**Uma avaliação evolutiva dos protocolos
de gerenciamento da Internet
e suas implementações:
SNMPv1, SNMPv2 e SNMPv3.**

Fábio Teruo Morishita

orientador: Prof. Dr. Edson dos Santos Moreira

Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos,,
Universidade de São Paulo, como parte dos requisitos para a obtenção do título de
Mestre em Ciências – Área de Ciências de Computação e Matemática Computacional.

São Carlos
Dezembro de 1997

*Aos meus pais,
Kogi e Shizue.*

*E a minha noiva,
Maria Helena.
Pelo seu amor, compreensão e
incentivo durante todos esses anos*

Agradecimentos

Ao Prof. Dr. Edson dos Santos Moreira pela orientação, apoio, incentivo e muita dedicação.

Aos amigos Shiro (Antônio Marcos), Rogério (Mouro), Adriano, Mário, Ciro e Robson pelo grande apoio, incentivo e companheirismo.

A imensa família que se tornou o laboratório Intermídia, em especial aos amigos de todas as horas Taboca, Roger, Pissioli, Elderclei, Manzato, Fabrício, Vidigal e Mac.

Aos companheiros de república Maurão, Tchelo, Hasegawa que sempre estavam a disposição para incentivar e colaborar.

Aos professores Onofre, Graça Pimentel, Graça Nunes, André, Marcos Santana pelo apoio e incentivo.

A todos os amigos do ICMSC, Bete, Laura, Sandrinha, Paulinho, Ângelo (Tec), Rogério (Tec), Luciano (Tec), Sônia, Cabral, Eduardo e Peleteiro pelo atenção e amizade.

A todos os funcionários do ICMSC que direta ou indiretamente contribuíram com este trabalho.

Aos amigos de Prudente Rogério, Ronaldo, Milton e Silvio pelo incentivo e amizade.

À FAPESP, Fundação de Amparo à Pesquisa do Estado de São Paulo, pela credibilidade e apoio financeiro.

A todos as pessoas que de alguma forma me incentivaram ou contribuíram com este trabalho.

Aos meus pais e irmãos Edgar e Alessandra que sempre me apoiaram e incentivaram.
À família de minha noiva pelo grande apoio e carinho.

A minha noiva Maria Helena pelo amor, dedicação e incentivo que venho recebendo desde que a conheci.

Resumo

O desenvolvimento da padronização do modelo de gerenciamento de rede é uma necessidade que todos os grupos envolvidos (pesquisadores, fabricantes de equipamentos de rede, fornecedores de soluções de rede, etc.), buscam para possibilitar a interoperabilidade. Neste trabalho foram realizados estudos dos principais modelos de gerenciamento, *Internet* e o *OSI*, apresentando suas características e as deficiências. Os estudos concentraram-se sobre os protocolos de transporte das mensagens de gerenciamento, *CMIP* e *SNMP*, por serem a chave para realizar o gerenciamento, e especialmente o *SNMP* versão 2. Uma ferramenta de gerenciamento de rede vem sendo desenvolvida no Instituto de Ciências Matemáticas de São Carlos – USP, denominada *NetTracker*, que passou por uma reestruturação para adaptar às novas tendências tecnológicas, Java, HTTP, CORBA, SSL, entre outras. Recentemente foram publicadas documentos (RFCs) que propõem alterações no protocolo *SNMP*, batizado de *SNMPv3*. Verifica-se que a nova versão do modelo estrutural apresenta similaridades ao do *NetTracker*. Com as formalizações das especificações tanto da ferramenta de gerenciamento como do protocolo, o projeto estruturou um modelo de um sistema gerente que absorve as novas tecnologias, adaptado-o as novas características do protocolo e da ferramenta.

Abstract

The development of the network management standard model is a target that all groups involved (researchers, network equipment manufacturers, network solution providers, etc.), are looking for in their quest for interoperability. In this work, the main networks management models, Internet and OSI, were studied, showing both their strength and weakness. The target of the study were the network management message transport protocol, CMIP and SNMP, the keys of network management and, especially, the version 2 of the late. The Instituto de Ciências Matemáticas de São Carlos – USP carries out a work on development a network management tool, the NefTracker. The framework of the NefTracker was envisaged to use the new set technology for the development of distributed applications such as: Java, HTTP, CORBA, SSL, and others. Recently it published the new proposal SNMP protocol, named SNMPv3. The Framework for this version fits very well to the NefTracker achitecture. As the SNMPv3 proposal gets the status of standard, the final contribution of this work is the mapping of its modules over structure of the NefTracker.

Índice

1- INTRODUÇÃO.....	1
2- GERENCIAMENTO DE REDE.....	4
2.1 STRUCTURE OF MANAGEMENT INFORMATION (SMI)	6
2.2 MANAGEMENT INFORMATION BASE (MIB).....	7
2.3 HISTÓRICO DO GERENCIAMENTO INTERNET	7
3- PROTOCOLOS DE GERENCIAMENTO.....	9
3.1 COMMON MANAGEMENT INFORMATION PROTOCOL (CMIP).....	9
3.2 COMMON MANAGEMENT INFORMATION SERVICE AND PROTOCOL OVER TCP/IP (CMOT)	13
3.3 SIMPLE NETWORK MANAGEMENT PROTOCOL (SNMP).....	13
3.3.1 Arquitetura de Gerenciamento Internet	14
3.3.2 Funcionamento do Protocolo.....	14
3.3.3 Controle de Acesso às Informações no Protocolo SNMP	16
3.3.5 Limitações do Protocolo SNMP.....	23
3.4 SIMPLE NETWORK MANAGEMENT PROTOCOL VERSION 2 (SNMPv2).....	24
3.4.1 Mensagens SNMPv2	28
3.4.2 Processamento das novas Primitivas.....	29
3.4.5 Arquiteturas de Gerenciamento Alternativos Baseado no Protocolo SNMPv2	32
3.5 BUSCA DO PADRÃO PARA A SEGURANÇA DO PROTOCOLO SNMPv2	34
3.6 O DESENVOLVIMENTO DO PADRÃO SNMPv3.....	36
3.6.1 Uma nova arquitetura para o SNMP	36
4- NOVAS TECNOLOGIAS DE INTERESSE À IMPLEMENTAÇÃO DE PROTOCOLOS DE GERENCIAMENTO	39
4.1 HYPERTEXT TRANSFER PROTOCOL (HTTP)	39
4.1.1 HyperText Markup Language (HTML).....	40
4.3 JAVA	40
4.3.1 Plataforma Java.....	42

4.3.2 Características Adicionais do Java	43
4.4 COMMON REQUEST BROKER ARCHITECTURE (CORBA)	43
4.4.1 CORBA e Java	44
5- ADAPTANDO A NOVA ESTRUTURA SNMPV3 AO SISTEMA NETTRACKER	45
5.1 A NOVA ESTRUTURA DO NETTRACKER.....	45
5.1.1 Camada de Módulos de Programas de Aplicação.....	46
5.1.2 Camada de Interface de Extensão de Aplicações	46
5.1.3 Camada de Módulos de Extensão.....	47
5.1.4 Biblioteca de objetos Java	48
5.1.5 Máquina Virtual Java	48
5.2 ESTRUTURA E RELACIONAMENTO DOS OBJETOS DO SISTEMA.	48
5.3 MAPEAMENTO DO SNMPV3 NA ESTRUTURA DO NETTRACKER.....	50
5.3.1 ManagerSNMP	51
5.4 ESPECIFICAÇÃO DAS FUNÇÕES BÁSICAS PARA O DESENVOLVIMENTO DO SNMPv2	52
5.4.1 Representação dos Dados.....	52
5.4.2 Objetos básicos do protocolo SNMPv2.....	58
6- CONSIDERAÇÕES FINAIS	61
6.1 CONCLUSÕES.....	61
6.2 TRABALHOS FUTUROS	62
GLOSSÁRIO	64
REFERÊNCIAS BIBLIOGRÁFICAS.....	66

Introdução

Nos anos 70 e início dos anos 80, a conexão de computadores era representada apenas por *mainframes* e seus terminais. Nessa configuração todo o processamento e gerenciamento do sistema eram centralizados nos *mainframes*.

A partir de meados dos anos 80 houve uma grande transformação na estrutura organizacional dos centros de processamento de dados (CPDs), surgindo um novo modelo estrutural de conectividade, motivado por fatores econômicos e tecnológicos. No qual destaca-se: o surgimento de estações de trabalho com real poder de processamento, à preços relativamente baixos; e redes de computadores que pudessem suportar maior tráfego, nas quais as estações de trabalho pudessem compartilhar recursos e trocar informações.

Inicialmente esse modelo de configuração começou a ser aplicado a nível local, criando-se as redes locais (LAN) ou departamentais. Estas foram crescendo em quantidade de elementos e em tamanho, com a interconexão das redes locais, que rapidamente alcançaram níveis de conexões internacionais (WAN, Internet).

A grande variedade de equipamentos e tecnologias de rede, tornavam os sistemas muito complexos, dificultando o gerenciamento e a manutenção. Verificou-se então a necessidade de criação de um padrão para o modelo de gerenciamento [RO91], de forma a conseguir que os equipamentos de rede de fabricantes distintos e tecnologias de rede se interoperassem.

Sugiram várias propostas para a padronização do modelo de gerenciamento de rede, destacando-se dois deles:

- O modelo de gerenciamento OSI (*Open Systems Interconnect*) é o mais completo e vem sendo utilizado como base para o desenvolvimento de outros modelos de gerenciamento, baseado no protocolo de transporte CMIP (*Common Management Information Protocol*).

- O modelo de gerenciamento Internet que utiliza o protocolo de transporte SNMP (*Simple Network Management Protocol*) e é adotado pelos fabricantes de equipamentos de rede como um padrão.

No Instituto de Ciências Matemáticas de São Carlos (ICMSC) está sendo desenvolvido um sistema gerenciador de rede baseado no modelo de gerenciamento Internet, denominado Multview/NetTracker [OD94,CI94,LI95,MO97], que possui algumas características especiais: como por exemplo o suporte a multimídia, com a transferência de sons e imagens entre as informações de gerenciamento, e o reconhecimento automático da topologia da rede.

Nos últimos anos houve uma grande explosão e popularização da “Internet”, motivados principalmente pelas tecnologias emergentes como *Java* e *WWW* (World Wide Web), e estas novas tecnologias estão sendo incorporadas nos ambientes departamentais das empresas como ferramentas de *Groupware* e interface padrão para os softwares aplicativos (*Intranet*).

Para manter o *NetTracker* atualizado com estas tecnologias, foi desenvolvido uma nova arquitetura organizacional [MO97], que as absorveu, tornando-o mais flexível. Foram realizados estudos sobre o protocolo *SNMP*, de forma a determinar suas principais características.

Esta dissertação encontra-se dividida em seis capítulos, dos quais este constitui a Introdução. O capítulo 2 discorre-se sobre o gerenciamento de redes, apresentando as principais características de um sistema de gerenciamento e as estruturas necessárias. Como o projeto é centrado no modelo de gerenciamento *Internet*, é também apresentado um breve histórico da busca pela padronização do modelo:

No capítulo 3 são abordados os principais protocolos de gerenciamento de rede (CMIP, CMOT, SNMP, SNMPv2, SNMPv3), e os modelos de gerenciamento *OSI* e *Internet*, e como o protocolo *SNMP* tornou-se um padrão *de fato*, há um estudo mais detalhado das primitivas de gerenciamento, apresentando as principais características (SNMP e SNMPv2). São discutidas também alguns dos problemas encontradas para a padronização *de fato* do protocolo *SNMPv2*. O capítulo 4 estão caracterizadas as tecnologias emergentes, que possibilitaram o desenvolvimento de soluções na área de redes de computadores, principalmente aquelas relacionadas à tecnologia da informação, destacando-se: *World Wide Web* (WWW), protocolos de comunicação, multimídia e segurança.

O capítulo 5 detalha o projeto e a operação da nova arquitetura do *NetTracker*, com o mapeamento de sua estrutura na nova especificação do protocolo SNMPv3. É demonstrada as capacidades destas novas estruturas no desenvolvimento detalhado do módulo gerente SNMP sobre o *NetTracker*. Uma especificação das funções básicas para o desenvolvimento do protocolo *SNMP* é também apresentada. Nos capítulos seguintes são apresentadas as conclusões finais e propostas para trabalhos futuros e finalizado com as referências bibliográficas utilizadas para o desenvolvimento deste projeto.

Gerenciamento de Rede

“O impacto de adicionar o gerenciamento de rede aos nós gerenciados deve ser mínimo, refletindo o menor denominador comum”

Marchall T. Rose

O gerenciamento de rede busca maximizar a produtividade e a eficiência, para alcançar esses objetivos a ISO (*International Organization for Standards*) definiu cinco funções:

- 1- Gerenciamento de Configuração:** Responsável por detectar e controlar o estado da rede, tanto a parte lógica como a física.
- 2- Gerenciamento de Desempenho:** Responsável por controlar e analisar o uso e a taxa de erros na rede, mantendo um histórico de informações.
- 3- Gerenciamento de Falhas:** Responsável por detectar, isolar e controlar procedimentos anormais que estejam ocorrendo na rede.
- 4- Gerenciamento de Contabilidade:** Responsável por coletar e processar dados relacionados ao uso da rede.
- 5- Gerenciamento de Segurança:** Responsável por controlar o acesso aos recursos da rede, utilizando-se de técnicas de autenticação e políticas de autorização.

Essas funções de gerenciamento de rede podem ser agrupadas em duas categorias [ST93]: monitoração e controle de rede.

Na monitoração tem-se as funções de gerenciamento de desempenho, falha e contabilização, que se destinam à observação e análise, dos estados da rede e seus elementos.

As informações monitoradas podem ser de três tipos:

- **Estáticas:** São as informações que sofrem poucas alterações. Por exemplo: a descrição de uma estação de trabalho em uma rede (nome e endereço).
- **Dinâmicas:** São informações que alteram com certa frequência. Por exemplo: o número de pacotes com erro em uma transmissão.
- **Estatísticas:** São as informações derivadas das dinâmicas. Por exemplo: o número de pacotes com erro em um determinado tempo.

No controle tem-se as funções de gerenciamento de configuração e segurança, que se envolvem com as alterações nas configurações das redes e seus elementos.

A estrutura adotada pelos sistemas de gerenciamento de rede segue a figura abaixo (Fig. 1): o gerenciamento é realizado com o compartilhamento de variáveis armazenadas na Base de Informações Gerenciáveis (MIB), trocando informações através de um protocolo de comunicação e seguindo o modelo Cliente-Servidor.

Figura 1 - Troca de informações entre cliente-servidor

As informações de gerenciamento de rede podem ser coletadas de duas formas:

- **Polling:** É a forma tradicional pela qual o gerente requisita as informações ao agente e este as retorna. O seu uso é aconselhável para o gerenciamento de informações que se alteram com pouca frequência.

- *Trap*: Ao contrário do *Polling*, é gerado pelo agente e enviado ao gerente, sendo utilizado em dois casos: para enviar informações sobre o estado atual das informações gerenciáveis e para alertá-lo sobre alguma situação anômala que esteja ocorrendo ou que irá ocorrer.

A escolha de que forma deverá ser realizada a coleta de informações dependerá da configuração de cada rede a ser gerenciada, comumente é adotado o seguinte esquema: na iniciação utiliza-se o *Polling* e posteriormente o *Trap*.

2.1 Structure of Management Information (SMI)

O SMI é responsável por descrever as estruturas e o esquema de identificação usado na definição das informações de gerenciamento. Inclui a descrição de um modelo de informação dos objetos para o gerenciamento de rede, assim como um conjunto de tipos usados para descrever informações de gerenciamento. As estruturas são definidas formalmente através da linguagem ASN.1.

Cada objeto gerenciado é especificado na SMI através de uma macro ASN.1, que inclui: nome, sintaxe, nível de acesso e informações relacionadas à obrigatoriedade da implementação do objeto.

Para identificar unicamente um determinado objeto, o nome ou identificador é formado por uma seqüência de números inteiros não negativos, obtidos através da árvore de identificação ASN.1 (Fig. 2) [RO91, ST93].

Figura 2 - Árvore de identificação ASN.1

2.2 Management Information Base (MIB)

A Base de Informação de Gerenciamento descreve os objetos a serem implementados e armazena os valores associados, de forma que possam ser vistos como uma base de informação virtual [CA93e]. Por exemplo, todos os dispositivos da rede Internet que se constituem em nós gerenciáveis devem implementar a MIB padrão Internet.

A MIB divide os objetos em grupos. A MIB padrão Internet apresenta oito grupos de objetos: *System*, *Interface*, *Address Translation*, *IP*, *ICMP*, *TCP*, *UDP* e *EGP*; a MIB II, dois grupos adicionais: *Transmission* e *SNMP*.

2.3 Histórico do Gerenciamento Internet

No início dos anos 80, verificou-se que o número de redes conectadas à Internet estava expandindo-se; além disso várias organizações começavam a assumir responsabilidade de gerência sobre partes da rede, tanto a nível de *backbone*, como a níveis regional e institucional.

Essas organizações utilizavam equipamentos de diferentes fornecedores, que trabalhavam com tecnologias de rede diferentes, causando problemas no gerenciamento da rede.

Assim, em 1987 surgiram três propostas para a padronização do modelo de gerenciamento de rede: o SGMP (*Simple Gateway Monitoring Protocol*), o HEMS (*High-Level Entity Management System*) e o CMOT (CMIP sobre TCP/IP).

Em fevereiro de 1988, o IAB (*Internet Activities Board*), na busca por uma padronização chega a duas novas propostas. Na primeira, seria desenvolvido um novo protocolo (SNMP) a partir do SGMP, destinado a ser uma solução a curto prazo, e a segunda concentrar-se-ia no desenvolvimento do CMOT, destinado a ser a solução a longo prazo.

Assim, em agosto de 1988, um modelo de gerenciamento comum às duas propostas é apresentado, chamado de Modelo de Gerenciamento de Rede Internet. Como parte desse documento encontravam-se dois outros documentos responsáveis pela definição das informações de gerenciamento, a SMI (*Structure of Management Information*) e a MIB (*Management Information Base*), ambos projetados visando à compatibilidade entre os dois modelos de gerenciamento [CA90].

Em abril de 1989, o SNMP se tornar-se-ia um padrão *de fato* para o gerenciamento de rede baseado no protocolo TCP/IP [CO91], em decorrência da aceitação dos fabricantes de produtos de rede. Por outro lado, o CMOT não apresentava soluções reais (implementação), deixando o SNMP como a única solução [RO91].

Em junho de 1989, o IAB decide-se, em uma nova reunião, pela utilização de modelos distintos; assim, os requisitos de compatibilidade do SMI e MIB entre os modelos ficavam suspensos, permitindo que se fizessem modificações sobre o SMI e a MIB, que as aproximassem mais das necessidades da comunidade Internet.

Em julho de 1992, documentos sobre um SNMP seguro (S-SNMP) foram publicados como proposta de padronização e no mesmo mês também foram publicados documentos que aconselhavam algumas modificações no SNMP, *Simple Management Protocol* (SMP). Ambos foram utilizados como base para o desenvolvimento da versão 2 do SNMP (SNMPv2), que corrige as deficiências.

Em março de 1993, foi publicado o conjunto de documentos de proposta de padronização do protocolo SNMPv2.

Protocolos de Gerenciamento

Dentre os vários protocolos desenvolvidos para a troca de informações de gerenciamento pode-se destacar: o CMIP (*Common Management Information Protocol*) baseado no Modelo de Gerenciamento OSI; o CMOT (*Common Management Information Service and Protocol over TCP/IP*), baseado no protocolo CMIP, que utilizava os serviços de transporte do conjunto de protocolos Internet (TCP/IP); e o SNMP (*Simple Network Management Protocol*), baseado no Modelo de Gerenciamento Internet.

3.1 *Common Management Information Protocol (CMIP)*

O CMIP é a especificação do protocolo de suporte para troca de informações de gerenciamento entre sistemas abertos do Modelo de Gerenciamento OSI (*Open Systems Interconnect*). Faz referência a uma coleção de documentos padronizados para o gerenciamento de rede da ISO (*International Organization for Standardization*), que inclui o gerenciamento de serviço (*Common Management Information Services - CMIS*) e conceitos associados [KL88, ST93].

O Modelo de Gerenciamento OSI é constituído de ferramentas e serviços, necessários para controlar e supervisionar as interconexões e os objetos gerenciados, subdividindo-se em três modelos:

- **Modelo Organizacional:** Descreve o gerenciamento OSI administrativamente, através de domínios de gerenciamento que por sua vez, apresentam subdomínios. O conceito de domínio reforça a criação de sistemas de gerenciamento distribuídos, com alto nível de cooperação e hierarquia bem definidas;
- **Modelo de Informação:** Estabelece normas para definir os objetos gerenciados e seus respectivos inter-relacionamentos, classes, atributos, ações e nomes. As informações de gerenciamento são armazenadas em uma base de informação (MIB). As convenções que permitem a descrição e identificação única das informações são definidas na Estrutura de Gerenciamento de Informação (SMI);

- Modelo Funcional: Divide as funções de gerenciamento em cinco grupos, que podem interagir entre si: gerenciamento de configuração, gerenciamento de desempenho, gerenciamento de falhas, gerenciamento de contabilização e gerenciamento de segurança.

Seguindo o modelo de referência para interconexão de sistemas abertos ISO/OSI [TA89], cada camada deve comunicar-se com a sua correspondente adjacente, o mesmo ocorrendo com o gerenciamento.

Existem Entidades de Gerenciamento de Camada, *Layer Management Entity* (LME), que se comunicam com a Entidade de Aplicação de Gerenciamento do Sistema, *System Management Application-Entity* (SMAE), usando um mecanismo local (Fig. 3).

Figura 3 - Comunicação entre entidades de gerenciamento no modelo OSI

O SMAE é responsável pela comunicação com outras entidades de gerenciamento a nível de aplicação. A troca de informações de gerenciamento é realizada através de protocolo CMIP, que define as regras de comunicação, sendo este acessado pelo SMAE na camada de aplicação.

O CMIP é um protocolo baseado no modelo orientado a conexão: antes da atividade de gerenciamento iniciar, tem-se uma conexão a nível de aplicação entre duas entidades de gerenciamento (Gerente-Cliente), criando um canal virtual entre as duas entidades para se comunicarem e executarem as funções de gerenciamento. Neste contexto, o gerenciamento de rede proposto baseia-se nos seguintes protocolos de aplicações (Fig. 4) [WA89]:

- *Association Control Service Element (ACSE)*: Para que as operações de gerenciamento sejam executadas usando o CMIP, uma associação deve ser estabelecida entre duas entidades de aplicação.
- *Remote Operation Service Element (ROSE)*: Permite a execução de operações remotas, operações de requisição/resposta e é também utilizado como mecanismo de retorno de erros.
- *Common Management Information Service Element (CMISE)*: O CMISE ou CMIS/CMIP é responsável por transferir informações de gerenciamento de um sistema ao outro, utilizando-se do ROSE e do ACSE. Fornece serviços para a notificação de eventos, obtenção e manipulação de dados de gerenciamento, executados através dos seguintes elementos de serviços:
 - *M_INITLALIZE*: Estabelece uma associação com um usuário CMISE remoto, atendendo ao pedido de um usuário CMISE para as trocas de informações de gerenciamento;
 - *M_TERMINATE*: Libera uma associação;
 - *M_ABORT*: Aborta uma associação;
 - *M_EVENT_REPORT*: Notifica um evento ocorrido com um objeto gerenciado a um usuário CMISE remoto;
 - *M_GET*: Requisita informações de gerenciamento de um usuário CMISE remoto;
 - *M_SET*: Requisita a modificação das informações de gerenciamento de um usuário CMISE remoto;
 - *M_ACTION*: Requisita a execução de uma ação para um usuário CMISE remoto;
 - *M_CREATE*: Requisita a criação de uma outra instância do objeto gerenciado a um usuário CMISE remoto;

- **M_DELETE:** Requisita a eliminação de uma instância de um objeto gerenciado a um usuário CMISE remoto.
- **M_LINKED_REPLAY:** Fornece várias respostas a uma única requisição.

Figura 4 - Conjunto de protocolos CMIP

As funções de gerenciamento definidas pelo CMIS são:

- **Get:** Utilizado para recuperar informações específicas de gerenciamento.
- **Set:** Utilizado para manipular as informações de gerenciamento.
- **Action:** Utilizado para executar comandos imperativos (ex.: reboot a estação).
- **Create:** Utilizado para criar uma nova instância de algum objeto gerenciável (ex.: uma entrada na tabela de rota).
- **Event-report:** Utilizado pelos agentes para informar sobre algum evento extraordinário ao gerente.

3.2 Common Management Information Service and Protocol over TCP/IP (CMOT)

Foi proposto pelo *Internet Activities Board* (IAB) para ser uma solução a longo prazo para o gerenciamento de redes TCP/IP [BE90], baseia-se no Modelo de Gerenciamento OSI, utilizando os serviços de transporte do ambiente TCP/IP [WA89].

Os serviços de gerenciamento são fornecidos pelo CMIS e os protocolos de aplicação, definidos pelo OSI: ACSE, ROSE e CMIS/CMIP (Fig. 5), implementados sobre o TCP UDP/IP. Isso é possível através do protocolo *Lightweight Presentation Protocol* (LPP) que permite o uso dos serviços de apresentação OSI sobre os protocolos de transporte Internet [WA89].

Figura 5 - Conjunto de protocolos CMOT

3.3 Simple Network Management Protocol (SNMP)

O SNMP começou a ser desenvolvido a partir de 1988 como uma resposta às necessidades de gerenciamento da rede Internet. O protocolo foi desenvolvido para ser uma solução provisória, enquanto se esperava o desenvolvimento de um novo protocolo mais completo (CMOT).

Esse novo protocolo não evoluiu, deixando o SNMP como única solução de padronização, sendo assim adotado por todos os fabricantes de equipamento de rede. O SNMP é o protocolo a nível de aplicação, utilizado para o transporte de informações de gerenciamento do Modelo de Gerenciamento Internet.

3.3.1 Arquitetura de Gerenciamento Internet

O Modelo de Gerenciamento Internet é baseado na classe de protocolos TCP/IP e composto por quatro elementos principais [ST93]:

Estação de Gerenciamento: Elemento responsável por prover a interação do gerente com o sistema de gerenciamento de rede. É composto por aplicações de análise de dados, interface de monitoração, controle de recursos de rede e uma base de dados, contendo as informações extraídas das entidades gerenciadas.

Agentes de Gerenciamento: Responsáveis por atenderem às requisições e executarem as tarefas designadas pela estação de gerenciamento. Além disso, podem alertar a qualquer momento a estação de gerenciamento sobre a ocorrência de eventos importantes.

Base de Informações de Gerenciamento: Formada pelos objetos representativos dos recursos disponíveis, esses objetos são padronizados entre os elementos pertencentes a uma mesma classe e representados como variáveis, mediante cuja leitura as funções de monitoração são efetuadas e as de controle são modificadas.

Protocolo de Gerenciamento de Rede: Responsável por transportar as informações entre os agentes e os gerentes, permite que sejam efetuadas as seguintes classes de operações:

- *GET*: Permite a obtenção, por parte da estação de gerenciamento, dos valores associados aos objetos da MIB.
- *SET*: Possibilita que a estação de gerenciamento efetue alterações dos valores dos objetos em um dado agente.
- *TRAP*: Permite que um agente notifique a estação de gerenciamento sobre a ocorrência de algum evento importante.

3.3.2 Funcionamento do Protocolo

Uma configuração típica do protocolo SNMP em uma estação de gerenciamento (Fig. 6) apresenta um processo gerente controlando o acesso a uma MIB central, provendo interface para o gerenciador de rede. O processo gerenciador executa suas funções de gerenciamento de rede através do SNMP, que é um protocolo a nível de aplicação projetado para operar sobre UDP [CO91b, HA95], parte da classe de protocolos TCP/IP.

Figura 6 - Configuração do SNMP

Nas estações há um processo agente responsável por interpretar as mensagens SNMP e efetuar o controle necessário sobre a MIB (Fig. 7).

Figura 7 - Mecanismo de Gerenciamento SNMP

Em um ambiente com grande número de agentes, seria inviável a estação de gerenciamento efetuar regularmente uma amostragem no conteúdo de cada MIB remota; desta forma adotam-se os seguintes mecanismos para a obtenção das informações:

No momento do início e a intervalos determinados pelo gerente, a estação de gerenciamento efetuará a amostragem das informações principais contidas em cada um dos agentes gerenciados.

Uma vez estabelecidas as condições operacionais dos agentes, estes ficam responsáveis por sinalizar à estação de gerenciamento a ocorrência de algum evento importante (ex.: falha de um

dispositivo), através de uma mensagem SNMP especial, *TRAP*. De posse da informação sobre a ocorrência de uma situação anômala, a estação gerente poderá tomar as providências necessárias.

Uma característica importante do protocolo SNMP é a possibilidade de gerenciar elementos de rede que não possuam agentes (ex: repetidores, *hubs*, *modems*, etc.) e elementos que possuam agentes de gerenciamento, mas não trabalhem com o protocolo SNMP.

Para realizar este processo de gerenciamento o protocolo SNMP utiliza os agentes *proxies*, que podem exercer a função de agente para os elementos de rede que não os possuam, ou trabalhar como um *gateway* a nível de aplicação, mapeando as funções de gerenciamento SNMP para as funções de gerenciamento do outro protocolo (Fig.8).

Figura 8 - Modelo do gerenciamento Proxy

3.3.3 Controle de Acesso às Informações no Protocolo SNMP

O SNMP possui características típicas encontradas em todas as aplicações distribuídas. Grupos não necessariamente disjuntos de agentes são gerenciados por uma estação gerente, onde cada agente deve, além de controlar a sua MIB local, permitir o controle desta pela estação gerente [HA95]. Essa relação de controle apresenta três características:

Serviço de Autenticação: Capacidade de limitar o acesso às estações gerentes autorizadas.

Política de Acesso: Possibilidade de estabelecer diferentes privilégios de acesso às estações de gerenciamento.

Serviço de *Proxy*: Capacidade de um gerente funcionar como um agente proxy, sendo um gerente-agente intermediário para a outra estação de gerenciamento.

Todos envolvem-se com conceitos de segurança de modo a permitir que os agentes protejam a si mesmos e suas MIBs contra tentativas de acessos não autorizados. O SNMP provê apenas uma primitiva com capacidade limitada de segurança - o conceito de comunidade.

Uma comunidade SNMP é uma relação de autenticação, controle de acesso e características de *proxy* entre um agente e um conjunto de estações gerentes. É definida localmente aos agentes, mantida pelas estações de gerenciamento e identificada por um nome único, sob contexto local conhecido como *nome de comunidade*. Para o serviço de autenticação, o *nome de comunidade* funciona como um mecanismo rudimentar, onde cada mensagem trocada entre agentes e gerentes é autenticada pela inclusão do *nome de comunidade* correto.

A possibilidade de definir várias comunidades por parte do agente permite estabelecer um método de controle de acesso para cada comunidade, especificando uma categoria diferente de acesso à MIB. Existem dois aspectos que devem ser considerados:

Visão de MIB SNMP: Um subconjunto dos objetos em uma MIB.

Modo de acesso SNMP: READ-ONLY, apenas leitura; e READ-WRITE, leitura e escrita.

A combinação de uma visão da MIB e um modo de acesso é conhecida como um *Perfil de Comunidade SNMP* e permite delinear o grau de acesso de uma determinada comunidade SNMP. O conceito de comunidade é importante sob o ponto de vista de serviço *proxy*, no qual os elementos envolvidos podem manter uma política de acesso entre as partes.

O SNMP define duas técnicas de identificação de instâncias de objetos: técnica de acesso sequencial e técnica de acesso aleatório [ST93]. Cada mensagem trocada entre uma estação de gerenciamento e um agente inclui o número de versão SNMP, um nome de comunidade e um dos cinco tipos de unidades de dados do protocolo (PDU), conhecidos como as primitivas operacionais (Fig 9).

Para permitir o intercâmbio de múltiplos objetos, todas as PDUs possuem um campo de conexão variável, consistindo de uma seqüência de referências para instâncias de objetos, juntamente com os valores associados [CA90].

Além disso, cada mensagem SNMP pode agrupar operações de um mesmo tipo (*Get*, *Set*, *Trap*), permitindo que uma estação de gerenciamento obtenha os valores de todos os objetos

Figura 9 - Formato da mensagem SNMP e os PDUs

escalares de um mesmo grupo de um agente particular.

3.3.4 As Primitivas SNMP

3.3.4.1 Response-PDU

O Response-PDU é gerado por uma entidade SNMP como resposta a uma requisição de GetRequest-PDU, GetNextRequest-PDU ou SetRequest-PDU. Se o campo do *error-status* do Response-PDU for diferente de zero, os valores dos campos da lista de *variable-binding* serão ignorados. Caso os campos *error-status* e *error-index* do Response-PDU sejam diferentes de zero, então o valor do campo *error-index* é o índice da variável cuja requisição falhou. O primeiro índice na lista de *variable-binding* é o 1 (um), o segundo 2 (dois), e assim sucessivamente.

As entidades SNMP que exercem a função de gerente devem estar habilitadas a manipular um Response-PDU com o campo do *error-status* igual a “noSuchName”, “badValue”, ou “readOnly”.

Com o recebimento do Response-PDU, a entidade SNMP gerente apresentará o conteúdo da mensagem à aplicação que gerou a requisição, a qual possui o mesmo *request-id*.

3.3.4.2 GetRequest-PDU

O GetRequest-PDU é gerado e transmitido pelo gerente para requisitar informações aos agentes. O agente SNMP que receber uma mensagem de GetRequest-PDU processará cada variável da lista *variable-binding* e os inserirá em uma nova lista para produzir um Response-PDU.

Todos os campos do Response-PDU possuirão os mesmos valores dos seus correspondentes da requisição, exceto em alguns casos:

Se a variável da lista *variable-binding* possuir um prefixo OBJECT IDENTIFIER que não for acessível pela requisição, então o seu campo *valor* irá receber o valor “noSuchObject”.

Se a variável da lista *variable-binding* não for acessível pela requisição, então o seu campo *valor* receberá o valor de “noSuchInstance”.

Caso contrário, o valor correspondente à variável será atribuído ao campo *valor*.

Se o processamento de qualquer variável da lista *variable-binding* falhar por qualquer outro motivo, o Response-PDU é reformatado com os mesmos valores recebidos pelo GetRequest-PDU, atribuindo-se “genErr” ao campo *err-status* e o índice da variável que falhou ao campo *error-index*.

Se não ocorrer nenhum erro, atribui-se o valor “noError” ao campo *err-status* e zero ao campo *error-index*.

3.3.4.3 GetNextRequest-PDU

O GetNextRequest-PDU é gerado e transmitido por uma aplicação SNMP gerente. Seu tratamento é realizado por um agente SNMP que irá processar cada variável da lista *variable-binding* e as armazenará em uma nova lista para produzir um Response-PDU.

Todos os campos do Response-PDU possuirão os mesmos valores dos campos correspondentes da requisição, exceto em alguns casos onde cada variável da lista *variable-binding* é processada como se segue:

A variável é localizada numa lista ordenada lexicograficamente pelos nomes das variáveis que são acessíveis pela requisição, cujo nome é o primeiro sucessor do nome da variável da lista *variable-binding* do GetNextRequest-PDU.

Se o nome da variável na requisição não for localizado na lista, então a variável da lista *variable-binding* produzida pelo Response-PDU receberá o valor de “endOfMibView”.

Caso o processamento de qualquer variável da lista *variable-binding* falhar por outro motivo, então o Response-PDU é reformatado e atribui-se a seu *request-id* e aos campos das variáveis da lista *variable-binding* valores iguais aos recebidos no GetNextRequest-PDU. O campo *error-status* receberá o valor de “genErr” e os valores dos campos *error-index* das variáveis da lista *variable-binding* que falharam receberão os valores dos índices das variáveis da lista *variable-binding* recebida.

Se nenhum erro ocorrer, o campo *error-status* receberá o valor de “noError”, e os valores dos campos *error-index* serão zero.

3.3.4.4 SetRequest-PDU

O SetRequest-PDU é gerado e transmitido pelo gerente para requisitar que o agente faça alterações em algumas variáveis em sua MIB.

A entidade SNMP que receber o SetRequest-PDU primeiro determinará o tamanho da mensagem, encapsulando o Response-PDU com os mesmos valores dos campos *request-id*, *error-status*, *error-index* e da lista *variable-binding* do SetRequest-PDU. Se o tamanho da mensagem gerada for maior que o limite local ou o máximo suportado pela fonte da requisição, então um Response-PDU alternativo será gerado.

Esse Response-PDU alternativo gerado é formatado com os mesmos valores nos campos *request-id* que os do SetRequest-PDU, com os valores dos campos *error-status* ajustados para “tooBig”, o valor dos campos *error-index* para zero e os campos das variáveis da lista *variable-binding* com vazio.

Após o seu processamento, o Response-PDU é encapsulado dentro de uma mensagem. Se o tamanho da mensagem resultante for menor ou igual ao limite local e ao tamanho máximo que a fonte da requisição suportar, ela será transmitida para o originador do SetRequest-PDU. Senão, a mensagem resultante será descartada.

Caso contrário, a entidade SNMP receptora processará cada variável da lista *variable-binding* para produzir um Response-PDU. Todos os campos do Response-PDU possuirão os mesmos valores dos campos correspondentes aos da requisição recebida, exceto os indicados abaixo. As variáveis da lista *variable-binding* são conceitualmente processadas em duas operações. Na primeira

fase, cada variável da lista *variable-binding* é validada, se todas as validações forem bem sucedidas, então cada variável é alterada na segunda fase. Seguem-se abaixo os testes executados na primeira fase, até que todas as variáveis da lista *variable-binding* sejam testadas com sucesso, ou até que uma falhe:

- Se uma variável da lista *variable-binding* especificar uma variável que não seja acessível por esta requisição, então o valor do campo *error-status* será atualizado para “noAccess”, e o valor do campo *error-index* será atualizado com o índice da variável que falhou.
- Se uma variável da lista *variable-binding* especificar uma variável que não existe, então o valor do campo *error-status* será atualizado para “noCreation”, e o valor do *error-index* será atualizado para o índice da variável que falhou.
- Se uma variável da lista *variable-binding* especificar uma variável que existe, porém não pode ser modificada, então o valor do campo *error-status* será ajustado para “noWritetable”, e o valor do campo *error-index* será atualizado para o valor do índice da variável que falhou.
- Se o valor do campo da variável da lista *variable-binding* especificado segundo o ASN.1 for de um tipo inconsistente com o requerido para a variável, então o campo *error-status* receberá o valor “wrongType”, e o valor do campo *error-index* será ajustado com o índice da variável que falhou.
- Se o valor do campo da variável da lista *variable-binding* especificado segundo o ASN.1, tiver o tamanho inadequado com o requerido pela variável, então o campo *error-status* será ajustado para “wrongLength”, e o valor do campo *error-index* para o valor do índice da variável que falhou.
- Se o valor do campo da variável da lista *variable-binding* for uma codificação ASN.1, inconsistente com os campos *tag* do ASN.1, então o valor do campo *error-status* será ajustado para “wrongEncoding”, e o valor do campo *error-index* para o índice da variável que falhou.

- Se o valor do campo da variável da lista *variable-binding* especificar um valor que não poderia ser associado a essa variável, então o valor do campo *error-status* receberá o valor “wrongValue”, e o valor do campo *error-index* receberá o índice da variável que falhou.
- Se o nome da variável da lista *variable-binding* especificar uma variável que não existe, então o valor do campo *error-status* será de “inconsistentName”, e o valor do campo *error-index* será o índice da variável que falhou.
- Se o campo do valor da variável da lista *variable-binding* especificar um valor que poderia, sob outras circunstâncias, ser associado, mas é no momento inconsistente, então o valor do campo *error-status* será de “inconsistentValue”, e o valor do *error-index* será o valor da variável que falhou.
- Se a associação do valor especificado pelo campo da variável da lista *variable-binding* para a variável especificada requerer a alocação de recursos os quais estão no momento não habilitados, então o valor do campo *error-status* será de “resourceUnavailable”, e o valor do *error-index* será o valor da variável que falhou.
- Se o processamento da variável da lista *variable-binding* falhar por outras razões que não foram mencionadas acima, então o valor do campo *error-status* será de “genErr” e o valor do *error-index* será o índice da variável que falhou.
- Caso contrário, a validação da variável da lista *variable-binding* será aceita.

No final da primeira fase, se todas as validações forem bem sucedidas, então o valor do campo *error-status* do Response-PDU será de “noError”, e o valor do campo *error-index*, zero.

Para cada variável da lista *variable-binding* na requisição, o valor especificado é associado para ela. Cada uma dessas associações de variáveis ocorrerá simultaneamente para todas as associações especificadas na requisição. Mas, se uma mesma variável é nomeada mais de uma vez, com diferentes associações de valores, então somente a última associação será feita na lista *variable-binding*.

Se qualquer uma dessas associações falhar, então todas as associações serão desfeitas, e o Response-PDU será modificado para ter o valor do *error-status* ajustado para “commitFailed”, e o valor do campo *error-index* será o do índice da variável que falhou.

Se e somente se, não for possível desfazer todas as associações, então o Response-PDU será modificado para ter o valor do campo *error-status* ajustado para “undoFailed” e o valor do *error-index* para zero.

E, afinal, se tudo ocorrer normalmente o Response-PDU gerado será encapsulado dentro de uma mensagem, e transmitido para o originador do SetRequest-PDU.

3.3.4.5 Trap-PDU

O Trap-PDU é gerado por uma entidade SNMP que atua como agente e transmitido a um gerente. A transmissão da mensagem será feita periodicamente para manter o gerente informado ou para o caso de ocorrer uma situação anormal.

Esse tipo de mensagem não espera um retorno. O destino para o qual um Trap-PDU é enviado é determinado através de uma consulta à tabela *aclTable*.

3.3.5 Limitações do Protocolo SNMP

Ben-Artzi, Chandna e Warriier (1990) identificaram as potenciais limitações que recaem sobre o protocolo SNMP [ST93]:

- Não é adequado para o gerenciamento de grandes redes de computadores, dado ao fato das limitações de desempenho para o *polling*.
- protocolo não é adequando para a aquisição de grandes quantidades de informações.
- Provê apenas um mecanismo rudimentar de autenticação, sendo assim mais adequado ao monitoramento do que ao controle.
- Não suporta comandos imperativos.
- Não dá suporte à comunicação gerente-a-gerente, sendo impossível o intercâmbio de informações entre as várias estações de gerenciamento.

3.4 Simple Network Management Protocol version 2 (SNMPv2)

A primeira versão do SNMP apresentava-se como um protocolo restrito, ineficiente e inseguro, se exposto a ambientes relativamente grandes e complexos. Para minimizar esses problemas foi elaborada uma evolução do protocolo que conhecemos por SNMP versão 2 [CA93a].

O protocolo SNMPv2 suporta plenamente tanto uma estratégia de gerenciamento centralizada quanto distribuída, podendo algumas entidades desempenharem tanto as funções de gerente como as de agente simultaneamente.

A estrutura de informação de gerenciamento (SMI), que é baseada no SMI para SNMPv2, possui uma especificação mais elaborada, eliminando as ambigüidades nas definições dos objetos encontradas na especificação anterior. Essas alterações exercem influência direta sobre a implementação das MIBs SNMPv2 e seus esquemas de funcionamento [CA93b].

O protocolo SNMPv2 provê três tipos de operações de acesso às informações de gerenciamento [CA93e]:

- Requisição/Resposta Gerente-Agente: Uma entidade cumprindo o papel de gerente envia uma mensagem de requisição à entidade agente que por sua vez atende e envia a resposta.
- Requisição/Resposta Gerente-Gerente: Uma entidade gerente envia uma mensagem com informações a outra entidade gerente, que por sua vez retorna uma confirmação.
- Agente/Gerente não confirmada: Uma entidade SNMPv2, no papel de agente, envia uma mensagem, Trap, a uma outra entidade que esteja cumprindo o papel de gerente, que não envia uma mensagem de resposta.

Pode-se observar que somente o segundo item apresenta uma característica inovadora no protocolo SNMPv2. Foram também incorporadas novas primitivas operacionais:

- GetBulkRequest-PDU: Esta primitiva representa uma das maiores melhorias no protocolo SNMPv2. Através dela é possível especificar múltiplos sucessores lexicográficos para serem selecionados, aumentando-se a eficiência do protocolo perante as requisições para grandes quantidades de informações. Para isto, a GetBulkRequest-PDU contém dois campos adicionais: “nonrepeaters” e “maxrepetitions”, responsáveis sucessivamente por identificar o número de variáveis que deverão ser tratadas normalmente (N) e a quantidade máxima de sucessores lexicográficos retornados para cada variável restante.

- **InformRequest-PDU:** É enviado por uma entidade SNMPv2, que esteja desempenhando o papel de gerente, a uma outra entidade atuando como gerente. A mensagem é gerada sem a necessidade de uma requisição, como o que ocorre com o Trap-PDU, porém o InformRequest-PDU espera por uma mensagem de Respose-PDU.

Outra melhoria realizada é na forma como as respostas são tratadas na versão SNMPv2 - mesmo que um dos valores das instâncias dos objetos não possa ser processado, ainda assim um Response-PDU é gerado pelo agente SNMPv2 e conseqüentemente, devolvido à entidade gerente. O processamento das variáveis é efetuado mediante a aplicação das seguintes regras: no caso da inexistência do prefixo de identificação do objeto requerido, o valor do campo de resposta correspondente é preenchido com “noSuchObject”; no caso da inexistência da variável requerida, o valor do campo correspondente será preenchido com “noSuchInstance”; caso contrário, o valor do campo é preenchido com o valor da variável requisitada.

Se por algum motivo não for possível processar o valor da variável requisitada, então o campo correspondente na mensagem de resposta é deixado em branco. Outro problema que pode ocorrer está relacionado ao limite máximo imposto ao tamanho da mensagem de resposta. Caso este limite exceda, o Response-PDU será descartado e um novo será constituído contendo um sinalizador de erro “tooBig” e os campos das variáveis de amarração serão deixados vazios.

Basicamente as capacidades de comunicação Gerente-Gerente adicionada ao protocolo SNMPv2 recaem sobre a definição de objetos que compõem uma MIB específica. A MIB Gerente-Gerente, cujos objetos são responsáveis por descreverem o comportamento de uma entidade SNMPv2 agindo no papel de gerente. Esta MIB é constituída por dois grupos [CA93c]:

- **Grupo de Alerta:** Uma coleção de objetos que permite a descrição e configuração dos limites de alerta de uma entidade SNMPv2 que esteja atuando tanto como agente como gerente. O grupo *snmpAlarm* da MIB Gerente-Gerente consiste de uma simples tabela, *snmpAlarmTable*, e de uma variável escalar simples, *snmpAlarmNextIndex*, cuja finalidade é indicar o número do índice seguinte disponível para a criação de tuplas.
- **Grupo Eventos:** Uma coleção de objetos que permite a descrição e configuração dos eventos de uma entidade SNMPv2 que esteja atuando como agente e gerente. O grupo *snmpEventTable* é responsável por suportar as definições dos eventos e as configurações das notificações. Consiste de duas tabelas, *snmpEventTable* e *snmpEventNotifyTable*, mais os

objetos escalares associados. Cada tupla na tabela *snmpEventTable* define um evento que pode ser disparado por uma condição associada na tabela *snmpAlarmTable*, ou através das condições definidas na macro NOTIFICATION-TYPE.

A tabela *snmpEventNotifyTable* define as notificações que deverão ocorrer quando um evento associado for disparado. As tuplas desta tabela descrevem os tipos e o destino para um InformRequest-PDU enviado em um evento particular. A tabela é indexada por dois objetos não colunares desta tabela: *snmpEventIndex* identifica o *snmpEventEntry* que, quando engatilhado, gerará uma notificação como configurada nesta tupla; *partyIdentify* identifica a entidade par de destino para a qual a notificação será enviada.

O protocolo SNMPv2 supre as necessidades do ponto de vista de segurança, implementando mecanismos adequados. Inclui o conceito de visão de MIB, definida em termos de uma coleção de subárvores, onde cada subárvore pode ser incluída ou excluída de uma visão [GA93b].

Inclui também o conceito de contexto, que nada mais é do que uma coleção de objetos gerenciados por uma entidade SNMPv2, podendo ser acessado localmente ou remotamente. Um contexto local é identificado como uma visão MIB e pode ser acessado mediante os mecanismos de acesso local. Um contexto remoto é identificado como um relacionamento de procuração, onde a entidade SNMPv2 desempenha o papel de um agente *proxy* de modo a obter acesso às informações de gerenciamento identificadas pelo contexto SNMPv2.

Os contextos SNMPv2 cumprem as funções de relacionar o controle de acesso às visões de MIBs. Tanto no inter-relacionamento entre as entidades SNMPv2 gerentes e agentes, quanto entre entidades gerentes e agente proxy, os privilégios de controle de acesso selecionado são especificados pelo contexto associado.

Com o uso de contextos, uma abordagem de controle de acesso pode ser composta por quatro elementos:

- Alvo: Uma entidade SNMP efetua as operações de gerenciamento requisitada por uma entidade sujeito.
- Sujeito: Uma entidade SNMP origina requisições de operações de gerenciamento para serem atendidas por uma entidade alvo.

- Recursos: As informações de gerenciamento nas quais essas operações podem ser efetuadas, expressas como uma visão local de MIB ou um relacionamento “proxy”;
- Privilégios: São as operações permitidas, definidas em termo dos PDUs pertencentes a um determinado contexto.

Neste caso, a política de controle de acesso pode ser definida por três parâmetros. Uma entidade *sujeito* requisita uma *operação de gerenciamento* a uma entidade *alvo* que identifica o contexto da requisição, que pode especificar uma visão de MIB local ou uma entidade remota, atendida por procuração. Para um dado par de entidades sujeito/alvo, pode haver políticas múltiplas de controle de acesso, um para cada contexto. O contexto é comunicado por uma entidade alvo através do cabeçalho de mensagem SNMPv2.

As principais características de segurança adicionadas ao SNMPv2 estão relacionadas aos serviços de integridade, autenticação e confiabilidade da informação.

- Integridade dos Dados: Com o objetivo de assegurar que a mensagem original não venha a sofrer alterações por entidades externas não autorizadas, o SNMPv2 utiliza um mecanismo de assinatura para mensagem, onde o algoritmo é responsável por calcular uma assinatura de 128-bits sobre uma porção apropriada da mensagem SNMPv2. Essa assinatura é incluída na mensagem para assegurar que não haja modificações. Além disso, a entidade SNMPv2 que recebe a mensagem utiliza-se do “timestamp” incluído pelo remetente para verificar se a mensagem é recente e determinar a sequência correta de mensagens múltiplas.
- Autenticação: Refere-se ao propósito de assegurar que as mensagens sejam recebidas como foram enviadas; para isso utiliza-se do mecanismo de assinatura empregado pelo serviço de integridade dos dados. A assinatura de uma mensagem é calculada sobre uma porção apropriada da mensagem SNMPv2.
- Confiabilidade dos Dados: Para assegurar que os dados não sejam acessíveis pelas entidades externas não autorizadas, uma porção apropriada de uma mensagem SNMPv2 é criptografada usando um algoritmo de criptografia simétrico. O SNMPv2 utiliza o algoritmo DES (*Data-Encryption Standard*).

3.4.1 Mensagens SNMPv2

As entidades SNMPv2 que estiverem atuando como agentes deverão estar habilitadas a gerar o Responde-PDU e SNMPv2-Trap-PDU e capacitadas a tratar o GetRequest-PDU, GetNextRequest-PDU, GetBulkRequest-PDU e Set Request-PDU.

Já as entidades SNMPv2 que estiverem atuando como gerentes deverão estar habilitadas a gerar o GetResquest-PDU, GetNextRequest-PDU, SetRequest-PDU, InformRequest-PDU e Response-PDU e capacitadas a receberem o Response-PDU, SNMPv2-Trap-PDU e o InformRequest-PDU.

O tamanho das mensagens SNMPv2 é limitado pelo:

- Tamanho máximo da mensagem que a entidade SNMP destino pode suportar, indicado pelo campo *partyMaxMessage*.
- Tamanho máximo da mensagem que a entidade fonte pode gerar, imposto por uma limitação local.

Cada *Transport Mapping*, datagramas, para o SNMPv2 indica o tamanho máximo que uma implementação SNMPv2 está habilitada a consumir e produzir. Embora as implementações possam suportar o máximo da capacidade possível, uma implementação segura não deverá alcançar tais limites. Formato da mensagem SNMPv2 e os novos PDUs (Fig. 10):

Mensagem				
PrivDestino	Informações de Autorização	Destino	Fonte	PDU
InformRequest-PDU				
PDU Type	Request-ID	0	0	Variable-Bindings
GetBulkRequest-				
PDU Type	Request-ID	Non-repeaters	Max-repetitions	Variable-Bindings

Figura 10 - Formato das mensagens SNMP e os PDUs

3.4.2 Processamento das novas Primitivas

3.4.2.1 GetBulkRequest-PDU

GetBulkRequest-PDU é gerado e transmitido por uma aplicação SNMPv2 que atua como gerente. O propósito do GetBulkRequest-PDU é o de requisitar as transferências de grande quantidade de dados, que deverão estar organizados na forma de tabela. A entidade receptora do GetBulkRequest-PDU processará cada variável da lista *variable-binding* e inserirá em uma nova lista *variable-binding*, produzindo o Response-PDU.

O campo *request-id* do Response-PDU receberá o mesmo valor do GetBulkRequest-PDU recebido. O processamento começará examinando os valores dos campos *non-repeaters* e *max-repetitions*. Se o valor do campo *non-repeaters* é menor que zero, então será atribuído o valor zero ao campo, o mesmo ocorrendo com o *max-repetitions*.

Os valores dos campos *non-repeaters* e *max-repetitions* na requisição especificarás o processamento requerido. Uma única variável da lista *variable-binding* no Response-PDU é requisitada para as primeiras N ; e M variáveis da lista *variable-binding* são requisitadas para cada R variável restante da requisição. Consequentemente, o número total de requisições é dado por $N + (M * R)$, onde N é o mínimo do valor do campo *non-repeaters* na requisição e o número da *variable-binding* na requisição; M é o valor do campo *max-repetitions* na requisição; R é o máximo entre o número de variáveis requisitadas menos o N e zero.

Se N é maior que zero, as primeiras N variáveis da requisição do Response-PDU serão produzidas da mesma forma que no GetNextRequest.

Se o M e o R forem diferentes de zero, as variáveis $(N+1)$ em diante da requisição serão processadas de forma similar. Para cada interação i , de modo que i seja maior que zero e menor que ou igual a M , e para cada variável repetida, r , de modo que r é maior que zero e menor ou igual a R , os $(N + ((i-1) * R) + r)$ variáveis da lista *variable-binding* do Response-PDU serão produzida como se segue:

A variável que está na lista, ordenada lexicograficamente dentre os nomes de todas as variáveis que são acessíveis pela requisição e é o i -ésimo sucessor lexicográfico do $(N+r)$ -ésimo nome de variável recebida pelo GetBulkRequest-PDU, é localizada, e o nome da variável da lista *variable-binding* e o valor do campo receberão o valor do nome e o valor da variável localizada.

Se não existir o i -ésimo sucessor lexicográfico, então a variável da lista *variable-binding* correspondente produzida no Response-PDU terá os valores dos seus campos ajustados para “endOfMibView”, e os nomes dos campos receberão os valores do último sucessor lexicográfico.

Enquanto o número máximo de variáveis na lista *variable-binding* no Response-PDU for limitado por $N + (M \cdot R)$, a resposta gerada poderá possuir um número menor de *variable-binding* (possivelmente zero) por duas razões:

Se o tamanho da mensagem encapsulada do Response-PDU contiver o número de variáveis maior que um limite local ou maior que o tamanho máximo da mensagem da fonte da requisição, uma resposta será gerada com um número menor de variáveis.

Esse menor número é um conjunto ordenado das variáveis da lista *variable-binding* com algumas das variáveis da lista *variable-binding* do conjunto recebido.

Deste modo, o tamanho da mensagem encapsulado do Response-PDU é aproximadamente igual, mas nunca não maior que o limite local e que o tamanho máximo de mensagem da fonte da requisição.

Note-se que o número de variáveis removidas da lista *variable-binding* e os valores de N , M e R não possuem qualquer relação.

A resposta também pode ser gerada com um número menor de variáveis, se alguns valores da interação I tiverem os valores dos campos atualizados para “endOfMibView”. Neste caso, a lista *variable-binding* pode ser truncada depois de $(N + (i \cdot R))$ variáveis.

Se o processamento de qualquer variável da lista *variable-binding* falhar por qualquer outra razão além as listadas acima, então o Response-PDU é reformatado com os mesmos valores do GetBulkRequest-PDU recebido, com o valores dos campos *error-status* e *error-index* ajustados respectivamente para “genErr” e para o índice da variável que falhou. Caso contrário, o valor do campo *error-status* do Response-PDU será ajustado para “noError”, e o valor do campo *error-index* para zero.

O Response-PDU gerado é então encapsulado dentro da mensagem. Se o tamanho da mensagem resultante for menor ou igual ao limite local e ao tamanho máximo de mensagem que a

fonte suporta, ela é transmitida para o originador do GetBulkRequest-PDU. Caso contrário a mensagem é descartada.

3.4.2.2 InformRequest-PDU

Um InformRequest-PDU é gerado e transmitido por uma entidade SNMPv2 gerente, com o objetivo de enviar informações de gerenciamento a uma outra entidade SNMPv2 gerente. O destino para o qual o InformRequest-PDU será enviado dependerá de uma inspeção na tabela *snmpEventNotifyTable*.

As primeiras duas variáveis da lista *variable-binding* do InformRequest-PDU são *sysUpTime.0* e *snmpEventID.i* respectivamente. Se a cláusula OBJECTS estiver presente na invocação de uma macro NOTIFICATION-TYPE, então cada variável correspondente, instanciada por esta notificação, será copiada, em ordem, para o campo da variável da lista *variable-binding*.

A entidade SNMPv2 que receber o InformRequest-PDU irá encapsular um Response-PDU com os mesmos valores dos campos *request-id*, *error-status*, *error-index* e da lista *variable-binding* do InformRequest-PDU recebido.

Se o tamanho da mensagem for maior que o limite local ou maior que o tamanho máximo do que o fonte suporta, então um Response-PDU alternativo é formatado com os mesmos valores dos campos do InformRequest-PDU recebido, com o valor do campo *error-status* atualizado para “tooBig”, o do campo *error-index* para zero, e um campo vazio na lista *variable-binding*.

Esse Response-PDU alternativo é então encapsulado dentro da mensagem. Se o tamanho da mensagem resultante for menor ou igual ao limite local e ao tamanho máximo suportado pela fonte da requisição, ele será transmitido para o originador do InformRequest-PDU. Caso contrário, a mensagem resultante será descartada.

3.4.5 Arquiteturas de Gerenciamento Alternativas Baseado no Protocolo SNMPv2

Serão apresentadas algumas arquiteturas de gerenciamento de redes alternativas, que utilizam o protocolo SNMPv2 para o tráfego de informações de gerenciamento e suas novas características para melhorarem a qualidade e desempenho dos seus sistemas.

3.4.5.1 Arquitetura de Gerenciamento para Sistemas Distribuídos

Utilizando-se da estrutura dos Sistemas Distribuídos, dividem-se as tarefas de gerenciamento em domínios [SL93], criando-se grupos de objetos gerenciados. Dessa forma cada gerente do sistema terá a função de gerenciar certos domínios.

O modelo de gerenciamento segue a estrutura mostrado na figura 11 (Fig. 11): Uma política

Figura 11 - Política de Gerenciamento

de gerenciamento é adotada pelo gerente, que é aplicada sobre os recursos por ele gerenciados.

Destaca-se com esta estrutura a capacidade de alterar a política de gerenciamento em *run-time* (tempo de execução), sem a necessidade de interromper todo o sistema ou reiniciá-lo. Com a comunicação Gerente-Gerente, incorporada ao protocolo SNMPv2, tem-se um ambiente onde cada gerente do sistema executará a função de um Gerente-Proxy, havendo pelo menos um gerente que irá gerenciar os Gerentes-Proxy.

Figura 12 - Problemas de gerenciamento no modelo convencional

3.4.5.2 Arquitetura de Gerenciamento Hierárquico

Normalmente utiliza-se de uma estrutura hierárquica para organizar a rede, dividindo-a em subredes, porém a estrutura utilizada para o gerenciamento de rede é uma estrutura plana, que traz algumas dificuldades (Fig. 12):

Através da figura apresentada pode-se identificar alguns problemas:

Problemas com o Roteador (A): Uma falha no roteador entre o gerente e os agentes implicará no não gerenciamento do mesmo. Enquanto o roteador não estiver ativo, todas as informações dos agentes não estarão disponíveis, sendo que para o gerente não é somente o roteador que não está ativo, mas toda a rede que se segue.

Baixo *Bandwidth* (B): O gerenciador não gerenciará apenas redes locais (LAN), mas poderá gerenciar agentes conectados através de uma conexão de longa distância (WAN), que normalmente utilizam-se de conexões de baixa velocidade. O monitoramento intensivo desses agentes poderá causar uma perda considerável de *bandwidth* e em muitos casos o custo da troca da conexão por uma mais rápida para melhorar o gerenciamento não se justificaria.

Qualidade de Serviço (C e D): A interface SNMP e as aplicações necessitam de tempo de CPU para processarem suas tarefas. Com o aumento das redes, aumentará o número de agentes e consequentemente aumentará a quantidade de informações de gerenciamento a serem processadas, degradando o desempenho da estação gerenciadora.

Uma forma de resolver esses problemas seria utilizar uma arquitetura de gerenciamento hierárquico [SC95], distribuindo o papel de gerente entre outras estações, Sub-Gerentes, que estariam localizadas em posições estratégicas, eliminando os problemas causados pelo isolamento, “gargalos” causados pelo *bandwidth* e melhorando a qualidade de serviços prestados pelas estações de gerenciamento.

Esse tipo de estrutura utiliza-se da comunicação Gerente-Gerente do protocolo SNMPv2, com algumas modificações na MIB-Gerente, sendo os Sub-Gerentes gerenciados por um Master-Gerente (Fig. 13).

Figura 13 - Uso de subgerentes no gerenciamento de redes

3.5 Busca do padrão para a segurança do protocolo SNMPv2

Devido as dificuldades de se definir um padrão *de fato* para as áreas de segurança, estrutura administrativa e configuração remota do protocolo SNMPv2, o *IETF SNMP Working Group* em setembro de 1995 definiu:

1. Abandonar as especificações para segurança, estrutura administrativa e configuração remota;
2. Publicou uma atualização para apenas parte do protocolo SNMPv2 (RFCs de 1902 à 1908) [CA96b,CA96c,CA96d,CA96e,CA96f,CA96g,CA96h], tornando-os como *Drafts Standards*;
3. Alterou o *status* dos RFCs 1445-1447, que descrevem os aspectos de segurança, estrutura administrativa e configuração remota, de *Drafts Standards* para *Historic*, e
4. Publicou especificação experimental para a estrutura administrativa conhecido como SNMPv2c (RFC 1901) [CA96a], baseado nas especificações do SNMPv1, descartando qualquer modelo de segurança.

Em dezembro de 1995 o *IETF SNMPv2 Working Group* é desativado e grupos independentes são formados para estudos e experimentos de alternativas que pudessem preencher as lacunas deixadas no protocolo SNMPv2. As últimas especificações definidas pelo *IETF* deixavam o

protocolo com as mesmas deficiências do SNMPv1, impedindo o desenvolvimento de ferramentas que gerenciam de rede que executassem além das funções de monitoração, executassem as funções de controle.

Dentre as propostas feitas pelos grupos independentes, duas se destacaram SNMPv2u e SNMPv2* (SNMPv2 *Star*), ambas baseadas no modelo definido no *User-based Security Model for SNMPv2* (RFC 1910) [CA96i], sendo assim possuem muitos pontos em comum nos projetos da estrutura administrativa mas com filosofias diferentes.

O SNMPv2u suporta o mais atualizado modelo segurança e com uma especificação mínima, não se preocupou com os aspectos relacionados aos gerenciamento de grandes redes para encorajar o seu rápido desenvolvimento. O SNMPv2* suporta os atuais padrões determinados para a segurança, mas preparado para receber possíveis atualizações assegurando sua escalabilidade, e a sua estrutura está adequada para o uso de agentes proxys, habilitando-o para o gerenciamento de média à grande redes.

Em agosto de 1996, uma equipe foi formada, *Security and Administrative Framework Evolution Advisory*, para desenvolver e publicar uma recomendação de um único documento que eliminaria as diferenças entre os modelos SNMPv2u e SNMPv2*, em buscava-se um padrão que deveria ser seguido por todos. No encontro realizado pelo IETF realizado em San Jose/Califórnia em dezembro de 1996 a equipe apresentou estas recomendações. E em março de 1997, um novo grupo de trabalho foi formado, *SNMPv3 Working Group*, objetivando dar continuidade aos trabalhos de desenvolvimento do protocolo SNMPv2, definindo padrões para a segurança e estrutura administrativa do protocolo, baseado nos estudos realizados no SNMPv2u e SNMPv2*.

3.6 O desenvolvimento do padrão SNMPv3

O desenvolvimento da versão 3 do protocolo SNMP [SI97], baseia-se nos RFCs 1902-1908, adicionado das especificações, conceitos e técnicas de segurança, e controle remoto desenvolvidos nos SNMPv2u e SNMPv2*, reutilizando as documentações já existentes.

Para o sucesso da nova especificação, o resultado final dos documentos deveria ter como principal objetivo definir:

- Segurança: possibilitando o uso de comando de controle (*SET*) sobre as operações na rede, que é a principal deficiência do SNMPv1.

- Melhorias para suportar o gerenciamento de redes de grande porte, possibilitando o uso de agentes *proxys*.
- O protocolo deveria ser modular, de modo a possibilitar a incorporação de novas tecnologias que possam surgir.
- Os documentos devem ser claros e sem ambigüidades.
- A manutenção e configuração dos aspectos de segurança e administrativos deveriam ser fáceis.
- O protocolo deveria manter, quanto possível, a mesma simplicidade que o tornou um padrão *de fato* de mercado.

3.6.1 Uma nova arquitetura para o SNMP

Para o desenvolvimento de uma nova arquitetura do protocolo *SNMP*, tentou-se utilizar o máximo do material já existente das pesquisas realizadas (*SNMPv2u* e *SNMPv2**), para sanar os principais problemas das versões anteriores (*SNMPv1* e *SNMPv2c*): segurança e controle remoto dos dispositivos.

O desenvolvimento de uma estrutura modular é uma das principais características para torná-la mais tolerável e adaptável aos requisitos de um ambiente tão variado e sujeito a mudanças como o ambiente de rede. Como proposta a esta nova estrutura tem-se um novo modelo para a entidade *SNMP* (Fig. 14), que será constituído de um *SNMP engine* e um módulo de aplicações associadas a esta entidade.

O *SNMP engine* é constituído dos serviços de recebimento e envio de mensagens, autenticação e criptografia, e controle de acesso aos objetos gerenciáveis, servido pelos seguintes sub-sistemas:

- *Dispatcher*: Coordena a comunicação entre os sub-sistemas, ele irá permitir que a entidade *SNMP* possa comunicar-se com múltiplas versões;
- *Message Processing Modules*: Processa as mensagens *SNMP*, extraindo as informações das mensagens recebidas ;

- *Security Modules*: Baseado nos protocolos SNMPv2u e SNMPv2*, iniciará a inclusão de segurança nas mensagens SNMP com mecanismos de autenticação e criptografias que forem sendo disponibilizadas no sistema de gerenciamento.
- *Access Control Modules*: Com a utilização de sistemas de segurança, este modulo será responsável por autenticar ou aceitar o controle de acesso servidos pelo protocolo.

SNMP Entity

Figura 14 : Entidade SNMPv3

O Módulo Aplicações será composto de vários tipos de aplicações, que farão uso do *Engine SNMP*. E dependendo das funções adicionadas as entidades SNMP serão agentes ou gerentes dependendo. Com esta nova configuração espera-se conseguir dar maior longevidade à estrutura, tornando o tão simples quanto foi a sua primeira versão, que o fez tornar-se um padrão *de fato*.

Novas Tecnologias de Interesse à Implementação de Protocolos de Gerenciamento

O avanço tecnológico da infra-estruturas de conexão de rede de computadores (maior capacidade de transmissão) , e equipamentos mais rápidos e baratos, possibilitou e determinou o desenvolvimento de novas tecnologias (*WWW*, *Java*, *CORBA*, ..), que possuem como principal característica disponibilizar ao usuário final aplicativos mais interativos e de fácil compreensão.

Neste sentido, a tecnologia da informação foi a área que mais se desenvolveu, no qual destacam-se: *World Wide Web* (*WWW*), protocolos de comunicação, multimídia e segurança.

4.1 *HyperText Transfer Protocol (HTTP)*

Talvez os maiores responsáveis pela imensa proliferação do uso destas novas tecnologias são a Internet/Intranet e o *WWW*, que possuem em comum o protocolo de comunicação *HTTP*. É um protocolo aberto que foi especificado pelo *IETF* [F197], baseado na arquitetura *TCP/IP* e definido à nível de aplicação, utiliza como protocolo de transporte o *UDP*.

A “*Web*” consiste de um grande sistema baseado no modelo cliente-servidor, no qual todas as transações são do tipo requisição-resposta. Os clientes sempre estarão requisitando informações (documentos) e os servidores sempre que possível retornando as informações requisitadas (Fig. 15). Normalmente, os clientes utilizam como aplicativo de interface os softwares comumente chamados de navegadores (*Internet Explorer*, *Netscape*, ...), que possuem como características comuns: operar com o protocolo de comunicação *HTTP* e capacidade de interpretar documentos *HyperText Markup Language* (*HTML*). Desta forma, disponibilizando uma interface padrão independente do sistema em que esteja sendo executada a aplicação.

Figura 15 : Processo de Comunicação entre Cliente e Servidor Web

4.1.1 *HyperText Markup Language (HTML)*

O padrão HTML (Hypertext Markup Language) tornou-se amplamente utilizado como formato para hipertextos disponibilizados na *Web*, e é formalmente definido por um *DTD*, que define as regras para a especificação de uma classe de documentos, e *SGML* (*Standard Generalized Markup Language*). Desse modo, define como um conjunto de elementos deve ser estruturado para compor um documento, permitindo também entre seus elementos, elos de hipertexto e inclusão de outras mídias.

Uma das mais recentes melhorias ao *HTML* e aos navegadores foi incorporar a capacidade de executar uma *Applet Java*, programa transferido dinamicamente através da rede da mesma forma que um arquivo padrão e apresentado pelos navegadores, que os executam através do *Java Virtual Machine*. A diferença consiste que um *Applet Java* corresponde ao código de programa, não apenas uma animação ou um arquivo de formato de mídia, logo é um programa que pode reagir a ações do usuário e modificar-se dinamicamente [NA96]. Uma referência a uma applet é embutida em um documento *HTML* utilizando-se de um *tag* especial.

4.3 *Java*

No ano de 1991, a *Sun Microsystems Inc.* iniciou o desenvolvimento da linguagem de programação que a tornaria um marco no processo de desenvolvimento de software aplicativos. Baseado no modelo orientado ao objeto, destaca-se das demais linguagem de programação por possuir as seguintes características:

Uma linguagem simples de se entender e de aprender, contrastando com outras linguagens de desenvolvimento orientado a objetos, por exemplo o C++.

- Linguagem que segue e integra todos os conceitos do modelo orientado ao objetos [BO91];
- Linguagem neutra, ou seja, os programas desenvolvidos em *Java* poderá ser executado independente da plataforma. Foi uma das principais características, responsável pela rápida aceitação no mercado. O código gerado pelo compilador, bytecode, será processado por um interpretador que foi desenvolvido para cada plataforma, *Java Virtual Machine* [L197].
- Código transportável, característica que possibilitou integrar o *Java* às páginas *WWW*, está integração é conhecido por *Java applets*. Estes *Java applets* serão executados pelos navegadores que possuem um *Java Virtual Machine* que interpretará os códigos *Java*. Embora seja uma linguagem interpretada há a possibilidade de a partir dos *bytecodes* seja gerado um novo arquivo com os códigos nativos do equipamento.
- Própria para o desenvolvimento de aplicações distribuídas, possuindo uma extensa quantidade de bibliotecas prontas para a implementação de aplicativos de rede.
- Por *Java* ter sido desenvolvido para ambientes de rede, os recursos de segurança receberam muita atenção. A eliminação do uso de ponteiros, em favor do uso de vetores, objetos e outras estruturas substitutivas traz benefícios em termos de segurança. O programador é proibido de obter acesso a memória que não pertence ao seu programa, além de não ter chances de cometer erros comuns tais como “reference aliasing” e uso indevido de aritmética de ponteiros. Estas medidas são particularmente úteis quando pensarmos em aplicações comerciais desenvolvidas para a internet.
- Muitas aplicações necessitam que alguns de seus processos executem “simultaneamente”, o *Java* possui a habilidade de trabalhar com *Multithreads*, ou seja uma aplicação pode executar vários *threads* (processos) ao mesmo tempo. Recurso pode ser melhor explorado nas versões mais atualizadas dos sistemas operacionais (Unix, Windows e Mac).
- Resolução de dependências em tempo de execução. Esta característica é de extrema importância, pois permite não apenas a carga dinâmica de classes como também a

utilização de referência cruzadas nas definições das mesmas, ou seja, classes distintas podem possuir referências entre si.

4.3.1 Plataforma Java

A plataforma Java é composto por dois módulos principais, *Java Virtual Machine* e *Java API* (Fig. 16).

Figura 16 : Plataforma Java

- *Java Virtual Machine*: é a chave da independência da plataforma e do sistema operacional, no qual exercerá o papel de um interpretador de *Java bytecode* para cada plataforma em que tiver sido desenvolvido. Tornando desta forma o *Java bytecode* portátil à qualquer plataforma que implemente o *Java Virtual Machine*., eliminando a necessidade de rescrever os códigos da programa.
- *Java API* – Forma uma interface padrão para applets e aplicativos, independente do sistema operacional, sendo a estrutura essencial para o desenvolvimento das aplicações. A *API* especifica um conjunto de interfaces essenciais em um número crescente de áreas chaves que os desenvolvedores poderão utilizar para construir suas aplicações *Java*, assegurando que suas aplicações irão executar em qualquer lugar.

4.3.2 Características Adicionais do Java

Novas características estão sendo acrescentadas às APIs padrão do Java, principalmente relacionadas a conectividade à banco de dados remotos e ao desenvolvimento de aplicações de sistemas distribuídos.

- Java Database Connectivity (JDBC): É API desenvolvida pela *JavaSoft* para disponibilizar uma interface padrão para a conectividade a bancos de dados relacional *SQL*. As grandes empresas que desenvolvem sistemas de gerenciadores de banco de dados já disponibilizam driver para *JDBC*. Estes drivers são adicionados à uma lista dos drivers que poderão ser gerenciados pela classe *DriverManager* que controla a comunicação com o banco de dados.
- Remote Method Invocation (RMI): Uns dos mecanismos para a comunicação objeto-a-objeto, trabalhando diretamente com os objetos *RMI* existentes. O *RMI* é como um *Remote Procedure Call*, (*RPC*), um objeto faz uma chamada a um método num objeto em outra máquina e obtém um resultado. Assim como na maioria dos sistema *RPC*, a *RMI* requer que o objeto cujo método está sendo invocado esteja ativo e rodando.

4.4 Common Request Broker Architecture (CORBA)

Em busca de nova arquitetura que permitisse utilizar o máximo de recurso que as novas estruturas de rede estariam disponibilizando, o *Object Management Group* (*OMG*) propôs a especificação do *Common Object Request Broker Architecture* (*CORBA*) [BE95], que de uma forma simples, permite a comunicação entre os objetos de aplicações, independentes de suas localidades. Junto à *CORBA*, definiu-se a *Interface Definition Language* (*IDL*) e as APIs necessárias para desenvolver aplicações distribuídas, este modelo é de fato padrão, líder para o desenvolvimento para sistemas distribuídos heterogêneos.

O relacionamento entre os objetos distribuídos é realizado através de uma nova arquitetura cliente-servidor, onde o há a presença dos ORB (*Object Request Broker*) como agente intermediário. Através dele o cliente pode executar um método de um objeto local ou remoto qualquer. De fato, o ORB intercepta a chamada a um método, localiza o objeto destino que o implementa, executa e por fim retorna os resultados. Tudo isso acontece de forma totalmente transparente, sem a necessidade do cliente saber onde o objeto destino encontra-se, linguagem de programação empregada em sua

implementação, sistema operacional ou plataforma onde reside, ou qualquer outro aspecto relacionado à interface com o objeto.

4.4.1 CORBA e Java

Da união destas duas tecnologias surgiu uma solução muito interessante para sistemas cliente-servidor, baseado sobre um sistema distribuído heterogêneo, resolvendo problemas técnicos apresentados por outros tipos de implementação e que simplificam muito o desenvolvimento.

Para que os clientes *Java* estejam habilitados para *CORBA* foi desenvolvida além de uma *API IDL*, as ferramentas básicas para gerar o ambiente necessário para habilitar a comunicação, por exemplo *idlgen*, responsável por gerar os *stubs* clientes. Uma das grandes vantagens deste modelo de solução *Java/CORBA* sobre *Java/RMI* é que com o *CORBA* pode-se trabalhar com outras linguagens de programação, podendo por exemplo: invocar um método num aplicativo C++ rodando no servidor a partir de um *applet*.

Adaptando a nova estrutura SNMPv3 ao sistema NetTracker

O sistema de gerenciamento de redes em desenvolvimento no Instituto de Ciências Matemáticas de São Carlos – USP, *NetTracker*, sempre se caracterizou por possuir características especiais que o mantêm atualizado com as mais recentes avanços tecnológicos. Por exemplo, a possibilidade de incorporar informações multimídia ao sistema de gerenciamento.

Com o objetivo de mantê-lo atualizado com as novas tecnologias em desenvolvimento (*Java*, *WWW*, *CORBA*), foram realizados estudos para o desenvolvimento para uma nova estrutura organizacional do sistema [MO97], tecnologicamente superior e mais flexível, de modo a facilitar o relacionamento de comunicação entre os módulos que compõe o ambiente de gerenciamento, disponibilizando uma interface padronizada.

5.1 A Nova estrutura do NetTracker

Pode-se observar na Figura 17 o relacionamento de comunicação entre os módulos que compõem o novo ambiente gerenciado pela ferramenta *NetTracker*. É importante notar que a interface do gerente cumpre não apenas o papel de agente difusor de informação, mas também o papel de elemento descentralizador de gerenciamento.

Figura 17: Relacionamento de comunicação intramódulos

Em termos arquiteturais, a Figura 18 apresenta uma visão, em alto nível, do sistema *NefTracker*, que é composta por cinco camadas interdependentes: *Módulos de Programas de Aplicação (NAPM)*, *Interface de Extensão de Aplicações (NEAPI)*, *Módulos de Extensão (NEM)*, *Biblioteca de Objetos e Máquina Virtual Java*. Estas camadas possuem, cada qual, papéis funcionais bem definidos, e que são sumariados a seguir:

5.1.1 Camada de Módulos de Programas de Aplicação

A camada MAPM trata dos aspectos funcionais agregados ao sistema *NefTracker*, incorporando as aplicações responsáveis por proverem as funções de gerenciamento e de comunicação com o administrador. Como exemplos de elementos componentes desta camada, temos a aplicação de Filtragem de *Traps* e o Reconhecimento Automático da Topologia da Rede.

5.1.2 Camada de Interface de Extensão de Aplicações

Nesta camada, encontram-se disponíveis as interfaces de programação que fornecem acesso às funções implementadas pelos Módulos de Extensão. No contexto da tecnologia de objetos, que será amplamente empregado neste projeto, esta camada é composta por todas as interfaces dos objetos codificados na camada MEM.

Figura 18 - Arquitetura do Sistema NetTracker

5.1.3 Camada de Módulos de Extensão

Todos os objetos responsáveis por proverem funções básicas de apoio às funções de gerenciamento serão integrados à esta camada. Tais funções desempenharão o papel de alicerce para os programas de aplicações que se encontram na camada superior. Basicamente, esta é composta pelos seguintes módulos:

- **HTTPServer** - Permite ao *NetTracker* disponibilizar serviços de configuração remota através de requisições HTTP. Este módulo possibilita que as interfaces de usuário possam ser transparentemente enviadas ao administrador, independentemente de sua localidade no ambiente.
- **Interface** - implementa as funções de comunicação entre os programas de aplicação e a interface do administrador, a qual lhe será enviada em forma de *Applets* Java, como parte componente de uma resposta HTTP disparada por um Browser com capacidades Java embutidas (i.é, Netscape 2.0, Explorer 3.0, Hotjava, etc.). Ver Figura 18 acima.

- **Database** - Este módulo é responsável por prover, de forma transparente, acesso à mais variada gama de SGBDs disponíveis no mercado. Para tal prevê-se a utilização do protocolo ODBC (Object Database Connection) ou ainda de *drivers* nativos oferecidos pelos fornecedores dos produtos. Inicialmente será utilizado uma implementação não comercial de uma banco de dados educacional, Mini-SQL software de domínio público.
- **SNMP** - todas os recursos de gerenciamento presentes no protocolo SNMP serão agrupados neste módulo. Em uma primeira instância, serão agregadas as funcionalidades de uma implementação de domínio público para este módulo.
- **SNMPv2** - com propósito semelhante ao anterior, este módulo disponibilizará as funções de gerenciamento da nova versão do SNMP.
- **Security** - este módulo tem o objetivo de disponibilizar todas as funções que se relacionam, e que tratem dos aspectos de segurança da ferramenta *NefTracker* (i. é, SSL, MD5, DES, RSA, S/Key, entre outros).

5.1.4 Biblioteca de objetos *Java*

Nesta camada encontram-se todos os objetos que compõem as classes da linguagem *Java*, e que são responsáveis por permitirem a implementação de grande parte das camadas superiores.

5.1.5 Máquina Virtual *Java*

Sem dúvida, é o coração da nova arquitetura do *NefTracker*. Sobre ela estão assentadas todas as principais características inovadoras da ferramenta: carga e descarga dinâmica de módulos, independência de plataforma, linkedição dinâmica e em tempo de execução de classes, acesso remoto a objetos, códigos binários transportáveis, etc. Está previsto, para o futuro, a incorporação de um *Just-in-Time Compiler* (JIT), que nada mais é do que um compilador em tempo de execução, mas que possibilitará incrementar em muito a velocidade de execução do sistema. *Don't touch*

5.2 Estrutura e Relacionamento dos objetos do sistema.

A ferramenta *NefTracker* é composta por duas categorias principais de objetos: Objetos de Sistema e Objetos de Aplicação. Os primeiros definem os objetos que integram o ambiente básico de execução do sistema, onde serão efetuado a carga dinâmica e posterior ativação dos Objetos de

Aplicação. Estes últimos, por sua vez, definem de fato as aplicações do sistema *NetTracker*. A Figura 19 nos dá uma visão mais detalhada desta estrutura organizacional.

Figura 19 - Estruturação dos Objetos NetTracker

Desta forma, o processo de ativação e execução do sistema *NetTracker* pode ser descrito pelos seguintes passos:

1. Após o processo de ativação da Máquina Virtual Java, a classe principal do sistema (*NetTracker*) é carregada, checada suas dependências e posteriormente ativada. Este procedimento é comumente aplicado às aplicações Java;
2. Após ser instanciado pela etapa anterior, e de posse do controle executivo, o método construtor configura o sistema criando as estruturas de suporte e controle necessárias. Neste processo são instanciados e ativados os objetos ditos essenciais, tais como *HttpServer*, *Manager*, *Loader*, entre outros. O primeiro funciona como um servidor HTTP e fica responsável por permitir o interfaceamento do sistema *NetTracker* com o administrador da rede, através de um *Browser* Web qualquer. Já o segundo é responsável por controlar a criação e o gerenciamento dos objetos *NetTracker* (*MVApplication*) que são instanciados a partir de classes carregadas dinamicamente pelo ambiente. O objeto

VLoader é tratado logo a seguir. Ao término desta etapa, todos os objetos de sistema estão estabelecidos e em processo de execução;

3. Ativado como uma *thread* Java, o objeto VLoader inicia o procedimento de carga e registro de todos os objetos de aplicação. Para tal é examinado o arquivo de cadastro de aplicações (MVAppReg.cfg), buscando e efetuando a carga dinâmica de cada um dos módulos de aplicações cadastradas. A princípio estão cadastradas as aplicações de suporte ao sistema (*ManagerSNMP*, *Logger*, *Trigger*, *Database* e *Trapper*).
4. Após a carga, e durante o processo de ativação dos módulos de aplicação mencionados anteriormente, cada qual efetua o registro de seus atributos em um objeto de registro de atributos (NTApplicationAttributes). Isto é necessário, pois, através deste registro é possível o controle e monitoração de cada atributo pelo gerenciador do sistema (NTManager).

Ao término deste processo de ativação o sistema está plenamente operacional e apto a interagir com o administrador da rede.

5.3 Mapeamento do SNMPv3 na estrutura do NetTracker

O desenvolvimento de novas estruturas tanto para o *NetTracker* como no *SNMPv3* possuem os mesmos objetivos, torná-los mais adaptativos, e apresentam muitas semelhanças nos aspectos organizacionais, tornando a adaptação do protocolo *SNMPv3* ao *NetTracker* um processo quase que automático.

Seguindo as especificações da estrutura do *SNMPv3* tem-se que o *SNMP Engine*, devido às suas características de prover os recursos básicos do protocolo, será adicionado à camada de módulos de extensão (NEM) do *NetTracker*, onde estarão implementados os serviços de envio e recebimento de mensagens, autenticação e criptografia das mensagens, e o controle de acesso aos objetos gerenciáveis, e as *Aplicações do SNMPv3* serão adicionadas à camada de módulos de programas de aplicação (NAPM), como o Filtro de *Traps* e o *ManagerSNMP*.

5.3.1 ManagerSNMP

O desenvolvimento dos módulos que compõem a nova estrutura do *NetTracker*, serão implementados com o *Java*, linguagem de programação orientada ao objeto de propósito geral, que prevê extensões para o desenvolvimento de aplicações distribuídas (cliente-servidor), tendo como

característica ser transportada e executada pelos navegadores, disponibilizando uma interface padrão e multiplataforma.

A aplicação *ManagerSNMP* é responsável por gerenciar as informações SNMP contidas em MIBs (*Management Information Base*) controladas e mantidas por agentes SNMP. O processo de obtenção de informações é realizado de duas formas: *polling*, a aplicação gerente faz requisições periódicas aos clientes; e o *trapping*, que são mensagens enviadas ao gerente sobre algum evento anormal ou pré-determinado (Arquivo de configuração *ManagerSNMP.conf*), sem que estas tenham sido requisitadas pelo gerente.

Seguindo a estrutura das aplicações do *NetTracker* tem-se que o objeto *ManagerSNMP* é composto das seguintes classes: *ManagerSNMP*, *MibControl* e *MibTrapper* que se interrelacionam com o *NetTracker Extension Module* (Fig. 20), onde encontra-se as funções básicas para a implementação das classes.

Figura 20: Estrutura do SNMPManager sobre a estrutura do NetTracker

- *ManagerSNMP*: Implementa as funções que monitoram os limites aceitos para as variáveis do ambiente gerenciado, para que no momento que estas variáveis alcançarem os seus limites o gerente SNMP dispare um evento que irá tratar o problema.
- *MibControl*: As funções de monitoramento e controle dos agentes SNMP serão implementadas nesta classe, no qual a interface as aplicações será realizada através de páginas HTML.
- *MibTrapper*: Responsável por receber as notificações (Traps) enviadas pelos agentes SNMP da rede.

5.4 Especificação das funções básicas para o desenvolvimento do *SNMPv2*

Para realizar o gerenciamento sobre um elemento de rede o sistema gerente deverá possuir uma descrição dos objetos gerenciáveis (*Management Information Base- MIB*). A MIB é um arquivo de configuração que descreve os objetos gerenciáveis através uma linguagem formal, *Abstract Syntax Notation One (ASN.1)*, definida pela ISO. Neste arquivo estão contidas as seguintes informações dos objetos: os OIDs (Object ID) e suas características (permissões de acesso , tipo, etc.).

Somente com a descrição dos objetos gerenciáveis o gerente poderá requisitar (*Get*) ou alterar (*Set*) informações em um agente *SNMP*, e identificar as informações contidas em uma mensagem (*Response*, *Trap* e *Inform*) recebida de um agente ou de outro gerente (*SNMPv2*).

As mensagens trocadas entre os agentes e gerentes através do protocolo *SNMP* são codificadas de forma a garantir a interoperabilidade do sistema, esta codificação é realizada utilizando-se do *Basic Encoding Rules (BER)*.

5.4.1 Representação dos Dados

Em um sistema de informação distribuído existem problemas para a troca de informações entre as aplicações localizadas remotamente, principalmente relacionados às informações de tipos de dados estruturados. As trocas destas informações através da rede é possível através da definição de uma sintaxe de transferência e da codificação dos dados.

Na sintaxe de transferência, que é de conhecimento tanto do emissor como do receptor, a entidade emissora irá estruturar as informações de forma que a entidade receptora dos dados tenha condições de isolar as unidades de informação e reestruturá-las de uma forma compreensível a ele.

Através da codificação dos dados, não os estruturando de forma alguma, os dados são transferidos na sua forma original apenas codificado em uma única *string*, que será utilizada pela entidade receptora para a decodificação na estrutura original.

A ISO definiu uma notação que permite que tipos de dados simples e complexos possam ser estruturados, bem como os valores que tais tipos possam assumir. Esta notação é denominada ASN.1 (*Abstract Syntax Notation One*) [IS88a]. A codificação dos dados em uma *string*, é realizado por algoritmos denominados BER (*Basic Encoding Rules*) [IS88b], que determinam os valores dos octetos utilizados.

5.4.1.1 Abstract Syntax Notation One (ASN.1)

Em um sistema distribuído os valores que são transmitidos podem ser simples e complexos (formados de vários tipos simples combinados). Cada tipo recebe uma denominação que o distingue de todos os demais tipos.

O ASN.1 é uma notação que permite definir tipos de dados simples e complexos, e especificar valores que estes tipos podem assumir. Os dados simples são representados por: *BOOLEAN*, *INTEGER*, *BITSTRING*, *OCTETSTRING* e *NULL* e as estruturas complexas são definidas agregando os tipos simples, utilizando-se de algumas formas estruturadas definidas:

- *SEQUENCE*: lista ordenada de tipos;
- *SEQUENCE OF*: interação ilimitada de um único tipo;
- *SET*: lista não ordenada de tipos estruturados;
- *SET OF*: interação ilimitada de um único tipo estruturado;
- *CHOICE*: um campo que consiste de um valor dentre os tipos listados.

Com o uso destas estruturas e dos tipos primitivos é possível definir uma estrutura complexa como por exemplo o formato dos PDUs (*Packet Data Unit*) SNMPv2, que definem as estruturas e os tipos das informações dos pacotes SNMPv2.

```
SNMPv2-PDU DEFINITIONS ::= BEGIN
    PDUs ::= CHOICE { get-request      GetRequest-PDU,
```

```

get-next-request  GetNextRequest-PDU,
get-bulk-request  GetBulkRequest-PDU,
response          Response-PDU,
set-request       SetRequest-PDU,
inform-request    InformRequest-PDU,
snmpV2-trap       SNMPv2-Trap-PDU }

-- PDUs
GetRequest-PDU    ::= [0] IMPLICIT PDU
GetNextRequest-PDU ::= [1] IMPLICIT PDU
GetBulkRequest-PDU ::= [5] IMPLICIT BulkPDU
Response-PDU      ::= [2] IMPLICIT PDU
SetRequest-PDU    ::= [3] IMPLICIT PDU
InformRequest-PDU ::= [6] IMPLICIT PDU
SNMPv2-Trap-PDU  ::= [7] IMPLICIT PDU

max-bindings INTEGER ::= 2147483647

PDU ::= SEQUENCE { request-id   Integer32,
                   error-status INTEGER {
                                           --- algumas vezes ignorado
                                           noError (0),
                                           tooBig (1),
                                           noSuchName (2),
                                           badValue (3),
                                           readOnly (4),
                                           genError (5),
                                           noAccess (6),
                                           wrongType (7),
                                           wrongLength (8),
                                           wrongEncoding (9),
                                           wrongValue (10),
                                           noCreation (11),
                                           inconsistentValue (12),
                                           resourceUnavailable (13),
                                           commitFailed (14),
                                           undoFailed (15),
                                           authorizationError (16),
                                           noWritable (17),
                                           inconsistentName (18) },
                   error-index  INTEGER (0..max-bindings), -- algumas vezes ignorado

```

```

variable-binding VarBindList }

BulkPDU ::= SEQUENCE {
    Request-id      Integer32,
    non-repeaters   INTEGER (0..max-bindings),
    max-repetitions INTEGER (0..max-bindings),
    variable-binding VarBindList }

VarBind ::= SEQUENCE {
    name ObjectName,
    CHOICE {
        value      Object Syntax,
        unspecified NULL,
        noSuchObject [0] IMPLICIT NULL,
        noSuchInstance [1] IMPLICIT NULL,
        endOfMibView [2] IMPLICIT NULL } }

VarBindList ::= SEQUENCE (SIZE (0..max-bindings)) OF VarBind

END

```

5.4.1.2 Basic Encoding Rules (BER)

O *Basic Encoding Rules* provê um algoritmo que especifica como um valor qualquer estruturado, definido usando ASN.1, deve ser codificado (seqüência de octetos) e decodificado para transmissão através da rede.

A codificação de um tipo de dados ASN.1 permite que um receptor, sem conhecimento da definição dos tipos, reconheça o início e o fim das construções (*SEQUENCE*, *SET*, etc.) e os octetos que representam os tipos básicos de dados (*BOOLEAN*, *INTEGER*, etc.).

Quando o rótulo ASN.1 (“[]”) é empregado, tanto o valor do rótulo como o valor identificado na construção do tipo de dado básico, que foi rotulado, é codificado na mensagem, a menos que seja utilizado o chamado rótulo implícito (*IMPLICIT*), que indica não obrigatoriedade da sua codificação, pois esta pode ser percebida pelo contexto e assim minimizar a quantidade de octetos transferidos.

Basicamente cada valor de dados será codificado em uma representação de três campos:

- *Type*: octeto que representa , que possui a seguinte estrutura:
 - bits 1 à 5: Tipo de informação está sendo transmitido (Tab. 1);
 - bit 6: Campo *tag* que identifica se é uma construção primitiva ou construída;
 - bit 7 e 8: Identifica a classe ASN.1 (Tab. 2).
- *Length*: seqüências de octetos que identifica o comprimento, quando não for definido o final da seqüência será identificado através de dois *octetos* nulos no final da *string*;
- *Value*: seqüência de octetos que descrevem o valor, que pode ser uma série de valores TLV

Universal	0	0
Application-wide	0	1
Context-specific	1	0
Private	1	1

Tab. 1 - Classes ASN.1

As regras de codificação sempre conduzem o rótulo do tipo, explicitamente ou implicitamente, bem como alguma representação do valor do tipo. Estas regras constituem outro padrão que aplicadas ao valor de um certo tipo definido pela ASN.1 resultam na especificação completa dos valores daquele tipo durante a transmissão.

Universal Tag	Tipo ASN.1
1	BOOLEAN
2	INTEGER
3	BIT STRING
4	OCTET STRING
5	NULL

6	OBJECT IDENTIFIER
7	ObjectDescriptor
8	EXTERNAL
9	REAL
10	ENUMERATED
12-15	Reservado
16	SEQUENCE, SEQUENCE OF
17	SET, SET OF
18	NumericString
19	PrintableString
20	TeletexString
21	VideotexString
22	IA5String
23	UTCTime
24	GeneralizedType
25	GraphicsString
26	VisibleString
27	GeneralString
28	CharacterString
29-...	Reserved

Tab. 2 - Tipos ASN.1

Através destas regras de codificação pode-se por exemplo codificar o valor inteiro 100, na seguinte seqüências de octetos:

Classe	Tag	Tipo ASN.1
00	0	2

Comprimento
1

Valor binário de 100							
0	1	1	0	0	1	0	0

5.4.2 Objetos básicos do protocolo SNMPv2

Portanto para o desenvolvimento de uma *API SNMP Manager (SNMP Engine)* será utilizado o uma biblioteca básica para o *SNMP*, que disponibiliza uma biblioteca com as funções mínimas necessárias, compostos por vários objetos agrupados em três principais categorias:

1. Classes *Snmpv2* de variáveis e tipos básicos: São as classes que defines os tipos e estruturas básicas de todo o pacote, assim como as constantes. Aqui também encontra-se a classe abstrata que contém os métodos de codificação e decodificação das estruturas ANS.1. Abaixo tem-se alguns desses objetos e suas especificações.

OidSpecification	Public String PathTree; Public int Syntax; Public int Access; Public int Status; Public String Name;	
MibTree	Public String Name; Public int Index; Public MibTree Lista[];	public objects PrintTree();

2. Classe *Mib Snmpv2*: São classes que trabalham com o objeto *Mib*, que provêem métodos para realizar o *parser* sobre o arquivo de especificação, montar uma tabela das informações, identificando os objetos e seus atributos, e métodos que manipulam estas informações.

MibStructure	public HashTable Table; public MibTree MibTreeName;	public OidSpecification IsOid(String PathTree); public object MibStructure(String FileName);
--------------	--	---

3. Classe *Snmpv2 Communication*: São classes que controlam as sessões de comunicações entre os agentes e o gerente. Possui ainda os métodos (herdados) responsáveis por estruturar e identificar as informações das mensagens ASN.1. E para que as informações trafeguem através da rede há os método que codificam e decodificam as mensagens em BER.

SnmpPeer	public String IpNumber; public int Port; public int Retries;	
----------	--	--

	...	
<code>SnmpPdu</code>	<code>public int TypeCommand;</code> <code>public OidInfo VarBind[];</code>	<code>public void PrintVarBind();</code>
<code>SnmpSession</code>	<code>public SnmpPdu Snmp_PDU;</code> <code>public SnmpPeer Snmp_PEER;</code>	<code>public void SnmpSession(int Snmp_TYPE);</code> <code>public boolean OpenSession();</code> <code>public SnmpPdu SendPdu();</code> <code>public boolean CloseSession();</code>

Pode-se exemplificar o uso do pacote em uma simulação de implementação da aplicação que irá executar a função *SnmpGET*.

```
// snmpget.class
import java.util.*;
import java.net*;
import Snmpv2.*;

public class snmpget
{
    public static void main(String args[])
    {
        // sintaxe do comando: snmpget <mib_filename> <OID> <host_ip>
        // nesta implementação não preocupo-se na verificação dos parâmetros
        MibStructure m = new MibStructure(args[0]);
        OidSpecification Oid_S = new OidSpecification();

        if( (Oid_S = m.IsOid(args[1])) == null)
        {
            System.err.println("Erro no OID!!!");
            return;
        }

        if( Oid_S.Access != ACCESS_READ)
        {
            System.err.println("Erro de permissão de leitura!!!");
            return;
        }

        SnmpPeer RemoteHost = new SnmpPeer();
        RemoteHost.IpNumber = args[2];
        SnmpSession Snmp_GET = new SnmpSession(GET_METHOD);
```

```
Snmp_GET.Snmp_PEER = RemoteHost;
```

```
Try
```

```
{
```

```
    if (Snmp_GET.OpenSession() != null)
```

```
        Snmp_GET.Snmp_PDU.PrintVarBind();
```

```
    Else
```

```
    {
```

```
        System.err.println("Erro na conexão");
```

```
        return;
```

```
    }
```

```
} catch (IOException e)
```

```
{
```

```
    System.out.println("Errol!!!");
```

```
    Return;
```

```
}
```

```
}
```

Considerações Finais

6.1 Conclusões

A disseminação do uso de tecnologias de rede, principalmente relacionadas à *Internet* (*WWW*, *HTTP* e *Java*), vem impulsionando a proliferação de redes de computadores cada vez maiores e mais complexas (heterogêneas), sendo necessário o desenvolvimento de ferramentas que auxiliem a tarefa de gerenciamento.

As ferramentas baseadas no modelo *Internet* possuem como protocolo de transportes de mensagens o *SNMP*, que na sua primeira versão atendia somente aos requisitos de monitoração, pois a falta de recursos de segurança no protocolo não estimulava os fabricantes de equipamentos de rede habilitar as funcionalidade de controle remoto.

A especificação da versão 2 do protocolo *SNMP*, aguardado com grande expectativa, indicava ser a solução para os problemas encontrados na versão anterior, principalmente aqueles relacionados à segurança. Contudo, problemas na nova estrutura de gerenciamento e a dificuldade de migração de uma versão a outra, levaram a *IETF* a reeditar o documento da primeira versão que descreve a arquitetura de gerenciamento. Com isso os problemas de segurança do protocolo que haviam sido resolvidos retornaram, tornando a sua utilização em funções de gerenciamento de controle não indicada.

Uma especificação para o protocolo *SNMPv2*, seguindo as documentações originais, foi desenvolvida para ser incorporada ao *NetTracker*, mas verificou-se os mesmos problemas de incompatibilidade entre versões, e a nova estrutura tornava o seu desenvolvimento muito complexo, fugindo da principal característica que o tornou um padrão, a simplicidade de implementação.

As indefinições sobre o *SNMPv2* impediram que a nova especificação tornasse um padrão *de fato*. Além disso algumas propostas alternativas foram sendo desenvolvidas, que buscavam sanar

principalmente os problemas de falta de segurança na comunicação. Neste contexto o protocolo *SNMPv2* dificilmente tornar-se-ia um padrão para o mercado.

No decorrer do desenvolvimento deste projeto, surgiu o protocolo *SNMPv3*, que resolve os problemas de estrutura da versão anterior, possuindo uma estrutura modular, permite que evoluções do protocolo possam ser incorporadas mais facilmente. Este modelo de estrutura modular pode ser facilmente empregado no novo sistema gerenciador de rede, *NetTracker*, em desenvolvimento no Instituto de Ciência Matemáticas de São Carlos, e o ponto final deste projeto destina-se à tarefa de mostrar como aquele mapeamento pode ser feito.

Com a nova estrutura pode-se inicialmente carregar o módulo *SNMPv1*, e assim que for desenvolvido o protocolo *SNMPv2* e *SNMPv3*, suas funcionalidades serão incorporadas ao sistema, capacitando-o a trabalhar com todas as versões do protocolo.

Este modelo de estrutura consegue com isso garantir uma maior longevidade ao sistema, incorporando as novas características, agregando novas funções às anteriores sem prejuízo de incompatibilidade, adaptando-se assim ao ambiente.

6.2 Trabalhos Futuros

Devido a falta de definição sobre um padrão para o protocolo *SNMPv2*, o projeto serviu de base para o estudo do protocolo *SNMP* em sua totalidade, no qual são apresentados detalhes de funcionamento de cada primitiva do protocolo, processos de trocas de mensagens de gerenciamento e um modelo de estrutura de uma biblioteca básica para o desenvolvimento de *APIs SNMP*.

Considerando o desenvolvimento da nova versão do protocolo *SNMP*, surge como proposta de trabalho futuro a implementação do módulo gerente do *NetTracker*. Utilizando a modularidade das estruturas do *NetTracker* e do *SNMPv3*, partes do sistema gerente poderão ser desenvolvidas separadamente e ao final serem agregadas ao sistema, tendo como principal objetivo o desenvolvimento do módulo das funções básicas do protocolo.

O uso da linguagem *Java* para o desenvolvimento do projeto possibilita a incorporação do *CORBA*, que teve grandes avanços nesse sentido no último ano (*RMI* e *IDL*). O desenvolvimento do módulo de interface do sistema de gerenciamento com esta tecnologia possibilitará a apresentação das informações sem atrasos, sendo atualizado assim que o módulo gerente atualizar as informações, recebidas ou requisitadas, na base de dados.

ASN.1 - Abstract Syntax Notation One: Linguagem OSI para descrever sintaxe abstrata.

ACSE - Association Control Service Element: Elemento de serviço de aplicação responsável pelo estabelecimento e liberação de conexões.

BER - Basic Encoding Rules: Linguagem OSI para a descrição da sintaxe de transferência.

CMIP - Common Management Information Protocol: Protocolo OSI para o gerenciamento de redes.

CMIS - Common Management Information Service: Serviço oferecido pelo CMIP.

CMISE - Common Management Information Service Element: Elemento de serviço de aplicação, responsável pela troca de informações de gerenciamento de rede.

CMOT - CMIP over TCP/IP: Mapeamento do modelo de gerenciamento de redes OSI sobre o conjunto de protocolos TCP/IP.

HEMS - High-level Entity Management System: Primeiro experimento para o gerenciamento de redes.

IAB - Internet Activities Board: Organização técnica que controla o desenvolvimento do conjunto de protocolos Internet.

ISO - International Organization for Standardization: Organização que produz muitos dos padrões mundiais.

ISODE - ISO Development Environment: Ferramenta de pesquisa desenvolvida para o estudo das camadas superiores do OSI.

LME - Layer Management Entity: Entidade de uma camada que interage com a SMAE.

LPP - Lightweight Presentation Protocol: Protocolo que implementa o serviço de apresentação OSI mínimo, utilizando um protocolo de propósito especial.

LAN - Local Area Network: Rede de computadores de alta velocidade, transferência de baixa latência e limitada geograficamente.

MIB - Management Information Base: Coleções de objetos, variáveis, que podem ser acessados via um protocolo de gerenciamento de rede.

OSI - Open Systems Interconnect: Esforço internacional para facilitar a comunicação entre computadores de diferentes fabricantes.

PDU - Protocol Data Unit: Objeto de dado trocado entre as camadas contendo dados do usuário e informações de controle do protocolo.

ROSE - Remote Operation Service Element: Elemento do serviço de aplicação responsável pelas interações de *request/replay*.

SGMP - Simple Gateway Monitoring Protocol: Predecessor do SNMP.

SMAE - System Management Application-Entity: Processo responsável pela coordenação entre as LMEs e o protocolo de gerenciamento.

SMI - Structure of Management Information: Regras usadas para definir os objetos que podem ser acessados via um protocolo de gerenciamento de rede.

SNMP - Simple Network Management Protocol: Protocolo de aplicação do conjunto de protocolos Internet, que oferece serviço de gerenciamento de rede.

SNMPv2: Segunda versão do protocolo SNMP.

TCP/IP: Conjunto de Protocolos Internet.

UDP - User Datagram Protocol: Protocolo do conjunto de protocolos Internet que oferece o serviço de transporte não orientado a conexão.

WAN - Wide Area Network: Rede de computadores com conexões distantes.

Referências Bibliográficas

- [BE90] BEN_ARTZ, A.; CHADNA, A.; WARRIER, U., *Network Management of TCP/IP Networks: Present and Future*. IEEE Network, v.4., n.4., p.35-43, 1990.
- [BE95] Bem-Natan, R., *A guide to Common Object Request Broker Architecture*, McGraw Hill, 1995.
- [BO91] Booch, G., *Object Oriented Design with Applications*, The McGraw Hill, 1995
- [CA90] CASE, J. et. al., *A Simple Network Management Protocol (SNMP)*, Network Working Group, Request for Comments 1157, 1990.
- [CA93a] CASE, J. et.al., *Introduction to version 2 of the Internet-standard Network Management Framework*, SNMP Research Inc., Hughes LAN System, Dover Beach Consulting Inc., Request for Comments 1441, 1993.
- [CA93b] CASE, J. et.al., *Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)*, SNMP Research Inc., Hughes LAN System, Dover Beach Consulting Inc., Request for Comments 1442, 1993.
- [CA93c] CASE, J. et.al., *Textual Conventions for version 2 of the Simple Network Management Protocol (SNMPv2)*, SNMP Research Inc., Hughes LAN System, Dover Beach Consulting Inc., Request for Comments 1443, 1993.
- [CA93d] CASE, J. et.al., *Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)*, SNMP Research Inc., Hughes LAN System, Dover Beach Consulting Inc., Request for Comments 1448, 1993.
- [CA93e] CASE, J. et.al., *Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)*, SNMP Research Inc., Hughes LAN System, Dover Beach Consulting Inc., Request for Comments 1450, 1993.
- [CA93f] CASE, J. et.al., *Manager-to-Manager Management Information Base*, SNMP Research Inc., Hughes LAN System, Dover Beach Consulting Inc., Request for Comments 1451, 1993.
- [CA96a] CASE, J. et.al., *Introduction to Community-based SNMPv2*, SNMP Research Inc., SNMPv2 Working Group, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, Request for Comments 1901, 1996.
- [CA96b] CASE, J. et.al., *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)*, SNMP Research Inc., SNMPv2 Working Group, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, Request for Comments 1902, 1996.
- [CA96c] CASE, J. et.al., *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)*, SNMP Research Inc., SNMPv2 Working Group, SNMP Research, Inc., Cisco

Systems, Inc., Dover Beach Consulting, Inc., International Network Services, Request for Comments 1903, 1996.

- [CA96d] CASE, J. et.al., *Conformance Statements for Version 2 of the Simple Network Management Protocol*, SNMP Research Inc., SNMPv2 Working Group, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, Request for Comments 1904, 1996.
- [CA96e] CASE, J. et.al., *Protocol Operations for Version 2 of the Simple Network Management Protocol*, SNMP Research Inc., SNMPv2 Working Group, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, Request for Comments 1905, 1996.
- [CA96f] CASE, J. et.al., *Transport Mappings for Version 2 of the Simple Network Management Protocol*, SNMP Research Inc., SNMPv2 Working Group, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, Request for Comments 1906, 1996.
- [CA96g] CASE, J. et.al., *Management Information Base for Version 2 of the Simple Network Management Protocol*, SNMP Research Inc., SNMPv2 Working Group, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, Request for Comments 1907, 1996.
- [CA96h] CASE, J. et.al., *Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework*, SNMP Research Inc., SNMPv2 Working Group, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, Request for Comments 1908, 1996.
- [CA96i] CASE, J. et.al., *User-based Security Model for SNMPv2*, SNMP Research Inc., SNMPv2 Working Group, SNMP Research, Inc., Cisco Systems, Inc., Dover Beach Consulting, Inc., International Network Services, Request for Comments 1910, 1996.
- [CI94] CICILINI, R., *Desenvolvimento de uma Agente SNMP para Plataformas rodando DOS*, Dissertação para obtenção do título de Mestre na área de Ciências de Computação e Matemática Aplicada, 1994.
- [CO91a] COMER, D. E., *Internetworking with TCP/IP - Principles, Protocols and Architecture*. 2.ed., Volume 1, Englewood Cliffs, Prentice Hall, 1991.
- [CO91b] COMER, D. E.; STEVENS, D. L., *Internetworking with TCP/IP - Design, Implementation and Internals*. Volume 2, Englewood Cliffs, Prentice Hall, 1991.
- [FI97] Fielding, R., et al, *Hypertext Transfer Protocol (HTTP/1.1)*, (DDS Network Information Center, SRI International), Request for Comment 2068, 1997
- [HA93] Hauser, C.; Jacobi, C., *Using Threads in Interactive Systems: A Case Study*, ACM SIGOPS, p.94-105, dezembro de 1993.
- [HA95] HARNEDY, S., *TOTAL SNMP - Exploring The Simple Network Management Protocol*, CBM Books, 1995.

- [HU94] HUNT, C., *TCP/IP - Network Administration*, O'Reilly & Associates, Inc., 1994.
- [IS88a] Recommendation X.208 and ISO 8824, Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), as extended by Addendum 1 to ISO 8824, 1988.
- [IS88b] Recommendation X.209 and ISO 8825, *Information processing systems - Open systems interconnection - Specification of basic encoding rules for abstract Syntax Notation One (ASN.1)*, as extended by Addendum 1 to ISO 8825, 1988.
- [KL88] KLERER, S. M., *The Management Architecture: an Overview.*, IEEE Network, v.2, n.2, p.20-29, 1988.
- [LI95] LIEIRA, J., *Utilização de Video e Audio em Sistemas de Gerenciamento de Redes de Computadores*, Dissertação para obtenção do título de Mestre na área de Ciências de Computação e Matemática Aplicada, 1995.
- [LI97] Lindholm, T., et al., *The Java Virtual Machine Specification*, Addison-Wesley, 1997
- [MO97] MOURO, R. B., *Uma Arquitetura Operacional Extensível por Ferramentas de Gerenciamento de Redes*, Dissertação para obtenção do título de Mestre na área de Ciências de Computação e Matemática Aplicada, 1997.
- [NA96] Naughton, P., *The Java Handbook*, Osborne McGraw-Hill, Berkeley, 1996.
- [OD94] ODA, C. S., *Desenvolvimento de uma Sistema Monitor Gráfico para Gerência de Redes Locais*, Dissertação para obtenção do título de Mestre na área de Ciências de Computação e Matemática Aplicada, 1994.
- [PR96] PROSISE, J., Choosing a route - How multithreading works in 32-bit systems, PC Magazine, Março de 1996.
- [RO91] ROSE, M. T., *The Simple Book - An Introduction to Management of TCP/IP-based Internets*, Prentice Hall, 1991.
- [SC95] SCHALLER, H. N., A concept for hierarchical, decentralized management of the physical configuration in the Internet, KiVS95, p.20-24, Chemnitz, Feb. 1995.
- [SI97] The Simple Times, The Quarterly Newsletter of SNMP Technology, Comment and Events, Volume 5 Numer 1, Dez. 1997.
- [SL93] SLOMAN, M.; MAGEE, J.; TWIDLE, K.; KRAMER, J., *An Architecture for Managing Distributed Systems*, Proceedings of Fourth IEEE Workshop on Future Trends of Distributed Computing Systems, p.40-4, Lisbon-Portugal, Sep. 22-24, 1993.
- [ST90] STEVENS, W.R., *UNIX Network Programming*, Prentice Hall, Software Series, 1990.
- [ST93] STALLINGS, W., *SNMP, SNMPv2 and CMIP - The Practical Guide to Network-Management Standards*, Addison Wesley, 1993.
- [TA89] TANENBAUM, A. S., *Computer Network 2.ed.*, Englewood Cliffs, Prentice Hall, 1989.

[WA89] WARRIER, V.; BESAW, L., *The Common Management Information Services and Protocol over TCP/IP (CMOT)*. (DDN Network Information Center, SRI International), Request for Comments 1095, 1989.

Apêndice A

An Adaptative Systeem for Detecting Intrusion in Network

Artigo submetido e aceito no:

“III International Congress of Information Engineering”

Buenos Aires/Argentina – Abril/1997

An Adaptive System for Detecting Intrusion in Networks

Adriano M. Cansian (adriano@ibilce.unesp.br)^{1 2}

Edson S. Moreira, Rogério B. Mouro, Fábio T. Morishita, André C. P. de L. F. de Carvalho
(edson, mouro, teruo, andre@icmsc.sc.usp.br)³

Abstract

As the Internet expands both in number of hosts connected and in terms of the number of services provided, security has become a key issue for the technology developers. This work presents current development status of a system which, positioned at key points of the network, will keep looking at the passing packets, in search of suspicious connections. The system provides a list of such connections for the administrator, enabling him/her to take the proper action at an early stage of the intrusion. Neural Networks are used to look for profiles of intrusion within the analysed data streams. The assessment is done through comparison with well known profiles of intrusion. The system is highly adaptive, since new profiles can be added to the data base and the Neural Network re-trained to consider them.

1. Introduction

The recent increase in the use of networks has been followed by an increased concern about the security of information stored in the computers interconnected. Additionally, an ever-increasing number of essential activities are performed through networks (especially over the Internet), and hence a correct and reliable operation is vital. On the other hand, piracy acts, intrusion attempts, consummate invasions and break-in actions are becoming frequent and involve an increasingly high number of computers [1-2]. This scenario brings up the need for special security techniques in modern computer systems; ones which go beyond the traditional “locking up the doors” practice.

Various Intrusion Detection Systems (IDS) have been developed and some of them have been introduced experimentally. These systems are divided into host-based [3-7] and network-based [8-10]. Host-based systems use audit trails (which are usually daemons) as the main action to detect intrusions, while network-based systems build up their own set of information using the network traffic “sniffed” from the communication medium.

¹UNESP - Universidade Estadual Paulista
Inst. Biociências, Letras e Ciências Exatas
R. Cristóvão Colombo, 2265
15055-000 São José do Rio Preto - SP - Brasil
Phone: (+55 17) 224-4966 Fax: 224-8692

²Instituto de Física de São Carlos - USP
IFSC - Cx Postal 336
13560-970 São Carlos SP - Brasil

³Instituto de Ciências Matemáticas de São Carlos - USP
ICMSC - Cx Postal 668
13560-970 São Carlos SP - Brasil

This paper describes the project of a network-based security system that uses techniques of Artificial Intelligence and Neural Networks to find out intruders, to register and learn about their mode of operation and then to provide elements that help the system administrator take action against them. The detection system is formed by one or more security agents, which are placed at strategic points in the network and that provide information to the managing personnel or systems.

In the context of this paper some definitions need to be established:

- **Connection:** indicates a set of information packets of which source-destination addresses are the same. Even though the addresses aren't enough to determine a connection, they are sufficient for an initial analysis. For a more accurate appraisal, the numbers of ports and the identification of the connection may be used.
- **Domain:** indicates an IP address, a network or a range of numbers, which will be considered as belonging to the same group, in order to restrict security areas. It has not the same function, as those defined in the NIS or DNS context.
- **Host:** a machine to which an IP number was given.

2. Detecting Suspect Conduct

Attack attempts take place according to some access techniques. In most cases and environments, the intruder is physically away from the system under attack [2]. The early IDS models, designed for isolated computers, use basic algorithmics which include multinomial functions evaluation and covariant matrixes approximation in order to detect deviation from normal user behaviour, as well as expert systems to detect security policy violation. The latest models monitor a great number of network computers and transfer monitored information to be processed in a central equipment by employing distributed systems techniques. These models also have the traffic over network as part of their detection algorithmics. Most of the IDS have an auditor process (daemon) in each machine, responsible for capturing security violation actions within that machine. Network-based systems rather than using audit trails, analyse the traffic of packets within the network to detect intrusive behaviour [5][13].

One of the proposals for innovating our management system consists of introducing a security agent able to detect intrusive behaviour in established connections. This agent acts by capturing and deciphering packets which are transmitted through the network under monitoring. In order to make inference about the security condition of the connections, the agent employs an expert system and a neural network which will provide a number, which, based on previously recorded intrusive information, will give an idea about the severity of attack or the degree of suspicion of the activities in that connection.

The system is based on the fact that an intrusion can be detected from an analysis of predetermined models, which are anomalous compared with normal actions. The following examples illustrate instances of suspicious behaviour [5][13]:

- Someone trying to access a certain system would produce an irregular rate of password mismatch errors, be it in relation to a computer in particular or to the system as a whole.
- Someone who gains illegitimate access to the system and on to an unauthorised account and password, would develop a different session to that of a legitimate user in terms of logging

time, access placement or type of connection. Besides, intruders may differ considerably from the legitimate users in terms of access to directories or in terms of the running of programmes which seek information about the system status.

- A user trying to get into the security mechanism of an operational system should execute a different set of programmes to the regular user's, and trigger more protection violations in the attempt to access unauthorised files and programmes. Being successful he/she would have access to programmes or files which he/she is not usually eligible to access.
- Someone trying to access documents sensitive to security can do logins during unusual hours and direct data into discs and printers that are not normally used by him/her.
- An intruder able to monopolise a resource (for example, the network) should have an abnormally high rate of activity in relation to the resource, whereas the activity of all the others users is in contrast normally low.

Our proposal is that the use of Neural Networks can provide mechanisms for the recognition of successful attacks, as well as having the capacity to adapt in response to the changes in the intruder's techniques.

3. The Security Agent

The agent is placed in a safe machine, one that is logically invisible to the others. It is also in a place where physical access is restricted, so it can be accessed only under certain special conditions. This safe machine is placed at sensitive points of the network system. These sensitive points can be:

- Directly in the network under monitoring (*Figure 1*);
- Close to interconnection points in internal networks, and/or sharing the backbone to which routers or gateways are connected to provide access to the external world. (*Figure 2*);
- At the access points of WANs intersections. In this case we suggest the development of a monitoring system that works over serial lines in point-to-point connections, as in the telephone trap (*Figure 3*). Note that the importance of this configuration is due to the fact that routers may generally have ports linked to more than one network.

Figure 1 - Possible conditions of monitoring: subnetwork monitoring

Figure 2 - Possible conditions of monitoring: multiple network monitoring

In all of the above situations, the agent passively monitors the network and captures the circulating packets through the use of the network interface in a promiscuous mode.

Figure 3 - Monitoring an external serial connection

The agent is organised in a four-layer model (Figure 4). The layers manage the flow of packets and provide the stimulus vector to the neural network. The lowest level only captures a flow of data in the network and passes the ordered packets to the second layer.

The second layer is consisted of 2 modules: packets pre-selection module, and expert system module. The pre-selection module makes an initial packet filtering, that may represent interesting events, such as what kind of protocol will be monitored or which origins or destinies should be considered. The previously filtered packets then pass through an analysis made by the expert system.

The expert system uses the following information when making decisions:

- Which are the expected paths of connection: connection source and destination (which ones could be risky) and source and destination ports involved;
- Sensibility of the machines and reliability of the domains.

These data work as an element of analysis to monitor the risky events, that are stored in a standing row where they stay for a definite time. The data stored in this row are vectors that include origin and destiny of the connection, ports involved, a “security level” (SG) and a time stamp. Each connection is represented by a vector in the row.

The “security level” is a numeric value that grows according to the arrival of monitored events coming from a given source-destination pair or from the same domain. For example, the system detects a *finger* command moving from host A to host B, it then includes its vector in the row with a security level value previously established for the *finger* command (SG = 10, for instance). As a new event is detected, for example a *telnet* (SG = 15, for instance), the row is scanned and once found an entrance coming from the same source-destination, the security level of this vector will be added to the value of the telnet security level, being SG = 25 now. In this case, a new vector is not created in the row.

Figure 4 - Security System Organisation

Another case to be considered is one in which the detected event comes from a domain where another event had previously taken place (already registered in the row), but not necessarily generated from the same machine. In this case, a new vector is inserted in the row, with an equal

security level to which is added a weight indicating that a previous attempt coming from that domain had already been detected.

This row is periodically checked, and when the security level field of a vector reaches to a pre-established threshold, the expert system “locks in” that connection or, alternatively, all of the connections coming from that domain. There is still the possibility of making different kinds of searches and checkings in the row, considering not only the threshold for a source-destination pair, but also related informations about the domains or a sub-set of related addresses. In this manner, when the expert system identifies patterns of behaviour different from the acceptable ones, all the packets travelling in that connection are sent to the next level (Connection Module).

Figure 5 - Standing row representation (risk events)

The third module is based on the hierarchic model of Network Security Monitor (NSM) [8] and receives the packets and organises them in a cause-effect relationship, identifying an unidirectional data flow. This is done through the analysis of source and destination fields, ports, and packets sequence number, thus solving the fragmentation problem. This procedure can be done at IP or TCP level.

Once identified and ordered, the stream consists of data being transferred from a machine to another through a unified set of ports using a determined protocol. These packets are mapped into a “Flow Vector”, which is the data transcription of a particular connection. If necessary, flow vectors' pairs are formed in order to represent a bi-directional data stream, i.e., a host-to-host connection.

Figure 6 - Building the connection vector

The flow vectors' pairs are represented by a "Connection Vector", which contains all the stream of data that is travelling between the two hosts through the connection being monitored. The connection vectors are then sent to be processed by a "Semantic Analyser".

The semantic analyser acts upon connection vectors, searching for attack profiles that would appear the data. The profiles are stored in a data base and updated according to need. These profiles are attack signatures and contain information on how a suspicious session behaves. The data in the connection are analysed, also how many and which of them match to the profile. These information are sent to a last post-processing module, which unifies the semantic analyser (matched profiles) to the expert system information (sensibility of services and connections) and its data base, in order to form a stimulus vector for the Neural Network.

The stimulus vector will contain values and information about the connection:

- Service capacity level. For example, *telnet* offers more capacities than *FTP*;
- Service authentication level, i.e., *telnet* requires more authentication than *SMTP*;
- Security level of the source and destination machines, which may be a numeric value provided by a pattern analyser system, such as COPS, SATAN or Tigger Scripts, or even a default value or a value referring to the operational system confidence, as attributed by international organisations such as FIRST [19], CERT [20] or COAST [21];
- Quantity of transferred data, which may be an important indicative of suspicious actions if related to the kind of service used;
- Connection time. Activities, services and amount of data transferred in a specific time (for example work time) may be suspicious if occurring in another time (for example, at night or weekends);
- And, at last, the amount of suspicious strings and attack signatures located and matched by the semantic analyser in the data, such as: *su*, *root*, *passwd*, *telnet*, *debug*, *rlogin*, *rsh*, *chown*, *service disable*, among others.

Figure 7 - Stimulus vector representation

The neural system analyses the stimulus vector with its respective weights that represent the importance of the occurrence of events, and tries to attribute a suspicion degree, that represents the suspicion state of a particular connection. Before the neural system can identify potential attacks, it must be trained with a meaningful and large enough amount of stimulus vector, that represent the behaviour of the suspicious connections and the legitimate ones. Once trained, the network must use its characteristic of generalisation to identify correctly the users who show characteristics similar to those included in the intruding actions used to train it.

The management system (remote) receives, then, the numeric value that represents the state of security of suspicious connections and map them, classified by security level, helped by a colour

code. From this point many things can be done, such as the transmission of many levels of alerts or warnings to the administrators (depending on the risk level of the detected events), shots logging processes, activation of counter-measures to isolate the host or the domain that caused the attack (active adjust of wrappers [17], firewalls or filters [18]), among others.

The neural network may also be adjusted, in order to reflect intruders' behaviour patterns changes. The administrators may detect highly intrusive behaviours, but that the neural network is not identifying correctly or is not identifying their severity as it was expected. In this case, new patterns can be added to the network and be retrained in order to learn to recognise the new patterns adequately. This adaptive capacity is the most important innovation in this project and does not appear in other known IDS.

4. Management System

The management module is independent of the security module. Both modules communicate with each other by TCP/IP. The possibility of putting a management module at any point is an interesting characteristic. We are working to establish a safe method of agent-manager data transfer. The management module also checks other information and data in the network, coming from the security agent:

- Which is the usage of external connection;
- Which are the machines, networks or domains most accessed internally and externally;
- What are the main hours for connections;
- Which are the most used services and by who they are used.

5. Current Status and Conclusions

At present time the capture agent is working, and several simulated intrusions attempts are being monitored, in order to acquire attack signature data, to be used to train the neural network system. About 120 different types of attacks over a networked environment were tested, and they are being converted into a binary code, which will be sent to be processed by the neural network system.

Our Intrusion Detection System intends to provide a fast and versatile instrument to attack the problem of computers intrusion on networks. It is based on the fact that most intruders acts in a determined pattern. These patterns are stored in a data base and used dynamically when the intrusion is being followed, with the help of expert systems and neural networks. Although there are not yet enough data to show the accuracy of the method, we believe that there is a good chance for the system to be successful. It does not try to state definitely about the presence of intrusion, instead, it provides a intrusion level that will indicate the likelihood of intruder activities.

Another important characteristic of this method is the adaptability. The administrators knowledge is easily introduced into the system (and to the re-trained network) in a way that new important information in the process may be incorporated dynamically.

Last, the system makes it possible to create an automatic environment of learning, which would permit that there be a meaningful increase of the expertise in the area of detection and manipulation of intrusions.

6. Acknowledgements

We would like to thank FAPESP and CAPES for the support given to this project through scholarships and equipment.

7. References

- [1] R. Bace. *A New Look at Perpetrators of Computer Crime*. In Proc. 16th Department of Energy Computer Security Conference, 1994.
- [2] P. Neumann & D. Parker. *A Summary of Computer Misuse Techniques*. In Proc. 12th National Computer Security Conference, pages 396-407, 1989.
- [3] H.S.Javitz & A.Valdez. *The SRI IDES Statistical Anomaly Detector*. In Proc. 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, 1991.
- [4] J.R.Winkler & W.J.Page. *Intrusion and Anomaly Detection in Trusted Systems*. In Proc. Fifth Annual Computer Security Applications Conference, Tucson, AZ, Dec,1990. 115-124.
- [5] D.E. Denning. *An Intrusion-Detection Model*. IEEE Trans. on Software Engg. Vol. SE-13, pp. 222-232, Feb.1987.
- [6] T.F. Lunt et al. *IDES: A Progress Report*. In Proc. Sixth Annual Computer Security Applications Conference. Tuscon, AZ, Dec. 1990.
- [7] T.F. Lunt et al. *A Real Time Intrusion Detection Expert System (IDES)*. Interim Progress Report, Project 6784, SRI International, May 1990.
- [8] L.T. Heberlein et al. *A Network Security Monitor*. In Proc.1990 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, 1990, pp 296-304.
- [9] L.T. Heberlein et al.. *Towards Detecting Intrusions in a Networked Environment*. In Proc. 14th DOE Conference on Computer Security. Concord, CA, May 1991, pp 17-47.
- [10] L.T. Heberlein, K.N. Levitt and B. Mukherjee. *A Method to Detect Intrusive Activity in a Networked Environment*. In Proc. 14th National Computer Security Conference. Washington, DC. Oct. 1991, pp 362-371.
- [11] S.R. Snapp et al. *DIDS (Distributed Untrusion Detection System) - Motivation, Archieteture, and an Early Prototype*. In Proc. 14th National Computer Security Conference. Washington, DC. Oct. 1991.
- [12] S.R. Snapp et al. *A System for Distributed Intrusion Detection*. In Proc. IEEE COMPCON 91. San Francisco, CA. Feb.1991, pp 170-176.
- [13] L.T. Heberlein, B. Mukherjee and K.N. Levitt. *Internet Security Monitor: An Intrusion Detection System for Large-Scale Networks*. In Proc. 15th National Computer Security Conference. Baltimore, MD. Oct. 1992. and B. Mukherjee , L.T. Heberlein and K.N. Levitt. *Network Intrusion Detection*. IEEE Network may/june 1994., pg. 26-41.
- [14] C. Ko et al. *Analysis of an Algorithm for Distributed Recognition and Accountability*. In Proc. First ACM Conference on Computer and Communicatons Security. Faifax, VA, Nov.1993. pp 154-164.
- [15] S.G.Stanford-Chen. *Distributed Tracing of Intruders*. M.S.Thesis. Divison of Computer Science, University of California, Davis, CA. 1995.
- [16] S.E.Smaha. *Haystack: An Intrusion Detection System*. In Proc. IEEE Fourth Aerospace Computer Security Applications Conference, Orlando FL, Dec. 1988.
- [17]W.Venema. *TCP Wrapper: Networking Monitoring, Access Control and Booby Traps*. ftp://cert.sei.cmu.edu/pub/network_tools/tcp_wrapper.txt
- [18] S.Garfinkel & G.Spafford. *Pratical UNIX and Internet Security*. O'Reilly and Associates, 2nd Edition, 1996.
- [19] <http://www.first.org/first/>
- [20] <http://www.sei.cmu.edu/technology/cert.cc.html>

- [21] <http://www.cs.purdue.edu/coast/coast.html>
- [22] Rumelhart, D.E.; McClelland, J.L. and The PDP Research Group. *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, MIT Press 1986.
- [23] Carvalho, A.C.; Fairhurst, M.C. e Bisset, D.L. An integrated Boolean neural network for pattern classification, *Pattern Recognition Letters*, Vol. 15, pages 807-813, august 1994.
- [24] Lupo, C.J. Defense Applications of Neural Networks, *IEEE Communications Magazine*, Vol. 27, No. 11, pag. 82-88, november 1989.
- [25] Wong A. J. Recognition of General Patterns Using Neural Networks, *Biological Cybernetics*, Springer-Verlag, Vol. 58, pag. 361-372, 1998.

Apêndice B

NetTracker: Uma Arquitetura Operacional Extensível

Para Ferramentas de Gerenciamento de Redes

Artigo submetido e aceito no:

“XV Brazilian Symposium on Computer Networks, SBRC'97”

São Carlos/São Paulo – Maio/1997

NetTracker: Uma Arquitetura Operacional Extensível para Ferramentas de Gerenciamento de Redes.

Rogério Borges Mouro, Fábio Teruo Morishita e Edson S. Moreira.

(mouro, teruo, edson@icmsc.sc.usp.br)

Instituto de Ciências Matemáticas de São Carlos - ICMSC/USP

Po Box 668 - CEP 13560-970 - São Carlos - SP - Brasil

Resumo

O avanço tecnológico em muito tem dificultado a atualização das ferramentas de gerenciamento de rede, principalmente por não existirem propostas concretas para uma arquitetura operacional. Neste artigo é apresentado uma avançada arquitetura operacional, para sistemas de gerenciamento de redes, que possui como características a facilidade e flexibilidade de incorporação de novas tecnologias e adequação a diferentes plataformas. Ao final são apresentadas resultados preliminares obtidos na aplicação destes conceitos no sistema de gerenciamento NetTracker, em termos de capacidade de atualização, funcionalidade e portabilidade.

Abstract

Technological advances have contributed to make difficult upgrades of network management tools, mainly due to the fact that there are not concrete proposals for operational architecture in this field. In this article we present an advanced operational architecture for network management systems, that has as main features the easiness and flexibility for the incorporation of new technologies and the porting to different platforms. Finally we present some preliminary results obtained on the application of this approach on the development of the NetTracker, a network management system, in terms of upgrading capability, functionality and portability.

1. Introdução

Inovações e evoluções tecnológicas refletem as necessidades organizacionais e estruturais dos ambientes computacionais atuais. A velocidade com que tais transformações vêm ocorrendo surpreende mesmo as organizações que contribuem diretamente com esta escalada tecnológica, deixando-as constantemente sujeitas a mobilizarem esforços no sentido de manterem atualizados seus produtos.

Nos últimos anos, muito tem se discutido sobre gerenciamento de redes, e várias foram as contribuições dadas por pesquisadores em todo mundo para se tentar sanar deficiências na área. Arquiteturas de gerenciamento foram propostas, como a *OSI Management Architecture* (ANS87) e a *Internet-standard Network Management Architecture* (CAS90). Contudo, estas contribuições permaneceram restritas ao campo funcional, estabelecendo os requisitos desejáveis para eventuais ferramentas de auxílio às atividades de gerenciamento de redes. Muito pouco tem sido discutido ou proposto no sentido de se estabelecer arquiteturas operacionais tecnologicamente superiores, e que tragam como características centrais a facilidade e flexibilidade de atualização tecnológica, bem como a capacidade de adequação a diferentes plataformas.

Inicialmente, neste artigo, são estabelecidos os requisitos essenciais a uma arquitetura operacional avançada para ferramentas de gerenciamento de redes. Esta arquitetura, aliada aos requisitos funcionais definidos pela proposta de gerenciamento ISO/OSI (KLE88), dispõe de grande flexibilidade

em termos de capacidade de atualização e independência tecnológica. A seguir é apresentado o protótipo da ferramenta de gerenciamento de redes NetTracker, desenvolvida adotando-se os conceitos da nova arquitetura proposta. Ao final, são apresentados os resultados preliminares obtidos por esta ferramenta, em termos de capacidade de atualização, funcionalidade e portabilidade.

2. Definição de Requisitos

Dentro da necessidade evolutiva e mediante um criterioso levantamento em busca de eventuais inadequações tecnológicas nas ferramentas de gerenciamento existentes, chegou-se a conclusão de que os efeitos depredatórios da evolução, sobre uma dada ferramenta, têm como principal causa a ineficácia das arquiteturas operacionais existentes, inibindo e coibindo, de sobremaneira, a capacidade de absorção de novas tecnologias. Neste contexto e objetivando subsídios para a elaboração de uma nova e moderna arquitetura operacional, foram definidos os requisitos básicos e essenciais a serem seguidos.

Tais subsídios foram elaborados mediante uma extensiva prospecção tecnológica, tendo como critério fundamental a adesão às especificações de padrões abertos internacionais, como os propostos pelas organizações ISO (*International Standard Organization*), ITU (*International Telecommunication Union*), OMG (*Object Management Group*) IAB (*Internet Architecture Board*) e W3C (*World Wide Web Consortium*). Outros elementos contribuintes para a concepção dos requisitos vieram da avaliação de produtos em áreas de tecnologias afins, tais como Jigsaw© (W3C, MIT, INRIA), Netscape Server's Architecture (Netscape™, Inc), SystemView® (IBM™) e Jeeves© (Sun Microsystems™).

Centrado nas premissas: uma arquitetura tecnologicamente superior e capacidade de atualização fácil, foram elaborados os requisitos que se seguem.

- Uma ferramenta de gerenciamento deve ser facilmente portátil para diferentes plataformas de sistemas operacionais e arquiteturas de hardware, exigindo-se um mínimo de esforço neste sentido;
- Uma ferramenta de gerenciamento deve ser amplamente extensível em relação às suas capacidades, possibilitando a inserção de novas funcionalidades sem a necessidade de re-codificações das partes já existentes. Além do mais, deve apresentar como característica desejável a capacidade de extensão dinâmica, sem a necessidade de interrupções de seu funcionamento;
- Uma ferramenta de gerenciamento deve possuir capacidades modernas e eficientes de comunicação com o administrador da rede, provendo-o de total controle sobre os recursos disponíveis, independentemente da sua localização geodésica e do grau de segurança do meio de comunicação utilizado. Com tal intuito, protocolos confiáveis e seguros de comunicação deverão ser empregados;
- Uma ferramenta de gerenciamento deve utilizar-se das tecnologias e especificações abertas complacentes, devidamente padronizadas por entidades a contento, e de reconhecida superioridade funcional.

O direcionamento destes requisitos teve como resultado o estabelecimento de uma nova e poderosa arquitetura, baseada em tecnologias emergentes (Java, CORBA, HTTP, SSL, SNMPv2), e que serviu como base para a implementação do protótipo inicial da ferramenta de gerenciamento de redes NetTracker, apresentada ao longo deste documento. Antes, entretanto, são apresentadas as tecnologias utilizadas em sua implementação.

A cada dia, surgem novas e modernas tecnologias que possuem como característica comum a generalização de suas aplicações nas mais variadas áreas de interesse. No segmento de Tecnologia da Informação, este comportamento pode ser amplamente observado; diga-se de passagem a integração de áreas como Hipertextos, Protocolos de Comunicação, Criptografia de Dados e Multimídia, que culminou em um dos mais extraordinários sistemas de informação de nossa era: a Internet e, mais recentemente, sua malha de navegação *World Wide Web*. Seguem-se algumas das tecnologias que integram a arquitetura da ferramenta de gerenciamento de redes NetTracker.

Java - Desenvolvida pela Sun Microsystems, a linguagem de programação Java (GOS96) apresenta uma série de características únicas que tornaram-na um marco na história da informática moderna. Tais características incluem e não limitam-se às que se seguem:

- Java é um linguagem fortemente baseada em tipos e que integra todos os conceitos de orientação a objetos (BOO91);
- Resolução de dependências em tempo de execução. Esta característica é de extrema importância pois permite não apenas a carga dinâmica de classes como também a utilização de referências cruzadas nas definições das mesmas (i.é. duas classes distintas podem possuir referências entre si).
- Independência de plataforma. Característica importante da linguagem Java, foi a principal responsável pela rápida e massiva aceitação da mesma. O código gerado pelo compilador (conhecido por bytecode) não é direcionado a nenhuma arquitetura específica de processador, mas sim à especificação de uma máquina virtual conhecida como *Java Virtual Machine* (LIN97).
- Código Transportável. Característica advinda das demais, permite o envio e recebimento de código, possibilitando sua posterior execução após uma completa checagem de segurança. Tais códigos são conhecidos por *Java Applets* e são comumente executados na máquinas virtuais atreladas aos ambientes de navegação (*Browsers*).

CORBA - O aumento e a rápida proliferação do número de hardwares e softwares disponíveis impulsionaram o desenvolvimento em busca de uma arquitetura que permitisse a interoperabilidade entre dispositivos de rede. Como resposta a esta necessidade, o OMG (*Object Management Group*) propôs a especificação CORBA (*Common Object Request Broker Architecture*) (BEN95), que de uma forma simples, permite a comunicação entre objetos de aplicações, independente de suas localidades. Junto à CORBA, o OMG também definiu a IDL (*Interface Definition Language*) e as APIs (*Application Programming Interfaces*), utilizados para o desenvolvimento de aplicações distribuídas.

O relacionamento entre os objetos distribuídos é realizado através de uma arquitetura cliente-servidor, onde há a presença do ORB (*Object Request Broker*) como agente intermediário. Através do ORB um cliente pode executar um método de um objeto local ou remoto qualquer. De fato, o ORB intercepta a chamada a um método, localiza o objeto que o implementa, executa o método e por fim retorna os resultados. Tudo isso acontece de forma transparente, sem a necessidade do cliente saber onde o objeto destino encontra-se, a linguagem de programação, o sistema operacional ou qualquer outro aspecto relacionado à interface com o objeto. Neste cenário, a IDL tem como proposta simplificar o desenvolvimento destas aplicações, pois permite descrever detalhadamente as interfaces dos métodos de cada objeto que compõe uma aplicação distribuída.

HTTP - O *HyperText Transfer Protocol* (DAV97) é um protocolo aberto, responsável por permitir o transporte de informações de tipos variados sobre uma rede de arquitetura TCP/IP (TAN89). Este protocolo é comumente empregado no tráfego de códigos hypertextos HTML (FIE97), executáveis bytecode Java, multimídia, textos planares ASCII, entre outros.

SSL - O padrão de segurança de informação SSL (*Secure Sockets Layer*), desenvolvido inicialmente pela Netscape™ (FRE96), especifica uma camada de segurança entre a aplicação e o Socket (STE90) de comunicação, protegendo o conteúdo da informação mediante sua criptografia. Tal especificação encontra-se atualmente em fase de padronização internacional, devendo emergir como uma alternativa definitiva para o transporte de informações de caráter restritivo. O protocolo em si é baseado no algoritmo de criptografia de dados RSA (RIV78) e provê suporte à certificação de autenticidade, tanto do servidor quanto do cliente envolvidos na troca das informações.

SNMPv2 - De acordo com o escopo do gerenciamento Internet, um ambiente gerenciado é constituído de entidades agentes que provêem acesso às informações de gerenciamento e no mínimo de uma entidade gerente atuando no papel de centralizador e coordenador de ações. A troca das informações de gerenciamento é efetuada mediante a utilização de um protocolo específico, através do qual se definem as formas de autenticação, autorização e controle de acesso.

O SNMP (*Simple Network Management Protocol*) surgiu como uma proposta rápida às necessidades do mercado por um protocolo de gerenciamento em ambientes TCP/IP (ROS91). Em princípio o SNMP deveria ser apenas um protocolo provisório, sendo substituído pelo CMIP (*Common Management Information Protocol*). Devido ao atraso no desenvolvimento do CMIP, entre outros fatores, o SNMP tornou-se o principal padrão de gerenciamento. Entretanto, devido a seu caráter provisório, apresentava deficiências relacionadas a segurança e baixo desempenho em ambientes grandes.

Com o objetivo de solucionar tais problemas foi proposto uma nova versão do protocolo SNMP (STA93)(CAS93a). O SNMPv2 corrige problemas de segurança, além de adicionar novas funcionalidades, onde destaca-se a capacidade de gerenciamento distribuído via primitivas de comunicação gerente-gerente (CAS93b) e as formas de tratamento e transporte dos dados.

3. Arquitetura do Sistema

A pesar da estrutura de gerenciamento Internet (ROS90) mostrar-se desatualizada frente às novas proposta tecnológicas (DER96), ainda sim apresenta-se como o principal padrão existente, com funcionalidade comprovada. Sendo assim é evidente a necessidade de sua adoção como elemento de adequação a realidade atual.

Figura 3.1 - Estrutura de Comunicação intramódulos

A Figura 3.1 sintetiza o relacionamento entre os elementos da estrutura de gerenciamento que integram o ambiente gerenciado pela ferramenta NetTracker, e que seguem de perto a estrutura do gerenciamento Internet. Tais elementos são descritos a seguir:

- **Recurso Gerenciado** - representa todos os elementos que compõem a rede e que sejam providos de capacidades de gerenciamento Internet (i.e., *Routers, Bridges, Hosts, etc*), comunicando-se diretamente com a estação de gerenciamento, via protocolo SNMP e SNMPv2.
- **Estação Gerente** - parte fundamental de uma ferramenta integrada de gerenciamento este elemento atua como agente centralizador das informações de gerenciamento, ficando responsável por prover todas as funções de controle e monitoração ao administrador.
- **Interface do Administrador** - é o elemento responsável por permitir a interação entre a ferramenta NetTracker e o administrador da rede. É importante notar que esta cumpre não apenas o papel de

agente difusor de informação, mas também o papel de elemento descentralizador de gerenciamento. Em princípio, determinadas partes da interface serão codificadas como códigos transportáveis (*Applets Java*) que poderão ser executado por qualquer *Browser* com capacidades Java.

Sob a ótica de gerenciamento, a Figura 3.2 apresenta uma visão de alto nível da arquitetura da ferramenta NetTracker, que é composta por cinco camadas interdependentes: *Módulos de Programas de Aplicação (APM)*, *Interface de Programas de Aplicação (API)*, *Módulos de Extensão (EM)*, *Biblioteca de Objetos Java e Máquina Virtual Java*. A seguir são descritos os papéis funcionais de cada uma destas

Figura 3.2 - Arquitetura Operacional NetTracker

camadas.

3.1 Camada de Módulos de Programas de Aplicação

A camada APM trata dos aspectos funcionais agregados ao sistema NetTracker, incorporando as aplicações responsáveis por proverem as funções de gerenciamento e de comunicação com o administrador. Como exemplos de elementos componentes desta camada, podemos citar a aplicação de Filtragem de Eventos SNMP, juntamente com as aplicações de gerenciamento de ações (*Trigger*) e de busca e reconhecimento automático da topologia da rede (*Discovery*).

3.2 Camada de Interface de Programas de Aplicações

Nesta camada, encontram-se disponíveis as interfaces de programação que fornecem acesso às funções implementadas pelos Módulos de Extensão. No contexto da tecnologia de objetos, que será amplamente empregado neste projeto, esta camada é composta por todas as interfaces dos objetos codificados na camada EM.

3.3 Camada de Módulos de Extensão

Todos os objetos responsáveis por proverem funções básicas de apoio às funções de gerenciamento serão integrados à esta camada. Tais funções desempenharão o papel de alicerce para os programas de aplicações que se encontram na camada superior. Basicamente, esta é composta pelos seguintes sub-módulos:

- **HTTPServer** - Permite que a Ferramenta de gerenciamento disponibilize serviços de configuração remota através de requisições HTTP. Este módulo possibilita que as interfaces de usuário possam ser transparentemente enviadas ao administrador, independentemente de sua localidade no ambiente.
- **Database** - Este módulo é responsável por prover, de forma transparente, acesso à mais variada gama de SGBDs disponíveis no mercado. Para tal prevê-se a utilização do protocolo JDBC (*Java Database Connection*) ou ainda de *drivers* nativos oferecidos pelos fornecedores dos produtos. Em princípio, no protótipo, é utilizado a implementação do mSQL para a plataforma Unix®.
- **SNMP** - todas os recursos de gerenciamento presentes no protocolo SNMP serão agrupados neste módulo. Em uma primeira instância, serão agregadas as funcionalidades de uma implementação de domínio público. Prevê-se, contudo, a mobilização de esforços no sentido de implementá-lo no futuro.
- **SNMPv2** - com propósito semelhante ao anterior, este módulo disponibilizará as funções de gerenciamento da nova versão do SNMP (SNMPv2). Existe um trabalho paralelo, de implementação deste módulo, sendo efetuado como parte do programa de mestrado do ICMSC.
- **Security** - este módulo tem o objetivo de disponibilizar todas as funções que tratem dos aspectos de segurança da ferramenta NetTracker (i. é, SSL, MD5, DES, RSA, S/Key, entre outros). Além do mais, existem perspectivas em se incorporar funcionalidades extras de segurança ao NetTracker mediante a integração de um módulo adaptativo de detecção de intrusão baseado em redes neurais.

3.4 Biblioteca de objetos Java

Nesta camada encontram-se todos os objetos que compõem as classes da linguagem Java, e que são responsáveis por permitirem a implementação de grande parte das camadas superiores.

3.5 Máquina Virtual Java

Sem dúvida, é o coração da nova arquitetura do NetTracker. Sobre ela estão assentadas todas as principais características inovadoras da ferramenta: carga e descarga dinâmica de módulos, independência de plataforma, linkedição dinâmica e em tempo de execução de classes, acesso remoto a objetos, códigos binários transportáveis, etc. Está previsto, para o futuro, a incorporação de um *Just-in-Time Compiler* (JIT), que nada mais é do que um compilador em tempo de execução, mas que possibilitará incrementar em muito a velocidade de execução do sistema.

4. A estrutura e o relacionamento dos objetos no sistema.

A ferramenta NetTracker é composta por duas categorias principais de objetos: Objetos de Sistema e Objetos de Aplicação. Os primeiros definem os objetos que integram o ambiente básico de execução do sistema, efetuado a carga dinâmica e posterior ativação dos Objetos de Aplicação. Estes últimos, por sua vez, definem de fato as aplicações do sistema NetTracker. A Figura 4.1 nos dá uma visão mais detalhada desta estrutura organizacional.

Figura 4.1 - Estruturação dos Objetos NetTracker

Desta forma, o processo de ativação e execução do sistema NetTracker pode ser descrito pelos seguintes passos:

1. Após o processo de ativação da Máquina Virtual Java, a classe principal do sistema (NetTracker) é carregada, checadadas suas dependências e posteriormente ativada. Este procedimento é comumente aplicado às aplicações Java;
2. Após ser instanciado pela etapa anterior, e de posse do controle executivo, o método construtor configura o sistema criando as estruturas de suporte e controle necessárias. Neste processo são instanciados e ativados os objetos ditos essenciais, tais como MVHttpServer, MVManager, MVLoader, entre outros. O primeiro funciona como um servidor HTTP e fica responsável por permitir o interfaceamento do sistema NetTracker com o administrador da rede, através de um *Browser Web* qualquer. Já o segundo é responsável por controlar a criação e o gerenciamento dos objetos NetTracker (MVApplication) que são instanciados a partir de classes carregadas dinamicamente pelo ambiente. O objeto MVLoader é tratado logo a seguir. Ao término desta etapa, todos os objetos de sistema estão estabelecidos e em processo de execução;
3. Ativado como uma *thread* Java, o objeto MVLoader inicia o procedimento de carga e registro de todos os objetos de aplicação. Para tal é examinado o arquivo de cadastro de aplicações (MVAppReg.cfg), buscando e efetuando a carga dinâmica de cada um dos módulos de aplicações cadastradas. A princípio estão cadastradas as aplicações de suporte ao sistema (*ManagerSNMP*, *Logger*, *Trigger*, *Database* e *Trapper*).
4. Após a carga, e durante o processo de ativação dos módulos de aplicação mencionados anteriormente, cada qual efetua o registro de seus atributos em um objeto de registro de atributos (MVApplicationAttributes). Isto é necessário, pois, através deste registro é possível o controle e monitoração de cada atributo pelo gerenciador do sistema (MVManager).

Ao término deste processo de ativação o sistema está plenamente operacional e apto a interagir com o administrador da rede. Segue-se uma descrição das aplicações de suporte ao sistema.

4.1 *ManagerSNMP*

A aplicação *ManagerSNMP* é responsável por gerenciar as informações de gerenciamento SNMPv2 contidas em MIBs (*Management Information Base*) controladas e mantidas por agentes SNMPv2 (MCC91). O processo de obtenção de informações é realizado de duas formas: *polling*, a aplicação gerente faz requisições periódicas aos clientes; e o *trapping*, que são mensagens enviadas ao gerente sobre algum evento anormal ou pré-determinado (Arquivo de configuração *ManagerSNMP.conf*), sem que estas tenham sido requisitadas pelo gerente.

Seguindo a estrutura das aplicações do NetTracker tem-se que o objeto *ManagerSNMP* é composto das seguintes classes: *ManagerSNMP*, *MibControl* e *MibTrapper* que se interrelacionam com o *Multview Extension Module*.

- *ManagerSNMP*: Implementa as funções que monitoram os limites aceitos para as variáveis do ambiente gerenciado, para que no momento que estas variáveis alcançarem os seus limiares o gerente SNMP dispare um evento que irá tratar o problema.
- *MibControl*: As funções de monitoramento e controle dos agentes SNMP serão implementadas nesta classe, no qual a interface as aplicações será realizada através de páginas HTML.
- *MibTrapper*: Responsável por receber as notificações (Traps) enviadas pelos agentes SNMP da rede.

4.2 *Logger*

A função do *Logger* é gerenciar os arquivos de *log* das aplicações do sistema NetTracker, que devem referenciar o objeto *Logger* para poder enviar as mensagens. Estas mensagens possuem três campos: *type* (determina o tipo da mensagem), *priority* (define a prioridade de tratamento da mensagem) e *message* (mensagem que será armazenada no log). A determinação em qual arquivo de *log* será armazenada a mensagem é de responsabilidade do administrador, que através de uma interface HTTP (via método *perform()*) determina qual arquivo está relacionado ao par (*type*, *priority*).

4.3 *Trigger*

O *Trigger* é responsável por executar funções ou tarefas que sejam dependentes de algum evento que possa ocorrer no ambiente. Estas funções são configuradas seguindo um relacionamento com algum determinado evento, desta forma poderíamos ter por exemplo: o recebimento de uma notificação da perda de comunicação entre um par de roteadores poderia disparar uma tarefa que estabeleceria uma rota alternativa, através da inserção de uma rota extra na tabela de roteamento de um roteador disponível.

4.4 *Database*

A *Database* gerencia as consultas à base de dados realizadas pelas aplicações, executando os comandos SQL e retornando o resultado da consulta. Para se ter acesso as funcionalidades do *Database* as aplicações irão referenciar o objeto, que possui todos os métodos necessários para a criação de manutenção da base, por exemplo; as informações recebidas e coletadas pelo *ManagerSNMP* são armazenadas em uma base de dados para que estas estejam disponíveis às outras aplicações, que também farão acesso às informações através do *Database*.

4.5 Trapper

Esta aplicação é responsável por receber notificações de ocorrências de eventos importantes, que possam vir a ocorrer em um agente SNMP ou SNMPv2, e filtrá-las de modo a apresentar informações baseadas em padrões e critérios pré-definidos pelo administrador da rede. Desta

Figura 4.2 - Objetos da Aplicação de Filtro de Traps

forma pode-se estabelecer ações que possam ser disparadas automaticamente pelo sistema, em reação a um estímulo fornecido por um dos filtros.

Em princípio está sendo tratado apenas o protocolo SNMP via Advent API (ADV96), dado a falta de uma API SNMPv2 disponível. Existe, entretanto, um trabalho que está sendo desenvolvido que visa implementar uma API SNMPv2, a qual poderá ser incluída posteriormente no sistema NetTracker. A composição de objetos e o interrelacionamento destes, dentro da aplicação de filtragem de eventos, pode ser observado na Figura 4.2. De fato, iniciada através da carga e ativação da classe principal (*Trapper*), são criados e ativados os objetos *TrapReceiver*, *TrapFilter*, cujas funções estão fora do escopo deste artigo. O *Trapper*, por outro lado, atua como o objeto essencial da aplicação, implementando as funções *run* e *perform*.

5. Definição e operação das aplicações

Cada aplicação ou módulo de aplicação a ser integrado ao sistema é composto por um conjunto de definições de objetos, os quais interagem entre si para proporcionar a funcionalidade necessária e esperada. Para tal, um destes objetos, dito objeto de ativação, deve obedecer a duas regras essenciais:

- A definição (Classe) de um objeto principal de aplicação deve necessariamente herdar os atributos e métodos da superclasse *MVApplication* (ver Figura 5.1). Esta última, por sua vez, define os métodos abstratos de interfaceamento dos objetos do sistema, apresentados anteriormente, com os objetos de aplicação;
- Todo objeto de ativação deve implementar obrigatoriamente os métodos *run* e *perform* (Figura 5.3). O primeiro deve codificar as instruções e operações que estabelecem, de fato, a aplicação no ambiente operacional do sistema, e será invocado logo após sua carga pelo mesmo. O último deve conter código de manuseio de solicitações HTTP. Em outras palavras estes métodos podem ser descritos como o coração e o cérebro da aplicação.


```

/* @(#) MVManager.java r0.1 alpha      16/12/96 Rogério Borges Mouro
 *
 * NetTracker 1.0 (LEGO)
 * Copyright (c) 1996, University of São Paulo at São Carlos.
 * Intermedia Research Lab. All Rights Reserved.
 */
public class MVTrapper extends MVApplication {
    ...
} // end of MVManager

```

Figura 5.1 - Definição de uma aplicação NetTracker

Além da obrigatoriedade do cumprimento destas regras, e caso haja necessidade de se definir

```

...
protected run() {
    while( alive ) {
        thread.suspend();

        ...

    }
    end();
} // end of run method

protected Reply perform( Request request ) {
    Reply reply = null;
    String URI = request.getURI();

    ...

    return reply;
} // end of perform method

```

Figura 5.3 - Funções essenciais

atributos de controle para a aplicação, cada um destes atributos devem ser definidos utilizando-se os objetos preestabelecidos para este propósito (MVInteger, MVFloat, MVString, MVBoolean e MVByte). É importante ainda ressaltar que cada atributo deverá registrar-se explicitamente, através do método **register**, no objeto de atributos (MVApplicationAttributes) da aplicação do qual fazem parte.

Seguindo estas regras na definição de uma aplicação, seu relacionamento com outros objetos no sistema se dá de forma transparente, mesmo quando as classes alvo do relacionamento ainda não estão carregadas e disponíveis. Neste caso o próprio ambiente Java se incumba de resolver tais dependências, carregando-as e ativando-as quando necessário.

6. Conclusões.

A ferramenta NetTracker apresenta-se amplamente flexível no sentido de permitir a inclusão de aplicações e, consequentemente, de novas tecnologias, sem a necessidade de recodificação das partes já existentes. É evidente que esta capacidade somente é verificada caso não haja dependência das aplicações anteriores com a aplicação que esteja sendo incorporada (i.e. uma aplicação já funcional não pode ter acesso aos métodos de uma aplicação desenvolvida posteriormente a ela).

Em termos do desempenho, é evidente que o ônus operacional introduzido pela máquina virtual Java traz como consequência uma degradação geral no processamento da ferramenta. Entretanto é factível a incorporação um módulo JIT (*Just-in-time compile*), na base da máquina virtual, que agiria de forma

perceptível no incremento da velocidade de execução da ferramenta e, conseqüentemente, das aplicações por ele controladas. É importante salientar que este módulo JIT não introduziria penalidades adicionais de portabilidade, pois o código binário gerado por demanda é compatível com a arquitetura de processamento local.

De fato, a única preocupação de quem venha a desenvolver aplicações de gerenciamento que tenham como meta agregar funcionalidades extras ao NetTracker, e a de codificar as funções *run* e *perform* no corpo de suas aplicações, como discutidas anteriormente, além de utilizar adequadamente os recursos de classes e aplicações já implementadas..

7. Referências

- [ADV96] Advent Network Management, Inc., *Uniform Resource Locators* (URL), <http://www.advent.com/index.html>, 1996.
- [ANS87] ANSI X3T5/87-305: Tutorial Material for Development of OSI Management Model Section V - Information Model, Novembro de 1987.
- [BEN95] Bem-Natan, R., *A guide to Common Object Request Broker Architecture*, McGraw Hill, 1995
- [BOO91] Booch, G.: *Object Oriented Design with Applications*. The Benjamin/Cummings Publishing Co., Inc., 1991
- [CAS90] CASE, J. et. al. *A Simple Network Management Protocol (SNMP)*, Network Working Group, *Request for Comments* 1157, 1990
- [CAS93a] Case, J. et. al. *Introduction to version 2 of the Internet-standard Network Management Framework*. SNMP Research, Inc., Hughes LAN System, Dover Beach Consulting, Inc., Carnegie Mello University, *Request for Comments* 1441, 1993.
- [CAS93b] CASE, J. et. al. *Manager-to-Manager Management Information Base*. SNMP Research, Inc., Hughes LAN System, Dover Beach Consulting, Inc., Carnegie Mello University, *Request for Comments* 1451, 1993.
- [DAV97] Dave, R., *Uniform Resource Locator* (URL), <http://www.w3.org/pub/WWW/TR/REC-html32.html>, 1997.
- [DER96] Deri, L, *Surfin' Network Resources across the Web*, *Uniform Resource Locator* (URL), <http://misa.zurich.ibm.com/Webbin/~Ide/Surfin.ps.Z>, 1996.
- [FIE97] Fielding, R, et al, *Hypertext Transfer Protocol - HTTP/1.1*, *Request for Comment* 2068, 1997.
- [FRE96] Freir, A., O.; Karlton, P.; Kacher, P., C., *The SSL Protocol version 3.0*, Transport Layer Security Working Group, *Uniform Resource Locator* (URL), <ftp://ds.internic.net/internet-drafts/draft-ietf-tls-ssl-version3-00.txt>, 1996.
- [GOS96] Gosling, J.; Joy, B.; Steele, G., *The Java Language Specification*, Addison-Wesley, 1996.
- [KLE88] Klerer, S. M.: *The OSI Management Architecture: An Overview*. IEEE Network, Vol. 2, No. 2, pg. 20-29, Março de 1988.
- [LIN97] Lindholm, T., et al., *The Java Virtual Machine Specification*, Addison-Wesley, 1997.
- [MCC91] McCloghrie, K.; ROSE, M. T., *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. (DDS Network Information Center, SRI International), *Request for Comment* 1213, 1991.
- [RIV78] Rivest, R., L.; Shamir, A.; Adleman, L. M., *A method for obtain digital signature and public-key cryptosystems*, *Communications of the ACM*, 21(2), p.120-126, 1978.

- [ROS90] Rose, M. T., Structure and Identification of Management Information for TCP/IP-based internets. (DDS Network Information Center, SRI International), Request for Comment 1155, 1990
- [ROS91] Rose, M. T., The Simple Book - *An Introduction to Management of TCP/IP-based Internets*. Englewood Cliffs, Prentice-Hall, 1991.
- [STE90] Stevens, R., Unix Network Programming, Prentice Hall, 1990. . .
- [STA93] Stallings, W., SNMP, SNMPv2, and CMIP - *The Pratical Guide to Network-Management Statandards*. Addison-Welley, 1993.
- [TAN89] Tanenbaum, A., S., Computer Network 2.ed, Englewood Cliffs, Prentice Hall, 1989.