UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE SOFTWARE DE GERÊNCIA SNMP PARA O AMBIENTE WINDOWS NT

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

ANDRÉ MAFINSKI

BLUMENAU, JUNHO / 1999.

1997/1 - 04

PROTÓTIPO DE SOFTWARE DE GERÊNCIA SNMP PARA O AMBIENTE WINDOWS NT

ANDRÉ MAFINSKI

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

_____

SÉRGIO STRINGARI - PROF. ORIENTADOR

_____

JOSÉ ROQUE VOLTOLINI DA SILVA - COORDENADOR

BANCA EXAMINADORA

_____

PROF. SÉRGIO STRINGARI

_____

PROF. ANTONIO CARLOS TAVARES

_____

PROF. MIGUEL A. WISINTAINE

AGRADECIMENTOS

Agradeço, primeiramente, as pessoas que de alguma forma me ajudaram a chegar ao fim desta importante etapa da minha vida.

Aos meus amigos e colegas, que sempre estiveram comigo, me apoiando e me dado coragem e determinação para sempre seguir em frente.

Ao meu professor e orientador Sérgio Stringari, que sempre demonstrou muito interesse e preocupação no meu trabalho, me motivando, até nas horas em que não via mais esperanças, a continuar tentando.

Aos meus irmão, e principalmente aos meus pais, que em toda a minha vida, me educaram, fizeram parte de todas as minhas conquistas, por mais simples que fossem. Fizeram-me aproveitar sempre as oportunidades, e nunca negaram esforços referentes a importância de jamais deixar de querer aprender.

E a Deus, por tudo.

RESUMO

Este trabalho apresenta um estudo sobre gerência de redes de computadores, enfatizando as considerações do protocolo SNMP (*Simple Network Management Protocol*). Descreve também a especificação e implementação de um protótipo de gerência de redes para o "sistema operacional" de rede Windows NT.

ABSTRACT

This paper introduces a study about local computer network, giving emphasis to the considerations about the network management, about the SNMP (*Simple Network Managment Protocol*) protocol concepts and foundations. Too relates the especification and implementation of a network managment prototype for the network operational system Windows NT.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

API - *Apllication Program Interfaces*

ASN.1 - *Abstract Sintax Notation.One*

CCITT - *Consultative Commitee for International Telegraph and Telephone*

CMIP - *Commom Management Information Protocol*

FTP - *File Transfer Protocol*

IETF - *Internet Engineering Task Forcel*

IP - *Internet Protocol*

ISO - *International Organization for Standardization*

LAN - *Local Area Network*

MAN - *Metropolitan Area Network*

MIB - *Management Information Base*

OID - *Object-Identifier*

OSI - *Open System Interconection*

PDU - *Protocol Data Unit*

RFC - *Request for Comment*

SNMP - *Simple Network Management Protocol*

TCP - *Transmission Control Protocol*

WAN - *Wide Area Network*

1 INTRODUÇÃO

Surgida há a apenas algumas décadas, as redes de computadores, tem ganho cada vez mais espaço em diversos ambientes, como empresas, indústrias e centros de educação e pesquisa, automatizando as mais variadas atividades e servindo como um eficiente meio de comunicação.

As redes de computadores são classificadas em: redes locais (LAN), redes metropolitanas (MAN) e redes geograficamente distribuídas (WAN).

As redes geograficamente distribuídas (WAN's - Wide Area Network) surgiram da necessidade de se compartilhar recursos especializados por uma grande comunidade de usuários geograficamente dispersos. Por estas redes possuírem um custo de comunicação bastante elevado, são geralmente públicas.

As redes metropolitanas (MAN's - Metropolitan Area Network) são redes consideradas integradoras de redes locais. Possuem características semelhantes às redes locais, porém geralmente cobrem distâncias maiores [SOA95].

As redes locais (LAN's - Local Area Network) são redes de computadores que permitem a interconexão de equipamentos numa distância entre 100m e 25km (atualmente), ou aos limites usuais de áreas consideradas privadas, como por exemplo um prédio ou escritório [ZAK88].

Com a tecnologia de redes de computadores descentralizou-se o acesso as informações, e as empresas, tanto de pequeno, médio ou grande porte optaram por utilizar esta tecnologia.

Um dos motivos que levou as empresas a optarem pelo uso de redes, foi a quantidade de computadores instalados, contendo as mesmas informações, as quais poderiam ser centralizadas num computador só. Outro motivo seria que, mais de um centro de computação interligado proporciona maior confiabilidade e segurança, devido ao fornecimento alternativo de processamento.

Segundo [RIG96], as facilidades providas pelas redes tendem a estimular o crescimento das mesmas, demandando necessidade de manutenção e planejamento futuro. Em um ambiente com poucos computadores conectados em rede, uma única pessoa é capaz de

gerenciar estes computadores. Mas considerando um ambiente onde a rede esteja distribuída entre vários locais a manutenção torna-se onerosa, consumindo tempo e recursos.

Gerenciamento de rede é o procedimento que consiste em controlar todos os componentes de *hardware* e *software* de uma rede de computadores [RIG96]. As atividades básicas do gerenciamento de redes consistem na detecção e correção de falhas, em um tempo mínimo, e no estabelecimento de procedimentos para a previsão de problemas futuros.

Monitorando as linhas cujo tráfego esteja aumentando ou os roteadores (dispositivo eletrônico que examina cada pacote de dados recebidos e, em seguida, decide como enviá-los a seu destino) que estão se sobrecarregando, é possível tomar medidas que evitem o "colapso" da rede, como a reconfiguração das rotas e troca do roteador por um modelo mais adequado.

A eficiência na realização destas tarefas está diretamente ligada a presença de ferramentas que as automatizem e de pessoal qualificado.

O gerenciamento de uma rede não é uma tarefa trivial e sua complexidade é diretamente proporcional ao tamanho da rede gerenciada.

Existem ferramentas para facilitar as tarefas de monitoração, análise e detecção de falhas, porém, associadas a elas deve existir uma estrutura que coordene as atividades de gerenciamento, assegurando sua eficiência e não permitindo que essas atividades se tornem mais um peso na tarefa de operação da rede.

1.1 OBJETIVOS

Caracterizam-se como objetivos deste trabalho, a pesquisa e o estudo das tecnologias de gerenciamento de redes de computadores, mais especificamente as redes locais afim de especificar e implementar um protótipo de software de gerenciamento de redes locais baseado no protocolo SNMP, em sistemas operacionais Windows NT .

1.2 ORGANIZAÇÃO DO TRABALHO

O capítulo 1 apresenta a estrutura geral do trabalho: introdução, os objetivos e a localização dos assuntos abordados.

O capítulo 2 trata especificamente do gerenciamento de redes: conceitos, características, modelos e protocolos de gerenciamento de redes.

O capítulo 3 é voltado ao desenvolvimento do protótipo, onde se apresenta a especificação e implementação do mesmo.

No capítulo 4 é dedicado às conclusões e sugestões de continuidade do trabalho.

E por fim, o capítulo 5 apresenta as bibliografias utilizadas para todo o desenvolvimento deste trabalho.

2 GERENCIAMENTO DE REDES

Segundo [BRI93] o gerenciamento de redes é o conjunto de funções que visa promover a produtividade da planta e dos recursos disponíveis a integrar, de forma organizada, as funções de operação, administração e manutenção de todos os elementos da rede e dos serviços de telecomunicações.

O gerenciamento de rede é o procedimento que consiste em controlar todos os componentes de hardware e software da rede [RIG96].

A gerência está associada ao controle de atividades e ao monitoramento do uso de recursos da rede. As tarefas básicas da gerência em redes, simplificadamente, são obter informações da rede, tratar estas informações, possibilitando um diagnóstico, e encaminhar as soluções dos problemas [SZT96].

Um sistema de gerenciamento é um dispositivo capaz de monitorar e configurar todos os elementos de uma rede, tanto de *hardware* quanto de *software*. Em tal sistema, cada elemento utiliza uma peça de *software*, denominada de *agente*, para comunicar suas informações de *status* e configuração ao *software* central de gerenciamento. Como os elementos de uma rede e os atributos dos dispositivos gerenciados se encaixam em um modelo hierárquico (estrutura de árvore), esta é a forma comum de organizar tais informações [STA94].

O contínuo crescimento em número, diversidade dos componentes das redes de computadores e a busca constante por uma maior produtividade, tem tornado cada vez mais necessária a atividade de gerenciamento de redes, incentivando assim, pesquisas mais aprofundadas nessa área.

2.1 CONSIDERAÇÕES SOBRE O GERENCIAMENTO DE REDES

As atividades básicas do gerenciamento de redes consiste na detecção e correção de falhas em um tempo mínimo e no estabelecimento de procedimentos para a previsão de problemas futuros.

Os sistemas de gerenciamento de redes apresentam a vantagem de ter um conjunto de ferramentas para análise e depuração da rede. Estes sistemas podem apresentar também uma série de mecanismos que facilitam a identificação, notificação e registro de problemas.

Os sistemas de gerenciamento são baseados na interação de diversas entidades, ou componentes da rede, como segue :

a) elementos de rede;

b) agentes;

c) objeto gerenciado;

d) base de informação de gerenciamento;

e) gerente;

f) protocolos de informação de gerenciamento;

g) interfaces de programas aplicativos;

h) interfaces com o usuário.

Elementos de Rede, também chamados de *Dispositivos Gerenciados*, são dispositivos de *hardware* como os computadores, roteadores e serviços de terminais que estão conectados a rede.

Agentes são módulos de *software* que residem nos elementos da rede. Eles coletam e armazenam informações de gerenciamento como o número de pacotes de erros recebidos pelo elemento de rede. São eles que respondem as solicitações dos agentes.

Objeto Gerenciado é qualquer elemento que possa ser gerenciado.

Base de Informação de Gerenciamento é uma coleção de objetos gerenciados. Contém dados recuperados pelos agentes, que são usados para enviar informações ao software de gerenciamento.

Gerentes são responsáveis pelo processamento dos dados sobre os dispositivos da rede e tornam estas informações disponíveis para um administrador de rede por meio de interfaces textuais, gráficas, etc.

Protocolos de Informação de Gerenciamento são usados para transportar informações entre agentes e os programas de gerenciamento.

Interfaces de Programas Aplicativos (API - *Apllication Program Interface*) são especificações que descrevem o modo como os programas aplicativos podem interagir com o sistema operacional.

Interfaces Com o Usuário é a maneira pela qual o aplicativo se apresenta para o usuário, ou seja, o *rosto* da aplicação.

Existem dois modelos básicos de gerenciamento de redes :

A especificação da ISO (*International Organization for Standardization*) nasceu através do desenvolvimento de uma arquitetura de gerenciamento de redes ISO/OSI (*Open Systems Interconection*), mais conhecida pelo seu protocolo de gerenciamento CMIP (*Common Management Information Protocol*).

O sistema de gerenciamento de redes da arquitetura Internet TCP/IP (*Transmission Control Protocol / Internet Protocol*), opera na camada de aplicações (camada na qual os dados são apresentados ao usuário, conforme modelo de referência OSI da arquitetura de redes de computadores) e baseia-se no protocolo SNMP (*Simple Network Management Protocol*), utilizado exclusivamente para a troca de informações de gerenciamento.

Esse sistema de gerenciamento segue o modelo *gerente-agente*. A tarefa do *agente* é responder as requisições feitas pelo *gerente* em relação ao equipamento no qual o *agente* está instalado, ou seja, ao *agente* cabe a coleta de informações de gerenciamento e ao *gerente* o processamento das mesmas.

Maiores informações sobre o gerenciamento de redes podem ser encontradas em [BRI93].

2.2 MIB - BASE DE INFORMAÇÕES DE GERENCIAMENTO

Todo sistema necessita armazenar as informações manipuladas em algum tipo de base de dados. A MIB (*Management Information Base*) é o nome conceitual para a informação de gerenciamento, incluindo os objetos gerenciados e seus atributos.

Segundo [WIE97] a MIB é o conjunto dos objetos gerenciados, que procura abranger todas as informações necessárias para a gerência de redes, possibilitando assim, a automatização de grande parte das tarefas de gerência.

Basicamente, são definidas quatro tipos de MIBs: MIB I, MIB II, MIB experimental e MIB privada. As MIBs do tipo I e II fornecem informações gerais sobre o equipamento gerenciado, sem levar em conta as características específicas deste equipamento. A MIB II, em verdade, é uma evolução da MIB I, que introduziu novas informações além daquelas encontradas na MIB I.

Portanto, através das MIBs do tipo I e II é possível obter informações como tipo e status de interface (Ethernet (padrão de *hardware*, de comunicação e cabeamento de rede local) , FDDI (padrão que utiliza cabos e fibra ótica), Token-Ring (rede local que utiliza a passagem de fichas para regular o tráfego na rede e evitar colisões)), número de pacotes transmitidos, número de pacotes com erro, informações de protocolos de transmissão etc.

As MIBs experimentais são aquelas que estão em fase de testes, com a perspectiva de serem adicionadas ao padrão e que, em geral, fornecem características mais específicas sobre a tecnologia dos meios de transmissão e equipamentos empregados.

As MIBs privadas são específicas dos equipamentos gerenciados, possibilitando que detalhes peculiares a um determinado equipamento possam ser obtidos. É desta forma que é possível se obter informações sobre colisões, configuração, e muitas outras, de um HUB. Também é possível fazer um teste, reinicializando ou desabilitando uma ou mais portas do HUB através de MIBs proprietárias [SZT96].

2.3 MODELO OSI DE GERENCIAMENTO

O gerenciamento no modelo OSI da ISO baseia-se na teoria da orientação a objetos. O sistema representa os recursos gerenciados através de entidades lógicas chamadas de objetos gerenciados. Ao desenvolver uma aplicação de gerenciamento, utilizam-se processos distribuídos conhecidos como gerentes (os quais gerenciam) e agentes (os que realizam ações) [SZT96].

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

Existem cinco áreas funcionais no gerenciamento OSI :

a) Gerência de configuração (estado da rede);

b) Gerência de desempenho (vazão e taxa de erros);

c) Gerência de falhas (comportamento anormal);

d) Gerência de contabilidade (consumo de recursos);

e) Gerência de segurança (acesso).

O modelo de gerenciamento OSI subdivide-se, em relação à sua estrutura, em :

a) Gerenciamento de sistemas;

b) Gerenciamento de camada;

c) Operações de camada.

O gerenciamento de sistemas exige funções de apoio em todas as sete camadas do modelo OSI (camada de aplicação, de apresentação, de enlace de dados, de rede, de sessão, de transporte e física).

O gerenciamento de camada é feito através de protocolos especiais, assim como de funções de apoio internas à camada, sendo independente dos protocolos de gerenciamento das outras camadas.

A operação de camada gerencia uma única instância de comunicação em uma camada através de protocolos normais de cada camada. É o que apresenta menores exigências com relação a funções de apoio [BRI93].

Com relação à MIB, pode-se dizer que ela guarda as informações transferidas ou modificadas pelo uso de protocolos de gerenciamento OSI. Tais informações podem ser fornecidas por agentes administrativos locais ou remotos. É importante ressaltar que 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 encontram-se num mesmo ambiente local. Para a troca de informações de gerenciamento entre sistemas remotos é definido um protocolo denominado CMIP (*Common Management Information Protocol*), que baseia-se em um modelo orientado à conexão.

Existem três tipos de hierarquias de gerenciamento usadas pelos Sistemas de Gerenciamento OSI: Herança, Nomeação e Registro.

a) Hierarquia de Herança;

A hierarquia de herança, também denominada de hierarquia de classe, está relacionada às propriedades associadas aos objetos descritas através de seus atributos, comportamento, pacotes condicionais, operações e notificações. Dentro desta hierarquia, define-se, então, o conceito de classes de objeto hierarquizadas às quais pertencem objetos com propriedades similares. Existem, então, superclasses às quais estão subordinadas subclasses.

b) Hierarquia de Nomeação;

A hierarquia de nomeação, também chamada de hierarquia de *containment*, descreve as relações entre instâncias de objetos com seus respectivos nomes.

Dentro desta hierarquia, define-se um relacionamento "estar contido em" aplicado aos objetos. Objetos de uma classe podem conter objetos da mesma classe e de classes diferentes. Um objeto gerenciado que contém outro objeto é dito superior; o objeto contido é dito subordinado. Um objeto gerenciado está contido dentro de um e somente um objeto gerenciado superior.

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

Isto implica que o objeto gerenciado existe somente se o seu objeto superior existir e que todo objeto gerenciado tem um nome que é derivado de um relacionamento de *containment.*

c) Hierarquia de Registro;

A hierarquia de registro, por sua vez, é usada para identificar de maneira universal os objetos, independentemente das hierarquias de heranças e nomeação. Esta hierarquia é especificada segundo as regras estabelecidas pela notação ASN.1 para árvore de registros usada na atribuição de identificadores a objetos. Cada nó desta árvore está associado a uma autoridade de registro que determina como são atribuídos os seus números. Desta maneira, cada objeto é identificado por uma seqüência de números, cada um correspondente a um nó.

2.4 MODELO DE GERENCIAMENTO DA INTERNET

As especificações do SNMP (*Simple Network Management Protocol*) foram publicadas pela primeira vez em 1988, pelo Departamento de Defesa dos Estados Unidos e por empresas que desenvolviam produtos baseados no TCP/IP na tentativa de desenvolver uma forma de gerenciamento das diferentes topologias de rede encontradas em sistemas complexos de redes interligadas. Desde, então, o SNMP evolui, transformando-se num protocolo de gerenciamento de redes, e em maio de 1990, o tornou-se um padrão dentro do protocolo TCP/IP [WIE97].

O SNMP é um protocolo da camada de aplicação designado para facilitar a troca de informações de gerenciamento entre dispositivos de rede. Usando os dados transportados pelo SNMP, os administradores de rede podem gerenciar mais facilmente a performance da rede, encontrar e solucionar problemas e planejar com mais precisão uma possível expansão da rede.

O protocolo SNMP é a solução adotada na Internet para permitir que gerentes de redes possam localizar e corrigir problemas. Nele são definidas tanto a sintaxe como o significado das mensagens trocadas entre os clientes e os servidores [SZT96].

O SNMP também define as relações administrativas entre os vários *gateways* que estão sendo gerenciados, determinando a autenticação necessária para os clientes acessarem os objetos gerenciados. Ele é baseado no paradigma conhecido como "busca-armazenamento" (*fetch-store*), isto é, todas as operações previstas para este protocolo são derivadas de operações básicas de busca e armazenamento.

A versão atual do protocolo SNMP é a 2.0 (SNMPv2). A principal diferença entre esta versão e a anterior é a existência de um mecanismo de comunidade melhorado e uma mudança na semântica, além de permitir o uso futuro de protocolos assimétricos de segurança, com o uso de chaves públicas.

Em sua primeira versão, o SNMP (SNMPv1) é apresentado através da três documentos que constituem a base para a estrutura de gerenciamento de rede padrão TCP/IP:

a) RFC 1155 - *Structure of Management Information;*

b) RFC 1157 - *Simple Network Management Protocol.*

c) RFC 1213 - *Management Information Base.*

O modelo de gerenciamento consiste em um esquema centralizado, isto é, uma estação (*host*) é configurada como gerente e os demais elementos da rede desempenham o papel de *agentes* ou *proxy agentes*. Um *proxy agente* serve de procurador para aqueles equipamentos que não implementam SNMP. Cada agente possui uma MIB que contem as variáveis relativas aos objetos gerenciados [BRI94]. O modelo genérico compreende três componentes :

a) um conjunto de objetos gerenciados, correspondente a um agente e a uma MIB associada;

b) uma estação de gerenciamento de rede;

c) um protocolo de gerenciamento de rede que é usado pela estação gerente e pelos agentes na troca de informações de gerenciamento.

Um objeto gerenciado representa um recurso que pode ser classificado na categoria de sistema hospedeiro (estação de trabalho, servidor de terminal ou impressora), sistema *gateway* (roteadores) ou equipamentos de meio (*modem, bridge, hub* ou multiplexador).

A estação gerente corresponde a um sistema hospedeiro que executa as aplicações de gerenciamento e o protocolo de gerenciamento de rede, tomando as decisões de acordo com as informações obtidas junto ao agente.

O protocolo de gerenciamento é visto sob o paradigma de observação remota, isto é, ela não transporta simplesmente operações de gerenciamento que devem ser executadas pelos objetos gerenciados, cada objeto é visto como uma coleção de variáveis cujo valor pode ser lido ou alterado, possibilitando, assim, a monitoração e o controle da cada elemento da rede.

Segundo [BRI93] o SNMP tem como base a técnica "*fetch-store*", ou seja, todas as suas operações previstas são derivadas de operações básicas de busca e armazenamento.

Um gerente interage com um agente de acordo com as regras estabelecidas pelo *framework* de gerenciamento. Em geral, o gerenciamento da rede impõe *overheads* significativos, pois cada nó apenas produz algumas variáveis que serão lidas e usadas para sua monitoração.

As mensagens deste protocolo não possuem campos fixos e são especificadas na notação ASN.1 (*Abstract Syntaxe Notation.* 1). Elas consistem em três partes principais:

versão de protocolo, identificador da comunidade SNMP e área de dados. Para cada uma das operações mencionadas anteriormente, é definido um tipo específico de mensagem de protocolo, isto é um tipo de PDU (*Protocol Data Unit*) [BRI93].

2.4.1 OPERAÇÕES DO PROTOCOLO SNMP

O que deve ser feito com os objetos num ambiente de gerenciamento, é definido através das operações aplicadas nos objetos, que são enviadas ao servidor pelo cliente. Duas operações (comandos) básicas no protocolo SNMP, são:

A operação SET é usada por um cliente para alterar um ou mais atributos de um objeto gerenciado (*set-request*);

A operação GET é usada por um cliente para obter o valor(es) de um ou mais atributos de um objeto gerenciado (*get-request* para o pedido e *get-response* para obter o retorno deste pedido).

Uma operação GET ou SET somente se refere a uma única instância de um objeto representada através de seu nome. No protocolo SNMP, as operações são atômicas, isto é, todas as operações de um pedido devem ser executadas. Não existem execuções parciais de um pedido (no caso, operações aplicadas a múltiplos objetos). Se ocorrer algum erro durante a execução de uma operação, os resultados produzidos por esta operação devem ser ignorados [SZT96].

Antes de executar um pedido, o servidor deve mapear apropriadamente os nomes dos objetos codificados em ASN.1(*Abstract Sintax Notation.One*) nos objetos internos que armazenam as características das entidades da rede (através dos atributos do objeto).

Todas as operações do SNMP consistem em um cabeçalho de autenticação e na unidade de dados de protocolo propriamente dita. Esse cabeçalho de autenticação inclui, por exemplo, número da versão e informações de controle de acesso, e é usado pelo agente para determinar se o gerente está ou não autorizado a operação especificada [BRI94].

Segundo [BRI94], as operações disponíveis no protocolo SNMP são:

a) GET-REQUEST - obter atributos de um objeto gerenciado; utilizada para recuperar uma informação de gerenciamento específica;

b) GET-NEXT-REQUEST - obter os atributos sem conhecer o nome exato do objeto; usada quando não se tem conhecimento das variáveis que se deseja recuperar; a operação devolve o próximo valor armazenado, através de esquadrinhamento;

c) GET-RESPONSE - resposta de uma operação de busca de atributos de um objeto; responde a uma operação de GET-REQUEST, GET-NEXT-REQUEST, ou SET-REQUEST; contém os valores dos objetos em questão ou uma mensagem de erro, indicando a falha de operação;

d) SET REQUEST - alterar os atributos de um objeto gerenciado; usada para atribuir valores às variáveis que contêm informações de gerenciamento;

e) TRAP - resposta acionada devido a ocorrência de um evento; usada para relatar a ocorrência de eventos extraordinários.

Essas 5 operações estão presentes tanto no SNMPv1 quanto no SNMPv2; para o SNMPv2, existe outras duas operações que oferecem maior flexibilidade na recuperação da informação de gerenciamento e na distribuição das atividades de gerenciamento [BRI94] :

a) GET-BULK-REQUEST - permite a recuperação da informação de múltiplos valores através de uma única troca de PDU, cobrindo uma deficiência do SNMPv1 na recuperação de grandes blocos de informações de gerenciamento;

b) INFORM-REQUEST - possibilita que um gerente encaminhe informações de gerenciamento para outro gerente, suportando, assim, em esquema distribuído de gerenciamento de rede.

Numa operação GET-NEXT o nome do objeto não só especifica o objeto a acessar (para obter seus atributos, como na operação GET normal), como também é utilizado para descobrir qual o próximo objeto na seqüência léxica. Como retorno, a operação informa o nome do próximo objeto na hierarquia da MIB, e os valores dos seus atributos (obtidos através da execução de uma operação GET normal sobre o objeto).

Uma TRAP que é usada para informar a ocorrência de eventos, permitindo aos servidores SNMP enviarem informações aos clientes sempre que ocorrer algum evento que informa a ocorrência de alterações nos objetos (no protocolo, foram definidas somente algumas *traps*).

A operação GET-NEXT é útil para obter os atributos dos objetos de uma tabela de tamanho desconhecido, pois um cliente pode enviar continuamente requisições GET-NEXT a um servidor que se encarregará de enviar os atributos do objeto e o nome do próximo objeto. Cada novo pedido deve especificar o nome do objeto retornado pelo pedido anterior, o que permite varrermos a tabela sem saber qual o próximo objeto desta tabela. Este processo é denominado de caminhamento na tabela. Devido ao ASN.1 não apresentar nenhum mecanismo para implementar tabelas ou para indexá-las, denota-se os elementos individuais (objetos) de uma tabela através de um sufixo [SZT96] .

Para facilitar o uso do comando GET-NEXT em tabelas, alguns nomes de objetos na MIB correspondem tabelas completas ao invés de objetos individuais, não podendo ser utilizados em uma operação GET (pois esta falhará), mas podem ser usados como parâmetro para a operação GET-NEXT, indicando o primeiro objeto da tabela. Não será necessário conhecer o nome do próximo objeto, pois cada comando GET-NEXT retornará o nome do próximo item da tabela. Executando este processo sucessivamente até que todos os itens da tabela tenham sido acessados, teremos varrido toda a tabela.

A implementação de uma estrutura de dados que suporte o comando GET-NEXT pode ser complicada devido a esta operação poder pular o próximo objeto simples (na ordem lexicográfica) devido a existência de objetos vazios. Como conseqüência, não se pode utilizar simplesmente a ordem lexicográfica presente na árvore para determinar quais objetos satisfazem a um comando GET-NEXT, devendo também existir um programa que examine os objetos, ignore aqueles objetos que estejam vazios e descubra o primeiro objeto simples pertencente a um objeto não vazio.

Para o suporte das funções GET, SET e GET-NEXT sobre tabelas, ao contrário do que acontece com objetos simples mapeados em memória, é necessário um software adicional para mapear a tabela numa estrutura interna de dados. No caso das tabelas MIB, o servidor SNMP deve providenciar algum mecanismo que permita a cada tabela ter três funções para implementar as operações GET, SET e GET-NEXT. Para o servidor descobrir qual função deve ser utilizada, o software que implementa o servidor deve usar a tabela para escolher a função correta, através do uso de um ponteiro para uma tabela que conterá ponteiros para cada uma das operações [SZT96].

As entradas em uma tabela apontam para outras tabelas que não contém o identificador completo do objeto, mas somente o prefixo deste identificador, porque o identificador completo do objeto para um item da tabela é formado pelo prefixo que identifica a tabela, mais um sulfixo que identifica uma entrada particular na tabela em que o objeto está armazenado.

Uma vez determinado o prefixo correspondente ao objeto e formado o nome do objeto, a função de acesso correspondente a operação pedida é invocada. No caso das tabelas, a função de acesso obtém o sufixo do identificador do objeto, e o utiliza para selecionar uma das entrada da tabela. Para a maioria das tabelas, é usado o endereço IP para selecionar uma entrada. O endereço IP é codificado no identificador do objeto utilizando-se a representação decimal *point*.

2.4.2 MIB DA INTERNET

A MIB da Internet define os objetos que podem ser gerenciados por camada do protocolo TCP/IP. Estes objetos estão sobre a guarda de um *agente* de gerenciamento e a comunicação entre este *agente* e um *gerente*.

Como todos os padrões da tecnologia TCP/IP, as definições utilizadas no gerenciamento SNMP foram publicadas na série RFC (*Requests For Comments*). As definições originais do protocolo SNMP, bem como dos objetos gerenciados foram publicados em 1989. Em 1990, foi realizada uma revisão da MIB, que passou a se chamar de MIB-II. Em 1993, foi publicado um conjunto de padrões novos, chamado SNMPv2, com alterações do protocolo e extensões às definições dos objetos.

Cada objeto em uma MIB é identificado por um único rótulo universalmente conhecido como OID (*Object-Identifier*). O espaço do nome do objeto é implementado como um esquema de nomeação hierárquico de múltiplas partes. Um esquema de nomeação hierárquico pode ser visualizado como uma árvore invertida com os galhos apontados para baixo. Cada ponto onde um novo galho é adicionado é referenciado como um nó. Esse OID é internacionalmente aceito e permite que os desenvolvedores e os fabricantes criem novos componentes e recursos e atribuam um único OID para cada novo componente ou recurso [MIC97].

O esquema de nomeação OID é administrado pela IETF (*Internet Enginneering Task Force*I). O IETF concede autorização para partes do espaço do nome para organizações individuais, tais como a *Microsoft*. A *Microsoft*, por exemplo, tem autorização para atribuir os OIDs que podem ser derivados para ramificações descendentes a partir do nó da árvore de nomes MIB que começa em 1.3.6.1.4.1.311.

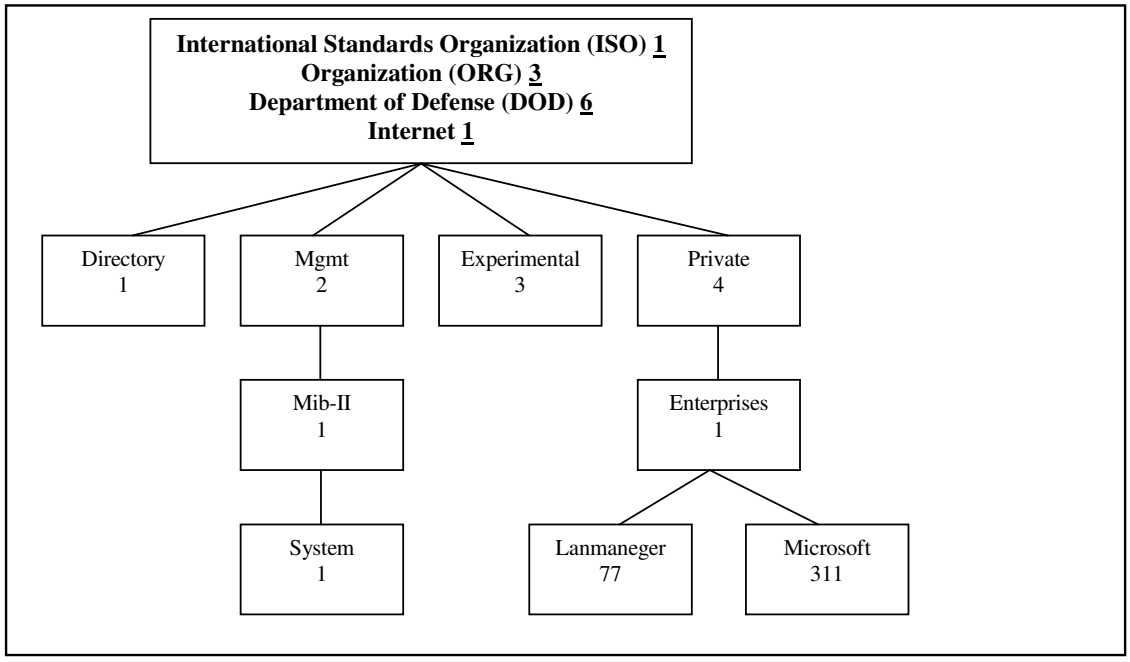


Figura 1 : **Árvore da MIB-II**

Os programas SNMP utilizam o OID para identificar os objetos em cada computador que pode ser gerenciado com o SNMP. Por exemplo, quando um administrador de rede requer informações sobre objetos monitorados a partir de algum computador da rede, o programa de gerenciamento SNMP envia uma mensagem pela rede solicitando informações sobre o objeto como é identificado pelo OID. O computador que recebe a mensagem pode usar o OID para recuperar as informações do objeto específico no computador e devolvê-las para o programa de gerenciamento do SNMP [MIC97].

A MIB divide os objetos em vários grupos. A tabela seguir mostra a MIB-I e os grupos nela definidos.

Grupo	Objetos para	Nro de Objetos
system	informações básicas do sistema	3
interfaces	interfaces de rede	22
at	tradução de endereço	3
ip	software de protocolo IP	33
icmp	protocolo de estatíst	26
tcp	software de protocolo TCP	17
udp	software de protocolo UDP	4
egp	software de protocolo EGP	6

Tabela 1 : **Número de objetos nos grupos da MIB-I**

A tabela a seguir mostra a MIB-II e os grupos nela definidos.

Grupo	Objetos para	Nro de Objetos
system	informações básicas do sistema	7
interfaces	interfaces de rede	23
at	tradução de endereço	3
ip	software de protocolo IP	38
icmp	protocolo de estat. para contr	26
tcp	software de protocolo TCP	19
udp	software de protocolo UDP	7
egp	software de protocolo EGP	18
transmiss	transmissão. Média-específica	0
snmp	aplicações snmp	30

Tabela 2 : **Número de objetos nos grupos da MIB-II**

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

A seguir, são listados alguns exemplos de objetos de alguns grupos:

a) Grupo System

- sysDescr : completa descrição do sistema (versão, hardware, sistema operacional)
- sysObjectID : objeto para identificação do vendedor

- sysUpTime : tempo desde a última reinicialização
- sysContact : nome da pessoa de contato
- sysServices : serviços oferecidos pelo dispositivo

b) Grupo IP

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

c) Grupo TCP

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

d) Grupo UDP

- udpInDatagrams : número de datagramas UDP entregues aos usuário UDP
- udpNoPorts : número de datagramas UDP recebidos para aqueles onde não existe aplicação para aquela porta de destino

- udpInErrors : número de datagramas UDP recebidos que não podem ser entregues por diversas razões, menos a falta de uma aplicação para a porta de destino
- udpOutDatagrams : número de datagrama UDP enviados por esta entidade

e) Grupo Interfaces

- ifIndex : número da interface
- ifDescr : descrição da interface
- if Type : tipo da interface
- ifMtu : tamanho do maior datagrama IP
- ifAdminisStatus : status da interface
- ifLastChange : hora em que a interface inicializou o status corrente

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

Além disso, na Internet não são previstos mecanismos para definir ações e eventos associados a um objeto gerenciado [SZT96].

3 DESENVOLVIMENTO DO PROTÓTIPO

O protótipo desenvolvido neste trabalho visa validar a função de gerente, no conceito de gerenciamento de redes, que interage com agentes SNMP do ambiente Windows NT, com uma interface gráfica *Microsoft Windows*.

Neste capítulo são apresentadas as técnicas e ferramentas utilizadas, tanto para a especificação quanto para a implementação do protótipo proposto.

Tanto a especificação quanto a implementação do protótipo, foram baseadas no uso da biblioteca WinSnmp, desenvolvida pela empresa *MG-SOFT Corporation* , que implementa as funções necessárias para a criação de softwares de gerência de redes, sendo esta comentada no decorrer deste capítulo.

3.1 ESPECIFICAÇÃO DO PROTÓTIPO

Para especificação foram consideradas as seguintes tecnologias:

a) linguagens gráficas livres;

b) sistema operacional de redes WindowsNT;

c) biblioteca *WinSnmp*, que implementa funções SNMP.

3.1.1 LINGUAGENS GRÁFICAS LIVRES

Na representação do protótipo foram consideradas as linguagens gráficas livres, utilizadas para especificação de processos através de fluxogramas.

As linguagens gráficas utilizam uma simbologia gráfica acompanhada de cadeias de símbolos alfanuméricos ou especiais. Têm a vantagem de permitir que um grande número de informações sejam assimiladas rapidamente.

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

3.1.2 SISTEMA OPERACIONAL DE REDE WINDOWS NT

O serviço SNMP que funciona sob o Windows NT implementa a versão 1 do SNMP e fornece um agente SNMP que permite o gerenciamento SNMP remoto e centralizado de:

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 e pode ser instalado nos computadores baseados no Windows NT que usam os protocolos TCP/IP e IPX. O protocolo TCP/IP deve ser instalado antes da instalação do SNMP.

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. As MIBs adicionais da *Microsoft* para o DHCP, WINS e o Internet Information Server estendem o gerenciamento SNMP a esses serviços baseados em Windows NT. Os programas agentes que implementam essas MIBs adicionais são chamados de agentes de extensão.

O diagrama a seguir mostra uma interação simples entre um computador gerenciador SNMP e um computador baseado no Windows NT com um programa agente do SNMP.

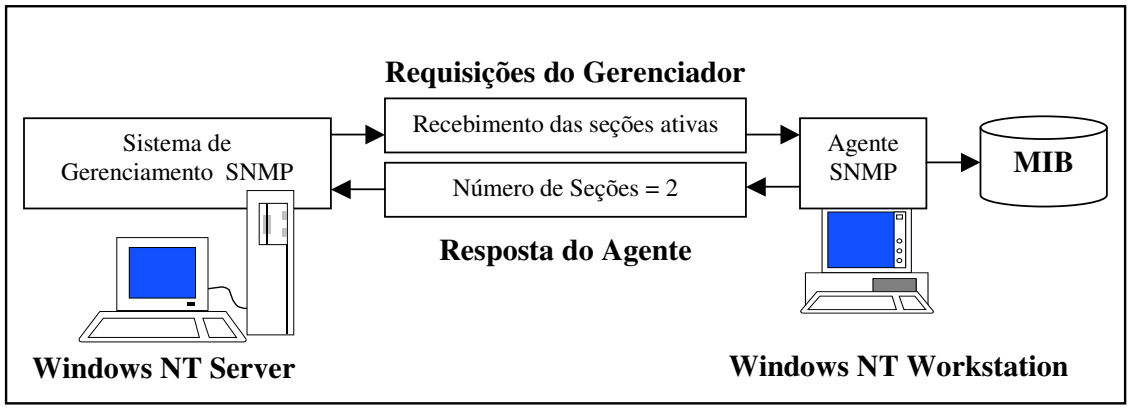


Figura 2 : **Interação do gerenciador e do agente SNMP.**

A tabela a seguir descreve alguns arquivos instalados em um computador Windows NT com o serviço SNMP instalado.

Nome do Arquivo	Descrição
Inetmib1.dll	MIB-II DLL
Mgmtapi.dll	Um gerenciador SNMP API baseado em Windows NT que recebe solicitações de gerenciamento e as envia, para e recebe respostas de agentes SNMP
Mib.bin	Instalada com o serviço SNMP e usada pelo Gerenciamento API. Mgmtapi.dll, para mapear nomes de objetos baseados em textos para OIDs numéricas
Snmp.exe	O agente de Serviço SNMP é um agente que aceita requisições de um programa de gerenciamento e envia as solicitações para DLL apropriada para o processamento.
Snmptrap.exe	Recebe uma mensagem SNMP de um agente SNMP e envia para um gerenciador SNMP API no console de gerenciamento

Tabela 3 : **Arquivos do serviço SNMP**

Informações mais detalhadas sobre o ambiente Windows NT e o serviço SNMP encontram-se em [MIC97].

3.1.3 BIBLIOTECA WINSNMP

WinSNMP é uma proposta que define uma interface de programação para aplicações de gerência de redes sobre ambiente Microsoft Windows. A interface suporta tanto SNMPv1 quanto SNMPv2. SNMPv1 é visto no WinSNMP como um subconjunto de SNMPv2.

WinSNMP é definido para estimular o desenvolvimento de aplicações que gerenciem objetos em uma rede. Como esperado em extensões de API em ambiente Windows, WinSNMP implementa as funções da interface em uma DLL. Por sua vez a DLL utiliza a interface padrão winsock para realizar suas tarefas.

WinSNMP foi desenhado tendo em vista a construção de aplicações de gerenciamento de redes. Com tal interface desenvolvedores são capazes de criar gerentes para os objetos da

rede que rodem sobre ambiente Microsoft Windows. O modelo de programação WinSNMP entretanto não serve para o desenvolvimento de agentes SNMP. O objetivo principal é a criação dos gerentes, sem dúvida nenhuma, uma carência do ambiente Windows, já que este nos permite apenas criar agentes.

Entre as principais vantagens de se escrever um gerente sobre WinSNMP temos:

a) WinSNMP "esconde" os detalhes de ASN.1, BER e o protocolo SNMP;

b) WinSNMP fornece independência de fornecedor. Como as funções são implementadas em uma DLL e a interface é padrão, as aplicações rodam em qualquer sistema operacional independentemente do fornecedor da DLL;

c) Suporte uniforme a SNMPv1 e SNMPv2. Uma aplicação WinSNMP não precisa saber a versão do SNMP das entidades alvo (agentes). A implementação do WinSNMP fará todos os mapeamentos necessários entre SNMPv1 e SNMPv2, de acordo com os RFCs apropriados.

Os fornecedores da DLL que implementam as funções do WinSNMP devem indicar quão ampla é sua implementação. Tal implementação pode ocorrer em quatro níveis distintos:

a) Nível 0 - funções para codificação e decodificação de mensagens apenas;

b) Nível 1: funções do nível 0 mais funções de interação com agentes SNMPv1;

c) Nível 2: funções do nível 1 mais funções de interação com agentes SNMPv2;

d) Nível 3: funções do nível 2 mais funções de interação com outros gerentes SNMPv2 ;

Maiores informações sobre a biblioteca WinSnmp pode ser encontradas em [GRA96].

3.1.4 APRESENTAÇÃO DA ESPECIFICAÇÃO

O modelo de programação windows é orientado ao processamento de eventos sinalizados à aplicação, através de mensagens. Por esta razão, entre outras, este é o modelo de programação odotado pelo WinSNMP.

Neste modelo as aplicações tipicamente devem responder a vários tipos de eventos. Dentro dessa filosofia, WinSNMP possui três funções cruciais:

a) SnmpSendMsg;

b) SnmpRecvMsg;

c) SnmpRegister .

Destas, logicamente SnmpRecvMsg é a que tem maior impacto no modelo de programação. Assim, o modelo básico de programação WinSNMP, possui os seguintes passos:

a) a aplicação abre uma sessão com a função SnmpOpen;

b) se a aplicação estiver interessada em tratar traps, ele indica isso através da função SnmpRegister;

c) a aplicação prepara uma ou mais PDUs para transmissão e processamento para implementação WinSNMP através de mensagens WinSNMP. Isso é feito através da função SnmpCreatePdu;

d) a aplicação submete uma ou mais requisições consistindo de uma PDU, através do comando SnmpSendMsg;

e) a aplicação recebe notificações que uma resposta a uma requisição está disponível ou que uma trap registrada ocorreu. A notificação é feita através do "canal" de notificação informado na função SnmpOpen;

f) a aplicação recupera a resposta através de SnmpRecvMsg;

g) a aplicação processa a resposta através da lógica específica da aplicação;

h) a aplicação fecha a sessão WinSNMP através de SnmpClose.

A figura 3 apresenta uma especificação genérica do protótipo:

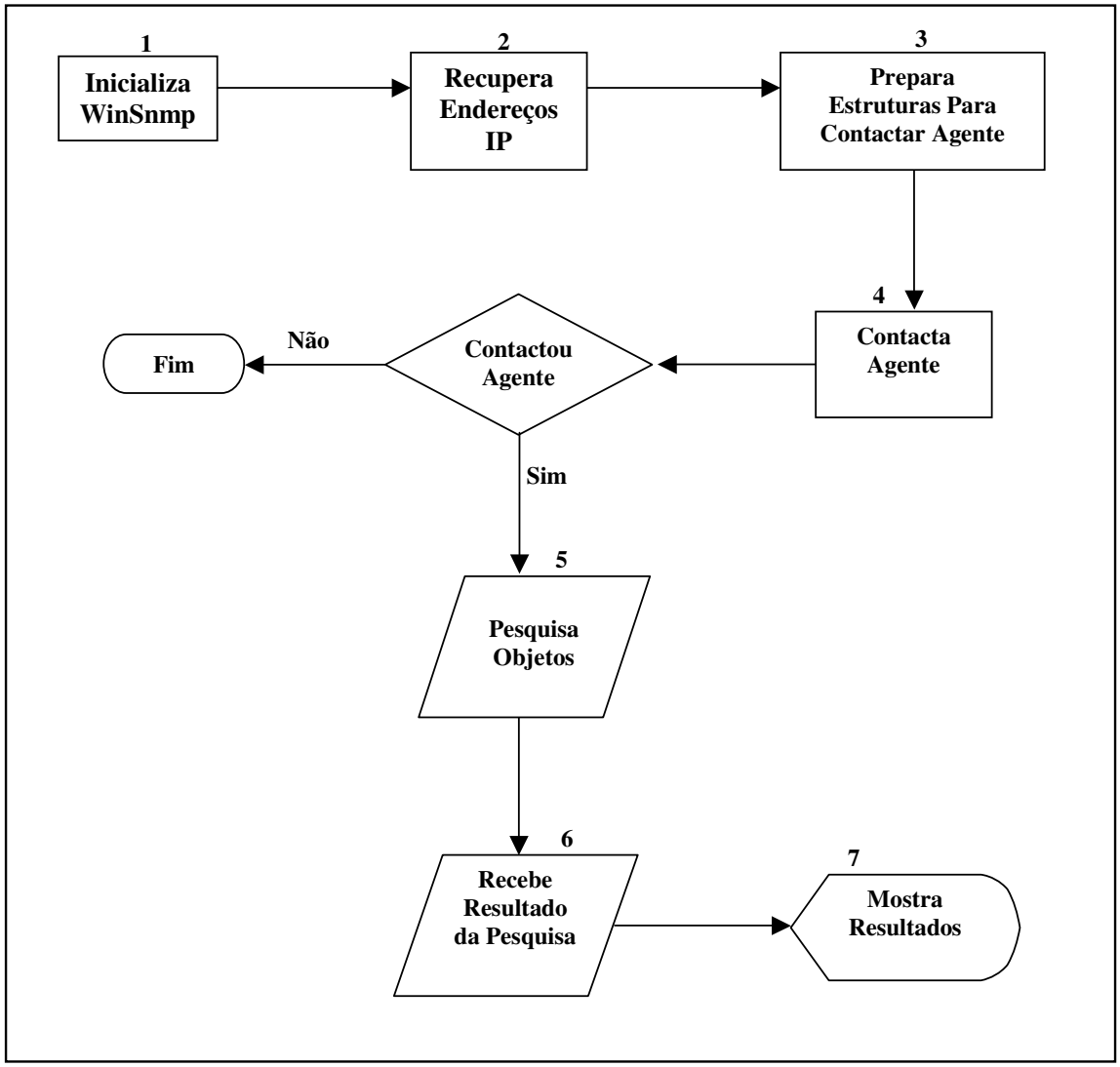


Figura 3: **Representação gráfica do protótipo**

Processo 1 : É criado um vínculo entre a aplicação e a biblioteca WinSnmp.

Processo 2 : Aplicação obtêm tanto o endereço IP local (gerente) quanto o endereço IP remoto (agente).

Processo 3 : Aplicação cria entidades necessárias (agente, gerente) e carrega dados como comunidade, valor de *time_out* e retransmissão.

Processo 4 : Aplicação contacta agente. Nesta etapa um objeto já é referenciado para que se possa testar se o agente foi contatado ou não.

Processo 5 : A aplicação recebe a resposta de uma requisição.

Processo 6 : Aplicação carrega o resultado da resposta do processo 5.

Processo 7 : Aplicação processa os resultados da pesquisa e mostra ao usuário;

Uma especificação mais detalhada é apresentada nas figuras 4, 5, 6, 7, 8 e 9:

a) Inicializa WinSnmp;

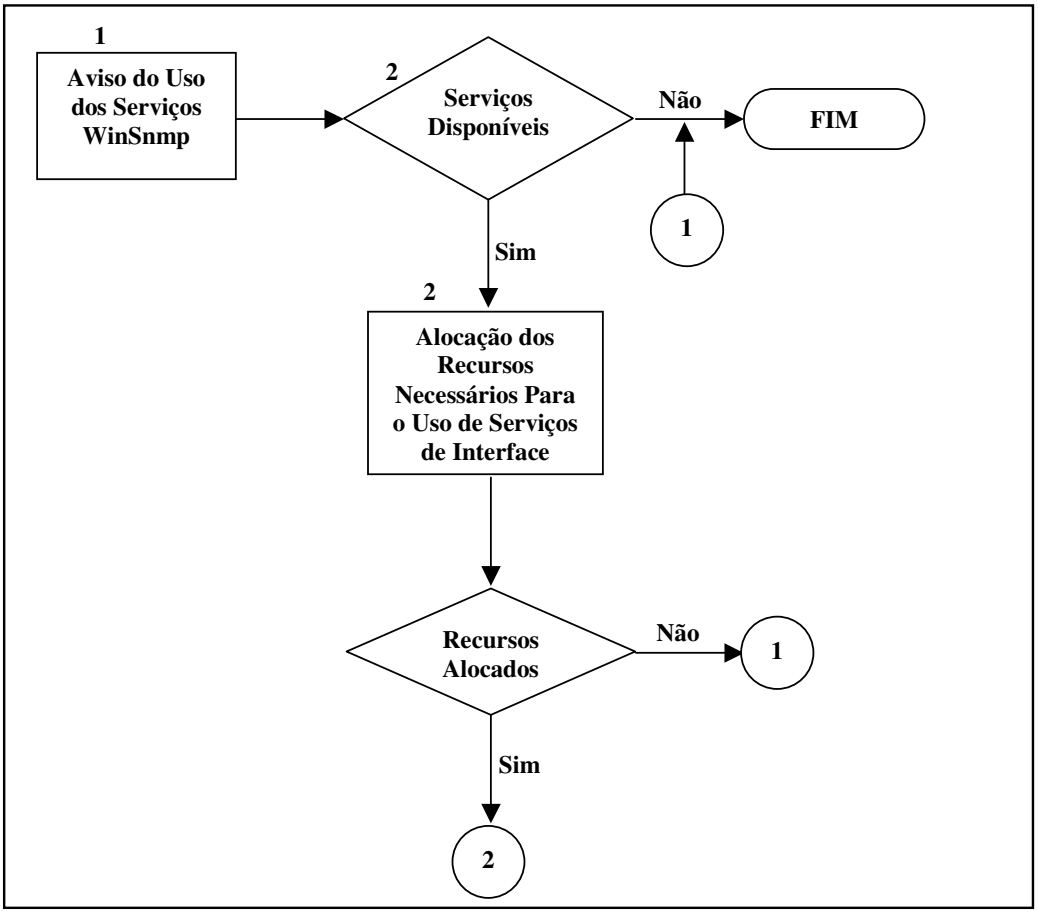


Figura 4: **Processo Inicializa WinSnmp.**

Processo 1: Aplicação avisa a biblioteca de funções que usará o seus serviços SNMP, abilitando a implementação a realizar quaisquer procedimentos de inicialização e alocação de recursos e retornar alguma informação útil a aplicação. Isto é feito através da função **SnmpStartup( ).**

Processo 2: Aplicação aloca e inicializa a memória, recursos, mecanismos de comunicação e estruturas de dados necessários para que a aplicação possa usar os serviços da interface. Isto é feito através da função **SnmpOpen( )**, que retorna a identificação da sessão criada e continuará sendo usada pela aplicação para a chamada a outras funções.

b) Recupera endereços IP;

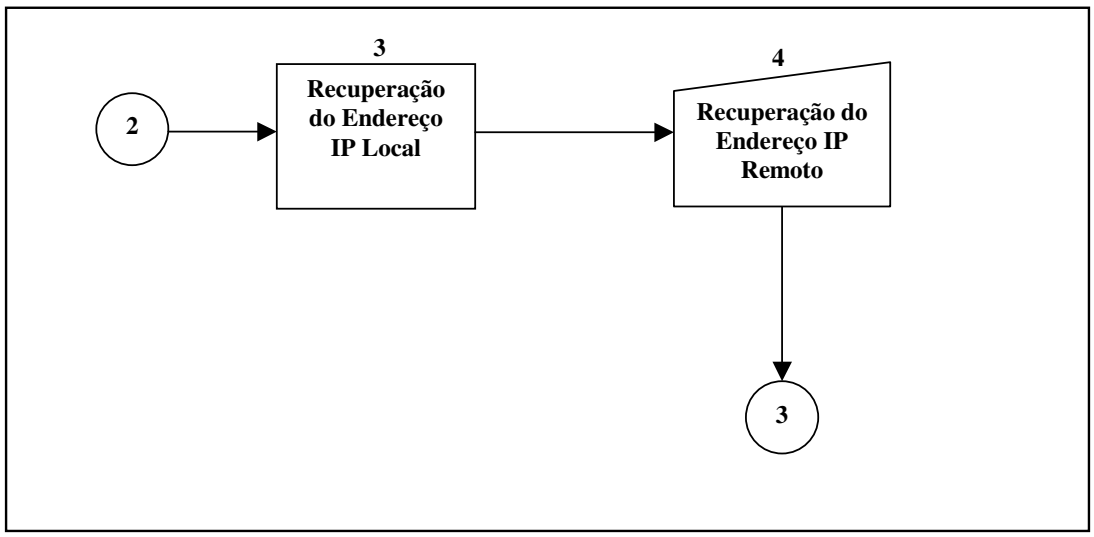
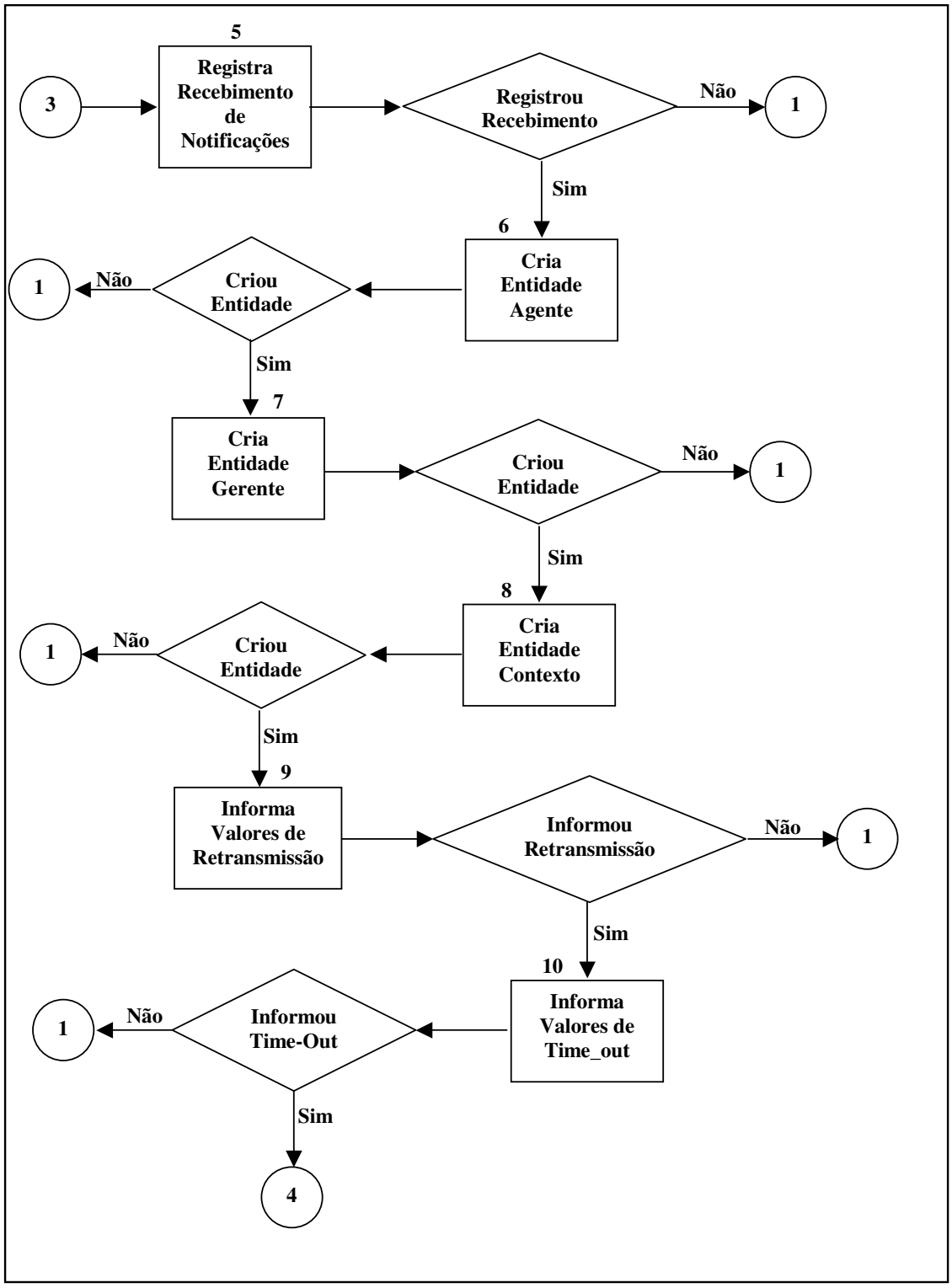


Figura 5: **Processo Recupera Endereços IP.**

Processo 3 : Aplicação recupera o endereço IP do Host local (gerente) através da função **GetHostName( )** contida no ambiente de desenvolvimento *Borland Delphi*

Processo 4 : Usuário informa à aplicação o endereço IP do Host remoto (agente), que irá ser contatado.

c) Prepara estrutura para contatar agente;



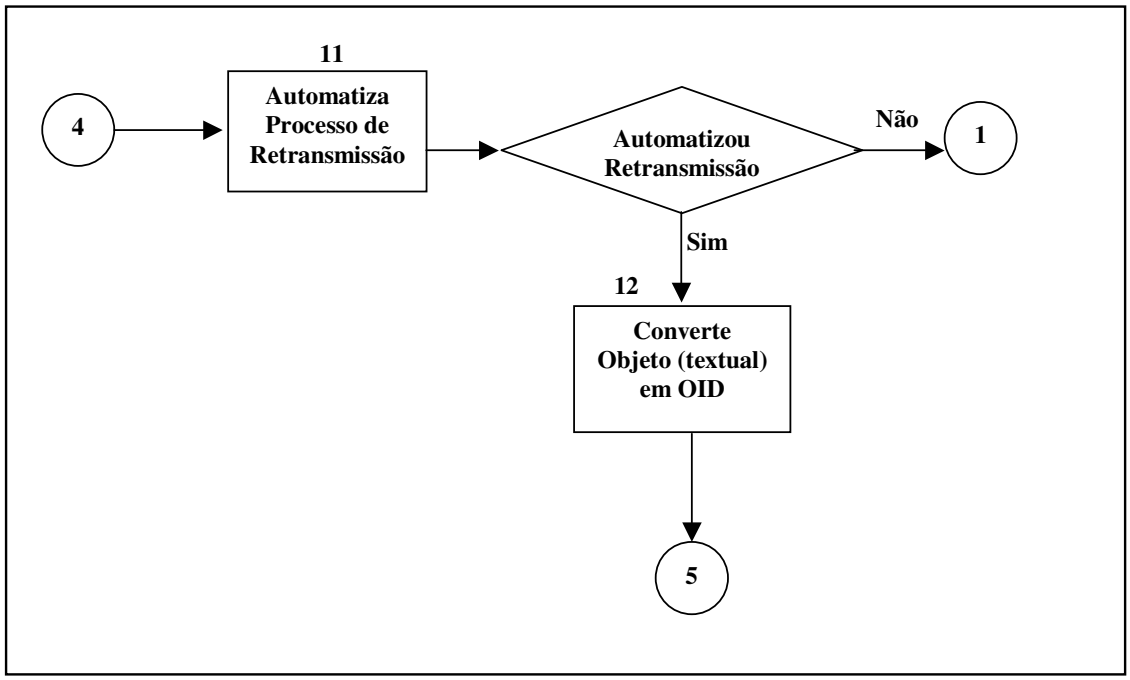


Figura 6 : **Processo Prepara Estrutura Para Contatar Agente**

Processo 5 : A aplicação registra o uso de traps, através da função **SnmpRegister()**.

Processo 6 : Neste processo a aplicação cria a entidade agente. Esta recebe todos os dados que contêm a sessão criada anteriormente, e informa o IP que identifica o agente. Isto é feito através da função **SnmpStrToEntity( ).**

Processo 7 : Neste processo a aplicação cria a entidade gerente. Esta recebe todos os dados que contêm a sessão criada anteriormente, e informa o IP que identifica o agente. Isto é feito através da função **SnmpStrToEntity( ).**

Processo 8 : Aplicação informa qual a comunidade usada, através da função **SnmpStrToContext( )**.

Processo 9 : Aplicação informa qual os valores de retransmissão a serem usados, através da função **SnmpSetRetry( )**.

Processo 10 : Aplicação informa os valores de Time-Out (em centésimos de segundos) a serem usados, através da função **SnmpSetTimeOut( )**.

Processo 11 : Aplicação automatiza o processo de retransmissão dos pedidos de pesquisa a objetos através da função **SnmpSetRetransmitMode( )**.

Processo 12 : Aplicação identifica qual o objeto a ser pesquisado, através da função **SnmpStrToOid( )**, que converte uma representação textual de um objeto em um *Object Identifier*.

d) Contata agente ;

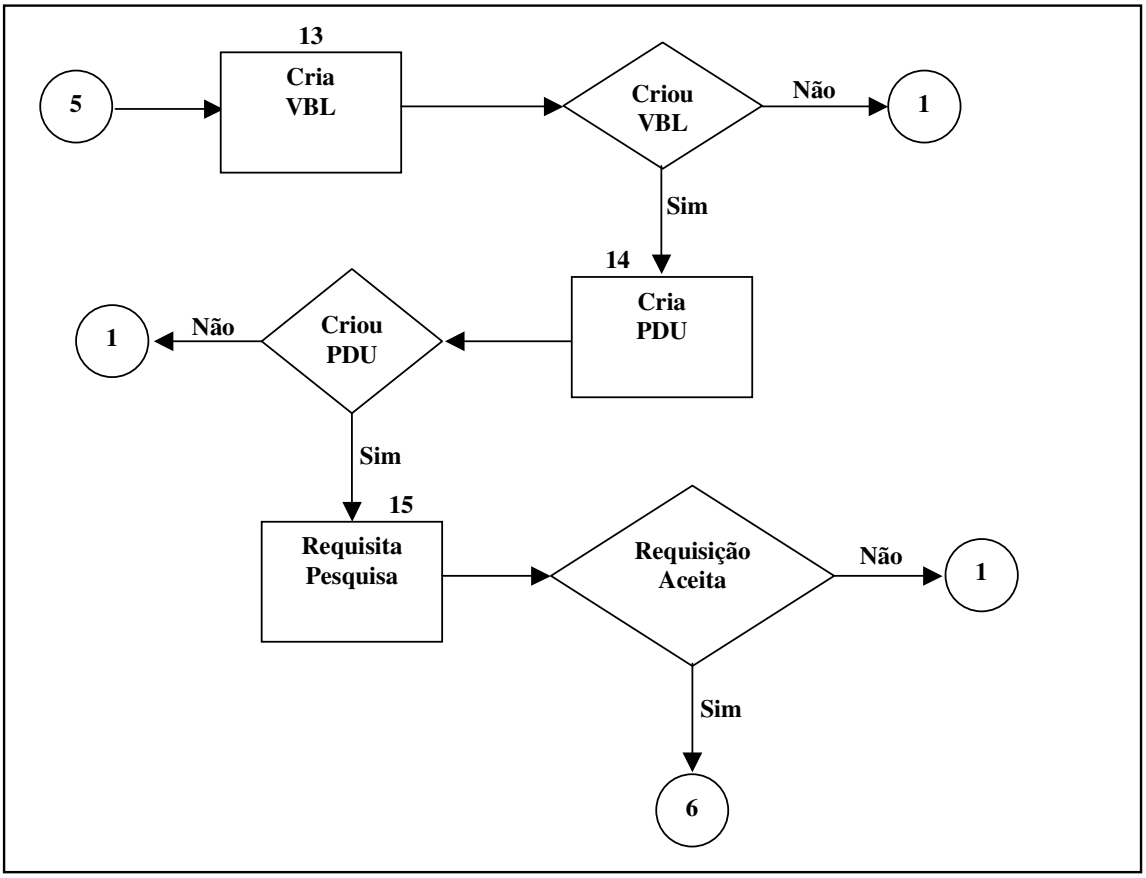


Figura 7 : **Processo Contata Agente.**

Processo 13 : Aplicação cria uma estrutura varbindlist (*Variable Binding List*), que é responsável por indicar qual a quantidade de objetos que estão sendo pesquisados, através da função **SnmpCreateVbl( )**.

Processo 14 : Aplicação aloca um unidade de dados do protocolo SNMP para as subsequentes chamadas e retornos de pesquisas a objetos, através da função **SnmpCreatePdu( ).**

Processo 15 : Aplicação transmite a PDU especificada para a entidade de destino, usando o contexto especificado, através da função **SnmpSendMsg(**). Aqui são feitas as requisições de pesquisa nos objetos.

Obs.: A função **SnmpSendMsg** é utilizada tanto para contatar um agente quanto para solicitar uma pesquisa, por isso o processo Pesquisa Objetos é detalhado juntamente com o processo Contata Agente.

e) Recebe resultado da pesquisa;

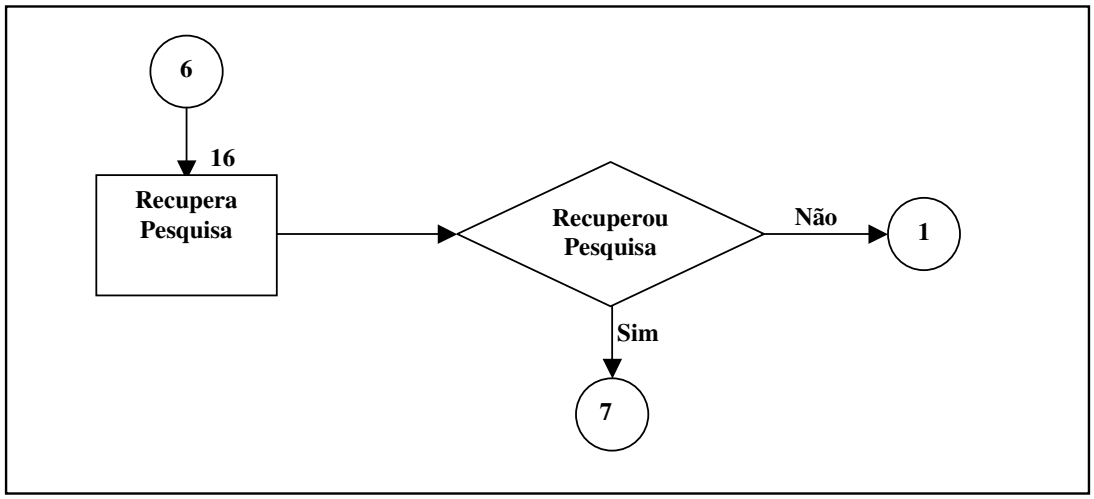


Figura 8 : **Processo Recebe Resultado da Pesquisa.**

Processo 16 : Aplicação recupera o resultado de uma requisição através da função **SnmpRecvMsg(**).

f) Mostra resultado.

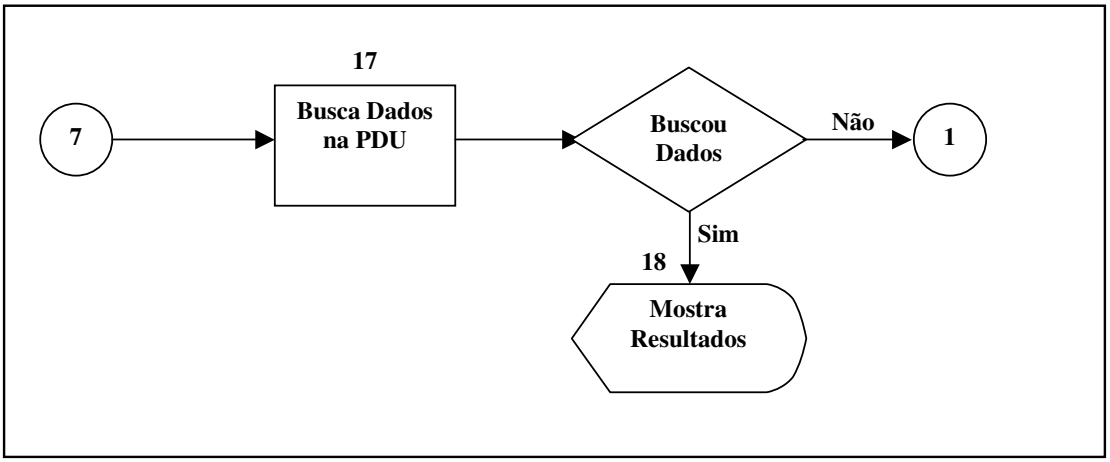


Figura 9 : **Processo Mostra Resultados**

Processo 17 : Aplicação extrai os elementos de dados da PDU e copia para as estruturas passadas como parâmetro através da função **SnmpGetPduData( )**.

Processo 18 : Como os retornos do processo 18 podem ser de vários tipos (inteiros, tempo, hora), a aplicação identifica cada um dos tipos definidos e os trata para que possam ser visualizados pelo usuário.

3.2 IMPLEMENTAÇÃO DO PROTÓTIPO

Para viabilizar a implementação do protótipo utilizou-se do ambiente de desenvolvimento *Delphi 3.0* para *Windows*;

Os *softwares* modernos procuram cada vez mais ter uma interface mais simples, proporcionando ao usuário um melhor entendimento e facilidade no seu uso. Neste mundo de interfaces mais simples e visuais, a primeira linguagem de programação a surgir foi o VB (*Visual Basic*). Essas novas interfaces permitiam o programador criar "visualmente" a interface com o usuário utilizando apenas o mouse, em vez de construí-las somente no código. Após o VB surgiram outras linguagens seguindo as mesmas características, sendo uma delas o Delphi [ENG97].

O Delphi oferece uma base sólida na construção de aplicativos visuais oferecendo muitas vantagens reais de produtividade para o programador. É um ambiente de programação

visual desenvolvido pela empresa *Borland*, baseada na linguagem de implementação Object Pascal, utilizadas para a criação de aplicações para sistemas operacionais Windows.

Por ser descendente do Pascal, linguagem conhecida pelo autor deste trabalho, e também por permitir a utilização de funções que interagem com DLLs, foi feita a escolha do Delphi para a implementação do protótipo.

Outras informações sobre o Borland Delphi encontram-se em [ENG97].

3.3 FUNCIONAMENTO DO PROTÓTIPO

A seguir será feita uma descrição das principais funções, bem como do funcionamento do protótipo.

Primeiramente, é criado o vínculo entre a aplicação e a biblioteca WinSnmp , o endereço IP local é recuperado e os parâmetros de configuração são carregados. O endereço IP remoto é fornecido pelo usuário, para identificar qual máquina este deseja contatar o agente. Os parâmetros de configuração também podem ser fornecidos manualmente, como mostra a figura 10.

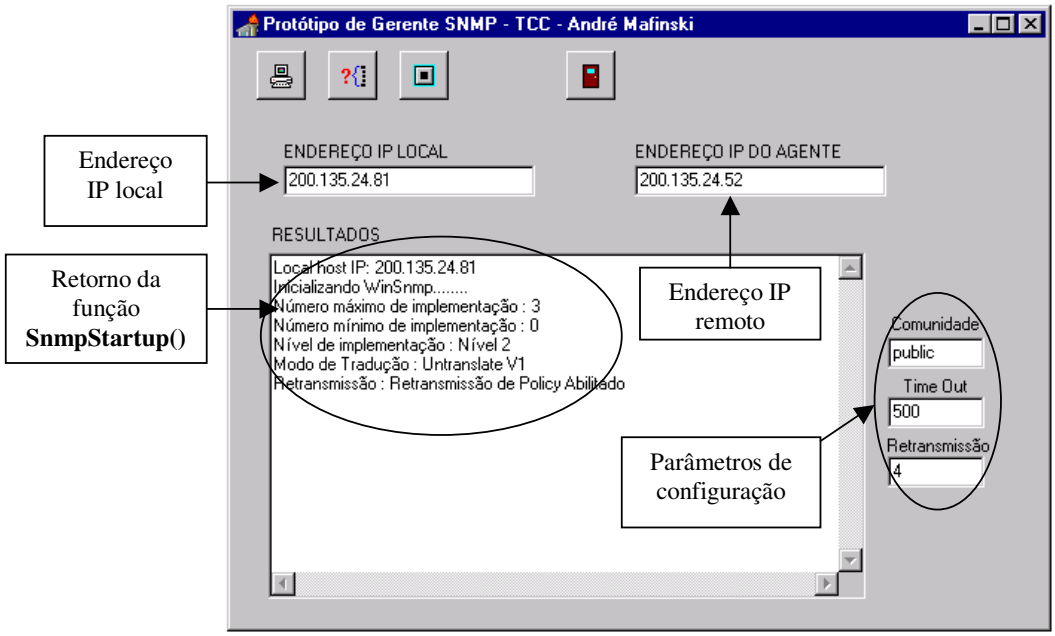


Figura 10: **Inicialização do protótipo.**

A função **SnmpStartup( )**, responsável pelo uso da biblioteca WinSnmp, traz os seguintes retornos:

a) número máximo de implementações, e o número mínimo de implementações que a biblioteca suporta;

b) o nível da implementação, sendo que:

c) o modo de tradução;

d) e modo de retransmissão, podendo este estar abilitado ou não;

Após isso o agente, identificado pelo endereço IP remoto, é contatado através do botão Contacta Agente. É retornado o tempo que a máquina esta ligada, como confirmação que o agente foi contatado, como mostra a figura 11. Caso não houver nenhum retorno o agente não foi contactado com sucesso.

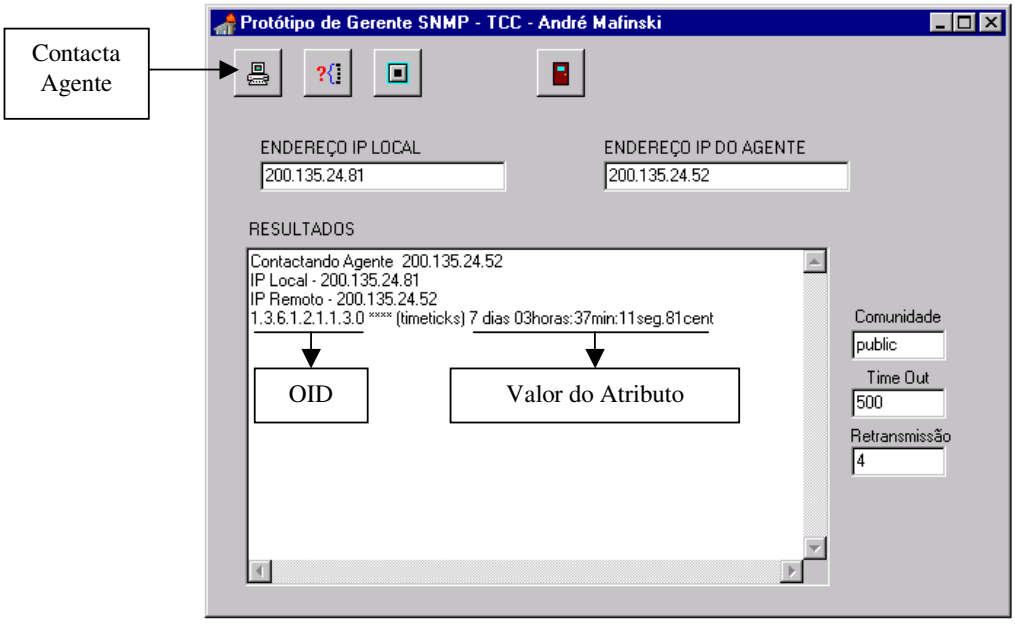


Figura 11: **Contato do agente SNMP**

Em seguida temos a pesquisa dos objetos já definidos na aplicação, buscando os atributos dos objetos no host agente contatado anteriormente.

Os objetos pré-definidos, conforme ordem de pesquisa, e como nos mostra a figura 12, são os seguintes:

a) SysDescr: objeto do grupo *system* da MIB-II, relacionado a descrição do sistema, identificado pelo OID 1.3.6.1.2.1.1.1.0;

b) SysUpTime: objeto do grupo *system* da MIB-II, relacionado ao tempo da última reinicialização do sistema, identificado pelo OID 1.3.6.1.2.1.1.3.0;

c) SysName: objeto do grupo *system* da MIB-II, relacionado ao nome do host contatado, identificado pelo OID 1.3.6.1.2.1.1.5.0;

d) IfDescr: objeto do grupo *interfaces* da MIB-II, relacionado a descrição de interface de redes do host contatado, identificado pelo OID 1.3.6.1.2.1.2.2.1.2.1;

e) IfType: objeto do grupo *interfaces* da MIB-II, relacionado ao tipo de interface de redes do host contactado, identificado pelo OID 1.3.6.1.2.1.2.2.1.3.1;

f) IfSpeed: objeto do grupo *interfaces* da MIB-II, relacionado ao tipo de interface de redes do host contatado, identificado pelo OID 1.3.6.1.2.1.2.2.1.5.1.

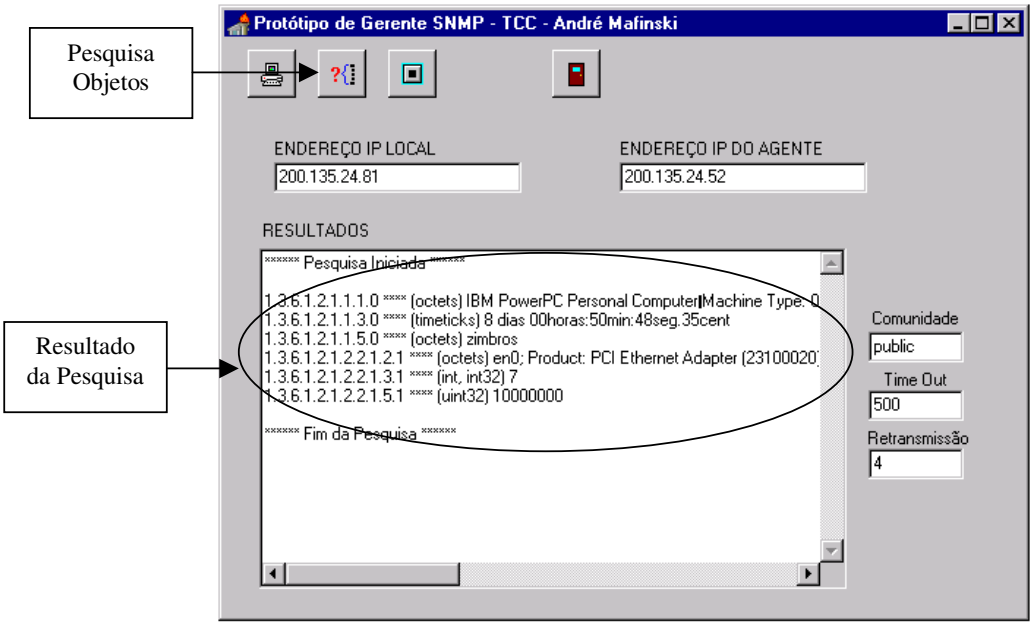


Figura 12: **Pesquisa dos objetos**

A cada novo agente que se desejar contatar, primeiramente deverá ser feito um cancelamento do agente contatado anteriormente. Após isso informa-se o endereço IP do novo

agente que se deseja contatar e repete-se os passos de contato do agente descritos anteriormente, como mostra a figura 13.

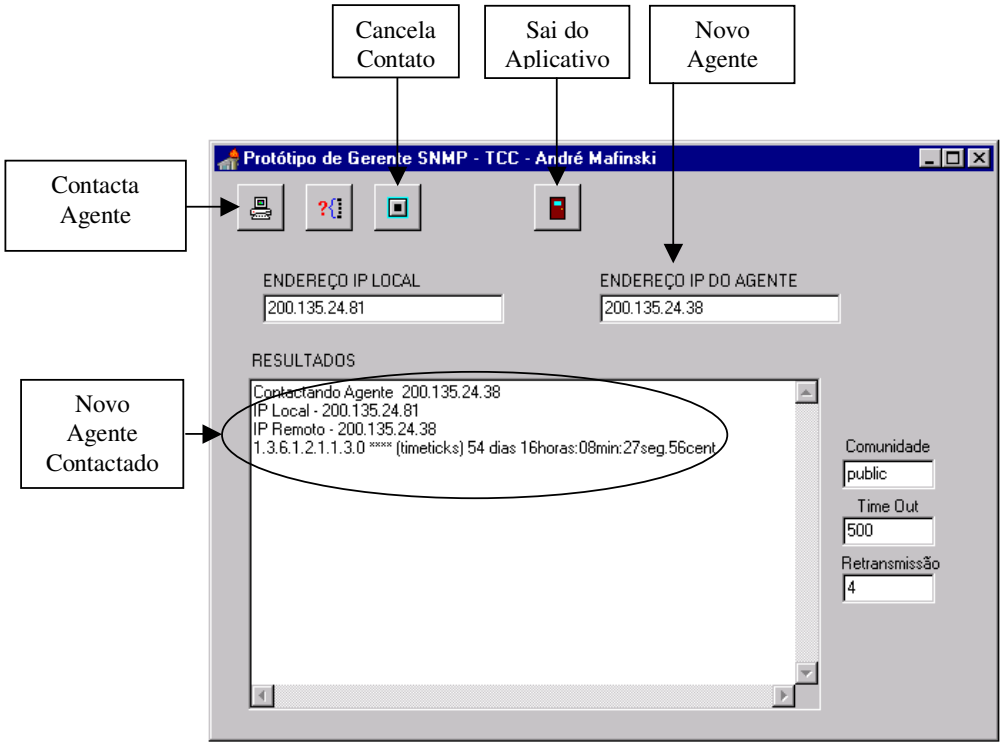


Figura 13: **Contato de um novo agente**

4 CONCLUSÃO

A conclusão deste trabalho foi conseqüência do cumprimento de todas as etapas, uma de cada vez, seguindo-se rigorosamente o plano de desenvolvimento, conforme o cronograma apresentado e aprovado na proposta de TCC.

No levantamento bibliográfico foi constatado que, apesar de haver várias fontes de pesquisa sobre o gerenciamento de redes, tecnologias e conceitos, poucas destas aprofundavam sobre o estudo do protocolo SNMP, sendo que se fez necessário uma busca mais apurada sobre este assunto.

O Gerenciamento de Redes ainda é tema pouco conhecido e difundido no meio acadêmico profissional. Por esta razão houve uma carência de material que defina a funcionalidade de um aplicativo de gerência de redes. Este conhecimento foi adquirido através do uso contínuo de outras ferramentas que implementavam as funções SNMP de gerenciamento.

Para viabilizar o estudo do ambiente Windows NT Server, no que diz respeito ao uso do protocolo SNMP e a criação de agentes, se fez necessário uma máquina com o ambiente instalado em rede. Quanto a isso, houveram alguns contratempos devido a não disponibilidade de uma máquina exclusiva para este fim e também devido a escassa bibliografia do referido assunto.

O uso biblioteca de funções de gerenciamento WinSnmp, foi de suma importância para a especificação e implementação do protótipo, sendo que seria inviável, devido ao tempo deste trabalho, implementar tais funções de comunicação. Fez-se necessário uma busca a materiais que explicassem de forma mais detalhada a uso das funções disponibilizadas pela biblioteca.

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

Embora atualmente existam algumas aplicações de gerenciamento de redes um pouco mais sofisticadas, a maioria destas aplicações possibilita apenas o monitoramento dos nós de

uma rede e não possui inteligência para auxiliar os administradores de redes na execução de suas tarefas.

O estado de desenvolvimento da a maioria das aplicações SNMP disponíveis atualmente é desencorajador. As aplicações ou são muito específicas, ou são genéricas, e em geral muito simples.

O protótipo deste trabalho tem como objetivo, monitorar apenas alguns objetos de um nó da rede, buscando apenas o resultado de uma pesquisa. Por isso, como continuidade deste trabalho, seria interessante, a interação do software de gerenciamento, com objetos que possam ser manipulados, neste ou até em outros ambientes de rede.

5 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 : TELEBRÀS, 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.

[DEM89] DEMARCO, Tom. **Análise Estruturada e Especificação de Sistema**. Rio de Janeiro : Campus, 1989.

[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.

[GRA96] GRANVILLE, Lisandro Zambenedetti. **Tutorial WinSnmp.** Porto Alegre : Universidade Federal do Rio Grande do Sul, 1996.

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

[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.

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

[RFC1156] McCLOGHRIE, K.; ROSE, M. **Management Information Base for Network Management of TCP/IP - based internets**, Network Working Group, Request for Comments: 1156, maio 1990.

[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.

[SOA95] SOARES, Luíz Fernando Gomes. **Redes de Computadores : das Lan's, Man's e Wan's às Redes ATM**. 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, 1994.

[THO97] THOMAS, Robert M. **Introdução às Redes Locais**. São Paulo : Makron Books, 1997.

[ZAK88] ZAKIR Jr, José. **Redes locais - O Estudo de Seus Elementos**. Rio de Janeiro : LTC - Livros Tecnológicos e Científicos, 1988.

[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.

ANEXO 1 – CÓDIGO FONTE DO PROTÓTIPO

```
// Universidade Regional de Blumenau.
// Trabalho de Conclusão de Curso.
// André Mafinski
// Protótipo de um Gerente SNMP
unit Prot;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, WinSnmp,Winsock,Wsnmpmib, StdCtrls, Buttons;


const
  CharSet: set of Char = [Char(9)..Char(13),' '..'~'];
  WM_MENSAGEM = WM_USER + 1;

type
  // Record que possui todas as varáveis referentes ao uso da biblioteca
  // WinSnmp

 EntidadeSNMP = Record
        IPGerente, IPAgente:array[0..80]of CHAR;
        SessaoSNMP:HSNMP_SESSION;
        Handle2: HWND;
        Mensagem: UINT;
        EntidadeAge,EntidadeGer:HSNMP_ENTITY;
        Contexto:HSNMP_CONTEXT;
        OID:SMIOID;
        VBL:HSNMP_VBL;
        PDU:HSNMP_PDU;
        Request:INTEGER;
        POID:SMIOID;
        POIDVal:array[0..MAXOBJIDSIZE]of SMIUINT32;
        Comunidade:SMIOCTETS;
    end;

    TForm1 = class(TForm)

    Memo: TMemo;
    ContacAgente: TSpeedButton;
    Agente: TEdit;
    Label1: TLabel;
    Label2: TLabel;
```

```
    Gerente: TEdit;
    Label3: TLabel;
    Comuni: TEdit;
    Time_Out: TEdit;
    Retra: TEdit;
    PesquisaObjetos: TSpeedButton;
    FechaContato: TSpeedButton;
    FechaTelaPricipal: TSpeedButton;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;

    procedure FormCreate(Sender: TObject);
    procedure ContacAgenteClick(Sender: TObject);
    procedure PesquisaObjetosClick(Sender: TObject);
    procedure FechaContatoClick(Sender: TObject);
    procedure FechaTelaPricipalClick(Sender: TObject);


private
    { Private declarations }

public
    { Public declarations }

  LocalHostIp : TInAddr;
  Es:EntidadeSNMP;
  procedure CriaContatoPesquisa(var Es: EntidadeSNMP);
end;

var

  Form1: TForm1;

implementation

{$R *.DFM}
procedure TForm1.FormCreate(Sender: TObject);
var
  Status:SNMPAPI_STATUS;
  Maior, Menor,Nivel,Trans, Retrans:SMIUINT32;
  Sessao,Ni,Transmi, Retransmi:STRING;
  WVersionRequested : WORD;
  WsaData : TWSAData;
  Err : Integer;
  LocalIP : TInAddr;
  He : PHostEnt;
  HeLocalIP : ^PInAddr;
```

```
  HeLocalHostName : array[0..255] of Char;
  I : Integer;

begin
   IsMultithread := True;
   memo.clear;
   Retra.Text:='4';
   Comuni.Text:='public';
   Time_Out.Text:='500';
   Randomize;
   Es.Request := Random(MAXINT);

   // Pega Endereço IP Local

   WVersionRequested := MakeWord(1, 1);
   Err := WSAStartup(wVersionRequested, wsaData);
   if err = 0 then begin
     if (LOBYTE(wsaData.wVersion) <> 1) OR
       (HIBYTE(wsaData.wVersion) <> 1) then begin
        WSACleanup();
     end
     else begin
     if 0 = gethostname(@heLocalHostName[0], 255) then begin
       He := gethostbyname(@heLocalHostName[0]);
       if he <> nil then begin
         heLocalIP := Pointer(he.h_addr_list);
         while Pointer(heLocalIP^) <> nil do begin
           localHostIp.S_addr := heLocalIP^^.S_addr;
           Gerente.Text := inet_ntoa(localHostIp);
           Memo.Lines.ADD('Local host IP: ' + inet_ntoa(localHostIp));
           Inc(heLocalIP, 4);
           Break;
         end;
       end;
     end;
     end;
   end;

   Memo.Lines.Add('Inicializando WinSnmp.....');

   // Aplicação avisa a biblioteca de funções que usará  o seus serviços SNMP,
   // abilitando a implementação a realizar quaisquer procedimentos de
   // inicialização e alocação de recursos e retornar alguma informação
   // útil a aplicação. Isto é feito através da função SnmpStartup( ).
   Status:=SnmpStartup(@Maior, @Menor, @Nivel,
                       @Trans, @Retrans);

   if Status = SNMPAPI_FAILURE then
```

```
  begin
     Memo.Text:='ERRO NA INICIALIZAÇÃO';
     Exit;
  end;
  Memo.Lines.Add('Número máximo de implementação : '+INTTOSTR(Maior));
  Memo.Lines.Add('Número mínimo de implementação : '+INTTOSTR(Menor));
  case Nivel of
    SNMPAPI_NO_SUPPORT:  Ni := 'Nível o';
    SNMPAPI_V1_SUPPORT:  Ni := 'Nível 1';
    SNMPAPI_V2_SUPPORT:  Ni := 'Nível 2';
    SNMPAPI_M2M_SUPPORT: Ni := 'Nível 3';
  end;

  Memo.Lines.Add('Nível de implementação : '+ Ni);
  case Trans of
    SNMPAPI_TRANSLATED:Transmi:='Translated';
    SNMPAPI_UNTRANSLATED_V1:Transmi:='Untranslate V1';
    SNMPAPI_UNTRANSLATED_V2:Transmi:='Untranslate V2';
  end;
  Memo.Lines.Add('Modo de Tradução : '+ Transmi);
  case Retrans of
    SNMPAPI_OFF: Retransmi:='Retransmissão de Policy Não Abilitado';
    SNMPAPI_ON: Retransmi:='Retransmissão de Policy Abilitado';
  end;
  Memo.Lines.Add('Retransmissão : '+ Retransmi);

  // Carrega os Módulos referentes as MIBs.
  // São usados para que os nomes dos objetos sejam achados.
  // Os arquivos RFC1155-SMI e RFC1213-MIB devem estar no diretório corrente.
  // O não uso destes arquivos não acarreta no funcionamento
  // normal do aplicativo.
  WsmLoadModule('RFC1155-SMI');
  WsmLoadModule('RFC1213-MIB');


end;

procedure TForm1.ContacAgenteClick(Sender: TObject);
var
  Stat:SNMPAPI_STATUS;
  Temp:array[0..40] of CHAR;

begin
  memo.clear;
  Memo.Lines.Add('Contactando Agente  '+Agente.text);


  StrCopy(Es.IPGerente,Pchar(Gerente.text));
```

```
StrCopy(Es.IPAgente,Pchar(Agente.text));
Memo.Lines.Add('IP Local - '+ ES.IPGerente);
Memo.Lines.Add('IP Remoto - '+ES.IPAgente);
Es.Comunidade.LEN:=STRLEN(PCHAR(comuni.text));
Es.Comunidade.PTR:=PBYTE(comuni.text);
Es.Handle2:=HANDLE;
Es.Mensagem:=WM_MENSAGEM;

//Aplicação aloca e inicializa a memória, recursos, mecanismos de
//comunicação e estruturas de dados necessários para que a aplicação
//possa usar os serviços da interface. Isto é feito através da
//função SnmpOpen( ), que retorna a identificação da sessão criada e
//continuará sendo usada pela aplicação para a chamada a outras funções.
if (Es.SessaoSNMP = 0) then
begin
  Es.SessaoSNMP:=SnmpOpen(Es.Handle2,Es.Mensagem);
end;
if Es.SessaoSNMP=SNMPAPI_FAILURE then
  exit;

// A aplicação registra o uso de traps,
// através da função SnmpRegister().
Stat:= SnmpRegister(Es.SessaoSNMP,0,0,0,NIL,SNMPAPI_ON);
if Stat = SNMPAPI_FAILURE  then
  exit;

// A aplicação cria a entidade agente.
// Esta recebe todos os dados que contêm a sessão criada anteriormente,
// e informa o IP que identifica o agente. Isto é feito através da função
// SnmpStrToEntity( ).

Es.EntidadeAge:=SnmpStrToEntity(Es.SessaoSNMP,@Es.IPagente[0]);
if Es.EntidadeAge = SNMPAPI_FAILURE  then
  exit;

// A aplicação cria a entidade Gerente.
// Esta recebe todos os dados que contêm a sessão criada anteriormente,
// e informa o IP que identifica o agente. Isto é feito através da função
// SnmpStrToEntity( ).
Es.EntidadeGer:=SnmpStrToEntity(Es.SessaoSNMP,@Es.IPgerente[0]);
if Es.EntidadeGer = SNMPAPI_FAILURE  then
  exit;

// Aplicação informa  qual a comunidade usada,através da função
// SnmpStrToContext( ).
Es.Contexto:=SnmpStrToContext(Es.SessaoSNMP,@Es.Comunidade);
if Es.Contexto = SNMPAPI_FAILURE  then
  exit;
```

```
  // Aplicação informa qual os valores de retransmissão a serem usados,
  // através da função SnmpSetRetry( ).
  Stat:=SnmpSetRetry(Es.EntidadeGer,StrToInt(Retra.text));
  if Stat = SNMPAPI_FAILURE  then
    exit;


  // Aplicação informa os valores de Time-Out (em centésimos de segundos)
  // a serem usados, através da função SnmpSetTimeOut( ).
  Stat:=SnmpSetTimeOut(Es.EntidadeGer,StrToInt(Time_Out.text));
  if Stat = SNMPAPI_FAILURE  then
    exit;

  // As funções SnmpSetTimeOut e SnmpSetRetry são usados tanto para entidades
  // agentes quanto para entiades gerentes.
  Stat:=SnmpSetRetry(Es.EntidadeAge,StrToInt(Retra.text));
  if Stat = SNMPAPI_FAILURE  then
    exit;

  Stat:=SnmpSetTimeOut(Es.EntidadeAge,StrToInt(Time_Out.text));
  if Stat = SNMPAPI_FAILURE  then
    exit;


  // Aplicação automatiza o processo de retransmissão dos pedidos de pesquisa
  // a objetos através da função SnmpSetRetransmitMode( ).
  Stat:=SnmpSetRetransmitMode(SNMPAPI_ON);
  if Stat = SNMPAPI_FAILURE  then
    exit;

  Temp:='1.3.6.1.2.1.1.3';{Tempo do sistema no ar}

  if Es.Oid.Ptr <> nil then
    SnmpFreeDescriptor(SNMP_SYNTAX_OID,@ES.OID);

  // Aplicação identifica qual o objeto a ser pesquisado, através da função
  // SnmpStrToOid( ), que converte uma representação textual de um objeto em
  // um Object Identifier.
    if (SnmpStrToOid(Temp,@ES.OID)= SNMPAPI_FAILURE)then
       exit;
    Es.Poid.Len:=0;
    Es.Poid.Ptr:=@Es.PoidVal[0];
    CRIACONTATOPESQUISA(ES);
end;

procedure TFORM1.CRIACONTATOPESQUISA(var Es:EntidadeSNMP);
var
```

```
  Stat:SNMPAPI_STATUS;
  AgenteRecv, GerenteRecv: HSNMP_ENTITY;
  ContextoRecv:HSNMP_CONTEXT;
  PduRecv: HSNMP_PDU;
  VblRecv: HSNMP_VBL;
  TipoPdu,Staer,Inder: SMIINT;
  Idreq:SMIINT32;
  OID2:SMIOID;
  Vale:SMIVALUE;
  ContOID,J,I,D,H,M,S,TH,Aux:Integer;
  ComparaResult:SMIINT;
  Ts,Ts1 : array[0..2048] of char;
  OIDP : smiLPUINT32;
  Charp : smiLPBYTE;
  MibNode : HWSM_NODE;
  Msg:TMESSAGE;
  printHex : Boolean;

begin

  // Aplicação cria uma estrutura VarbinList(Variable Binding List), que é
  // responsável por indicar qual a quantidade de objetos que estão sendo
  // pesquisados, através da função SnmpCreateVbl( ).
  Es.Vbl:=SnmpCreateVbl(Es.SessaoSNMP,@ES.OID,NIL);
  if Es.Vbl = SNMPAPI_FAILURE then
    exit;
  Es.Request:=Es.Request+1;

  // Aplicação aloca um unidade de dados do protocolo SNMP para as
  // subsequentes chamadas e retornos de pesquisas a objetos, através da
  // função SnmpCreatePdu( ).
  Es.Pdu:=SnmpCreatePdu(Es.SessaoSNMP,SNMP_PDU_GETNEXT,
                  Es.Request,0,0,Es.Vbl);
  if Es.Pdu=SNMPAPI_FAILURE then
    exit;

  // Aplicação transmite a PDU especificada para a entidade de destino,
  // usando o contexto especificado, através da função SnmpSendMsg( ).
  // Aqui são feitas as requisições de pesquisa nos objetos.
  Stat:=SnmpSendMsg(Es.SessaoSNMP,Es.EntidadeGer,Es.EntidadeAge,
              Es.Contexto,Es.Pdu);
  if Stat = SNMPAPI_FAILURE then
    exit;

  // Libera VBL e PDU.
  Stat:=SnmpFreeVbl(Es.Vbl);
  if Stat = SNMPAPI_FAILURE  then
    exit;
```

```
Stat:=SnmpFreePdu(Es.Pdu);
if Stat = SNMPAPI_FAILURE  then
   exit;

// Variáveis usadas para o retorno de mensagens.
AgenteRecv:=0;
GerenteRecv:=0;
ContextoRecv:=0;
PduRecv:=0;

// Aplicação recupera o resultado de uma requisição através
// da função SnmpRecvMsg( ).
Stat:=SnmpRecvMsg(Es.SessaoSNMP, @AgenteRecv, @GerenteRecv,
                  @ContextoRecv,@PduRecv);

if Stat = SNMPAPI_FAILURE  then
   exit;

// Aplicação extrai os elementos de dados da PDU e copia para as estruturas
// passadas como parâmetro através da função SnmpGetPduData( ).
Stat:=SnmpGetPduData(PduRecv,@TipoPdu,@IdReq,@Staer,
                     @Inder,@VblRecv);
if Stat = SNMPAPI_FAILURE  then
   exit;
// Conta a quantidade de objetos da lista VBL.
ContOid:=SnmpCountVbl(VblRecv);
if ContOid = SNMPAPI_FAILURE  then
   exit;
for I:=1 to ContOid do
begin
   Stat:=SnmpGetVB(VblRecv,I,@oid2,@Vale);
   if I = 1 then
   begin
      SnmpOidCopy(@oid2,@Es.Oid);
   end;
end;

Ts := '';
Aux := Oid2.len;
// Caso os módulos RFC1155 e RFC1213 estiverem carregados, a aplicação
// procurara o caminho do referido objeto na árvore e trará uma string
// contendo o nome deste objeto.
for j := Oid2.len downto 0 do begin
  OID2.len := j;
  mibNode := WsmMapOidToNode(@OID2);
  if (mibNode <> 0) then
  begin
```

```
    i := 500;

    // Pega nome do objeto.
    WsmGetNodeString(mibNode, WSM_INFO_NAME, @i, @ts1[0]);
    StrCat(ts, ts1);
    break;
  end;
end;
OID2.len := AUX;

oidp := OID2.ptr;
if j < 0 then j := 0;
  inc(oidp, j);
for i := j + 1 to OID2.len do begin
  if i <> 1 then
    StrCat(ts, '.');
  StrCat(ts, PChar(IntToStr(oidp^)));
  inc(oidp);
end;

StrCat(ts, ' **** ');

// Testa o tipo do valor dos atributos do objeto retornados
case vale.syntax of
  SNMP_SYNTAX_INT:            StrCat(ts, '(int, int32)');
  SNMP_SYNTAX_UINT32:         StrCat(ts, '(uint32)');
  SNMP_SYNTAX_CNTR32:         StrCat(ts, '(cntr32)');
  SNMP_SYNTAX_TIMETICKS:      StrCat(ts, '(timeticks)');
  SNMP_SYNTAX_OCTETS:         StrCat(ts, '(octets)');
  SNMP_SYNTAX_CNTR64:         StrCat(ts, '(cntr64)');
  SNMP_SYNTAX_IPADDR:         StrCat(ts, '(ipaddr)');
  SNMP_SYNTAX_OPAQUE:         StrCat(ts, '(opaque)');
  SNMP_SYNTAX_NSAPADDR:       StrCat(ts, '(nsapaddr)');
  SNMP_SYNTAX_OID:            StrCat(ts, '(oid)');
  SNMP_SYNTAX_NULL:           StrCat(ts, '(null)');
  SNMP_SYNTAX_NOSUCHOBJECT:   StrCat(ts, '(nosuchobject)');
  SNMP_SYNTAX_NOSUCHINSTANCE: StrCat(ts, '(nosuchinstance)');
  SNMP_SYNTAX_ENDOFMIBVIEW:   StrCat(ts, '(endofmibview)');
end;
StrCat(ts, ' ');


// Os valores dos atributos dos objetos são tratados conforme o seu tipo
// retornado.
case vale.syntax of
  //SNMP_SYNTAX_INT32:,
  SNMP_SYNTAX_INT:
    StrCat(ts, PChar(IntToStr(vale.sNumber)));
```

```
    SNMP_SYNTAX_UINT32:
      StrCat(ts, PChar(IntToStr(vale.uNumber)));
    SNMP_SYNTAX_CNTR32:
     StrCat(ts, PChar(IntToStr(vale.uNumber)));
    SNMP_SYNTAX_TIMETICKS: begin
      d := vale.uNumber;
      th := d mod 100; d := d div 100;
      s := d mod 60;   d := d div 60;
      m := d mod 60;   d := d div 60;
      h := d mod 24;
      d := d div 24;
      StrCat(ts, PChar(Format('%d dias
%02.02dhoras:%02.02dmin:%02.02dseg.%02.02dcent', [d,h,m,s,th])));
    end;
    SNMP_SYNTAX_CNTR64 : begin
      StrCat(ts, PChar(IntToHex(vale.hNumber.hipart, 8)));
      StrCat(ts, '.');
      StrCat(ts, PChar(IntToHex(vale.hNumber.lopart, 8)));
      StrCat(ts, ' (hex)');
    end;
    SNMP_SYNTAX_OCTETS: begin
      if vale.pString.len < 1 then begin
        StrCat(ts, ' (zero-length)');
      end;
      printHex := False;
      charp := vale.pString.ptr;
      ts1 := '';
      for i := 1 to vale.pString.len do begin
        if ((NOT (Char(charp^) IN CharSet)) OR
            ((Char(charp^) = Char(0)) AND
             (i <> vale.pString.len))) then begin
             printHex := True;
        end;
        if i <> 1 then
          StrCat(ts1, '.');
        StrCat(ts1, PChar(IntToHex(charp^, 2)));
        inc(charp);
      end;
     if printHex then begin
       StrCat(ts, ts1);
       StrCat(ts, ' (hex)');
     end
     else begin
       Move(vale.pString.ptr^, ts1, vale.pString.len);
       ts1[vale.pString.len] := Char(0);
       StrCat(ts, ts1);
     end;
    end;
```

```
    SNMP_SYNTAX_IPADDR : begin
      charp := vale.pString.ptr;
      for i := 1 to vale.pString.len do begin
        if i <> 1 then
          StrCat(ts, '.');
        StrCat(ts, PChar(IntToStr(charp^)));
        inc(charp);
      end;
    end;

    SNMP_SYNTAX_OPAQUE : begin
      charp := vale.pString.ptr;
      for i := 1 to vale.pString.len do begin
        if i <> 1 then
          StrCat(ts, '.');
        StrCat(ts, PChar(IntToHex(charp^, 2)));
        inc(charp);
      end;
      StrCat(ts, ' (hex)');
    end;
    SNMP_SYNTAX_NSAPADDR: begin
      charp := vale.pString.ptr;
      for i := 1 to vale.pString.len do begin
        if i <> 1 then
          StrCat(ts, '.');
        StrCat(ts, PChar(IntToHex(charp^, 2)));
        inc(charp);
      end;
      StrCat(ts, ' (hex)');
    end;

    SNMP_SYNTAX_OID : begin
      // Analisa se o objeto nao possui atributos
      if vale.oid.len = 2 then begin
        oidp := vale.oid.ptr;
        if oidp^ = 0 then begin
          inc(oidp);
          if oidp^ = 0 then begin
            StrCat(ts, '(null-oid) ');
          end;
        end;
      end;
    end;
  end;
  MEMO.LINES.ADD(TS);
end;

procedure TForm1.PesquisaObjetosClick(Sender: TObject);
```

```
var
  Obj: array [0..40] of char;
begin


  // Para cada objeto que se queira pesquisar chama-se o procedimento
  // CriaContatoPesquisa.
  Memo.Clear;
  Memo.Lines.Add('****** Pesquisa Iniciada ******');
  Memo.Lines.Add('');
  Obj:='1.3.6.1.2.1.1.1';
  if ES.OID.PTR <> nil then

     // A função SnmpFreeDescriptor é usada pela aplicação para informar
     // que a mesma não precisa ter acesso ao descritor de objeto
     // preenchido anteriormente.
     SnmpFreeDescriptor(SNMP_SYNTAX_OID,@Es.Oid);
  if (SNMPSTRTOOID(Obj,@Es.Oid)= SNMPAPI_FAILURE)then
      exit;
  Es.Poid.Len:=0;
     Es.Poid.Ptr:=@Es.POidVal[0];
     CRIACONTATOPESQUISA(ES);

  Obj:='1.3.6.1.2.1.1.3';
  if ES.OID.PTR <> nil then
     SnmpFreeDescriptor(SNMP_SYNTAX_OID,@Es.Oid);
  if (SNMPSTRTOOID(Obj,@Es.Oid)= SNMPAPI_FAILURE)then
      exit;
  Es.Poid.Len:=0;
     Es.Poid.Ptr:=@Es.POidVal[0];
     CRIACONTATOPESQUISA(ES);


  OBJ:='1.3.6.1.2.1.1.5';
  if ES.OID.PTR <> nil then
     SnmpFreeDescriptor(SNMP_SYNTAX_OID,@Es.Oid);
  if (SNMPSTRTOOID(Obj,@Es.Oid)= SNMPAPI_FAILURE)then
      exit;
  Es.Poid.Len:=0;
     Es.Poid.Ptr:=@Es.POidVal[0];
     CRIACONTATOPESQUISA(ES);


  OBJ:='1.3.6.1.2.1.2.2.1.2';
  if ES.OID.PTR <> nil then
     SnmpFreeDescriptor(SNMP_SYNTAX_OID,@Es.Oid);
  if (SNMPSTRTOOID(Obj,@Es.Oid)= SNMPAPI_FAILURE)then
      exit;
```

```
        Es.Poid.Len:=0;
        Es.Poid.Ptr:=@Es.POidVal[0];
        CRIACONTATOPESQUISA(ES);


      OBJ:='1.3.6.1.2.1.2.2.1.3';
      if ES.OID.PTR <> nil then
        SnmpFreeDescriptor(SNMP_SYNTAX_OID,@Es.Oid);
      if (SNMPSTRTOOID(Obj,@Es.Oid)= SNMPAPI_FAILURE)then
         exit;
      Es.Poid.Len:=0;
        Es.Poid.Ptr:=@Es.POidVal[0];
        CRIACONTATOPESQUISA(ES);

      OBJ:='1.3.6.1.2.1.2.2.1.5';
      if ES.OID.PTR <> nil then
        SnmpFreeDescriptor(SNMP_SYNTAX_OID,@Es.Oid);
      if (SNMPSTRTOOID(Obj,@Es.Oid)= SNMPAPI_FAILURE)then
         exit;
      Es.Poid.Len:=0;
        Es.Poid.Ptr:=@Es.POidVal[0];
        CRIACONTATOPESQUISA(ES);
      Memo.Lines.Add('');
      Memo.Lines.Add('****** Fim da Pesquisa ******');

    end;

    procedure TForm1.FechaContatoClick(Sender: TObject);
    var
      Stat: SNMPAPI_STATUS;
    begin

      // Libera recursos para que outros agentes possam ser contatados.
      Stat := SnmpFreeContext(Es.Contexto);
      Es.Contexto := 0;
      Stat := SnmpFreeEntity(es.EntidadeGer);
      Es.EntidadeGer := 0;
      Stat:= SnmpFreeEntity(Es.EntidadeAge);
      Es.EntidadeAge := 0;
      Stat := SnmpClose(Es.SessaoSNMP);
      Es.SessaoSNMP := 0;
    end;

    procedure TForm1.FechaTelaPricipalClick(Sender: TObject);
    begin
      Close;
    end;
end.
```