondente.

Processo e: Alterar o valor do objeto desejado.

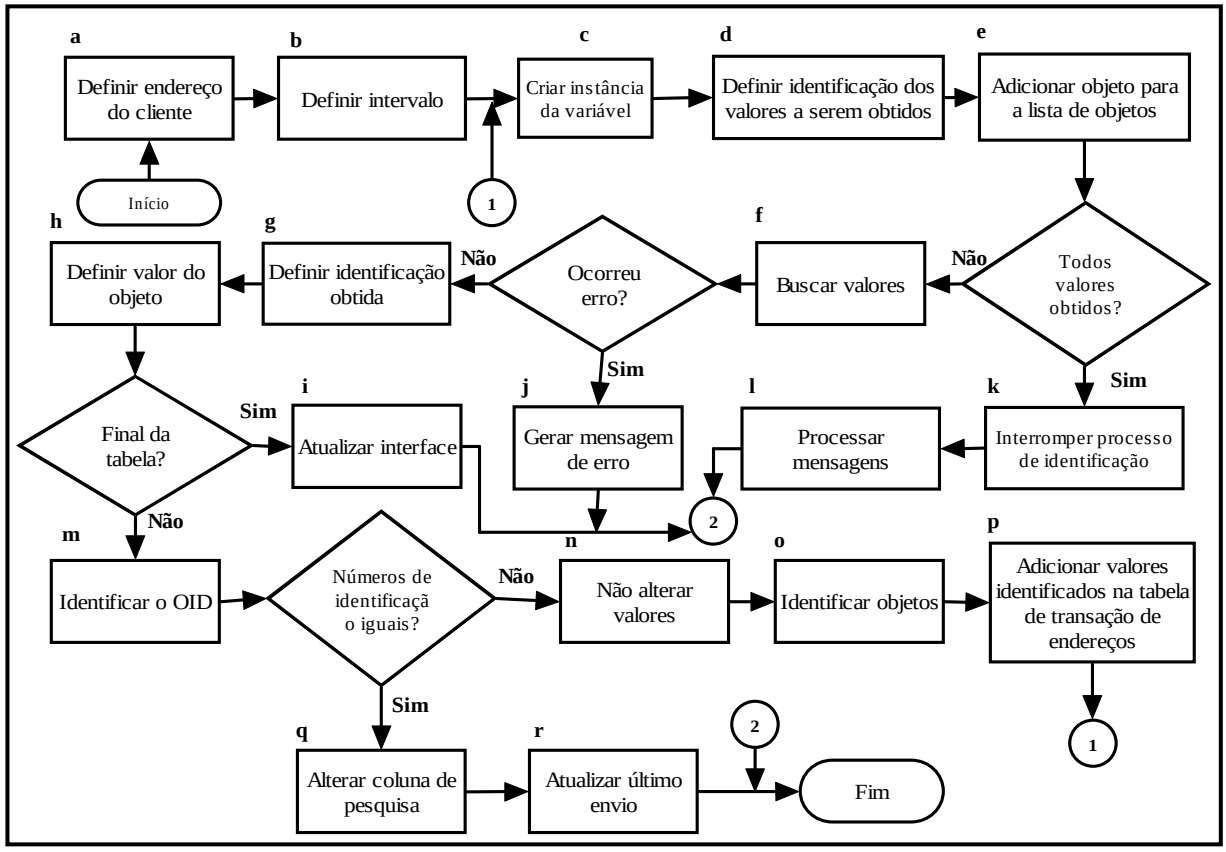
Processo f: Atualizar a interface para que a mudança possa ser visível após a alteração.

Processo g: O erro ocorrido é interceptado e identificado para que esta ocorrência possa ser mostrada.

e) Visualizar endereço do cliente selecionado

Como visualizar o endereço de um cliente pode ser visto na fig. 20 (processo 5 do esquema de visualização geral):

FIGURA 20: Visualização de endereço do cliente selecionado.



Descrição dos processos

Processo a: Especificar qual o endereço na rede para o cliente.

Processo b: Definir intervalo de espera, caso não haja resposta ou não seja possível detectar um cliente ou um objeto.

Processo c: Criar nova instância de uma variável para adicionar múltiplos clientes.

Processo d: Especificar quais valores/objetos devem ser obtidos.

Processo e: Quando encontrado, adicionar o objeto à lista de objetos para obtenção dos valores.

Processo f: Baseado nos itens *d* e *e* efetuar a busca pelos valores.

Processo g: Especificar qual objeto foi obtido.

Processo h: Obter o valor do objeto obtido no item *g*.

Processo i: Atualizar a interface para que a mudança possa ser visível após a alteração.

Processo j: O erro ocorrido é interceptado e identificado para que esta ocorrência possa ser mostrada.

Processo k: Se todos os valores forem obtidos parar a busca pelos objetos.

Processo l: Processar as informações recebidas após os valores terem sido obtidos.

Processo m: Identificar o número de identificação do objeto que deve ser obtido.

Processo n: Manter os valores de coluna e suas relações sem nenhuma alteração.

Processo o: Identificar quais objetos foram reconhecidos.

Processo p: Adicionar objetos reconhecidos para a lista de transação de endereços do cliente em questão.

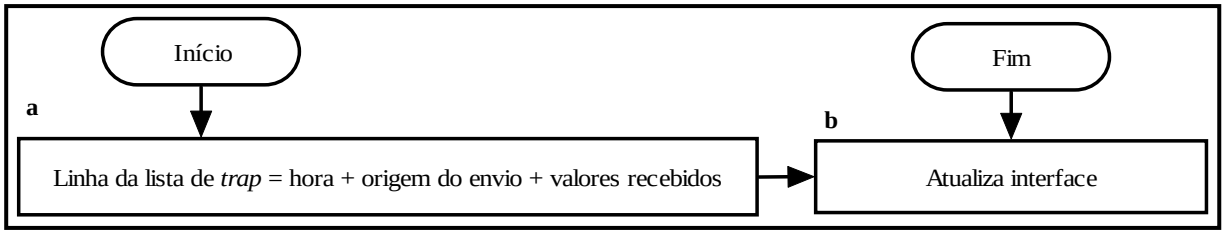
Processo q: Caso os 19 primeiro número forem iguais deve-se alterar a coluna de pesquisa, já que os valores serão todos similares, para obter os dados dos objetos desejados.

Processo r: Atualizar os dados do último envio.

f) Visualizar lista de traps

Como visualizar um *trap* enviado por um cliente pode ser visto na fig. 21 (processo 6 do esquema de visualização geral):

FIGURA 21: Visualizar lista de traps.



Descrição dos processos

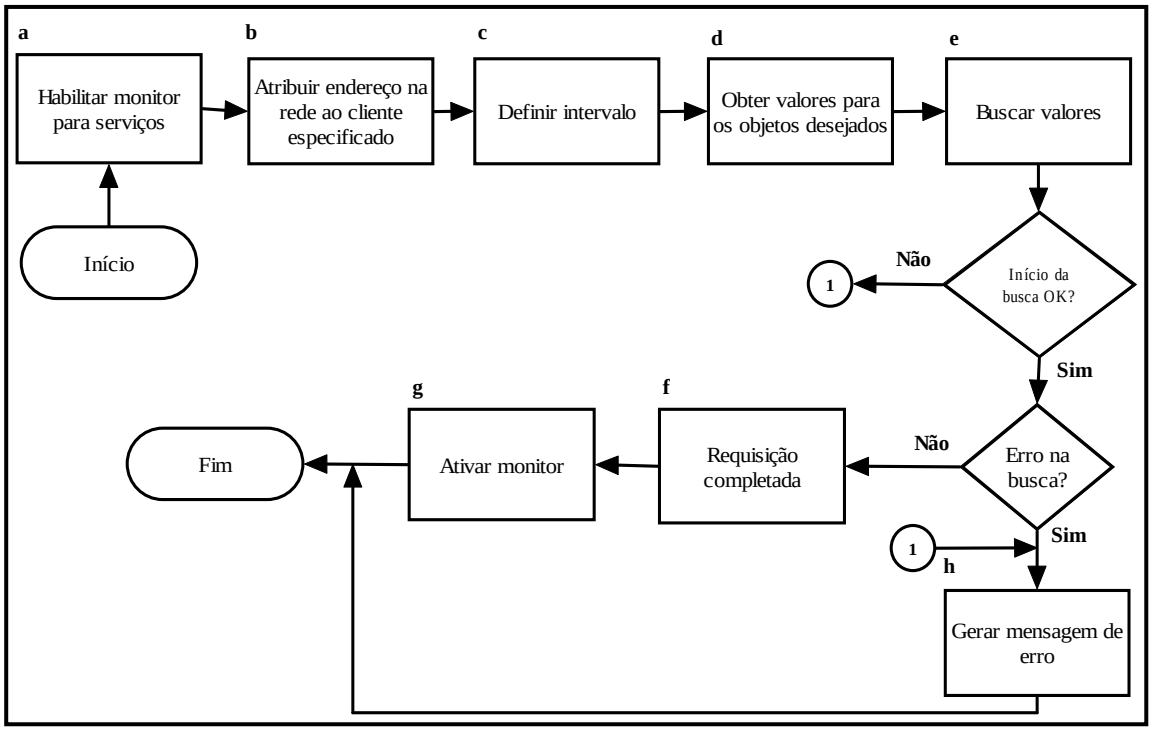
Processo a: A linha de mensagem de um *trap* é composto pela hora de envio, que originou a mensagem e quais valores foram enviados junto com o *trap*.

Processo b: Atualizar a interface para que a mudança possa ser visível após a alteração.

g) Identificar o OID (número de identificação)

Como identificar o OID de um objeto em um cliente pode ser visto na fig. 22 (processo 7 do esquema de visualização geral):

FIGURA 22: Identificação do OID.



Descrição dos processos

Processo a: Definir o estado do monitor para *trabalhando* para possibilitar as operações provenientes de envio, busca, entre outras.

Processo b: Especificar qual o endereço na rede para o cliente, a partir do qual deve ser buscado o OID desejado.

Processo c: Definir intervalo de espera, caso não haja resposta ou não seja possível detectar um cliente ou um objeto.

Processo d: Obter os valores para os objetos

Processo e: Efetuar a busca baseado nos itens *b*, *c* e *d*.

Processo f: Finalizar busca com sucesso.

Processo g: Definir o estado do monitor para *ativo* para possibilitar o monitoramento.

Processo h: O erro ocorrido é interceptado e identificado para que esta ocorrência possa ser mostrada.

5.3 IMPLEMENTAÇÃO DO PROTÓTIPO

No período de desenvolvimento do protótipo foram considerados dois fatores principais:

a) A linguagem de programação a ser utilizada;

b) O sistema operacional para o funcionamento do protótipo.

A seguir segue um pequeno detalhamento dos dois fatores mencionados acima.

5.3.1 AMBIENTE DE PROGRAMAÇÃO DELPHI

Para que a implementação do protótipo fosse possível utilizou-se o ambiente de programação *Borland Delphi 4.0* para Windows.

O aumento do uso do sistema operacional Windows, com sua interface gráfica para o usuário (*Graphical User Interface* - GUI), começava a trazer um ambiente de trabalho padrão para usuários de computadores e programadores. As interfaces puramente textuais estavam sendo cada vez mais substituídas pela interfaces gráficas. Estas interfaces gráficas eram mais fáceis de serem utilizadas, facilitando a vida de iniciantes e experientes com computadores, fazendo com que o usuário manuseasse o computador de forma mais intuitiva e não meramente por linha de comandos. O desenvolvimento de programas de quaisquer gêneros era uma tarefa demorada e em alguns casos muito longa. Tendo em foco principal o desenvolvimento a empresa *Borland* (hoje *Inprise*) criou o ambiente de programação *Delphi* em 1995 [CAN98] [CAR98].

O ambiente de programação *Delphi* utiliza a linguagem de programação *Object Oriented Pascal*, que é uma evolução do Pascal padrão [CAN98] [CAR98] [OLI98] [REI99].

No *Delphi*, o desenvolvimento de aplicativos inicia com a montagem de componentes em janelas, como se fosse um programa gráfico. Este componentes são auxiliares para o desenvolvimento da interface gráfica ao usuário e podem ser nativos (próprios do ambiente) ou desenvolvidos por terceiros (*shareware*) e distribuídos de graça (*freeware*) ou através do pagamento de uma taxa, como se fossem programas. Os usuários podem criar novos componentes e utilizá-los conforme surgirem as necessidades para cada um [ENG97] [OLI98] [REI99] [SHI99].

Mais informações sobre o ambiente de programação *Delphi* e a linguagem OOP podem ser encontradas em [ENG97], [CAN98], [CAR98], [OLI98] e [REI99].

5.3.2 SISTEMA OPERACIONAL WINDOWS NT

O sistema operacional Windows NT oferece a implementação do protocolo SNMP na sua primeira versão e juntamente uma agente que possibilita o gerenciamento remoto da rede ou de seus segmentos. Entre as opções gerenciáveis, segundo [MAF99] e [MIC97] estão:

a) Computadores Windows NT Server;

b) Computadores Windows NT Workstation;

c) Servidores WINS baseados no Windows NT;

d) Servidores DHCP baseados no Windows NT;

e) Computadores Internet Information Server baseados no Windows NT;

f) Servidores LAN Manager.

O agente SNMP baseado no Windows NT é implementado como um serviço. A implementação de agentes como serviço oferece uma maior segurança aos dados e funções do próprio sistema operacional, enquanto, em contrapartida, se ao agente funcionar somente como aplicativo, ele oferece menos segurança operando a nível do usuário conectado a estação monitorada. O agente implementado como serviço pode ser instalado nos computadores baseados no Windows NT que utilizam os protocolos TCP/IP e IPX. Para o funcionamento apropriado do agente, o protocolo TCP/IP deve ser instalado antes da instalação do SNMP [MAF99] [MIC97].

O serviço SNMP é implementado como um serviço de 32 bits do Windows utilizando o *Windows Sockets* (porta virtual que permite a conexão entre servidor e clientes) sobre o TCP/IP e o IPX/SPX. MIBs adicionais de diversos fornecedores para especificação de objetos gerenciáveis complementam o gerenciamento SNMP aos serviços baseados em Windows NT existentes. Os programas agentes que implementam e fornecem essas MIBs adicionais são chamados de agentes de extensão ou complementares [MAF99] [MIC97].

5.4 FUNCIONAMENTO DO PROTÓTIPO

Neste tópico serão apresentadas as principais funções dos dois módulos desenvolvidos: o módulo cliente e o módulo monitor.

a) O módulo cliente

Ao iniciar o módulo monitor são carregados dados referente a descrição da entidade. Dados como, descrição, identificação do fornecedor na MIB, pessoa para contato, o administrador responsável pela estação ou segmento e localização do módulo, são carregados a partir de dados contidos (pré-carregados) no registro do Windows. Caso não existirem dados no registro, é criada um nova chave de valores no registro com valores pré-definidos no programa. O armazenamento dos dados no registro permite a busca de dados acelerada destes dados, já que não é preciso armazená-los no em arquivo. Ainda, são inicializadas os valores a serem gerenciados e o tempo de atividade deste módulo. Os dados, referentes à identificação

do módulo, também podem ser alterados manualmente. Os dados alterados são atualizados automaticamente no registro e, na próxima execução do módulo neste segmento, já estão disponíveis conforme a última alteração.

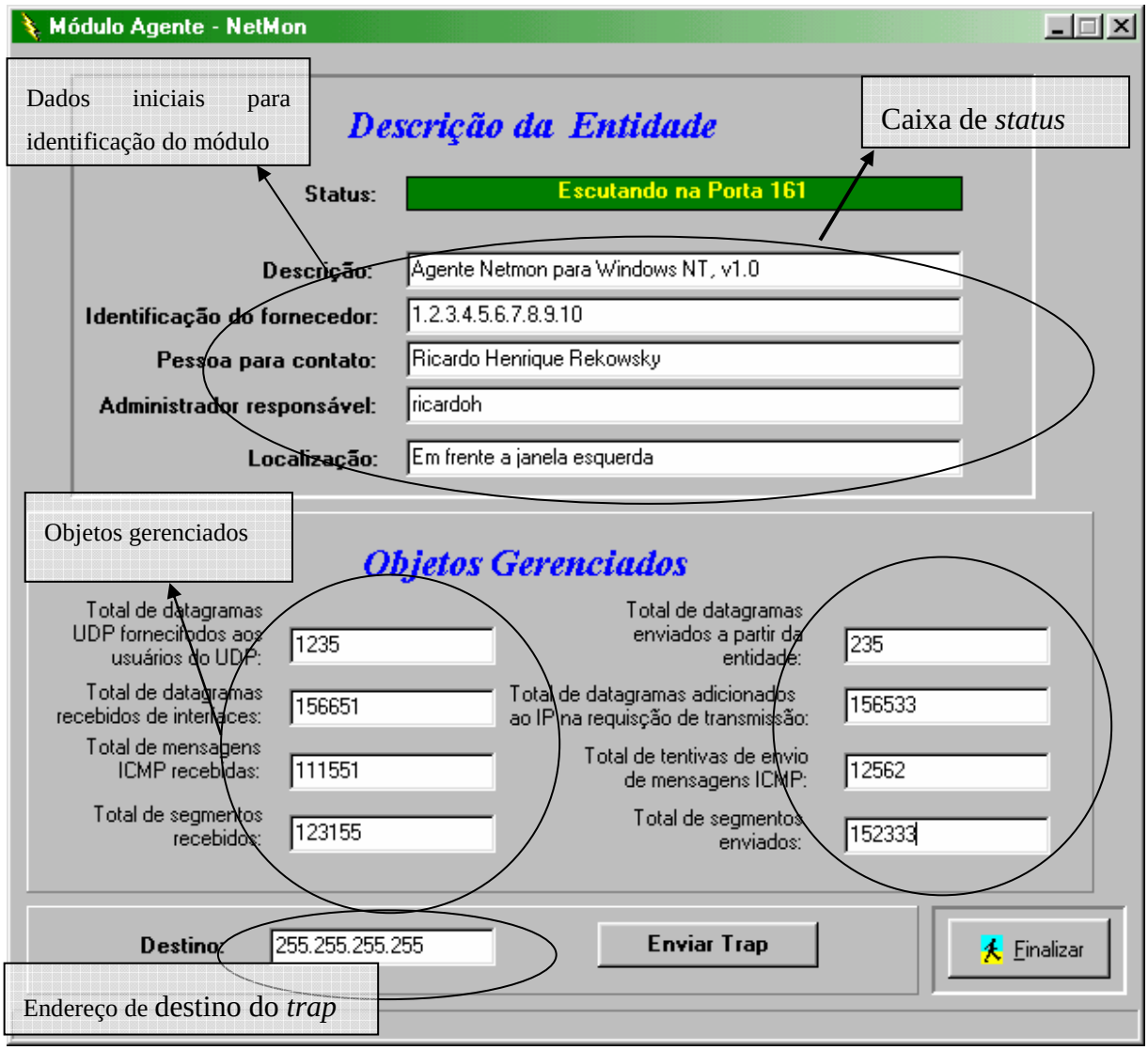
Para visualizar o funcionamento do módulo há uma caixa que exibe o *status* corrente do módulo. Em caso de funcionamento correto, aparece nesta caixa a mensagem *"Escutando na Porta 161"* . Caso houver algum erro, outra mensagem será mostrada para que o estado do módulo sempre esteja visível no momento de visualização do módulo.

O módulo cliente monitora oito objetos:

1) Total de datagramas UDP fornecidos aos usuários UDP;
2) Total de datagramas enviados a partir da entidade;
3) Total de datagramas recebidos de interfaces;
4) Total de datagramas adicionados ao IP na requisição de transmissão;
5) Total de mensagens ICMP recebidas;
6) Total de tentativas de envio de mensagens ICMP;
7) Total de segmentos recebidos;
8) Total de segmentos enviados.

A fig. 23 mostra a tela principal do módulo cliente.

FIGURA 23: Tela inicial do módulo cliente.



Ainda, no momento de inicialização do cliente, é iniciado o contador de tempo responsável pelo tempo total de atividade do módulo cliente. Este tempo serve para visualizar a quanto tempo o módulo está em atividade. Desta forma, também é possível verificar quedas desconhecidas pelo módulo monitor e o correto funcionamento do módulo cliente.

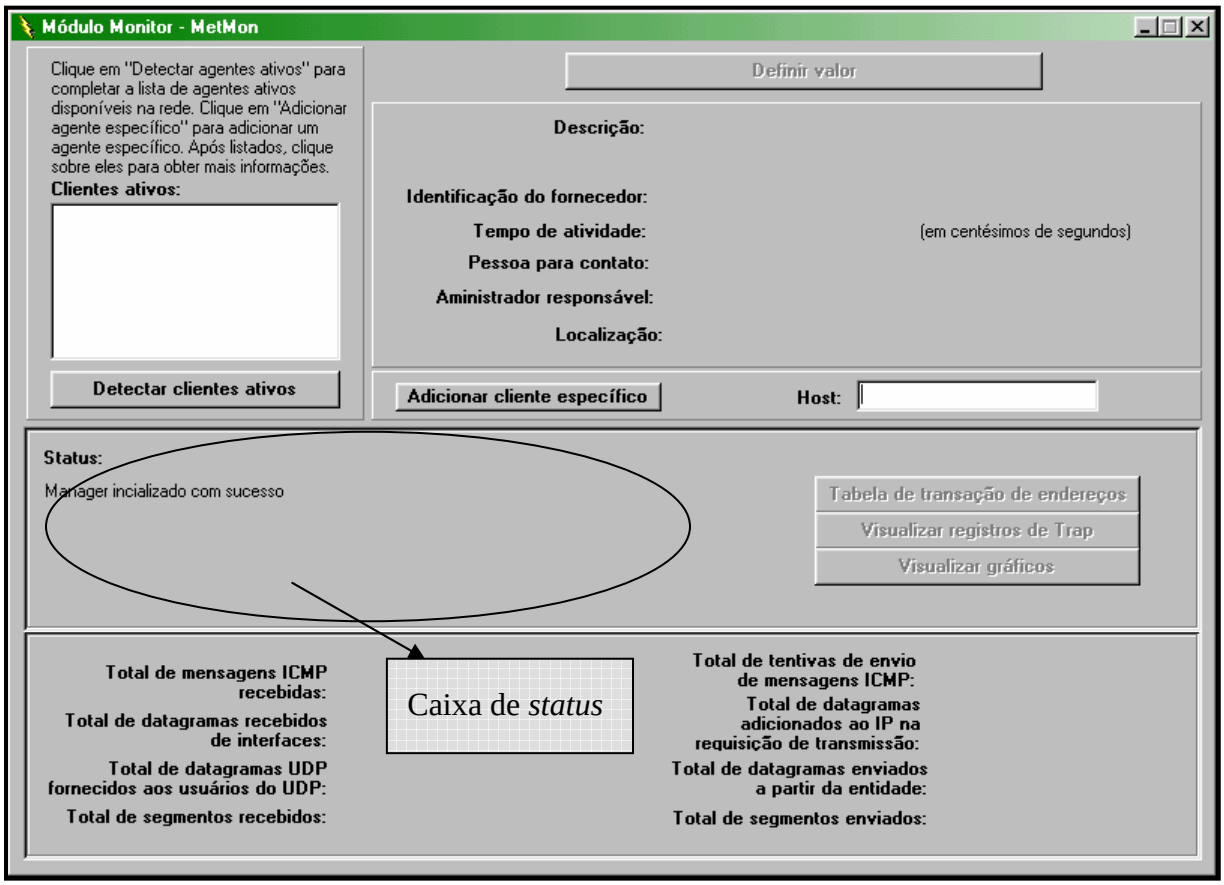
As alterações ocorridas podem ser notificadas ao módulo monitor através do envio de um *trap*, isto é, de uma mensagem especificando que houve algum tipo de alteração. O envio de um *trap* é feito através da definição de um endereço IP na rede, para onde destina-se o envio da mensagem (fig. 23). Neste protótipo somente são enviados valores para especificar o

envio de *trap*, sem definir valores de quaisquer objetos gerenciados, mas sim valores pré-definidos.

b) O módulo monitor

Quando o módulo monitor é executado, ele é iniciado na porta 162 para que ele possa comunicar-se com os módulos clientes que estiverem ativados em algum local da rede ou segmento. A porta 162 é a porta de comunicação padrão para monitores, especificado no protocolo SNMP. Se durante o início do monitor não ocorrer nenhum tipo de erro, este módulo estará apto a começar o monitoramento dos clientes ativos existentes na rede. Para poder informar o funcionamento do módulo, suas mensagens de erro ou conclusão há uma caixa de *status* (fig. 24) que informa o resultado das solicitações e requisições efetuadas. A fig. 24 apresenta a tela inicial do módulo cliente.

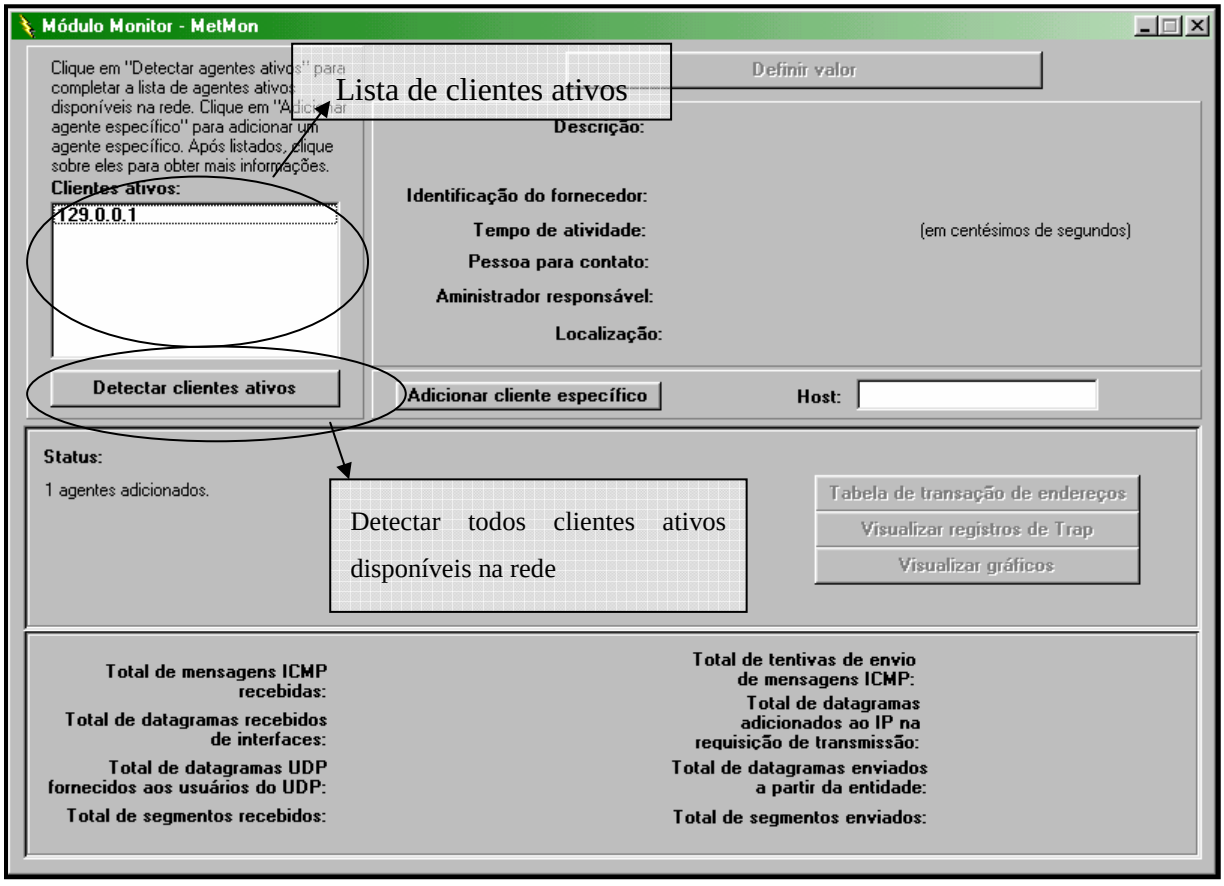
FIGURA 24: Tela inicial do módulo monitor.



Para iniciar o monitoramento dos clientes é preciso primeiro detectar se existem clientes ativos disponíveis na rede. A detecção de clientes ativos pode ser feita de duas formas: a primeira ocorre através da busca de todos os clientes ativos existentes na rede; já a segunda ocorre através da definição de um endereço IP correspondente a um módulo cliente disponível na rede.

A detecção de todos agentes disponíveis na rede ocorre quando o botão "*Detectar agentes ativos*"' é pressionado (fig. 25). Quando este botão é pressionado, o módulo monitor inicia uma busca na rede por clientes que estiverem inicializados e quando encontra um cliente ativo, adiciona-o na lista de clientes ativos (fig. 25). A fig. 25 mostra a visualização da detecção de clientes ativos na rede.

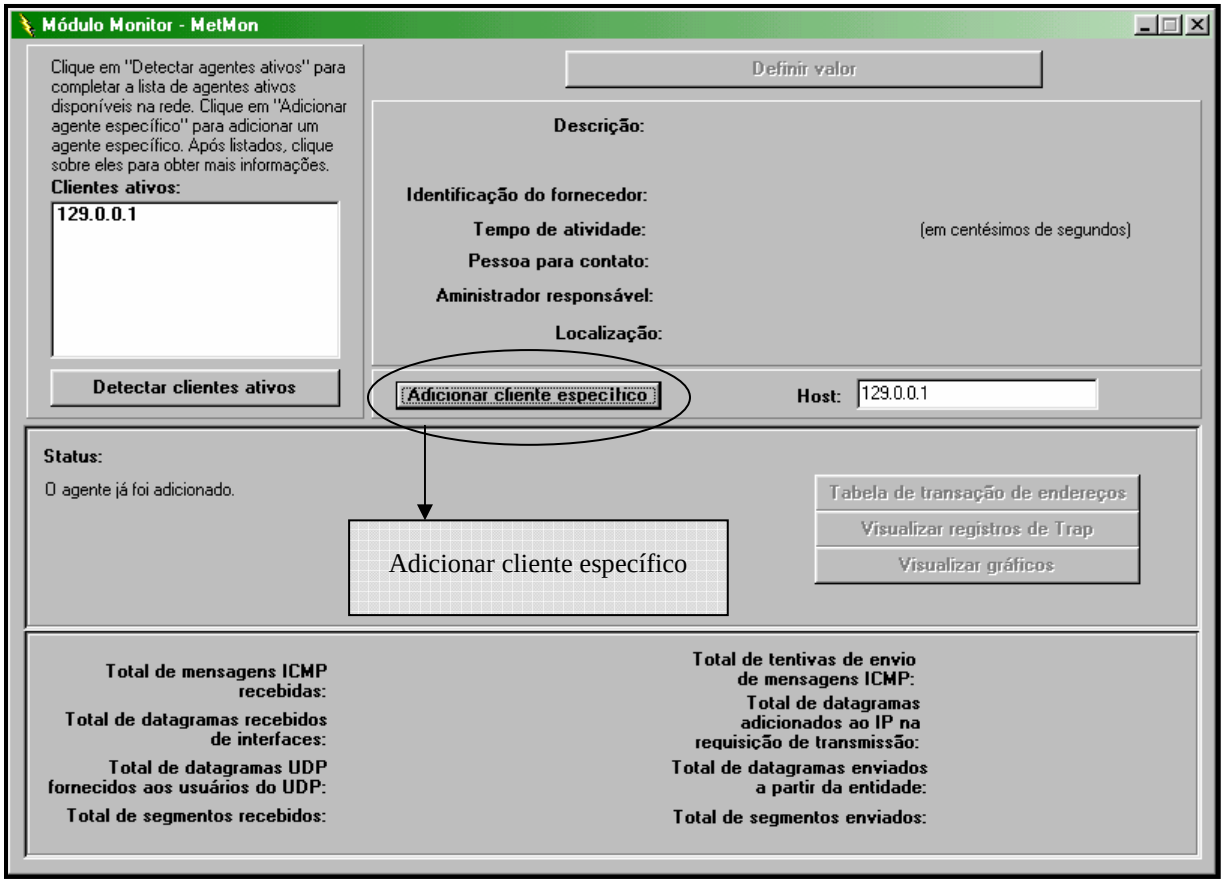
FIGURA 25: Detecção de todos clientes ativos disponíveis na rede.



A segunda forma possível, ou seja, a detecção de um cliente específico, ocorre através da entrada de um endereço IP correspondente a um cliente ativo na rede. Após a entrada deste endereço é primeiramente analisado, se o cliente indicado não se encontra disponível na lista

de clientes ativos. Se o cliente desejado já estiver disponível, é gerada uma mensagem indicando sua existência. Caso o cliente não estiver disponível, é iniciada uma busca na rede pelo endereço especificado e se encontrado, o cliente especificado é adicionado à lista de endereços de clientes ativos. A fig. 26 mostra a detecção de um cliente específico.

FIGURA 26: Detecção de um cliente específico.

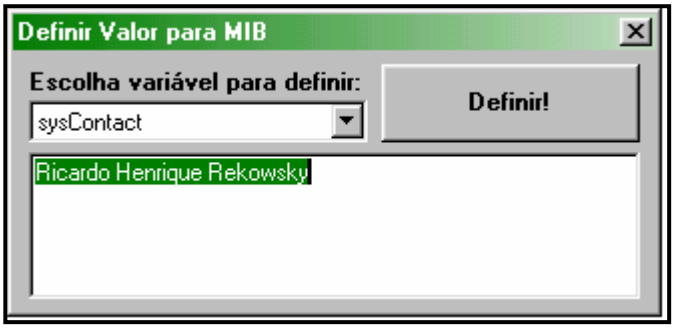


Após a detecção de clientes ativos na rede, outras funções tornam-se disponíveis para execução. A primeira a ser apresentada, é a definição de valores de clientes disponíveis na lista de clientes ativos.

Para definir valores, é necessário selecionar um cliente para obter os dados correspondentes a ele. Assim que for selecionado todos os dados disponíveis, referentes ao cliente especificado, estarão visíveis. A alteração de valores ocorre quando o botão "*Definir valor*" é pressionado. Ao pressionar este botão apresenta-se uma nova tela aonde as variáveis de identificação do módulo cliente selecionado pode ser alterado. O valor alterado será, ao fechar esta tela, automaticamente atualizado no módulo monitor e o envio dos novos valores

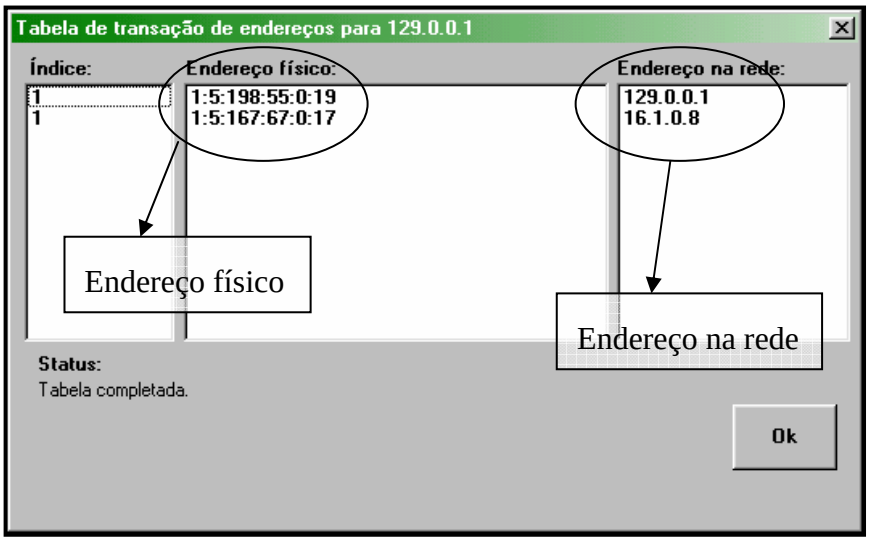
será feito destinado ao cliente que teve seu valor alterado para a módulo cliente possa Ter seus dados atualizados, tanto na interface como no registro. A alteração de valores pode ser útil para atualizar dados, definir exatamente aonde o módulo cliente encontra-se, sem que, necessariamente, o administrador da rede precise deslocar-se até a entidade gerenciada. A tela de alteração de valores pode ser vista na fig. 27.

FIGURA 27: Alteração de valores de clientes.



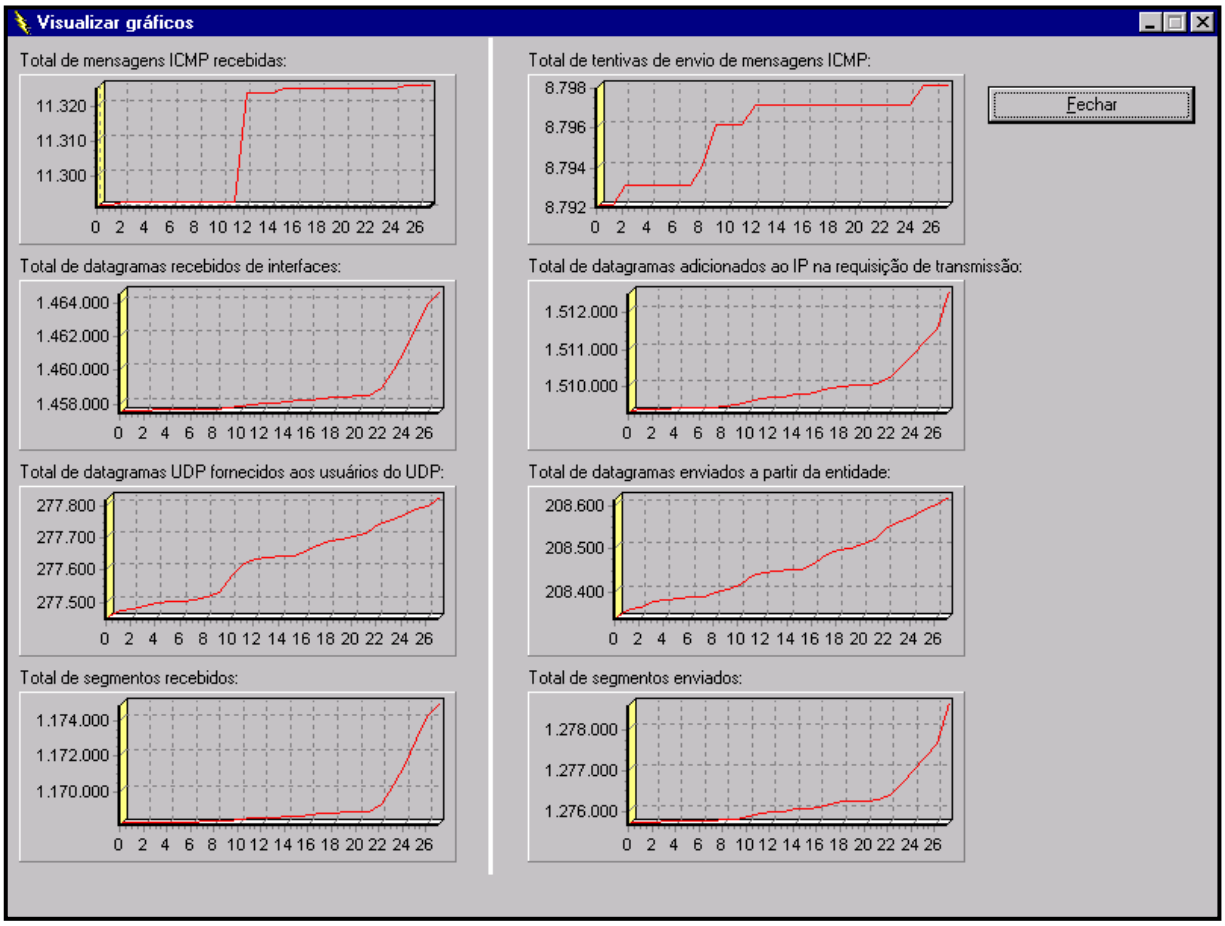
Outra função que se torna disponível quando um cliente é encontrado, é a visualização da tabela de transação de endereços para o cliente selecionado. Esta função é executada quando o botão "*Tabela de transação de endereços*" é pressionado (fig. 28). Ao pressionar este botão é efetuada uma busca na rede pelo endereço físico da placa e o endereço do cliente na rede, do cliente selecionado. A visualização tabela de transação de endereços pode ser vista na fig. 28.

FIGURA 28: Tabela de transação de endereços.



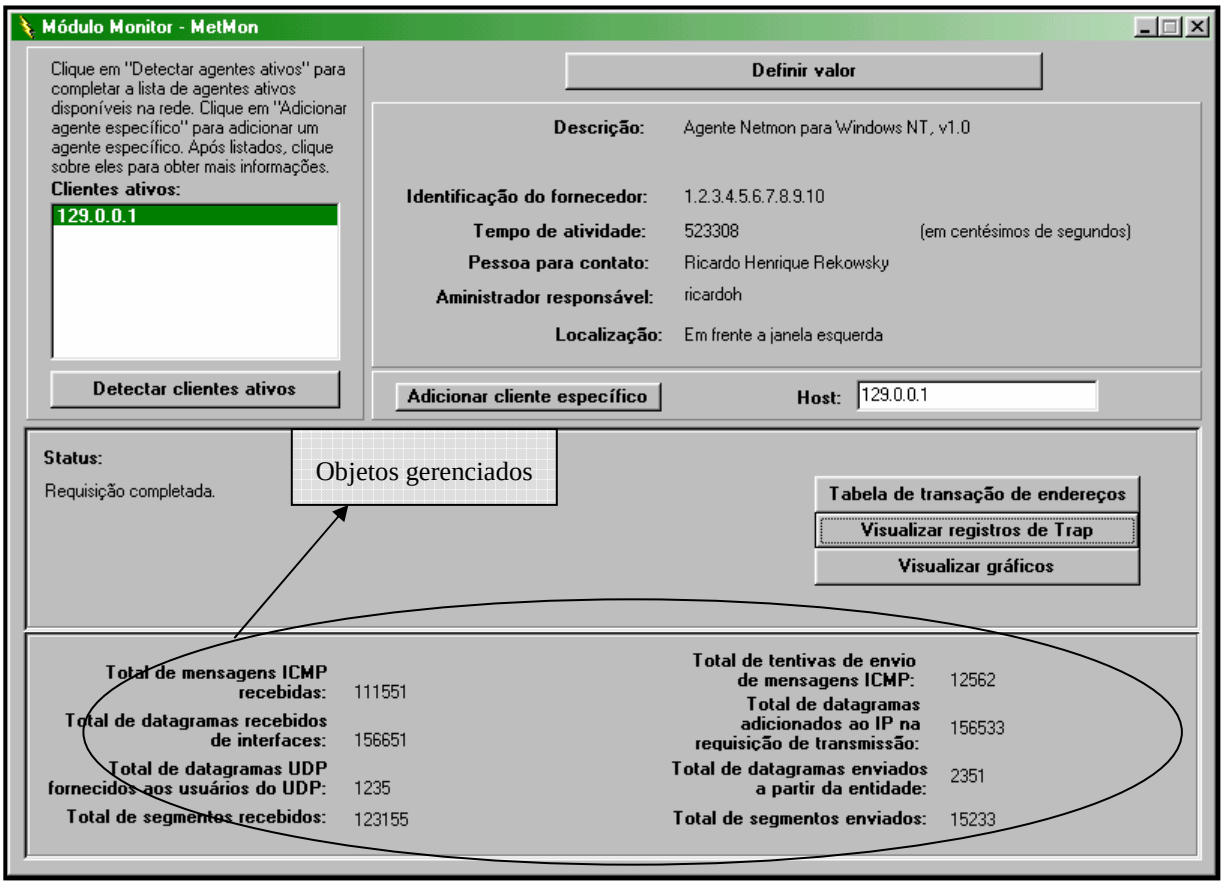
A visualização de gráficos é outra função que pode ser executada a partir do momento, em que um cliente é selecionado. Para que os gráficos possam ser vistos, é preciso pressionar o botão "*Visualizar gráficos*". Quando pressionado é mostrada uma nova janela com os gráficos disponíveis. A partir dos gráficos é possível visualizar, graficamente, como está o uso da rede, no tocante aos objetos gerenciados. Os gráficos, na sua forma inicial, são vistos conforme mostra a fig. 29.

FIGURA 29: Visualização gráfica.



Os valores dos objetos gerenciados situam-se na parte de baixo do módulo monitor. Este valores somente são atualizados ao selecionar um dos clientes ou, ainda, o mesmo cliente que já estiver selecionado. A atualização através da seleção do cliente desejado permite somente atualizar os dados, referentes ao cliente selecionado, quando desejado. Assim, evita-se uma sobrecarga da rede em busca de dados, quando muitos segmentos estão sendo monitorados. Os valores obtidos são correspondentes aos valores do cliente selecionado. A fig. 30 mostra a área de dados dos objetos gerenciados.

FIGURA 30: Objetos gerenciados.



A visualização de registros de *traps* somente pode ocorrer quando o botão "*Visualizar registros de trap*" estiver disponível. Para que este botão torne-se disponível é preciso que um dos cliente ativos disponíveis na rede e presente na lista de clientes ativos, envie um *trap* ao módulo monitor. Assim, o botão correspondente à esta função, poderá ser pressionado. Ao pressionar este botão surge uma nova tela, que mostra a mensagem *trap* enviada. A mensagem contém a hora de seu envio, qual módulo monitorado que enviou-a e, ainda, informações que foram compartilhadas ao módulo monitor. Os valores ilustrados neste protótipo são valores que, meramente, representam a funcionalidade do envio e recebimento do *trap*. A fig. 31 mostra a tela de visualização de registros da lista de *trap.*

FIGURA 31: Visualização de registros de *trap*.

6 CONCLUSÃO

Este TCC foi resultado de trabalho contínuo e disciplina, no tocante ao cronograma estabelecido na proposta e aceitação deste trabalho. Ainda, a pesquisa contínua e o questionamento junto ao professor orientador formaram os principais alicerces para a fundamentação e conclusão deste TCC.

A área de Gerenciamento de Redes é uma área em constante evolução, mas ainda não é uma área a qual está levantando-se valor no âmbito profissional e acadêmico. A importância desta área vem sendo negligenciada pelos profissionais do ramo, como foi possível de ser constatado ao longo da pesquisa por produtos, que incorporam a tecnologia utilizada ao longo deste TCC.

Ao longo do levantamento bibliográfico pode-se notar a ausência de material suficiente para pesquisa sobre o principal assunto abordado, o RMON. Os livros disponíveis auxiliaram na busca de informações genéricas sobre o assunto, tais como, fundamentos sobre as diversas áreas, o protocolo SNMP, definições e conceitos. Para obter-se informações mais detalhadas sobre os assuntos principais, o RMON e o protocolo SNMP fez-se necessária a busca de informações através de outros meios, como por exemplo, meios eletrônicos, *Internet*, outras instituições de ensino e pesquisa mediante conhecedores do assunto.

Para a implementação do protótipo proposto neste TCC, utilizou-se do componente *SNMP Tool* da empresa *Dart Communications* (*shareware*), que foi de fundamental importância para a concretização deste TCC, devido ao curto espaço de tempo disponível para a pesquisa, a fundamentação e a realização da implementação do protótipo. O componente utilizado oferece diversos recursos, como, por exemplo, a implementação do protocolo SNMP, que viabiliza a comunicação entre os módulos monitor e cliente, bem como a compilação de MIBs, contendo os objetos que devem ser gerenciados pelo módulo monitor. Sem a utilização deste componente auxiliar, não seria possível estabelecer a comunicação entre os módulos citados anteriormente, já que seria necessária a implementação do protocolo SNMP e outras funções auxiliares para concretização do protótipo. O estudo para entendimento também foi um fator relevante, já que sem este conhecimento, não seria capaz de manusear o componente em suas funções e entender como as funções e operações estavam sendo executadas. Existem, ainda, outros componentes que podem exercer a mesma

funcionalidade para a comunicação utilizando o protocolo SNMP, como por exemplo o *WinSnmp*.

A técnica utilizada, o RMON, mostra-se eficaz no tocante aos objetos referenciados. Através da especificação dos objetos na MIB, o RMON pode acessá-los através de sua identificação na árvore e após interpretação de seus conteúdos, disponibilizá-los por meio de gráficos ao administrador da rede. Todos os objetos que devem ser gerenciados devem ser especificados no momento de inicialização do módulo monitor. Os objetos escolhidos não podem ser alterados dinamicamente neste protótipo.

Atualmente, existem diversos produtos comercializados no mercado, que empregam a técnica RMON. Mas muitos deste produtos incorporam funções extremamente específicas de um fornecedor, ou quando genéricas, englobam o monitoramento de periféricos e objetos que não são de grande utilidade na maioria das corporações, devido ao nível tecnológico diferenciado.

O entendimento do usos de objetos para a implementação do protótipo foi um fator importante a ser levado em consideração, para que fosse possível observar o tráfego de informações na rede.

O protótipo desenvolvido apresenta a característica de monitorar apenas alguns objetos específicos para monitoramento do desempenho da rede e mostrar o uso do RMON. Este protótipo serve, também como resultado da pesquisa realizada durante o período do TCC.

6.1 SUGESTÃO DE CONTINUIDADE

Como complemento há a possibilidade de alteração dinâmica dos objetos gerenciados, a inclusão de outros objetos do RMON nas suas versões I e II, o monitoramento de *hubs*, roteadores, impressoras e demais periféricos, bem como gráficos adicionais, referentes a outros objetos gerenciados.

REFERÊNCIAS BIBLIOGRÁFICAS

[BRI93] BRISA, Sociedade Brasileira para Interconexão de Sistemas Abertos. **Gerenciamento de redes :** uma abordagem de sistemas abertos. São Paulo : Makron Books; Brasília, DF : TELEBRÁS, 1993..

[BRI94] BRISA, Sociedade Brasileira para Interconexão de Sistemas Abertos. **Arquitetura de redes de computadores OSI e TCP/IP.** São Paulo : Makron Books; Rio de Janeiro : EMBRATEL; Brasília, DF : SGA, 1994.

[CAM97] CAMPBELL, Patrick T. **Instalando redes em pequenas e médias empresas.** São Paulo : Makron Books, 1997.

[CAN98] CANTÚ, Marco. **Dominando o Delphi 3 :** a bíblia. São Paulo : Makron Books, 1998.

[CAR98] CARVALHO, Faical Farhat de. **Programação orientada à objetos usando Delphi 3.** São Paulo : Erica, 1998.

[COM98] COMER, Douglas E. **Interligação em rede com TCP/IP :** Princípios, protocolos e arquitetura. Rio de Janeiro : Campus, 1998.

[COR95] CORNEL, Gary. **Delphi :** segredos e soluções. São Paulo : Makron Books, 1995.

[EMB86] EMBRATEL. **Redes locais de computadores :** protocolos de alto nível e avaliação de desempenho. São Paulo : McGrawl-Hill, 1986.

[ENG97] ENGO, Frank. **Como programar em Delphi 3.** São Paulo : Makron Books, 1997.

[HAM86] HAMMOND, Joseph L.; O'REILY, Peter J.P.. **Performance analysis of local computer networks.** EUA : Addison-Wesley Publishing Company, 1986.

[HEL92] HELD, Gilbert. **Network management :** techniques, tools and systems. Lanarkshire : Wiley, 1992.

[KEE95] KEE, Eddie. **Redes de computadores ilustrada**. Rio de Janeiro : Axcel Books, 1995.

[LIM99] LIMA, Michele Mara de A. E. **Métricas para a Internet 1999.** Endereço eletrônico: http://www.rnp.br/newsgen/9805/metricas.shtml .

[MAF99] MAFINSKI, André. **Protótipo de software de gerência SNMP para o ambiente Windows NT.** Universidade Regional de Blumenau : Trabalho de Conclusão de Curso de Ciências da Computação - Bacharelado, 1999.

[MAR94] MARTIN, James et all. **Local area network :** architectures and implementations. EUA : Prentice Hall, 1994.

[MEI99] MEIRELES, L.F.T. **Modelo de referência SNMP 1999.** Endereço eletrônico: http://redes.ucpe.tche.br/documentos/snmp/ .

[MEN89] MENDES, Sueli. **Métodos Para Especificação de Sistemas**. São Paulo : Editora Edgard Blucher, 1989.

[MIC97] MICROSOFT. **Microsoft Windows NT Server 4.0 networking guide**. São Paulo : Makron Books, 1997.

[MOU86] MOURA, José Antão Beltrão et all. **Redes locais de computadores :** protocolos de alto nível e avaliação de desempenho. São Paulo : Mc-Graw-Hill, 1986.

[OLI98] OLIVEIRA, Adelize Generini de. **Tópicos avançados de programação em Delphi 3.** Florianópolis : Advanced, 1998.

[RAB99] RABINOVITCH, Eddie. **Network management performance** : tips and tools 1999. Endereço eletrônico: http://www.uniforum.org/web/pubs/uninews/970411/feature2.html .

[REI99] REISDORPH, Kent. **Aprenda em 21 Dias Delphi 4.** Rio de Janeiro : Campus, 1999.

[RFC1157] CASE, J. D.; FEDOR, M.; SCHOFSSTALL, M. L.; DAVIN, C. **Simple network management protocol (SNMP)**, Network Working Group, Request for Comments : 1157, maio 1990.

[RFC1213] McCLOGHRIE, K.; ROSE, M. **Management information base for network management of TCP/IP - based internets : MIB-II**, Network Working Group, Request for Comments: 1213, março 1991.

[RFC1271] WALDBUSSER, S. **Remote network monitoring management information base**. Network Working Group, Request for Comments: 1271, novembro 1991.

[RFC1513] WALDBUSSER, S. **Token Ring Extensions to the Remote Network Monitoring MIB**. Network Working Group, Request for Comments: 1513, setembro de 1993.

[RIG96] RIGNEY, Steve. **Planejamento e gerenciamento de redes**. Rio de Janeiro : Campus, 1996.

[RNT91] Revista nacional de Telemática, dezembro de 1991.

[SHI99] SHIRAISHI, Kazuhiro. **Conhecendo e trabalhando com Delphi 4.** São Paulo : Erica, 1999.

[SOA95] SOARES, Luiz Fernando Gomes. **Redes de computadores :** das LANs, MANs e WANs às redes de computadores. Rio de Janeiro : Campus, 1995.

[SZT96] SZTAJNBERG, Alexandre. **Gerenciamento de Redes**. Rio de Janeiro : Universidade Federal do Rio de Janeiro, 1996.

[TAN94] TANENBAUM, Andrew S.. **Redes de computadores.** Rio de Janeiro : Campus, 1997.

[THO97] THOMAS, Rosert M.. **Introdução às redes de computadores.** São Paulo : Makron Books, 1997.

[VEL86] VELLOSO, F. de Castro. **Informática :** uma introdução. Rio de Janeiro : Campus, 1986.

[WIE97] WIETHORN Jr, Rogério Antonio. **Protótipo de um software de gerência de redes baseado em SNMP para ambiente Netware**. Universidade Regional de Blumenau : Trabalho de Conclusão de Curso de Ciências da Computação - Bacharelado, 1997.

[ZAK88] ZAKIR Jr, José. **Redes locais - o estudo de seus elementos**. Rio de Janeiro : LTC - Livros Tecnológicos e Científicos, 1988.