UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM SISTEMA DE MONITORAMENTO DE DESEMPENHO DE REDES DE COMPUTADORES BASEADO NO PROTOCOLO SNMPV3.

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

ANDERSON KARING

BLUMENAU, NOVEMBRO/2002

2002/2-03

PROTÓTIPO DE UM SISTEMA DE MONITORAMENTO DE DESEMPENHO DE REDES DE COMPUTADORES BASEADO NO PROTOCOLO SNMPV3.

ANDERSON KARING

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

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Francisco Adell Péricas — Orientador na FURB

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Francisco Adell Péricas

_____

_____

AGRADECIMENTOS

Agradeço a todos os professores do curso de Bacharelado em Ciências da Computação da Universidade Regional de Blumenau pela ajuda prestada ao longo do curso, principalmente aos professores Francisco Adell Péricas por ter me orientado neste trabalho final e o professor Roberto Heinzle pelo apoio dado num momento crítico do curso quando eu pensava em desistir. Muito obrigado pela paciência, preocupação e dedicação.

Agradeço a todos os novos amigos que conheci em Blumenau no decorrer da Faculdade, principalmente ao Renato, Rafael, Fernando, Carlos, Erasmo, Hensel, Gilson, Edson “tatu”, Marcelo, e a todo o pessoal do NI, especialmente ao Fábio por acreditar em meu potencial. Na ala feminina agradeço à Marilda, Silvanira, Fernanda e Michele. Obrigado a todos vocês pela força, apoio e companhia no decorrer do curso.

Agradeço a toda galera da sala, que mesmo nos momentos mais difíceis sempre ajudavam uns aos outros.

Agradeço principalmente aos meus pais Valmor e Renate e aos meus irmãos Max e Ana, por terem me dado todo o apoio nessa luta, não fosse a dedicação deles, não poderia estar aqui agora redigindo meu trabalho final.

Agradeço aos motoristas da Geneve por levarem e buscarem de Brusque para Blumenau, a mim e a todos os demais estudantes de Brusque, com segurança durante estes seis anos.

Agradeço ao pessoal da Igreja Luterana de Bateias, por ter permitido que eu deixasse minha “*bike*” lá guardada enquanto estudava, facilitando assim minha vida.

Agradeço também ao Governo Estadual pelo auxílio financeiro prestado, e podem ter certeza: fiz valer cada centavo em mim investido.

Finalmente, agradeço a Marcela e a Bárbara por serem as garotas mais incríveis (e lindas!) que já conheci, e por terem compartilhado dos momentos mais marcantes da minha vida até hoje.

EPÍGRAFE

“Os verdadeiros vencedores não são aqueles que chegam na frente, mas sim aqueles que com muita garra, paixão e dedicação chegam ao final”.

(autor desconhecido)

SUMÁRIO

LISTA DE FIGURAS ..... **VII**

LISTA DE ABREVIATURAS E LISTA DE SIGLAS ..... **VIII**

LISTA DE ABREVIATURAS E LISTA DE SIGLAS ..... **IX**

RESUMO ..... **XIII**

ABSTRACT ..... **XIV**

1 INTRODUÇÃO **1**

1.1 OBJETIVOS DO TRABALHO 3
1.2 ESTRUTURA DO TRABALHO 3

2 FUNDAMENTAÇÃO TEÓRICA **5**

2.1 BREVE HISTÓRICO DE REDES DE COMPUTADORES 5
2.2 REDES DE COMPUTADORES 6
2.2.1 Uso Das Redes De Computadores 7
2.2.2 Hardware De Rede 8
2.2.3 Software De Rede 9

3 GERÊNCIA DE REDES DE COMPUTADORES **13**

3.1 ARQUITETURA DE UM SISTEMA DE GERENCIAMENTO DE REDES 16
3.2 GERENCIAMENTO DE DESEMPENHO 19
3.2.1 Função De Monitoramento De Desempenho 22

4 SIMPLE NETWORK MANAGEMENT PROTOCOL **24**

4.1 CONCEITOS BÁSICOS DE SNMP 25
4.1.1 Arquitetura 26
4.1.2 SMI – Structure Of Management Information 28
4.1.2.1 Estrutura MIB 29
4.1.2.1.1 Subgrupo IP 32
4.1.2.2 ASN.1 34
4.1.2.3 Codificação 35
4.1.3 Trap-directed Polling 35
4.1.4 Proxy 37
4.1.5 Comparativo Entre As Versões SNMP 38
4.1.6 Considerações Finais Sobre O SNMP 41

5 SNMPV3 ..... **42**

5.1 MOTOR SNMPV3 43
5.1.1 Despachante 44
5.1.2 Subsistema De Processamento De Mensagens 44
5.1.3 Subsistema De Segurança 45
5.1.4 Subsistema De Controle De Acesso 46

5.1.5 Gerente Snmp Tradicional 47
5.1.6 Agente SNMP Tradicional 49
5.2 APLICAÇÕES SNMP 50
5.3 PROCESSAMENTO DE MENSAGENS SNMPV3 51
5.3.1 Formato Das Mensagens SNMPv3 51
5.4 ALGORITMOS CRIPTOGRÁFICOS USADOS PELO SNMPV3 53
5.4.1 Conceitos Básicos De Criptografia 53
5.4.2 Data Encryption Standard (DES) 55
5.4.3 Message Digest 5 (MD5) 56
5.4.4 Secure Hash Function (SHA-1) 57
5.4.5 Autenticação De Mensagens Com O HMAC 57
5.5 USER-BASED SECURITY MODEL – USM 58
5.5.1 Parâmetros De Segurança USM 59
5.5.2 Funções Criptográficas Usadas Pelo USM 63
5.5.3 Processamento De Mensagens USM 64
5.5.4 Descoberta 70
5.6 GERENCIAMENTO DE CHAVES 71
5.6.1 Algoritmo De Transformação De Senha Para Chave 71
5.6.2 Localização De Chaves 72
5.6.3 Atualização De Chaves 74
5.7 VIEW-BASED ACCESS CONTROL MODEL - VACM 75
5.7.1 A MIB VACM 81
5.8 A API ADVENTNET SNMPV3 85

6 DESENVOLVIMENTO DO PROTÓTIPO 88

6.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 88
6.2 ESPECIFICAÇÃO 89
6.2.1 Diagramas de Casos de Uso 89
6.2.2 Diagrama de Classes 90
6.2.3 Diagramas de Seqüência 93
6.3 IMPLEMENTAÇÃO 98
6.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS 98
6.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO 107
6.4 RESULTADOS E DISCUSSÃO 112

7 CONCLUSÕES 113

7.1 EXTENSÕES 115

REFERÊNCIAS BIBLIOGRÁFICAS 116

LISTA DE FIGURAS

FIGURA 1 – Estrutura das camadas 10

FIGURA 2 – Camadas do modelo de referência OSI 11

FIGURA 3 – Elementos de um sistema de gerenciamento de redes 17

FIGURA 4 – Arquitetura de uma entidade *proxy* 18

FIGURA 5 – O papel do SNMP 27

FIGURA 6 - Grupos de objetos MIB-2 30

FIGURA 7 – Configuração de *proxy* 37

FIGURA 8 – Entidade SNMPv3 43

FIGURA 9 – Subsistema de processamento de mensagens 44

FIGURA 10 – Subsistema de segurança 45

FIGURA 11 – Subsistema de controle de acesso 46

FIGURA 12 – Gerente SNMPv3 tradicional 48

FIGURA 13 – Agente SNMPv3 convencional 49

FIGURA 14 – Formato da mensagem SNMPv3 51

FIGURA 15 – Criptografia convencional 54

FIGURA 16 – Formato da mensagem SNMPv3 com USM 61

FIGURA 17 – Processamento de mensagens USM: transmissão 65

FIGURA 18 - – Processamento de mensagens USM: recepção 67

FIGURA 19 – Localização de chaves 73

FIGURA 20 – Lógica VACM 78

FIGURA 21 – Fluxograma VACM 80

FIGURA 22 – A MIB VACM 83

Figura 23 - Diagrama de casos de uso 89

Figura 24 - Diagrama de Classes: parte agente 90
Figura 25 - Diagrama de Classes: parte gerente 91
Figura 26 - Obter informações de desempenho 93
Figura 27 - Inicialização do gerente 95
Figura 28 - Inicialização do agente 97
Figura 29 - Tela Inicial da aplicação gerente 108
Figura 30 - Medição inicial 111
Figura 31 - Medição final 112

LISTA DE ABREVIATURAS E LISTA DE SIGLAS

AGR - Aplicação de Gerenciamento de Rede

API - *Aplication Programing Interface*

ARPA - *Advanced Research Projects Agency*

ASN.1 - *Abstract Syntax Notation dot One*

BER - *Basic Encoding Rules*

CBC - *Cipher Block Chaining*

CMIP - *Common Management Information Protocol*

CMIS - *Common Management Information Service*

CMOT - *Common Management Information Protocol over TCP/IP*

CORBA - *Common Object Request Broker Architecture*

DES - *Data Encryption Standard*

DoD - *Department of Defense*

DQDB - *Distributed Queue Dual Bus*

ECB - *Electronic Codebook*

EGP - *Exterior Gateway Protocol*

EGR - Entidade de Gerenciamento de Rede

EJB - *Enterprise Java Beans*

FTP - *File Transfer Protocol*

HMAC - *Hashing for Message Authentication Code*

HMP - *Host Monitoring Protocol*

HTTP - *Hipertext Transfer Protocol*

IAB - *Internet Architecture Board*

ICMP - *Internet Control Message Protocol*

IEC - *International Electrotecnical Comission*

IEEE - *Institute of Electrical and Eletronics Engineers*

IETF - *Internet Engineering Task Force*

IP - *Internet Protocol*

ISO - *International Organization for Standardization*

JNI - *Java Native Interface*

JVM – *Java Virtual Machine*

LAN - *Local Area Network*

MAC - *Message Authentication Code*

MAN - *Metropolitan Area Network*

MD5 - M*essage-digest 5*

MIB - M*anagement Information Base*

MIT - *Massachussets Institute of Technology*

MS-DOS – Microsoft *Disk Operation System*

MTU - *Maximum Transfer Unit*

NCP - *Network-Control Protocol*

NIST - *National Institute of Standards and Tecnology*

NMS - *Network Management Station*

OSI - *Open Systems Interconection*

PC - *Personal Computer*

PDU - P*rotocol Data Unit*

PhD - *Doctor of Philosophy*

PING - *Packet Internet Groper*

PPP - *Point to Point Protocol*

QoS - *Quality of Service*

RFC - *Request for Comment*

RMI - *Remote Method Invocation*

SAS - SNMP *Applet Server*

SET - *Secure Electronic Transaction*

SGMP - *Simple Gateway Monitoring Protocol*

SHA-1 - *Secure Hash Algorithm 1*

SMI - S*tructure of Management Information*

SNA - *System Network Architecture*

SNMP - *Simple Network Management Protocol*

SNMPv3 - *Simple Network Management Protocol version three*

SSL - *Secure Socket Layer*

TCP - *Transmission Control Protocol*

TTL - *Time-To-Live*

UDP - *User Datagram Protocol*

UI - *User Interface*

UML - *Unifield Modeling Language*

XML - *Extensible Markup Language*

XNS - *Xerox Network Systems*

XOR - *Xtended OR*

WAN - *Wide Area Network*

WWW - *World Wide Web*

RESUMO

Este trabalho apresenta um estudo de usabilidade do protocolo de gerenciamento de redes *Simple Network Management Protocol version three* (SNMPv3), através da especificação e implementação de um protótipo de um sistema de gerenciamento de desempenho para dispositivos de uma *Local Area Network* (LAN), utilizando na sua implementação a *Aplication Programing Interface* (API) da empresa AdventNet escrita com a linguagem Java.

ABSTRACT

This paper presents a study of usability of the Simple Network Management Protocol version three (SNMPv3) through the specification and implementation of a prototype of a performance management system for Local Area Network (LAN) devices, using for it's implementation the AdventNet's Aplication Programing Interface (API) written with the Java language.

1 INTRODUÇÃO

Segundo Kurose (2001), devido ao fato de uma rede de computadores consistir de muitas partes complexas de hardware e software tais como *links*, equipamentos, pontes, roteadores e outros dispositivos, quando centenas ou milhares destes dispositivos são conectados uns aos outros para formar uma rede, é de se esperar que componentes irão eventualmente funcionar mal, que elementos de rede poderão ser desconfigurados, que recursos da rede serão superutilizados, ou que componentes de rede irão simplesmente "quebrar" (como por exemplo, o corte de um cabo). O administrador de redes deve estar apto a solucionar (e melhor ainda, evitar) tais problemas. O administrador de redes precisa claramente de ferramentas para ajudar a monitorar, analisar, gerenciar e controlar a rede.

Lynch (1993) cita que, até recentemente, o gerenciamento de redes se baseava nos avanços técnicos em outras áreas de redes baseadas em padrões abertos; de certa forma ainda é assim. Uma das razões para isso é que há uma discordância sobre o que realmente significa gerenciamento de redes. Como resultado, tem havido fundamentalmente diferentes abordagens ao problema de gerenciamento de redes. Algumas destas abordagens foram práticas, obtendo grande aceitação para solucionar parte do problema de gerenciar redes baseadas em protocolos de rede abertos. O protocolo *Simple Network Management Protocol* (SNMP) é a chave da estrutura de gerenciamento de redes de computadores. É um padrão aberto e operacional. A estrutura de gerenciamento SNMP foi originalmente desenhada para uso em redes *Transmission Control Protocol/Internet Protocol* (TCP/IP), mas vêm encontrando aplicação em áreas bem distantes daquelas para as quais foi inicialmente planejada. A estrutura SNMP constitui um padrão aberto para gerenciamento de redes, como estabelecido pelo *Internet Architecture Board* (IAB). Conseqüentemente, o SNMP pode ser dito como sendo um padrão declarado *de jure*. Padrões declarados que não estejam disponíveis são de pouca utilidade. Entretanto, este certamente não é o caso do SNMP. Vendedores o implementaram, consumidores o adquiriram, desenvolvedores de aplicações de redes o disponibilizaram, e administradores de redes o usam ativamente.

Conceitualmente, para Zeltserman (1999), o SNMPv3 nada mais é do que uma estrutura que estende o SNMP original. As duas maiores extensões são a adição de primitivas de segurança e de administração. Além disso, o SNMPv3 define novas *Management*

Information Base – MIBs, para configurar segurança, notificações, redirecionamento de *proxy* e controle de acesso baseado em visões.

Isto é uma “faca de dois gumes” pois pela primeira vez possui-se uma forma padrão para que um administrador de redes possa remotamente configurar estas características. Por outro lado, configurar segurança e controles de acesso baseados em visões adiciona complexidade.

De acordo com Carvalho (1993), a abrangência do gerenciamento de redes é muito grande, envolvendo principalmente as áreas de:

a) gerenciamento de falhas: responsável pela manutenção e monitoramento do estado de cada um dos objetos gerenciados e pelas ações necessárias ao restabelecimento das unidades com problemas;
b) gerenciamento de configuração: tem por função a manutenção e monitoração da estrutura física e lógica da rede, incluindo a existência de componentes e sua interconectividade;
c) gerenciamento de segurança: aborda os aspectos de segurança essenciais para operar uma rede corretamente e proteger os objetos gerenciados;
d) gerenciamento de contabilização: preocupa-se com a manutenção e monitoração de quais recursos e de quanto desses recursos estão sendo utilizados;
e) gerenciamento de desempenho: preocupa-se com o desempenho corrente da rede, incluindo parâmetros estatísticos tais como atrasos, vazão, disponibilidade e número de retransmissões. Consiste em um conjunto de funções responsáveis por manter e examinar registros, com histórico dos estados do sistema para fins de planejamento e análise.

O gerenciamento de desempenho é confundido, às vezes, com o gerenciamento de falhas. Muitos tendem a confundir desempenho com disponibilidade. O gerenciamento de desempenho é importante não só para garantir a qualidade de serviço acordada com os usuários, como também para assegurar que esta é atingida com os menores custos possíveis. Pode-se, por meio do gerenciamento de desempenho, adequar os recursos utilizados pelos usuários às suas necessidades, auxiliando o setor responsável pela administração de redes a antecipar-se aos usuários na manutenção dos níveis de desempenho dos serviços oferecidos,

como por exemplo, o tempo de resposta. O gerenciamento de desempenho está diretamente relacionado ao planejamento da capacidade do sistema sob gerenciamento.

O trabalho proposto apresenta como relevância em computação a implementação na linguagem Java de um protótipo de um sistema para a gerência de redes de computadores utilizando o protocolo SNMPv3, visando o monitoramento de desempenho de dispositivos de uma LAN.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho de conclusão de curso é a especificação e implementação de um protótipo de um sistema para a gerência de dispositivos de uma LAN utilizando algumas funções de gerência de desempenho através do protocolo SNMPv3.

Os objetivos específicos do trabalho são:

a) analisar as novas funcionalidades do protocolo SNMPv3;
b) especificar um protótipo de um sistema de gerência de redes para o gerenciamento de desempenho de alguns dispositivos de uma LAN;
c) validar estas funcionalidades através da implementação deste protótipo.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos, conforme apresentados a seguir.

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

O capítulo 2 apresenta um breve histórico das redes de computadores, sua estrutura básica, características e tecnologias.

O capítulo 3 trata especificamente do gerenciamento de redes: conceitos, características e modelos são apresentados, e o modelo de gerenciamento de desempenho de redes é estudado com mais detalhes.

O capítulo 4 aborda o protocolo de gerenciamento de redes SNMP. Suas características, estrutura e elementos são brevemente estudados.

O capítulo 5 aborda especificamente o protocolo SNMPv3: suas características, estrutura e seus elementos. Também apresenta a criptografia e seus conceitos básicos como autenticação, privacidade, gerenciamento de chaves, modelos e algoritmos utilizados no desenvolvimento do protótipo.

O capítulo 6 é voltado ao desenvolvimento do protótipo, onde são apresentadas a especificação e a implementação do protótipo.

E por fim, o capítulo 7 é dedicado às conclusões e sugestões de continuidade do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

A seguir será apresentada uma breve fundamentação teórica, com o objetivo de posicionar o trabalho no tema abordado.

2.1 BREVE HISTÓRICO DE REDES DE COMPUTADORES

Segundo Kurose (2001), devido ao crescimento da importância dos computadores no início dos anos 60, e do advento dos computadores com processamento paralelo, foi natural considerar a questão de como os computadores poderiam ser interligados, podendo assim, ser compartilhados entre usuários geograficamente distribuídos.

Mas o tráfego gerado por tais usuários causava “pesados” períodos de atividade, como o envio de um comando a um computador remoto, seguido de períodos de inatividade enquanto este aguardava o retorno do resultado ou enquanto se estudava a resposta recebida. Era preciso resolver este problema, e a solução surgiu em 1964 através de três grupos de pesquisa independentes: Leonard Kleinrock do *Massachussets Institute of Technology* (MIT), Paul Baran do *Rand Institute* e Donald Davis e Roger Scantlebury do *National Physical Laboratory* da Inglaterra, todos inconscientes uns dos outros, inventaram o conceito de troca de pacotes como uma alternativa eficiente e robusta para substituir a transmissão de dados por circuitos, usada até então. J.C.R Licklider e Lawrence Roberts, ambos colegas de Kleinrock, publicaram um plano geral para a chamada ARPAnet da *Advanced Research Projects Agency* (ARPA) em 1969, criando a primeira rede de computadores usando a troca de pacotes e ancestral direta da atual Internet. Em 1972 o primeiro protocolo *host-to-host* usado na ARPAnet conhecido como *Network-Control Protocol* (NCP) estava pronto. Com um protocolo fim-a-fim disponível, a partir daquele momento aplicações poderiam ser escritas.

A ARPAnet era uma rede simples e fechada. Entretanto, na metade dos anos 70, outras redes baseadas na troca de pacotes surgiram. Dentre elas podem-se citar a ALOHAnet, uma rede de microondas ligando as universidades da ilha do Hawai; a Telenet, uma rede comercial pertencente ao BBN, e a Tymnet e Transpac, ambas redes de troca de pacotes francesas. O número das redes de computadores começava a crescer. Em 1973, a tese de *Ph.D.* de Robert Metcalfe demonstrou os princípios da Ethernet, que mais tarde levaram a um enorme crescimento das chamadas LANs. Era chegado o momento de desenvolver uma arquitetura

para conectar as redes entre si. Trabalhos pioneiros sobre interconexão de redes foram feitos por Vinton Cerf e Robert Kahn, novamente sob patrocínio da ARPA. Estes princípios arquiteturais de interconexão foram incorporados ao protocolo TCP. Alias, os três protocolos chave usados hoje nas redes de computadores – TCP, *User Datagram Protocol* (UDP) e IP – estavam conceitualmente definidos já no final dos anos 70.

A tecnologia Ethernet representou um passo importante para a interconexão de redes. Cada LAN Ethernet era em si uma rede e, com a proliferação do número de LANs, a necessidade de interligá-las tornou-se cada vez mais importante. Muitas companhias desenvolveram suas próprias arquiteturas de redes proprietárias como a DEC com a DECnet (1975), a Xerox com a arquitetura *Xerox Network System* - XNS e a IBM com a arquitetura *System Network Architecture* - SNA.

Os anos 80 foram um período de tremendo crescimento. Muito do crescimento foi resultado do grande esforço de criar redes de computadores ligando as universidades entre si. Em 1983 a ARPAnet terminou o desenvolvimento do TCP/IP e o adotou em sua rede como o novo protocolo padrão em substituição ao NCP.

Nos anos 90 dois eventos simbolizaram a evolução e a comercialização das redes. Primeiro, a progenitora da Internet, a ARPAnet deixou de existir, tornando-se comercial dando origem a atual Internet. Mas o principal evento foi o surgimento da *World Wide Web* (WWW), que trouxe a Internet, e por conseqüência, as redes, às casas e negócios de milhões e milhões de pessoas ao redor do mundo. Além disso, ao longo dos anos 90, pesquisas e desenvolvimento em redes de computadores realizaram avanços nas áreas de roteadores de alta velocidade, roteamento, aplicações de tempo real e LANs, gerando uma enorme expansão no uso das redes de computadores nos dias atuais.

2.2 REDES DE COMPUTADORES

Em seguida será apresentada uma pequena discussão sobre as redes de computadores, suas tecnologias, utilização e arquiteturas, assim como *hardware* e softwares por elas utilizados.

2.2.1 USO DAS REDES DE COMPUTADORES

Segundo Tanenbaum (1997), devido ao rápido progresso tecnológico nas áreas de telefonia, rádio e tv, satélite e computadores, estas áreas estão convergindo rapidamente e há uma diferença cada vez menor entre coleta, transporte, armazenamento e processamento de informações. À medida que cresce a capacidade de colher, processar e distribuir informações, torna-se ainda maior a necessidade de formas de processamento de informações ainda mais sofisticadas. Esta “fusão” entre os computadores e as comunicações foi de grande importância para o surgimento das redes de computadores.

Uma rede de computadores é um conjunto de computadores autônomos interconectados. Dois ou mais computadores são ditos “interconectados” quando podem trocar informações (Tanenbaum, 1997).

Algumas razões econômicas e tecnológicas para a instalação de redes de computadores são:

a) compartilhamento de recursos: disponibilizar todos os programas, recursos e dados a todos os usuários da rede, independentemente da localização física dos recursos e dos usuários;
b) aumento de confiabilidade: viabilizar fontes alternativas de fornecimento de recursos (como exemplo, pode-se citar um arquivo salvo em três computadores diferentes e assim, se um deles falhar, o arquivo poderá ser obtido de um dos outros dois computadores);
c) economia de dinheiro: a relação custo/desempenho dos computadores de pequeno porte é muito melhor do que a dos computadores de grande porte;
d) escalabilidade: possibilitar o aumento gradual do desempenho do sistema à medida que cresce o volume de carga, bastando para tal, que se adicionem mais processadores;
e) meio de comunicação: uma rede de computadores representa um meio de comunicação altamente eficaz para funcionários que trabalham em locais muito distantes uns dos outros, aumentando assim, o espírito de equipe entre grandes grupos de pessoas.

Além destas razões de “eficiência corporativa” podem-se citar outros motivos para a interconexão de computadores:

a) acesso às informações remotamente (ex: transações financeiras e comércio eletrônico);
b) comunicação pessoa a pessoa (ex: correio eletrônico);
c) entretenimento (ex: jogos “*on-line*”).

2.2.2 HARDWARE DE REDE

Há duas dimensões gerais nas quais as redes de computadores podem ser classificadas segundo Tanenbaum (1997): a escala e a tecnologia de transmissão.

São dois, os tipos principais de tecnologia de transmissão:

a) redes de difusão (*broadcasting*): possuem apenas um canal de comunicação que é compartilhado por todas as máquinas;
b) redes ponto a ponto (*unicasting*): consistem em conexões entre pares individuais de máquinas.

Quanto à escala, as redes de computadores podem ser classificadas em:

a) redes locais: também chamadas de LANs, são redes geralmente privadas, contidas em um prédio ou em um *campus* universitário;
b) redes metropolitanas: também chamadas *Metropolitan Area Networks* (MANs), são uma versão ampliada de uma LAN, podendo abranger uma cidade inteira e que utilizam um protocolo de transmissão especial: o *Distributed Queue Dual Bus* (DQDB). Podem ser públicas ou privadas;
c) redes geograficamente distribuídas: também chamadas *Wide Area Networks* (WANs), abrangem uma ampla área geográfica, com freqüência um país ou continente. São compostas por equipamentos (máquinas que executam programas de usuários) e sub-redes (compostas por linhas de transmissão e elementos de comutação).

As redes locais possuem três características que as diferenciam das demais:

a) tamanho: é restrito, o que significa que o pior tempo de transmissão é limitado e conhecido. O conhecimento desse limite permite a utilização de determinados tipos

de projetos que em outras circunstâncias seriam inviáveis, além de simplificar o gerenciamento da rede, foco do presente trabalho;

b) tecnologia de transmissão: quase sempre consiste em um cabo ao qual todas as máquinas estão conectadas;
c) topologia: as LANs de difusão aceitam várias topologias, dentre as quais pode-se citar a de barramento e a de anel.

Para o desenvolvimento do presente trabalho será utilizada uma LAN de difusão conectada por cabos com uma topologia de barramento padrão LAN Ethernet, conforme definido pelo *Institute of Electrical and Eletronics Engineers* (IEEE) em IEEE 802.3.

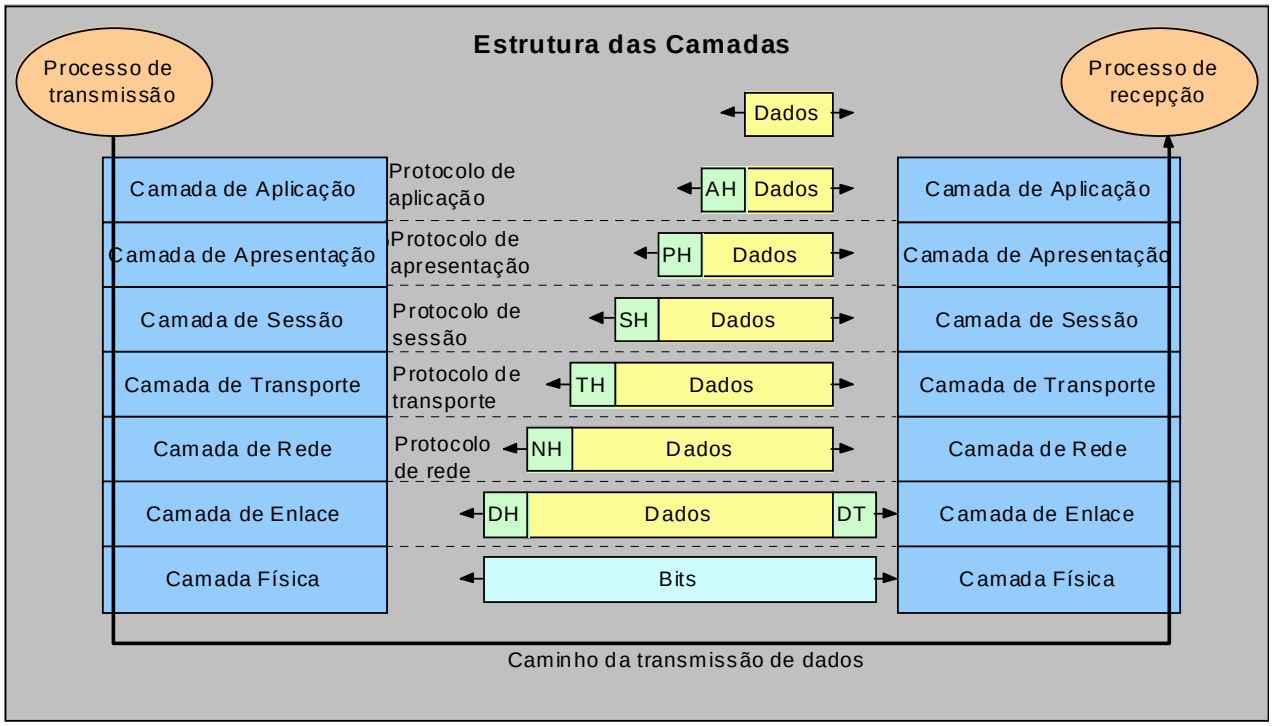
2.2.3 SOFTWARE DE REDE

Tanenbaum (1997) cita que para reduzir a complexidade do projeto, a maioria das redes foi organizada como uma série de camadas ou níveis que são colocados um em cima do outro. O objetivo de cada camada é oferecer determinados serviços para as camadas superiores, ocultando detalhes de implementação desses recursos. Um serviço é um conjunto de primitivas (operações) que uma camada oferece para a camada imediatamente acima dela. Os serviços podem ser de dois tipos: orientados à conexão e sem conexão. Já as operações podem ser: *Request*, *Indication*, *Response* e *Confirm*. Os serviços são classificados também pela sua qualidade de serviço (QoS - *Quality of Service*) que é a garantia de que uma mensagem chegue íntegra ao seu destino. Os elementos ativos em cada camada são freqüentemente chamados de entidades. As entidades de um mesmo nível/camada que se encontram em diferentes máquinas são chamadas de pares (*peers*) e a comunicação entre os pares é feita usando o protocolo da camada.

Um protocolo é basicamente um conjunto de regras que controla o formato e o significado dos quadros, pacotes, ou mensagens trocadas pelas entidades pares contidas em uma camada.

Entre cada par há uma interface que define as operações e os serviços que a camada inferior tem a oferecer a camada superior. Para compreender este modelo veja a figura 1.

FIGURA 1 – Estrutura das camadas



FONTE: Adaptado de Tanenbaum (1997, pág. 39).

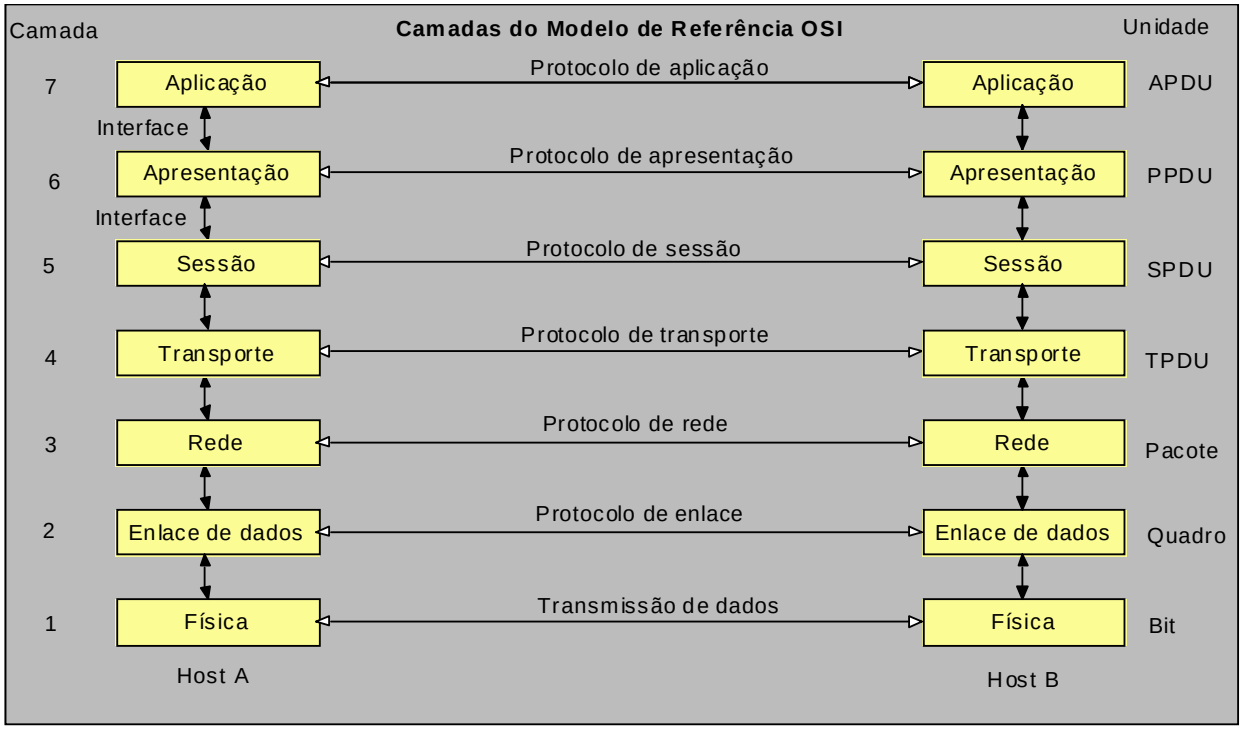
Como se vê, na verdade os dados não são diretamente transmitidos da camada *n* de uma máquina para a camada *n* da outra. Cada camada, na verdade, transfere os dados e as informações de controle para a camada imediatamente abaixo dela, até a última camada ser alcançada. Abaixo desta camada está o meio físico através do qual se dá a comunicação propriamente dita. Na outra ponta o processo inverso é realizado.

Um conjunto de camadas de protocolos é chamado de arquitetura de rede, e uma lista de protocolos usados por um determinado sistema, um protocolo por camada, é chamado de pilha de protocolos.

Dentre as arquiteturas de rede podem-se citar o modelo de referência *Open Systems Interconnection* (OSI) e o modelo de referência Internet.

O modelo de referência OSI foi especificado pela *International Organization for Standardization/International Electrotechnical Comission* (ISO/IEC) e possui sete camadas (fig. 2):

FIGURA 2 – Camadas do modelo de referência OSI



FONTE: Adaptado de Tanenbaum (1997, pág. 33).

Cada camada é responsável por uma função específica, conforme descrito a seguir:

a) física: trata da transmissão de bits brutos através de um canal de comunicação;
b) enlace: transforma um canal de transmissão de dados brutos em uma linha que pareça livre dos erros de transmissão não detectados na camada de rede;
c) rede: especifica o modo como os pacotes são roteados da origem para o destino;
d) transporte: sua principal função é realizar a divisão dos dados em pacotes de tamanhos compatíveis com a camada de rede a ser utilizada e, reagrupá-los sem erros na outra extremidade. Esta é a verdadeira camada fim-a-fim que liga a origem ao destino através da resolução de seus endereços e nomes. É também a camada responsável pelo estabelecimento das conexões e pelo controle de fluxos;
e) sessão: permite que usuários de diferentes máquinas estabeleçam sessões entre si. Um dos serviços desta camada é gerenciar o controle de tráfego;
f) apresentação: preocupa-se com a sintaxe e a semântica das informações transmitidas, como por exemplo, a codificação dos dados de acordo com o padrão estabelecido;
g) aplicação: possui aplicações específicas para o protocolo, tais como transferência

de arquivos, gerência de rede, etc.

Observa-se que o modelo OSI em si não é uma arquitetura de rede, pois não especifica os serviços e os protocolos que devem ser usados em cada camada. Ele apenas informa o que cada camada deve fazer.

O modelo OSI possui três conceitos fundamentais:

a) serviços;
b) interfaces;
c) protocolos.

Esta separação entre estes três conceitos talvez seja a maior contribuição do modelo OSI, pois torna explícita a distinção entre estes três conceitos. O serviço informa o que a camada faz. A interface de uma camada informa como os processos acima dela podem acessá-la, e finalmente, que os protocolos utilizados em uma camada são de responsabilidade apenas dessa camada e estes especificam a forma com que a camada implementa seus serviços através das interfaces.

O modelo de referência Internet segundo Kurose (2001) possui cinco camadas:

a) aplicação: responsável pelas aplicações de rede. Inclui vários protocolos como *Hipertext Transfer Protocol* (HTTP), *File Transfer Protocol* (FTP), SNMP, etc;
b) transporte: fornece os serviços para o transporte das mensagens da camada de aplicação entre os lados cliente e servidor da aplicação. Possui dois protocolos: TCP (orientado à conexão) e o UDP (não orientado à conexão).
c) rede: responsável pelo roteamento de datagramas de um equipamento para outro. Possui o IP, que define os campos do datagrama IP e protocolos de roteamento, que ditam a rota a ser tomada pelos datagramas IP entre a origem e o destino;
d) enlace: a camada de rede roteia um pacote entre uma série de transmissores de pacotes (chamados roteadores) entre a origem e o destino. Como exemplos de camadas de enlace podem-se citar o padrão Ethernet e o *Point to Point Protocol* (PPP);
e) física: enquanto o trabalho da camada de enlace é mover *frames* inteiros de um nó para o outro, o trabalho da camada física é o de mover bits individuais de cada *frame* de um nó para o outro.

3 GERÊNCIA DE REDES DE COMPUTADORES

Segundo Stallings (2001), as redes de computadores e os sistemas de processamento distribuído são de crescente importância e, de fato, se tornaram essenciais no mundo dos negócios. Hoje em dia, uma organização típica possui uma arquitetura de redes grande e crescente, mas amórfica, com várias redes locais (LANs) e redes de larga escala (WANs), suportadas por pontes (*bridges*) e roteadores (*routers*), e uma grande variedade de serviços e dispositivos computacionais distribuídos, incluindo *Personal Computers* - PCs, estações de trabalho, e servidores (inclusive *mainframes*). Com esse crescimento em escala das redes de computadores, dois fatos se tornam dolorosamente evidentes:

a) a rede e seus recursos a ela associados, se tornaram indispensáveis para a organização;
b) mais coisas podem “dar errado”, interrompendo o funcionamento da rede ou de parte dela, degradando o desempenho para um nível inaceitável.

Já para Rose (1994), há dois motivos pelos quais a gerência de redes é necessária:

a) dispositivos heterogêneos: pelo fato da interconexão de redes permitir diferentes tipos de dispositivos participarem da rede, estes componentes (*hosts*, *routers*, *switches*, etc) irão quase sempre ser de fabricantes diferentes, e heterogêneos por natureza. Atualmente, há no mercado dispositivos de centenas de fabricantes, cada um com vários modelos de cada linha de produto, que implementam o protocolo TCP/IP. Evidencia-se que uma tecnologia de gerenciamento específica de um vendedor é inutilizável em tais ambientes. Assim, como apenas a utilização de uma tecnologia de interconexão "aberta", ou seja, não proprietária como o TCP/IP é o que torna um ambiente com múltiplos fabricantes de dispositivos uma realidade, também uma tecnologia de gerenciamento de redes "aberta" (SNMP), é o que torna possível gerenciar estes diferentes dispositivos;
b) administrações diferentes: pelo fato de interconexão de redes permitir várias redes com tamanhos e propósitos diferentes se interconectarem, estas redes irão quase sempre estar sob administrações diferentes, como é o caso atual da Internet.

Para gerenciar estes sistemas e redes, que continuam a crescer em escala e diversidade, um conjunto rico e automatizado de ferramentas de gerenciamento de redes se faz necessário.

É fundamental, para o uso destas ferramentas e aplicações em um ambiente com

múltiplos fabricantes, a utilização de técnicas padronizadas para representar e trocar informações relacionadas ao gerenciamento de redes (Stallings, 2001).

Para isto, é preciso definir o que significa o termo “gerenciamento de redes”. Abaixo duas definições são apresentadas:

“O gerenciamento de redes/sistemas é a soma total de todos os procedimentos e produtos para planejamento, configuração, controle, monitoramento e gerenciamento de redes de computadores e sistemas distribuídos, e a remoção de erros destes sistemas. Estes recursos devem prover suporte econômico e amigável aos usuários que trabalham com a rede e seus componentes. Logo, cobre todas as precauções e atividades necessárias para assegurar o uso efetivo e eficiente dos recursos gerenciados” (Hegering, 1994).

“A gerência de redes inclui o desenvolvimento, integração, e coordenação de hardware, software e elementos humanos para monitorar, testar, juntar (*poll*), configurar, analisar, avaliar, e controlar a rede e seus recursos para alcançar em tempo real, o desempenho operacional e a qualidade de serviço requisitados a um custo razoável.” (SAYDAM[1], apud KUROSE, 2001, p. 630-631).

Hegering (1994), ainda divide o gerenciamento de redes em três dimensões:

a) a dimensão funcional: esta dimensão se preocupa com a designação de tarefas de gerenciamento em áreas funcionais. O modelo ISO de gerenciamento provê uma subdivisão com as seguintes áreas: configuração, falhas, desempenho, contabilização e segurança;
b) a dimensão temporal: divide os processos que implementam as funções de gerenciamento em fases com diferentes ciclos de vida, incluindo as fases de planejamento, implementação e operação;
c) a dimensão de cenário: recentemente, além do gerenciamento clássico de redes, um número de outros “cenários de gerenciamento” tais como gerenciamento de sistemas, gerenciamento de aplicações e gerenciamento empresarial, vem surgindo. Estes cenários são diferenciados pelo fato de que os objetos alvo de gerenciamento são diferentes em cada caso, o que geralmente leva a diferentes aplicações de gerenciamento.

Kurose (2001) cita que a ISO/IEC criou um modelo de gerenciamento de redes que define cinco áreas funcionais de gerência:

a) gerência de desempenho: o objetivo desta área é quantificar, medir, relatar, analisar e controlar o desempenho de diferentes componentes da rede. Estes componentes podem ser dispositivos individuais, como roteadores e equipamentos, ou abstrações, tais como um caminho ao longo da rede;
b) gerência de falhas: registrar, detectar e responder às condições de falhas na rede são os objetivos desta área. A linha entre gerência de falhas e gerenciamento de desempenho é imprecisa. Pode-se pensar na gerência de falhas como o tratamento imediato de falhas transientes da rede, enquanto que o gerenciamento de desempenho busca ter uma visão aprofundada da rede para poder fornecer níveis aceitáveis de desempenho tendo em vista as várias demandas de tráfego e ocasionais falhas de dispositivos de rede. A gerência de falhas é geralmente corretiva enquanto que a gerência de desempenho é quase sempre preventiva;
c) gerência de configuração: permite ao administrador da rede rastrear quais dispositivos fazem parte da rede gerenciada e realizar a configuração de hardware e software destes dispositivos;
d) gerência de contabilização: esta área permite ao administrador da rede especificar, registrar e controlar o acesso dos usuários aos dispositivos da rede. Cotas de uso, cobranças baseadas em uso de recursos e a alocação de privilégios de acesso aos recursos, fazem parte da gerência de contabilização;
e) gerenciamento de segurança: o objetivo da gerência de segurança é controlar o acesso aos recursos da rede de acordo com políticas bem definidas. Os centros de distribuição de chaves e autoridades certificadoras são componentes da gerência de segurança. O uso de *firewalls* para monitorar e controlar pontos de rede de acesso externo é outro componente fundamental.

Em resposta a estas necessidades de gerenciamento, administradores de rede e usuários se voltaram para um padrão: o SNMP. O SNMP foi especificado no final dos anos 80 e rapidamente se tornou o padrão para o gerenciamento de redes em plataformas com múltiplos fabricantes. O SNMP se refere na verdade, a um conjunto de padrões para o gerenciamento de redes, incluindo um protocolo, uma especificação de uma estrutura de base de dados e um conjunto de objetos de dados. Entretanto, o SNMP era muito limitado para

atender a todas as necessidades críticas de gerenciamento de redes. Três melhorias (1993 – SNMPv2, 1995 – revisão do SNMPv2, 1998 – SNMPv3) solidificaram o papel do SNMP como uma ferramenta indispensável para o gerenciamento de redes (Stallings, 2001).

O SNMP será abordado com mais detalhes no capítulo 4.

3.1 ARQUITETURA DE UM SISTEMA DE GERENCIAMENTO DE REDES

Um sistema de gerenciamento de redes, segundo Stallings (2001), é uma coleção de ferramentas para o monitoramento e controle da rede que são integradas da seguinte forma:

a) é uma única e poderosa interface operacional, porém amigável e com um conjunto de comandos para realizar quase todas, senão todas as tarefas de gerenciamento de redes;
b) uma quantidade mínima de equipamentos é necessária.

Um sistema de gerenciamento de redes consiste de adições incrementais de hardware e software implementados entre os componentes já existentes na rede. Um sistema de gerenciamento de redes é designado para visualizar a rede como uma arquitetura unificada.

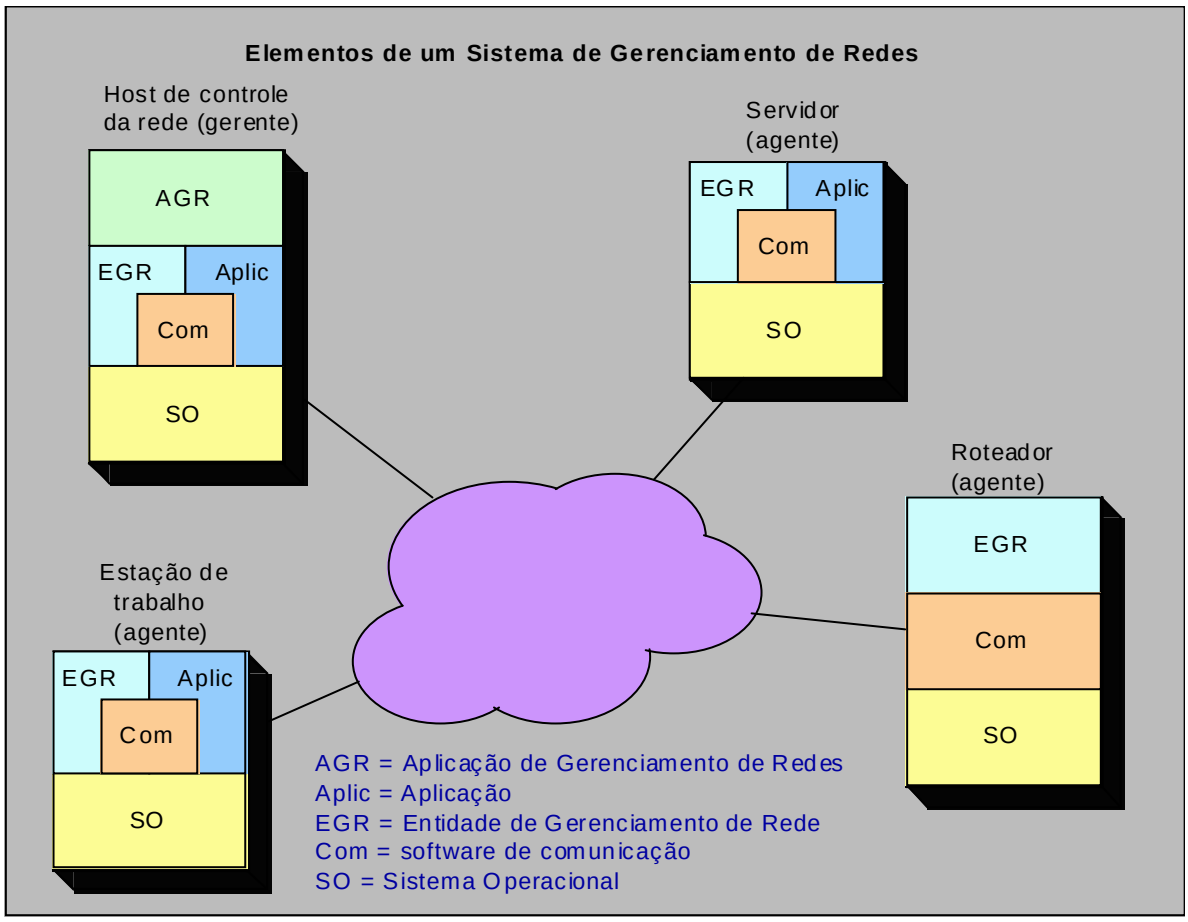
Cada nó da rede contém uma coletânea de softwares responsável pela tarefa de gerenciamento, e são referenciados como uma entidade de gerenciamento de rede (EGR ou *Network Management Station* - NMS), conforme mostrado na figura 3.

Cada EGR pode realizar as seguintes tarefas:

a) coletar estatísticas sobre atividades relacionadas à rede e às comunicações;
b) armazenar estatísticas localmente;
c) responder aos comandos do centro de controle da rede incluindo comandos para:
   - transmitir as estatísticas coletadas para o centro de controle;
   - alterar um parâmetro;
   - fornecer informações sobre o estado da rede;
   - gerar tráfego artificial para realizar um teste;
d) enviar mensagens ao centro de controle quando condições locais sofrerem mudanças significativas.

Pelo menos um equipamento na rede é designado como o equipamento de controle da rede, ou gerente. Além dos elementos de uma EGR, o equipamento de controle inclui uma coleção de softwares chamados de aplicação de gerenciamento de rede (AGR). A AGR inclui uma interface operacional para permitir que um usuário autorizado gerencie a rede. A AGR responde aos comandos do usuário através da exibição de informações e/ou atribuindo comandos às EGRs da rede. Esta comunicação é realizada usando um protocolo de nível de aplicação de gerenciamento de rede, que emprega a arquitetura de comunicação da mesma forma que qualquer outra aplicação distribuída.

FIGURA 3 – Elementos de um sistema de gerenciamento de redes



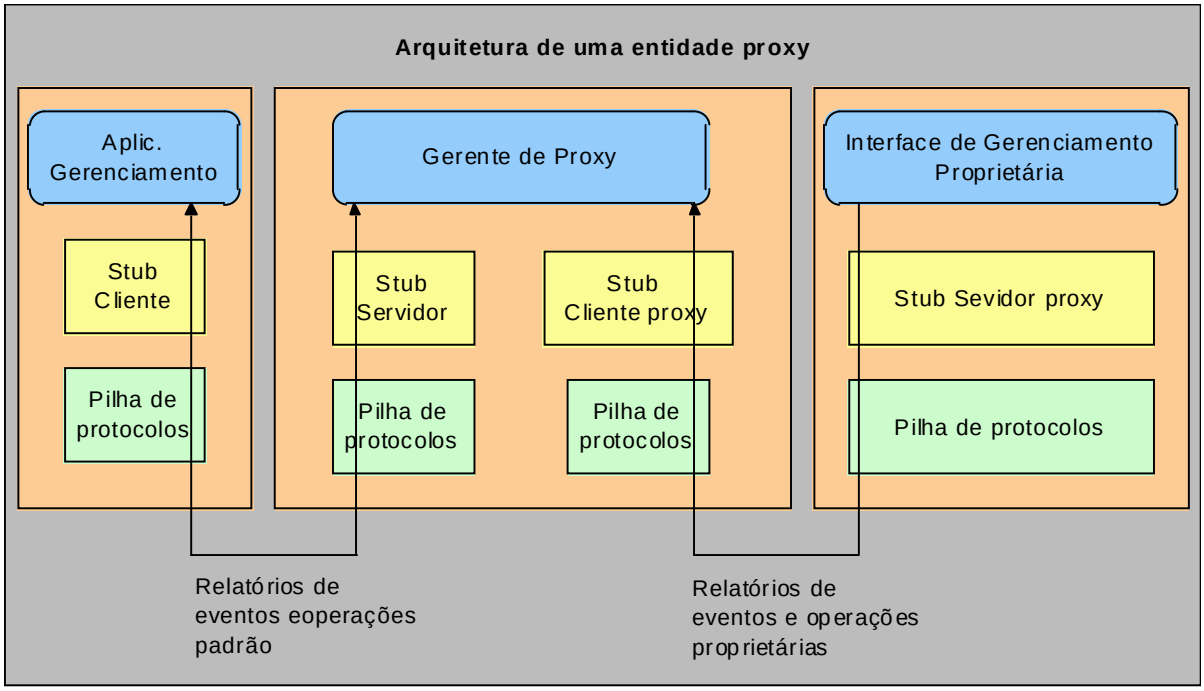
FONTE: Adaptado de Stallings (2001, pág. 8).

Outros nós na rede que fazem parte do sistema de gerenciamento de redes incluem uma EGR que responde aos pedidos de um gerente do sistema. A EGR em tais sistemas é geralmente um módulo agente, ou simplesmente, agente. Os agentes são implementados em

sistemas que suportam aplicações de usuários finais bem como em nós que provêem serviços de comunicação, tais como pontes e roteadores.

A configuração da figura 3 dá a entender que cada componente da configuração que é de interesse de gerenciamento, inclui uma entidade de gerenciamento de rede, com um software de gerenciamento de rede comum que atua sobre todos os agentes e gerentes. Na configuração atual, isto pode não ser prático, ou até mesmo impossível. Para lidar com tais casos, é comum ter um dos agentes no sistema atuando como um *proxy*, para um ou mais nós da rede (fig. 4).

FIGURA 4 – Arquitetura de uma entidade *proxy*



FONTE: Adaptado de Stallings (2001, pág. 15).

Quando um agente atua como um *proxy*, ele age em benefício de um ou mais nós da rede. Um gerente que desejar obter informações do nó ou controlá-lo, deve se comunicar com o agente *proxy*. O agente *proxy* irá traduzir a requisição do gerente usando qualquer protocolo de gerenciamento disponível, de modo que o sistema alvo receba e entenda a requisição feita. A resposta recebida do sistema alvo é similarmente traduzida e repassada ao gerente.

3.2 GERENCIAMENTO DE DESEMPENHO

Segundo Stallings (2001), as redes de comunicação de dados modernas são compostas de muitos componentes, que devem se intercomunicar e compartilhar dados e recursos. Em alguns casos, é crítico, para a eficácia de uma aplicação, que a comunicação pela rede tenha certos limites de desempenho.

O gerenciamento de desempenho de uma rede de computadores engloba duas grandes categorias funcionais: monitoramento e controle. Monitoramento é a função que rastreia atividades na rede. A função de controle habilita a gerência de desempenho a fazer ajustes para aumentar o desempenho da rede. Algumas questões de desempenho, no que diz respeito ao administrador da rede são:

a) qual é a utilização do nível de capacidade?
b) há tráfego excessivo?
c) o *throughput* foi reduzido a níveis inaceitáveis?
d) existem “gargalos” na rede?
e) o tempo de resposta está aumentando?

Para lidar com estes problemas, o administrador de rede deve focar-se em algum conjunto inicial de recursos a serem monitorados para medir os níveis de desempenho. Isto inclui associar métricas apropriadas e valores com recursos relevantes da rede, como indicadores de diferentes níveis de desempenho.

Hegering (1994) vê o gerenciamento de desempenho como uma continuação consistente do gerenciamento de falhas. Enquanto que o gerenciamento de falhas é responsável por assegurar que a rede de comunicação permaneça funcionando, isto não é suficiente para o gerenciamento de desempenho, que se posiciona com o objetivo de assegurar que o sistema como um todo esteja oferecendo uma boa qualidade de serviço.

Algumas subtarefas da gerência de desempenho incluem:

a) determinar parâmetros de qualidade de serviço;
b) monitorar a rede de comunicação em busca de “gargalos”;
c) executar medidas de desempenho;
d) processar os dados medidos e gerar relatórios;
e) planejar a capacidade e o desempenho da rede.

Para Stallings (2001), o gerenciamento de desempenho deve monitorar muitos recursos para fornecer informações para determinar os níveis de operação da rede. Pela coleta destas informações, sua análise e o uso dos resultados obtidos como retorno (*feedback*) ao conjunto de valores prescritos, o administrador da rede pode se tornar mais e mais apto a reconhecer situações indicativas de degradações de desempenho presentes ou iminentes.

Antes de usar uma rede para executar uma aplicação em particular, um usuário final pode querer saber coisas como o pior tempo de resposta e a confiabilidade dos serviços da rede. Por isso o desempenho deve ser conhecido em detalhes suficientes para responder aos questionamentos específicos dos usuários finais. Os usuários finais esperam que os serviços sejam gerenciados de forma a garantir bons tempos de resposta para obter assim, uma boa utilização de suas aplicações.

Os gerentes de rede precisam de estatísticas de desempenho para ajudá-los a planejar, gerenciar e manter grandes redes funcionando. As estatísticas de desempenho podem ser usadas para identificar potenciais “gargalos” antes que causem problemas aos usuários finais. Ações corretivas apropriadas podem então ser tomadas. Estas ações podem tomar a forma de alterações nas tabelas de roteamento para balancear ou redistribuir a carga de tráfego durante períodos de pico, ou quando um “gargalo” é identificado por um crescimento rápido de carga em uma área específica. De modo geral, um planejamento de capacidade baseado em tais informações de desempenho pode indicar as decisões apropriadas a serem tomadas, como por exemplo, a expansão de linhas naquela área.

Por tudo isso, um pré-requisito absoluto para o gerenciamento de uma rede de computadores é a habilidade de medir o desempenho da rede, ou monitoramento de desempenho. Não se pode gerenciar e controlar um sistema ou atividade se não for possível monitorar o seu desempenho. Uma das dificuldades que o administrador de redes encontra é na seleção e uso dos indicadores apropriados para medir o desempenho da rede. Dentre os problemas pode-se citar os seguintes:

a) existem muitos indicadores para usar;
b) o significado de muitos destes indicadores não são claramente compreendidos;
c) alguns indicadores são suportados apenas por alguns fabricantes;
d) muitos indicadores não servem para fazer comparações entre si;

e) freqüentemente, os indicadores são medidos corretamente, mas interpretados incorretamente;
f) em muitos casos, o cálculo dos indicadores consome muito tempo, e o resultado final já não pode mais ser usado para controlar o ambiente.

Há dois tipos de indicadores: medidas orientadas a serviços e medidas orientadas à eficiência. As medidas orientadas a serviços possuem os seguintes indicadores:

a) disponibilidade: o percentual de tempo que uma rede, um componente, ou uma aplicação está disponível para o usuário;
b) tempo de resposta: quanto tempo é preciso para uma resposta aparecer no terminal do usuário após uma solicitação feita pelo mesmo;
c) precisão: o percentual de tempo em que não ocorrem erros na transmissão e recepção de informações.

Já as medidas orientadas a eficiência possuem indicadores de:

a) *throughput*: a taxa na qual as aplicações orientadas a eventos ocorrem;
b) utilização: o percentual da capacidade teórica de um recurso que está sendo utilizado.

Kurose (2001) cita que a área de monitoramento de desempenho da rede no gerenciamento de redes é responsável por observar e analisar o estado e o comportamento dos sistemas fim (*end systems*), dos sistemas intermediários e sub-redes que compõe os recursos gerenciados.

De acordo com Stallings (2001), a arquitetura funcional de monitoramento de redes é composta de:

a) aplicação de monitoração: este componente inclui as funções de monitoramento de rede que são visíveis ao usuário, tais como monitoramento de desempenho, monitoramento de falhas e monitoramento de configuração;
b) função gerente: este é o módulo no monitor da rede que realiza a função de monitoramento básica, buscando informações de outros elementos da configuração;
c) função agente: este módulo obtém e registra informações de gerenciamento para um ou mais elementos de rede e repassa essas informações ao monitor;

d) objetos gerenciados: estas são as informações gerenciadas, que representam os recursos da rede e suas atividades;

É importante mencionar um módulo funcional adicional relacionado com informações estatísticas:

e) agente de monitoração: este módulo adicional gera sumários e análises estatísticas das informações gerenciadas. Se afastado do gerente, este módulo age como um agente e comunica estas informações ao gerente.

O monitoramento de desempenho engloba todas as cinco áreas funcionais. As informações disponíveis para a monitoração da rede podem ser classificadas como segue:

a) estática: esta é a informação que caracteriza a configuração corrente e os elementos na configuração corrente, tais como os números e identificações das portas de um roteador;
b) dinâmica: estas informações são relativas a eventos na rede, tais como: mudanças de estado de uma máquina ou a transmissão de um pacote pela rede;
c) estatística: estas são informações que podem ser derivadas das informações dinâmicas, como por exemplo, o número médio de pacotes transmitidos por unidade de tempo por um sistema fim.

3.2.1 FUNÇÃO DE MONITORAMENTO DE DESEMPENHO

Segundo Stallings (2001), a função de monitoramento de desempenho engloba três componentes:

a) medida de desempenho: é a reunião de estatísticas sobre o tráfego da rede e temporização;
b) análise de desempenho: consiste de software para reduzir e apresentar os dados;
c) geração de tráfego sintético: consiste de software que permite observar a rede sob uma carga controlada.

A medida de desempenho é quase sempre completada por módulos agentes junto aos dispositivos da rede (equipamentos, roteadores, pontes, etc). Estes agentes estão em posição de observar a quantidade de tráfego que entra e sai de um nó, o número de conexões e o tráfego por conexão, além de outras medidas que fornecem um cenário detalhado do

comportamento daquele nó. Em uma rede compartilhada, tal como uma LAN, muitas das informações necessárias podem ser coletadas por um monitor externo ou remoto que simplesmente observa o tráfego na rede.

Alguns tipos de medidas que podem ser obtidas em uma LAN típica são as seguintes:

a) histograma de tipos de pacotes circulando na rede;
b) histograma com os tamanhos dos pacotes;
c) histograma com os tamanhos dos pacotes de dados;
d) distribuição ou utilização de *throughput*;
e) histograma do tempo de chegada de pacotes;
f) histograma com a demora de aquisição de canal;
g) histograma com a demora de comunicação;
h) histograma com a contagem de colisões;
i) histograma com o contador de transmissões;

Com estas medidas é possível responder às várias perguntas feitas por um administrador de redes relacionadas ao desempenho de uma LAN tais como:

a) o tráfego está igualmente distribuído entre os usuários da rede ou há pares origem/destino com tráfego pesado?
b) qual é o percentual de cada tipo de pacote? Existem alguns tipos de pacotes com alta freqüência de uso, indicando um erro ou um protocolo ineficiente?
c) qual é a distribuição dos tamanhos dos pacotes de dados?
d) quais são as demoras na aquisição de canal e as distribuições de comunicação? Esses tempos são excessivos?
e) qual é a utilização do canal e seu *throughput*?

Quando a rede possui uma carga de tráfego muito pesada, pode não ser prático coletar os dados exaustivamente. A alternativa é tratar cada parâmetro como uma variável aleatória realizando uma amostragem do fluxo de modo a estimar o valor desta variável. Entretanto, deve-se tomar cuidado ao se usar e interpretar as estimativas de resultados estatísticos. O indivíduo responsável por desenvolver funções de amostragem e por interpretar os resultados precisa ter alguma familiaridade com os princípios estatísticos.

4 SIMPLE NETWORK MANAGEMENT PROTOCOL

Nos anos 60, segundo Stallings (2001), o assunto gerência de redes não existia. Nos anos 70 e metade dos anos 80, apenas o *Internet Control Message Protocol* (ICMP) e o *Packet Internet Groper* (PING) eram utilizados para tal tarefa. Com o crescimento da Internet no fim dos anos 80, surgiu a necessidade de se criar ferramentas para o gerenciamento das redes existentes. Primeiro foi o *Simple Gateway Monitoring Protocol* (SGMP) em 1987. A partir dele surgiram o *Host Monitoring Protocol* (HMP), o primeiro protocolo de gerenciamento usado na Internet, o SNMP (uma versão melhorada do SGMP), e o *Common Management Information Protocol over TCP/IP* (CMOT).

A IAB aprovou em 1988 o SNMP como uma solução temporária e o CMOT como o padrão a ser adotado mais tarde. Para tal a IAB determinou que ambos os protocolos usassem a mesma base de dados de objetos gerenciados. Assim ambos usariam uma única estrutura de gerenciamento de informações (*Structure of Management Information* – SMI), e uma única base de gerenciamento de informações (MIB). O objetivo era tornar mais fácil a futura transição do SNMP para o CMOT. Mas logo se tornou claro que tal ligação seria impossível.

No gerenciamento OSI, os objetos gerenciados são vistos como entidades sofisticadas, com atributos, procedimentos associados, e capacidades de emissão de mensagens, além de outras características complexas associadas com a tecnologia orientada a objetos. Para manter o SNMP simples, ele não foi designado para trabalhar com estes conceitos sofisticados. De fato, os objetos no SNMP nada mais são do que variáveis com algumas características básicas, como tipos de dados e atributos de somente-leitura (*read-only*) ou leitura-escrita (*read-write*). Por isso a IAB relaxou seus termos e com isso o SNMP e o CMOT evoluíram independentemente e em paralelo. Os desenvolvedores do SNMP, livres das restrições de compatibilidade com o modelo OSI desenvolveram rapidamente o SNMP e logo ele estava sendo usado em larga escala pelos vendedores de equipamentos e seu uso espalhou-se pela Internet. Enquanto isso, os esforços no CMOT se esvaeceram. Atualmente, virtualmente todos os grandes vendedores de equipamentos, estações de trabalho, roteadores, pontes e *hubs*, oferecem suporte ao SNMP.

As três especificações fundamentais do SNMP são:

a) *Request for Comment* - RFC 1155: descreve como os objetos gerenciados contidos nas MIBs são definidos (SMI);
b) RFC 1213: descreve os objetos gerenciados contidos na MIB (MIB-II);
c) RFC 1157: define o protocolo usado para gerenciar os objetos (SNMP).

Todas as RFCs citadas no presente trabalho podem ser encontradas em RFC/STD/FYI/BCP Archives (199-).

Para Lynch (1993), o gerenciamento de redes moderno possui muitos requisitos importantes. Primeiro, a estrutura de gerenciamento deve suportar tanto monitoramento quanto controle. Segundo, deve possuir a habilidade de gerenciar todas as camadas da implementação do modelo OSI fornecendo um gerenciamento topo-base (*top-down*). Terceiro, pela máxima extensão possível e economicamente viável, o escopo do gerenciamento da rede deve ser fim-a-fim. Deve englobar todos os sistemas da rede. Finalmente, o impacto dessa estrutura de gerenciamento de redes não deve ser visível, ou perceptível. A largura de banda consumida pelas funções de gerenciamento deve estar dentro de certos limites. Então, por que não usar o *Common Management Information Protocol* - CMIP?

Para Lynch (1993), primeiro porque a sobrecarga de implementar o *Common Management Information Service* - CMIS e CMIP em sistemas com recursos limitados pode ser muito caro. Segundo, estes protocolos ocupam uma pequena fatia do mercado atualmente, logo, os vendedores escolhem implementar o SNMP ao invés do CMIP e, como resultado, o mercado ocupado pelo CMIP permanece pequeno. Finalmente, enquanto que o CMIS e CMIP são padrões internacionais bem definidos, muito da sua infraestrutura de suporte, tal como os equivalentes ISO do SMI, MIBs e perfis de transporte, não alcançaram ainda este mesmo nível de padronização e estabilidade.

4.1 CONCEITOS BÁSICOS DE SNMP

A seguir serão apresentados os conceitos básicos relacionados ao SNMP, tais como sua estrutura, seus elementos e a relação entre estes elementos.

4.1.1 ARQUITETURA

O modelo de gerência de redes TCP/IP, de acordo com Stallings (2001), é composto pelos seguintes elementos:

a) estação de gerenciamento (gerente);
b) agentes;
c) base de informações de gerenciamento;
d) protocolo de gerenciamento de redes.

A estação de gerenciamento serve como uma interface pela qual o administrador da rede se comunica com o sistema de gerenciamento. No mínimo, uma estação de gerenciamento terá:

a) um conjunto de aplicações para análise de dados, recuperação de falhas, etc;
d) uma interface pela qual o administrador da rede pode monitorar e controlar a rede;
e) a capacidade de traduzir os requisitos do administrador da rede no atual controle e monitoramento dos elementos remotos da rede;
f) uma base de dados com informações extraídas das MIBs de todas as entidades gerenciadas da rede.

Apenas os dois últimos elementos estão sujeitos à padronização SNMP.

Outro elemento ativo no gerenciamento da rede é o agente. O agente responde aos pedidos de informações e às ações vindas do gerente e pode prover de forma assíncrona, informações não solicitadas, porém importantes, para a estação de gerenciamento.

Os recursos da rede podem ser gerenciados pela representação dos mesmos como objetos. Cada objeto é, essencialmente, uma variável de dados que representa um aspecto do agente gerenciado. Esta coleção de objetos é denominada MIB. Estes objetos são padronizados através de sistemas de uma classe em particular. Uma estação de gerenciamento realiza a função de monitoramento obtendo os valores dos objetos da MIB.

A estação de gerenciamento e os agentes são ligados por um protocolo de gerenciamento de rede. O protocolo usado para o gerenciamento de redes TCP/IP é o SNMP, que possui as seguintes capacidades chave:

a) *get*: habilita a estação de gerenciamento a obter os valores dos objetos no agente;

b) *set*: habilita a estação de gerenciamento a configurar os valores dos objetos no agente;

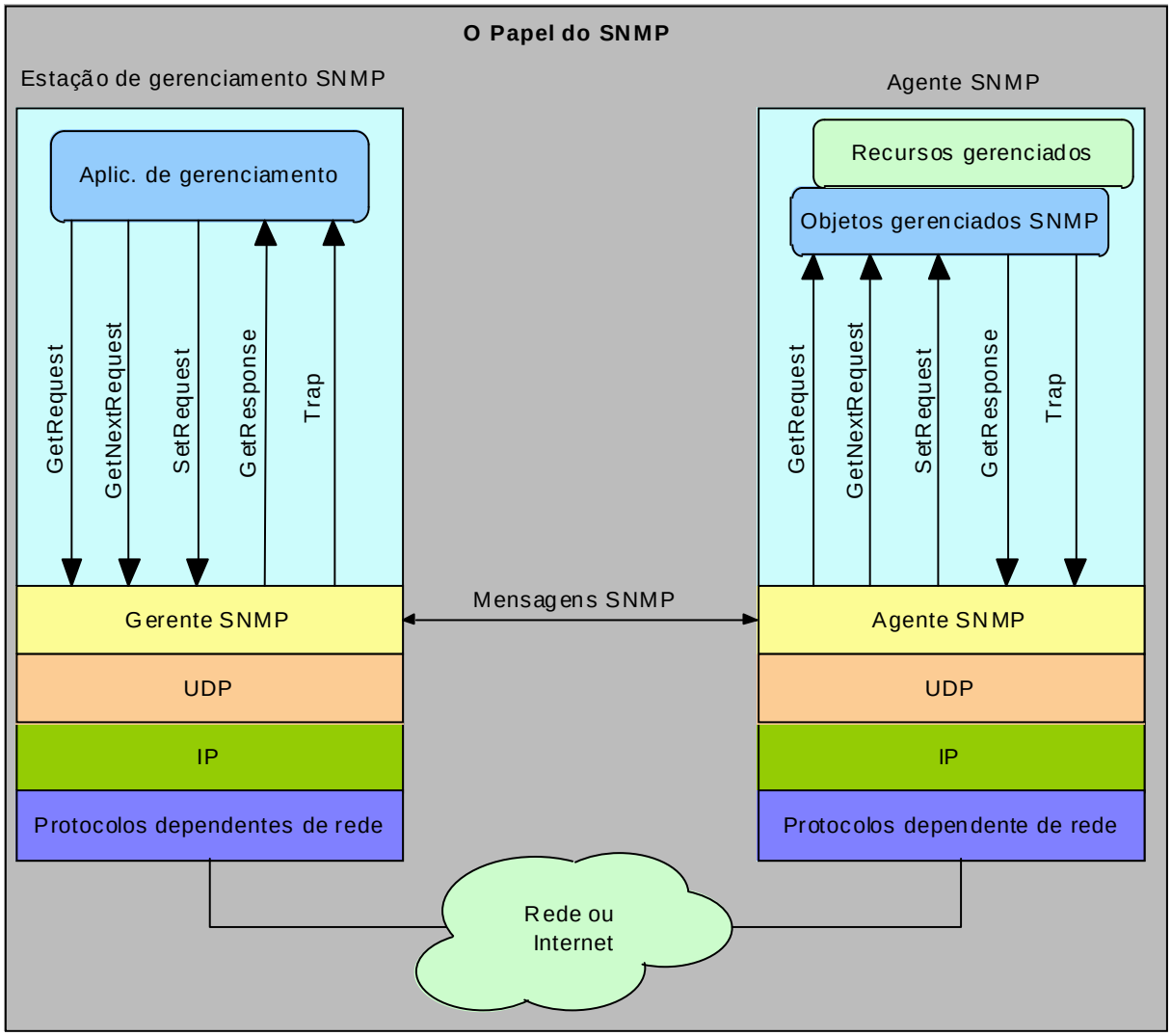
c) *trap*: habilita o agente a notificar espontaneamente a estação de gerenciamento sobre eventos significativos;

Além destas primitivas, o SNMPv2 define uma nova primitiva:

e) *inform*: habilita o agente e o gerente a pedir confirmação de eventos de notificação.

O SNMP foi designado para ser um protocolo de nível de aplicação que é parte do conjunto de protocolos TCP/IP. Ele foi projetado para operar sobre o UDP. Cada agente deve no mínimo, implementar o SNMP, o UDP e o IP para poder agir como um agente (fig. 5).

FIGURA 5 – O papel do SNMP



FONTE: Adaptado de Stallings (2001, pág. 81).

Um processo agente interpreta as mensagens SNMP e controla sua MIB. Três tipos de mensagem agem em favor de uma aplicação de gerenciamento na estação de gerenciamento: *GetRequest*, *GetNextRequest*, e *SetRequest*. As duas primeiras são variações da função *get*. Todos os três tipos de mensagens são respondidos por um agente na forma de uma mensagem *GetResponse*, que é passada para a aplicação de gerenciamento. Além destes tipos de mensagens, um agente pode enviar uma mensagem de *Trap* em resposta a um evento que afete a MIB e por conseqüência, os recursos gerenciados. Como o SNMP depende do UDP, que é um protocolo não orientado à conexão, o SNMP é em si, sem conexão.

4.1.2 SMI – STRUCTURE OF MANAGEMENT INFORMATION

Segundo Stallings (2001), a base do sistema de gerenciamento usado no modelo SNMP é um banco de dados contendo informações sobre os elementos gerenciados. Tanto no ambiente TCP/IP quanto no OSI, esta base de dados é chamada de base de informações ou simplesmente MIB, onde cada recurso gerenciado é representado por um objeto. A MIB é uma coleção de tais objetos. Para o SNMP a MIB é, em essência, uma base de dados estruturada na forma de uma árvore. Cada elemento da rede contém uma MIB que reflete o estado dos recursos gerenciados pelo sistema de gerência. Uma entidade de gerenciamento de rede pode monitorar os recursos através da leitura dos valores dos objetos na MIB e pode controlar estes recursos gerenciados modificando estes valores.

Para que a MIB possa atender às necessidades de um sistema de gerenciamento, ela deve atender certos objetivos:

a) o objeto ou objetos usados para representar um recurso em particular deve ser o mesmo em todos os sistemas;
b) um projeto comum para representação destes objetos deve ser usado, para suportar a interoperabilidade.

O item (b) é alcançado no SNMP pela definição de uma estrutura de gerenciamento de informações, o SMI.

O SMI, que é especificado na RFC 1155, define a estrutura principal na qual a MIB pode ser definida e construída. O SMI identifica os tipos de dados que podem ser usados na MIB e especifica como os recursos na MIB serão representados e nomeados. A filosofia por

trás do SMI é encorajar a simplicidade e a extensibilidade das MIBs. O SMI não suporta a criação ou a recuperação de estruturas de dados complexas. As MIBs irão inevitavelmente conter tipos de dados definidos pelos vendedores de equipamentos de rede e, a menos que restrições na definição de tais tipos de dados sejam definidas, haverá perda de interoperabilidade.

Para fornecer uma forma padrão de representação de informações de gerenciamento, o SMI deve fazer o seguinte:

a) fornecer uma técnica padrão para definir a estrutura de uma MIB em particular;
b) fornecer uma técnica padrão para definir objetos individuais, incluindo a sintaxe e valor de cada objeto;
c) fornecer uma técnica padrão para codificação dos valores dos objetos.

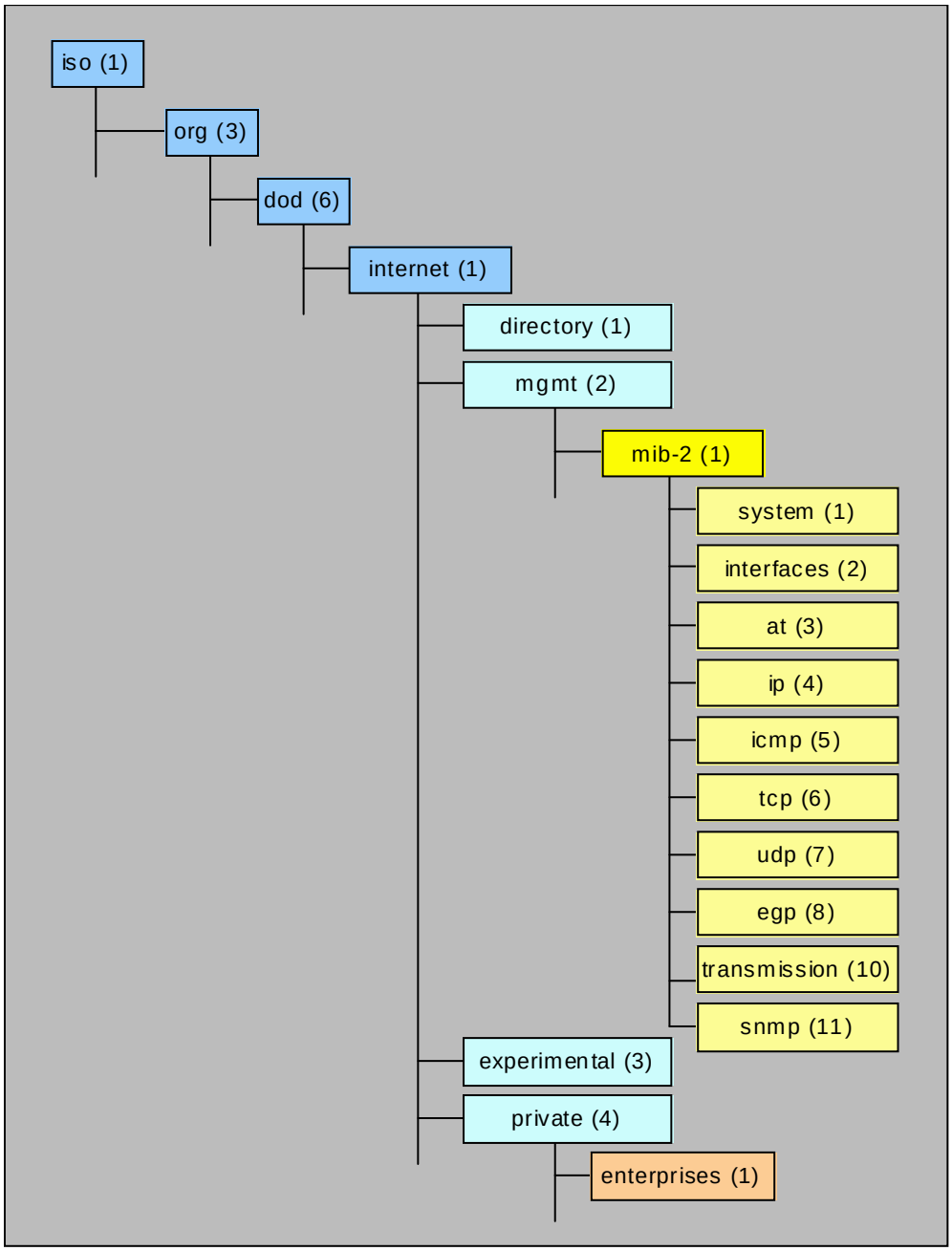
4.1.2.1 ESTRUTURA MIB

Para Stallings (2001), todos os objetos gerenciados no ambiente SNMP estão distribuídos em uma estrutura hierárquica ou em árvore. Os objetos “folhas” desta árvore são os objetos gerenciados em si, cada um deles representando algum recurso, atividade, ou informação relevante que pode ser gerenciada. A estrutura em si define um agrupamento de objetos em conjuntos de objetos logicamente relacionados. Associado com cada tipo de objeto na MIB há um identificador do tipo *Abstract Syntax Notation One* - ASN.1 chamado *OBJECT IDENTIFIER* que serve para nomear o objeto. Em adição, devido ao valor associado ao tipo *OBJECT IDENTIFIER* ser hierárquico, a convenção de nomeação também serve para identificar a estrutura dos tipos de objetos. Partindo da raiz (*root*) da árvore *OBJECT IDENTIFIER*, cada valor componente do identificador de objeto identifica uma folha na árvore (fig. 6).

Partindo da raiz, há três nós no primeiro nível: *iso*, *ccitt*, e *joint-iso-ccitt*. Abaixo do nó *iso*, uma das subárvores é para uso por parte de outras organizações, uma delas sendo o U.S. *Department of Defense* (*dod*). A RFC 1155 assume que uma subárvore abaixo do *dod* será alocada para administração pela IAB como segue: *internet OBJECT IDENTIFIER* ::= {*iso* (1) *org* (3) *dod* (6) 1}.

Por isso o nó *internet* tem o identificador de objeto 1.3.6.1. Este valor serve como prefixo para os nós abaixo deste nível.

FIGURA 6 - Grupos de objetos MIB-2



FONTE: Adaptado de Stallings (2001, pág. 82).

Como mostra a figura 6, o SMI define quatro nós abaixo do nó *internet*:

a) *directory*: reservado para uso futuro com o diretório OSI;

b) *mgmt*: usado por objetos definidos em documentos aprovados pelo IAB;

c) *experimental*: usado para identificar objetos usados em experimentos na Internet;
d) *private*: usado para identificar objetos definidos unilateralmente.

A subárvore *mgmt* contém as definições de bases de gerenciamento MIB que foram aprovados pelo IAB. Atualmente, duas versões da MIB foram desenvolvidas, *mib-1* e *mib-2*.

A segunda é uma extensão da primeira e é o padrão usado pelo SNMPv3. Ambas possuem os mesmos identificadores de objeto na subárvore, mas apenas uma das MIBs está presente em qualquer configuração. Objetos adicionais podem ser definidos para uma MIB das seguintes maneiras:

a) a subárvore *mib-2* pode ser expandida ou substituída por uma revisão completamente nova (presumivelmente "*mib*-3"). Para expandir a *mib-2* uma nova subárvore é definida;
b) uma MIB experimental pode ser construída para uma aplicação em particular. Tais objetos podem subseqüentemente ser removidos da subárvore *mgmt*;
c) extensões privadas podem ser adicionadas a subárvore *private*.

Hegering (1994) define um único nível estrutural integrante da MIB *internet*, o nível *group*. Este é subdividido em 10 subgrupos conforme segue:

a) *system*: informações de configuração sobre o nó gerenciado como um todo;
b) *interfaces*: informações de interface sobre os sistemas conectados;
c) *at – addres translation*: informações para resolução de endereços;
d) *ip, icmp, tcp, udp, egp, snmp*: informações sobre os protocolos IP, ICMP, TCP, UDP, *Exterior Gateway Protocol* – EGP, e SNMP;
e) *transmission*: informações sobre tipos específicos de interfaces de rede tais como Ethernet, *Token Ring*, *loopback*, etc.

Atualmente, segundo Stallings (2001), a subárvore *private* contém apenas um nó definido: o nó *enterprises*. Esta parte da subárvore é usada por vendedores para aperfeiçoar o gerenciamento de seus dispositivos e para compartilhar informações com outros usuários e vendedores que necessitem interoperar com seus sistemas.

A divisão do nó *internet* em quatro subárvores fornece uma fundamentação forte para a evolução das MIBs. Com os experimentos de novos objetos por parte dos vendedores e de

outros desenvolvedores, se ganha um conhecimento prático desses novos objetos antes que eles sejam definidos como parte integrante dos padrões para o subgrupo *mgmt*. Com isso, a MIB é útil imediatamente para gerenciar objetos que se encaixam na parte padrão da MIB e é flexível o bastante para se adaptar às mudanças de ofertas de produtos e tecnologias.

Ao conjunto de operações de gerenciamento permitidas a uma aplicação gerente por um agente denomina-se “política de acesso”; a coleção de objetos que são visíveis para estas operações é denominada visão MIB, ou simplesmente uma visão (Hegering, 1994).

4.1.2.1.1 SUBGRUPO IP

Para o desenvolvimento do protótipo, serão utilizados alguns dos objetos escalares existentes no grupo *ip* da *mib-2*. Estes objetos serão utilizados para realizar o monitoramento de desempenho, através da medição da utilização de *throughput* da rede - no que diz respeito aos pacotes IP transmitidos e recebidos - pelo dispositivo analisado, fornecendo assim informações dinâmicas sobre a utilização destes recursos feitos pelo dispositivo em questão e o seu impacto na utilização total da largura de banda da rede na qual está inserido.

Segundo Zeltserman (1999), o subgrupo *ip* é composto de objetos escalares, uma tabela de endereços, uma tabela de roteamento e uma tabela de endereços de rede para mídias. Alguns dispositivos implementam diversos módulos de roteamento e mantém objetos do grupo MIB para cada módulo. É possível acessar subgrupos *ip* diferentes através de *strings* de comunidade *proxy* ou através de endereços IP diferentes.

Os objetos escalares que atualmente fazem parte do subgrupo *ip* são:

a) *ipForwarding*: pode ser *forwarding* (1) ou não *forwarding* (2). Quando o *ipForwarding* estiver habilitado, o dispositivo age como um roteador normal e irá encaminhar os pacotes IP de uma sub-rede para outra quando requisitado. Quando o *ipForwarding* estiver desabilitado, o dispositivo descarta os pacotes IP não endereçados para uma de suas interfaces definidas localmente. Irá também incrementar o contador *ipInAddrErrors* para cada pacote descartado;
b) *ipDefaultTTL*: tempo de vida (*Time-To-Live* - TTL) padrão;
c) *ipInReceives*: número total de datagramas de entrada recebidos de todas as interfaces, incluindo aquelas com erros;

d) *ipInHdrErrors*: número de datagramas de entrada descartados devido a erros no seu cabeçalho IP. Isto inclui falhas de *checksum*, número de versão incompatível, outros erros de formato, tempo de vida excedido, e erros descobertos no processamento das opções IP;
e) *ipInAddrErrors*: número de datagramas de entrada descartados porque o endereço IP de destino não é válido. Isto inclui endereços inválidos, endereços de classes não suportadas e pacotes que não podem ser encaminhados porque o *ipForwarding* está desativado;
f) *ipForwDatagrams*: número de datagramas encaminhados;
g) *ipInUnknownProtos*: número de datagramas recebidos com sucesso mas descartados porque o protocolo era ou desconhecido ou não suportado;
h) *ipInDiscards*: número de datagramas de entrada descartados devido a limitações de recursos, como por exemplo, a falta de espaço em *buffer*;
i) *ipInDelivers*: número de datagramas entregues para os protocolos de usuário IP local;
j) *ipOutRequests*: número de datagramas IP requisitados para transmissão. Este contador não inclui qualquer datagrama contado no *ipForwDatagrams*;
k) *ipOutDiscards*: número de datagramas de saída descartados devido à falta de recursos;
l) *ipOutNoRoutes*: número de datagramas descartados porque não foi possível encontrar o roteador no endereço de destino;
m) *ipReasmTimeout*: valor de *"timeout"* em segundos pelo qual fragmentos IP recebidos esperam enquanto aguardam o reagrupamento;
n) *ipReasmReqds*: número de fragmentos IP recebidos que precisam ser reagrupados;
o) *ipReasmOKs*: número de datagramas IP reagrupados com sucesso;
p) *ipReasmFails*: número de falhas de reagrupamento;
q) *ipFragsOKs*: número de datagramas IP fragmentados com sucesso;
r) *ipFragsFails*: número de datagramas IP que necessitam fragmentação mas que precisam ser descartados porque possuem a *flag* IP "não fragmentar" configurada;
s) *ipFragCreates*: número de fragmentos IP criados;
t) *ipRoutingDiscards*: número de entradas na tabela de roteamento que foram descartadas devido a limitações de recursos.

Estes objetos podem ajudar a identificar especificamente:

a) limitações de recursos;
b) grande número de pacotes sendo fragmentados. Isto pode ser causado devido a incompatibilidades na *Maximum Transfer Unit* - MTU. A redução da quantidade de fragmentação irá provavelmente aumentar o desempenho da rede;
c) grande número de pacotes sendo reagrupados. Novamente, isto pode estar acontecendo devido a incompatibilidades nas MTUs;
d) um grande percentual de falta de rotas. Isto pode indicar que um dispositivo não está recebendo atualizações de roteamento apropriadamente ou que a tabela de roteamento foi desconfigurada e não contém rotas válidas;
e) um grande percentual de reagrupamentos falhos. Pode indicar tanto que fragmentos estão sendo corrompidos ou que fragmentos IP estão sendo descartados devido à falta de recursos;
f) um grande percentual de fragmentações falhas. Provavelmente indica que os dispositivos estão configurados com a *flag* “não desfragmentar”.

Enquanto que o conhecimento de que está se detectando um grande número de pacotes com erros de cabeçalhos ou de endereçamento é fácil, a solução do problema específico não é tão fácil e geralmente dependente de outros fatores que não serão abordados no presente trabalho.

4.1.2.2 ASN.1

Para Lynch (1993), na camada de aplicação as estruturas de dados trocadas pelas entidades de protocolo são potencialmente muito mais complexas. Portanto, é necessário introduzir um novo formalismo para descrever estas estruturas. Esse novo formalismo é denominado sintaxe abstrata, que é usada para definir dados sem considerar as estruturas orientadas à máquina e suas restrições. Na estrutura de gerenciamento, uma linguagem OSI denominada ASN.1 é usada para atender este princípio. Vale lembrar que a ASN.1 é usada por duas razões distintas pela estrutura de gerenciamento:

a) definir o formato dos dados (objeto e seu respectivo valor) trocados pelo protocolo de gerenciamento; e
b) definir os objetos que são gerenciados.

Logo, a sintaxe abstrata é usada para descrever ambos, as estruturas de dados trocadas no nível de protocolo e a informação gerenciada que é transportada por estas estruturas de dados.

Segundo Stallings (2001), cada objeto em uma MIB SNMP é definido formalmente; a definição especifica os tipos de dados dos objetos, suas formas aceitáveis e limites de valores, e seu relacionamento com outros objetos da MIB. A notação ASN.1 é usada para definir cada objeto individualmente e, além disso, definir toda a estrutura da MIB.

4.1.2.3 CODIFICAÇÃO

Os objetos na MIB são codificados usando regras básicas de codificação (B*asic Encoding Rules* - BER) associadas com a ASN.1. Apesar de não ser a forma mais compacta ou eficiente de codificação, a BER é uma estrutura de codificação amplamente usada e padronizada. (Stallings, 2001)

4.1.3 TRAP-DIRECTED POLLING

De acordo com Stallings (2001), as informações que são úteis para o monitoramento da rede são coletadas e armazenadas por agentes e disponibilizadas para um ou mais sistemas gerentes. Se uma estação de gerenciamento é responsável por um grande número de agentes, e se cada agente guarda um grande número de objetos, então se torna impraticável para a estação de gerenciamento regularmente sondar (*poll*) todos os agentes para obter todos os seus dados de objetos passíveis de leitura.

Ao invés disso, duas técnicas são usadas para tornar as informações do agente disponíveis ao gerente: *polling* e *event reporting*.

Polling é uma interação pergunta-resposta entre gerente e agente. O gerente pode consultar qualquer agente (ao qual possuir autorização de acesso) e requisitar os valores de vários elementos de informação, e o agente responde com as informações da sua MIB.

Um sistema gerente pode usar o *polling* para aprender sobre a configuração que está gerenciando, para obter periodicamente uma atualização de condições, ou investigar em

detalhe uma área depois de ter sido alertado de um problema. O *polling* também é usado para gerar relatórios para o usuário e para responder a consultas específicas.

No caso do *event reporting*, a iniciativa é do agente e o gerente atua como um ouvinte, aguardando por novas informações. Um agente pode gerar um relatório periódico para informar ao seu gerente seu estado atual. O período do relatório pode ser pré-configurado ou definido pelo gerente. Um agente também pode gerar um relatório quando um evento significativo (por exemplo, uma mudança de estado) ou um evento incomum (por exemplo, uma falha) ocorrer. O *event reporting* é útil para detectar problemas tão logo eles ocorram. É também mais eficiente do que o *polling* para monitorar objetos cujos estados ou valores raramente são alterados. Um sistema de gerenciamento de redes geralmente faz uso dos dois métodos.

Para Rose (1994), com o *polling*, uma aplicação de gerência periodicamente pergunta ao nó gerenciado sobre como estão as coisas. Isto fornece a vantagem de manter a aplicação gerente no controle bem como determinar qual é o “quadro maior” real. A desvantagem é o custo em relação ao tempo. Como a aplicação gerente saberá quais elementos de rede deve sondar e com que freqüência? Se o intervalo for muito curto, a largura de banda é desperdiçada; se for muito longo, a resposta a eventos catastróficos é muito lenta. Uma segunda desvantagem é que tráfego adicional é introduzido na rede. Correspondentemente, a aplicação gerente deve possuir recursos de armazenamento adicionais para atender a este aumento de tráfego.

A estratégia, segundo Stallings (2001), é a seguinte: na inicialização, e talvez em intervalos incomuns, tais como uma vez ao dia, uma estação de gerenciamento pode sondar (*polling)* todos os agentes que conheça para obter algumas informações chave, tais como as características da interface e talvez algumas informações estatísticas básicas. Uma vez estabelecida esta linha-base, a estação de gerenciamento se abstém da sondagem. Agora, cada agente é responsável por notificar a estação de gerenciamento sobre qualquer evento incomum. Estes eventos assíncronos são comunicados no SNMP através de mensagens conhecidas como *traps*. Uma vez alertada sobre uma condição excepcional, a estação de gerenciamento pode decidir tomar ou não alguma ação. Neste ponto, a estação de gerenciamento pode sondar diretamente o agente que relatou a condição, para diagnosticar qualquer problema e para obter mais informações específicas sobre a condição excepcional. O

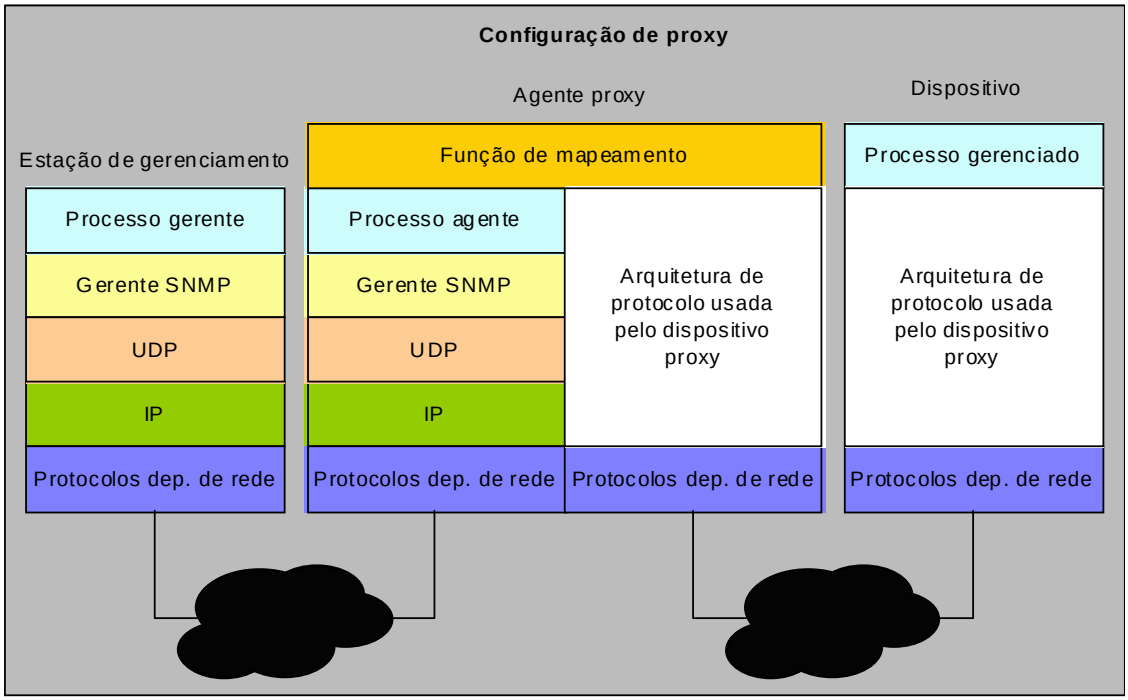
trap-directed polling pode resultar em ganhos substanciais de capacidade da rede e tempo de processamento no agente.

Por isso o SNMP utiliza o *trap-directed pooling*. Quando um evento extraordinário ocorre, o nó gerenciado envia um único e simples *trap* para a aplicação gerente. A aplicação gerente é então responsável por iniciar futuras interações com o nó gerenciado para determinar a natureza e extensão do problema. Esta abordagem é surpreendentemente efetiva: o impacto nos nós gerenciados permanece pequeno; o impacto na largura de banda da rede é minimizado; e, os problemas podem ser resolvidos rapidamente. É claro, como os *traps* são enviados usando um protocolo inseguro (UDP), eles servem apenas como um alerta antecipado; baixas freqüências de *polling* são necessárias como *back-up*.

4.1.4 PROXY

Segundo Stallings (2001), o uso do SNMP requer que todos os agentes, bem como as estações de gerenciamento, suportem um conjunto de protocolos, tais como UDP e IP. Isto limita diretamente o gerenciamento de tais dispositivos e exclui outros, como algumas pontes e modens, que não suportam qualquer parte dos protocolos TCP/IP.

FIGURA 7 – Configuração de *proxy*



FONTE: Adaptado de Stallings (2001, pág. 82).

Além disso, pode haver numerosos sistemas menores (PCs, controladores programáveis) que implementam o TCP/IP para suportar suas aplicações, mas para os quais não é desejável adicionar a estrutura adicional do SNMP, como a lógica do agente e a manutenção da MIB. Para acomodar dispositivos que não implementam o SNMP, o conceito de *proxy* foi criado (fig. 7).

Neste cenário, um agente SNMP age como um *proxy* para um ou mais dispositivos, ou seja, o agente SNMP age em nome dos dispositivos a ele relacionados.

Um agente *proxy* na estrutura do SNMP, segundo Lynch (1993), é um agente que tem os recursos e a autorização para responder às pesquisas e comandos em nome de outro sistema gerenciado.

Isto ocorre porque alguns sistemas não possuem suporte nativo ao protocolo SNMP, mas com este mecanismo de *proxy* podem fazer parte da estrutura de gerenciamento SNMP.

4.1.5 COMPARATIVO ENTRE AS VERSÕES SNMP

O padrão original da estrutura de gerenciamento SNMPv1, segundo o SNMP *International Research* (2001), consiste de três documentos:

a) RFC 1155: define a Estrutura de Gerenciamento de Informações - SMI;
b) RFC 1212: define um mecanismo de descrição mais conciso, mas totalmente consistente com o SMI;
c) RFC 1157: define o protocolo SNMP usado para o acesso via rede aos objetos gerenciados.

Os dois primeiros documentos descrevem a linguagem de definição de dados do SNMPv1. O terceiro documento descreve as operações do protocolo SNMPv1 realizadas pelas *Protocol Data Units* - PDUs em listas de ligação de varáveis. As operações definidas pelo SNMPv1 são: *get*, *get-next*, *get-response*, *set-request*, e *trap*. O posicionamento do SNMPv1 na camada de transporte usando um serviço de transporte sem conexão também é definido. Muitos destes conceitos fazem parte da estrutura do SNMPv3.

A estrutura do SNMPv1 descreve o encapsulamento das PDUs SNMPv1 em mensagens SNMP enviadas entre entidades SNMP, e diferencia entre entidades de aplicação e

entidades de protocolo. No SNMPv3, estas entidades são renomeadas como aplicações e motores, respectivamente.

A estrutura do SNMPv1 também introduz o conceito de um serviço de autenticação com suporte a um ou mais esquemas de autenticação. No SNMPv3, o conceito de um serviço de autenticação é expandido para incluir outros serviços, tais como privacidade.

Finalmente, a estrutura do SNMPv1 introduz o controle de acesso baseado em um conceito chamado visão de MIB SNMP. O SNMPv3 especifica um conceito fundamentalmente similar chamado controle de acesso baseado em visões.

Entretanto, enquanto que a estrutura SNMPv1 antecipou a definição de múltiplos esquemas de autenticação, esta não define nenhum esquema, senão um esquema trivial de autenticação baseado em *strings* de comunidade. Esta é uma das principais fraquezas da estrutura SNMPv1. Entretanto, naquela época, pensava-se que a definição de uma grade de segurança comercial poderia não ser satisfatória no seu projeto e difícil de ser aprovada porque “segurança” significa muitas coisas diferentes para pessoas diferentes. Devido a isso, e porque alguns usuários não requeriam um forte serviço de autenticação, o SNMPv1 estruturou um serviço de autenticação como um bloco separado, a ser definido mais tarde. O SNMPv3 fornece uma arquitetura para ser usada neste bloco, bem como uma definição para seus subsistemas.

A estrutura de gerenciamento SNMPv2 é completamente descrita da RFC 1902 até a RFC 1907. A coexistência e assuntos relacionados à transição do SNMPv1 para o SNMPv2 são discutidas na RFC 1908.

O SNMPv2 fornece várias vantagens sobre o SNMPv1:

a) tipos de dados expandidos: contadores de 64 bits;
b) melhorias de eficiência e desempenho: operador *get-bulk*;
c) confirmação de eventos de notificação: operador *inform*;
d) tratamento de erros mais elaborado: erros e exceções;
e) melhorias nos conjuntos: especialmente na criação e exclusão de colunas;
f) linguagem de definição de dados “afinada”.

Entretanto, a estrutura SNMPv2, conforme descrita nas RFCs de 1902 até 1907, é incompleta no sentido de que não atinge os objetivos originais de desenvolvimento do projeto SNMPv2. Os objetivos não alcançados incluem a provisão de segurança e administração, também chamados de segurança de “grau comercial” com:

a) autenticação: identificação de origem, integridade das mensagens, e alguns aspectos de proteção contra retransmissão e duplicação de mensagens:
b) privacidade: confidencialidade;
c) controle de acesso e autorização; e
d) capacidades de administração e configuração remota conveniente a estas características.

Já a estrutura de gerenciamento SNMPv3, como descrito nas RFCs de 2570 até 2575, corrige as deficiências encontradas no SNMPv2 relacionadas à segurança e à administração. As RFCs SNMPv3 foram produzidas pelo SNMPv3 *Working Group* do *Internet Engineering Task Force* (IETF). O SNMPv3 *Working Group* não “reinventou a roda”, mas reusou os documentos padrões do SNMPv2. Como resultado, o SNMPv3 é o SNMPv2 com segurança e administração.

As novas características do SNMPv3 são:

a) segurança:
   - autenticação e segurança;
   - autorização e controle de acesso;
b) estrutura administrativa:
   - nomeação de entidades;
   - pessoas e políticas de acesso;
   - nomes de usuários e gerenciamento de chaves;
   - notificações de destinos;
   - relacionamentos de proxy.

Estas novas características serão estudadas com mais detalhes no decorrer do trabalho.

4.1.6 CONSIDERAÇÕES FINAIS SOBRE O SNMP

A estrutura de gerenciamento SNMP, segundo Lynch (1993), tem obtido um crescimento e sucesso fenomenal porque preenche os requisitos de gerenciamento de redes atuais. O SNMP é simples, com operações de protocolo elegantes, um projeto de nomeação de variáveis, e provisões para extensão da MIB, características pelas quais se tornou popular entre vendedores de equipamentos de rede bem como entre os administradores de redes. Foram eles, os administradores de rede, que tornaram o SNMP um padrão *de facto* para gerenciamento aberto através de forças de mercado, fazendo com que importantes organizações de padronização abraçassem o SNMP e as especificações a ele referentes como padrões *de jure*. O crescimento do SNMP continua, ao ser aplicado em novas áreas além daquelas as quais se destinava originalmente, como o gerenciamento de largas áreas de trocas de pacotes TCP/IP. O SNMP não serve apenas para *gateways*, mas é utilizado também para monitorar e controlar muitos tipos de sistemas de redes, de dispositivos da camada *Message Authentication Code* - MAC a aplicações de rede. O SNMP não é usado apenas para redes, mas está sendo usado também na administração de sistemas e outras aplicações não voltadas ao gerenciamento de redes. O SNMP não é mais usado apenas para redes TCP/IP, mas está sendo aplicado a todos os tipos de tecnologia de redes, incluindo outras famílias de protocolos, tais como DECnet e OSI, ou em casos onde não há um protocolo de interconexão presente (incluindo UDP e IP), para suportar o gerenciamento de redes. A popularidade do SNMP é internacional.

Como resultado deste crescimento e sucesso inesperado, a estrutura de gerenciamento SNMP, o chamado padrão de “curto-prazo” para gerenciamento de redes aberto não é mais de “curto-prazo”, mas irá continuar a crescer e prosperar enquanto continuar atendendo às necessidades de seus usuários constituintes.

O SNMP não é perfeito. Entretanto, a habilidade de explorar mecanismos de extensibilidade dentro da sua estrutura, tal como as provisões para acomodar a definição de novas variáveis MIB e de melhorar as provisões de segurança, irão ajudar a estrutura a atender os requisitos dos administradores de redes. Estes mecanismos irão ajudar a garantir que o SNMP continue a ocupar um lugar importante na arte e ciência do gerenciamento de redes abertas nos anos que virão (Lynch, 1993).

5 SNMPV3

Provavelmente a característica mais importante do SNMPv3 segundo Zeltserman (1999), é a segurança. O SNMPv3 atinge este objetivo ao oferecer autenticação forte e criptografia de dados. O que a autenticação oferece é:

a) que apenas grupos autenticados possam se comunicar. Isto garante que uma estação gerenciadora possa coletar dados de um dispositivo ou configurá-lo apenas se o dispositivo permitir que aquela estação gerenciadora o acesse;
b) que as mensagens sejam recebidas em um tempo prático (*timely fashion*). Isto evita que as mensagens sejam salvas e/ou adulteradas e repassadas mais tarde de forma a causar danos.

A privacidade só pode ser usada se a autenticação também for usada. Logo as escolhas de segurança disponíveis são: sem autenticação e sem segurança; apenas autenticação; com autenticação e segurança.

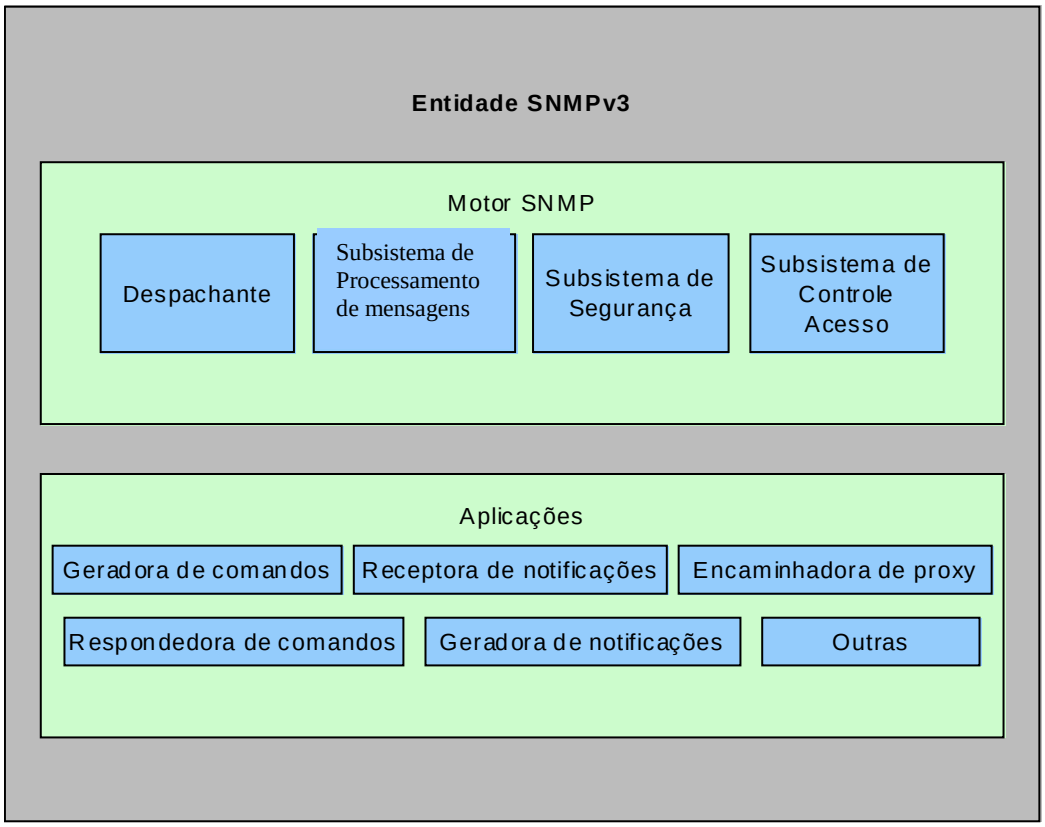
Há duas partes na gerência SNMPv3. Uma é a configuração das aplicações SNMPv3, especificamente fontes de notificações e encaminhadoras de *proxy*:

a) configurar para quem enviar *traps* ou mensagens informativas;
b) configurar encaminhadores de *proxy*.

A outra parte é definir visões MIB que possam ser acessadas por usuários diferentes. Usando isto, se pode controlar que variáveis da MIB determinado usuário pode acessar. Notificações, encaminhadores de *proxy* e controle de acesso às variáveis MIBs podem ser configurados remotamente através de um conjunto complexo de tabelas MIB.

A razão para a popularidade do SNMP e seu contínuo crescimento é a sua simplicidade. De fato, ele possui apenas quatro operações – duas para obter dados, uma para modificar dados, e uma para um dispositivo enviar notificações de forma assíncrona. A complexidade está realmente no gerenciamento dos dados que o SNMP acessa. Nem todos os dados mantidos por um dispositivo são úteis. Parte do que torna difícil a construção de aplicações de gerenciamento de redes úteis é entender que dados de gerenciamento usar e como analisá-los.

FIGURA 8 – Entidade SNMPv3



FONTE: Adaptado de Zeltserman (1999. pág. 98).

A maior missão, entretanto, no desenvolvimento do SNMPv3 foi a de criar uma estrutura modular que pudesse ser facilmente extendida. Com isso se espera que não seja necessária a criação de um “SNMPv4” no futuro. Uma entidade SNMPv3 é composta de duas partes: um motor SNMP e aplicações SNMP (fig. 8). O que era antes chamado de agentes e gerentes SNMP agora se chamam entidades.

5.1 MOTOR SNMPV3

Cada entidade SNMP contém um motor SNMP. Colocando em termos mais simples, um motor SNMP realiza o processamento de mensagens SNMP, com segurança e controle de acesso. Como se vê na figura 8, um motor SNMP é constituído dos seguintes componentes:

a) despachante;
b) subsistema de processamento de mensagens;
c) subsistema de segurança;

d) subsistema de controle de acesso.

5.1.1 DESPACHANTE

É responsável por enviar e receber mensagens. Quando uma mensagem é recebida, o despachante tenta determinar o número de versão da mensagem e então passa a mensagem para o sistema de processamento de mensagens adequado. Se a mensagem não puder ser processada e, assim o número de versão não puder ser determinado, então o *snmpInASNParseErrs* é incrementado e a mensagem é descartada. Se a versão não for suportada pelo subsistema de processamento de mensagens então o contador *snmpInBadVersions* é incrementado e a mensagem é descartada. O despachante é também responsável por enviar PDUs para as aplicações, e por selecionar o meio de transporte mais adequado para enviar as mensagens;

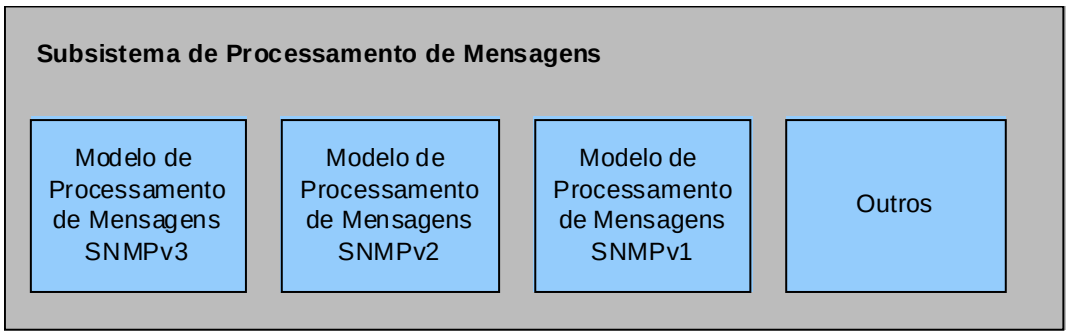
5.1.2 SUBSISTEMA DE PROCESSAMENTO DE MENSAGENS

É constituído de um ou mais modelos de processamento de mensagens. A figura 9 mostra um subsistema de processamento de mensagens que suporta o SNMPv3, o SNMPv2, o SNMPv1 e um modelo chamado “Outros”.

Este modelo permite que outros modelos possam ser acrescentados ao subsistema. O subsistema de processamento de mensagens é responsável por:

a) preparar mensagens para serem enviadas;

FIGURA 9 – Subsistema de processamento de mensagens



FONTE: Adaptado de Zeltserman (1999. pág. 99).

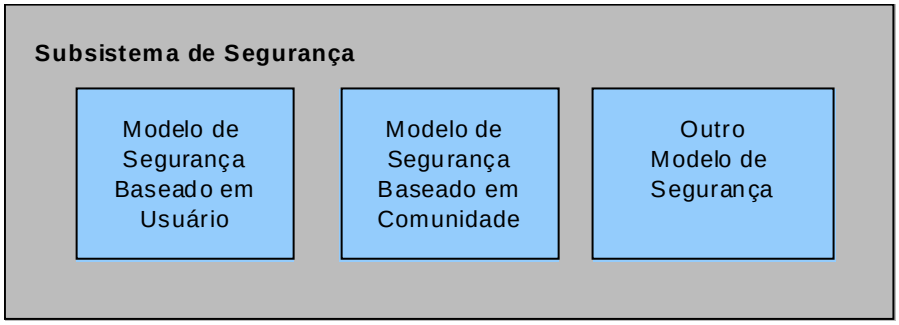
b) extrair dados das mensagens recebidas.

Exemplo: o despachante recebe uma mensagem SNMPv3 válida. O despachante determina a versão da mensagem e a repassa ao modelo de processamento de mensagens SNMPv3. Este por sua vez processa a mensagem extraindo os dados da mesma e chama então o subsistema de segurança para descriptografar a porção de dados da mensagem (se necessário) e se assegura que a mensagem está devidamente autenticada. Neste ponto o despachante irá encaminhar a PDU da mensagem para a aplicação SNMP apropriada.

5.1.3 SUBSISTEMA DE SEGURANÇA

O subsistema de segurança fornece serviços de segurança tais como autenticação de mensagens e criptografia/descriptografia de mensagens para obter privacidade.

FIGURA 10 – Subsistema de segurança



FONTE: Adaptado de Zeltserman (1999. pág. 100).

A figura 10 mostra um subsistema de segurança que suporta o modelo para o SNMPv3, o modelo baseado em comunidade, e um modelo chamado “Outro”.

O modelo baseado em comunidade dá suporte às versões SNMPv1 e SNMPv2c. Um subsistema de segurança define entre outras coisas:

a) as ameaças de segurança para as quais fornece proteção;
b) os serviços que fornece;
c) os protocolos de segurança utilizados para prover os serviços utilizados, tais como privacidade e autenticação.

O modelo de segurança baseado em usuário protege as mensagens SNMPv3 das seguintes ameaças:

a) um usuário autorizado que envie uma mensagem que seja alterada em trânsito por uma entidade SNMP não autorizada;

b) um usuário não autorizado tentando se mascarar como um usuário autorizado;
c) modificações no fluxo das mensagens;
d) espionagem.

O modelo de segurança baseado em usuário atualmente define o uso do *Hashing for Message Authentication Code – Message Digest* 5 – 96 bits (HMAC-MD5-96) e *Hashing for Message Authentication Code – Secure Hash Function* – 96 bits (HMAC-SHA-96) como os protocolos de autenticação e o *Chain Block Cipher – Data Encryption Standard* (CBC-DES) como o protocolo de privacidade. Os modelos de segurança das versões SNMPv1 e SNMPv2c oferecem apenas uma autenticação fraca (nomes de comunidade) e nenhuma privacidade.

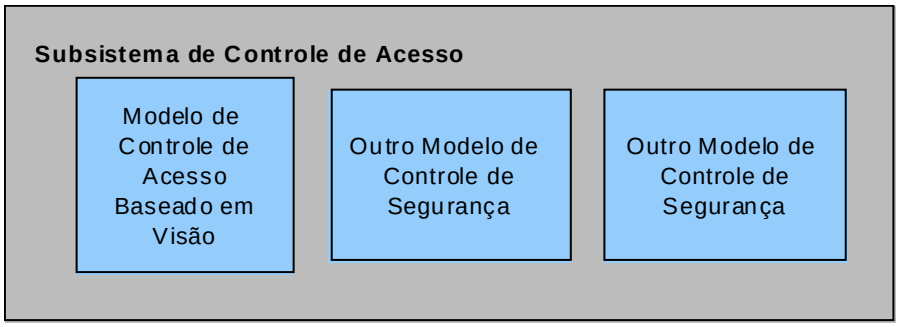
5.1.4 SUBSISTEMA DE CONTROLE DE ACESSO

É responsável por determinar se o acesso a um objeto gerenciado deve ou não ser permitido (fig. 11).

Atualmente há um modelo de controle de acesso definido: o *View-based Access Control Model* (VACM). Com o VACM é possível controlar quais usuários e quais operações podem ter acesso aos objetos gerenciados.

Qualquer aplicação SNMP que precise acessar os objetos gerenciados podem chamar o VACM. Atualmente, seriam as aplicações processadoras de comandos e as geradoras de notificações. Também neste caso é possível definir novos modelos de controle de acesso para serem adicionadas ao subsistema de controle de acesso.

FIGURA 11 – Subsistema de controle de acesso



FONTE: Adaptado de Zeltserman (1999. pág. 102).

Stallings (2001) define que uma entidade SNMPv3 pode ser de dois tipos: uma entidade SNMPv3 gerente tradicional (fig. 12) ou uma entidade SNMPv3 agente tradicional (fig. 13).

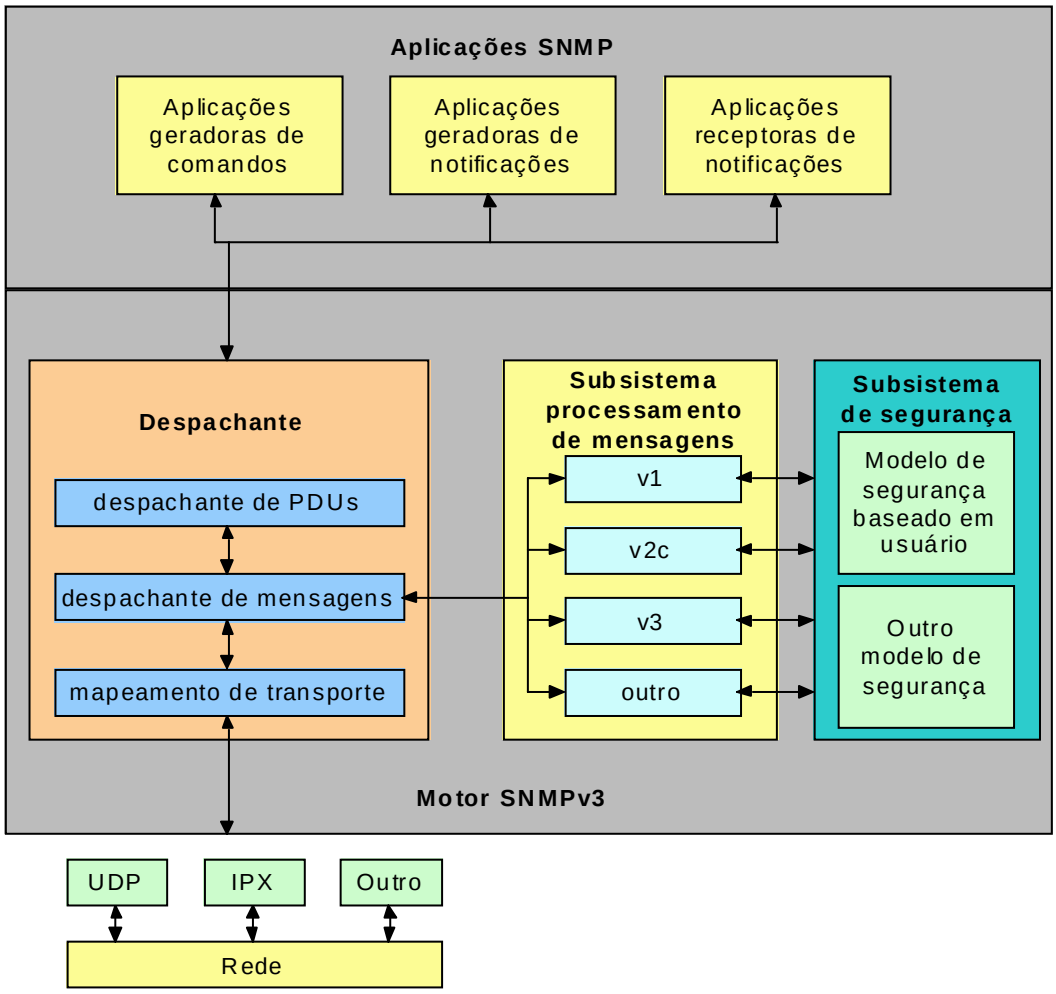
5.1.5 GERENTE SNMP TRADICIONAL

Na terminologia SNMPv3 tradicional, um gerente SNMP inclui três categorias de aplicações (fig. 12). As aplicações geradoras de comandos monitoram e manipulam dados gerenciáveis em agentes remotos; elas fazem uso das PDUs SNMPv1 e SNMPv2, incluindo *Get*, *GetNext*, *GetBulk*, e *Set*. Uma aplicação geradora de notificações inicia mensagens assíncronas; no caso de um gerente tradicional, a PDU *InformRequest* é usada por esta aplicação. Uma aplicação receptora de notificações processa as mensagens assíncronas que chegam ao gerente; estas incluem as PDUs *InformRequest*, SNMPv2-*Trap* e SNMPv1-*Trap*. No caso de uma PDU *InformRequest* chegar ao gerente, a aplicação receptora de notificações irá responder com uma PDU *Response*. Todas estas aplicações fazem uso dos serviços fornecidos pelo motor desta entidade. O motor SNMP realiza duas funções principais:

a) ele aceita PDUs vindas das aplicações SNMP, realiza o processamento necessário, incluindo a inserção de códigos de autenticação e criptografia, e então encapsula as PDUs em mensagens para transmissão;
b) ele aceita mensagens que chegam da camada de transporte, realiza o processamento necessário, incluindo autenticação e descriptografia, e então extrai as PDUs das mensagens e as passa para a aplicação SNMP apropriada.

Em um gerente SNMP tradicional, o motor SNMP contém um despachante, um subsistema de processamento de mensagens e um subsistema de segurança. O despachante é simplesmente um gerente de tráfego. Para as PDUs de saída, o despachante aceita PDUs vindas das aplicações e realiza as seguintes funções: para cada PDU, o despachante determina o tipo de processamento de mensagem requerido (SNMPv1, SNMPv2, SNMPv3) e passa a PDU no módulo de processamento de mensagens adequado no subsistema de processamento de mensagens. Depois disso, o subsistema de processamento de mensagens retorna a mensagem contendo a PDU incluindo os cabeçalhos apropriados da mensagem. O despachante passa então esta PDU para a aplicação apropriada.

FIGURA 12 – Gerente SNMPv3 tradicional



FONTE: Adaptado de Stallings (2001, pág. 458).

O subsistema de processamento de mensagens aceita as PDUs de saída vindas das aplicações via despachante e as prepara para transmissão, envolvendo-as no cabeçalho apropriado da mensagem e retornado-as ao despachante. O subsistema de processamento de mensagens também aceita mensagens que chegam à entidade e ao serem repassadas pelo despachante, processa cada cabeçalho destas mensagens, e retorna a PDU anexada ao despachante para que ele possa encaminha-la à aplicação adequada.

O subsistema de segurança realiza funções de autenticação e criptografia. Cada mensagem a ser enviada é passada ao subsistema de segurança pelo subsistema de processamento de mensagens. Dependendo dos serviços requisitados, o subsistema de segurança pode criptografar a PDU em anexo, e/ou gerar um código de autenticação para ser

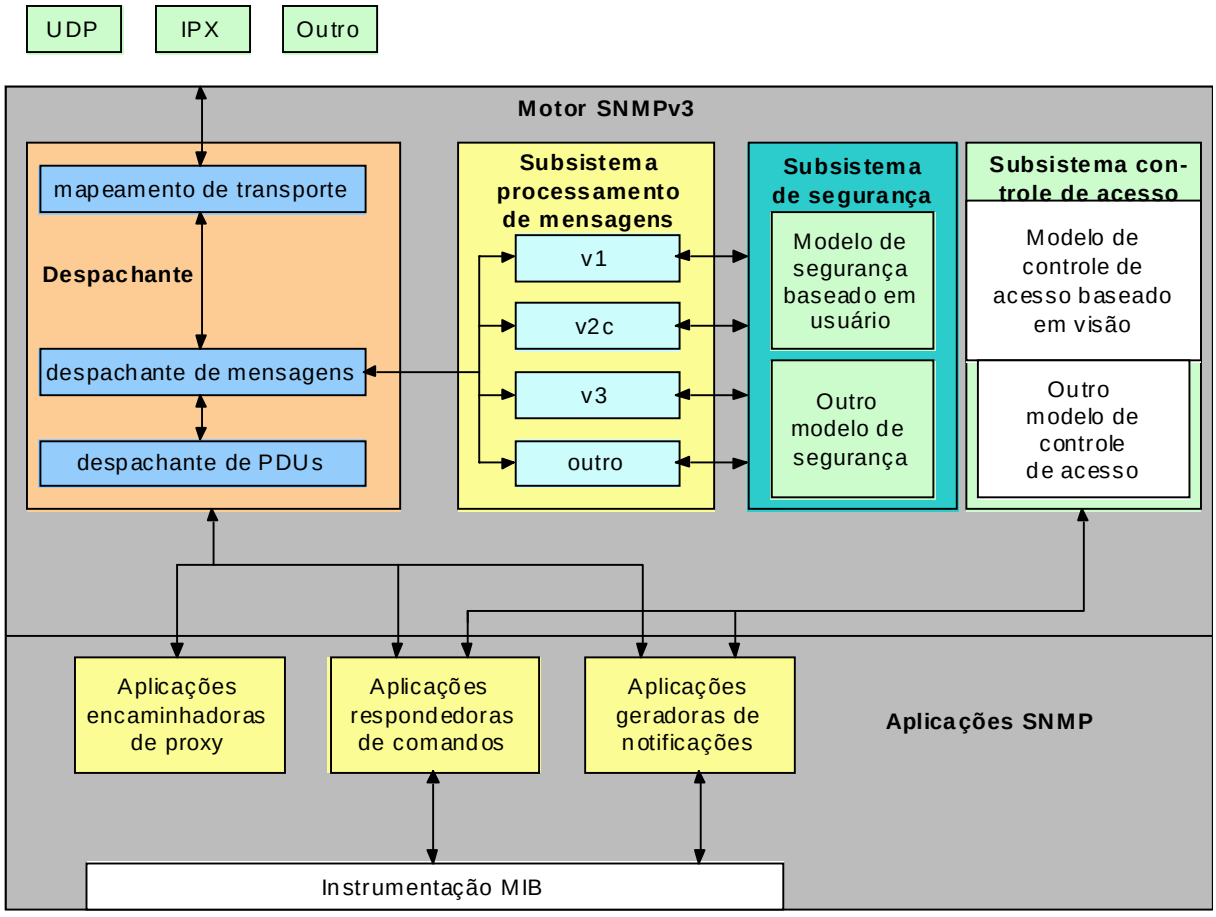
inserido no cabeçalho da mensagem. A mensagem é então retornada ao subsistema de processamento de mensagens. Similarmente, cada mensagem recebida pela entidade gerente é passada ao subsistema de segurança pelo subsistema de processamento de mensagens. Se requisitado, o subsistema de segurança verifica o código de autenticação e realiza a descriptografia da PDU. Após estas operações a mensagem processada é retornada ao subsistema de processamento de mensagens.

5.1.6 AGENTE SNMP TRADICIONAL

Um agente tradicional (fig. 13) deve conter três tipos de aplicações: processadoras de comandos; geradoras de notificações e encaminhadoras de *proxy*.

A aplicação processadora de comandos provê acesso aos dados gerenciados.

FIGURA 13 – Agente SNMPv3 convencional



FONTE: Adaptado de Stallings (2001, pág. 460).

Estas aplicações respondem as requisições que chegam através da obtenção e/ou configuração de objetos gerenciados e emitem então uma PDU *Response*.

Uma aplicação geradora de notificações inicia mensagens assíncronas; no caso de um agente tradicional, a PDU SNMPv2-*Trap*, a PDU SNMPv1- *Trap* ou a PDU *Inform* é usada por esta aplicação. Uma aplicação encaminhadora de *proxy* encaminha mensagens entre entidades.

Um motor SNMP de um agente tradicional possui todos os componentes encontrados no motor SNMP de um gerente tradicional, mais um subsistema de controle de acesso. Este subsistema fornece serviços de autenticação para controlar o acesso às variáveis MIBs para a leitura e configuração de objetos gerenciados. Estes serviços são realizados com base no conteúdo das PDUs.

Note que as funções relacionadas à segurança estão organizadas em dois subsistemas separados: segurança e controle de acesso. O subsistema de segurança se preocupa com a privacidade e a autenticação, e age sobre mensagens SNMP. O subsistema de controle de acesso se preocupa com o acesso autorizado as informações gerenciadas, e age sobre as PDUs SNMPv2c.

5.2 APLICAÇÕES SNMP

No caso do SNMPv3, quando se faz referência às aplicações, está se fazendo referência às aplicações que possuem uma entidade SNMP, como por exemplo, uma aplicação de gerência de redes. Atualmente existem cinco tipos de aplicações definidas:

a) geradoras de comandos: geram comandos SNMP para coletar ou configurar dados gerenciados;
b) processadoras de comandos: fornecem acesso aos dados gerenciados;
c) geradoras de notificações: geram *Traps* ou mensagens *Inform*;
d) receptoras de notificações: recebem e processam *Traps* ou mensagens *Inform*;
e) encaminhadoras de *proxy*: encaminham mensagens entre entidades SNMP.

A partir desta lista, pode-se ver que as aplicações receptoras de notificações e as geradoras de comandos são o que se chamava de gerente SNMP, enquanto que as

processadoras de comandos e geradoras de notificação são o que se chamava de agente SNMP.

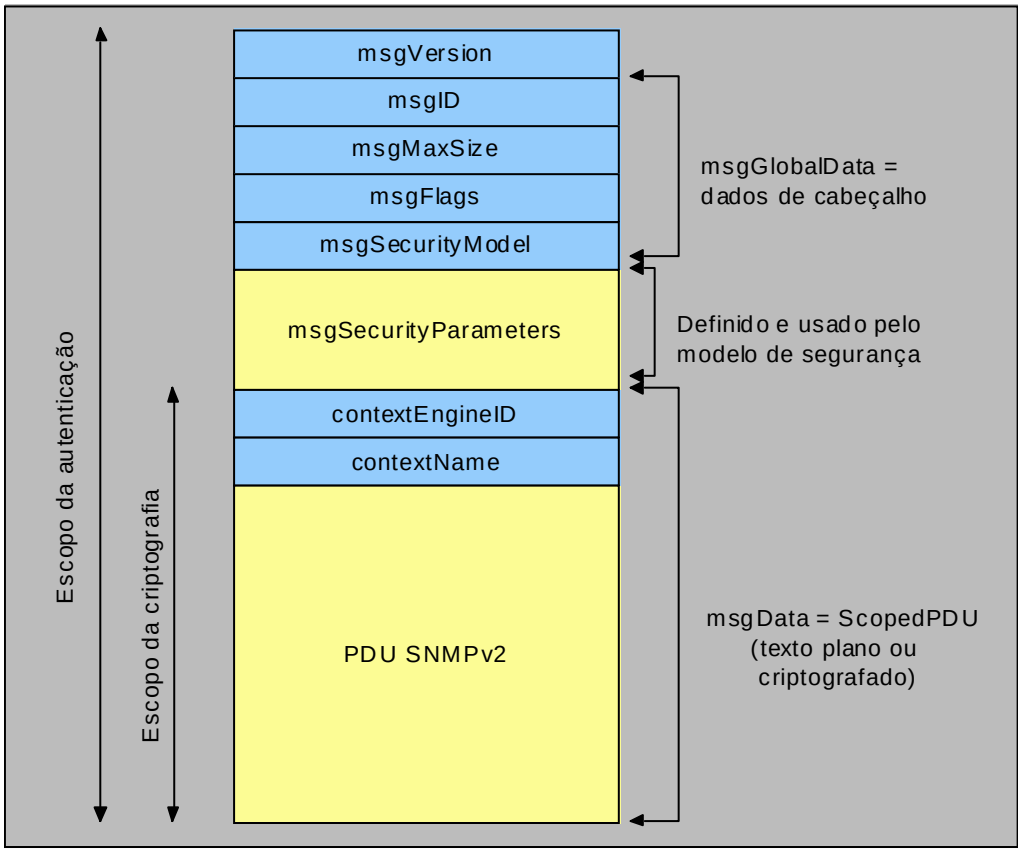
5.3 PROCESSAMENTO DE MENSAGENS SNMPV3

A seguir será demonstrado como é realizado o processamento das mensagens pelas entidades SNMPv3.

5.3.1 FORMATO DAS MENSAGENS SNMPV3

Segundo Stallings (2001), cada mensagem inclui um cabeçalho e uma PDU. A estrutura da mensagem é ilustrada na figura 14; os campos escuros são aqueles criados e processados pelo subsistema de processamento de mensagens.

FIGURA 14 – Formato da mensagem SNMPv3



FONTE: Adaptado de Stallings (2001, pág. 460).

A mensagem SNMPv3 inclui os seguintes campos:

a) *msgVersion*: serve para informar sobre a versão SNMP utilizada na composição da mensagem;

b) *msgID*: um identificador único usado entre duas entidades SNMP para coordenar mensagens de requisição e resposta, e pelo processador de mensagens para coordenar o processamento da mensagem por diferentes modelos de subsistemas existentes na arquitetura das entidades;

c) *msgMaxSize*: contém o tamanho máximo suportado pela mensagem em octetos. Este é o tamanho máximo de segmento que o remetente pode aceitar de outro motor SNMP;

d) *msgFlags*: um octeto contendo três *flags* nos três últimos bits significativos: *reportableFlag*, *privFlag*, *authFlag*. Se *reportableFlag*=1 então uma *Report* PDU deve ser retornada ao remetente sob as condições que podem causar a geração de uma *Report* PDU; quando *reportableFlag*=0, uma *Report* PDU não deve ser retornada. A *flag reportableFlag* é configurada para 1 pelo remetente em todas as mensagens contendo um *request* (*Get*, *Set*) ou um *Inform*, e configurado para 0 para mensagens contendo uma *Response* PDU, uma *Trap* PDU, ou uma *Report* PDU. O *flag reportableFlag* é uma ajuda secundária na determinação de quando enviar uma *Report* PDU. É usado somente nos casos em que a porção PDU da mensagem não pode ser decodificada (por exemplo, quando a descriptografia falha, devido a uma chave incorreta). As *flags privFlag* e *authFlag* são configuradas pelo remetente para indicar o nível de segurança que foi aplicado à mensagem. Para *privFlag*=1, criptografia foi aplicada e para *authFlag*=1, autenticação foi aplicada. Todas as combinações são permitidas exceto *privFlag*=1 e *authFlag*=0; ou seja, não é permitido criptografia sem autenticação;

e) *msgSecurityModel*: um identificador que indica qual modelo de segurança foi usado pelo remetente para preparar a mensagem, e portanto, qual modelo de segurança deve ser usado pelo destinatário para processar a mensagem. Os valores reservados incluem: “1” para SNMPv1, “2” para SNMPv2c, e “3” para o USM.

f) *msgSecurityParameters*: um octeto que contém parâmetros gerados pelo subsistema de segurança na entidade SNMP que enviou a mensagem e processado pelo subsistema de segurança na entidade receptora. O conteúdo deste campo não é

interpretado pelo subsistema de processamento de mensagens nem pelo despachante.

g) *contextEngineID*: um octeto que identifica unicamente uma entidade SNMP. Para as mensagens que chegam, este campo determina para qual aplicação a *scopedPDU* será encaminhada para processamento. Para as mensagens que serão enviadas, este valor é fornecido pela aplicação ao emiter uma requisição para enviar uma mensagem;
h) *contextName*: um octeto que identifica unicamente um contexto em particular com o escopo do motor de contexto associado;
i) PDU SNMPv2: uma PDU. O modelo de processamento de mensagem SNMPv3 especifica que esta deve ser uma PDU SNMPv2.

A figura 14 mostra como estes campos estão organizados. Os campos *msgID*, *msgMaxSize*, *msgFlags*, e *msgSecurityModel* são referenciados na definição ASN.1 pelo nome de *msgGlobalData*. Este bloco contém parâmetros necessários pelo subsistema de processamento de mensagens para coordenar o tratamento e processamento da mensagem. Já os campos *contextEngineID*, *contextName*, e PDU SNMPv2 são referenciados pelo nome *msgData* e possui um tipo *scopedPduData*. Uma PDU de escopo é uma PDU que contém informações nomeadas em um contexto em que são unicamente identificadas pelo *contextEngineID* e pelo *contextName*. Este bloco contém informações necessárias pela aplicação para processar a PDU.

5.4 ALGORITMOS CRIPTOGRÁFICOS USADOS PELO SNMPV3

A seguir serão apresentados alguns conceitos básicos de criptografia e um breve estudo dos algoritmos criptográficos utilizados pelo SNMPv3 para prover a autenticação e a privacidade das mensagens trocadas entre as entidades SNMPv3.

5.4.1 CONCEITOS BÁSICOS DE CRIPTOGRAFIA

Criptografia, do grego *kryptós* (escondido, oculto) + *grápho* (grafia, escrita), é a arte ou a ciência de escrever em cifra ou em código; em outras palavras, é um conjunto de técnicas que permitem tornar incompreensível uma mensagem originalmente escrita com clareza, de

forma a permitir que apenas o destinatário a decifre e a compreenda. Quase sempre o deciframento requer o uso de uma chave (Luchesi, 1986).

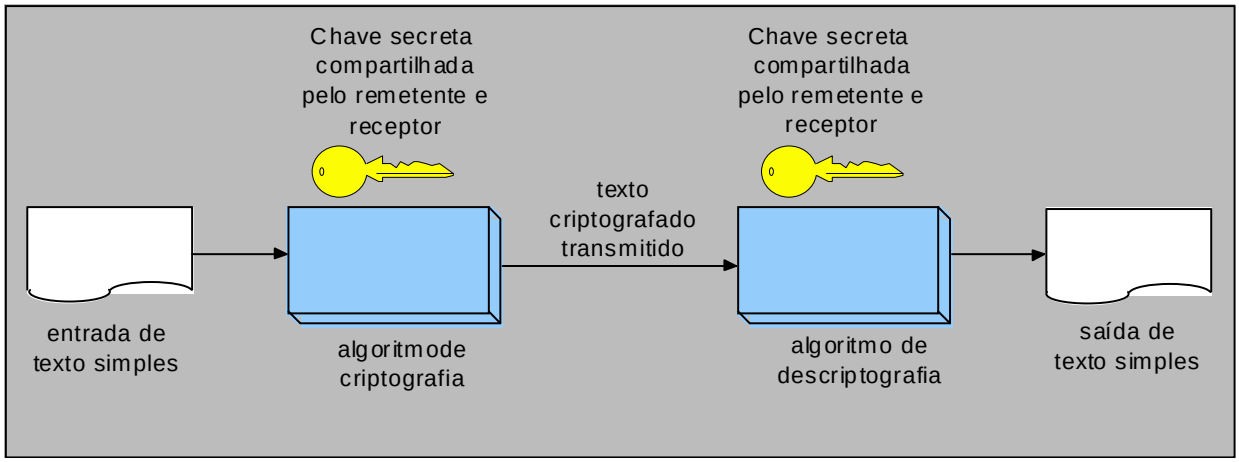
A criptoanálise, do grego *kryptós* + *análysis* (decomposição) é a arte ou ciência de determinar a chave ou decifrar mensagens sem conhecer a chave (Luchesi, 1986).

A criptologia, do grego *kryptós* + *logos* (estudo, ciência), é a ciência que reúne a criptografia e a criptoanálise (Luchesi, 1986).

Stallings (2001) cita que uma estrutura criptográfica padrão possui cinco elementos (fig. 15):

a) texto simples: esta é a mensagem legível ou os dados que são fornecidos como entrada para o algoritmo criptográfico;
b) algoritmo de criptografia: um algoritmo que realiza várias substituições e transformações no texto simples;
c) chave secreta: a chave secreta também é uma entrada para o algoritmo. As substituições e transformações exatas realizadas pelo algoritmo dependem da chave secreta;

FIGURA 15 – Criptografia convencional



FONTE: Adaptado de Stallings (2001, pág. 428).

d) texto cifrado: esta é a mensagem “misturada” produzida como saída do algoritmo. Esta depende da chave secreta e do texto simples. Para uma mensagem dada, duas chaves diferentes irão gerar dois textos cifrados diferentes;

e) algoritmo de descriptografia: é essencialmente o algoritmo de criptografia executado de forma reversa. Ele recebe o texto cifrado e a mesma chave secreta e a partir destes, produz o texto simples original.

Basicamente, segundo Luchesi (1986), a criptografia computacional é usada para permitir:

a) sigilo de informações: permite que somente os usuários autorizados tenham acesso à informação, ou consigam torná-la inteligível;
b) integridade de informações: garantia oferecida ao usuário de que a informação correta, original, não foi alterada, nem intencionalmente nem acidentalmente;
c) autenticação de usuário: processo que permite ao sistema verificar se a pessoa ou processo com quem está se comunicando é de fato a pessoa ou processo que alega ser;
d) autenticação de remetentes: processo que permite a um usuário certificar-se que a mensagem recebida foi de fato enviada pelo remetente, podendo inclusive provar, perante um juiz, que o remetente enviou determinada mensagem;
e) autenticação de destinatários: consiste em se ter uma prova de que a mensagem enviada foi como tal, recebida pelo destinatário;
f) autenticação de atualidade: consiste em provas de que a mensagem é atual, não se tratando de mensagens antigas reenviadas.

5.4.2 DATA ENCRYPTION STANDARD (DES)

Segundo Stallings (2001), a estrutura de criptografia mais utilizada atualmente, é a baseada no DES, adotada em 1977 pelo *National Bureau of Standards*, agora o *National Institute of Standards and Tecnology* (NIST).

O DES cifra blocos de 64 bits em blocos de 64 bits, usando uma chave de 56 bits (64 bits dos quais oito bits são de paridade). Assim, para cifrar, escolhe-se uma chave e divide-se a mensagem em blocos de 64 bits, cifrando cada bloco separadamente.

Por usar a mesma chave tanto para a criptografia quanto para a descriptografia, sendo necessário manter esta chave em segredo (apenas o remetente e o destinatário devem conhecê-la), o DES é dito um algoritmo simétrico de chave secreta (Luchesi, 1986).

O SNMPv3, segundo Stallings (2001), define uma versão melhorada do DES original chamada de *Cipher Block Chaining Mode – Data Encryption Standard* (CBC-DES). Isto porque o DES processa apenas blocos de dados de 64 bits por vez. Para grandes quantidades de texto, é necessário quebrar o texto em blocos maiores que 64 bits. A maneira mais simples de proceder é utilizar o modo *electronic codebook* (ECB), no qual o texto simples é tratado 64 bits de cada vez e cada bloco de texto simples é criptografado usando a mesma chave. O termo "*codebook*" é usado porque, para uma dada chave, há um único texto cifrado para cada bloco de 64 bits de texto simples. Com o ECB, se o mesmo bloco de 64 bits de texto simples aparecer mais de uma vez na mensagem, ele sempre produzirá a mesma saída. Para resolver este problema é preciso usar um mecanismo que gere blocos de textos cifrados diferentes, mesmo usando o mesmo bloco de texto simples. Uma forma simples de fazer isso é usar o modo CBC. Neste esquema, a entrada do algoritmo de criptografia é o *Xtended OR* - XOR do bloco de texto simples atual e do bloco de texto cifrado anterior; sendo que a mesma chave é usada para cada bloco. Como efeito, encadeia-se juntamente o processo da seqüência de blocos de texto simples. A entrada para a função de criptografia para cada bloco de texto simples não possui nenhuma relação com o bloco de texto simples. Portanto, a repetição de padrões de 64 bits não é exposta. Para obter maiores informações sobre o DES consulte Menezes (1997) e Schneier (2001).

5.4.3 MESSAGE DIGEST 5 (MD5)

Stallings (2001) cita que um elemento essencial nas estruturas de autenticação e assinaturas digitais é uma função *hash* segura. Uma função *hash* aceita uma mensagem de tamanho variável como entrada e produz um código *hash* de tamanho fixo, chamado às vezes de sumário da mensagem (*message digest*), como saída. Há duas funções *hash* especificadas para uso com o SNMPv3: a MD5 e a SHA-1.

O algoritmo de sumário de mensagem (*message digest*) - MD5, descrito na RFC 1321 foi desenvolvido por Ron Rivest no MIT. Até recentemente, o MD5 era o algoritmo *hash* seguro mais usado.

Este algoritmo pega uma mensagem de tamanho arbitrário como entrada e produz como saída um sumário de mensagem de 128 bits. A entrada é processada em blocos de 512 bits.

O algoritmo MD5 tem a propriedade de que cada bit do código *hash* é uma função de cada bit da entrada. O autor do MD5 conjectura que a dificuldade de se gerar duas mensagens com o mesmo sumário é da ordem de 2^{64} operações, e a dificuldade de gerar uma mensagem que produza um dado sumário é da ordem de 2^{128} operações. Para obter maiores informações sobre o algoritmo MD5 consulte Menezes (1997) e Schneier (2001).

5.4.4 SECURE HASH FUNCTION (SHA-1)

O algoritmo SHA foi desenvolvido pelo NIST e publicado como o padrão de processamento de informações federais a ser usado pelo norte-americano em 1993. Uma revisão foi feita em 1995 e foi chamada de SHA-1.

O algoritmo SHA-1 possui como entrada uma mensagem com um tamanho máximo menor que 2^{64} bits, e produz como saída um sumário de mensagem de 160 bits. A entrada é processada em blocos de 512 bits. Para obter mais informações sobre o algoritmo SHA-1 consulte Menezes (1997) e Schneier (2001).

5.4.5 AUTENTICAÇÃO DE MENSAGENS COM O HMAC

Segundo Stallings (2001), a autenticação de mensagens é um procedimento que permite a grupos que se comuniquem verificar se as mensagens recebidas são autênticas. Há dois aspectos importantes que são: verificar se o conteúdo da mensagem não foi alterado e se a origem da mensagem é autêntica. O MAC é uma técnica amplamente usada para realizar a autenticação das mensagens, e um algoritmo surgiu como o padrão Internet para uma grande variedade de aplicações: HMAC.

Um algoritmo MAC envolve o uso de uma chave secreta para gerar um pequeno bloco de dados, conhecido como um código de autenticação da mensagem, que é acrescentado à mensagem. Esta técnica assume que dois elementos se comunicando, digamos Alice e Bob, compartilham a chave secreta comum *K*. Se Alice quiser enviar uma mensagem para Bob, ela calcula o código de autenticação da mensagem como uma função da mensagem e da chave. A mensagem mais o código é transmitida para Bob. Bob então realiza o mesmo cálculo na mensagem recebida, usando a mesma chave secreta, para gerar um novo código de

autenticação para a mensagem. O código recebido é comparado com o código calculado. Se o código recebido é igual ao código calculado, então:

a) o receptor tem certeza de que a mensagem não foi alterada;
b) o receptor tem certeza que a mensagem é mesmo do remetente especificado;
c) se a mensagem incluir um número de seqüência, então o receptor tem certeza que a seqüência das mensagens é apropriada, porque seria improvável para um atacante alterar com sucesso o número seqüencial.

Um grande número de algoritmos pode ser usado para gerar o código, mas a alternativa mais eficiente e popular é o HMAC.

O HMAC é descrito na RFC 2104, e foi escolhido como a implementação MAC mandatória (*mandatory-to-implement* MAC) para a segurança no IP, e usado em outros protocolos Internet, tais como o *Secure Socket Layer* (SSL) e *Secure Electronic Transaction* (SET).

A maior vantagem do HMAC é tratar a função *hash* como uma “caixa preta”. Isto possui dois benefícios. Primeiro, uma implementação já existente de uma função *hash* pode ser usada como um módulo na implementação do HMAC. Segundo, se for desejável trocar uma função *hash* específica de implementação do HMAC por outra, tudo o que é necessário é remover o módulo com a função existente e colocar em seu lugar o novo módulo. Com isso, se a segurança da função *hash* for comprometida, a segurança do HMAC pode ser obtida novamente simplesmente substituindo a função *hash* comprometida por uma nova função mais segura (por exemplo, substituir o MD5 pelo SHA-1).

5.5 USER-BASED SECURITY MODEL – USM

Segundo Stallings (2001), a RFC 2274 define o modelo de segurança baseado no usuário (USM) para o SNMPv3. Esta especificação inclui:

a) autenticação: provê integridade de dados e autenticação da origem dos dados. O código de autenticação de mensagens HMAC, que usa atualmente as funções *hash* MD5 ou SHA-1, fornece esta autenticação;
b) sincronização (*timeliness*): protege contra demoras ou duplicação de mensagens;

c) privacidade: protege contra revelação do conteúdo da mensagem. O modo de ciframento encadeado de blocos (CBC) do DES é usado para a criptografia da mensagem;
d) formato da mensagem: define o formato do campo *msgSecurityParameters*, que suporta as funções de autenticação, tempo e privacidade.
e) descoberta: define procedimentos pelos quais um motor SNMP obtém informações sobre outro motor SNMP;
f) gerenciamento de chaves: define procedimentos para a geração de chaves, sua atualização e uso.

5.5.1 PARÂMETROS DE SEGURANÇA USM

A RFC 2274 define uma estrutura, *UsmSecurityParameters*, que especifica o formato interno do campo *msgSecurityParameters* de uma mensagem SNMPv3. A figura 16 mostra o formato de uma mensagem SNMPv3 já com o uso do USM. Os campos escuros indicam os campos que são criados e processados pelo USM.

Antes de examinar os campos do *msgSecurityParameters*, é preciso definir o conceito de um motor SNMP autorizado. Em qualquer mensagem transmitida, uma das duas entidades envolvidas (emissor ou receptor) é designada como um motor SNMP autorizado, de acordo com as seguintes regras:

a) quando uma mensagem SNMP contém uma carga que exige uma resposta (por exemplo, uma PDU *Get*, *GetNext*, *GetBul*k, *Set* ou *Inform*), então o receptor de tal mensagem é autorizado;
b) quando uma mensagem SNMP contém uma carga que não exige uma resposta (por exemplo, um *Trap*-SNMPv2, ou uma PDU *Response* ou *Report*), então o emissor de tal mensagem é autorizado.

Esta designação serve a dois propósitos:

a) O tempo da mensagem é determinado através de um relógio mantido pelo motor autorizado. Quando um motor autorizado envia uma mensagem (*Trap*, *Response*, *Report*), esta contém o valor corrente do relógio deste motor, desta forma o motor não autorizado pode se sincronizar com este relógio. Quando um motor não autorizado envia uma mensagem (*Get*, *GetNext*, *GetBulk*, *Set*, *Inform*), este inclui o

valor de seu tempo corrente, estimado com base no relógio de destino, permitindo ao destino avaliar o tempo da mensagem.;

b) um processo de localização de chave, descrito mais adiante, habilita um único diretor (usuário) a possuir as chaves armazenadas em múltiplos motores; estas chaves são localizadas pelo motor autorizado de tal forma que o diretor é responsável por uma única chave, evitando o risco de segurança de armazenar múltiplas cópias da mesma chave em uma rede distribuída.

Faz sentido designar o receptor das mensagens geradas pelo gerador de comandos e PDUs *Inform* como um motor autorizado pelo fato deste ser o possuidor do relógio autorizado em uma troca. Se uma resposta ou *trap* está atrasada ou é duplicada, poucos danos podem ocorrer. Entretanto, geradores de comandos e, de certa forma, PDUs *Inform* resultam em operações de gerenciamento, tais como leitura ou configuração de objetos MIB. Por isso, é importante garantir que tais PDUs não estejam atrasadas ou duplicadas, o que poderia causar efeitos indesejados.

Quando uma mensagem a ser enviada é passada ao USM pelo processador de mensagens, o USM preenche o campo *msgSecurityParameters*. Quando uma mensagem recebida é passada ao USM pelo processador de mensagens, o USM processa os valores contidos no *msgSecurityParameters* repassando em seguida a mensagem novamente ao processador de mensagens. Os campos do *msgSecurityParameters* são:

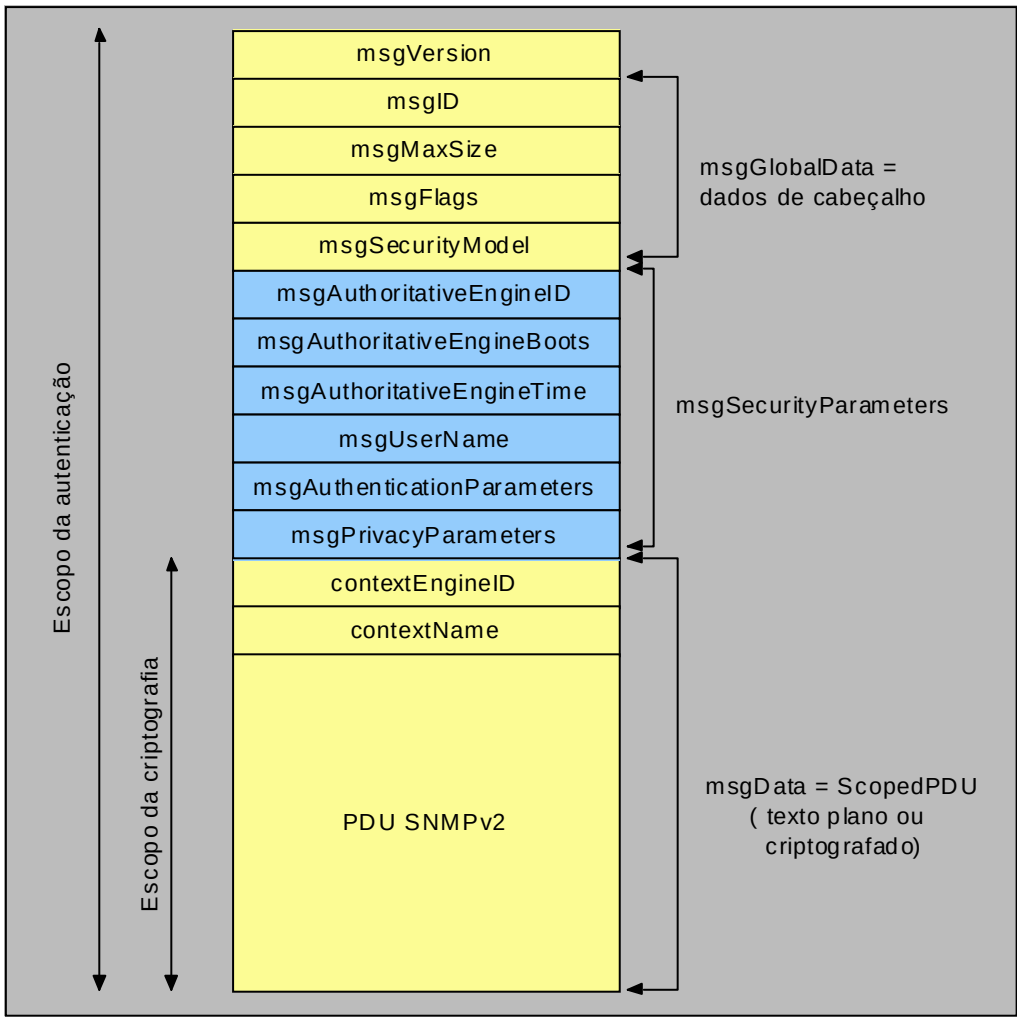
a) *msgAuthoritativeEngineID*: o *snmpEngineID* do motor SNMP autorizado envolvido na troca da mensagem;

b) *msgAuthoritativeEngineBoots*: o valor *snmpEngineBoots* do motor SNMP autorizado envolvido na troca da mensagem. O objeto *snmpEngineBoots* é um inteiro que representa o número de vezes que este motor SNMP inicializou ou reinicializou a si mesmo desde sua configuração inicial;

c) *msgAuthoritativeEngineTime*: o valor *snmpEngineTime* do motor SNMP autorizado envolvido na troca da mensagem. O objeto *snmpEngineTime* é um inteiro que representa o número de segundos desde que este motor SNMP autorizado incrementou pela última vez o objeto *snmpEngineBoots*. Cada motor SNMP autorizado é responsável por incrementar seu próprio valor *snmpEngineTim*e a cada segundo. Um motor SNMP não autorizado é responsável

por incrementar sua “noção” de *snmpEngineTime* para cada motor autorizado remoto com o qual se comunica;

FIGURA 16 – Formato da mensagem SNMPv3 com USM



FONTE: Adaptado de Stallings (2001, pág. 428).

d) *msgUserName*: o diretor (usuário) em favor do qual a mensagem está sendo trocada;

e) *msgAuthenticationParameters*: é nulo se a autenticação não for usada na comunicação agente/gerente. Caso contrário este é um parâmetro de autenticação. Pela definição atual do USM, o parâmetro de autenticação é um código de autenticação de mensagem HMAC;

f) *msgPrivacyParameters*: é nulo se não é usado privacidade na comunicação agente/gerente. Caso contrário este é um parâmetro de segurança. Pela definição

atual do USM, o parâmetro de privacidade é um valor usado para formar o valor inicial no algoritmo CBC-DES (*privPassword* e *SnmpEngineID)*;

O conjunto de mecanismos de sincronização do USM inclui:

a) gerenciamento de relógios autorizados: cada motor SNMP que possa vir a agir como um motor autorizado (geralmente um agente), deve manter dois objetos, *snmpEngineBoots* e *snmpEngineTime*, que se referem ao seu tempo local. Com isso todos os motores que possam receber PDUs *Inform* devem manter estes dois objetos;
b) sincronização: um motor SNMP que opere de modo não autorizado deve permanecer permanentemente sincronizado com cada motor SNMP autorizado com o qual este se comunica. Para este fim, o motor não autorizado mantém uma cópia local de três variáveis (*snmpEngineBoots*, *snmpEngineTime*, e *latestReceivedEngineTime*) para cada motor SNMP autorizado que é conhecido por este motor não autorizado (geralmente um gerente);
c) checagem de tempo por um motor autorizado: o SNMPv3 dita que uma mensagem deve ser recebida em uma janela de tempo razoável, para evitar demora e ataques de duplicação. O tempo da janela deve ser escolhido para ser o mais pequeno possível dada a precisão do relógio envolvido, os atrasos de comunicação do "*round-trip*", e a freqüência com que os relógios são sincronizados. O teste atual para determinar este tempo depende se o receptor da mensagem é autorizado ou não;
d) checagem de tempo por um motor não autorizado: para cada mensagem de entrada a ser autenticada cujo *msgAuthoritativeEngineID* não for igual ao valor do *snmpEngineID* para este motor, o motor compara os valores do *msgAuthoritativeEngineBoots* e *msgAuthoritativeEngineTime* da mensagem recebida com os valores do *snmpEngineBoots* e *snmpEngineTime* que este motor mantém, referente ao motor SNMP remoto correspondente.

Segundo Stallings (2001), cada entidade SNMP mantém uma MIB, a *usmUser*, que informa sobre os usuários (diretores) locais e remotos. Este grupo consiste de um objeto escalar, *usmUserSpinLock*, e uma tabela *usmUserTable*. O objeto *usmUserSpinLock* habilita aplicações geradoras de comandos a coordenar o uso de suas fábricas para troca de chaves

usando a *usmUserTable* e para garantir a atomicidade quando da realização de operações nesta tabela a fim de evitar inconsistências nos dados da mesma.

5.5.2 FUNÇÕES CRIPTOGRÁFICAS USADAS PELO USM

Segundo Stallings (2001), há duas funções criptográficas definidas para o USM: autenticação e criptografia. Para suportar estas funções, um motor SNMP requer dois valores: uma chave privada (*privKey*) e uma chave de autenticação (*authKey*). Valores separados destas duas chaves são mantidas pelos seguintes usuários:

a) usuários locais: qualquer diretor neste motor SNMP para o qual são autorizadas operações de gerenciamento;
b) usuários remotos: qualquer diretor em um motor SNMP remoto com o qual a comunicação é desejada.

Estes valores são atributos de usuário armazenados para cada diretor (usuário) relevante. Os valores da *privKey* e *authKey* não são acessíveis via SNMP.

Para a autenticação o USM permite o uso de um dos seguintes protocolos de autenticação: HMAC-MD5-96 ou HMAC-SHA-96. Para o HMAC-MD5-96, o código de autenticação de mensagem HMAC é usado, com o MD5 atuando como a função *hash*. Uma *authKey* de 128 bits é usada como entrada para o algoritmo HMAC. O algoritmo produz uma saída de 128 bits, que é truncada para 96 bits. Este valor é o código de autenticação da mensagem que será inserido no campo *msgAuthenticationParameters* da mensagem SNMPv3. Para o HMAC-SHA-96, a função *hash* usada é a SHA-1. A *authKey* possui 160 bits de tamanho. O algoritmo produz uma saída de 160 bits que é truncada para 96 bits.

Para a criptografia e descriptografia o USM usa o modo CBC do DES. Uma *privKey* de 128 bits é fornecida como entrada ao protocolo de criptografia. Os primeiros 64 bits desta *privKey* são usados como uma chave DES. Como o DES requer apenas uma chave de 56 bits, o último bit significativo de cada octeto é ignorado. Para o modo CBC (64 bits restantes), um vetor de inicialização (VI) de 64 bits é requerido. Este vetor é produzido da seguinte maneira:

a) os oito últimos octetos da *privKey* são usados como um pré-VI;
b) para garantir que dois VIs são usados para duas entradas de textos planos diferentes criptografados com a mesma chave, um valor-sal (*salt-value*) de oito octetos é

gerado. O valor-sal é igual à concatenação do valor corrente da *snmpEngineBoots* existente neste motor (que possui um tamanho de quatro octetos) e um inteiro de 32 bits mantido pelo algoritmo de criptografia local. Este inteiro de 32 bits é dependente de implementação e deve ser modificado depois de cada uso do mesmo;

c) o valor-sal sofre uma operação XOR bit a bit com o pré-VI para produzir o VI.

O valor-sal é colocado no campo *msgPrivacyParameters* da mensagem SNMPv3 para permitir que a entidade receptora possa computar o VI corretamente. Este esquema faz o seguinte: como o valor-sal muda toda vez que é usado, um VI diferente é usado para diferentes textos planos. E como apenas o valor-sal é transmitido via SNMP, um atacante não pode determinar o VI.

5.5.3 PROCESSAMENTO DE MENSAGENS USM

A figura 17 mostra em termos gerais, os procedimentos seguidos pelo USM para o processamento de mensagens recebidas e enviadas. A seguir será mostrado como é feito este processamento ignorando-se alguns erros básicos que podem interromper o processo, tais como um campo *msgSecurityParameters* inválido. O USM realiza os seguintes passos ao receber um *generateRequestmsg* (primitiva gerada pelo subsistema de processamento de mensagens) tratado por uma aplicação geradora de comandos:

a) o USM determina se há uma entrada na *usmUserTable* para o destino *securityEngineID* e a fonte *securityName*. Se não houver, uma indicação de erro é retornada;

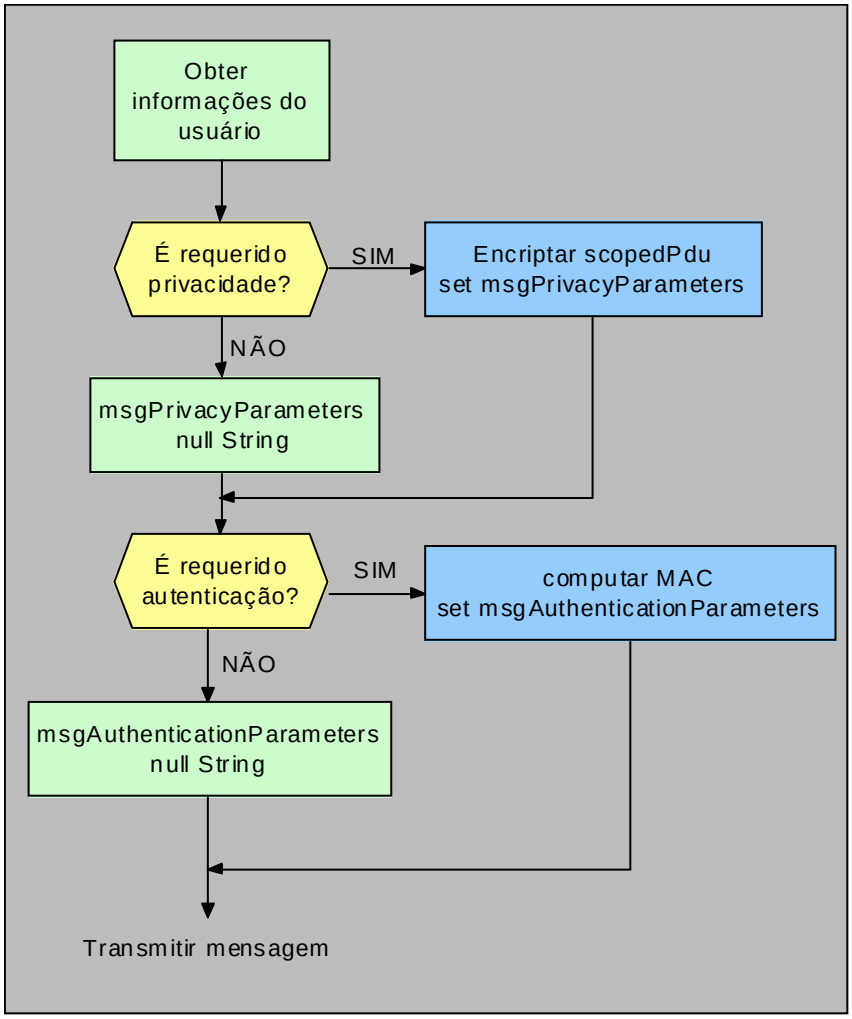
b) o USM determina se o *securityModel* requisitado é suportado por este usuário e, se não for, retorna uma indicação de erro;

c) se o *securityLevel* requisitar privacidade, então o valor *scopedPdu* é criptografado usando o CBC-DES e a chave privada de criptografia localizada na *privKey* deste usuário. O texto cifrado gerado é armazenado no campo *scopedPdu*, e o valor-sal derivado do valor da *privKey* é armazenado no campo *msgPrivacyParameters* da mensagem SNMPv3. Se a privacidade não for requisitada, então o campo *msgPrivacyParameters* é configurado para *NULL*;

d) o parâmetro *snmpEngineID* é armazenado no campo *msgAuthoritativeEngineID*;

e) o parâmetro *securityName* é armazenado no campo *msgUserName*;

FIGURA 17 – Processamento de mensagens USM: transmissão



FONTE: Adaptado de Stallings (2001, pág. 512).

f) se o *securityLevel* requisitar autenticação, o valor corrente do *snmpEngineBoots* e *snmpEngineTime* correspondentes ao parâmetro *snmpEngineID* são armazenados nos campos *msgAuthoritativeEngineBoots* e *msgAuthoritativeEngineTime*, respectivamente. Então o código de autenticação da mensagem é calculado sobre toda a mensagem, usando o HMAC e a chave de autenticação localizada na *authKey* deste usuário é armazenada no *msgAuthenticationParameters*. Se o *securityLevel* não requisitar autenticação, um valor zero é armazenado nos campos *msgAuthoritativeEngineBoots* e *msgAuthoritativeTime* e o campo *msgAuthenticationParameters* é configurado para *NULL*;

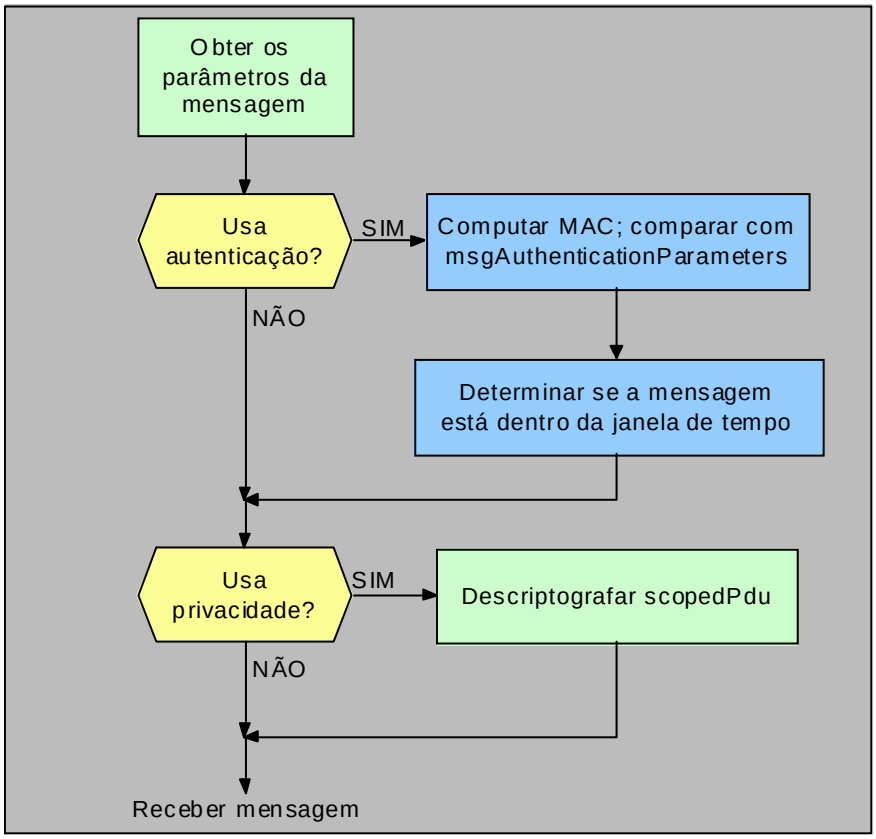
g) a mensagem completa com seu tamanho é retornada ao módulo de processamento de mensagens que fez a requisição.

Algum tempo após a transmissão da mensagem de requisição por uma entidade SNMP, uma mensagem de resposta deve ser recebida. Esta mensagem é tratada pelo despachante e pelo processador de mensagens. O processador de mensagens invoca o USM com a primitiva *processIncomingMessage*. Este processamento (fig. 18) inclui os seguintes passos:

a) os valores dos parâmetros de segurança são extraídos do campo *securityParameters*;
b) se o valor do *msgAuthoritativeEngineID* no *securityParameters* for desconhecido, e este motor SNMP é capaz de realizar descobertas, ele pode opcionalmente criar uma nova entrada na sua MIB *usmUserGroup*. Caso contrário, uma indicação de erro é retornada;
c) o USM determina se há uma entrada na *usmUserTable* para o *securityEngineID* remoto autorizado e o *securityName* local. Se não houver, uma indicação de erro é retornada;
d) o USM determina se o *securityLevel* requisitado é suportado para este usuário e, se não for, uma indicação de erro é retornada;
e) se o *securityLevel* especificar que a mensagem é para ser autenticada, então a mensagem é autenticada, usando o HMAC e a chave privada de autenticação localizada na *authKey* deste usuário, através do cálculo do código de autenticação da mensagem para esta mensagem e comparando o resultado com o do campo *msgAuthenticationParameters* na mensagem. Se os resultados não combinarem, o USM pára o processamento e retorna uma indicação de erro. Se o resultado conferir, a mensagem é declarada autêntica e o processamento continua;
f) o USM realiza a sincronização;
g) o USM realiza a checagem do tempo. Se a mensagem não estiver na janela de tempo, o USM pára o processamento e retorna uma indicação de erro. Se a mensagem estiver dentro da janela de tempo o processamento continua;
h) se o *securityLevel* indicar que foi usada criptografia, então a *scopedPdu* é descriptografada usando o CBC-DES e a chave privada de criptografia localizada na *privKey* deste usuário;
i) a *scopedPdu* é retornada ao módulo de processamento de mensagens que fez a requisição.

Muitos dos passos citados anteriormente são os mesmos utilizados por uma aplicação processadora de comandos, mas há algumas diferenças importantes. A seguir são exibidos os passos realizados pelo USM quando este é invocado com uma primitiva *processIncomingMessage* vinda do subsistema de processamento de mensagens.

FIGURA 18 - – Processamento de mensagens USM: recepção



FONTE: Adaptado de Stallings (2001, pág. 512).

Esse processamento é feito enquanto o processador de mensagens está tratando uma mensagem *Request* que acaba de chegar. Os passos são os seguintes:

a) os valores dos parâmetros de segurança são extraídos do *securityParameters*. Um bloco *cachedSecurityData* é preparado para servir de cache para as seguintes informações:
   - *MsgUserName*;
   - *SecurityEngineID*;
   - *SecurityLevel*.

b) se o valor do *msgAuthoritativeEngineID* no *securityParameters* for desconhecido, uma indicação de erro é retornada;

c) o USM determina se há uma entrada no *usmSecurityTable* para o *securityEngineID* local autorizado e o *securityName* remoto. Se não houver, uma indicação de erro é retornada;

d) o USM determina se o *securityLevel* requisitado é suportado para este usuário, e se não for, uma indicação de erro é retornada;

e) se o *securityLevel* especifica que esta mensagem deve ser autenticada, então a mensagem é autenticada usando o HMAC e a chave privada de autenticação localizada na *authKey* deste usuário, através do cálculo do código de autenticação da mensagem e comparando o resultado obtido ao resultado do campo *msgAuthenticationParameters* da mensagem. Se eles não combinarem, o USM pára o processamento e uma indicação de erro é retornada. Se eles combinarem, a mensagem é declarada autêntica e o processamento continua;

f) o USM realiza a checagem do tempo. Se a mensagem não estiver no tempo da janela, o USM pára o processamento e retorna uma indicação de erro. Se a mensagem estiver dentro da janela de tempo o processamento continua;

g) se o *securityLevel* indicar que a criptografia foi utilizada, então a *scopedPdu* é descriptografada usando o CBC-DES e a chave privada de criptografia localizada na *privKey* deste usuário;

h) as seguintes informações são adicionadas ao bloco *cachedSecurityData* preparados no passo *a*:

- *usmUserAuthProtocol*;
- *usmUserAuthKey*;
- *usmUserPrivProtocol*;
- *usmUserPrivKey*;

i) uma referência ou ponteiro para este bloco em cache é colocado na saída do parâmetro *securityStateReference*;

j) a *scopedPdu* é retornada ao módulo de processamento de mensagens que realizou a requisição.

Após o retorno do USM para o processador de mensagens, o processador de mensagens subseqüentemente retorna a PDU recuperada para a aplicação processadora de

comandos. Enquanto isso, o processador de mensagens armazena em cache o valor do *securityStateReference* que é recebido em retorno do USM, como parte do conjunto dos parâmetros do processador de comandos que este possui no cache. Subseqüentemente a isso, o processador de comandos pode preparar uma PDU *Response* e invocar então o processador de mensagens com uma primitiva *prepareResponseMsg*. O processador de mensagens irá então invocar o USM com a primitiva *generateResponseMsg*, que possui os mesmos parâmetros de entrada e saída do *generateRequestMsg* com uma exceção: *generateResponseMsg* inclui o *securityStateReference* como um parâmetro de entrada. O processador de mensagens obtém este valor do seu cache e passa o valor do *securityStateReference* para a requisição *Request* recebida que combine com o *Response* da saída.

O USM realiza os seguintes passos após o recebimento de uma *generateResponseMsg*:

a) usando o valor recebido de *securityStateReference*, o USM obtém as informações do usuário armazenando-as na cache. Estas informações provêm da mensagem *Request* processada anteriormente;
b) o USM determina se o s*ecurityLevel* requisitado é suportado para este usuário, e se não for, uma indicação de erro é retornada;
c) se o *securityLevel* requisitar privacidade então o valor da *scopedPdu* é criptografado usando o CBC-DES e a chave privada de criptografia localizada na *privKey* deste usuário.O texto cifrado resultante é armazenado no campo *scopedPdu*, e o valor-sal derivado do valor da *privKey* é armazenado no campo *msgPrivacyParameters* da mensagem SNMPv3. Se não for requisitado privacidade, então o campo *mspPrivacyParameters* é configurado para *NULL*;
d) o parâmetro *snmpEngineID* é armazenado no campo *msgAuthoritativeEngineID* da mensagem SNMPv3;
e) o parâmetro *securityName* é armazenado no campo *msgUserName*;
f) os valores correntes de *snmpEngineBoots* e do *snmpEngineTime* para este motor local autorizado são armazenados nos campos *msgAuthoritativeEngineBoots* e *msgAuthoritativeEngineTime*, respectivamente. Se o *securityLevel* requisitar autenticação, o código de autenticação da mensagem é calculado sobre toda a mensagem usando o HMAC e a chave privada de autenticação localizada na *authKey* deste usuário, através do cálculo do código de autenticação da mensagem

para esta mensagem e comparando o resultado com o do campo *msgAuthenticationParameters* na mensagem;

g) a mensagem completa com seu tamanho é retornada ao módulo de processamento de mensagens que solicitou a requisição.

O passo (f) mostra a diferença entre um motor autorizado e um não autorizado. No caso de um motor não autorizado, os valores dos campos *msgAuthoritativeEngineBoots* e *msgAuthoritativeEngineTime* do cabeçalho da mensagem são configurados somente se a mensagem deve ser autenticada por um receptor autorizado. Esta restrição faz sentido porque o receptor autorizado irá checar estes campos apenas se a mensagem deve ser autenticada. Entretanto, para uma mensagem *Response* vinda de um motor autorizado, os valores *msgAuthoritativeEngineBoots* e *msgAuthoritativeEngineTime* presentes no cabeçalho da mensagem são sempre configurados. No caso de um remetente autorizado, estes valores representam os valores locais “oficiais” do *snmpEngineBoots* e *snmpEngineTime*. Quando uma mensagem *Response* é recebida por um motor não autorizado, este deve usar estes valores apenas para a sincronização da mensagem se a mensagem for autenticada.

5.5.4 DESCOBERTA

O USM requer o uso do processo de descoberta para obter informações suficientes sobre outros motores SNMP para poder se comunicar com eles. Em particular, um motor SNMP não autorizado deve aprender o valor *snmpEngineID* de um motor autorizado antes que a comunicação seja realizada. Este processo é realizado em duas etapas.

Primeiro, o motor não autorizado envia uma mensagem *Request* ao motor autorizado ao qual deseja descobrir com um *securityLevel* requisitado de *noAuthnoPriv*. A mensagem inclui um campo *msgUserName* com o valor “*initial*” e um *msgAuthoritativeEngineID* nulo. A PDU *Request* anexada possui uma *varBindList* vazia. O motor autorizado irá responder com uma mensagem *Report* contendo seu *snmpEngineID* no campo *msgAuthoritativeEngineID*;

Segundo, se uma comunicação autenticada é requisitada, o motor não autorizado precisa estabelecer uma sincronização de tempo com o motor autorizado. Para fazer isso, o motor não autorizado envia uma mensagem *Request* autenticada com o campo

msgAuthoritativeEngineID configurado com o valor do *snmpEngineID* remoto recentemente aprendido e com os valores dos campos *msgAuthoritativeEngineBoots* e *msgAuthoritativeEngineTime* configurados com um valor de zero. A resposta a esta mensagem autenticada será uma mensagem *Report* do motor autorizado com seus valores correntes de *snmpEngineBoots* e *snmpEngineTime* presentes nos campos *msgAuthoritativeEngineBoots* e *msgAuthoritativeEngineTime*. A partir deste momento o motor não autorizado já pode se comunicar (se permitido) usando autenticação (e criptografia se necessário) com o motor autorizado.

5.6 GERENCIAMENTO DE CHAVES

Segundo Stallings (2001), um dos requisitos para o uso dos serviços de autenticação e privacidade no SNMPv3 é que, para qualquer comunicação entre um diretor em um motor não autorizado e um motor autorizado remoto, uma chave secreta de autenticação e uma chave secreta de privacidade devem ser compartilhadas.

Estas chaves permitem a um usuário de um motor não autorizado (tipicamente um sistema de gerenciamento) utilizar autenticação e privacidade com sistemas remotos autorizados que o usuário gerencia (tipicamente, um sistema agente). A RFC 2274 fornece um guia para a criação, atualização, e gerenciamento de chaves.

Para simplificar o gerenciamento de chaves centrado nos diretores, cada diretor é requisitado a manter apenas uma única chave para autenticação e uma única chave para privacidade. Estas chaves não são armazenadas em uma MIB e não são acessíveis via SNMP.

5.6.1 ALGORITMO DE TRANSFORMAÇÃO DE SENHA PARA CHAVE

A RFC 2274 define um algoritmo para mapeamento de uma senha comum para uma chave de 128 ou 160 bits. Resumidamente a geração das chaves é feita da seguinte maneira:

a) pega-se a senha do usuário como entrada e se produz uma string de 1.048.576 octetos através da repetição da senha tantas vezes o quanto for necessário, truncando o último valor se necessário, para formar uma *string digest0*;
b) se for desejado criar uma chave de 128 bits, pega-se o hash MD5 do *digest0* para formar *digest1*. A saída é a chave do usuário.

Uma das vantagens deste algoritmo é que ele reduz bastante a possibilidade de um ataque de dicionário ou “força bruta” e “*decouples*” nas chaves do usuário de qualquer NMS em particular. Nenhum NMS é necessário para armazenar as chaves do usuário. Ao invés disso, uma chave de usuário é gerada quando necessário a partir da senha do usuário.

Uma única senha pode ser usada para gerar uma única chave usada tanto para autenticação quanto para privacidade. Entretanto seria mais seguro usar duas senhas, uma para gerar uma chave de autenticação e outra para gerar uma chave de privacidade/criptografia.

5.6.2 LOCALIZAÇÃO DE CHAVES

Uma chave localizada é definida na RFC 2274 como uma chave secreta compartilhada entre um usuário e uma entidade SNMP autorizada. O objetivo é que assim o usuário precisa manter apenas uma única chave (ou duas se usar autenticação e privacidade) e, portanto lembrar de apenas uma senha (ou duas). O processo pelo qual uma única chave do usuário é convertida em múltiplas chaves únicas, uma para cada motor SNMP remoto, é chamado de localização de chave. Dentre os principais objetivos para o gerenciamento de chaves podemos citar:

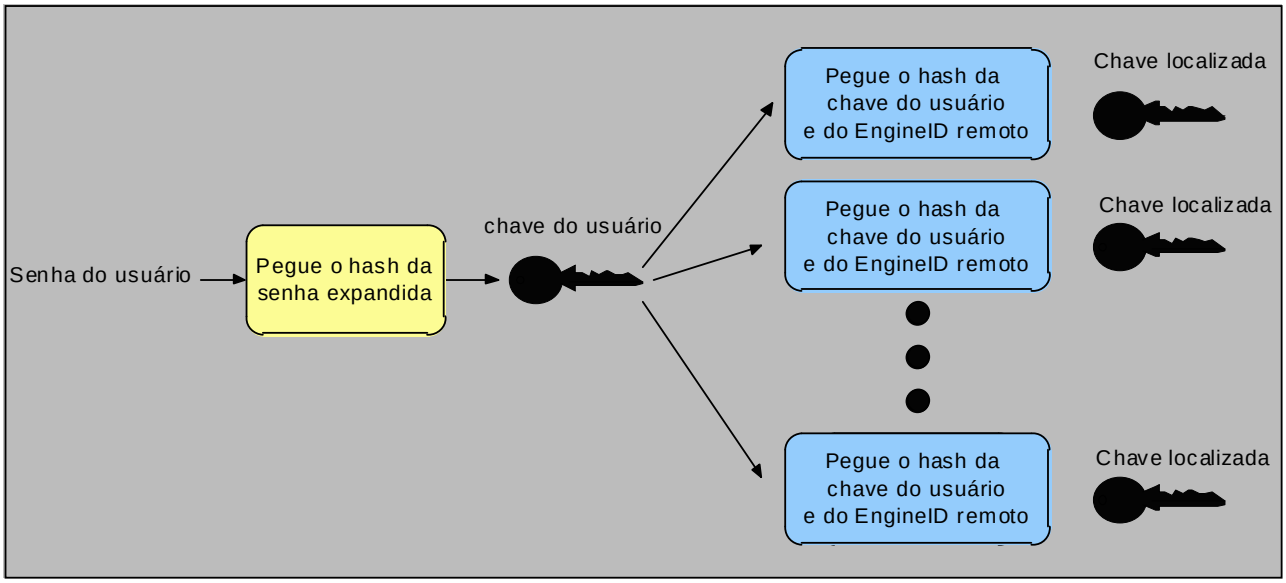
a) cada sistema agente SNMP em uma rede distribuída possui sua própria chave única para cada usuário autorizado a gerenciá-lo. Se múltiplos usuários estão autorizados como gerentes, o agente possui uma única chave de autenticação e uma única chave de criptografia para cada usuário. Por isso, se a chave de um usuário é comprometida, as chaves dos outros usuários não o são;
b) as chaves de um usuário são diferentes para agentes diferentes. Nesse caso, se um agente é comprometido, apenas as chaves de usuário para aquele agente são comprometidas e não as chaves de usuário em uso para os outros agentes.
c) o gerenciamento de rede pode ser realizado de qualquer ponto da rede, independente da disponibilidade de um sistema de gerenciamento de rede (NMS). Isto permite a um usuário realizar funções de gerenciamento a partir de qualquer estação gerenciada. Esta capacidade é fornecida pelo algoritmo de senha para chave.

É possível definir também alguns procedimentos a serem evitados:

a) um usuário precisa lembrar um grande número de chaves, número este que cresce com a adição de novos agentes;
b) um adversário que obtenha a chave de um agente é capaz de personificar qualquer outro agente para qualquer usuário, ou qualquer usuário para qualquer outro agente.

Para atender aos objetivos e considerações anteriores, uma única chave de usuário é mapeada pelo uso de uma função de mão única irreversível em diferentes chaves localizadas, para motores autenticados diferentes.

FIGURA 19 – Localização de chaves



FONTE: Adaptado de Stallings (2001, pág. 512).

O procedimento é o seguinte:

a) formar uma *string digest2* pela concatenação de *digest1*, o valor *snmpEngineID* do motor autorizado (agente) e *digest1*;
b) se uma chave de 128 bits for desejada, pega-se o *hash* MD5 do *digest2*. Se uma chave de 160 bits for desejada, pega-se o *hash* SHA-1 do *digest2*. A saída é a chave do usuário.

A chave localizada resultante pode então ser configurada no sistema agente de forma segura. Devido à natureza do MD5 e do SHA-1, é improvável que um adversário possa aprender uma chave de usuário, mesmo que o adversário venha a descobrir a chave localizada. A figura 19 resume bem esse processo.

5.6.3 ATUALIZAÇÃO DE CHAVES

O SNMPv3 assume que há algum meio seguro de entregar chaves localizadas para sistemas autenticados (agentes). Esta entrega segura está fora do escopo do SNMPv3; ela deve ser manual ou por outro protocolo seguro. Uma vez que uma chave inicial (ou par de chaves, no caso de autenticação e privacidade) tenha sido entregue a um agente, o SNMPv3 fornece um mecanismo para atualizar estas chaves de forma segura. Um usuário pode iniciar o processo de troca de chave através da requisição e fornecimento de uma nova senha. Alternativamente, o NMS pode iniciar o processo através da requisição de uma nova senha. Em ambos os casos a chave existente no NMS é atualizada. O NMS pode então calcular uma chave localizada para cada agente com o qual se comunica. O NMS deve então se comunicar com segurança com cada agente para que ele atualize sua chave localizada. Obviamente, o NMS não pode simplesmente enviar a chave em texto plano através da rede. Há duas possibilidades:

a) criptografar a nova chave usando a chave antiga como a chave de criptografia;
b) usar algum tipo de função de mão única para produzir um valor a partir da chave antiga. Deve-se então realizar um XOR deste valor com a nova chave e enviar o resultado ao agente. O agente pode então aplicar um XOR sobre a mensagem recebida usando a chave antiga para obter a nova chave.

A desvantagem da criptografia é a necessidade de usar criptografia mesmo em sistemas que suportam apenas autenticação de mensagens. Outra desvantagem da criptografia é as restrições feitas sobre a criptografia em vários países. Por isso o método adotado pelo SNMPv3 é uma variação do segundo método (XOR).

O método utilizado pelo SNMPv3 envolve o uso de objetos *KeyChange* localizados no *usmUserGroup*. Um diretor remoto ou um NMS configura este objeto, que é então automaticamente usado pelo agente para atualizar a chave correspondente. O algoritmo leva em consideração que um tamanho variável de chave pode ser usado por um algoritmo de autenticação ou criptografia em particular, o que complica o algoritmo de atualização de chaves.

O SNMPv3 permite que ambos, um administrador da rede ou um usuário possam atualizar uma chave de usuário em particular. Um administrador de rede pode configurar as

chaves iniciais para um usuário e pode requerer que o usuário troque as senhas de tempos em tempos, e cuidar da atualização de chaves quando isso acontecer. Além disso, o usuário pode trocar sua senha e chave(s) a qualquer momento. Para acomodar estes dois requisitos, o grupo *usmUSer* contém dois objetos de troca de chaves para cada uma das chaves. Para a chave de autenticação o *usmAuthKeyChange* deve ser configurado pelo administrador da rede para que a chave seja atualizada. O sistema agente é configurado de forma que apenas o administrador da rede tenha acesso a este objeto, permitido pelo subsistema de controle de acesso. O objeto *usmUserOwnAuthKeyChange* não é protegido por controle de acesso mas é definido de forma a permitir a atualização apenas se o requisitante possui o mesmo *userName* como objeto *usmUserName* para esta linha. Similarmente, o *usmUserPrivKeyChange* e *usmUserOwnPrivKeyChange* provêem capacidade de atualização para uma chave criptografada para o administrador da rede e outra para o usuário, respectivamente.

5.7 VIEW-BASED ACCESS CONTROL MODEL - VACM

Segundo Stallings (2001), o modelo de controle de acesso baseado em visão (VACM) definido na RFC 2275, possui duas características importantes:

a) o VACM determina se o acesso a um objeto gerenciado em uma MIB local por um diretor remoto deve ser permitido;
b) o VACM faz uso de uma MIB que define a política de controle de acesso para este agente, e torna possível o uso da configuração remota.

A RFC 2275 define cinco elementos que fazem parte do VACM:

a) grupos: um grupo é definido como um conjunto de zero ou mais tuplas <*securityModel*, *securityName*> nas quais os objetos de gerenciamento SNMP podem ser acessados. Um *securityName* se refere a um diretor, e direitos de acesso para todos os diretores em um dado grupo são idênticos. Um único *groupName* é associado com cada grupo. O conceito de grupo é uma ferramenta útil para categorizar os gerentes de acordo com seus direitos de acesso. Por exemplo, todos os gerentes de topo podem possuir um conjunto de direitos de acesso, enquanto que gerentes de nível intermediário podem ter um conjunto totalmente diferente de direitos de acesso. Qualquer combinação de *securityModel* e *securityName* pode pertencer no máximo a um grupo, ou seja, para um agente qualquer, um diretor

cujas comunicações são protegidas por um *securityModel* dado, pode somente ser incluído em um grupo;

b) nível de segurança: os direitos de acesso para um grupo podem ser diferentes dependendo do nível de segurança da mensagem que contém a requisição. Por exemplo, um agente pode permitir acesso somente de leitura para uma requisição feita por uma mensagem não autenticada, mas, pode requisitar autenticação para acesso de escrita;

c) contextos: um contexto MIB é um subconjunto nomeado de instâncias de objetos na MIB local. Os contextos fornecem uma forma útil de agregar objetos em coleções com diferentes políticas de acesso. O contexto é um conceito que se relaciona ao controle de acesso. Quando uma estação de gerenciamento interage com um agente para acessar informações de gerenciamento naquele agente, a iteração é entre o diretor de gerenciamento e o motor do agente SNMP, e os privilégios de controle de acesso são expressos em uma visão MIB que se aplica ao diretor e seu contexto. Os contextos possuem as seguintes características chave:
   - uma entidade SNMP, unicamente identificada por um *contextEngineID*, pode manter mais de um contexto;
   - um objeto ou uma instância de objeto pode aparecer em mais de um contexto;
   - quando múltiplos contextos existem, para identificar uma instância de objeto individual, seu *contextName* e *contextEngineID* deve ser identificado em relação ao seu tipo de objeto e sua instância.

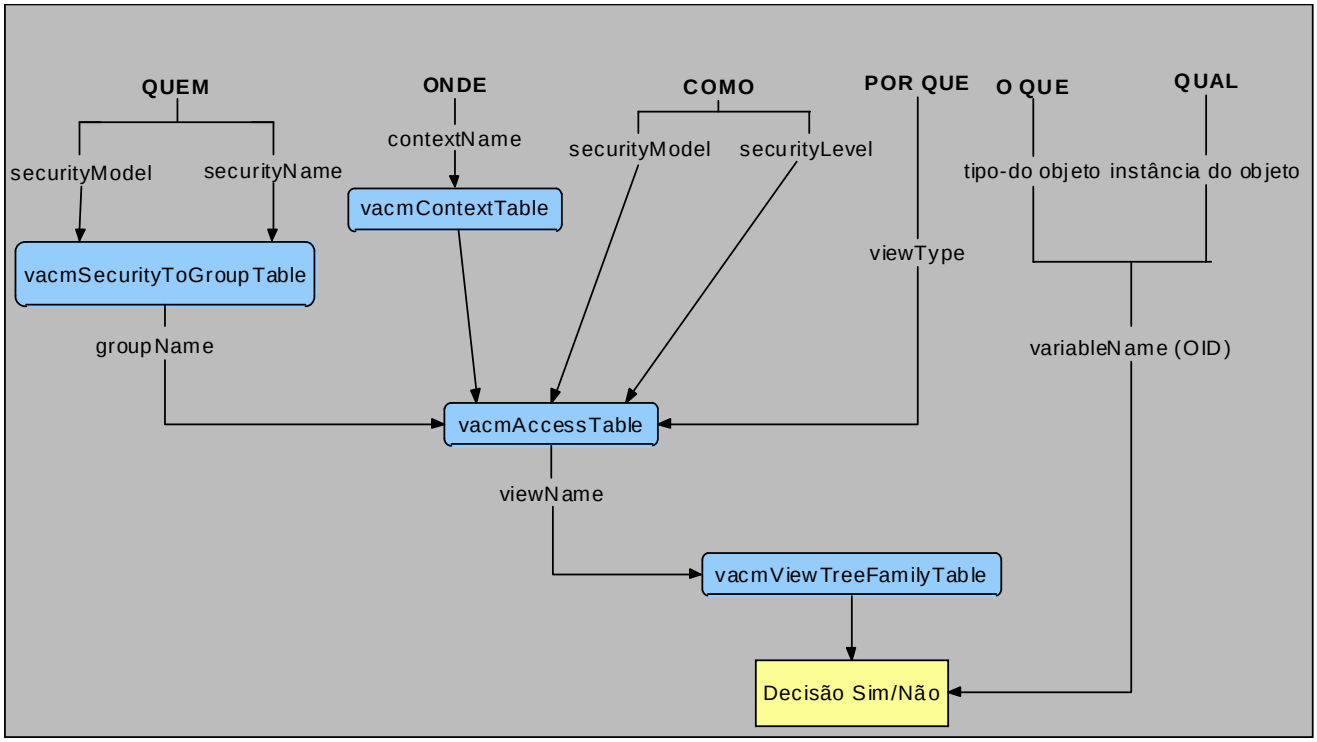
d) visões MIB: seria interessante restringir o acesso de um grupo particular a um subconjunto de objetos gerenciados em um agente. Para atingir este objetivo, o acesso a um contexto é feito através de uma visão MIB, que define um conjunto específico de objetos gerenciados. O VACM faz uso de uma técnica poderosa e flexível para definir visões MIB, baseadas no conceito de visões de subárvores e visões de famílias. A visão MIB é definida como uma coleção, ou família de subárvores, com cada subárvore sendo incluída ou excluída da visão. O SNMPv3 inclui o conceito de subárvore. Uma subárvore nada mais é do que um nó na hierarquia de nomeação da MIB mais todos os seus elementos subordinados. Mais formalmente uma subárvore pode ser definida como um conjunto de todos os objetos e instâncias de objetos que tenham um prefixo *OBJECT IDENTI*FIER

ASN.1 comum em seus nomes. Associados a cada entrada na *vacmAccessTable* estão três visões MIB, uma para leitura, uma para escrita e outra para notificação de acesso. Cada visão MIB consiste em um conjunto de visões de subárvores. Cada visão de subárvore na visão MIB, é especificada como sendo incluída ou excluída. Isto significa que, a visão MIB ou inclui, ou exclui todas as instâncias de objetos contidas na subárvore. Em acréscimo, uma máscara de visão é definida para reduzir a quantidade de informação de configuração requerida quando um controle de acesso mais refinado é requerido.

e) política de acesso: o VACM permite que um motor SNMP seja configurado para reforçar um conjunto particular de direitos de acesso. A determinação do acesso ou negação do mesmo depende dos seguintes fatores:
   - o diretor fazendo a requisição de acesso. O VACM torna possível para um agente permitir diferentes privilégios de acesso para diferentes usuários. Os diretores estão associados a grupos e a política de acesso é especificada em relação a estes grupos;
   - o nível de segurança pelo qual a requisição foi comunicada na mensagem SNMP. Tipicamente, um agente irá requerer o uso de autenticação para mensagens contendo uma requisição *set* (operação de escrita);
   - o modelo de segurança usado para processar a mensagem requisitada. Se múltiplos modelos de segurança estão implantados em um agente, o agente pode ser configurado para fornecer diferentes níveis de acesso, para requisições comunicadas via mensagens processadas por modelos de segurança diferentes;
   - o contexto da MIB usado para a requisição;
   - a instância do objeto específica para o qual acesso é requerido. Alguns objetos mantêm mais informações críticas ou sensitivas que outros, e, portanto a política de acesso deve depender na instância de objeto especifica requisitada;
   - o tipo de acesso requisitado (leitura, escrita, notificação). A leitura, escrita e notificação são operações de gerenciamento distintas, e diferentes políticas de controle de acesso podem ser aplicadas para cada uma destas operações.

O serviço fornecido pelo subsistema de controle de acesso é definido pela primitiva *isAccessAllowed*. A definição desta primitiva especifica que os parâmetros de entrada são *securityModel*, *securityName*, *securityLevel*, *viewType*, *contextName*, e *variableName*.

FIGURA 20 – Lógica VACM



FONTE: Adaptado de Stallings (2001, pág. 532).

Todos estes parâmetros são necessários para realizar a decisão sobre o controle de acesso. Colocando de outra maneira, o subsistema de controle de acesso é definido de forma a prover uma ferramenta muito flexível para configurar o controle de acesso no agente, através da divisão dos componentes de controle de acesso em seis variáveis separadas.

A figura 20 adaptada da figura existente na RFC 2275, provê uma forma útil de analisar as variáveis de entrada, e mostra como as várias tabelas na MIB VACM são usadas para realizar a decisão sobre o controle de acesso:

a) quem: a combinação do *securityModel* e do *securityName* define o “quem” da operação; identifica um dado diretor cujas comunicações são protegidas por um certo *securityModel*. Esta combinação pode pertencer a mais de um grupo neste motor SNMP. O *vacmSecurityToGroupTable* provê o *groupName*, dados o *securityModel* e o *securityName*;

b) onde: o *contextName* especifica “onde” o objeto gerenciado desejado pode ser encontrado. O *vacmContextTable* contém uma lista dos *contextName* reconhecidos;

c) como: a combinação do *securityModel* e do *securityLevel* define “como” a PDU *Inform* ou *request* foi protegida. A combinação de quem, onde, e como identificam entradas de zero ou um na *vacmAccessTable*;

d) por quê: o *viewType* especifica porque o acesso é requisitado: para uma operação de leitura, escrita ou notificação. A entrada selecionada na *vacmAccessTable* contém uma MIB *viewName* para cada um destes três tipos de operações, e o *viewType* é usado para selecionar um *viewName* específico. Este *viewName* seleciona a visão MIB apropriada do *vacmViewTreeFamilyTable*;

e) o que: a *variableName* é um identificador de objeto cujo prefixo identifica um tipo de objeto específico e cujo sufixo identifica uma instância de objeto específica. O tipo de objeto indica qual tipo de informação de gerenciamento é requisitada;

f) qual: a instância de objeto indica qual item específico de informação é requisitado.

Finalmente, a *variableName* é comparada com a visão MIB obtida. Se a *variableName* combina com um elemento incluído na visão MIB, então o acesso e fornecido.

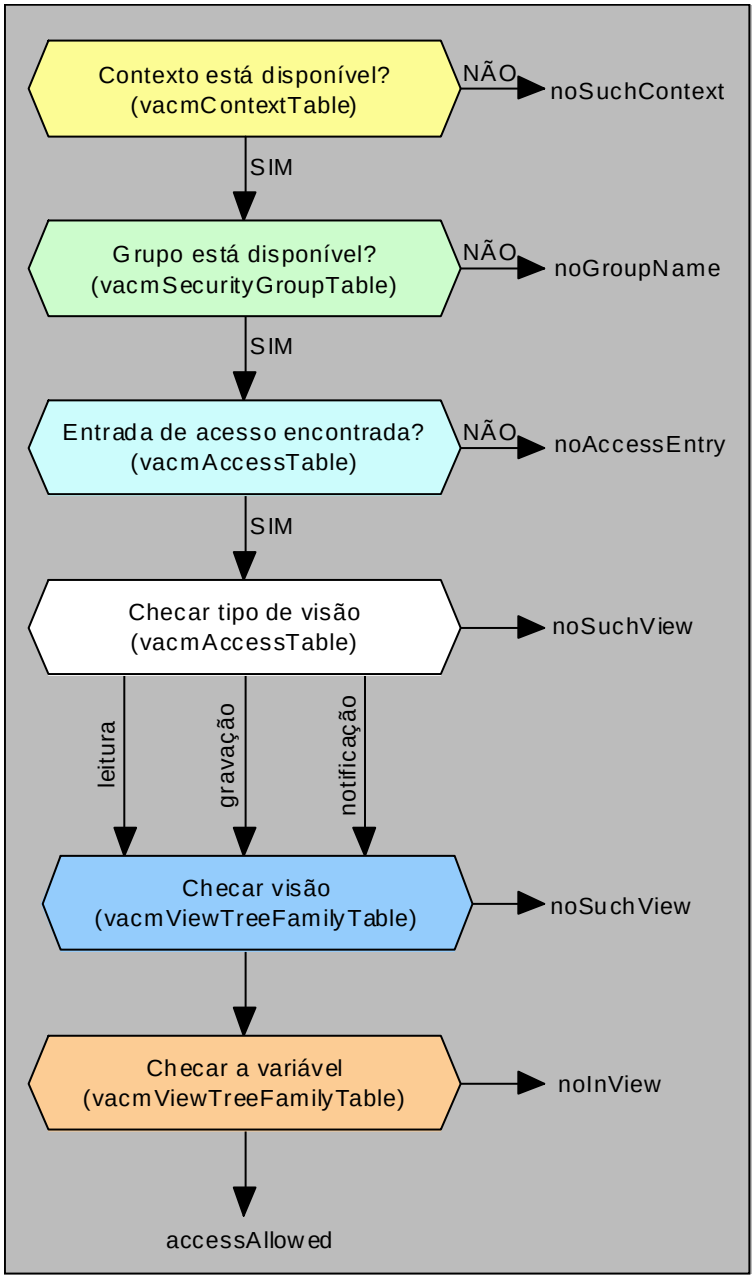
Agora é possível sumarizar os procedimentos usados pelo VACM para determinar se o acesso é permitido. Quando *isAccessAllowed* é invocada por uma aplicação, o VACM realiza os seguintes passos (fig. 21):

a) o VACM checa se há uma entrada no *vacmContextTable* para o *contextName*. Se houver, então este contexto é conhecido por este motor SNMP. Se não houver então um *errorIndication* de *noSuchContext* é retornado;

b) o VACM checa o *vacmSecurityToGroupTable* para determinar se há um grupo associado a este par <*secutityModel*, *securityName*>. Se houver, então este diretor operando sob este *securityModel* é um membro de um grupo configurado neste motor SNMP. Se não houver, então um *errorIndication* de *noGroupName* é retornado;

c) o VACM consulta em seguida o *vacmAccessTable* usando *groupName*, *contextName*, *securityModel*, e *securityLevel* como índices. Se uma entrada for encontrada, então uma política de controle de acesso foi definida para este *groupName*, operando sob este *securityModel*, neste *securityLevel*, para acessar este *contextName*. Se não for encontrada uma entrada, então um *errorIndication* de *noAccessEntry* é retornado;

FIGURA 21 – Fluxograma VACM



FONTE: Adaptado de Stallings (2001, pág. 534).

d) o VACM determina então se a entrada *vacmAccessTable* selecionada inclui uma referência a uma visão MIB de *viewType*. Se houver uma referência, então esta entrada contém um *viewName* para esta combinação de *groupName*, *contextName*, *securityModel*, *securityLevel*, e *viewType*. Se não houver uma referência, então um *errorIndication* de *noSuchView* é retornado;

e) o *viewName* é usado como um índice na *vacmViewTreeFamilyTable*. Se uma visão MIB for encontrada, então uma visão MIB foi configurada para este *viewName*. Caso contrário, um *errorIndication* de *noSuchView* é retornado;

f) o VACM verifica a *variableName* contra a visão MIB selecionada. Se esta variável está incluída na visão, então uma *statusInformation* de *accessAllowed* é retornada;

g) se esta variável não estiver incluída na visão, então um *errorIndication* de *noInView* é retornado.

5.7.1 A MIB VACM

A MIB VACM (fig. 22) contém informações nas seguintes categorias:

a) informações sobre contextos locais: a *vacmContextTable* lista os contextos localmente disponíveis por nome; esta informação é usada por aplicações geradoras de comandos para configurar a *vacmAccessTable* para controlar o acesso a todos os contextos na entidade SNMP. A *vacmContextTable* em si possui acesso de apenas leitura e não pode ser configurada por uma operação SNMP. Cada entrada na tabela é um nome "*human-readable*" que identifica um contexto em uma entidade SNMP em particular. Um contexto vazio, representa o contexto padrão. A *vacmContextTable* é independente da *vacmAccessTable*. A qualquer momento pode haver um contexto listado na *vacmContextTable* para o qual não há entradas na *vacmAccessTable* que referenciem este contexto, e pode haver entradas na *vacmAccessTable* que referenciem um contexto que ainda não exista no momento e que portanto não possua uma entrada atualmente na *vacmContextTable*;

b) informações sobre grupos: esta *vacmSecurityToGroupTable* provê um *groupName* dado um *securityModel* e um *securityName*. O *groupName* é usado para definir uma política de controle de acesso para um grupo de diretores sob um dado modelo de segurança. A tabela é indexada por *vacmSecurityModel* e *vacmSecurityName*, e o valor de *vacmGroupName* é usado como um índice na *vacmAccessTable*;

c) informações sobre direitos de acesso: a *vacmAccessTable* pode ser configurada para definir direitos de acesso para grupos. Para um dado grupo, os direitos de acesso podem ser definidos para um ou mais contextos, um ou mais modelos de segurança, e um ou mais níveis de segurança. Por isso, pode haver múltiplas

entradas na *vacmAccessTable* para um *groupName* em particular. Para deixar mais claro:

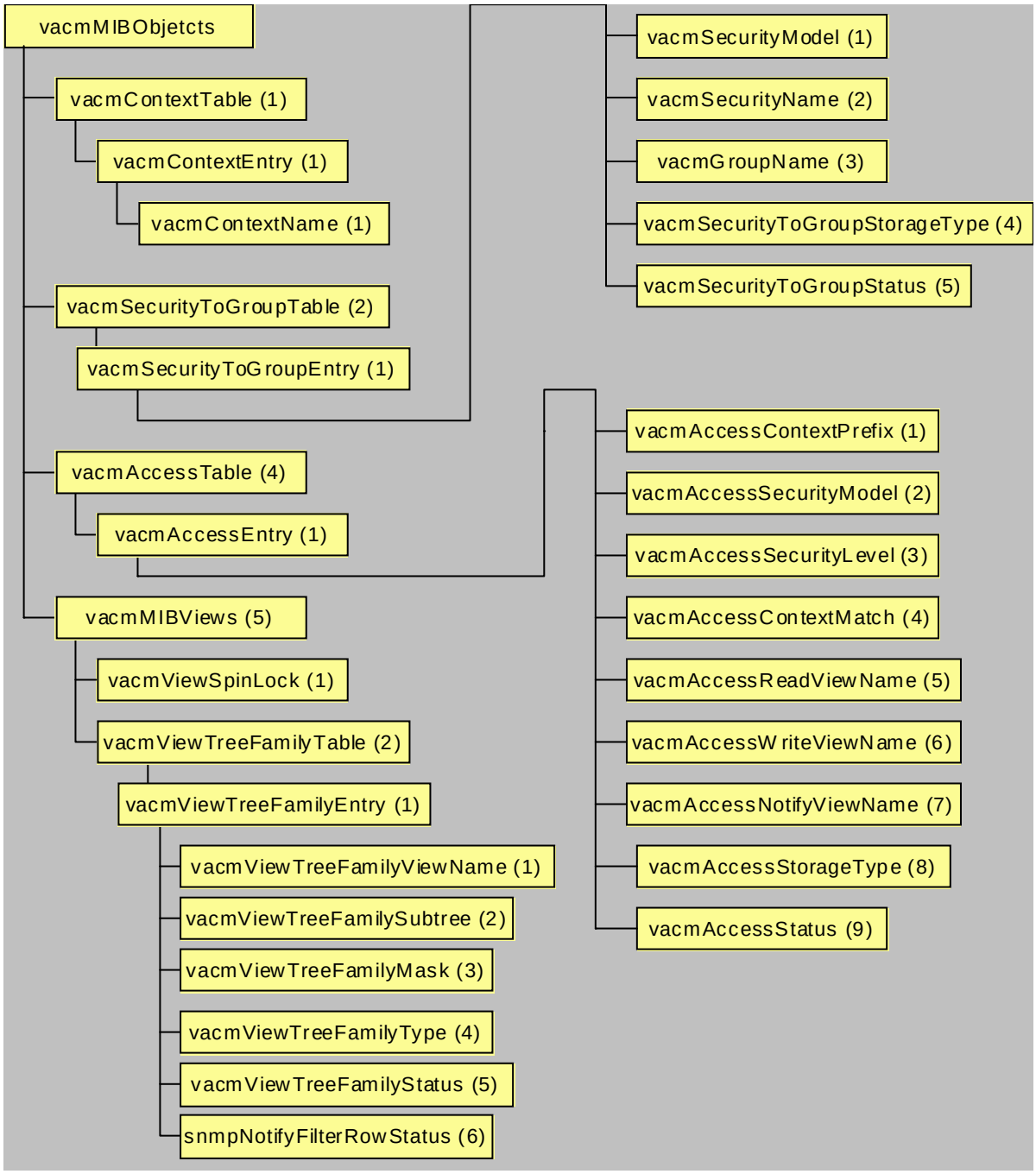
- um contexto se refere a um subconjunto de instâncias de objetos na MIB local. Para um grupo em particular, direitos de acesso podem ser definidos em mais de um contexto;
- se um agente suporta mais de um modelo de segurança, então os direitos de acesso para um grupo podem ser definidos separadamente em respeito a cada modelo de segurança. Em um dado modelo de segurança, os direitos de acesso de grupo podem ser enumerados para múltiplos contextos;
- os direitos de acesso de um grupo para um contexto em particular sob um modelo de segurança em particular podem depender no nível de segurança da troca. Por isso, um grupo pode ter maior direito de acesso para um contexto em particular sob um modelo de segurança em particular se a autenticação é utilizada e comparada, do que quando a autenticação não é utilizada.

Além do mais, para um dado grupo, contexto, modelo de segurança, e nível de segurança, o escopo da informação que pode ser acessada pode diferir para leitura, escrita, ou notificação;

d) informações sobre visões MIB: a estrutura *vacmMIBView* consiste de um único objeto escalar: *vacmViewSpinLock,* e uma tabela: *vacmViewTreeFamilyTable*. O objeto *vacmViewSpinLock*, habilita as aplicações geradoras de comandos a coordenarem seus usos da operação *Set* através da criação e modificação de visões. Pode haver múltiplas subárvores associadas com um dado *viewName,* caso no qual a visão MIB para este *viewName* consiste da união de todas estas subárvores. Cada entrada na *vacmViewTreeFamilyTable* define uma família de subárvores MIB, usando a subárvore, a máscara, e o tipo de objetos da coluna. Esta técnica é usada também no filtro de notificações. Em essência, uma subárvore consiste de um objeto MIB mais todos os objetos que estão subordinados a ela na hierarquia de nomeação. É possível excluir certas partes desta subárvore de forma que ela permaneça não como uma subárvore completa, mas apenas como uma subárvore parcial, consistindo de um objeto MIB e alguns de seus objetos subordinados. Esta é uma família de subárvores. Então se múltiplas entradas forem agrupadas, é

possível definir uma coleção arbitrária de objetos MIB para serem usados para um propósito em particular.

FIGURA 22 – A MIB VACM



FONTE: Adaptado de Stallings (2001, pág. 536).

Segundo Stallings (2001), a RFC 2275 sugere que uma configuração inicial da MIB VACM seja feita durante a instalação de um novo motor SNMP que suporte o VACM. A RFC 2275 define três possíveis configurações iniciais:

a) configuração inicial sem acesso;
b) configuração inicial semi-segura;
c) configuração inicial com segurança mínima.

Para a configuração inicial sem acesso, não há nenhuma configuração inicial e nenhum acesso às variáveis MIBs locais é permitido durante a instalação do sistema. Nos outros dois casos há uma diferença apenas nas entradas da *vacmViewTreeFamilyTable*. As demais entradas: *vacmContextTable*, *vacmSecurityToGroupTable*, e *vacmAccessTable* são iguais para estas duas configurações.

O *vacmContextTable* é configurado inicialmente com uma única entrada que possui o *contextName* de uma *string* vazia. Por convenção, este é o contexto padrão. Similarmente, a *vacmSecurityToGroupTable* é configurada inicialmente com uma entrada apenas. Esta entrada especifica o "USM" como o *securityModel*, um *securityName* com "*initial*" e um *groupName* com "*initial*". Privilégios de acesso serão configurados para este *groupName*. Subseqüentemente, se o gerente de configuração para este sistema desejar expandir o uso desta configuração semi-segura, este pode adicionar mais *securityNames* a este *groupName*. Não será necessário fazer outras alterações.

Tendo definido um contexto e um grupo, é preciso definir as visões MIB deste contexto para o qual este grupo tem acesso. As requisições podem chegar neste grupo com três níveis de proteção de mensagem: sem proteção, somente autenticação, e autenticação mais privacidade. A *vacmAccessTable* é configurada para este *groupName*, *contextName*, e *securityModel* como segue:

a) para comunicação sem proteção (*noAuthNoPriv*), o acesso a uma visão MIB restrita é permitido para operações de leitura e notificação, e nenhum acesso é permitido para operações de escrita;
b) para comunicações protegidas tanto por autenticação quanto por autenticação mais privacidade, é permitido o acesso à visão MIB "*internet*", que se refere a subárvore "*internet*" inteira da MIB-2.

Tudo o que ainda resta é definir as visões MIB que correspondem aos *viewName* "*internet*" e "*restricted*". Isto é feito na *vacmViewTreeFamilyTable*. Há uma entrada para "*internet*". Esta entrada possui um valor de subárvore 1.3.6.1, que é o identificador para o objeto *internet* MIB-2. Por isso, podemos fazer a seguinte afirmação: para todos os diretores que são membros do grupo "*initial*", que comunicam uma requisição usando autenticação ou autenticação e privacidade usando o *securityModel* "USM", acesso completo a subárvore "*internet*" sob o *contextName* "" é garantido para operações de leitura, escrita e notificações. Durante a instalação inicial, o único diretor que é membro deste grupo é o diretor com o *securityName* "*initial*".

As cinco entradas restantes da *vacmViewTreeFamilyTable* possuem um valor *vacmViewTreeFamilyViewName* com "*restricted*". Entretanto, quando esta tabela é indexada usando um *viewName* com "*restricted*", a visão MIB é definida pela coleção destas cinco entradas. Cada uma destas cinco entradas tem um tipo *included*, e cada uma inclui uma subárvore diferente. Por isso, a visão MIB, definida por estas cinco entradas, consistem da união das seguintes subárvores da MIB-2:

a) o grupo *system* da MIB-2 (1.3.6.1.2.1.1);
b) o grupo *snmp* da MIB-2 (1.3.6.1.2.1.11);
c) o *snmpEngine* (1.3.6.1.6.3.7.2.1) definido na RFC 2271;
d) o *snmpMPDStats* (1.3.6.1.6.3.8.2.1) definido na RFC 2272;
e) o *usmStats* (1.3.6.1.6.3.9.2.1) definido na RFC 2274.

Com isso, é possível fazer a seguinte afirmação: para todos os diretores que são membros do grupo "*initial*", que comunicam uma requisição sem autenticação usando o *securityModel* "USM", o acesso "*restricted*" ao conjunto de subárvores sob o *contextName* "" é permitido para operações de leitura e notificação. Em adição, acesso ao conjunto de subárvores do grupo "*internet*" é garantido para operações de leitura, escrita, e notificação para uma requisição comunicada com autenticação. No momento da instalação, o único diretor que é membro deste grupo é o diretor com *securityName* "*initial*".

5.8 A API ADVENTNET SNMPV3

Segundo a AdventNet (1995), a API SNMP AdventNet é o ambiente de desenvolvimento mais abrangente para construir aplicações e *applets* de gerenciamento

baseados na estrutura SNMP. Esta API consiste em pacotes Java que podem ser usados para desenvolver soluções e produtos para o gerenciamento de redes baseados em Java e na *Web*.

As principais características da API AdventNet SNMPv3 são:

a) comunicação SNMP para os protocolos SNMPv1, SNMPv2c e SNMPv3;
d) segurança, conforme definido pelos modelos USM e VACM;
e) suporte MIB para os formatos SMIv1 e SMIv2;
f) componentes Java Beans SNMP que aumentam a funcionalidade da API para uso em ferramentas de desenvolvimento visual em Java;
g) suporte a banco de dados para realizar o carregamento de MIBs;
h) suporte para armazenamento de informações de usuários SNMPv3 em banco de dados;
i) SNMP *Applet Server* (SAS) para facilitar a comunicação entre *applets* e dispositivos gerenciados;
j) suporte ao tunelamento (*tunneling*) HTTP para que os *applets* se comuniquem sobre uma rede com restrições de *firewall*;
k) suporte aos *Enterprise* Java Beans (EJB) para o desenvolvimento de aplicações de gerenciamento multicamadas escaláveis;
l) acesso a API SNMP via *Remote Method Invocation* (RMI) e *Common Object Request Broker Architecture* (CORBA) para fornecimento de suporte a computação distribuída;
m) ferramenta MibBrowser poderosa, que pode ser usada tanto como uma aplicação como um *applet*;
n) suporte a internacionalização para os componentes da API e da *User Interface* (UI);
o) ferramentas de linha de comando como *snmpget*, *snmpgetnext*, *snmpset*, *sendtrap*, etc.

Dentre os motivos para uso da API AdventNet para construir aplicações de gerenciamento pode-se citar:

a) suporte multiplataforma: suporta todas as maiores plataformas atuais como Solaris, WindowsNT e Linux com um código base comum para ambas as plataformas;

p) padrões abertos: construída sobre padrões abertos e seguindo as tecnologias padrão Internet correntes como Java Beans, HTTP, RMI, CORBA, EJB, etc. O benefício chave destas tecnologias é a rapidez entre desenvolvimento-venda e a facilidade no desenvolvimento de soluções personalizadas para o gerenciamento de redes;

q) customização (c*ustomization*): fornece uma ampla gama de interfaces Java para o desenvolvimento e entrega de soluções altamente personalizadas;

r) escalabilidade: fundamentalmente designada como um sistema multicamadas, baseada em tecnologias Internet amplamente usadas, para o desenvolvimento de soluções escaláveis;

s) flexibilidade: uma hierarquia de bibliotecas em pacotes Java é fornecida, permitindo a seleção do nível de suporte via bibliotecas desejado. Assim, é possível acessar informações detalhadas sobre o SNMP usando a API de baixo nível, ou usar Java Beans de alto nível para simplificar a programação, além de funcionalidades adicionais que fornecem a possibilidade de desenvolvimento sem a necessidade de negociar com os detalhes do SNMP;

t) gerenciamento de redes baseadas na *Web*: através de suporte ao HTTP e applets. A API SNMP AdventNet inclui módulos como SAS e HTTP que permitem aos usuários gerenciarem sua rede via Internet, ou até mesmo através de dispositivos sob *firewalls*. As APIs SAS podem ser estendidas para adicionar suporte a SSL;

u) conformidade com padrões: a API SNMP da AdventNet está em conformidade com as seguintes especificações Internet:
   - SNMPv1 - RFC 1155 e RFC 1157;
   - SNMPv2c - RFC 1901 e RFC 1907;
   - SNMPv3 - RFC 2571 e RFC 2572;
   - SNMPv3 - RFC 2574 (USM);
   - SNMPv3 - RFC 2575 (VACM);
   - co-existência entre SNMPv1, SNMPv2c, SNMPv3 - RFC 2576;
   - filtro de notificações e *proxy forwarding* - RFC 2573.

Para o desenvolvimento do protótipo proposto no presente trabalho serão utilizadas as facilidades existentes na API de alto nível.

6 DESENVOLVIMENTO DO PROTÓTIPO

A seguir serão apresentadas a especificação e a implementação do protótipo, bem como as técnicas, metodologias e ferramentas utilizadas no seu desenvolvimento.

6.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O sistema a ser desenvolvido deverá possuir essencialmente dois elementos:

a) um agente que irá fornecer informações de desempenho referentes ao elemento de rede no qual está instalado, mais especificamente, informações referentes ao subgrupo IP da Mib-2 deste equipamento e;
b) um gerente que fará a exposição gráfica dos resultados obtidos deste agente, permitindo desta forma, que o administrador da rede possa visualizar os dados do equipamento em tempo real.

A partir destes dados o administrador da rede pode observar o desempenho na transmissão e recepção de pacotes IP no equipamento analisado, bem como dados sobre pacotes com erros, desfragmentados, etc.

Mais importante, entretanto, são as primitivas de segurança utilizadas na comunicação entre o agente e o gerente. O agente deve fornecer informações apenas aos gerentes que estejam registrados e habilitados a obterem estas informações, obedecendo à sua política de controle de acesso interna definida e configurada pelo administrador da rede, evitando assim que estas informações sejam acessadas e/ou adulteradas por usuários não autorizados. Além disso, os dados transmitidos entre o agente e o gerente autorizado devem ser protegidos usando funções de criptografia, para evitar que um usuário malicioso tenha acesso às informações trocadas entre ambos.

Para atingir estes objetivos a comunicação entre o agente e o gerente será realizada obedecendo às novas especificações do SNMPv3, que incluem principalmente a autenticação e a privacidade na comunicação entre o agente e o gerente através do uso da criptografia, da autenticação via HMAC, do uso das funções de sincronização e de políticas de controle de acesso.

6.2 ESPECIFICAÇÃO

A técnica utilizada para o desenvolvimento da especificação do protótipo foi a orientação a objetos através da *Unifield Modeling Language* – UML. Para obter maiores informações sobre a orientação a objetos usando a UML consulte Booch (2000) e Furlan (1998).

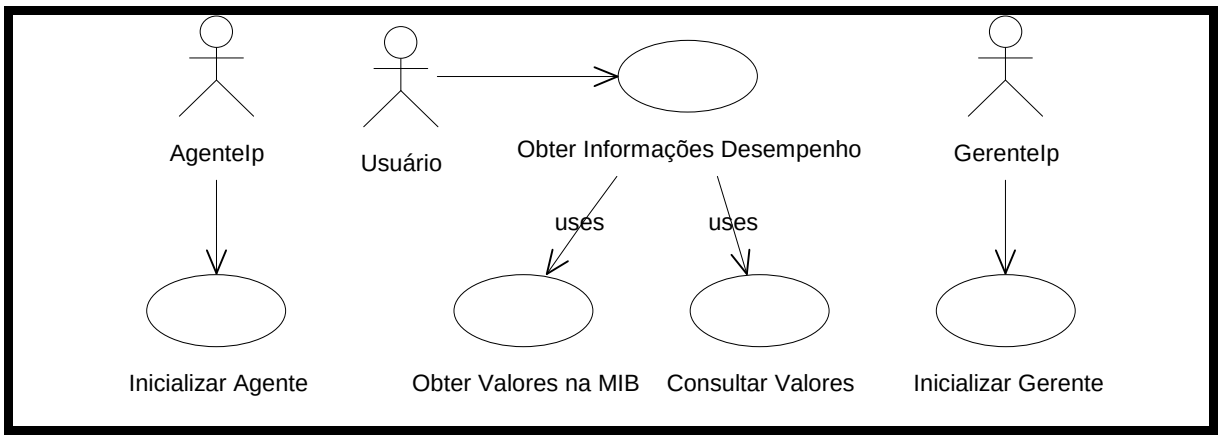
A ferramenta utilizada para a especificação do protótipo foi a versão *Student* do software Rational Rose. Para obter maiores informações sobre esta ferramenta consulte Rational (1995).

6.2.1 DIAGRAMAS DE CASOS DE USO

Conforme a notação da UML, e utilizando a ferramenta Rational Rose *Student*, foi desenvolvido o diagrama de casos de uso conforme a figura 23, onde se destacam os casos de uso principais do sistema:

a) obter informações de desempenho: o usuário acessa a aplicação gerente e solicita determinadas informações relacionadas ao desempenho. O gerente entra em contato com o agente e este após verificar que o pedido é válido, realiza a consulta às variáveis MIB repassando seus valores ao gerente que os exibe em um gráfico;
b) inicializar gerente: conjunto de operações necessárias para o funcionamento do gerente;
c) inicializar agente: conjunto de operações necessárias para o funcionamento do agente.

Figura 23 - Diagrama de casos de uso



6.2.2 DIAGRAMA DE CLASSES

Inicialmente, foram identificadas as principais classes do sistema, seus atributos, relacionamentos e operações, conforme figuras 24 e 25.

Foram definidas as classes AgenteIp, IpRequestHandler, IpInstrument, IpStat e GerenteDesempenhoIp para atender aos requisitos do protótipo.

Figura 24 - Diagrama de Classes: parte agente

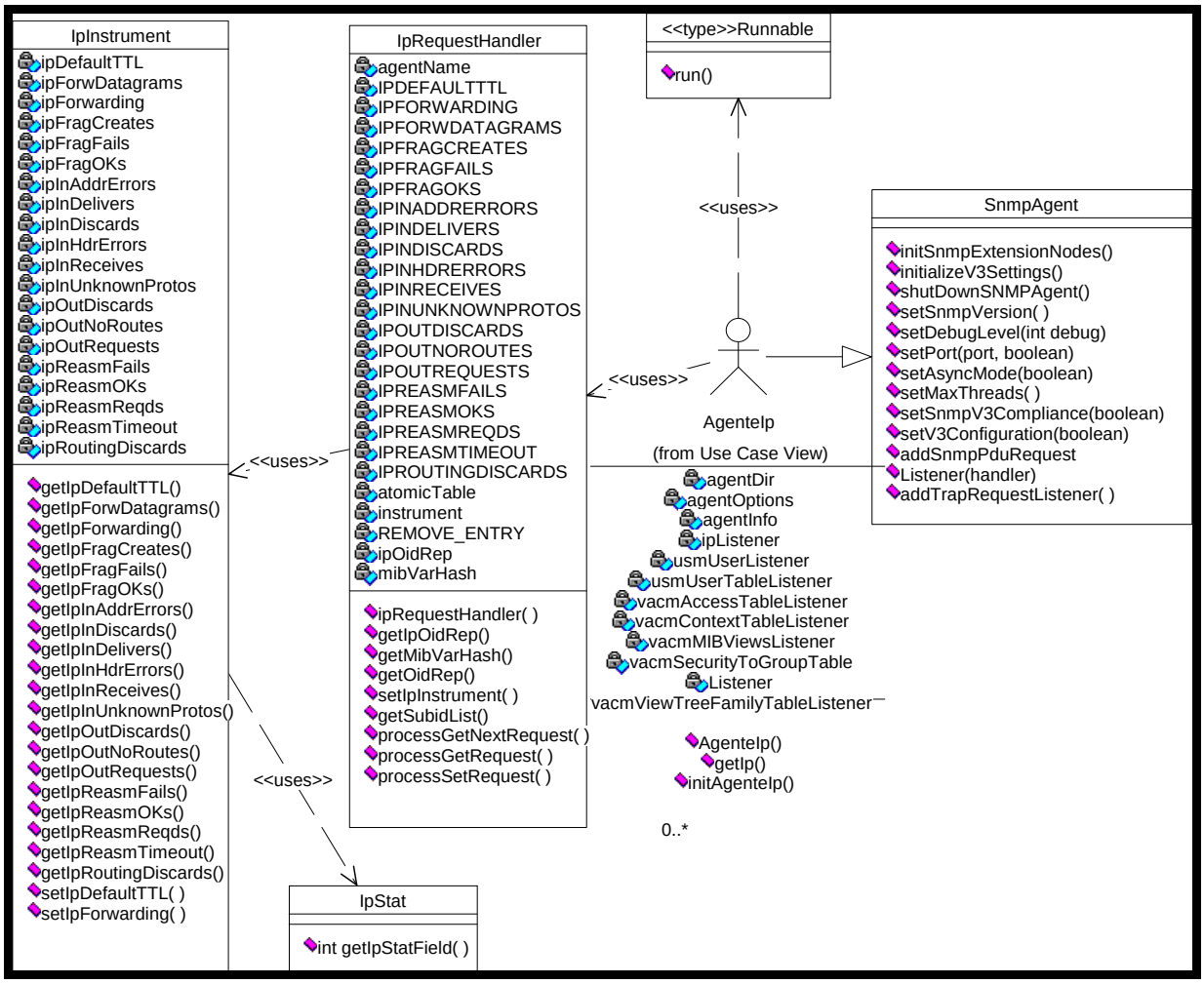
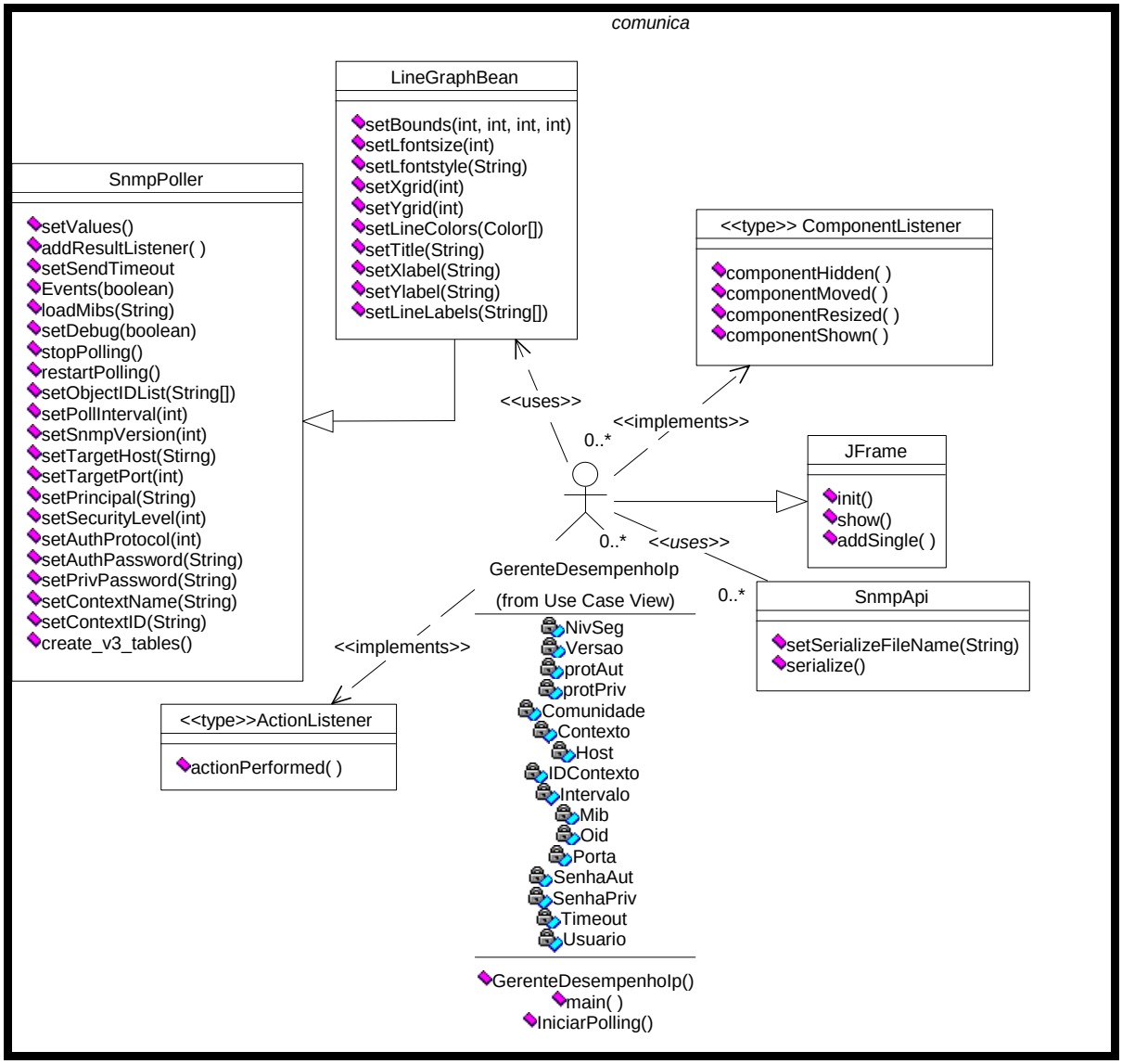


Figura 25 - Diagrama de Classes: parte gerente



A classe AgenteIp define métodos para acessar os dados pelos quais é responsável e para processar as mensagens SNMPv3, estando diretamente relacionada às seguintes classes:

a) SnmpAgent: classe herdada da API SNMPv3 da AdventNet que contém todos os métodos utilizados pela classe AgenteIp relacionados às operações SNMP;
b) Runnable: interface padrão da API Java que define os métodos utilizados pela classe AgenteIp para executar o agente no *prompt* de comando;
c) IpRequestHandler: classe que estende a classe SimpleRequestHandler da API SNMPv3 da AdventNet. Esta classe é responsável por direcionar as requisições

recebidas pelo agente às classes responsáveis pelo seu processamento e por processar automaticamente todas as mensagens SNMPv3 realizando a autenticação das mensagens e a criptografia/descriptografia dos dados quando necessário;

d) IpInstrument: classe responsável por definir e implementar os métodos utilizados para acessar e alterar as variáveis escalares MIB do grupo *ip* da Mib-2;
e) IpStat: classe nativa utilizada pela classe IpInstrument para acessar as informações diretamente do sistema operacional do equipamento no qual o agente está sendo executado.

A classe GerenteDesempenhoIp executa os métodos necessários para se conectar ao agente e para buscar e exibir os resultados em um gráfico. Assim como a classe AgenteIp, a classe GerenteDesempenhoIp faz uso de outras classes para realizar suas operações. Estas classes são as seguintes:

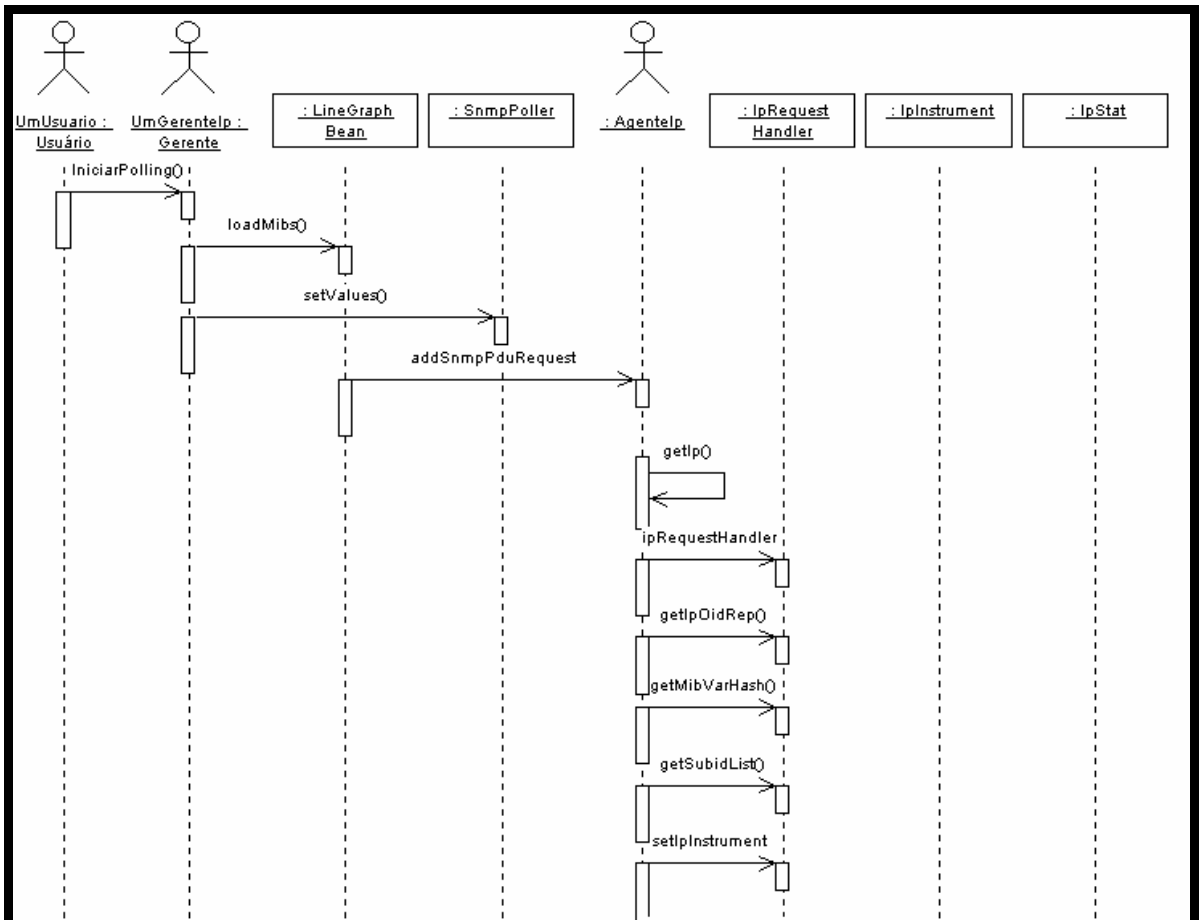
a) SnmpPoller: classe da API SNMPv3 da AdventNet que contém a definição de todos os métodos utilizados pela classe LineGraphBean para tratar as mensagens e parâmetros do SNMPv3, usados durante o *polling*, além de possuir métodos para realização da descoberta de novos agentes e de sincronização com estes agentes;
b) LineGraphBean: classe da API SNMPv3 da AdventNet responsável por realizar a exposição dos dados de *polling* na tela na forma de um gráfico de linhas;
c) SnmpApi: classe da API SNMP da AdventNet utilizada pela classe GerenteDesempenhoIp para armazenar os dados de descoberta e sincronização de agentes para uso posterior;
d) Jframe: classe padrão da API Java utilizada para exibir o gerente na tela;
e) ComponentListener: interface padrão da API Java implementada pela classe LineGraphBean para tratar os eventos desta classe;
f) ActionListener: interface padrão da API Java implementada pela classe LineGraphBean e utilizada para tratar os eventos de clique de *mouse* realizados nos botões de configuração e execução do gerente.

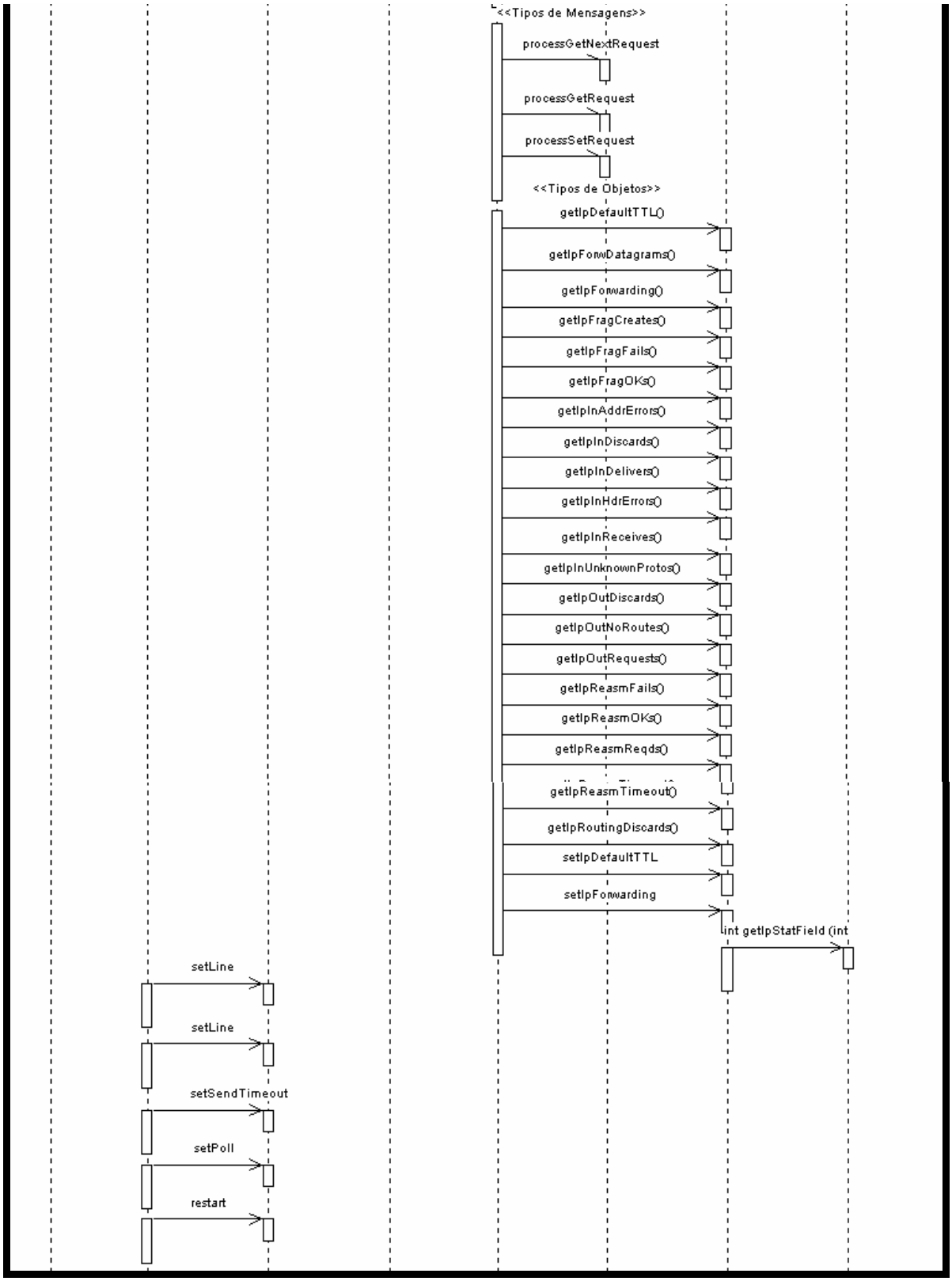
6.2.3 DIAGRAMAS DE SEQÜÊNCIA

Nos diagramas de seqüência pode-se observar a execução de cada caso de uso e verificar as mensagens trocadas entre os objetos (agente e gerente). Os principais diagramas de seqüência encontrados no sistema são:

a) obter informações de desempenho: o usuário utiliza a aplicação gerente para obter acesso às informações desejadas e utilizando a interface da aplicação gerente realiza todos os passos necessários para poder realizar esta operação. Primeiro o usuário fornece as informações que deseja analisar e os parâmetros SNMPv3, depois disso o gerente se encarrega de se conectar ao agente, solicitar as informações desejadas e, após obter as estas informações do agente, a aplicação gerente exibe os resultados em um gráfico na tela (fig. 26). Este caso de uso é composto de dois outros subcasos de uso:

Figura 26 - Obter informações de desempenho





- consultar valores: o gerente entra em contato com o agente especificado pelo usuário usando as opções de autenticação e segurança do SNMPv3 e, caso tenha permissão de acesso às variáveis solicitadas pelo usuário, o gerente após receber os

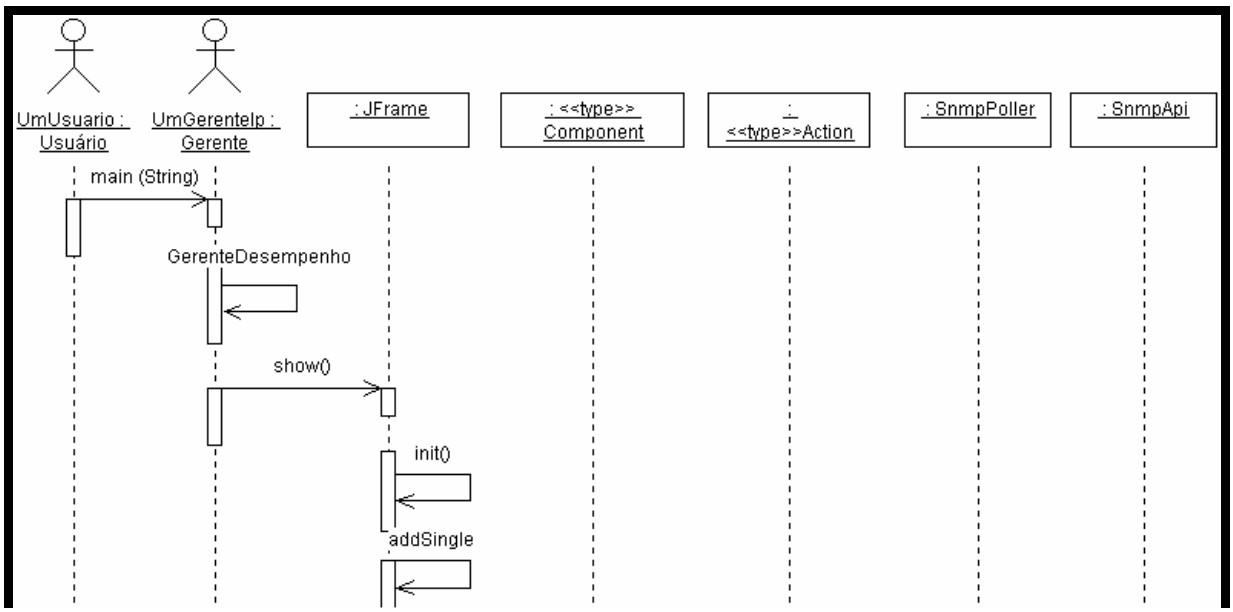
valores do agente os exibe no gráfico, caso contrário, exibe uma mensagem de erro informando a sua causa;

- obter valores na MIB: após verificar que o pedido é válido, ou seja, que o gerente tem permissão para acessar a(s) informação(ões) solicitada(s) e que a mensagem foi corretamente autenticada sem erros, o agente realiza a consulta às variáveis MIB obtendo seus valores e repassando-os ao gerente;

b) inicializar gerente: antes de poder solicitar qualquer tipo de informação de um determinado agente o gerente precisa inicialmente, se registrar com este agente para atender aos requisitos do SNMPv3. O gerente realiza esta operação fornecendo as informações de autenticação e privacidade e realizando um processo de descoberta (para obter o agente ao qual vai se registrar) e de sincronização (para poder usar autenticação e privacidade). Esta situação pode ser vista na figura 27;

c) inicializar agente: o agente deve realizar um processo de inicialização a fim de carregar todas as tabelas com as informações VACM e USM. Além disso deve definir a porta, o *host*, o protocolo usado, as MIBs que utilizará, enfim, todas as operações necessárias para que ele possa atuar como um agente no equipamento onde foi instalado (fig. 28).

Figura 27 - Inicialização do gerente



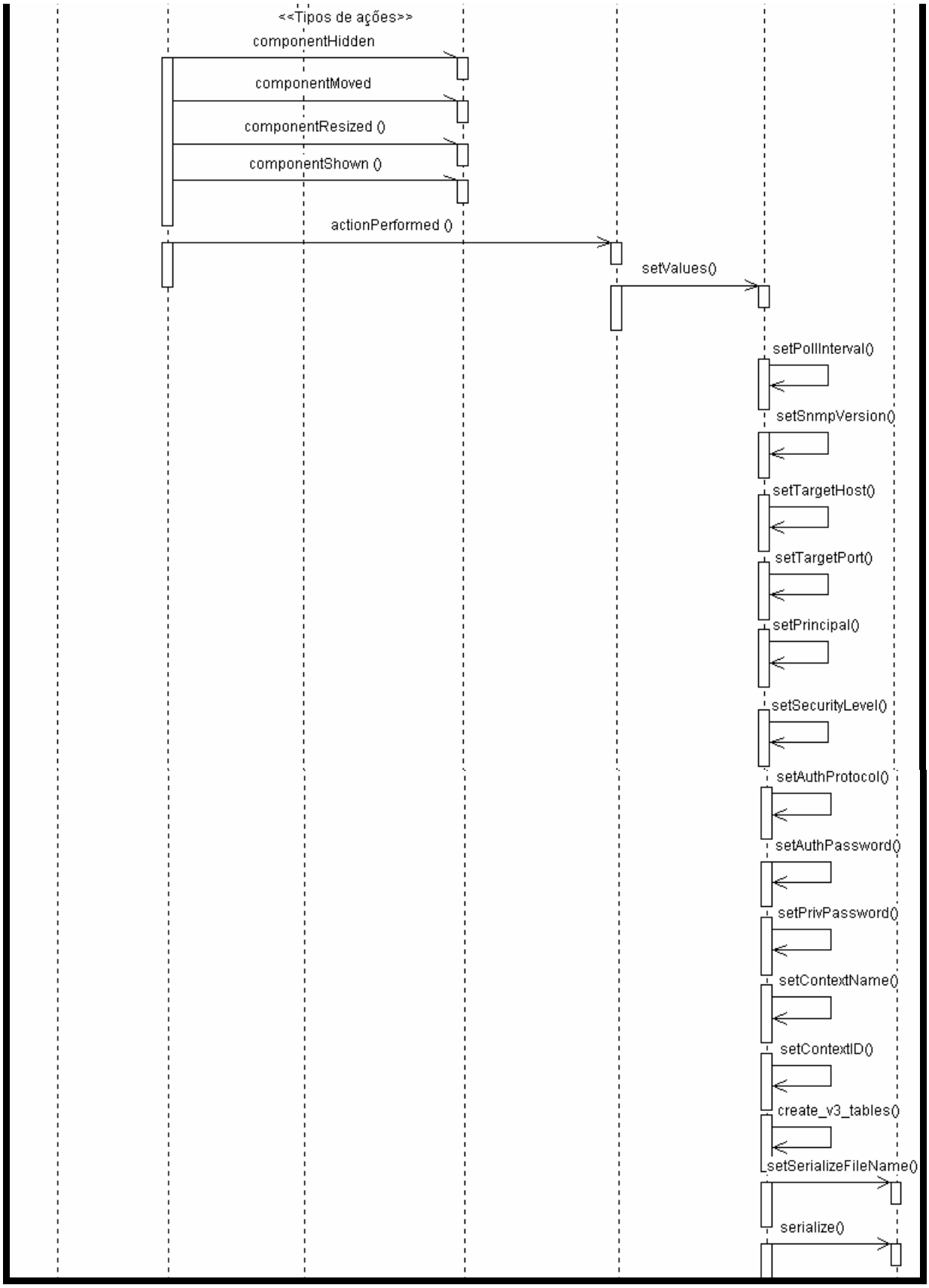
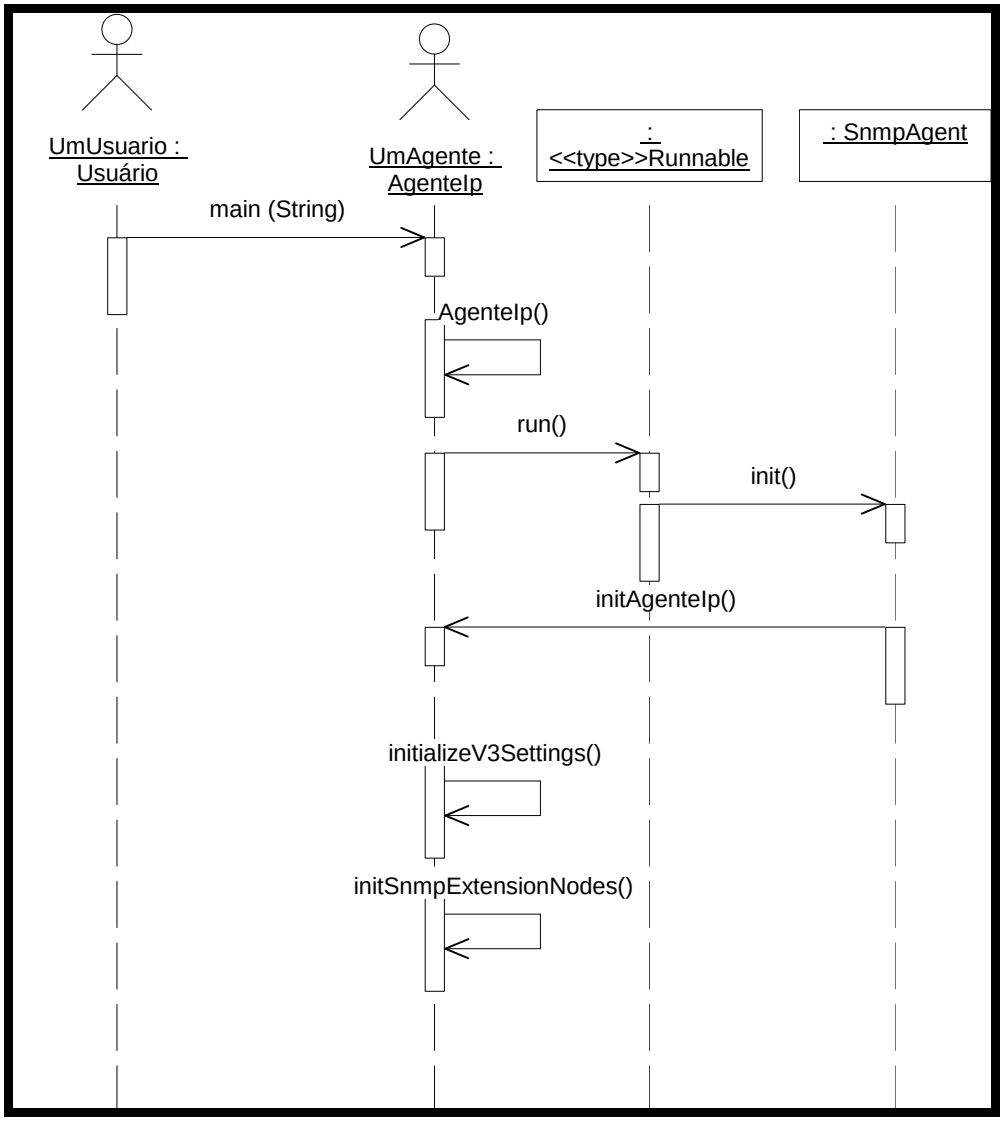

<<Tipos de ações>>
componentHidden
componentMoved
componentResized ()
componentShown ()
actionPerformed ()
setValues()
setPollInterval()
setSnmpVersion()
setTargetHost()
setTargetPort()
setPrincipal()
setSecurityLevel()
setAuthProtocol()
setAuthPassword()
setPrivPassword()
setContextName()
setContextID()
create_v3_tables()
setSerializeFileName()
serialize()

Figura 28 - Inicialização do agente



Devido ao fato de as operações de troca de mensagens entre as entidades agente e gerente serem extensas e numerosas, os diagramas de seqüência exibem apenas uma forma simplificada sobre como esse processamento é realizado. Por exemplo, o método processarPDU() pode se referir ao método processGetRequestMessage(), ao método processGetNetxRequestMessage(), ou ao método processSetRequestMessage() da classe IpRequestHandler. Além disso os diagramas de seqüência não permitem demonstrar processos que são processados simultaneamente, como por exemplo o processo de descoberta e sincronização.

6.3 IMPLEMENTAÇÃO

Nesta etapa será demonstrada a implementação do protótipo conforme a especificação desenvolvida anteriormente.

6.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para o desenvolvimento do protótipo foram utilizados a linguagem de programação Java, a API SNMPv3 da empresa AdventNet e o sistema operacional Windows2000 *Server* no qual foram realizados os testes do protótipo.

De acordo com Furlan (1998), "...uma das linguagens de programação que se adaptou à implementação de funções de gerenciamento foi a linguagem Java. O aparecimento da Java revolucionou o desenvolvimento de software para Internet, Intranet e quase todas as redes distribuídas. Além disso, a Java é uma linguagem totalmente orientada a objetos, dinâmica, independente de plataforma tecnológica, segura e relativamente fácil de usar".

Para o desenvolvimento de parte da funcionalidade do agente foi utilizada a Java *Native Interface* – JNI. Esta foi a forma encontrada para permitir o acesso por parte do agente, às informações sobre os pacotes IP do sistema operacional Windows2000 *Server*.

Também foram utilizados *Java Beans*, "...estruturas utilizadas para definir componentes de software reutilizáveis, incorporáveis e modulares." (Flanagan, 1999)

Para obter mais informações sobre a linguagem Java, *Java Beans* e JNI consulte Deitel (2001), Flanagan (1999) e Sun (199-).

Segundo a AdventNet (1995), a API SNMP AdventNet é o ambiente de desenvolvimento mais abrangente para construir aplicações e *applets* de gerenciamento baseados na estrutura SNMP. Esta API consiste em pacotes Java que podem ser usados para desenvolver soluções e produtos para o gerenciamento de redes baseados em Java e na *Web*.

Para o desenvolvimento do protótipo foi utilizada a API SNMPv3 versão 4 da empresa AdventNet. Para obter maiores informações sobre a API SNMPv3 da AdventNet consulte AdventNet (2001).

O protótipo também utiliza a *Extensible Markup Language* – XML para armazenar as informações das tabelas VACM e USM do agente. Para obter maiores informações sobre a XML consulte W3C (1996).

O protótipo foi executado em uma rede utilizando o sistema operacional Windows2000 *Server*. Para obter maiores informações sobre este sistema operacional consulte Minasi (2000).

Quadro 01: Inicialização do agente

```
String version = "V3";
int debugLevel =  com.adventnet.agent.logging.Level.WARN;
AgentUtil.setAgentDir(agentDir);
if(agentOptions.getVersion() != null){
        version = agentOptions.getVersion();   }
super.setSnmpVersion(version, false);
if(agentOptions.getDebugLevel() != -1){
        debugLevel = agentOptions.getDebugLevel(); }
super.setDebugLevel(debugLevel);
int port = 161;
if(agentOptions.getPort() != -1){
        port = agentOptions.getPort();   }
super.setPort(port, false);
int trapSendingPort =162;
if(agentOptions.getTrapSendingPort() != -1){
        trapSendingPort = agentOptions.getTrapSendingPort(); }
super.setAsyncMode(true);
super.setMaxThreads(4);
if(version.equalsIgnoreCase("V3")){
        // Compatibilidade com V3
        super.setSnmpV3Compliance(true);
        // Inicializa as configurações V3.
        super.setV3Configuration(true);
        initializeV3Settings();   }
```

Inicialmente foram criadas as classes AgenteIP, IpRequestHandler e IpInstrument. Na classe AgenteIp estão definidas todas as operações de inicialização do agente dentre as quais

destaca-se a *initAgenteIp()*, conforme mostrado no quadro 01 e a inicialização das tabelas VACM e USM realizada pelo método *initializeV3settings()* exibido no quadro 02.

Quadro 02: Inicialização das tabelas VACM e USM do agente

```
public void initializeV3Settings(){

String engineID = "127.0.0.1x8001";

super.getSnmpAPI().setSnmpEngineID(engineID.getBytes());

usmUserListener =  new UsmUserRequestHandler(this);

usmUserListener.addRegistrationListener(hdlr);

usmUserTableListener =  new UsmUserTableRequestHandler(this, "conf", "UsmUserTable.xml", "xml");

usmUserTableListener.addRegistrationListener(hdlr);

vacmMIBViewsListener = new VacmMIBViewsRequestHandler(this);

vacmMIBViewsListener.addRegistrationListener(hdlr);

vacmContextTableListener =  new VacmContextTableRequestHandler(this, "conf", "VacmContextTable.xml",
"xml");

vacmContextTableListener.addRegistrationListener(hdlr);

vacmSecurityToGroupTableListener = new VacmSecurityToGroupTableRequestHandler(this, "conf",
"VacmSecurityToGroupTable.xml", "xml");

vacmSecurityToGroupTableListener.addRegistrationListener(hdlr);

vacmAccessTableListener = new VacmAccessTableRequestHandler(this, "conf", "VacmAccessTable.xml",
"xml");

vacmAccessTableListener.addRegistrationListener(hdlr);

vacmviewTreeFamilyTableListener = new VacmViewTreeFamilyTableRequestHandler(this, "conf",
"VacmViewTreeFamilyTable.xml", "xml");

vacmviewTreeFamilyTableListener.addRegistrationListener(hdlr);

}
```

O método *initializeV3settings()* é responsável por ler as informações das tabelas USM e VACM e carregà-las para a memória do agente. No quadro 03 é listado parte de uma destas tabelas: a *UsmUserTable.xml*.

Quadro 03: Parte do arquivo *UsmUSerTable.xml*

```
<?xml version="1.0" encoding="UTF-8" ?>
<Table>
<row>
<column name="usmUserSecurityName" value="privUser" />
<column name="usmUserSecurityLevel" value="3" />
<column name="usmUserCloneFrom" value=".0.0" />
<column name="usmUserAuthProtocol" value="MD5_AUTH" />
<column name="usmUserPrivProtocol" value="CBC_DES" />
<column name="usmUserPublic" value="757365725075626c6963" />
<column name="usmUserAuthPassword"value="7a78817495778578" />
<column name="usmUserPrivPassword" value="7a78817495778578" />
<column name="usmUserStorageType" value="3" />
<column name="usmUserStatus" value="1" />
</row>
</Table>
```

Estes arquivos de tabela podem ser editados manualmente pelo administrador ou dinamicamente pelos gerentes. No caso de edição manual o agente deve ser reiniciado para que as novas alterações entrem em vigor. A listagem completa das tabelas de configuração utilizadas pelo agente SNMPv3 é a seguinte:

a) *UsmUserTable.xml*: armazena os usuários que podem realizar operações neste agente;
b) *V3TrapForwardingTable.xml*: armazena os usuários configurados para receber *traps* SNMPv3;
c) *VacmAccessTable.xml*: armazena informações sobre grupos de acesso;
d) *VacmContextTable.xml*: armazena informaçòes sobre os contextos;
e) *VacmSecurityToGroupTable.xml*: contém a relação *SecurityLevel/Group* que relacionam os usuários de um nível de segurança com um grupo específico;
f) *VacmViewTreeFamilyTable.xml*: armazena informações sobre as visões existentes.

Já a classe IpRequestHandler é responsável por processar todas as mensagens SNMPv3 recebidas pelo agente destinadas ao grupo *ip* da Mib-2, realizando a autenticação e criptografia/descriptografia encaminhando em seguida a requisição para a classe responsável por processar esta requisição. Parte deste processamento pode ser visto no quadro 04.

Quadro 04: Parte do processamento de mensagens GET-Request SNMPv3

```
// Método que processa todas as operações GET solicitadas.
protected void processGetRequest(SnmpVarBind varb,
                                 AgentNode node,
                                 VarBindRequestEvent pe)
                                 throws AgentSnmpException{
try{
        // Isto indica que o nó está fora da visão das mibs carregadas.
        if(node == null){
        AgentUtil.throwNoSuchInstance(pe.getVersion());
}
int[] oid =  (int[] )varb.getObjectID().toValue();

if(oid.length != ipOidRep.length + 2){
        AgentUtil.throwNoSuchInstance(pe.getVersion());
}
int[] inst = (int[] )AgentUtil.getInstance(oid,ipOidRep.length);
if(inst[1] != 0){
        AgentUtil.throwNoSuchInstance(pe.getVersion());
}
Object toReturn  = null;
switch(node.getSubId()){
        case IPFORWARDING:
                toReturn = instrument.getIpForwarding();
                if(toReturn == null){
                        AgentUtil.throwNoSuchInstance(pe.getVersion());
                }
                 SnmpInt var0 = new SnmpInt(((Integer )toReturn).intValue());
                varb.setVariable(var0);
                break;
```

Após o processamento, a classe IpRequestHandler envia uma requisição à classe IpInstrument (desde que seja uma requisição às variáveis do grupo *ip* da Mib-2) para que ela realize as operações necessárias sobre as variáveis IP. Em seguida a classe IpInstrument retorna o resultado da operação para a classe IpRequestHandler que monta uma PDU-Response e, realiza o processamento SNMPv3 necessário encaminhando a resposta ao

gerente. Parte do processamento realizado pela classe IpInstrument pode ser visto no quadro 05.

Quadro 05: Parte do processamento de requisições do grupo *ip* da Mib-2

```
/**
* Trata o pedido SNMP Set Request para a variável ipForwarding
* @param valor - O valor Integer a ser configurado
* @throws AgentException em caso de erro
*/
public synchronized void setIpForwarding(Integer value)
        throws AgentException{
        if(value == null){
        throw new AgentException("", CommonUtils.WRONGVALUE);
        }
        if(!((value.intValue() == 1)||(value.intValue() == 2))){
        throw new AgentException("", CommonUtils.WRONGVALUE);
        }
        ipForwarding = value;
}
/**
* Trata o pedido SNMP Get Request para a variável ipDefaultTTL
 */
public Integer getIpDefaultTTL()
        throws AgentException{
        int Val_ipStat = 0;
        Val_ipStat = IpStat.getIpStatField(1);
        Integer ipDefaultTTL = new Integer(Val_ipStat);
        return ipDefaultTTL;
}
```

Já por parte do gerente o principal método é o método *setValues()* (quadro 06) que obtém os valores das variáveis SNMPv3 da tela do gerente e as utiliza para montar PDUs *GET-Request* utilizadas para realizar o *polling* sobre as variáveis solicitadas pelo usuário, além de realizar a descoberta e sincronização com o agente salvando as informações em um arquivo para uso em sessões futuras com o mesmo agente.

Quadro 06: Método *setValues()* do arquivo *GerenteDesempenhoIp.java*

```
public void setValues() {
        // Configurar os Oids da sessão.
        String Oids = tfOid.getText();
        StringTokenizer stoken = new StringTokenizer(Oids, " ");
        int num = stoken.countTokens();
        grafLinhas.setTitle("Plotando "+num +" OIDs");
        grafLinhas.setXlabel ("Tempo em segundos");
        grafLinhas.setYlabel ("Número de pacotes");
        String listaOids[]  = new String[num];
         for(int i=0; i<num; i++)
        { // Obtém os Oids da caixa de texto.
                 listaOids[i] = stoken.nextToken();
        }
        // Transfere os OIDs para o objeto poller gerente.
        poller.setObjectIDList(listaOids);
        // Configura o intervalo de polling.
        try
        {
                poller.setPollInterval( Integer.parseInt(tfIntervalo.getText()) );
        }
        catch (NumberFormatException nfm)
        {
                JOptionPane.showMessageDialog( GerenteDesempenhoIp.this,
                "O valor do intervalo deve ser um número inteiro!",
                "Configuração", JOptionPane.ERROR_MESSAGE );
                tfIntervalo.setText(String.valueOf(poller.getPollInterval()));
        }
        // Configura a versão.
        String ver = new String((String) comboVersao.getSelectedItem ());
        if(ver.equalsIgnoreCase ("v1"))
        {
                poller.setSnmpVersion (poller.VERSION1);
        }
        else if(ver.equalsIgnoreCase ("v2"))
        {
```

```
            poller.setSnmpVersion (poller.VERSION2C);
}
else if(ver.equalsIgnoreCase ("v3"))
{
            poller.setSnmpVersion (poller.VERSION3);
}
// Configura o host.
poller.setTargetHost( tfHost.getText() );
// Configura a porta.
poller.setTargetPort (Integer.parseInt ((tfPorta.getText ())));
// Caso seja a versão 3 do SNMP faça:
if(ver.equalsIgnoreCase ("v3"))
{
            // Configura o usuário que fará as requisições.
            poller.setPrincipal (tfUsuario.getText ());
            // Configura o nível de segurança.
            String NivSeg = new String((String) comboNivSeg.getSelectedItem ());
            if(NivSeg.equalsIgnoreCase ("noAuth,noPriv"))
            {
                        poller.setSecurityLevel(poller.NO_AUTH_NO_PRIV);
            }
            else if(NivSeg.equalsIgnoreCase ("Auth,noPriv"))
            {
                        poller.setSecurityLevel(poller.AUTH_NO_PRIV);
            }
            else if(NivSeg.equalsIgnoreCase ("Auth,Priv"))
            {
                        poller.setSecurityLevel(poller.AUTH_PRIV);
            }
            // Configura o protocolo de autenticação.
            String ProtAut = new String((String) comboAut.getSelectedItem ());
            if(ProtAut.equalsIgnoreCase ("MD5"))
            {
```

```
poller.setAuthProtocol(poller.MD5_AUTH);
}
else if(ProtAut.equalsIgnoreCase ("SHA"))
{
                    poller.setAuthProtocol(poller.SHA_AUTH);
          }
          // Configura a senha para autenticação.
          poller.setAuthPassword(tfSenhaAut.getText());
          // Configura a senha para privacidade.
          poller.setPrivPassword(tfSenhaPriv.getText ());
          // Configura o contexto.
          poller.setContextName (tfContexto.getText ());
          // Configura o IDContexto.
          poller.setContextID (tfIDContexto.getText ());
          // Cria as tabelas USM e VACM necessárias e faz a sincronização.
          int criar = poller.create_v3_tables();
          // Testar se tudo funcionou.
          if(criar == 1)
          {
                    JOptionPane.showMessageDialog( GerenteDesempenhoIp.this,
                    "Inicialização e Sincronização Efetuada!",
                    "Configuração V3", JOptionPane.INFORMATION_MESSAGE );
          // Salvar as informações em disco para uso futuro (nao é necessário).
          SnmpAPI api = new SnmpAPI();
          api.setSerializeFileName ("ConfV3.ser");
          try {
                    if(api.serialize() == false)
                    {
                    JOptionPane.showMessageDialog( GerenteDesempenhoIp.this,
                    "Erro ao salvar as informações no arquivo!",
                    "Configuração V3", JOptionPane.ERROR_MESSAGE );
          }
   } catch (SnmpException se) { }
}
else
{
```

```
                    JOptionPane.showMessageDialog( GerenteDesempenhoIp.this,
                    "Inicialização e/ou Sincronização Falhou!",
            "Configuração V3", JOptionPane.INFORMATION_MESSAGE );
            }  // fim criar.      } // fim poller.getVersion==3.
else
{
            poller.setCommunity(tfComunidade.getText());
}
// Exibe os resultados no grafico.
grafLinhas.setLineLabels(Oids);
// E inicia o polling.
grafLinhas.restartPolling();
// O envio e recepção das mensagens é controlado automaticamente
// pela API, não é necessário fazer mais nada.
}
```

A seguir será discutida a operacionalidade do protótipo.

6.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Para a execução do protótipo é preciso tomar as seguintes medidas:

a) no(s) equipamento(s) onde será instalado/executado o agente e o gerente, esteja instalada a *Java Virtual Machine* – JVM versão 1.4 ou superior;

b) configurar o arquivo *java.security* presente em "C:\Program files\Java\j2re1.x.x\lib\security" acrescentando-se a seguinte linha ao mesmo: *security.provider.2=com.sun.crypto.provider.SunJCE*;

c) copiar todos os arquivos existentes no diretório "Agente" para um diretório qualquer no equipamento onde será executada a aplicação agente e todos os arquivos do diretório "Gerente" para um diretório qualquer no equipamento onde será executada a aplicação gerente;

d) copiar o arquivo *IpStatProj.dll* presente no diretório "IpStatNativo" para o diretório *System* do sistema operacional;

e) configurar as variáveis de ambiente Java executando o arquivo de configuração *ConfJavaVars.bat* presente no diretório “bin” das aplicações agente e gerente;

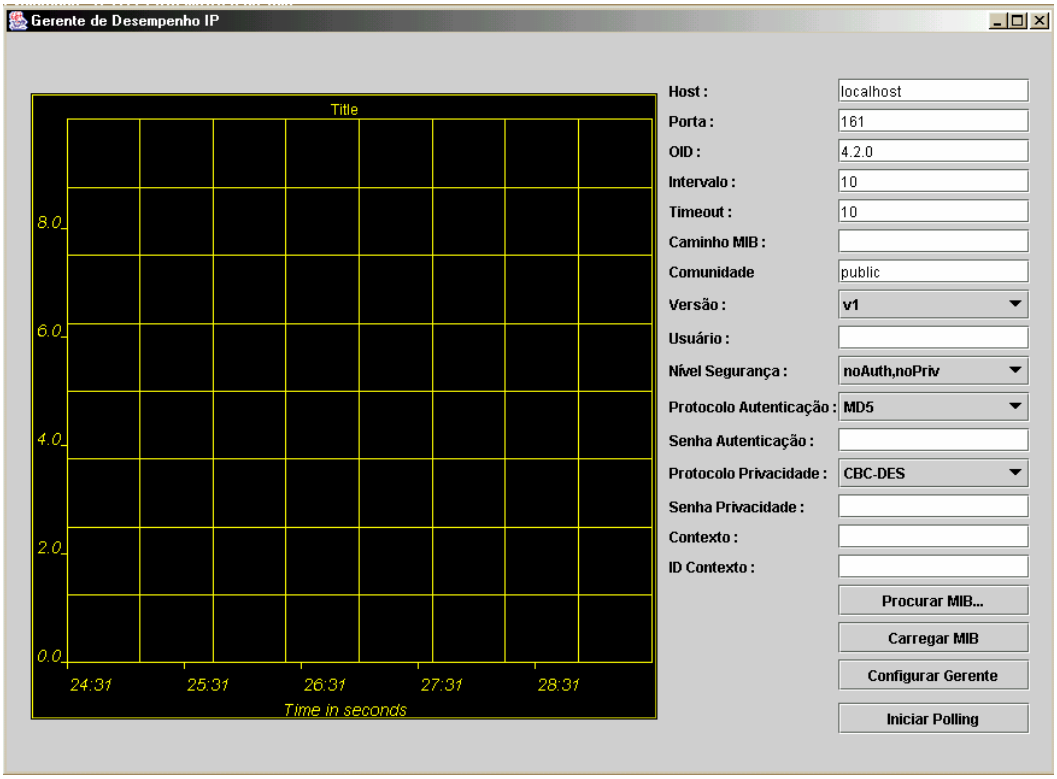
f) executar as aplicações agente e gerente através da execução do arquivo *run.bat* presente no diretório “bin” das aplicações agente e gerente.

Estes passos são necessários porque o protótipo foi desenvolvido utilizando a linguagem Java, algoritmos de privacidade e um método nativo (JNI) escrito em Java.

No caso do agente é possível executá-lo usando-se opções especiais de inicialização, executando-se a aplicação AgenteIp presente no diretório “bin” do agente, diretamente no *prompt* de comando do Microsoft *Disk Operating System* - MS-DOS. A seguir são exibidos todos os parâmetros adicionais que podem ser usados na inicialização do agente:

java AgenteIp [-d Debug_Level(0-6)] [-m arquivo_MIB] [-p porta] [-v versao] [-t TrapSendingPort]

Figura 29 - Tela Inicial da aplicação gerente



onde:

-d [nível depuração (0-6)] indica o nível de depuração a ser utilizado pelo agente;

-m [arquivo] indica o arquivo de definição MIB a ser utilizado pelo agente;

-p [porta] indica a porta na qual o agente irá aguardar por requisições;

-v [versão] indica qual versão SNMP o agente irá suportar;

-t [porta de envio de *trap*] indica a porta que o agente irá utilizar para enviar *traps*.

No caso do agente, um exemplo de inicialização seria o seguinte:

java AgenteIp –p 161 –m AKIP-MIB

No exemplo acima o agente será inicializado na porta 161 usando a MIB AKIP-MIB. Esta MIB contém a definição de todos os objetos escalares do grupo *ip* da Mib-2 Ela é utilizada no protótipo tanto pelo agente quanto pelo gerente para realizar o gerenciamento de desempenho.

A tela inicial do gerente, que surge logo após a execução do arquivo *run.*bat é mostrada na figura 29. Nesta tela é possível configurar todas as opções utilizadas pelo SNMPv3. Primeiramente deve-se carregar a MIB que a aplicação gerente utilizará realizando-se o seguinte procedimento:

a) clicar no botão “Procurar MIB...”;
b) escolher um arquivo de definição MIB que esteja presente no equipamento onde o agente se encontra. No caso do protótipo é preciso carregar a MIB AKIP-MIB presente no diretório “mibs” da aplicação gerente;
c) clicar no botão “Carregar MIB”. Se houverem erros de sintaxe no arquivo MIB carregado será exibida uma mensagem de erro no *prompt* do MS-DOS.

Em seguida o usuário deve preencher os campos SNMPv3 com as informações necessárias para se conectar ao agente com o qual deseja se comunicar.

Após preencher os campos o usuário deve clicar no botão “Configurar gerente”. Se as informações estiverem corretas e o processo de descoberta e sincronização com o agente ocorrerem com sucesso, será exibida uma mensagem informando que tudo correu bem e que é

possível a partir de agora consultar o agente, senão, uma mensagem de erro será exibida informando o motivo do erro.

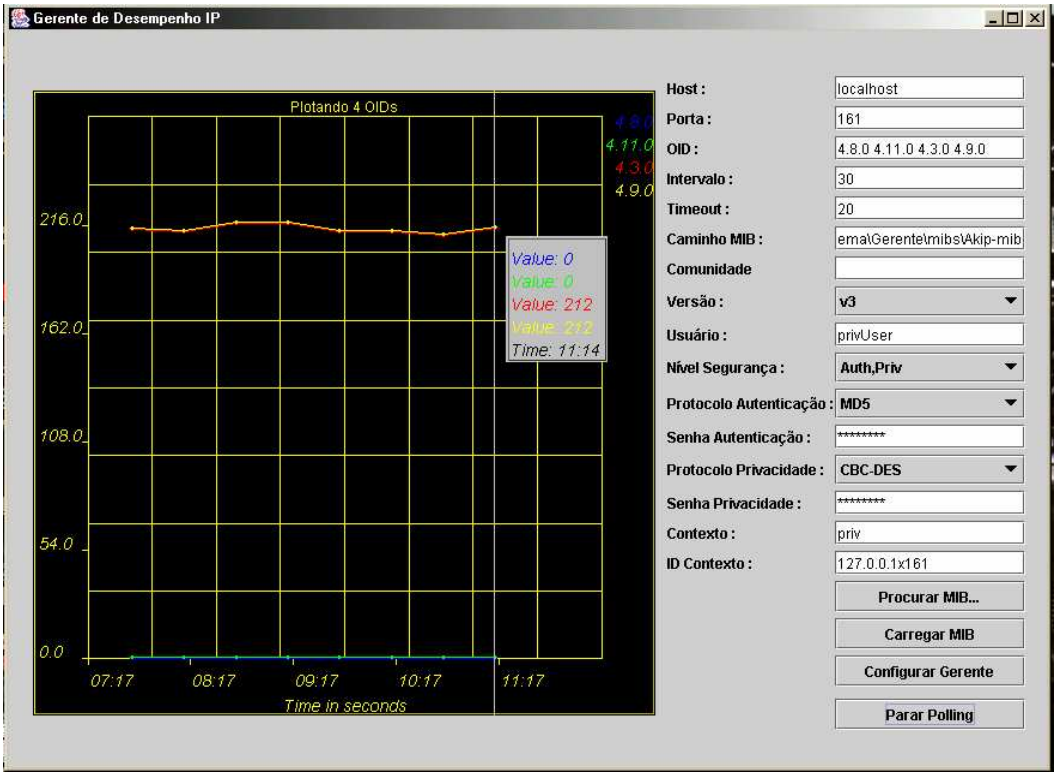
Agora o usuário deve preencher o campo OID especificando as variáveis do grupo *ip* que deseja consultar digitando a parte final de seu *ObjectID* ou OID (a aplicação gerente preenche internamente a parte inicial do OID – .1.3.6.2.1) É possível consultar mais de um OID ao mesmo tempo. Para isso basta preencher o campo OID separando as variáveis com um “espaço” simples.

É também possível utilizar a aplicação gerente para consultar agentes SNMPv1 e SNMPv2. Para tal, basta preencher o campo “Comunidade” e fornecer o(s) OID(s) que deseja consultar.

A aplicação gerente também pode consultar outras variáveis além daquelas existentes na MIB AKIP-MIB. Para poder consultar outras variáveis da Mib-2 basta carregar outra MIB contendo a definição de variáveis de outros grupos da Mib-2 tais como *if*, *system*, etc.

A seguir será exibido um exemplo do uso do sistema no monitoramento de um elemento de rede (PC) do laboratório Protem da Universidade Regional de Blumenau durante o período de 30 minutos. As informações de gerenciamento de desempenho coletadas foram as seguintes: *ipInDiscards*, *ipOutDiscards*, *ipInDelivers* e *ipInReceives.*

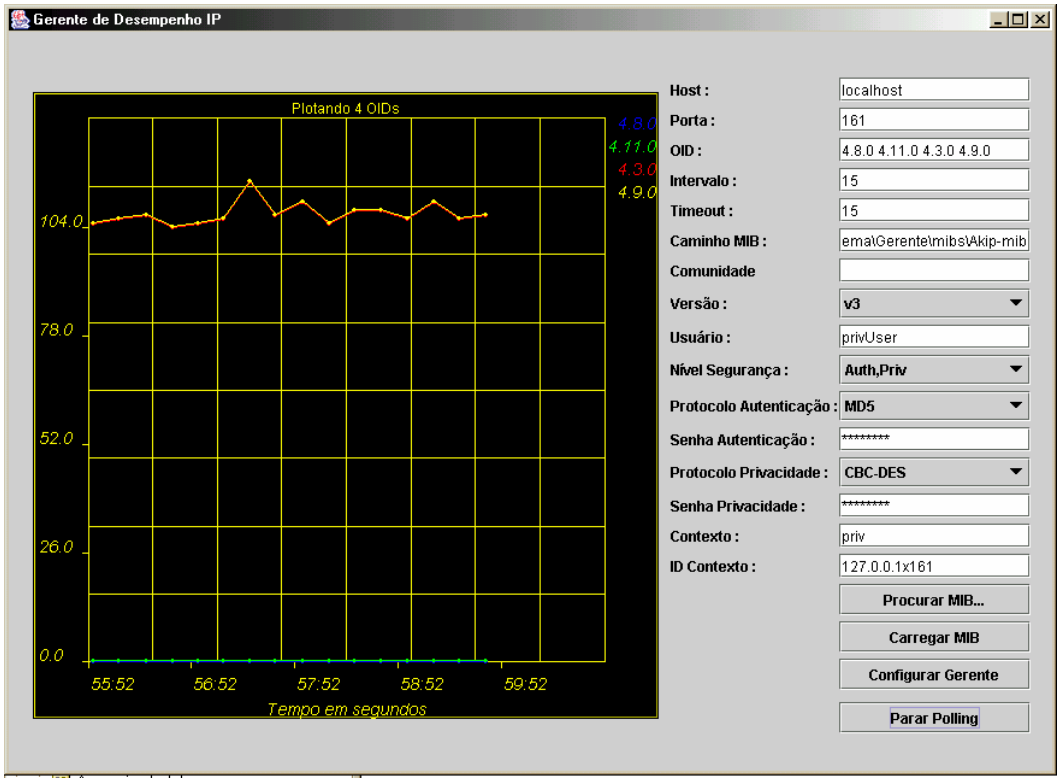
Figura 30 - Medição inicial



Foram realizadas duas coletas em tempos diferentes. A primeira foi realizada com intervalos de 20 segundos entre cada *polling* (fig. 30). A segunda coleta foi realizada com intervalos de 15 segundos entra cada *polling* (fig. 31).

Os resultados obtidos serão discutidos e analisados no próximo tópico.

Figura 31 - Medição final



6.4 RESULTADOS E DISCUSSÃO

É possível perceber pelos gráficos exibidos nas figuras 30 e 31 que na primeira coleta o equipamento estava utilizando muitos pacotes IP para a transmissão e recepção de dados. Na segunda coleta entretanto, percebe-se que o número de pacotes sendo transmitidos e recebidos diminuiu, indicando que o sistema operacional aumentou o tamanho dos pacotes IP, o que permitiu o aumento do desempenho da rede. Resultado este que pode ser observado ao se observar a taxa de transmissão de um arquivo sendo descarregado no equipamento analisado. Durante a primeira coleta a taxa de transmissão era de aproximadamente 15Kb por segundo sendo que no momento da segunda coleta esta taxa já estava em 32Kb por segundo.

Seria interessante poder testar o protótipo em um servidor *Web*, pois neste, há um grande número de pacotes IP sendo processados, sendo que neste caso, analisar o tráfego gerado pelos pacotes IP bem como o número de pacotes sendo descartados devido a erros ou falta de espaço em *buffer* seria essencial para um administrador de rede poder tomar decisões baseadas nas informações disponibilizadas pelo gerente do protótipo.

7 CONCLUSÕES

Atualmente, há um grande crescimento no uso das redes de computadores, e devido ao grande número de equipamentos de rede existentes atualmente e por estes equipamentos possuírem características heterogêneas e por serem produzidos por vários fabricantes diferentes, a possibilidade de que estes equipamentos venham a falhar ou que algum destes equipamentos possa comprometer o desempenho desta rede (sua qualidade de serviço) é também maior. Neste cenário, é essencial o uso de ferramentas automatizadas que possam gerenciar remotamente estes equipamentos. O SNMPv3 torna possível realizar esse gerenciamento de forma segura e eficaz, fornecendo funcionalidades para obtenção de segurança e controle de acesso via políticas de acesso.

O único objetivo que não foi completamente atingido foi o de gerenciar o desempenho real de uma rede de computadores. Isto aconteceu devido à falta de tempo hábil para realizar um estudo mais detalhado das funções de gerenciamento de desempenho existentes, além da pouca literatura existente até mesmo na Internet sobre o assunto.

O uso dos diagramas de seqüência, se mostrou parcialmente inadequado para representar o processamento e a troca de mensagens entre o agente e o gerente, principalmente por não permitir representar situações que ocorram em paralelo e situações onde existe mais de um caminho possível a ser seguido. Para resolver este problema sugere-se a utilização de diagramas de estado e de atividades que também fazem parte da UML.

As ferramentas utilizadas na implementação do protótipo se mostraram totalmente adequadas para o seu desenvolvimento. É importante destacar que a utilização da API SNMPv3 da AdventNet sem dúvida facilita e possibilita o rápido desenvolvimento de aplicações para o gerenciamento de redes utilizando o protocolo SNMP, além de disponibilizar um suporte rápido e gratuito aos desenvolvedores que utilizam a sua API.

A utilização do SNMP como uma plataforma para realizar o gerenciamento de redes de computadores é prática e fácil, entretanto, a terceira versão desta plataforma – SNMPv3 - já não é mais tão “simples”, pois a adição de privacidade e controle de acesso adiciona também, complexidade.

Uma das limitações do protótipo é o fato de o gerente não realizar nenhum processamento ou tomada de decisão sobre os dados coletados, ou seja, apresentar alguma “inteligência”. Novamente, em um sistema de gerência de redes real, seria indispensável que a aplicação gerente realizasse pelo menos um processamento bruto dos dados (um cálculo estatístico, o uso de alguma função de comparação de dados, etc), ou algo como “avisar” o administrador da rede sempre que certos limiares fossem atingidos.

Outra limitação é a de que apesar do protótipo ter sido desenvolvido em Java, parte da portabilidade (característica comum e importante desta linguagem) foi perdida, devido ao uso de um método nativo, sendo necessário realizar adaptações no presente protótipo para acessar as informações de desempenho em outras plataformas. Isto ocorreu, porque na época do desenvolvimento da proposta não se descobriu nenhum equipamento de rede que já disponibiliza, e por conseqüência, implementa as novas funcionalidades SNMPv3 sendo necessário então definir na proposta o desenvolvimento de todo um sistema de gerência (agente e gerente) ao nível de protótipo. Pouco antes do término do presente trabalho entretanto, descobriu-se que praticamente todos os roteadores da empresa Cisco já possuíam em seus equipamentos, a partir da versão 12 de seu sistema operacional, suporte via configuração de agentes Cisco, às novas funcionalidades do SNMPv3.

Como contribuições do presente trabalho pode-se citar as seguintes:

a) um estudo simplificado da estrutura SNMPv3, seus elementos, características e suas novas funcionalidades;
b) a criação de um protótipo simples mas funcional, implementando na prática as novas funcionalidades do SNMPv3;
c) um trabalho que pode ser estendido, podendo servir como base para novos trabalhos conforme as sugestões para extensão citadas na próxima sessão;
d) um estudo da viabilidade e praticidade no uso da API SNMPv3 da empresa AdventNet no desenvolvimento de aplicações para o gerenciamento de redes de computadores utilizando o SNMPv3;
e) a criação de um gerente SNMPv3, que pode vir a ser utilizado com outros softwares de gerência de redes de computadores baseados no SNMPv3 para a implantação da gerência de desempenho;

f) a criação de um agente SNMPv3 para o gerenciamento de desempenho de uma rede de computadores, que implementa parte do grupo *ip* da Mib-2.

7.1 EXTENSÕES

Como sugestões para trabalhos futuros pode-se citar as seguintes:

a) implementar alguma "inteligência" no gerente e no agente a fim de automatizar o processo de decisão quando do surgimento de problemas;

b) estender o agente para que ele possa tratar mensagens *Notification* e implementar o mecanismo de *Trap Forwarding* mecanismo este não implementado no presente trabalho;

c) estender o agente para que ele possa fornecer suporte a todos os objetos presentes na MIB-2, aumentando sua escalabilidade;

d) testar a extensibilidade do SNMPv3 através da implementação de novos módulos para realizar as tarefas de autenticação, privacidade e controle de acesso;

e) testar a modularidade do SNMPv3 através da utilização de novos algoritmos de criptografia e autenticação em substituição àqueles definidos pelo SNMPv3 além da definição de novas MIBs para criação de políticas de acesso especializadas;

f) realizar testes de desempenho para medir a utilização de largura de banda e processamento utilizados pelo SNMPv3 em comparação com as versões anteriores a fim de medir o custo "*overhead"* e perda de desempenho ocasionada pelas novas primitivas de autenticação e segurança do SNMPv3.

REFERÊNCIAS BIBLIOGRÁFICAS

ADVENTNET Inc. **AdventNet SNMP API 3.3 documentation**, Pleasanton, [2001]. Disponível em: <http://www.adventnet.com/products/snmp/help.html>. Acesso em: 14 nov. 2002.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML**: guia do usuário. 5. ed. Rio de Janeiro: Campus, 2000. 472p.

CARVALHO, Tereza Cristina Melo de Brito. **Gerenciamento de redes**: uma abordagem de sistemas abertos. Sao Paulo: Makron Books, 1993. 364p.

DEITEL, Harvey M, DEITEL, Paul J, SANTRY, S. (Sean), et al.. **Advanced Java 2 platform** : how to program. Upper Saddle River : Prentice Hall, 2001. 1811p.

FLANAGAN, David. **Java in a nutshell**: a desktop quick reference. 3.ed. Beijing: O'Reilly, 1999. 649p.

FURLAN, Jose Davi. **Modelagem de objetos através da UML-The Unifield Modeling Language**. São Paulo: Makron Books do Brasil, 1998. 329p.

HEGERING, Heinz-Gerd; ABECK, Sebastian. **Integrated network and system management**. Wokingham: Addison-Wesley, 1994. 491p.

HORSTMANN, Cay S, CORNELL, Gary. **Core Java 2**. São Paulo : Makron Books, 2001.

RFC/STD/FYI/BCP Archives. **Internet RFC/STD/FYI/BCP archives**, [199-]. Disponível em: < http://www.faqs.org/rfcs/>. Acesso em: 16 nov. 2002.

KUROSE, James F.; ROSS, Keith W. **Computer networking**: a top-down approach featuring the Internet. Boston: Addison Wesley, 2001. 711p.

LUCHESI, Claudio Leonardo; **Introdução à criptografia computacional**. Campinas: Ed. Papirus/Unicamp, 1986.

LYNCH, Daniel C.; ROSE, Marshall T. **Internet system handbook**. Boston: Addison Wesley, 1993.

MENEZES, A; OORSCHOT, P; VANSTONE, S. **Handbook of applied cryptography**. Canada: CRC Press, 1997. 780p.

MINASI, Mark. **Mastering Windows 2000 Server**. 2.ed. San Francisco: Sybex, 2000. 1593p.

RATIONAL Software Corporation. **Rational software**, Cupertino, [1995]. Disponível em: <http://www.rational.com/support/documentation/>. Acesso em: 27 ago. 2002.

ROSE, Marshall T. **The simple book**: an introduction to Internet management. 2. ed. Englewood Cliffs: Prentice-Hall, 1994. 456p.

SCHNEIER, Bruce. **Applied cryptography**: protocols, algorithms, and source code in C. 2. ed. Canada: Wiley & Sons, 2001. 762p.

SNMP International Research Inc. **SNMPv3 white paper**, New York, [2001?]. Disponível em: <http://www.snmp.com/snmpv3/v3white.html>. Acesso em: 14 ago. 2002.

STALLINGS, William. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2**. 3. ed. Boston: Addison Wesley, 2001. 628p.

SUN. **Java tutorial**, Palo Alto, [199-]. Disponível em: <http://docs.sun.com/>. Acesso em: 14 ago. 2002.

TANENBAUM, Andrew S. **Redes de computadores**. 3. ed. Tradução nome do tradutor/a. Rio de Janeiro: Campus, 1997.

W3C. **Extensible Markup Language**, Massachussets, [1996]. Disponível em: < http://www.w3.org/XML/>. Acesso em: 16 nov. 2002.

ZELTSERMAN, Dave. **A practical guide to SNMPv3 and network management**. Upper Saddler River: Prentice Hall, 1999. 337p.