Celso Kopp Webber

UMA MIB PARA APLICAÇÕES INTERNET

Dissertação de Mestrado apresentada para cumprimento parcial das exigências para o título de Mestre. Curso de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina.
Orientador: Prof. Vitório Bruno Mazzola
Co-Orientadora: Profa. Elizabeth Sueli Specialski

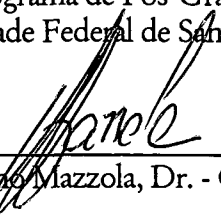
Florianópolis

1997

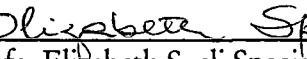
UMA MIB PARA APLICAÇÕES INTERNET

Celso Kopp Webber

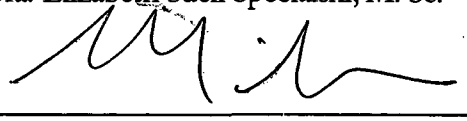
Esta dissertação foi julgada adequada para obtenção do título de Mestre em Ciência da Computação, na área de concentração Sistemas de Computação, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.



Prof. Vitório Bruno Mazzola, Dr. - Orientador

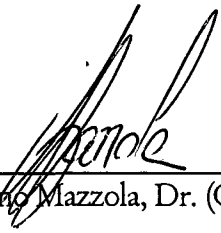


Profa. Elizabeth Sueli Specialski, M. Sc. - Co-orientadora

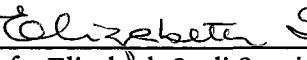


Prof. Murilo Silva de Camargo, Dr. - Coordenador

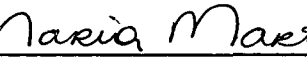
Banca Examinadora:



Prof. Vitório Bruno Mazzola, Dr. (CPGCC-UFSC)



Profa. Elizabeth Sueli Specialski, M. Sc. (CPGCC-UFSC)



Profa. Maria Marta Leite, M. Sc. (INE-UFSC)

Florianópolis, 25 de março de 1997

AGRADECIMENTOS

Gostaria de agradecer especialmente à professora Elizabeth Sueli Specialski, pelo constante apoio nas atividades do dia a dia, bem como na geração de idéias que permitem que trabalhos como este saiam das cabeças dos alunos e sejam postos no papel.

Ao professor Vitório Bruno Mazzola, pela força passada durante o tempo em que foi meu orientador.

A todos os professores, pelo tempo "perdido" em nos passar uma pequena fração de seu conhecimento, e pelo apoio que sempre encontramos.

Aos funcionários, que de uma forma ou de outra sempre nos ajudaram neste curso de pós-graduação. Em especial, gostaria de agradecer à Valdete e à Verinha, que constituem o cartão de visitas deste curso de pós-graduação.

Gostaria ainda a todos os meus amigos, mas em especial a três deles: Roberto, Iwens e Bráulio (a Diretoria). Sem eles não conseguiria ter chegado até aqui sem que tivesse ficado louco.

Por último, mas não por isso menos importante, gostaria de agradecer à minha família, pela constante força motivadora que me manteve no caminho certo.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E GLOSSÁRIO

API. *Application Programming Interface* – Interface de Programação de Aplicações. Conjunto de funções relacionadas para a programação de aplicativos.

ASN.1. *Abstract Syntax Notation One* – Notação de Sintaxe Abstrata Um. Uma notação para representação de dados em protocolos de comunicação, definida pela ISO. O gerenciamento Internet utiliza um subconjunto de ASN.1.

BER. *Basic Encoding Rules* – Regras Básicas de Codificação. Uma das várias possibilidades para se codificar mensagens ASN.1. SNMP utiliza um subconjunto de BER.

CCITT. *International Consultative Committee on Telegraphy and Telephony* – Comitê Consultivo Internacional para Telegrafia e Telefonia. Um dos mais importantes órgãos de padronização na área de telecomunicações. Seu nome atual é ITU-T.

CMIP. *Common Management Information Protocol* – Protocolo de Informações de Gerenciamento Comum. O protocolo de gerenciamento OSI.

CMOT. *CMIP Over TCP/IP* – CMIP sobre TCP/IP. Proposta da utilização do protocolo OSI de gerenciamento sobre TCP/IP.

DARPA. *Defense Advanced Research Projects Agency* – Agência de Projetos de Pesquisa Avançados de Defesa. A agência responsável pela criação da ARPANET, que originou a Internet.

DLL. *Dynamic Link Library* – Biblioteca de Ligação Dinâmica. Conjunto de funções na forma de uma biblioteca carregada e descarregada da memória em tempo de execução, conforme sua necessidade.

FTP. *File Transfer Protocol* – Protocolo de Transferência de Arquivos. O protocolo Internet para a transferência de arquivos entre *hosts*.

HEMS. *High-Level Entity-Management Protocol* – Protocolo de Gerenciamento de Entidades de Alto Nível. Uma das primeiras propostas de gerenciamento da Internet.

HTTP. *Hypertext Markup Language* – Linguagem de Marcação de Hipertexto. A linguagem utilizada na construção de páginas WWW.

IAB. *Internet Activities Board* – Comitê de Atividades Internet. O órgão responsável pelos padrões na Internet em um nível hierárquico superior ao IETF.

IANA. *Internet Assigned Numbers Authority* – Autoridade de Números Designados Internet. O órgão responsável por delegar números na Internet para diversos fins, como endereços IP, identificadores de empresas (OID), entre outras coisas.

ICMP. *Internet Control Message Protocol* – Protocolo de Controle de Mensagens Internet. O primeiro protocolo utilizado para a verificação do estado da rede na Internet.

ID. *Internet Draft* – Um documento resultante de um grupo de trabalho do IETF e ainda não publicado, disponível para revisões e comentários.

IETF. *Internet Engineering Task Force* – Força Tarefa de Engenharia da Internet. É o órgão que efetivamente define os padrões Internet.

IPX. *Internet Packet Exchange* – Troca de Pacotes entre Redes. Um protocolo para redes locais desenvolvido pela Novell.

ISO. *International Organization for Standardization* – Organização Internacional para Padronização. Um dos órgãos mais importantes no mundo de padronizações, com diversos estudos na área de computadores e particularmente de redes de computadores.

ISOC. *Internet Society* – Sociedade Internet. O órgão de nível mais alto na administração da Internet.

ISODE. *ISO Development Environment* – Ambiente de Desenvolvimento ISO. O ambiente de desenvolvimento de aplicações de rede da ISO.

ITU-T ou **ITU-TSS.** *International Telecommunications Union Telecommunication Standards Section* – União Internacional de Telecomunicações. Um dos mais importantes órgãos de padronização na área de telecomunicações. Antigo CCITT.

LME. *Layer Management Entity* – Entidade de Gerenciamento de Camada OSI.

MIB. *Management Information Base* – Base de Informações de Gerenciamento. Uma coleção de informações gerenciais sobre um determinado recurso.

OS. *Operating System* – Sistema Operacional (SO). O principal *software* de um computador, responsável por gerenciar os recursos da máquina, oferecendo suporte aos programas dos usuários.

OSI. *Open Systems Interconnection* – Interconexão de Sistemas Abertos. Um conjunto de padronizações para interconexão de sistemas em redes de computadores.

PDU. *Protocol Data Unit* – Unidade de Dados de Protocolo.

RFC. *Request For Comments* – Requisições Para Comentários. Os documentos publicados pelo IETF e que tratam de assuntos Internet. Nem todos os RFCs são padrões, e alguns jamais o chegam a ser.

SDK. *Software Development Kit* – Kit de Desenvolvimento de Software. Um conjunto de bibliotecas, ferramentas e documentação com o propósito de facilitar a tarefa de desenvolvimento de *software* para determinado fim.

SMAP. *System Management Application Process* – Processo da Aplicação de Gerenciamento de Sistemas OSI.

SMAE. *System Management Application Entity* – Entidade da Aplicação de Gerenciamento de Sistemas OSI.

SMI. *Structure of Management Information* – Estrutura da Informação de Gerenciamento. Define a maneira como as informações de gerenciamento são criadas e seus relacionamentos.

SMP. *Simple Management Protocol* – Protocolo de Gerenciamento Simples. Uma das propostas para substituir o SNMP.

SNMP. *Simple Network Management Protocol* – Protocolo de Gerenciamento de Redes Simples, o protocolo de gerenciamento utilizado na Internet.

SO. Sistema Operacional. Vide OS.

TCP/IP. *Transmission Control Protocol / Internet Protocol* – Protocolo de Controle de Transmissões / Protocolo Internet. Os protocolos básicos utilizados na Internet.

TFTP. *Trivial FTP* – FTP Trivial. Uma simplificação do protocolo FTP, geralmente usado para transferências automáticas de arquivos entre dispositivos ou *hosts*.

UPS. *Uninterruptiple Power Supply* – Fonte de Energia não Interrompível.

WG. *Working Group* – Grupo de Trabalho. Na terminologia do IETF, consiste em um grupo de pessoas discutindo as padronizações sobre um determinado item dentro de uma das áreas funcionais do IETF.

WWW. *World Wide Web* – Grande Teia Mundial. Nome dado ao conjunto de máquinas que trocam informações utilizando o protocolo HTTP.

RESUMO

Este trabalho trata do gerenciamento de aplicações cliente de rede da arquitetura de redes Internet. Para isso, é proposta uma extensão à MIB-II padrão da Arquitetura de Gerenciamento de Redes Internet. O objetivo desta nova MIB é oferecer informações que permitam monitorar e controlar aplicações de rede Internet em estações clientes de rede. Os esforços atuais nesta área apenas oferecem informações para a monitoração de aplicações.

Para a construção do módulo da MIB proposta, vários aspectos são discutidos, estabelecendo uma metodologia para o desenvolvimento de novas MIBs SNMP. Em seguida, esta metodologia é utilizada para especificar o módulo de MIB para o gerenciamento de aplicações Internet.

O trabalho ainda descreve algumas alternativas para a implementação de um agente SNMP para esta nova MIB.

O módulo de MIB especificado passa por uma etapa de verificação, através da utilização de um compilador de MIBs, que verifica se a sintaxe e estrutura do módulo estão corretas.

Por fim são apresentadas as possibilidades de utilização do novo agente em um ambiente de redes típico, além dos resultados obtidos no ambiente de testes.

Palavras-chave: gerenciamento de redes, SNMP, MIB, gerenciamento de aplicações de rede.

ABSTRACT

This work focuses on the management of client network applications based on the Internet Network Architecture, by presenting an extension to the Internet Management Framework MIB-II. The purpose of this new MIB is to gather information that permit the control and monitoring of Internet network applications running on network client workstations. Current efforts on this subject are limited to monitoring of applications.

To build the MIB module corresponding to the proposed MIB, some aspects are discussed, and a methodology for SNMP MIB development is established. This methodology is used to specify the MIB module for the management of Internet applications.

This work still describes alternatives to the implementation of an SNMP agent for this new MIB.

The specified MIB module passes through a verification step, where a MIB compiler is used to verify the structure and the syntax of the module.

At last, the possibilities of use of the agent in a typical network environment are presented, plus the results obtained on the tests environment.

Keywords: network management, SNMP, MIB, network applications management.

Capítulo 1

INTRODUÇÃO E MOTIVAÇÃO

1. Introdução e Motivação

Podemos dizer que hoje em dia as redes de computadores não são mais artigos de luxo. De sistemas sofisticadíssimos e de custos elevados as redes passaram a ser simples e mais confiáveis, a ponto de estarem presentes até mesmo em algumas residências. Assim, podemos encontrar com facilidade redes locais com centenas ou milhares de computadores interligados.

Entretanto, redes de computadores não são perfeitas e estão sujeitas a diversos problemas, que vão desde causas físicas (cabeamento, interferências, etc.) até problemas lógicos com os serviços oferecidos pela rede (servidores sobrecarregados, excesso de tráfego, etc.).

Não é muito difícil de imaginar que quanto maior o tamanho da rede, maior a possibilidade de problemas. Estes problemas, se não forem resolvidos em tempo hábil, trarão prejuízos diretos às pessoas que utilizam os serviços da rede, e conseqüentemente à toda a organização.

Para auxiliar o administrador de redes na difícil tarefa de manter tudo funcionando, desde os primórdios das redes de computadores, foram imaginados sistemas de gerenciamento de redes.

Inicialmente proprietários, os sistemas de gerenciamento tinham por objetivo gerenciar sistemas específicos de um determinado fabricante. Desse modo, o

administrador de redes lidava com uma quantidade de ferramentas de gerenciamento pelo menos igual à quantidade dos diferentes fabricantes dos equipamentos do seu ambiente de rede.

Para que houvesse interoperabilidade entre os sistemas de gerenciamento, diversos esforços começaram a surgir no sentido de se definirem arquiteturas de gerenciamento padronizadas.

Assim como nas padronizações internacionais para arquiteturas de redes não proprietárias, as arquiteturas de gerenciamento que mais se destacam são a Arquitetura de Gerenciamento Internet e a Arquitetura de Gerenciamento OSI da ISO.

Existem diversas arquiteturas de gerenciamento proprietárias, mas normalmente não costumam ser largamente adotadas, devido aos problemas de interoperabilidade com produtos de outros fabricantes.

1.1 Necessidades de Gerenciamento

O gerenciamento ideal de uma rede de computadores começa desde a sua concepção. Desde a escolha dos equipamentos de interconexão de redes até o *software* de gerenciamento, deve-se ter um projeto básico do futuro gerenciamento da rede.

Nem todos os casos permitem que na fase de projeto seja possível elaborar um plano de gerenciamento completo. Mas sempre será possível decidir que arquitetura de gerenciamento será utilizada, que recursos de gerenciamento os equipamentos deverão possuir e que características o *software* de gerenciamento deverá suportar.

As decisões quanto ao gerenciamento tanto durante a fase de projeto quanto durante a fase de operação recaem sobre as necessidades de gerenciamento do administrador da rede.

Um estudo feito por Terplan em 1992 [Sta 93] lista os seguintes itens como as justificativas para investimentos na área de gerenciamento de redes:

- Controlar recursos estratégicos da corporação: Redes e recursos computacionais distribuídos são cada vez mais recursos vitais para a maior parte das organizações. Sem um controle efetivo, estes recursos não fornecem o retorno esperado pelos gerentes da corporação.

- Controlar complexidade: O contínuo crescimento do número de componentes de rede, usuários, interfaces, protocolos, e fornecedores traz a necessidade de gerenciamento daquilo que está ligado à rede e como os recursos de rede estão sendo utilizados.

- Melhorar o serviço: Usuários esperam que o serviço atual permaneça o mesmo ou melhore conforme as informações da organização e os recursos computacionais crescem ou tornam-se distribuídos.

- Balancear várias necessidades: A informação e os recursos computacionais de uma organização devem atender um espectro de usuários com várias aplicações em diferentes níveis de suporte, com requisitos específicos nas áreas de desempenho, disponibilidade e segurança. O gerente de redes deve alocar e controlar recursos para balancear estas várias necessidades.

- Reduzir o tempo em que a rede permanece paralisada: Conforme os recursos de rede de uma organização tornam-se mais importantes, os requisitos mínimos de disponibilidade da rede atingem os 100 por cento.

Com um projeto de rede que inclua a redundância apropriada, para ser tolerante a falhas, o gerenciamento de redes possui a tarefa indispensável de garantir alta disponibilidade da rede.

- Controlar custos: A utilização de recursos deve ser monitorada e controlada para que as necessidades essenciais dos usuários sejam satisfeitas a um custo razoável.

Este trabalho trata da necessidade de se gerenciar aplicações de rede Internet rodando nas estações cliente de rede. A motivação para este tema surgiu no período em que atuei como administrador de redes no LIICT - Laboratórios Integrados de Informática do Centro Tecnológico da UFSC.

Neste período, um de nossos principais problemas era a falta de equipamentos disponíveis para todos os usuários do laboratório. As filas de espera por equipamentos eram comuns, e um esquema de reservas de horário foi montado.

Entretanto, ao entrar nos laboratórios e observar o que os usuários faziam, pudemos perceber que uma boa parte destes preocupava-se em usar a rede para atividades menos nobres, tais como: conversas *on-line* (*chat*, IRC), acesso a páginas WWW não adequadas para um ambiente de pesquisa (páginas pornográficas, etc.), ou mesmo para alguns tipos de jogos multiusuário que utilizam a rede como suporte.

Surgiu então a idéia de criar uma aplicação que pudesse informar ao administrador, sentado em frente a uma estação de gerenciamento, quais aplicações de rede cada usuário estivesse utilizando. Além disso, seria interessante poder cancelar remotamente uma aplicação de um usuário, bem como a

possibilidade de a própria aplicação rodando nos computadores dos usuários alertar o administrador cada vez que certas aplicações fossem executadas.

Para resolver o problema, iniciei uma busca na Internet por soluções deste tipo, mas no máximo encontrei duas soluções, ainda em padronização, que somente atendiam parte das necessidades. A partir deste momento, a idéia de desenvolver este trabalho tornou-se viável, já que as soluções existentes não contemplavam as necessidades.

A estrutura do restante do trabalho está organizada como se segue. Inicialmente os modelos de gerenciamento OSI e Internet são apresentados brevemente nos capítulos 2 e 3, respectivamente. O capítulo 4 trata das necessidades do gerenciamento de aplicações de rede nas estações clientes da rede. O capítulo 5 discute os passos a serem seguidos para a implementação de novas MIBs SNMP. O capítulo 6 propõe uma extensão à MIB-II Internet para o gerenciamento de aplicações, enquanto o capítulo 7 aborda algumas alternativas para a implementação de agentes e gerentes SNMP. O capítulo 8 apresenta um relatório da implementação da MIB proposta, bem como os resultados obtidos. Por fim, o capítulo 9 apresenta as considerações finais e as conclusões.

Capítulo 2

MODELO DE GERENCIAMENTO OSI

2. Modelo de Gerenciamento OSI

De todas as áreas de padronização OSI, o conjunto de padrões para o gerenciamento de sistemas OSI é o mais volumoso e complexo. O primeiro padrão relacionado com gerenciamento de redes publicado pela ISO foi uma parte da arquitetura de redes OSI, que especificava um suporte para o seu gerenciamento.

Posteriormente, a ISO publicou um conjunto de padrões para o gerenciamento de redes. O CCITT (atual ITU-T) também publicou sua série de documentos equivalentes aos da ISO, chamada de série X.700.

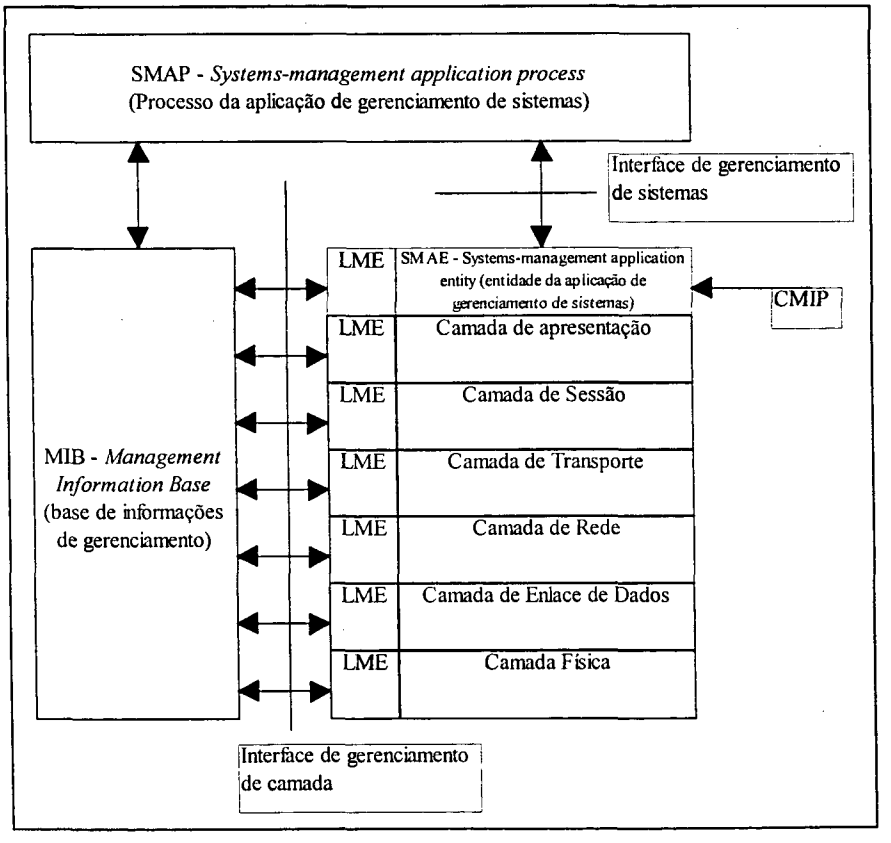
O modelo da arquitetura de gerenciamento OSI é mostrado na figura 2.1 [Sta 93]. Os elementos principais desta arquitetura incluem:

- Processo da aplicação de gerenciamento de sistemas (SMAP): Consiste no *software* local de um sistema que é responsável por executar as funções de gerenciamento de sistema no próprio sistema local (*host*, processador *front-end*, roteador, etc.). Este processo possui acesso a parâmetros e capacidades do sistema e pode portanto gerenciar todos os aspectos do sistema e coordenar-se com SMAPs em outros sistemas.
- Entidade da aplicação de gerenciamento de sistemas (SMAE): Esta entidade de nível de aplicação é responsável pela troca de informações de gerenciamento fim a fim com SMAEs em outros nós, especialmente com

o sistema que atua como o centro de controle da rede. Um protocolo de nível de aplicação padronizado, o protocolo comum para informações de gerenciamento (CMIP), é utilizado para este propósito.

- Entidade de gerenciamento de camada (LME): Junto a cada camada da pilha OSI, uma entidade de gerenciamento (LME) provê funções de gerenciamento de redes específicas àquela camada.
- Base de informações de gerenciamento (MIB): A coleção de informações para cada nó pertencente ao gerenciamento da rede.

Figura 2.1 - Modelo Arquitetural do Gerenciamento OSI



2.1 Áreas Funcionais do Gerenciamento OSI

Os padrões do gerenciamento OSI dividem o gerenciamento de redes em cinco áreas funcionais:

1. Gerenciamento de falhas: É o conjunto de ferramentas responsáveis por habilitar a detecção de falhas, isolamento e correção de operações anormais. O gerenciamento de falhas deve prover procedimentos para [Bri 92]:

   - detectar e reportar a ocorrência de falhas. Estes procedimentos permitem que um sistema gerenciado notifique seu gerente da detecção de uma falha, utilizando um protocolo padronizado para a notificação de eventos;
   - registrar o relatório de eventos recebido. Este registro pode então ser examinado e processado;
   - agendar e executar testes de diagnóstico, rastrear falhas, e iniciar a correção de falhas. Estes procedimentos podem ser invocados como o resultado da análise do registro de eventos.

2. Gerenciamento de contabilização: É formado por um conjunto de ferramentas responsáveis por permitir o estabelecimento dos pesos e custos associados ao uso dos recursos de rede. O gerenciamento de contabilização provê procedimentos para [Bri 92]:

   - informar usuários de custos envolvidos, utilizando *software* de relatório de eventos e de manipulação de dados;
   - permitir que os limites de uso dos recursos sejam estabelecidos;

- permitir que custos sejam combinados nos casos onde múltiplos recursos sejam utilizados para atingir um determinado objetivo.

3. Gerenciamento de configuração: Definido como um conjunto de ferramentas que viabilizam o controle sobre a configuração dos elementos que compõem o sistema de rede. O gerenciamento de configuração provê os seguintes procedimentos [Bri 92]:

- coletar e disseminar dados no que diz respeito ao estado corrente dos recursos. Mudanças iniciadas localmente ou mudanças devido a ocorrências imprevistas são comunicadas para a estação de gerenciamento através de um protocolo padronizado;
- ajustar e modificar parâmetros relacionados com componentes de rede e com o *software* da camada OSI;
- inicializar e finalizar objetos gerenciados;
- associar nomes com objetos e grupos de objetos.

4. Gerenciamento de desempenho: Constituído pelas ferramentas responsáveis por permitir o levantamento de dados estatísticos sobre a performance dos vários recursos da rede, para fins de otimização de seu funcionamento. Provê os procedimentos para [Bri 92]:

- coletar e disseminar dados no que diz respeito ao nível corrente de performance dos recursos;
- manter e examinar registros de performance para os propósitos de planejamento e análise.

5. Gerenciamento de segurança: É um conjunto de ferramentas que possibilitam ao gerente assegurar os recursos da rede contra as tentativas de acessos não

autorizados. O gerenciamento de segurança deve prover suporte aos serviços de [Sta 93]:

- facilidades de autorização;
- controle de acesso;
- criptografia e gerenciamento de chaves de acesso;
- autenticação;
- registros de segurança.

É importante ressaltar que o modelo OSI tem sido amplamente utilizado como referência para outras arquiteturas de gerenciamento, assim como a arquitetura de redes OSI o é.

Assim, as áreas funcionais de gerenciamento costumam ser o ponto mais disseminado do modelo OSI, e quando elas aparecerem no decorrer deste trabalho, seu significado é exatamente aquele descrito nesta seção.

Capítulo 3

GERENCIAMENTO INTERNET

3. Gerenciamento Internet

A Internet surgiu através do projeto ARPANET, iniciado em 1969 com o intuito de se estudar e construir, no território dos EUA, uma rede de computadores que atendesse os interesses do Departamento de Defesa (DoD) dos Estados Unidos. A continuação desta história é bastante conhecida; a ARPANET acabou virando Internet, e o resultado é que hoje encontramos computadores até mesmo nas residências com conexão à Internet.

Quando a arquitetura da Internet foi definida, adotando-se os protocolos TCP/IP como sua base, pouco se pensou em gerenciamento de redes. As pessoas que utilizavam a ARPANET estavam em geral envolvidas em algum aspecto do projeto ARPANET, e deixaram que os *experts* em protocolos de redes cuidassem do assunto [Sta 93].

O fato é que até o final da década de 70 não existiam ferramentas de gerenciamento para a Internet. A única ferramenta efetivamente utilizada na época era o protocolo ICMP [Sta 93]. O protocolo ICMP é capaz de transmitir mensagens entre roteadores ou *hosts* para um determinado *host* da rede, e detectar problemas no caminho. O ICMP está disponível em todos os dispositivos que suportam o protocolo IP.

Apesar das inúmeras coisas que se pode fazer com ICMP, desde seu uso simples através de comandos como o 'ping' até a construção de ferramentas que o

utilizem, o ICMP passou a não ser mais suficiente para todas as situações de gerenciamento da rede Internet. A Internet agora deixara de ser uma rede com dezenas de computadores para ser uma rede mundial com milhares de computadores a ela ligados.

Surgiram então três propostas para o gerenciamento da Internet [Sta 93]:

1. *High-level Entity Management System* (HEMS): esta solução era uma generalização do que tenha sido talvez o primeiro protocolo de gerenciamento da Internet, o protocolo de monitoração de *hosts* (*Host Monitoring Protocol* - HMP).
2. *Simple Network Management Protocol* (SNMP): uma versão avançada do SGMP - *Simple Gateway Monitoring Protocol* (protocolo de monitoramento de *gateways* simples).
3. *CMIP over TCP/IP* (CMOT): esta foi uma tentativa de incorporar, na máxima extensão possível, o protocolo (CMIP), serviços e estrutura da base de dados dos padrões de gerenciamento que estavam sendo padronizados pela ISO.

No início de 1988, o IAB, que regulamenta os padrões na Internet, adotou o SNMP como uma solução imediata, devido à sua simplicidade, e o CMOT como uma solução a longo prazo.

Para facilitar uma futura transição do SNMP para o CMOT, o IAB definiu que ambos os protocolos deveriam usar uma mesma estrutura da informação de gerenciamento (SMI), bem como a mesma base de informações de gerenciamento (MIB).

Logo percebeu-se que era impraticável os dois protocolos utilizarem as mesmas SMI e MIB. Enquanto o gerenciamento OSI trata os objetos gerenciados como entidades com alto grau de "inteligência", que possuem atributos, procedimentos

associados, e capacidade de notificações, além de outras características sofisticadas da tecnologia de orientação a objetos, o gerenciamento Internet assume que os objetos gerenciados são entidades simples, que possuem um conjunto de variáveis com algumas poucas características, capazes de serem apenas lidas ou escritas. Assim, o IAB permitiu que o desenvolvimento do CMOT utilizasse SMI e MIB distintas daquelas do SNMP.

3.1 Padronização da Arquitetura de Gerenciamento SNMP

Toda a padronização na Internet é desenvolvida dentro da organização chamada IETF. A supervisão técnica e de processos do IETF fica a cargo do IAB. O IAB, por sua vez, está hierarquicamente abaixo do ISOC, que é órgão administrativo de mais alto nível na Internet. O direcionamento técnico e a administração de processos do IETF fica a cargo do IESG, que é composto pelo *chairman* do IETF e pelos diretores das áreas funcionais das atividades do IETF.

Para compreender a evolução das padronizações referentes ao modelo de gerenciamento SNMP, é preciso entender como funciona o sistema de padronização do IETF.

Em cada área funcional das atividades do IETF, grupos de trabalho (WGs) são criados para completar tarefas específicas, que após seu término, são dissolvidos. A participação como membro de um grupo de trabalho é aberta a qualquer pessoa, que pode acompanhar as discussões nas listas de discussão dos grupos de trabalho. Os membros devem cooperar para que os objetivos do grupo de trabalho sejam atingidos.

O resultado do trabalho de um WG é composto de um ou mais documentos. Um documento que geralmente é disponibilizado para revisões, mas ainda não

publicado, recebe o nome de *internet-draft* (ID). Tal documento representa na verdade um trabalho em andamento. Um ID pode variar de um "*rough draft*" até um "*final draft*" antes de ser publicado pelo IETF. Quando publicado, o documento pode estar no processo de padronização (*standards track*) ou pode receber o status *Informational* ou *Experimental*. O primeiro passo de um documento no processo de padronização é o status *Proposed*. Este é seguido pelo status *Draft* e finalmente pelo status *Standard* ou *Full Standard*. Um documento que é trocado por uma versão atualizada recebe o status *Obsolete*. Um documento que é "aposentado" recebe o status *Historic* [Per 97].

Todo documento publicado pelo IETF recebe um número de RFC e é disponibilizado em um meio *on-line*. Aproximadamente de quatro em quatro meses, o IETF publica um documento chamado *INTERNET OFFICIAL PROTOCOL STANDARDS*, que especifica o status de todos os documentos publicados até aquele momento.

É importante ressaltar a diferença entre os status de documentos *Draft* e *internet-draft*. Enquanto o primeiro significa um documento oficialmente publicado pelo IETF, com um número de RFC associado, e portanto no processo de padronização, o último significa apenas um documento que informa sobre a situação de algum trabalho em andamento.

O conjunto de especificações que definem SNMP e suas funções e bases de dados relacionadas é bastante extensa e crescente. A seguir são discutidos os componentes da arquitetura de gerenciamento SNMP e seus respectivos padrões.

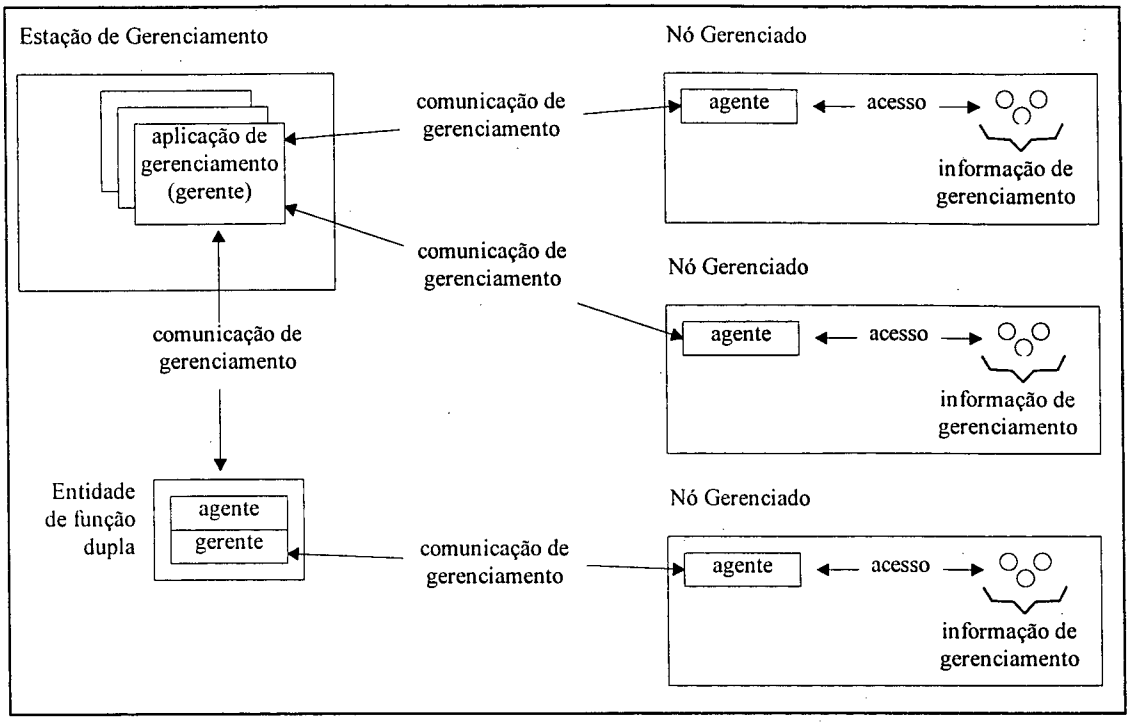
3.2 A Arquitetura do Gerenciamento de Redes Internet - SNMP

Hoje é comum o termo SNMP ser utilizado como referência à coleção de especificações para o gerenciamento de redes Internet, que inclui as definições do próprio protocolo SNMP, de uma base de dados e dos conceitos associados [Sta 93].

Uma rede gerenciada segundo o modelo de gerenciamento SNMP consiste dos seguintes elementos [Per 97]:

- um ou mais nós gerenciados, cada um contendo uma entidade de processamento chamada agente;
- pelo menos uma estação de gerenciamento contendo uma ou mais entidades de processamento chamadas aplicações de gerenciamento (ou gerentes);
- opcionalmente, entidades de processamento capazes de agir tanto no papel de gerente como de agente, chamadas de entidades de duplo comportamento (*dual-role entities*);
- informação de gerenciamento em cada nó gerenciado, que descreve a configuração, estado, estatísticas, e que controla as ações do nó gerenciado;
- um protocolo de gerenciamento, o qual os gerentes e agentes utilizam para trocar mensagens de gerenciamento.

Figura 3.1 - O modelo SNMP em uma rede gerenciada



A estação de gerenciamento é tipicamente uma máquina dedicada a este fim; mas é comum encontrar-se máquinas oferecendo outros serviços na rede. A estação de gerenciamento serve como a interface para a pessoa do gerente da rede.

A estação de gerenciamento deve possuir, no mínimo [Sta 93]:

- um conjunto de aplicações para análise de dados, recuperação de falhas, etc.;
- uma interface pela qual o gerente da rede possa monitorar e controlar a rede de computadores;

- a capacidade de traduzir os requisitos de gerente de redes em monitoração e controle efetivos dos elementos remotos na rede;
- uma base de dados de informações extraídas das MIBs de todas as entidades gerenciadas na rede.

As duas últimas características acima constituem os objetivos do SNMP [Sta 93]. As outras duas são características desejáveis que a maioria das plataformas de gerenciamento SNMP comerciais implementam, caso contrário não teriam muita utilidade para os gerentes de redes.

O outro componente ativo no gerenciamento Internet é o agente de gerenciamento. Normalmente o agente é implementado em elementos-chave da rede, como *hosts*, pontes, roteadores, *hubs*, *switches*, etc. Assim, cada um destes elementos pode ser gerenciado pela estação de gerenciamento. O agente responde a requisições enviadas pelo gerente. Estas requisições podem ser um simples pedido de uma informação, como também podem ser uma requisição para uma ação. Eventualmente, quando da ocorrência de determinados eventos, o agente pode notificar o gerente deste fato, sem que tenha sido solicitado para tal.

Os recursos da rede são representados como objetos. Cada objeto, na verdade, é uma variável que representa uma informação sobre o recurso gerenciado. O conjunto de todos os objetos de um recurso gerenciado é chamado de MIB. As variáveis da MIB podem ser vistas como pontos de acesso pelos quais um gerente pode atingir um agente. Normalmente, recursos gerenciados semelhantes (como roteadores) possuem o mesmo conjunto básico de variáveis (mas cada fabricante pode incluir outras informações adicionais sobre seu produto).

A estação de gerenciamento faz o monitoramento de um recurso gerenciado através da solicitação ao agente da leitura de valores da MIB do recurso gerenciado. O controle é feito através da solicitação ao agente da modificação de valores da MIB, e o agente deve reagir de acordo com as mudanças, tomando as ações previstas.

3.3 O Protocolo do Modelo de Gerenciamento SNMP

O protocolo de gerenciamento define o formato e significado da comunicação de gerenciamento que ocorre entre duas entidades SNMP. Existem duas versões do protocolo SNMP. A primeira versão é atualmente chamada de SNMPv1 (antigamente, apenas SNMP), e a versão mais recente é chamada SNMPv2. Infelizmente, a segunda versão do protocolo não ganhou, ao menos por enquanto, aceitação do mercado, mesmo contendo melhorias significativas em relação ao SNMPv1. Várias propostas foram feitas para modificar aspectos do SNMPv2 para que este receba aprovação do mercado [Per 97]. A tabela 3.1 lista as RFCs que atualmente definem o protocolo SNMP.

Tabela 3.1 - RFCs que definem o protocolo SNMP

Descrição	Publicada	RFC	Status
SNMPv1 Protocol	Ago. 1988	1067	Obsoleted by 1098
SNMPv1 Protocol (republicada)	Abril 1989	1098	Obsoleted by 1157
SNMPv1 Protocol (republicada)	Maio 1990	1157	Full Standard
Secure SNMP Protocol	Julho 1992	1352	Historic
SNMPv2 Protocol Operations	Maio 1993	1448	Obsoleted by 1905
SNMPv2 Transport Mappings	Maio 1993	1449	Obsoleted by 1906
SNMPv2 Protocol Operations (atualizada)	Jan. 1996	1905	Draft
SNMPv2 Transport Mappings (atualizada)	Jan. 1996	1906	Draft

As operações no protocolo SNMP estão restritas às funções de obtenção de informação de gerenciamento, modificação do valor da informação de gerenciamento, e comunicação de eventos. Cada classe (ou tipo) de informação de gerenciamento recebe uma identificação única, e é dita um objeto (*object*) ou tipo de objetos (*object type*). Uma instância de uma classe de informação de gerenciamento é dita uma variável, e recebe também uma identificação única baseada na identificação de sua classe e de sua identificação dentro da classe. Cada classe (ou tipo) de evento também recebe uma identificação única.

Existe uma operação de modificação de valores, chamada SET. Os operandos da operação SET são uma lista de pares. Cada par consiste da identidade de uma variável e do valor que se deseja dar para esta variável. Operações SET servem para configurar e controlar um sistema gerenciado [Per 97].

Há dois tipos de operação de obtenção de valores. Ambos os tipos de operações possuem como operandos uma lista de identificações de variáveis, e retornam os valores das variáveis, na forma de uma lista de pares, cada par contendo a identificação de uma variável e seu valor corrente. O primeiro tipo de operação de obtenção de valores requer que as identificações usadas como operandos coincidam exatamente com as identificações das variáveis cujos valores são retornados. Existe uma operação deste tipo, chamada GET, que é usada quando a identificação de cada variável desejada é conhecida. O outro tipo de operação utiliza as identificações de variáveis usadas como operandos como uma aproximação da identificação para uma variável. Cada identidade retornada corresponde à primeira variável acessível cuja identidade seja maior do que a identidade fornecida como operando. Há duas operações deste tipo, chamadas GETNEXT e GETBULK. Estas operações são utilizadas quando a identificação de uma classe de informação de gerenciamento é conhecida, mas as instâncias desejadas dentro daquela classe não o são. Além disso, estas operações podem ser

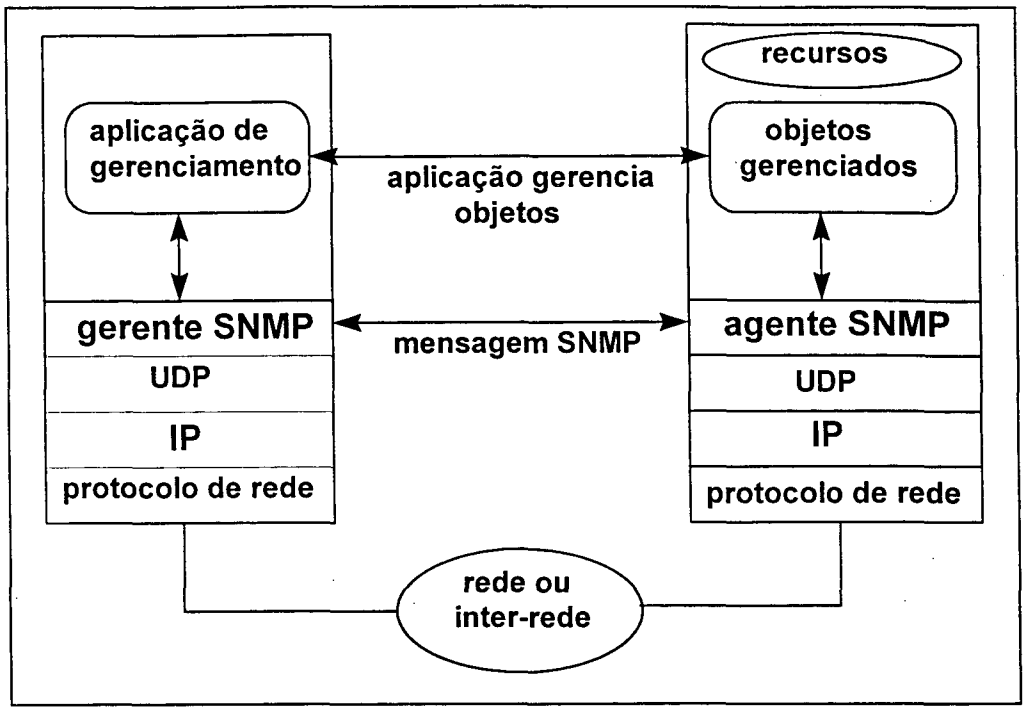
usadas para determinar que classes de informação de gerenciamento possuem instâncias acessíveis.

Existem duas operações de comunicação de eventos, chamadas TRAP e INFORM. Cada uma especifica uma lista de pares, cada par contendo a identidade de uma variável e seu valor correspondente. A operação INFORM só existe no SNMPv2, e é mais segura, pois possui confirmação, enquanto o TRAP não possui confirmação. Estas operações são usadas para informar a ocorrência de eventos em um sistema gerenciado para uma lista de gerentes configurados para receber eventos daquele sistema gerenciado.

Das operações descritas acima, apenas as operações GET, GETNEXT, SET e TRAP fazem parte da primeira versão do protocolo. As outras duas, GETBULK e INFORM, foram introduzidas na segunda versão do protocolo.

Operações SNMP ocorrem através da troca de mensagens sobre um serviço de transporte de mensagens. O formato de mensagens é definido utilizando-se um subconjunto da notação ASN.1. Esta notação é independente da representação de dados particular de um sistema, e uma mensagem deve ser primeiramente convertida para uma seqüência de octetos (*bytes*) antes de ser transmitida. SNMP usa um subconjunto das regras BER para definir o formato das mensagens codificadas (ou serializadas). A comunicação entre o gerente e um agente é ilustrada na figura 3.2:

Figura 3.2 - Comunicação gerente-agente na arquitetura Internet



Uma mensagem SNMP contém informações administrativas e uma PDU SNMP. A PDU identifica o tipo de mensagem, e contém campos de controle, que dependem do tipo da mensagem, e uma seqüência de pares. O primeiro elemento de cada par identifica uma informação de gerenciamento, enquanto o segundo especifica o valor de uma informação de gerenciamento.

3.4 A Definição da Informação de Gerenciamento e dos Eventos

O modelo da informação de gerenciamento SNMP também é descrito nos padrões, como mostra a tabela 3.2. O modelo de informação define os tipos de dados utilizados, e as regras para especificar tipos (classes) de informação de

gerenciamento. Também incluído está o modelo para os eventos e as regras para a especificação de classes (ou tipos) de eventos.

Originalmente, todas estas definições estavam especificadas em um único documento chamado *Structure of Management Information* (SMI) — Estrutura da Informação de Gerenciamento. Atualmente, tais definições estão especificadas em vários documentos, sendo o principal chamado SMI. Assim, o termo SMI pode tanto significar um documento particular do IETF, como o conjunto de regras definindo informação de gerenciamento e eventos no modelo SNMP.

Tabela 3.2 - RFCs definindo a Informação de Gerenciamento e Eventos

Descrição	Publicada	RFC	Status
SMIv1	Ago. 1988	1065	Obsoleted by 1155
SMIv1 with typos fixed	Maio 1990	1155	Full Standard
SMIv1 Concise MIB format	Mar. 1991	1212	Full Standard
SMIv1 Trap formats	Mar. 1991	1215	Informational[1]
SMIv2	Maio 1993	1442	Obsoleted by 1902
SMIv2 Textual Conventions	Maio 1993	1443	Obsoleted by 1903
SMIv2 Conformances	Maio 1993	1444	Obsoleted by 1904
SMIv2 (atualizada)	Jan. 1996	1902	Draft
SMIv2 Textual Conventions (atualizada)	Jan. 1996	1903	Draft
SMIv2 Conformances (atualizada)	Jan. 1996	1904	Draft

Um tipo ou classe de informação de gerenciamento (*object type*) deve possuir um tipo de dado, o máximo acesso permitido, sua identificação única, como as instâncias são definidas, e a sua semântica (ou comportamentos).

[1] Este documento não foi incluído no processo de padronização para o qual sua funcionalidade pertence.

O esquema de identificação dos tipos de objetos utilizado no SNMP é o mesmo utilizado em ASN.1 para única e universalmente identificar itens. Um identificador neste esquema é chamado de *object identifier* (OID) — identificador de objetos. A associação permanente de um OID a um item é chamada de registro. Uma vez que um item tenha sido registrado, nenhum outro item poderá ser registrado com o mesmo OID, as características do item não podem ser mudadas, e o item registrado não pode ser removido.

As definições para informações de gerenciamento, eventos, e requisitos de implementação associados são especificados em documentos chamados *Management Information Base* (MIB) *specifications*. Estas especificações contém descrições verbais e também descrições em um formato legível por computadores. As descrições legíveis por computadores, chamadas de *MIB information modules* (ou *MIB modules*), são escritas em um subconjunto adaptado da notação ASN.1 [Per 97].

Assim, a notação de definição de MIBs SNMP é uma notação própria, e não ASN.1 puro. Esta notação utiliza os módulos ASN.1 que definem macros, descrições textuais modificando e qualificando as macros e elementos de ASN.1, e outros ítens de uso comum. Programas de computador que interpretam a notação de módulos de MIB (*MIB module language*) são chamados de compiladores de MIB (*MIB compilers*).

3.5 Um Conjunto Básico de Informação de Gerenciamento

O gerenciamento SNMP requer que cada agente possua um conjunto mínimo de informações de gerenciamento. O primeiro conjunto de informações proposto definia um conjunto padrão de informação de gerenciamento para sistemas que implementam o conjunto de protocolos Internet. Esta especificação foi chamada

de MIB-I. Os ítens incluídos foram aqueles presentes em todos os sistemas, e permitiam o gerenciamento de configurações e de falhas. A quantidade de ítens definida foi relativamente pequena, e nenhum item era específico demais de forma que comprometesse o desempenho de algum dispositivo particular.

Alguns anos depois, baseando-se em experiências com implementações, o grupo de trabalho responsável pela MIB Internet padrão acrescentou alguns ítens, publicando um novo documento atualizado chamado MIB-II.

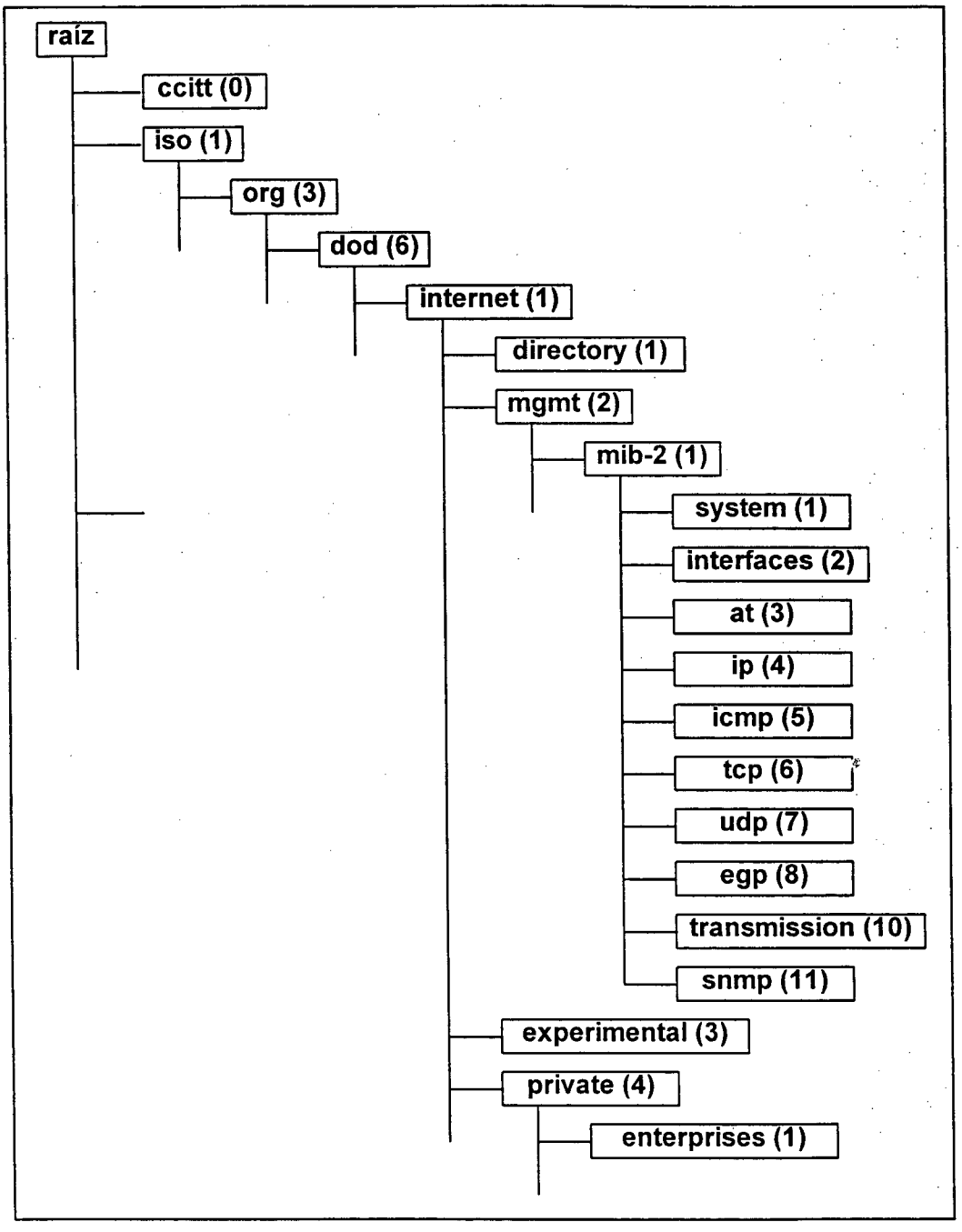
A MIB-II possui informações sobre o próprio sistema (grupo *system*), sobre as interfaces de rede (grupo *interfaces*), e sobre os protocolos básicos Internet (*grupos* IP, TCP, UDP, ICMP, EGP). A figura 3.3 ilustra a árvore de objetos da MIB-II Internet, organizada hierarquicamente.

O gerenciamento SNMP hoje em dia é utilizado para gerenciar muitas outras coisas além de computadores e dispositivos de interconexão de redes. Assim, é possível encontrar dispositivos contendo agentes SNMP que não implementam alguns grupos da MIB-II, bem como é possível encontrar dispositivos que implementem outras informações de gerenciamento.

Para tentar padronizar as informações de gerenciamento de diversos tipos, módulos de MIB foram escritos, como por exemplo uma MIB para gerenciar dispositivos do tipo UPS (*Uninterruptible Power Supply*), descrita na RFC 1628.

Mesmo os módulos de MIB para outros protocolos (como o FTP ou o HTTP) não foram incluídos na MIB-II. O grupo de trabalho responsável pela MIB Internet decidiu deixar novos módulos de MIB separados em vários documentos.

Figura 3.3 - Grupos de objetos da MIB-II



A tabela 3.3 lista os documentos que definem o conjunto básico de informações de gerenciamento.

Tabela 3.3 - O conjunto básico de Informações de Gerenciamento

Descrição	Publicada	RFC	Status
MIB-I	Ago. 1988	1066	Obsoleted by 1156
MIB-I with typos fixed	Maio 1990	1156	Obsoleted by 1158
MIB-II	Maio 1990	1158	Obsoleted by 1213
MIB-II updated to the concise format	Mar. 1991	1213	Full Standard
Interfaces MIB	Jan. 1994	1573	Proposed
SNMPv2 Core	Maio 1994	1450	Obsoleted by 1907
SNMPv2 Core (atualizada)	Jan. 1996	1907	Draft

Observando a árvore de registro de objetos Internet, na figura 3.3, pode-se verificar que a MIB-II está sob o OID 1.3.6.1.2.1. A SMI define ainda quatro nós dentro do ramo internet [Rose 90].

1. *directory*: Esta sub-árvore é reservada para uso futuro com o serviço de diretórios OSI (X.500);
2. *mgmt*: Esta sub-árvore é reservada para objetos definidos em documentos aprovados pelo IAB;
3. *experimental*: Esta sub-árvore é reservada para identificar objetos utilizados em experimentos Internet;
4. *private*: Esta sub-árvore é usada para identificar objetos definidos unilateralmente pelos fabricantes de produtos gerenciáveis SNMP.

3.6 Monitoração Remota - RMON MIB

A mais importante adição ao conjunto de padrões SNMP foi o RMON MIB (*Remote Monitoring MIB*), descrito no documento RFC 1757 [Wal 95].

Utilizando apenas informações da MIB-II, o gerente de redes nunca tinha uma idéia do tráfego das redes nas quais os recursos gerenciados estavam instalados. Isto porque a MIB-II presente nas entidades de gerenciamento fornece apenas informações locais ao próprio recurso gerenciado onde o agente está executando.

Em uma grande rede, com vários nós gerenciados e não gerenciados, fica praticamente impossível inspecionar variáveis da MIB-II de todos os agentes da rede para se ter uma idéia de seu tráfego.

Outro problema com os agentes SNMP tradicionais, é que estes não são capazes de analisar seus próprios dados, e por exemplo, serem programados para enviarem notificações quando certos limiares nas variáveis forem atingidos. Isto força a estação de gerenciamento a ficar periodicamente inspecionando (*polling*) as variáveis das diversas entidades de gerenciamento, podendo gerar um tráfego excessivo na rede.

Assim, fazia-se necessária a presença de um monitor instalado na rede que se desejava estudar, que pudesse coletar informações sobre toda a sub-rede, e eventualmente ser programado para enviar alarmes sobre a ocorrência de eventos. O monitor pode ser tanto um dispositivo dedicado à captura de dados e sua análise, como pode estar implementado em estações de trabalho, servidores, roteadores, *hubs*, etc. Este monitor eventualmente enviaria informações para a estação de gerenciamento, quando fosse conveniente, através da rede. Dessa forma, fica caracterizada a monitoração remota, onde a estação de gerenciamento deixa a cargo dos monitores remotos algumas tarefas específicas.

O RMON é essencialmente a definição de uma extensão à MIB Internet. Através da escrita em variáveis desta MIB, o gerente pode programar o monitor, que coleta dados e armazena-os em tabelas que podem ser obtidas pelo gerente a qualquer momento. Além disso, o monitor pode ser programado para avisar da ocorrência de determinadas situações, quando o gerente então deverá obter os valores dos dados relevantes no monitor.

O RMON está definido no RFC 1757 (que atualiza a RFC 1271) e foi projetado para atingir os seguintes objetivos [Wal 95]:

- operação *off-line*: o monitor remoto coleta e acumula estatísticas que podem ser recuperadas pela estação gerente a qualquer momento;
- monitoramento preemptivo: o monitor está sempre ativo, continuamente rodando diagnósticos e armazenando dados;
- detecção e alerta de problemas: o monitor pode verificar continuamente determinadas condições, e comunicá-las quando ocorrerem;
- resumo dos dados: o monitor é capaz de realizar algum processamento, como descobrir os 10 hosts mais ativos na rede;
- múltiplos gerentes: o monitor deve suportar várias estações gerentes.

Nem todas as implementações de monitores remotos precisam atingir todos os objetivos, mas o RMON MIB definida em [Wal 95] provê a base para o suporte de todos estes objetivos.

Capítulo 4

NECESSIDADES DE GERENCIAMENTO DE APLICAÇÕES DE REDE

4. Necessidades de Gerenciamento de Aplicações de Rede

Com a popularidade das redes de computadores hoje em dia, cada vez mais usuários utilizam diferentes aplicações de rede. O acesso à Internet hoje já não é mais considerado um privilégio de poucos, e há quem diga que no mundo moderno não se pode mais viver sem uma conexão à Internet.

Dentro das organizações, o uso mais intensivo de aplicações de rede mudou completamente o padrão de uso das redes de computadores. Se antigamente as pessoas utilizavam a rede para consultas esporádicas a bancos de dados ou para o uso específico de um recurso caro e compartilhado na rede, hoje as redes de computadores estão substituindo os meios de comunicação tradicionais, como o próprio telefone.

Basta tomar como exemplo o caso da Universidade Federal de Santa Catarina (UFSC). Hoje existem mais computadores na Universidade interligados em rede do que terminais telefônicos, e esta diferença tende a aumentar.

Com tanta utilização das redes de computadores nas mais diversas tarefas, é óbvio que o tráfego dentro da rede aumenta. Mas saber qual é o volume de tráfego não é suficiente para saber "o quê" os usuários estão fazendo.

É interessante conhecer quais são as aplicações que mais consomem largura de banda da rede, e que tipos de serviços de informações os usuários mais utilizam.

Nos dias de hoje, é comum pensarmos que o maior tráfego dentro da rede seja gerado pelos serviços de informação disponíveis via WWW (*World Wide Web*). Entretanto, nada nos diz com certeza de que este serviço seja o maior devorador de largura de banda. Outros serviços essenciais como o terminal virtual (*telnet*), transferência de arquivos (*ftp*) e o correio eletrônico (*e-mail*) estão sendo utilizados a todo o momento.

Assim, para conhecer que aplicações são responsáveis por quais fatias de tráfego na rede, fazem-se necessários mecanismos que permitam ao gerente de rede obter estas informações. As aplicações de gerenciamento tradicionais somente são capazes de obter informações sobre o tráfego total de cada máquina, com detalhamento dos protocolos de transporte e de níveis inferiores.

Existem no mercado produtos que conseguem informar o tráfego recebido e enviado pelos serviços executados nas máquinas servidoras, mas isto não é suficiente. Nem sempre o usuário está acessando máquinas servidoras da própria rede da organização. Portanto, medir tráfego nos servidores não mostra todo o tráfego das aplicações dos usuários, pois parte dele é direcionado para servidores em outras redes, fora da organização.

O gerenciamento de cada máquina cliente da rede poderia fornecer informações sobre o que os usuários estão usando, e dentro de cada rede seria possível ter uma noção dos padrões de tráfego das aplicações.

Dessa forma, o gerenciamento de aplicações de rede pode ter dois enfoques:

- Gerenciamento das estações servidoras de aplicações, isto é, com enfoque nos serviços. Para esta situação já existe uma tentativa, proposta em janeiro de 1994 sob a forma da RFC 1565 [Kil 94] - *Network Services Monitoring MIB*. Esta proposta consiste de um módulo de MIB, em

conformidade com a SMIv2, e que acrescenta 24 novos objetos para a monitoração de serviços de rede;

- Gerenciamento das estações clientes, com enfoque nas atividades dos *softwares* clientes. Não existe ainda nenhuma proposta no IETF neste sentido, mas boa parte do trabalho pode ser aproveitada da RFC 1565 [Kil 94].

Segundo a RFC 1565 [Kil 94], o gerenciamento efetivo de serviços Internet deve satisfazer dois requisitos:

1. Deve ser possível monitorar um grande número de componentes (tipicamente para uma grande organização); e
2. O monitoramento de aplicações deve ser integrado ao gerenciamento de redes genérico.

Para satisfazer estes dois requisitos, o módulo MIB proposto na RFC 1565 não inclui nenhum objeto que permita o controle dos serviços de rede em execução, para que a implementação seja a mais fácil possível. O monitoramento dos serviços de rede está integrado ao gerenciamento de redes genérico através do uso do modelo de gerenciamento SNMP.

Entretanto, o gerenciamento de aplicações nos clientes de rede pode exigir algumas situações onde sejam necessárias funções de controle. Assim, um agente construído para este fim deve ser capaz de:

- identificar as aplicações clientes de rede em execução;

- monitorar as conexões ativas das aplicações;
- coletar estatísticas de conexões e informações relacionadas;
- controlar o estado operacional das conexões;
- controlar o estado operacional de cada aplicação, sendo possível, por exemplo, suspender uma aplicação, restaurá-la ao estado normal, abortá-la, etc.;
- ser programado para informar a ocorrência de eventos relativos a conexões de rede.

Um agente SNMP com estas características estaria envolvido em três áreas funcionais (segundo o modelo OSI) do gerenciamento de redes:

1. Desempenho: a coleta de estatísticas através das funções de monitoração permite o conhecimento do uso que os usuários fazem da rede;
2. Configuração: as funções de controle permitem que sejam configuradas nas máquinas clientes quais serviços de rede podem ou não ser utilizados. Tais configurações têm efeito no desempenho da rede;
3. Segurança: através das funções de controle e de comunicação de eventos, a estação de gerenciamento pode ser notificada da tentativa de uma estação cliente conectar a *hosts* considerados não seguros, por exemplo.

As máquinas clientes que se deseja gerenciar deverão estar executando um agente SNMP que implemente uma MIB com as funções mencionadas acima. Atualmente, o ambiente de trabalho mais comum utilizado pelos usuários é uma máquina PC com sistema operacional Windows 95. Isto pode nos levar a

considerar dois aspectos sobre segurança. O primeiro deles refere-se à possibilidade do usuário desativar o agente SNMP, uma vez que suas funções de controle podem limitar o uso da rede, muitas vezes contra a vontade do usuário. Este problema tende a ser resolvido naturalmente, à medida que o sistema operacional utilizado nas máquinas dos usuários passe a implementar funções de segurança que impeçam o usuário de executar operações que normalmente somente o administrador da rede poderia fazer, como desativar o agente SNMP.

O segundo problema refere-se à própria segurança do protocolo SNMP. Toda a segurança do protocolo SNMP atualmente está baseada em *communities*. Outros mecanismos avançados foram propostos para o SNMPv2. Entretanto, tais aspectos de segurança mostraram-se o ponto mais polêmico e de maior dificuldade de implementação, e até o momento ainda não existe um consenso quanto às novas políticas de segurança para o modelo SNMP. Os problemas com segurança que podem surgir num ambiente que implemente a MIB proposta neste trabalho são os mesmos de qualquer ambiente que possua agentes com MIBs contendo variáveis de controle importantes. É comum muitos administradores e gerentes de redes jamais utilizarem funções de controle simplesmente desabilitando qualquer *community* com permissão *read-write* em todos os seus dispositivos que suportam SNMP.

Este segundo problema com segurança somente pode ser resolvido através de um modelo administrativo mais elaborado para a autenticação entre estações de gerenciamento e agentes SNMP, o que deve surgir em breve nas atividades dos grupos de trabalho do IETF ou da comunidade científica envolvida com a Internet.

Capítulo 5

CONSTRUINDO NOVAS MIBS

5. Construindo Novas MIBs

Criar novas definições de informação de gerenciamento é uma tarefa que deve tornar-se mais comum à medida em que novas tecnologias surgem e à medida que a experiência com as tecnologias existentes é acumulada.

Definições de novos tipos de informação de gerenciamento podem ser elaboradas pelos engenheiros que criam a nova tecnologia, pelos engenheiros que desenvolvem as aplicações para gerenciar a tecnologia, e pelos gerentes de *marketing* de produtos que representam as necessidades dos clientes (e usuários) do produto. Uma definição é bem sucedida se ela é largamente difundida e se agrega valor a sistemas de gerenciamento. O tempo tem mostrado que o sucesso inclui os seguintes ingredientes [Per 97]:

- A definição deve ser escrita e disponibilizada amplamente a um custo insignificante;
- Ela deve ser implementável por desenvolvedores de agentes;
- Ela deve ser implementável por desenvolvedores de aplicações de gerenciamento;
- As implementações de agentes e aplicações devem ser interoperáveis umas com as outras;

- Os resultados obtidos da aplicação devem possuir um valor maior do que o custo do sistema (e da rede) em termos de recursos necessários para executar a aplicação.

Para atingir estes objetivos, a pessoa ou grupo que esteja escrevendo as novas definições deve possuir uma certa experiência e habilidade com o assunto, incluindo [Per 97]:

- Conhecimento de como escrever definições válidas;
- Conhecimento da tecnologia a ser gerenciada;
- Compreensão de como a tecnologia precisa ser gerenciada;
- Compreensão dos custos de implementação de várias técnicas usadas para escrever definições tanto em agentes como em aplicações de gerenciamento;
- A habilidade de concisa e precisamente escrever as definições de forma que elas serão interpretadas com o mesmo significado pela maioria dos leitores (pessoas) das definições;
- A habilidade de escrever as definições em um nível de abstração de forma que elas possam ser estendidas e aplicadas para áreas para as quais não se tinha pensado, e ainda em um nível de abstração que possa ser eficientemente entendido e aplicado à geração atual de tecnologia.

Novos objetos podem ser adicionados a uma MIB SNMP através de uma entre três maneiras diferentes [Sta 93]:

1. A sub-árvore **mib-2** pode ser incrementalmente expandida ou completamente trocada por uma nova revisão (o que é pouco provável). Um exemplo de expansão da MIB-II é o RMON MIB [Wal 95];
2. Uma MIB experimental pode ser construída para uma aplicação particular. Tais objetos podem ser subseqüentemente movidos para a sub-árvore **mgmt**. Exemplos disso são as várias MIBs específicas a tipos de enlaces que tem sido definidas, como a IEEE 802.5 *token ring* LAN (RFC 1231);
3. Extensões privadas podem ser adicionadas à sub-árvore **private**. Uma que está documentada em uma RFC é a MUX (*multiplexer*) MIB (RFC 1227).

Este capítulo discute os principais aspectos da construção de novas MIBs SNMP, sem no entanto entrar nos detalhes das sintaxes da SMIv1 e da SMIv2 do modelo SNMP.

5.1 SNMPv1 ou SNMPv2?

Uma das primeiras perguntas que surgem para o desenvolvedor que vai definir um novo módulo MIB pela primeira vez, é quanto à sintaxe de SMI que deve ser usada: SNMPv1 ou SNMPv2?

Atualmente, toda MIB desenvolvida para um grupo de trabalho no IETF deve ser escrita na sintaxe da SMIv2. Entretanto, se a MIB estiver sendo desenvolvida para um fornecedor particular, a SMIv1 pode ser utilizada sem problemas.

Uma das maiores vantagens de adotar o SNMPv1 é sua larga aceitação no mercado, e portanto uma MIB em conformidade com a SMIv1 será lida corretamente por uma gama maior de produtos e de pessoas. Além disso, várias

empresas investiram durante os anos que se passaram em ferramentas de desenvolvimento que ainda não suportam a SMIv2. Outro ponto a favor é que o SNMPv1 é o único protocolo de gerenciamento que atualmente é um *Internet Standard.*

Entretanto, está claro que o protocolo SNMPv1 possui deficiências e que o caminho de evolução mais direto é o SNMPv2. À medida que as plataformas de gerenciamento começarem a suportar completamente o SNMPv2, os fornecedores sentir-se-ão pressionados a converterem suas MIBs para a sintaxe da SMIv2, bem como a desenvolverem novas MIBs em conformidade com a SMIv2. A SMIv2 possui uma sintaxe muito mais rica e mais precisa para definir MIBs, de forma que os desenvolvedores começarão a perceber que eles não ficarão mais "confinados" à sintaxe da SMIv1 [Per 97]. A sintaxe da SMIv2, apesar de alguns pontos ainda duvidosos, é uma grande melhoria à sintaxe da SMIv1.

Uma maneira de resolver este dilema é desenvolver MIBs para a sintaxe mais atualizada da SMIv2, e quando necessário, convertê-las para a sintaxe da SMIv1. Existem diversas ferramentas para este propósito, e algumas o muito bem. Converter MIBs SMIv2 para a sintaxe da SMIv1 é uma tarefa relativamente simples [Cas 96f].

É importante ressaltar que nada impede que uma MIB em conformidade com a SMIv2 seja utilizada em ambiente onde o protocolo de gerenciamento utilizado pelos agentes e gerentes seja o SNMPv1. A sintaxe de definição da MIB não está amarrada ao protocolo, já que os grupos de trabalho do IETF evitaram construções que impedissem o uso de uma MIB SMIv2 sobre o protocolo da primeira versão.

O documento do IETF descrevendo a coexistência entre o SNMPv1 e o SNMPv2 é atualmente a RFC 1908 [Cas 96f]. Este documento descreve as "regras" a serem seguidas para converter MIBs SMIv2 para a SMIv1, e vice-versa. Converter antigas MIBs SMIv1 para SMIv2 pode ser também uma alternativa interessante à medida que ferramentas que suportem a SMIv2 surgirem.

5.2 Modelagem de Objetos

Antes de desenvolver uma MIB, é necessário fazer uma análise dos objetos relacionados. É importante ressaltar a diferença entre a análise de objetos e a análise orientada a objetos.

A modelagem de objetos não necessariamente precisa seguir o modelo orientado a objetos. No caso de SNMP, os objetos da MIB estão organizados hierarquicamente em uma árvore, mas não seguem o modelo orientado a objetos. A maneira mais fácil de enxergar este fato é que SNMP não suporta de nenhuma forma o mecanismo de herança, sobre o qual (entre outros) o conceito de orientação objetos está fundamentado.

Durante o projeto de uma MIB, a fase de modelagem de objetos é tão importante quanto as fases que antecedem a programação de um sistema em determinada linguagem de programação. Sem um planejamento adequado, o resultado final jamais será o melhor possível.

Quando abordamos um novo projeto de MIB, é importante ter em mente algumas categorias de objetos, para pensar no problema de forma organizada [Per 97]:

- ações: controlam um sistema. Objetos do tipo ação são usados para permitir a execução de tarefas bem definidas, como: reinicializar um dispositivo, desativar um serviço de rede, etc.;

- estatísticas: fornecem informação útil sobre o que aconteceu no sistema desde o início de um certo intervalo de tempo. Estatísticas podem incluir ítens como: o número de pacotes transmitidos por uma interface de rede, ou o número de vezes que um usuário se conectou a uma máquina. As estatísticas são discutidas brevemente mais adiante neste capítulo;

- estados: indicam o estado corrente de um sistema. Estados podem incluir ítens como: uma placa está inicializando, ou um ar-condicionado está com defeito;

- componentes: ajudam a descrever conjuntos de dispositivos físicos e lógicos, ou serviços que estão sob controle do agente SNMP. Por exemplo, as placas presentes em um sistema de chassis com múltiplos *slots*, ou ainda os nomes dos serviços ativos em um servidor de arquivos;

- atributos: são as propriedades de um objeto modelado, que descrevem coisas relativamente estáticas sobre um dispositivo ou serviço. Exemplo: o número de portas em um *hub Ethernet*, ou a pessoa a chamar em caso de falha de um dispositivo, ou ainda que tipo de CPU está instalada no sistema.

O seguinte método proposto por Perkins e McGinnis [Per 97] pode ser utilizado: uma vez que todos os objetos estejam agrupados em suas categorias correspondentes, podem ser agrupados em uma tabela-resumo. Esta tabela deve conter uma linha para cada componente identificado, sendo que os sub-

componentes de um componente devem vir nas linhas que o seguem. Cada linha deve conter as seguintes colunas:

- componente: o nome do componente;
- cardinalidade: a quantidade que é possível existir no sistema para o componente;
- atributos: dos atributos identificados anteriormente, quais correspondem ao componente;
- estatísticas: das estatísticas identificadas anteriormente, quais correspondem ao componente;
- estados: dos estados identificados anteriormente, quais correspondem ao componente.

As ações devem ser representadas sob a forma de estados. A modelagem de ações em SNMP é discutida mais adiante neste capítulo.

Para converter a tabela resultante para a sintaxe de uma MIB SNMP, deve-se seguir as seguintes regras:

- sub-componentes com uma cardinalidade maior que 0 (zero) devem fazer parte de uma tabela;
- estatísticas que representam valores que sempre crescem correspondem a tipos contadores: *Counter* (SMIv1) ou *Counter32* (SMIv2);
- estatísticas representando o valor mais alto ou mais baixo de determinado item são inteiros: *INTEGER* (SMIv1), ou *Integer32* e *Unsigned32* (SMIv2);

- estados que representam estágios de operação discretos utilizam um tipo inteiro enumerado (*INTEGER*), sendo que cada valor enumerado corresponde a um dos estados mencionados;
- estados que possuem valores flutuantes constituem tipos "termômetro": *Gauge* (SMIv1) ou *Gauge32* (SMIv2);
- atributos de um objeto podem tomar duas formas: *OCTET STRINGS* podem ser usados para representar descrições textuais ou dados binários, e tipos inteiros são usados para representar quantidades mensuráveis.

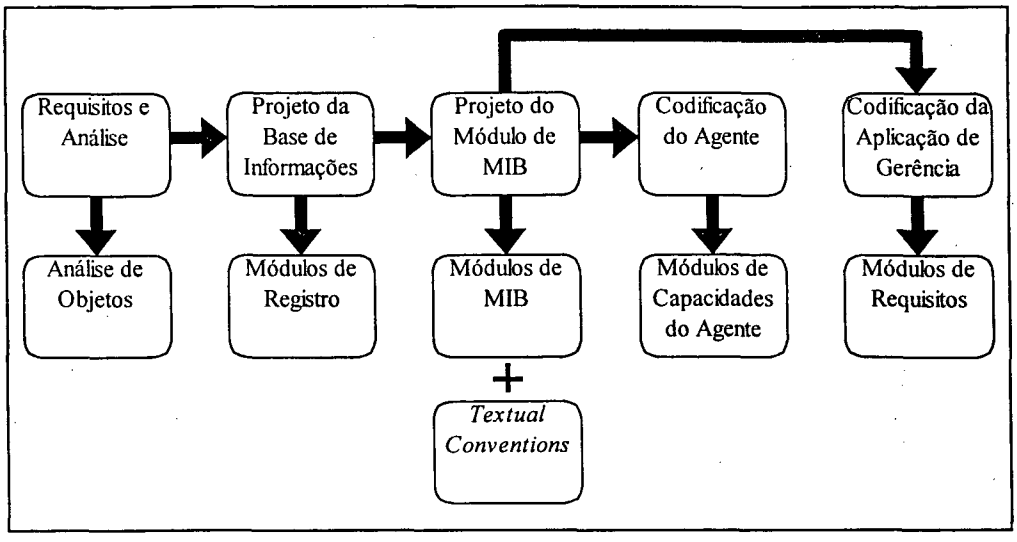
5.3 Um Guia para o Desenvolvimento de MIBs

A partir deste momento, o termo MIB será utilizado como o conjunto de documentos que definem a infraestrutura, objetos, e notificações para gerenciar um ou mais produtos (ou ítens).

O diagrama mostrado na figura 5.1 [Per 97] apresenta os passos necessários para projetar uma MIB. Abaixo de cada etapa, indicados pelas setas, estão os documentos gerados em cada uma delas.

Em cada estágio, um conjunto de documentos precisa ser produzido. O primeiro e mais importante estágio é a fase de requisitos e análise. Esta fase gera um documento descrevendo os objetos gerenciáveis que serão modelados na MIB [Per 97]. Este documento corresponde aos tipos de objetos e à tabela-resumo apresentados na seção anterior.

Figura 5.1 - Passos no Desenvolvimento de uma MIB



A segunda fase consiste em estabelecer a base geral de informações, e decidir como organizar as definições de objetos em um ou mais módulos de MIB. Esta fase deve gerar um módulo que apresente um *layout* da infraestrutura de identificadores de objetos (OID) [Per 97].

A terceira fase, projeto do módulo de MIB, é onde realmente o trabalho começa. Nesta fase são criadas as definições de objetos gerenciados que aparecerão no módulo de MIB. As duas últimas fases, de implementação do agente e da aplicação de gerenciamento, podem parecer não fazer parte do processo de escrever uma MIB, mas elas produzem módulos de informação que consistem em respostas à MIB recém-criada [Per 97].

Existe ainda uma sexta fase não mencionada que consiste na manutenção da MIB, incluindo edições, adições e remoções de objetos. A manutenção é uma

tarefa fácil, uma vez que foi estabelecida uma infraestrutura antes do início da definição dos objetos [Per 97].

5.3.1 Tipos de Módulos

As regras especificadas nos documentos da SMI são razoavelmente claras quanto aos mecanismos para produzir módulos de MIB, mas elas não fornecem nenhuma idéia do que se deve incluir nestes módulos. Algumas pessoas erradamente pensam que módulos de MIB contém somente definições de objetos gerenciados. Enquanto isto é verdade para a grande maioria de módulos de MIB criados nos últimos anos de desenvolvimento do SNMPv1, certamente isto não é uma obrigatoriedade. É perfeitamente correto escrever um módulo de MIB que não contenha nenhuma definição de um objeto gerenciado. Tal tipo de módulo tem sido chamado de *Information Module*, e somente pode conter os seguintes itens [Per 97]:

- registros de OIDs;
- convenções textuais (*textual conventions*) para serem compartilhadas entre outros módulos;
- requisitos de implementação;
- perfis de implementação;
- *traps* SNMPv1 ou notificações SNMPv2.

Por outro lado, um módulo de MIB pode conter quaisquer destes ítens, além da definição de objetos gerenciados. O ponto chave é que as definições podem estar

divididas em módulos menores, ao invés de um único grande módulo. Quando agrupados, estes módulos formam uma base de informações (*information framework*). Isto permite que todo o conjunto seja de mais fácil manutenção. Por exemplo, uma alteração em uma *textual convention* precisaria ser feita apenas em um documento, e não em todo o lugar onde ela aparecesse.

Apesar de não existirem regras de como estruturar a informação de gerenciamento, uma sugestão para um conjunto mínimo de módulos para uma base de informações é mostrada na tabela 5.1 [Per 97]:

Tabela 5.1 - Módulos em uma Base de Informações

Nome do Módulo	Propósito	Número de Documentos
<empresa>-TC	*Textual Conventions* compartilhadas para uma determinada empresa	1
<empresa>-REG	Registro central de OIDs para uma empresa	1
<empresa>-<area>-MIB	Objetos gerenciados e definições de eventos para uma área particular de uma empresa	Muitos
<empresa>-<area>-REQ	Requisitos de aplicações de gerência para uma empresa em uma área particular (uma linha de produtos, p. ex.)	Muitos

5.4 Organização da Infraestrutura de OIDs

Quando uma organização desenvolve MIBs, uma das dúvidas que surgem é como organizar a estrutura de identificadores de objetos (OIDs) dentro do espaço de OIDs sob responsabilidade da empresa.

A parte correspondente aos ramos superiores da árvore de registros de OIDs é fixa, definidas pelos órgãos de padronização. A parte "possuída" pela organização não tem uma estrutura tão óbvia, mas o planejamento cuidadoso pode evitar problemas futuros. Em geral, existem seis categorias a serem consideradas quando do mapeamento em um ramo privado da árvore de registro, como ilustra a tabela 5.2 [Per 97]:

Tabela 5.2 - *Layout* da infraestrutura de OID

Categoria	Propósito
Registro (*registration*)	Identificação de módulos, e componentes físicos e lógicos
Genérico (*generic*)	Objetos e eventos utilizados por múltiplos produtos
Produtos (*product*)	Objetos e eventos associados com produtos específicos
Capacidades (*capabilities*)	Perfis de implementação de agentes
Requisitos (*requirements*)	Requisitos de objetos gerenciados para aplicações de gerência
Experimental (*experimental*)	Objetos e eventos em desenvolvimento

Cada uma das áreas apresentadas na tabela anterior pode ser mapeada diretamente para uma sub-árvore sob o OID designado para a empresa. Mesmo em uma empresa pequena, desmembramentos em cada uma destas sub-árvores devem ser feitos para evitar um crescimento futuro desordenado dos objetos. Por exemplo, o sub-ramo *product* não deveria conter diretamente ramos para os produtos. O mais sensato seria criar sub-árvores dentro de *product* para agrupar os produtos por categoria [Per 97]. Estas regras valem também para os desenvolvedores de MIBs nos órgãos de padronização.

Uma empresa, para obter um identificador para sua organização, deve contatar a IANA:

Endereço Postal:	IANA	
	USC/Information Sciences Institute	
	4676 Admiralty Way	
	Marina del Rey, CA 90292	
Telefone:	+1 310 822-1511	
Fax:	+1 310 823-6714	
E-mail:	iana-mib@isi.edu	— para identificadores de empresas
	iana@isi.edu	— para outras correspondências
URL:	http://www.iana.org/iana	

A sub-árvore *registration* não contém definições de objetos e eventos, ela simplesmente é uma área onde são registrados: identificações de produtos, componentes de produtos e itens relacionados com MIBs. Estes itens nunca são requisitados por uma estação de gerenciamento. Na verdade, os valores destes OIDs podem ser retornados como valores de outros objetos.

As sub-árvores *generic* e *product* contém os OIDs para a definição de objetos e eventos. A única diferença entre as duas sub-árvores é que a sub-árvore *generic* contém objetos de propósito geral que servem para muitos produtos da empresa, enquanto o ramo *product* contém objetos específicos para certos produtos ou serviços.

A sub-árvore *capabilities* é usada para registrar a identificação de perfis de implementação de agentes. Estes perfis estão associados com partes de código fonte, que podem incluir um agente completo. Entretanto, normalmente estão associados com bibliotecas de código usadas para agrupar os agentes.

O ramo *requirements* é usado para registrar as especificações de requisitos das aplicações de gerenciamento. Estas especificações de requisitos detalham os objetos e eventos usados por uma área funcional de uma aplicação de gerenciamento.

Por fim, o ramo *experimental* serve para o registro de objetos e eventos ainda em desenvolvimento, evitando a possibilidade de conflito com objetos em outras MIBs. Do ponto de vista dos clientes da empresa, este ramo deve ser visto como algo suscetível a mudanças, e que objetos presentes neste momento podem não estar mais aqui numa próxima versão. Outro propósito deste ramo é alocar objetos que ainda não foram publicados pelo IETF em uma RFC, mas que a empresa pretende disponibilizar em um produto. O ramo *experimental* na parte controlada pelo IETF serve o mesmo propósito, e cada grupo de trabalho do IETF recebe números que podem ser utilizados no ramo *experimental* para seus objetos em desenvolvimento.

5.5 Nomeação de Módulos

Os documentos relativos ao SNMPv1 dão pouca ou nenhuma informação sobre como nomear novos módulos de MIB. O próprio IETF adotava a prática de mudar o nome de um módulo já publicado em uma RFC quando alguns objetos eram alterados ou adicionados. Para evitar confusões com nomes dos módulos de MIB, existem algumas convenções que podem ser seguidas [Per 97]:

- deve-se usar somente letras maiúsculas no nome de um módulo. Mesmo que somente a primeira letra seja obrigada a ser maiúscula, esta regra é uma convenção largamente aceita e utilizada;

- usar uma convenção de nomes com três partes do tipo: <ORGANIZAÇÃO>-<NOME>-<TIPO>. A primeira parte é o nome da organização. A segunda parte é o nome propriamente dito do módulo. A terceira parte identifica que tipo de informação ele possui: MIB para a definição de objetos; REG e OID para registros; TC para convenções textuais; TRAP para *traps* SNMPv1; NOTI para *notifications* SNMPv2;

CAP para perfis de implementação dos agentes; e REQ para especificações de requisitos;

- não mudar o nome de um módulo entre duas versões. A SMIv2 proíbe a mudança de nomes entre versões. As mudanças de características entre duas versões devem ser identificadas na construção MODULE-IDENTITY. As MIBs SMIv1 devem seguir também esta regra, para evitar confusões futuras.

5.6 *Layout* do Módulo

Uma vez definidas a estrutura geral da base de informações, a infraestrutura de OIDs da empresa, e os nomes dos módulos, resta descrever os módulos em si. Basicamente, podemos ter dois tipos de módulos com diferentes *layouts*: módulos que definem objetos e/ou eventos e módulos que definem registros, convenções textuais, perfis e requisitos.

5.6.1 Módulos para Objetos Gerenciados e Eventos

Estes módulos possuem seis seções básicas [Per 97]:

- as importações de outros módulos (*imports*);
- a especificação MODULE-IDENTITY;
- definições de quaisquer convenções textuais locais;
- registros da estrutura de mais alto nível desta MIB;
- definições dos objetos gerenciados e, opcionalmente, notificações;

- uma série de declarações de conformidade.

Da mesma maneira que discutimos a organização das sub-árvores de uma empresa dentro do seu identificador particular, é necessário também estruturar os objetos e eventos de uma MIB em sub-árvores, o que chamamos de definição dos "registros da estrutura de mais alto nível desta MIB". Cada módulo de MIB costuma seguir um padrão mais ou menos semelhante, e é comum os desenvolvedores de MIBs documentarem este padrão da seguinte forma [Per 97]:

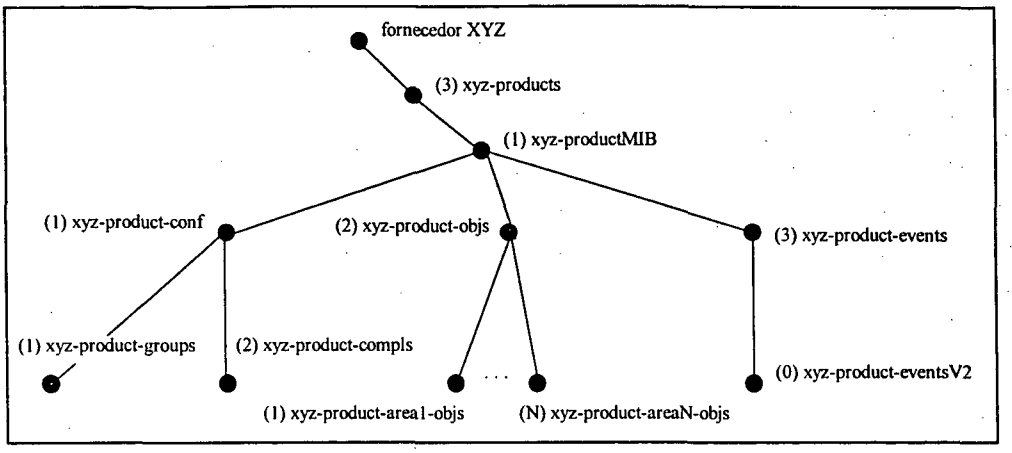
- cada área funcional recebe uma sub-árvore própria, de forma que um módulo pode conter uma ou muitas sub-árvores funcionais registradas. Objetos que se referem a um aspecto particular de um sistema gerenciado são normalmente mantidos juntos na mesma sub-árvore;

- declarações de conformidade, as quais incluem as construções OBJECT-GROUP, NOTIFICATION-GROUP, e MODULE-COMPLIANCE, possuem sua própria sub-árvore;

- Eventos (definidos com a construção NOTIFICATION-TYPE) são registrados em uma sub-árvore que segue a seguinte regra: sob a sub-árvore escolhida para conter os eventos, outra sub-árvore é registrada como o valor 0 (zero). Esta é a sub-árvore sob a qual os eventos são realmente registrados. Esta regra não é tão arbitrária, mas eventos registrados desta forma permitem a interoperabilidade entre o SNMPv1 e o SNMPv2. Algumas MIBs SMIv2 mais antigas não seguem esta regra porque foram escritas antes dela ser definida. Tais MIBs não devem servir de modelo, porque estão desatualizadas.

A figura 5.2 ilustra uma estrutura para um determinado fornecedor, seguindo esta sugestão:

Figura 5.2 - Organização da MIB para um fornecedor



5.6.2 Outros Módulos

Todos os outros módulos (*registration*, *textual conventions*, *profiles*, e *requirements*) possuem uma forma muito mais simples. Eles possuem apenas três seções [Per 97]:

- as importações de outros módulos (*imports*);
- a especificação MODULE-IDENTITY;
- definições.

A seção de definições é o módulo em si. É onde são definidas as convenções textuais, registros de OIDs, perfis de implementação, ou requisitos.

5.7 Definindo Estatísticas

É verdade que as estatísticas que os agentes SNMP fornecem podem ajudar o gerente de redes a identificar diversos problemas com serviços e equipamentos na rede. Entretanto, segundo [Per 97], é na modelagem das estatísticas que muitas MIBs tornam-se inúteis. Algumas vezes, os projetistas de MIBs incluem informação demais em seus módulos, na tentativa de não deixar nenhuma informação de fora. Os problemas com informação demais em uma MIB são evidentes: a aplicação de gerenciamento deverá ser configurada para extrair somente a informação que interessa no meio de informação inútil presente. Além disso, os desenvolvedores de agentes vêem-se obrigados a implementar toda a MIB, mesmo que boa parte das informações não seja utilizada.

Para evitar estes problemas, toda vez que um projetista incluir informações estatísticas em seu módulo de MIB, ele deve justificar sua inclusão, baseado em algumas questões [Per 97]:

- quem se beneficiará desta variável? (o usuário ou o desenvolvedor?)
- essa variável fornece informação que não pode ser encontrada em outro lugar?
- essa estatística é realmente útil? (o usuário pode tomar alguma ação baseado nisso?)
- é possível que esta estatística seja implementada no agente a um custo razoável?
- o significado desta estatística está claro para os desenvolvedores de agentes e os usuários?

Se uma estatística é útil para o usuário e o usuário pode tomar alguma ação baseado nesta, então ela vale a pena ser implementada. Este tipo de questionamento é relevante porque até mesmo nos documentos escritos pelos órgãos de padronização existem diversas informações de uso duvidoso, constituindo um verdadeiro desafio para desenvolvedores e usuários.

Outro tipo de estatística que deve ser descartada é aquela que consome muita largura de banda da rede para ser enviada do agente até o gerente [Per 97].

5.8 Implementação de Ações em uma MIB SNMP

Ações em SNMP estão intimamente ligadas ao conceito de estado. Na verdade, uma ação em SNMP nada mais é do que uma requisição da estação de gerenciamento para a mudança de um estado no agente.

Uma vez que o modelo SNMP somente é capaz de tratar ações implícitas (disparadas através da modificação de uma variável), as ações precisam ser cuidadosamente projetadas de forma que elas possuam uma série de estados que possam ser monitorados e manipulados pela estação de gerenciamento.

Entretanto, as ações no modelo SNMP podem ser limitadas pelo próprio modelo. Como normalmente existe somente uma variável de controle para que um gerente possa iniciar uma ação, existe dificuldade em permitir que mais de um gerente dispare a mesma ação no mesmo agente. Se um primeiro gerente inicia uma ação e, durante sua execução, um segundo gerente tenta iniciar o mesmo tipo de ação no mesmo agente, o segundo gerente deve ficar bloqueado, recebendo uma mensagem de erro correspondente. O problema é que as mensagens de erro do protocolo são limitadas, e não podem ser especificadas mensagens de erros com significados específicos.

Para resolver a situação, poder-se-ia incluir uma variável separada de estado, indicando ao segundo gerente que o agente naquele momento está ocupado com a execução de uma ação. Novamente, outros problemas de sincronismo podem ocorrer se, por exemplo, imediatamente após a ação disparada pelo primeiro gerente terminar, o segundo gerente iniciar a mesma ação. O primeiro gerente periodicamente tentará verificar se sua ação já completou, e encontrando a variável de estado indicando ação em andamento (do segundo gerente), poderá "pensar" que sua requisição para a execução da ação ainda não tenha terminado.

Dessa forma, o projetista de MIBs SNMP deve tomar cuidado ao introduzir variáveis de controle em seu módulo. As ações correspondentes devem executar o mais rápido possível, e devem ser ações simples que não gerem problemas de sincronismo e espera mais graves.

Por exemplo, uma ação correspondente à transferência de um arquivo de configuração de um gerente para um agente não é muito adequada ao modelo, pois pode ser uma operação demorada, causando problemas se mais de um gerente tentar executar a mesma operação no mesmo momento. Neste caso, o projetista pode optar por deixar determinadas ações para serem feitas de forma "manual", ou seja, por intervenção da pessoa do gerente de redes, ou então o projetista pode utilizar mecanismos mais sofisticados, como as tabelas.

O uso de tabelas em ações é apropriado para resolver o problema de sincronismo da seguinte maneira: cada gerente querendo executar uma ação preenche os parâmetros da ação nos campos de uma linha de uma tabela, isto é, o gerente cria uma linha em uma tabela com os seus dados. O agente, vai executando as ações na tabela uma a uma, e os gerentes podem verificar o andamento de suas ações observando um campo na tabela sobre o estado da ação.

O mecanismo para definir tabelas que permitem a criação de linhas através de operações SET vindas das estações de gerenciamento é discutido a seguir.

5.9 Tabelas

Algumas MIBs possuem tabelas que representam recursos dinâmicos que podem ser criados, suspensos, ou destruídos. Estas tabelas precisam utilizar um mecanismo padrão para a criação e remoção de linhas na tabela. Em algumas situações pode ser necessária a criação e remoção de linhas de uma tabela através do uso de operações SNMP, enquanto em outras situações isto não faz sentido.

Suponha, por exemplo, uma tabela contendo as interfaces de rede de uma determinada máquina. Cada linha na tabela corresponde a uma interface existente na máquina, e as colunas aos atributos da interface, como velocidade, tipo, etc. Não faz sentido uma aplicação de gerenciamento de redes criar e destruir linhas nesta tabela, pois a máquina não será capaz de abrir um armário, pegar uma nova placa e instalá-la em seu interior.

Neste caso, a criação e remoção de linhas é feita de forma automática pelo próprio agente SNMP responsável pela tabela. Se o usuário da máquina resolve desligá-la e instala uma nova interface, assim que o agente for novamente inicializado, ele deverá detectar a nova placa e incluir uma linha correspondente na tabela de interfaces.

Para a criação e remoção de linhas em tabelas via operações SNMP, a SMIv2 define um mecanismo "padrão", baseado no trabalho originado no desenvolvimento do RMON MIB. A SMIv2 define uma *textual convention* chamada RowStatus, que consiste em um inteiro enumerado que pode assumir

um entre 6 valores: *active*(1), *notInService*(2), *notReady*(3), *createAndGo*(4), *createAndWait*(5), e *destroy*(6).

Para usar este mecanismo, basta declarar um objeto que use esta convenção textual em sua cláusula SYNTAX, e então incluir o objeto como parte da definição da linha da tabela desejada. Os estados que a convenção textual RowStatus pode assumir são classificados em três tipos: estados de ação, desejados, e corrente.

Os estados de ação podem ser escritos, mas não lidos:

- *createAndGo* indica que o agente deve aceitar os valores para a linha especificados, criar os ítens dentro da linha, e ativá-la;
- *createAndWait* indica que o agente deve aceitar os valores especificados (se existirem), criar os ítens dentro da linha, mas não ativá-la;
- *destroy* indica que o agente deve desativar a linha e removê-la da tabela.

Os estados desejados podem ser escritos e lidos:

- *notInService* indica que a linha foi suspensa, isto é, a instância modelada pela linha não existe;
- *active* indica que a instância do recurso modelado pela linha existe.

O estado corrente somente pode ser lido:

- *notReady* indica que a linha não contém valores necessários para uma instância do recurso modelado pela linha ser criada. Quando todos os valores necessários da linha forem "preenchidos" através de operações SET vindas do gerente, o estado automaticamente passa para *active*.

Todos os valores que podem ser escritos em uma linha de uma tabela que segue ste esquema devem ter a cláusula MAX-ACCESS igual a *read-create*.

5.10 Projetando Eventos

As operações de comunicação de eventos no SNMP sempre foram alvo de críticas pela comunidade de gerenciamento de redes.

Uma das afirmações freqüentemente feitas contra a operação TRAP é o fato desta não possuir confirmação. Se um agente envia uma PDU TRAP para um gerente e a mensagem se perde no meio do caminho por um problema qualquer, o gerente jamais terá sido notificado da existência dessa mensagem, e o agente jamais tentará enviá-la novamente. Entretanto, este tipo de problema pode ser minimizado configurando o agente para enviar notificações de eventos para mais de uma estação de gerenciamento.

Outra crítica às funções de comunicação de eventos em geral, independentemente da arquitetura de gerenciamento utilizada, é o fato de que se a comunicação do agente com o gerente for interrompida por completo, quer seja por um problema no equipamento onde o agente está executando, quer seja por um problema físico na rede, o agente não terá a chance de informar o gerente do

fato. É comum ouvir a frase: "Como é que o meu roteador vai poder avisar-me que ele acabou de queimar?".

A verdade é que não importa quanto a operação TRAP é limitada, ela faz parte do modelo de gerenciamento SNMP, e deve ser cuidadosamente usada para ser efetiva.

Existem três categorias gerais para a definição de eventos [Per 97]:

- informacional, a qual representa condições não críticas;
- avisos, que indicam uma falha ainda pendente ou eventos inesperados;
- erros, os quais indicam que uma condição catastrófica ocorreu.

Entretanto, tais categorias não são absolutas, isto é, dependem do contexto no qual se inserem. Por exemplo, um servidor de arquivos cuja única interface de rede possui o estado corrente "inativa", pode ser considerada uma condição grave. Entretanto, se o mesmo servidor possuísse várias placas de rede, e uma delas estivesse no estado "inativa", poderia ser que a situação não fosse tão crítica. Poderia, por exemplo, ter ocorrido algum problema com o cabo a ela conectado, e temporariamente o servidor continuaria operando com as outras interfaces.

Eventos informacionais servem para indicar ao gerente que houve alguma mudança no sistema gerenciado, constituindo um evento que pode ocorrer normalmente durante o funcionamento do sistema gerenciado. Avisos constituem eventos um pouco mais sérios. Eles indicam que algum recurso

monitorado possui uma condição inesperada, e que algum problema ainda mais sério pode ocorrer. Erros indicam condições críticas do recurso monitorado.

O protocolo SNMPv1 e o protocolo SNMPv2 não suportam explicitamente estes diferentes níveis de alarmes, mas o desenvolvedor do módulo de MIB pode (e deve) especificar textualmente o nível de importância dos eventos em suas definições.

Cuidado deve ser tomado para não usar de forma incorreta as operações TRAP e NOTIFICATION. Não se deve esquecer que estas operações constituem uma iniciativa do agente em informar a ocorrência de um evento. Algumas MIBs definem erradamente eventos que devem ser enviados como confirmações de operações do gerente. Por exemplo, seria incorreto utilizar *traps* para indicar o fim de uma ação de transferência de arquivos. Se o agente estivesse configurado para notificar 10 gerentes diferentes, 90% do tráfego gerado pelos *traps* enviados seria inútil.

Por outro lado, eventos não devem ser usados para substituir as operações de *polling* dos gerentes. Mas eles podem ser úteis para antecipar ao gerente condições importantes antes do próximo ciclo de *polling*.

Assim, no momento da definição dos eventos é interessante perguntar, para cada evento, que tipo de problema ele ajuda a diagnosticar. Se esta pergunta não puder ser respondida, então provavelmente o evento não é relevante. Uma boa prática de projeto de eventos deveria endereçar estas três questões [Per 97]:

- "Em que contexto a condição ocorreu?"
- "Qual é o problema?"
- "Quando é apropriado informar ao gerente?"

A verdade é que a experiência do desenvolvedor pode ajudar a determinar que tipos de eventos são importantes. Dessa forma, deve-se considerar as questões levantadas anteriormente, e observar os resultados obtidos [Per 97].

5.11 Verificação de Conformidade de MIBs

Uma MIB, para ser válida, deve estar escrita corretamente. Verificar a conformidade de uma MIB consiste em verificá-la quanto à sintaxe e semântica que ela se propõe a seguir. Assim, uma MIB SMIv2 não deve utilizar convenções SMIv1 que a nova SMI proíbe ou não recomenda.

Compiladores de MIBs são programas que interpretam a notação utilizada na definição de MIBs. Além de serem usados em plataformas de gerenciamento para o registro de novos objetos na árvore de objetos, podem também ser utilizados para a verificação de sintaxe e semântica de novas MIBs em desenvolvimento.

Capítulo 6

UMA MIB PARA APLICAÇÕES CLIENTES INTERNET

6. Uma MIB para Aplicações Clientes Internet

Conforme foi dito no capítulo anterior, existem diferentes maneiras pelas quais objetos podem ser adicionados à árvore de registro de objetos da MIB SNMP.

No caso deste trabalho, um módulo de MIB foi definido para um uso particular, surgido das necessidades de gerenciamento levantadas no capítulo 4. Dados os objetivos desta nova MIB, ela será referenciada no restante deste trabalho como *Network Client Application MIB* (MIB para Aplicações Cliente de Rede).

A seguir são apresentados os resultados do desenvolvimento deste novo módulo de MIB, seguindo as etapas de desenvolvimento apresentadas no capítulo anterior.

6.1 SNMPv1 ou SNMPv2?

Antes de definir a MIB, é necessário decidir qual a versão de SMI SNMP deve ser utilizada. Tomando como base os pontos levantados no capítulo anterior sobre o uso da SMIv1 ou da SMIv2 na definição de novas MIBs, o módulo definido a seguir segue a sintaxe da SMIv2, pelas seguintes razões:

- toda nova MIB desenvolvida pelo IETF deve seguir a SMIv2, e portanto isto deve tornar-se uma tendência para os demais desenvolvedores;

- uma MIB na sintaxe da SMIv2 pode ser implementada para uso sobre qualquer versão do protocolo, SNMPv1 ou SNMPv2;
- a SMIv2 possui uma sintaxe muito mais rica e mais precisa para definir MIBs. Algumas construções só podem ser representadas na sintaxe da SMIv2;
- a sintaxe da SMIv2 é, sem dúvida, uma melhoria significativa sobre a SMIv1;
- quando for necessário converter uma MIB SMIv2 para a SMIv1, podem ser usadas ferramentas de conversão que funcionam corretamente para a grande maioria dos casos. Se necessárias conversões manuais, estas estão bem documentadas na RFC 1908 [Cas 96f].

6.2 Requisitos e Análise dos Objetos

A MIB definida a partir deste item visa o gerenciamento das aplicações clientes de rede, bem como seus estados, conexões, etc. Os objetos identificados para esta MIB estão agrupados segundo as cinco categorias sugeridas por [Per 97].

6.2.1 Componentes e Sub-Componentes

Os seguintes componentes foram identificados:

- aplicações clientes de rede: quais aplicações clientes de rede estão executando num determinado momento. As informações sobre as aplicações são mantidas em uma tabela. Cada instância de aplicação possui um identificador único;

- conexões ativas das aplicações: cada aplicação de rede ativa pode possuir zero ou mais conexões de rede ativas em determinado momento. Este componente, definido na forma de uma tabela, lista todas as conexões ativas no momento. A relação com as aplicações dá-se através do identificador da aplicação;

- aplicações já finalizadas: um histórico sobre as aplicações que o usuário já finalizou desde o momento em que o agente foi inicializado é mantido nesta tabela, que possui formato semelhante ao da tabela das aplicações ativas. O identificador de cada aplicação continua único, não se repetindo na tabela de aplicações ativas;

- conexões já finalizadas das aplicações: um histórico das conexões já feitas pelas aplicações. Uma conexão finalizada nesta tabela pode tanto pertencer a uma aplicação ainda ativa como a uma aplicação já finalizada;

- eventos relativos a conexões que devem ser notificados: uma tabela de eventos sobre conexões que devem ser notificadas aos gerentes configurados como receptores de *Trap* PDUs.

6.2.2 Atributos

Os atributos identificados para cada um dos componentes são os seguintes:

- aplicações cliente de rede:
  - nome da aplicação;
  - versão da aplicação;
  - identificador da aplicação.

- conexões ativas das aplicações:
  - identificador da aplicação que abriu esta conexão;
  - identificador da conexão em relação à sua aplicação;
  - *host* destino da conexão;
  - protocolo utilizado na conexão;
  - porta utilizada na conexão (um número de *socket*).
- aplicações já finalizadas:
  - nome da aplicação;
  - versão da aplicação;
  - identificador da aplicação;
  - motivo pelo qual a aplicação terminou.
- conexões já finalizadas das aplicações:
  - identificador da aplicação que abriu esta conexão;
  - identificador da conexão em relação à sua aplicação;
  - *host* destino da conexão;
  - protocolo utilizado na conexão;
  - porta utilizada na conexão (um número de *socket*);
  - motivo pelo qual a conexão terminou.
- eventos relativos a conexões que devem ser notificados:

- *host* destino da conexão;
- protocolo utilizado na conexão;
- porta utilizada na conexão (um número de *socket*).

6.2.3 *Ações*

- suspender uma aplicação;
- reanimar uma aplicação;
- terminar uma aplicação;
- suspender uma conexão;
- reanimar uma conexão;
- terminar uma conexão.

6.2.4 *Estatísticas*

Estatísticas para cada um dos componentes são as seguintes:

- aplicações cliente de rede:
  - *uptime* da aplicação (tempo decorrido desde o início da execução);
  - número de conexões ativas da aplicação;
  - número de conexões já finalizadas da aplicação;
  - quantidade de octetos recebidos pela aplicação;

- quantidade de octetos enviados pela aplicação.

- conexões ativas das aplicações:
  - *uptime* da conexão (tempo decorrido desde sua abertura);
  - quantidade de octetos recebidos pela conexão;
  - quantidade de octetos enviados pela conexão.
- aplicações já finalizadas:
  - tempo de duração da aplicação em execução;
  - número de conexões total que a aplicação abriu;
  - quantidade total de octetos recebidos pela aplicação;
  - quantidade total de octetos enviados pela aplicação.
- conexões já finalizadas das aplicações:
  - tempo de duração da conexão;
  - quantidade total de octetos recebidos pela conexão;
  - quantidade total de octetos enviados pela conexão.
- eventos relativos a conexões que devem ser notificados:
  - quantidade de ocorrências do evento.

6.2.5 Estados

Os estados administrativos dos componentes abaixo relacionados constituem estados especiais que permitem que ações sejam executadas pelo agente. Os

estados administrativos modificam indiretamente o estado operacional da aplicação ou conexão.

- aplicações cliente de rede:
  - estado operacional da aplicação;
  - estado administrativo da aplicação.
- conexões ativas das aplicações:
  - estado operacional da conexão;
  - estado administrativo da conexão.
- eventos relativos a conexões que devem ser notificados:
  - estado de criação do evento (*RowStatus*).

6.2.6 Tabela-Resumo dos Objetos da MIB

A tabela 6.1 resume todos os objetos identificados nesta seção. As ações são incluídas sob a forma de estados. No momento da conversão da tabela na sintaxe da SMI, as ações são identificadas como tal.

Tabela 6.1 - Tabela-Resumo dos Objetos da Nova MIB

Componente	Cardinalidade	Atributos	Estatísticas	Estados
Aplicações Ativas	0 ou mais	name, version, identifier	uptime, connections, old_connections, in_octets, out_octets	oper_status, admin_status
Conexões Ativas	0 ou mais	appl_ident, conn_ident,	uptime, in_octets,	oper_status, admin_status

Componente	Cardinalidade	Atributos	Estatísticas	Estados
		dest_host, protocol, port	out_octets	
Aplicações Finalizadas	0 ou mais	name, version, identifier, end_status	duration, connections, in_octets, out_octets	
Conexões Finalizadas	0 ou mais	appl_ident, conn_ident, dest_host, protocol, port, end_status	duration, in_octets, out_octets	
Eventos de Conexões	0 ou mais	dest_host, protocol, port	ocurrences	creation_status

6.3 Descrição da *Network Client Application MIB*

Devido à pequena quantidade de objetos identificadas, toda a MIB foi especificada em um único módulo, apresentado no anexo 1.

6.4 Verificação de Conformidade

Para a verificação de sintaxe e semântica da *Network Application MIB*, foi utilizado o compilador SMICng (SMIC *next generation*), desenvolvido por David T. Perkins [Per 97]. Este compilador é uma evolução do SMIC (*SNMP Management Information Compiler*), da *Bay Networks*, e é um *software* disponível livremente para uso por qualquer pessoa ou organização.

O compilador SMIC não serve para o uso com plataformas de gerenciamento, pois os resultados produzidos possuem formatos próprios para diversas opções

de *parsing* posterior. Atualmente, SMIC e SMICng são utilizados pelos grupos de trabalho do IETF para verificar MIBs antes de entrarem no processo de padronização (antigamente o compilador MOSY, que acompanhava o ISODE, era utilizado pelo IETF).

Executando o compilador SMICng, utilizando como entrada a *Network Client Application MIB*, e como opções solicitando a verificação segundo as regras mais restritas da SMIv2, alguns pontos incorretos foram detectados. Após sua correção, o compilador interpretou corretamente toda a MIB como estando em conformidade com a SMIv2.

Capítulo 7

BASES PARA A IMPLEMENTAÇÃO DE APLICAÇÕES DE GERENCIAMENTO

7. Bases para a Implementação de Aplicações de Gerenciamento

É natural que com a popularização do SNMP, diversas plataformas para o desenvolvimento de aplicações de gerenciamento para este modelo surjam no mercado. Cada produto implementa bibliotecas e mecanismos para tornar a programação de aplicações de gerenciamento uma tarefa mais fácil.

Existem, é claro, bons pacotes de domínio público, como por exemplo, o CMU SNMP, da *Carnegie Mellon University*.

Conforme o tipo de aplicação a ser desenvolvida, diferentes produtos podem ser necessários. A principal consideração é definir se a aplicação será uma aplicação no papel de agente, ou no papel de gerente. Algumas plataformas não possuem facilidades para a implementação de um tipo ou de outro de aplicações de gerenciamento. A seguir são discutidas brevemente algumas destas plataformas, com enfoque maior naquelas adequadas para o desenvolvimento de agentes, que é o principal objetivo deste trabalho.

7.1 Plataformas para o Desenvolvimento de Aplicações Gerente

7.1.1 APIs Proprietárias

As APIs proprietárias de diversos fabricantes estão geralmente incluídas em produtos mais completos, normalmente chamados de plataformas de

gerenciamento. Uma plataforma de gerenciamento costuma possuir algumas aplicações e ferramentas já prontas que permitem ao gerente de redes começar a gerenciar seus elementos de rede quase que imediatamente.

Tais plataformas incluem mecanismos para:

- descoberta da topologia da rede;
- interface gráfica para o console de gerenciamento;
- um MIB *Browser*, que permite a inspeção e modificação direta de variáveis da MIB de um agente SNMP;
- aplicações para alarmes e análise de estatísticas.

Entretanto tais aplicações costumam ser muito limitadas do ponto de vista de utilização prática por parte do usuário. O gerente de redes normalmente não deseja ter que chamar um MIB *Browser* para começar a inspecionar variáveis toda vez que um problema na rede ocorrer. Ao contrário, ele deseja uma interface mais inteligente, específica para os equipamentos gerenciados, que lhe forneçam informações relevantes em um formato de fácil entendimento. Por exemplo, ao invés do gerente ter que inspecionar as variáveis correspondentes ao estado de cada porta de um *hub Ethernet*, seria muito mais fácil ele ter à sua frente um desenho do *hub*, mostrando através de "luzes" de diferentes cores os estados das diversas portas do *hub*.

Este tipo de aplicação só pode ser criada (pelo próprio gerente de rede ou por terceiros) nas plataformas que possuem APIs para a criação de aplicações de gerenciamento. Tais APIs permitem que os fabricantes de equipamentos desenvolvam aplicações específicas para seus produtos. Dessa forma, integrando

aplicações às plataformas de gerenciamento, o usuário não precisa executar uma aplicação separada para gerenciar o *hub* do fabricante X, nem executar outra para gerenciar o *switch* do fabricante Y. Todas as aplicações passam a estar presentes em um único console.

O maior problema com as APIs proprietárias das plataformas de gerenciamento, entretanto, é que se algum dia o gerente resolver trocar de plataforma de gerenciamento, ele terá que re-escrever ou adquirir as aplicações que já possuía para a plataforma antiga.

Faz-se então necessária, a definição de APIs padronizadas para o desenvolvimento de aplicações de gerenciamento, de modo que a mudança de plataforma seja tão simples quanto recompilar o código fonte de uma plataforma na outra.

Hoje, existem duas APIs padronizadas que merecem maior atenção por parte dos desenvolvedores: WinSNMP e SNMP++, cujas principais características são descritas nos itens seguintes.

7.1.2 WinSNMP

Em janeiro de 1993, um grupo de funcionários da Microsoft começou a trabalhar numa especificação de uma API para a construção de aplicações de gerenciamento de redes SNMP sobre o ambiente Microsoft Windows. O objetivo era facilitar a tarefa de criação destas aplicações, livrando o desenvolvedor dos detalhes com os protocolos, PDUs, etc.

Além disso, a API visa a criação de aplicações portáveis. Uma aplicação escrita em conformidade com a especificação poderia executar em qualquer ambiente que possuísse uma implementação da API consistente com a especificação.

Depois de mudar algumas vezes de mãos, a especificação passou a ser trabalhada em sua maior parte por Bob Natale, funcionário da *American Computer & Electronics Corporation*. A versão mais recente da especificação de Bob Natale é a 1.1a, de agosto de 1995. Uma nova versão, 2.0, já está em sua fase final de desenvolvimento, e deve ser publicada no máximo até a metade do ano de 1997.

A idéia inicial desta base de desenvolvimento era produzir diferentes especificações que atendessem três aspectos do desenvolvimento de aplicações SNMP:

- WinSNMP/Manager API: uma API contendo funções, tipos de dados, e estruturas de dados, com o objetivo da programação de aplicações no papel de gerente;
- WinSNMP/MIB API: uma API que define um conjunto funções de banco de dados que fornecem às aplicações de gerenciamento uma maneira de pesquisar a MIB conhecida pela estação de gerenciamento. A WinSNMP/MIB pode ser vista como uma API que uma estação de gerenciamento pode usar para obter informações tanto de MIBs padronizadas como de MIBs privadas;
- WinSNMP/Agent API: consiste em uma base robusta, poderosa e compreensível para a implementação de agentes SNMP baseados no Microsoft Windows que sejam extensíveis, interoperáveis e independentes de fornecedores.

De todo o conjunto de APIs propostas inicialmente, apenas a API para a implementação de aplicações gerente vingou. Enquanto a API WinSNMP/MIB não encontrou aceitação pelos fornecedores de *software*, a API WinSNMP/Agent chegou a ter uma versão de divulgação implementada, mas não foi aceita principalmente porque é muito difícil de ser usada por implementadores. Assim, o termo WinSNMP atualmente refere-se à especificação WinSNMP/Manager API, cuja aceitação tem sido cada vez maior, principalmente entre desenvolvedores de aplicações o ambiente Windows.

Basicamente, esta especificação define funções, tipos de dados, estruturas de dados, e a semântica associada com a qual um desenvolvedor pode programar, e um fornecedor de *software* pode implementar a especificação.

Sob o ambiente Windows, a implementação da API é feita sob a forma de uma DLL (biblioteca de ligação dinâmica). Felizmente, para os desenvolvedores de *software* para múltiplas plataformas, muitos fornecedores de plataformas de gerenciamento estão incluindo suporte a WinSNMP em seus produtos, entre eles a IBM com o NetView, e a HP com o OpenView. Existem algumas implementações independentes da API para várias versões de UNIX.

Com isso, aplicações desenvolvidas sob Windows para a API podem ser facilmente portadas para o ambiente UNIX, com praticamente nenhuma mudança no código fonte.

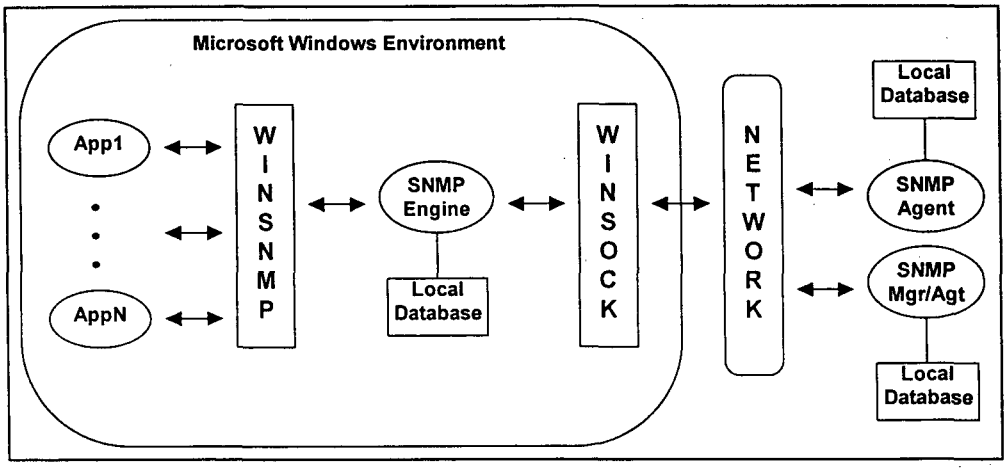
Em resumo, WinSNMP oferece três benefícios básicos [Nat 95]:

- consiste em uma implementação SNMP pronta para aplicações de gerenciamento de redes funcionais, isto é, a API "esconde" ASN.1, BER e os detalhes do protocolo SNMP;

- independência do fornecedor do *software* WinSNMP. Uma aplicação WinSNMP executará em qualquer implementação WinSNMP que esteja em conformidade com o padrão;
- suporte uniforme a SNMPv1 e SNMPv2. Um aplicação gerente WinSNMP não precisa conhecer a versão SNMP das entidades alvo atuando como agentes. A implementação WinSNMP fará qualquer mapeamento necessário entre SNMPv1 e SNMPv2 de acordo com as RFCs apropriadas.

Um cenário típico de uma aplicação WinSNMP é ilustrado na figura 7.1:

Figura 7.1 - Cenário WinSNMP típico



É claro que outras configurações diferentes da ilustrada na figura 7.1 podem existir, inclusive sobre outros sistemas operacionais e protocolos de transporte.

Atualmente, as funções definidas na WinSNMP/Manager API não permitem a criação de aplicações no papel de agente, pois não existe uma função do tipo

"listen", que permita à aplicação agente esperar por uma operação SNMP. Entretanto, os fornecedores de *software* que comercializam pacotes que implementam a WinSNMP incluem soluções proprietárias, que não fazem parte da especificação, e que permitem a programação de aplicações agente.

A próxima versão da especificação incluirá suporte a funções para a programação de agentes, conforme já anunciado pelo autor na lista de discussão sobre o assunto.

A importância da especificação já é reconhecida entre os fornecedores de *software* básico. Recentemente, a Microsoft Corporation anunciou que estará incluindo, sob a forma de OEM, uma implementação da WinSNMP/Manager API em seu sistema operacional Windows NT 5.0, que deve ser lançado no mercado até o final deste ano.

Isto significa que a Microsoft também deverá incluir suporte à programação de aplicações WinSNMP no SDK correspondente ao sistema operacional Windows NT 5.0.

A lista de discussão sobre WinSNMP é aberta, e qualquer pessoa pode participar, enviando e-mail para **majordomo@mailbag.intel.com**, contendo no corpo da mensagem, **subscribe WinSNMP**. A especificação da API pode ser encontrada na URL **ftp://SunSITE.unc.edu/pub/micro/pc-stuff/ms-windows/WinSNMP**.

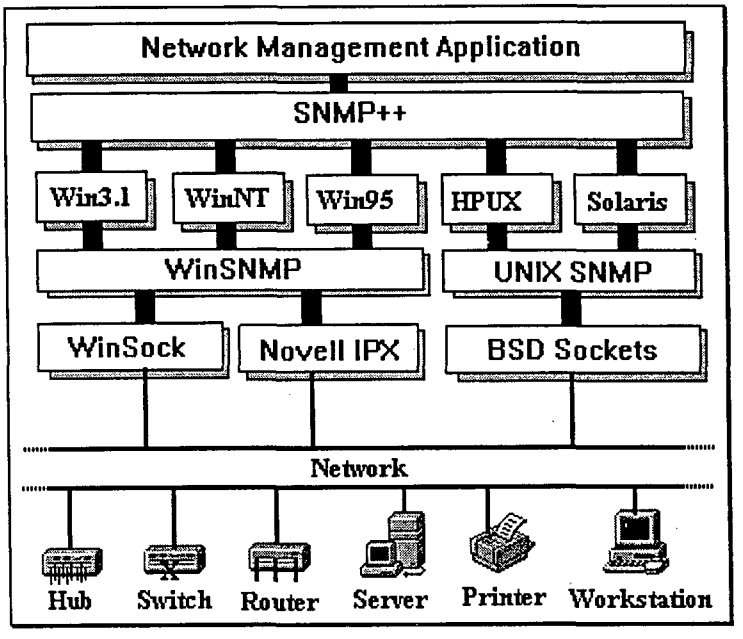
7.1.3 SNMP++

SNMP++ é um conjunto de classes de objetos C++ que fornecem serviços SNMP para um desenvolvedor de aplicações de gerenciamento de redes.

Entretanto, SNMP++ não consiste em mais uma implementação de SNMP. Na verdade, SNMP++ utiliza os mecanismos SNMP já existentes em outras APIs, como a WinSNMP, para trazer as vantagens da programação orientada a objetos para o desenvolvedor de aplicações SNMP [Mel 96].

A figura 7.2 ilustra o contexto de SNMP++ em relação ao sistema operacional e mecanismo SNMP utilizados:

Figura 7.2 - SNMP++ em relação ao SO e mecanismo SNMP



Esta especificação foi desenvolvida dentro da *Hewlett Packard Corporation*, por um grupo liderado por Peter Erik Mellquist, e pode ser encontrada, juntamente com o código fonte em C++ de sua implementação e exemplos, na *home-page* do SNMP++, URL **http://rosegarden.external.hp.com/snmp++**. A lista de discussão para assuntos relacionados a SNMP++ é a mesma para assuntos relacionados a WinSNMP.

O documento definindo a especificação SNMP++ [Mel 96], endereça quatro objetivos básicos:

- Facilidade de Uso: o usuário da API não precisa ser um *expert* em SNMP, nem um *expert* em C++ para programar usando SNMP++. Além disso, o programador tem a flexibilidade de ignorar, quando necessário, as classes SNMP++, e programar diretamente para a API SNMP existente;

- Segurança: a maioria das APIs SNMP forçam o programador a gerenciar uma série de recursos, incluindo OIDs, *variable bindings*, *variable bindings lists*, PDUs, *communities*, e estruturas de autenticação. O gerenciamento incorreto destes recursos pode resultar em regiões de memória corrompidas ou perdidas. SNMP++, através de suas classes projetadas como Tipos Abstratos de Dados, provê o gerenciamento destes recursos de forma transparente. Além disso, SNMP++ trata erros diversos, como datagramas perdidos, duplicados, ou desordenados.

- Portabilidade: um dos maiores objetivos de SNMP++ é prover uma API portável entre vários sistemas operacionais, sistemas operacionais de rede, e plataformas de gerenciamento de redes. Uma vez que os mecanismos internos de SNMP++ permanecem invisíveis, a interface pública permanece a mesma entre plataformas. Um programador que codifica para SNMP++ em uma plataforma não precisa fazer nenhuma mudança no código para transportá-lo para outra plataforma [Mel 96]. Atualmente, existem implementações SNMP++ para todas as versões do Microsoft Windows (Win16 e Win32), várias versões de UNIX de diferentes fabricantes (Solaris, SunOS, HP-UX) e para a plataforma de gerenciamento da HP, o OpenView. Outro aspecto da portabilidade é a possibilidade de executar SNMP++ sobre uma variedade de protocolos.

Atualmente, SNMP++ opera sobre o protocolo IP e sobre o protocolo IPX da Novell.

- Extensibilidade: SNMP++, além de ter sido estendido para suportar outros sistemas operacionais, pode ser estendido redefinindo classes SNMP++. Estas classes não possuem nenhuma informação específica de sistema operacional ou de *hardware*, e novos atributos podem ser facilmente adicionados através do mecanismos de *sub-classing* (herança) e redefinição de funções membro de C++.

7.2 Plataformas para o Desenvolvimento de Aplicações Agente

7.2.1 APIs Proprietárias

Assim como no desenvolvimento de aplicações gerente, a maioria das plataformas de gerenciamento inclui APIs para a construção de aplicações agente. Algumas plataformas são bastante simples neste sentido, permitindo que o usuário defina, em um formato particular que nada tem a ver com a SMI SNMP, conjuntos de variáveis que um agente deverá suportar.

Os bons pacotes de domínio público geralmente incluem algum suporte para a criação de novos agentes.

Entretanto, utilizar mecanismos proprietários, da mesma forma que para as APIs "gerentes", não é a melhor saída. O maior problema é a portabilidade. Existem inúmeras razões para se ter uma padronização neste sentido. Dentro do IETF, um grupo de trabalho está trabalhando em um *framework* padronizado para o desenvolvimento de agentes SNMP, chamado de AgentX.

As seções seguintes descrevem brevemente alguns aspectos do AgentX, e do agente extensível da Microsoft, pelo fato de ter obtido grande aceitação pelos

desenvolvedores, facilidade de uso, e pela própria popularidade dos sistemas operacionais da Microsoft.

7.2.2 Microsoft Extendible SNMP Agent

Desde a versão 3.5 do Microsoft Windows NT, a Microsoft inclui em sua linha de sistemas operacionais um agente SNMP. Este agente é extensível, e muitas de suas características assemelham-se às da antiga WinSNMP/Agent API.

Na prática, é muito fácil criar novos agentes para o agente extensível da Microsoft, uma vez que se tenha uma MIB corretamente definida. Atualmente, é o caminho mais fácil e rápido para a implementação de agentes sobre o ambiente Microsoft Windows.

Algumas deficiências já são reconhecidas pela própria Microsoft, que não pretende abandonar o projeto, uma vez que a proposta do AgentX deve ainda demorar algum tempo para ser finalizada. As novas versões de sistemas operacionais da Microsoft devem incluir novas versões deste agente.

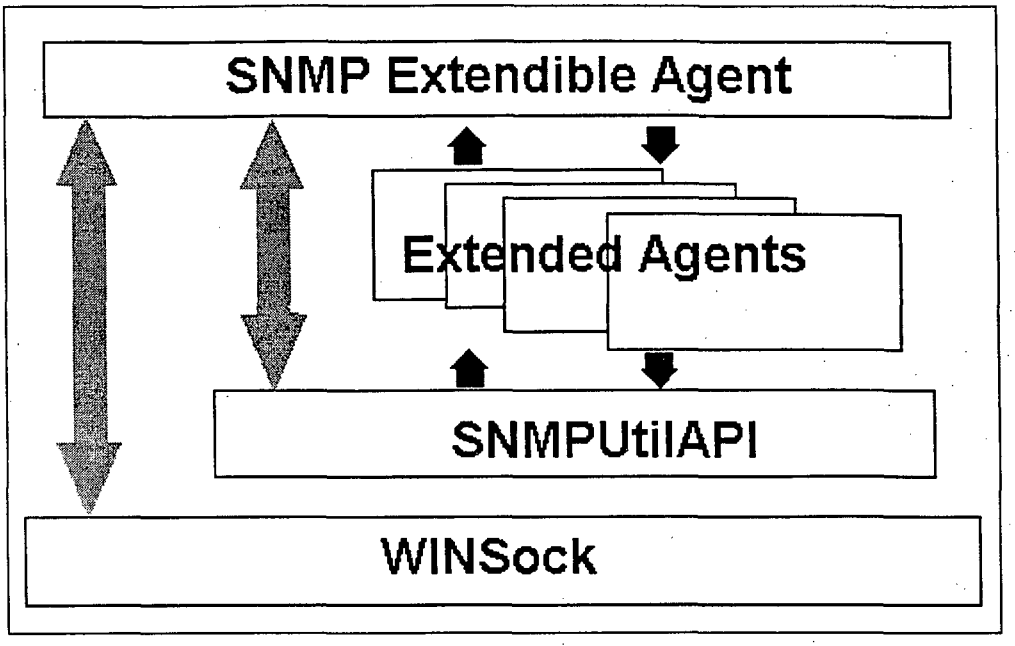
Um problema com este agente é sua documentação relativamente pequena. Aparentemente, existe apenas uma única referência bibliográfica sobre o assunto, e disponível apenas *on-line* em [Ros 95], ou nos CDs *TechNet* da própria Microsoft. O resto da documentação são os arquivos de ajuda que acompanham os SDKs para o Windows NT da Microsoft.

Não existe lista de discussão específica sobre o assunto na Internet, mas na lista sobre WinSNMP este assunto é amplamente aceito para discussão.

O serviço SNMP do Windows NT é um agente SNMP extensível que permite a desenvolvedores adicionar DLLs para servirem MIBs adicionais. O agente por si só não contém instrumentação para nenhuma MIB. Por outro lado, ele é responsável por receber as requisições SNMP que chegam na estação ou servidor NT e passá-las para as DLLs apropriadas para sua resolução. Os dados de resposta são então retornados para o agente, que é responsável por retornar a resposta para a estação de gerenciamento que fez a requisição. O agente extensível é também capaz de gerar *traps* em função de qualquer uma das DLLs que servem MIBs. Estes *traps* são enviados a estações de gerenciamento para as quais o agente extensível esteja configurado para notificar [Ros 95].

Conforme mostra a figura 7.3, o agente extensível utiliza a rede através da DLL Winsock (*Windows Sockets*). Esta DLL funciona sobre os transportes IP e IPX, e portanto isto habilita o agente extensível a ser gerenciado a partir de consoles de gerenciamento de redes operando sobre o IP ou o IPX. O agente pode ainda ser configurado para aceitar requisições SNMP somente de determinados endereços IP ou IPX particulares. As *communities* válidas também podem ser especificadas na configuração do agente.

Figura 7.3 - A Arquitetura do Agente Extensível Microsoft



O agente extensível determina quais DLLs subagentes, chamadas de *extension agents*, ele deve carregar, através de valores no registro do Windows NT, sob a chave **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet \Services\SNMP\Parameteres\ExtensionAgents**. Toda vez que o agente extensível é inicializado, esta chave é consultada, e para cada *extension agent DLL* existente, o agente extensível invoca a função SnmpExtensionInit. Através desta função o agente extensível constrói uma tabela das MIBs suportadas, contendo duas colunas: uma coluna para o prefixo de OID suportado pela DLL, e outra contendo o nome da DLL correspondente. A partir do momento que todas as DLLs são carregadas e a tabela inicializada, o agente extensível está pronto para processar requisições SNMP.

Todo *extension agent* construído para o agente extensível da Microsoft, deve suportar pelo menos as seguintes funções da API:

- SnmpExtensionInit: o *extension agent* retorna o OID do espaço de nomes suportado e o endereço de um evento a ser sinalizado pelo *extension agent* quando este quiser transmitir um *trap*;
- SnmpExtensionQuery: o *extension agent* processa a requisição Set/Get/GetNext associada com a *variable bindings* (lista de variáveis) passada como parâmetro pelo agente extensível;
- SnmpExtensionTrap: retorna com a Trap PDU formatada a ser transmitida.

Adicionalmente, um *extension agent* pode suportar uma quarta função, SnmpExtensionInitEx. Esta função permite que um único *extension agent* suporte múltiplas MIBs ou múltiplas visões de MIBs a partir de uma única DLL.

O agente extensível da Microsoft, apesar de possuir suporte transparente às duas versões do protocolo SNMP, e ser bastante versátil, possui alguns problemas que precisam ser resolvidos:

- não é possível configurar o agente para esperar por requisições SNMP em outras portas que não as portas padrão SNMP, ou seja, a porta 161 para requisições Get/Set/GetNext. A porta 162 serve para uma aplicação de gerenciamento esperar pela chegada de *traps* vindos de agentes SNMP;
- o envio de *traps* por parte de um *extension agent* é restrito ao formato da Trap PDU do SNMPv1. Não é possível a um agente enviar uma Notification do SNMPv2;

- o modelo de *communities* possui problema grave de segurança, uma vez que todas as *communities* aceitas pelo agente possuem acesso *read-write*. Não é possível especificar o tipo de acesso que uma *community* possui.

Espera-se que o agente extensível Microsoft que acompanhará o Windows NT 5.0 resolva estes e outros pequenos problemas. Aparentemente, a Microsoft não tem intenção de abandonar seu agente extensível em favor do IETF AgentX, que ainda deve demorar um pouco até se tornar uma especificação confiável.

7.2.3 IETF AgentX

AgentX é uma abreviação para o *Agent Extensibility Protocol* do IETF. Consiste em uma proposta para definir uma base padronizada para o desenvolvimento de aplicações agente baseadas em um agente extensível.

É ainda muito cedo para dizer quando o AgentX será um padrão correto, livre de erros, redundâncias, e largamente aceito no mercado. Entretanto, as expectativas são muito boas, uma vez que nomes de pessoas importantes na comunidade SNMP fazem parte do grupo de trabalho responsável por seu desenvolvimento. Entre eles, Bob Natale (autor da WinSNMP), Don Ryan da Microsoft (envolvido com o agente extensível da Microsoft), Aleksey Romanov (autor da antiga WinSNMP/Agent API), e Jeff Case (autor de várias RFCs sobre SNMP).

Dentro do escopo do AgentX, um *master agent* consiste no agente principal responsável pelas tarefas básicas relacionadas ao protocolo, enquanto os *subagents* consistem nos componentes responsáveis por alguma MIB. Um protocolo (AgentX) é usado para comunicação entre o *master agent* e seus *subagents*.

O maior problema em não se ter um base para o desenvolvimento de agentes extensíveis, é que freqüentemente dispositivos gerenciados tornam-se complexas coleções de componentes gerenciáveis que foram instalados de forma independente no nó gerenciado. Cada componente provê instrumentação para os objetos gerenciados definidos no módulo de MIB que ele implementa. Fazer com que vários agentes SNMP de diferentes fornecedores funcionem na mesma máquina pode ser uma tarefa não muito fácil. Tanto o SNMPv1 como o SNMPv2 nada falam sobre como objetos gerenciados podem ser dinamicamente adicionados ou removidos de um agente em um nó particular [Dan 96].

Esta necessidade fez com que surgissem uma variedade de "agentes extensíveis", os quais tipicamente compreendem:

- um agente principal que escuta requisições SNMP na porta e protocolo de transporte padrões;
- um conjunto de subagentes, cada um responsável por um conjunto de objetos gerenciados, isto é, a sua MIB;
- um mecanismo de conversação entre o agente principal e os subagentes, permitindo que os subagentes indiquem ao agente principal por quais objetos eles são responsáveis, e permitindo que o agente principal invoque os subagentes para processar as requisições SNMP que chegam pela rede;
- um conjunto de ferramentas que auxiliam o desenvolvimento de subagentes, e uma API *runtime* que esconde muito da operação do protocolo entre o agente principal e os subagentes.

O problema é que estas soluções são proprietárias. Um subagente escrito para um produto em particular não poderá ser facilmente portado para outro sem que antes a nova plataforma seja plenamente compreendida. Com a falta de um padrão Internet nesta área, um desenvolvedor hoje deve suportar múltiplos ambientes de agentes/subagentes (APIs) para poder suportar várias plataformas. O AgentX visa resolver estes problemas.

Um *master agent* dentro do contexto do AgentX deve executar as seguintes funções:

- aceitar o estabelecimento de sessões de *subagents*;
- aceitar o registro de regiões de MIB pelos *subagents*;
- enviar e aceitar mensagens do protocolo SNMP nos endereços de transporte especificados no agente;
- implementar o modelo administrativo do SNMP (MIB *views*, controle de acesso, etc.);
- prover instrumentação para os objetos de MIB definidos na RFC 1907 [Cas 96e];
- enviar e receber mensagens do protocolo AgentX para acessar informação de gerenciamento, baseado no registro corrente de regiões de MIB;
- propagar notificações (*traps* e *notifications*) em favor dos *subagents*.

Uma entidade atuando no papel de um *subagent* no modelo AgentX deve executar as seguintes funções:

- iniciar uma sessão AgentX com o *master agent*;
- registrar regiões de MIB junto ao *master agent*;
- instanciar objetos gerenciados;
- associar OIDs das regiões de MIBs registradas a variáveis reais;
- executar operações de gerenciamento sobre as variáveis;
- iniciar notificações.

O documento que atualmente descreve o AgentX [Dan 96] detalha todas os procedimentos que o *master agent* e os *subagents* devem adotar quando do recebimento de requisições SNMP, comunicação através do protocolo AgentX, etc.

Os aspectos de segurança (aplicação o modelo administrativo) ficam a cargo do *master agent*.

Capítulo 8

IMPLEMENTANDO A *NETWORK CLIENT APPLICATION MIB*

8. Implementando a *Network Client Application MIB*

A implementação de uma MIB, assim como a implementação de qualquer *software*, exige a escolha de algumas opções:

- Plataforma de *hardware*: foi escolhida a plataforma compatível com o IBM PC, por ser a plataforma mais difundida entre as máquinas utilizadas como cliente de rede;
- Sistema Operacional: o sistema operacional alvo foi o Windows 95, pelos mesmos motivos da escolha do *hardware*. Além disso, um programa desenvolvido para o Windows 95 executa perfeitamente sobre o Windows NT, mesmo em outras plataformas de *hardware* diferentes do IBM PC;
- Linguagem de programação: programar para a API do Windows na linguagem C é o caminho mais natural, e portanto a linguagem C++ será utilizada.

A implementação ainda possui algumas limitações decorrentes da dificuldade encontrada na implementação das operações de controle. Dessa forma, algumas funções de controle e de notificações ainda precisam ser resolvidas, conforme descrito no item sobre resultados obtidos.

8.1 Instalando o Novo Agente no Ambiente de Testes

O ambiente de testes foi composto de três microcomputadores interligados em uma rede local TCP/IP. Dois executavam o Windows 95, enquanto uma das máquinas executava o Windows NT 4.0.

A instalação do agente foi tão simples quanto copiar o arquivo DLL para o disco local de cada máquina, e adicionar uma entrada no registro do sistema operacional para que o agente extensível SNMP da Microsoft o reconhecesse.

Para gerenciar o novo agente, foi utilizado um *MIB Browser* de domínio público, executando sobre o ambiente Windows, e que é capaz de gerar operações SET, GET e GETNEXT.

8.2 Experimentos Realizados no Ambiente de Testes

Uma vez instalados os agentes, os seguintes experimentos foram realizados:

- em cada estação foram disparadas aplicações clientes de rede de diferentes serviços, a saber: WWW, FTP e TELNET;
- para cada tipo de aplicação, foram abertas conexões para três *hosts* diferentes;
- utilizando o *MIB Browser*, pôde-se verificar que as tabelas corretamente reportaram as aplicações executando e as conexões ativas;
- em cada tipo de aplicação, uma das conexões foi encerrada pelo próprio aplicativo;

- verificou-se com o *MIB Browser* que as tabelas com conexões ativas não continham mais as entradas correspondentes às conexões encerradas, e que a tabela de histórico de conexões agora as continha;
- utilizando o *MIB Browser*, operações SET alteraram o estado administrativo de cada aplicação para aborted;
- inspecionando as tabelas, verificou-se que as aplicações foram encerradas, sendo movidas para a tabela de histórico de aplicações. As conexões de cada aplicação também foram encerradas e movidas para a tabela correspondente;
- utilizando o *MIB Browser*, foi possível criar entradas na tabela de eventos a notificar, utilizando operações SET;
- analogamente, através do *MIB Browser*, foi possível remover entradas na tabela de eventos a notificar, utilizando operações SET.

8.3 Resultados Obtidos

Com a realização dos experimentos, verificou-se que a implementação do agente é viável, e que a sua instalação em um ambiente de redes real pode dar uma idéia mais apurada de sua utilidade.

Como as funções de controle das conexões ainda não haviam sido implementadas, não foi possível verificar como o sistema como um todo se comportaria nesta situação. Além disso, o estado administrativo de suspender uma aplicação não foi ainda implementado, por restrições no sistema operacional.

Apesar de ter sido possível adicionar e remover entradas na tabela de notificações a enviar com o uso de operações SET, o agente ainda não implementa a

notificação de tais eventos e, portanto, não foi possível capturar notificações do agente.

Capítulo 9

CONSIDERAÇÕES FINAIS E CONCLUSÕES

9. Considerações Finais e Conclusões

O gerenciamento de redes surge das necessidades do administrador do ambiente de redes, que então elaborará um plano de gerenciamento, muitas vezes sendo necessária a construção das próprias soluções.

O Gerenciamento de Redes ideal é aquele que começa desde a fase de projeto da rede de computadores. Desde a decisão de que características os equipamentos que vão compor a rede deverão ter até a escolha do *software* de gerenciamento, é uma tarefa nem sempre muito óbvia. Entretanto, o projetista de redes deve tentar antecipar ao máximo as necessidades de gerenciamento da rede que vai ser construída. A decisão por incluir gerenciamento somente no futuro pode acarretar em custos muito mais altos tanto financeiramente, pela troca de equipamentos, como nos problemas que podem surgir pela falta de um gerenciamento eficaz.

9.1 Conclusões

Nesta tentativa de resolver uma necessidade de gerenciamento, o modelo de gerenciamento Internet foi escolhido pela sua simplicidade e também por sua conveniência para o problema em particular.

Com os testes realizados, a arquitetura Internet mostrou-se (mais uma vez) ainda suficiente para resolver a maior parte dos problemas práticos com o gerenciamento de redes.

O modelo de gerenciamento Internet mostrou-se também um modelo em evolução. As novas operações da segunda versão do protocolo e, principalmente, as novas definições da SMIv2 são uma grande melhoria do primeiro modelo. No caso específico deste trabalho, a SMIv2 mostrou-se mais rica nos tipos e construções disponíveis para a definição de objetos e seus relacionamentos em uma MIB. A possibilidade de se criar e destruir linhas em uma tabela é um exemplo desta evolução.

A metodologia para o desenvolvimento de MIBs utilizada neste trabalho apresenta inclui diversos aspectos que não foram utilizados quando do desenvolvimento da *Network Client Application MIB*. Entretanto, esta metodologia pode servir como um guia para o desenvolvimento de módulos de MIB no contexto de uma organização.

O uso de ferramentas para o desenvolvimento de MIBs também mostrou-se extremamente útil. O uso do compilador de MIBs ajudou a identificação de alguns problemas na especificação que dificultariam seu uso em uma aplicação real.

As aplicações de redes mais populares em uso hoje são voltadas para a Internet. Um agente rodando em máquinas PC que forneça informações sobre as aplicações ali sendo executadas, pode ser de grande valia para o administrador que deseja conhecer a utilização que seus usuários fazem da rede, bem como pode permitir algum controle sobre estas aplicações.

Pelas experiências realizadas, é possível concluir que um agente com estas características pode ser extremamente útil nas áreas funcionais de desempenho, configuração e de segurança. O desempenho pode ser melhorado através da monitoração de tráfego e tomada de medidas preventivas e reconfiguração da máquina cliente, enquanto a segurança pode ser aumentada quando do controle do gerente da rede sobre que aplicações os usuários podem executar ou não.

Considerando que os sistemas operacionais para microcomputadores têm cada dia mais aprimorado sua segurança, é possível implementar este grau de controle sem que os usuários contornem a situação burlando os esquemas de controle do agente SNMP proposto.

A aplicação prática deste trabalho seria útil em um ambiente onde o administrador ou gerente de redes deseja ter um controle mais apurado sobre as atividades dos usuários na rede. Tal ambiente muitas vezes está presente nas universidades, pelo fato de que laboratórios de computadores costumam ser compartilhados por muitos alunos. Com poucos recursos computacionais disponíveis para os usuários, é interessante controlar as atividades dos mesmos, no sentido de que as máquinas sejam usadas para tarefas que condizem com os objetivos da universidade.

Outro ambiente interessante para o controle de aplicações dos usuários é o de empresas, que normalmente não desejam que seus funcionários passem o dia "brincando" na rede.

Como conclusão final, este trabalho pode ser visto como uma experiência no desenvolvimento de uma MIB e sua implementação. Enquanto a MIB proposta pode ter utilidade prática em um ambiente de redes típico, a experiência do desenvolvimento do agente pode ser útil para trabalho futuros.

9.2 Trabalhos Futuros

Muita coisa ainda pode ser feita no contexto deste trabalho. Em relação à metodologia utilizada, poder-se-ia criar um ambiente auxiliado por computador para o desenvolvimento de novas MIBs (poderíamos chamá-lo de CAMD - *Computer Aided MIB Development*).

Com alguns refinamentos, poder-se-ia submeter a MIB desenvolvida para padronização junto ao IETF. Adicionalmente, seria interessante contatar a IANA para a obtenção de um OID privado ou experimental para o registro desta MIB e de outros futuros trabalhos.

Por fim, uma vez provada a importância do agente implementado, o próximo passo seria a construção de uma aplicação específica para o seu gerenciamento. A melhor opção seria o desenvolvimento desta aplicação sobre uma API padronizada, como a WinSNMP ou a SNMP++. Uma aplicação escrita para WinSNMP ou SNMP++ poderia executar, além do ambiente Windows, sobre outros sistemas operacionais e em plataformas de gerenciamento que já suportam ou venham a suportar estas APIs.

ANEXO 1 - DESCRIÇÃO DA MIB

```
CLIENTAPPL-MIB DEFINITIONS ::= BEGIN

IMPORTS
    OBJECT-TYPE, MODULE-IDENTITY, NOTIFICATION-TYPE, experimental,
    Counter32, Gauge32, Integer32, TimeTicks
        FROM SNMPv2-SMI
    DisplayString
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF;


-- Network Client Application MIB

clientApplMIB MODULE-IDENTITY
    LAST-UPDATED "9703231400Z"
    ORGANIZATION "UFSC-CPGCC"
    CONTACT-INFO
        "Celso Kopp Webber
         Postal: Caixa Postal 476,
         Campus Universitario - Trindade
         88040-900
         Florianopolis - SC
         Brazil
         Phone: (048) 231-9738
         Email: celsinho@inf.ufsc.br"
    DESCRIPTION
         "The MIB module defining network client applications"
    ::= { experimental 9999  }

clientApplRun           OBJECT IDENTIFIER ::= { clientApplMIB 1 }
clientApplConn          OBJECT IDENTIFIER ::= { clientApplMIB 2 }
clientApplOldRun        OBJECT IDENTIFIER ::= { clientApplMIB 3 }
clientApplOldConn       OBJECT IDENTIFIER ::= { clientApplMIB 4 }
clientApplNotifications OBJECT IDENTIFIER ::= { clientApplMIB 5 }
clientApplConformance   OBJECT IDENTIFIER ::= { clientApplMIB 6 }


-- General application table. The table should have a size
-- enough to hold a reasonable number of applications running
-- at the same time

clientApplRunTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF ClientApplRunEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "The table holding all network applications currently
          running on the host"
    ::= { clientApplRun 1 }
```

```
clientApplRunEntry OBJECT-TYPE
    SYNTAX      ClientApplRunEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry associated with a network client application"
    INDEX { clientApplRunIndex }
    ::= { clientApplRunTable 1 }

ClientApplRunEntry ::= SEQUENCE {
    clientApplRunIndex
        Integer32,
    clientApplRunName
        DisplayString,
    clientApplRunVersion
        DisplayString,
    clientApplRunUpTime
        TimeTicks,
    clientApplRunConnections
        Gauge32,
    clientApplRunOldConnections
        Counter32,
    clientApplRunInOctets
        Counter32,
    clientApplRunOutOctets
        Counter32,
    clientApplRunOperStatus
        INTEGER,
    clientApplRunAdminStatus
        INTEGER
}

clientApplRunIndex OBJECT-TYPE
    SYNTAX      Integer32 (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An index that uniquely identify the client network
         application running at a given moment on the host."
    ::= { clientApplRunEntry 1 }

clientApplRunName OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The name the network client application reports to
         the Operating System."
    ::= { clientApplRunEntry 2 }

clientApplRunVersion OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number the network client application reports to
```

```
        the Operating System as its version."
    ::= { clientApplRunEntry 3 }

clientApplRunUpTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The time the network client application is being run,
         expressed in time ticks."
    ::= { clientApplRunEntry 4 }

clientApplRunConnections OBJECT-TYPE
    SYNTAX      Gauge32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of current connections of this network client
         application."
    ::= { clientApplRunEntry 5 }

clientApplRunOldConnections OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number connections already closed for this network
         client application."
    ::= { clientApplRunEntry 6 }

clientApplRunInOctets OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of octets this network client application has
         received since it was started."
    ::= { clientApplRunEntry 7 }

clientApplRunOutOctets OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of octets this network client application has
         sent since it was started."
    ::= { clientApplRunEntry 8 }

clientApplRunOperStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    running(1),
                    suspended(2),
                    blocked(3)
                }
    MAX-ACCESS  read-only
    STATUS      current
```

```
    DESCRIPTION
        "The operating status of the network client application.
         'running' means the application is up and running, the
         'suspended' state means the application was suspended,
         either locally or remotelly, the 'blocked' state means
         the application is waiting for something and is blocked
         by the operating system."
    ::= { clientApplRunEntry 9 }

clientApplRunAdminStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    running(1),
                    suspended(2),
                    aborted(3)
                }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administrative status a management station can put
         an network client application in. 'running' means that
         the application should be put running. 'suspended' means
         that the application should be temporarily suspended and
         that could be re-enabled by setting the
         clientApplRunAdminStatus to 'running'. The 'aborted' state
         is used to issue the operating system to abort the
         application. Since the application will be no longer
         running, its entries in the clientApplConnTable should
         be moved the the clientApplOldConnTable."
    ::= { clientApplRunEntry 10 }


-- The clientApplConnTable contains all active connections of the
-- currently running network client applications. The table should
-- have such a size that it can hold all active connections opened
-- at a given moment. A sugested size is about 5 times the size
-- of the application table

clientApplConnTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF ClientApplConnEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The table holding all connections of the applications
         currently running on the host"
    ::= { clientApplConn 1 }

clientApplConnEntry OBJECT-TYPE
    SYNTAX      ClientApplConnEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry associated with a network client application
         connection"
    INDEX { clientApplRunIndex, clientApplConnIndex }
    ::= { clientApplConnTable 1 }
```

```
ClientApplConnEntry ::= SEQUENCE {
    clientApplConnIndex
        Integer32,
    clientApplConnDestination
        DisplayString,
    clientApplConnProtocol
        INTEGER,
    clientApplConnPort
        Integer32,
    clientApplConnUpTime
        TimeTicks,
    clientApplConnInOctets
        Counter32,
    clientApplConnOutOctets
        Counter32,
    clientApplConnOperStatus
        INTEGER,
    clientApplConnAdminStatus
        INTEGER
}

clientApplConnIndex OBJECT-TYPE
    SYNTAX      Integer32 (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An index that uniquely identifies an connection
         within its network client application being run."
    ::= { clientApplConnEntry 1 }

clientApplConnDestination OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The destination name or IP address to which this
         connection was made."
    ::= { clientApplConnEntry 2 }

clientApplConnProtocol OBJECT-TYPE
    SYNTAX      INTEGER {
                    tcp(1),
                    udp(2)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The transport protocol used for this connection."
    ::= { clientApplConnEntry 3 }

clientApplConnPort OBJECT-TYPE
    SYNTAX      Integer32 (1..999999)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of the port this connection is using."
```

```
    ::= { clientApplConnEntry 4 }

clientApplConnUpTime OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The time (in time ticks) since this connection has
         begun."
    ::= { clientApplConnEntry 5 }

clientApplConnInOctets OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of octets this connection has received
         since it was initiated."
    ::= { clientApplConnEntry 6 }

clientApplConnOutOctets OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of octets this connection has sent since
         it was initiated."
    ::= { clientApplConnEntry 7 }

clientApplConnOperStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    opening(1),
                    closing(2),
                    ready(3),
                    busy(4)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current operation status of the connection.
         'opening' means the connection is initiating, while
         'closing' means the connection is ending. 'ready'
         means the connection is ready to send or receive data.
         'busy' means that one end of the connection is busy
         and the other party should wait."
  ::= { clientApplConnEntry 8 }

clientApplConnAdminStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    resume(1),
                    suspend(2),
                    abort(3)
                }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
```

```
        "The administration status of the association. 'suspend'
         should be used to temporarily suspend an association,
         while 'resume' should be used to restore the normal
         operation of the association. 'abort' could be used to
         cancel the association."
    ::= { clientApplConnEntry 9 }


-- The clientApplOldRunTable contains all applications that
-- were already closed since the system was started. Anytime
-- an application is closed, its entry in the 'RunTable'
-- is moved to this table. The table should be of a reasonable
-- size so that it should be able to have some relevant
-- history of the applications that ran in the system. Since
-- this table is always crescent, whenever its full capacity
-- is used, older entries should be removed first in favor of
-- the new entries.

clientApplOldRunTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF ClientApplOldRunEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
         "The table holding all network applications already
          finished on the host"
    ::= { clientApplOldRun 1 }

clientApplOldRunEntry OBJECT-TYPE
    SYNTAX      ClientApplOldRunEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry associated with a finished network client
         application"
    INDEX { clientApplRunIndex }
      -- the index is unique both for active and finished apps.
    ::= { clientApplOldRunTable 1 }

ClientApplOldRunEntry ::= SEQUENCE {
    clientApplOldRunName
        DisplayString,
    clientApplOldRunVersion
        DisplayString,
    clientApplOldRunDuration
        TimeTicks,
    clientApplOldRunConnections
        Integer32,
    clientApplOldRunInOctets
        Integer32,
    clientApplOldRunOutOctets
        Integer32,
    clientApplOldRunEndStatus
        INTEGER
}

clientApplOldRunName OBJECT-TYPE
```

```
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The name the network client application reports to
         the Operating System."
    ::= { clientApplOldRunEntry 1 }

clientApplOldRunVersion OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number the network client application reports to
         the Operating System as its version."
    ::= { clientApplOldRunEntry 2 }

clientApplOldRunDuration OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The time the network client application remained
         running, expressed in time ticks."
    ::= { clientApplOldRunEntry 3 }

clientApplOldRunConnections OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of connections of this network client
         application."
    ::= { clientApplOldRunEntry 4 }

clientApplOldRunInOctets OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of octets this network client application has
         received since it was started until it was finished."
    ::= { clientApplOldRunEntry 5 }

clientApplOldRunOutOctets OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of octets this network client application has
         sent since it was started until it was finished."
    ::= { clientApplOldRunEntry 6 }

clientApplOldRunEndStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    normal(1),
```

```
                    abortion(2)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The reason why this application has ended. 'normal' means
         the usual exit function of the application. 'abortion'
         means that the application was ended either locally by
         the user, or remotely by the management station."
    ::= { clientApplOldRunEntry 7 }


-- The clientApplOldConnTable contains all connections that
-- were already closed since the system was started. Anytime
-- a connection is closed, its entry in the previous table
-- is moved to this table. The table should be of a reasonable
-- size so that it should be able to have some relevant
-- history of the applications connections. Since this table
-- is always crescent, whenever its full capacity is used,
-- older entries should be removed first in favor of the new
-- entries.

clientApplOldConnTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF ClientApplOldConnEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The table holding all already closed connections on
         the host"
    ::= { clientApplOldConn 1 }

clientApplOldConnEntry OBJECT-TYPE
    SYNTAX      ClientApplOldConnEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry associated with a network client application
         association already closed"
    INDEX { clientApplRunIndex, clientApplConnIndex }
      -- the indices are unique both for active and finished conn.
    ::= { clientApplOldConnTable 1 }

ClientApplOldConnEntry ::= SEQUENCE {
    clientApplOldConnDestination
        DisplayString,
    clientApplOldConnProtocol
        INTEGER,
    clientApplOldConnPort
        Integer32,
    clientApplOldConnDuration
        TimeTicks,
    clientApplOldConnInOctets
        Integer32,
    clientApplOldConnOutOctets
        Integer32,
    clientApplOldConnEndStatus
```

```
        INTEGER
}

clientApplOldConnDestination OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The destination name or IP address to which this
         connection was made."
    ::= { clientApplOldConnEntry 1 }

clientApplOldConnProtocol OBJECT-TYPE
    SYNTAX      INTEGER {
                    tcp(1),
                    udp(2)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The protocol used for this connection."
    ::= { clientApplOldConnEntry 2 }

clientApplOldConnPort OBJECT-TYPE
    SYNTAX      Integer32 (1..999999)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of the port this association was using."
    ::= { clientApplOldConnEntry 3 }

clientApplOldConnDuration OBJECT-TYPE
    SYNTAX      TimeTicks
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The time (in time ticks) this association remained
         active."
    ::= { clientApplOldConnEntry 4 }

clientApplOldConnInOctets OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of octets this connection received
         since it was initiated until it was finished."
    ::= { clientApplOldConnEntry 5 }

clientApplOldConnOutOctets OBJECT-TYPE
    SYNTAX      Integer32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of octets this connection sent since
         it was initiated until it was finished."
```

```
    ::= { clientApplOldConnEntry 6 }

clientApplOldConnEndStatus OBJECT-TYPE
    SYNTAX      INTEGER {
                    normal(1),
                    server(2),
                    timeout(3),
                    abortion(4)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The reason why this connection has ended. 'normal'
         means the usual client application connection closing.
         'server' means that the server on the other side of the
         connection has finished it. 'timeout' means that the
         connection was terminated because a timeout error.
         'abortion' means that the connection was ended either
         locally  by the user, or remotely by the management
         station."
    ::= { clientApplOldConnEntry 7 }


-- Some additional scalar objects with read-write MAX-ACCESS could
-- be used to control the tables sizes. See ietf-draft-sysappl-mib


-- Conformance Macros

clientApplMIBCompliances  OBJECT IDENTIFIER ::= {
clientApplConformance 1 }
clientApplMIBGroups       OBJECT IDENTIFIER ::= {
clientApplConformance 2 }

clientApplMIBCompliance MODULE-COMPLIANCE
    STATUS current
    DESCRIPTION
        "Describes the requirements for conformance to the
         Network Client Application MIB"
    MODULE  -- this module
        MANDATORY-GROUPS { clientApplRunGroup,
                           clientApplConnGroup,
                           clientApplOldRunGroup,
                           clientApplOldConnGroup,
                           clientApplNotificationsGroup }
    ::= { clientApplMIBCompliances 1 }

clientApplRunGroup OBJECT-GROUP
    OBJECTS { clientApplRunName,
              clientApplRunVersion,
              clientApplRunUpTime,
              clientApplRunConnections,
              clientApplRunOldConnections,
              clientApplRunInOctets,
              clientApplRunOutOctets,
```

```
                clientApplRunOperStatus,
                clientApplRunAdminStatus }
    STATUS  current
    DESCRIPTION
        "The client application running group contains
         information about network client applications
         running at a given moment."
    ::= { clientApplMIBGroups 1 }

clientApplConnGroup OBJECT-GROUP
    OBJECTS { clientApplConnDestination,
              clientApplConnProtocol,
              clientApplConnPort,
              clientApplConnUpTime,
              clientApplConnInOctets,
              clientApplConnOutOctets,
              clientApplConnOperStatus,
              clientApplConnAdminStatus }
    STATUS  current
    DESCRIPTION
        "The client application connection group contains
         information about the active connections of network
         client applications at a given moment."
    ::= { clientApplMIBGroups 2 }

clientApplOldRunGroup OBJECT-GROUP
    OBJECTS { clientApplOldRunName,
              clientApplOldRunVersion,
              clientApplOldRunDuration,
              clientApplOldRunConnections,
              clientApplOldRunInOctets,
              clientApplOldRunOutOctets,
              clientApplOldRunEndStatus }
    STATUS  current
    DESCRIPTION
        "The client application old running group contains
         information about network client applications that
         already terminated in this system."
    ::= { clientApplMIBGroups 3 }

clientApplOldConnGroup OBJECT-GROUP
    OBJECTS { clientApplOldConnDestination,
              clientApplOldConnProtocol,
              clientApplOldConnPort,
              clientApplOldConnDuration,
              clientApplOldConnInOctets,
              clientApplOldConnOutOctets,
              clientApplOldConnEndStatus }
    STATUS  current
    DESCRIPTION
        "The client application old connection group contains
         information about the already closede connections of
         network client applications still running or already
         finished."
    ::= { clientApplMIBGroups 4 }
```

```
clientApplNotificationsGroup OBJECT-GROUP
    OBJECTS { clientApplOldConnDestination } -- just for testing
    STATUS  current
    DESCRIPTION
        "The notifications group contains the objects related
         to notifications to be sent about network client
         applications and its connections."
    ::= { clientApplMIBGroups 5 }

END
```

REFERÊNCIAS BIBLIOGRÁFICAS

[Bri 92] BRISA. Gerenciamento de Redes: Uma Abordagem de Sistemas Abertos. São Paulo: Makron Books, 1992.

[Cas 90] Case, J.; Fedor, M.; Schoffstall, M.; Davin, J. Simple Network Management Protocol, STD 15, RFC 1157. SNMP Research, Performance Systems International; MIT Laboratory for Computer Science, May 1990.

[Cas 96a] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S. Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1902. January, 1996.

[Cas 96b] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S. Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1903. January, 1996.

[Cas 96c] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S. Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1904. January, 1996.

[Cas 96d] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S. Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1905. January, 1996.

[Cas 96e] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S. Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1907. January, 1996.

[Cas 96f] Case, J.; McCloghrie, K.; Rose, M.; Waldbusser, S. Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework, RFC 1908. January, 1996.

[Dan 96] Daniele, Mike; Wijnen, Bert; Editor. Agent Extensibility (AgentX) Protocol, Internet-Draft. Digital Equipment Corporation; T.J. Watson Research Center, IBM Corp., November 1996.

[ISO 87] International Organization for Standardization. Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax One (ASN.1). International Standard 8824, December 1987.

[Kil 94] Kille, S.; Freed, N. Network Services Monitoring MIB, RFC 1565. ISODE Consortium; Innosoft International, January 1994.

[McC 91] McCloghrie, K.; Rose, M.; Editors. Management Information Base for Network Management of TCP/IP-based internets: MIB-II, STD 17, RFC 1213. Hughes LAN Systems; Performance Systems International, March 1991.

[Mel 96] Mellquist, Peter Erik. SNMP++ - An Open Specification for Object Oriented Network Management Development Using C++. Revision 2.5f. Hewllet Packard Company, 1996.

[Nat 95] Natale, Bob. Windows SNMP - The Open Interface for Programming Network Management Applications using the Simple Network Management Protocol under Microsoft Windows, WinSNMP/Manager API. American Computer & Electronics Corporation, August 1995.

[Per 97] Perkins, David; McGinnis, Evan. Understanding SNMP MIBs. Upper Saddle Riber, NJ: Prentice Hall PTR, 1997.

[Rom 94] Romanov, A.; White, W. H.; Wilson, P. Windows SNMP Agent - The Open Interface for Programming the Extensible SNMP V1 and V2 Agent Under Microsoft Windows. Paul Freeman Associates; Digital Equipment Corporation, December 1994.

[Rose 90] Rose, M.; McCloghrie, K. Structure and Identification of Management Information for TCP/IP-based internets, STD 16, RFC 1155. Performance Systems International; Hughes LAN Systems, May 1990.

[Rose 91] Rose, M.; McCloghrie, K.; Editors. Concise MIB Definitions, STD 16, RFC 1212. Performance Systems International; Hughes LAN Systems, March 1991.

[Ros 95] Rosato, Steve. Microsoft Windows NT SNMP Agent Extensions. http://www.microsoft.com/syspro/technet/boes/bo/winntas/technote/nt307.htm. Microsoft Corporation, 1995.

[Sap 96] Saperia, Jonathan; Krupczak, Cheryl; Sturm, Rick; Weinstock, Jonathan. Definitions of Managed Objects for Applications, Internet-Draft. BGS Systems Inc.; Empire Technologies, Inc.; Enterprise Management Professional Services, Inc.; General Instrument Corporation, September 1996.

[Sta 93] Stallings, William. SNMP, SNMPv2 and CMIP. The Practical Guide to Network-Management Standards. Reading, Mass.: Addison-Wesley, 1993.

[Wal 95] Waldbusser, S. Remote Network Monitoring Management Information Base, RFC 1757. Carnegie Mellon University, February 1995.