UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM AGENTE SNMP PARA UMA REDE LOCAL UTILIZANDO A PLATAFORMA JDMK

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

JORGE LUCAS DE MELLO

BLUMENAU, JUNHO / 2000.

PROTÓTIPO DE UM AGENTE SNMP PARA UMA REDE LOCAL UTILIZANDO A PLATAFORMA JDMK

JORGE LUCAS DE MELLO

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

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

Prof. Francisco Adell Péricas — Orientador na FURB

Prof. José Roque Voltolini da Silva - Coordenador do TCC

BANCA EXAMINADORA

Prof. Francisco Adell Péricas

Prof. Sergio Stringari

Prof. Marcel Hugo

AGRADECIMENTOS

Agradeço a todos os professores do curso de Bacharelado em Ciências da Computação da Universidade Regional de Blumenau pela ajuda prestada ao longo do curso, agradeço principalmente ao professor Francisco Adell Péricas por ter me orientado neste trabalho final e por ter muita paciência, preocupação e dedicação.

Agradeço a todos os novos amigos que conheci em Blumenau no decorrer da Faculdade, principalmente ao Bogo e Luiz de Óculos amigões mesmo, Marlon, Fabio Segundo, Heleno e Raphael.

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

Agradeço a várias pessoas que me acompanharam nessa jornada como minha irmã Luana e meu grande amigo Marcelo.

Agradeço principalmente aos meus pais Antonio e Gessi por terem me dado essa oportunidade tão valiosa e se não fosse a dedicação deles, não poderia estar redigindo meu trabalho final.

Agradeço principalmente a Andreia Krambeck, por ser a pessoa mais especial que já conheci e por ter compartilhado dos momentos mais marcantes da minha vida.

E finalmente agradeço a pessoa mais importante de todas, obrigado por tudo DEUS.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

ACL	*Access Control List*
API	*Application Program Interface*
ASCII	*American Standard Code for Information Interchange*
ASN.1	*Abstract Syntax Notation One*
CMIP	*Common Management Information Protocol*
EGP	*Exterior Gateway Protocol*
FDDI	*Fiber Distributed Data Interface*
FTP	*File Transfer Protocol*
IEEE	*Institute of Electrical and Electronics Engineers*
IP	*Internet Protocol*
ISO	*International Organization for Standardization*
HTML	*Hipertext Markup Language*
HTTP	*Hipertext Transfer Protocol*
JDMK	*Java Dynamic Management Kit*
JMX	*Java Management Extensions*
LAN	*Local Area Network*
MIB	*Management Information Base*
NMS	*Network Management Station*
OID	*Object Identifier*
PC	*Personal Computer*
PDU	*Protocol Description Unit*
RFC	*Request for Comment*
RMI	*Remote Method Invocation*
SMTP	*Simple Mail Transfer Protocol*
SNMP	*Simple Network Management Protocol*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
WAN	*Wide Area Network*

RESUMO

Este trabalho apresenta um estudo na aplicação do protocolo de gerenciamento *Simple Network Management Protocol* (SNMP) através da especificação e implementação de uma aplicação de um agente SNMP para dispositivos de uma rede *Local Área Network* (LAN), utilizado na implementação a linguagem Java e a plataforma de desenvolvimento *Java Dynamic Management Kit* (JDMK).

ABSTRACT

This paper presents a study in the application of the management protocol Simple Network Management Protocol (SNMP) through of the specification and implementation of an application of an agent SNMP for devices of a Local Area Network (LAN), used in the implementation the language Java and the development platform Java Dynamic Management Kit (JDMK).

1. INTRODUÇÃO

Através dos avanços das tecnologias de interconectividade e dos benefícios proporcionados pelas Redes de Computadores, cada vez mais computadores são interconectados nas organizações. Paralelamente, a diminuição dos custos dos equipamentos permite adquirir e agregar à rede cada vez mais equipamentos, de tipos diversos, tornando essas redes cada vez maiores e mais complexas. Com isso, as redes começaram a ser interconectadas muito rapidamente; redes locais conectadas a redes regionais, as quais, por sua vez, ligadas a *backbones* nacionais.

Hoje essas redes são extremamente importantes para o dia-a-dia de muitas empresas em todo o mundo, porque, junto com sua utilização, vem a eficácia e a competitividade. Essa importância vem crescendo de tal forma que as empresas têm se tornados altamente dependentes destas redes, sentindo imediatamente o impacto quando os seus recursos ficam indisponíveis ([COH2000]).

Este novo ambiente originou alguns problemas administrativos. As tarefas de configuração, identificação de falhas e controle de dispositivos e recursos da rede passaram a consumir tempo e recursos das organizações.

Cientes desse problema, a solução então passou a ser buscada na atividade de Gerenciamento de Rede. Esta atividade passou a evoluir de forma rápida e concisa, sendo hoje uma das especialidades da área de Redes de Computadores que mais cresce. Apesar desses avanços, ainda hoje, é difícil conceituar esta atividade.

Segundo [COH2000], ao longo da sua evolução muitas definições foram propostas para a Gerência de Redes. Resume-se a seguir algumas dessas definições:

a) consiste no controle e administração de forma racional dos recursos de hardware e software em um ambiente distribuído, buscando melhor desempenho e eficiência do sistema;

b) consiste no controle de uma rede de computadores e seus serviços;

c) tem por objetivo maximizar o controle organizacional das redes de computadores, de maneira mais eficiente e confiável, ou seja, planejar, supervisionar, monitorar e controlar qualquer atividade da rede;

d) consiste na detecção e correção de falhas, em um tempo mínimo, e no estabelecimento de procedimentos para a previsão de problemas futuros.

É clara a necessidade de prover monitoramento e controle sobre todos os componentes da rede, de forma a garantir que estes estejam sempre em funcionamento e que os problemas sejam identificados, isolados e solucionados o mais rápido possível. Entretanto, esta não é uma tarefa fácil. As redes das empresas têm assumido grandes proporções, com um grande número de computadores, além da constante inclusão de novos componentes, oferecendo integração dados/voz, multiplexadores e roteadores, entre outros, o que tem adicionado mais complexidade a esse ambiente.

Para atender a esta necessidade de gerenciamento foram desenvolvidos protocolos de gerenciamento. A principal preocupação de um protocolo de gerenciamento é permitir aos gerentes de rede realizar tarefas, tais como: obter dados sobre desempenho e tráfego da rede em tempo real, diagnosticar problemas de comunicação e reconfigurar a rede atendendo a mudanças nas necessidades dos usuários e do ambiente ([CHI1999]). Porém, vários obstáculos teriam que ser superados, entre eles a heterogeneidade dos equipamentos de rede (computadores, roteadores e dispositivos de meio), dos protocolos de comunicação e das tecnologias de rede. Adicionalmente, era necessário que esse gerenciamento fosse integrado, pois uma solução genérica e integrada auxiliaria os usuários a evitar os altos custos de uma solução específica, além de facilitar a manutenção, o monitoramento, o crescimento e a evolução da rede. Sem um gerenciamento integrado, a rede poderia degradar até se tornar completamente ineficiente.

Segundo [STA1996], as informações de gerenciamento permitem, dentre outras tarefas, produzir registros de auditoria para todas as conexões, desconexões, falhas e outros eventos significativos na rede. Esses registros, por sua vez, permitem determinar futuras necessidades de adicionar equipamentos, identificar e isolar erros comuns de um cliente, além de estudar outras tendências de uso. Deve-se destacar ainda, que estas informações

devem possibilitar uma atuação preventiva, e não meramente reativa, com relação aos problemas.

Ciente destes requisitos das empresas, a ISO (*International Organization for Standardization*) vem desenvolvendo padrões para o gerenciamento de redes OSI (*Open Systems Interconnection*), tendo como protocolo de gerência o CMIP (*Common Management Information Protocol*). Por outro lado, existe o IEEE (*Institute of Electrical and Electronics Engineers*) com um conjunto de padrões para o gerenciamento de redes TCP/IP (*Transmission Control Protocol* / *Internet Protocol*), chamado SNMP (*Simple Network Management Protocol*). Atualmente, quase todas as plataformas de gerenciamento de redes Internet comercialmente disponíveis implementam o protocolo SNMP devido à sua simplicidade de implementação em relação ao CMIP.

1.1 OBJETIVOS

O trabalho desenvolvido tem como objetivo estudar e aplicar o protocolo de gerenciamento SNMP através da especificação e implementação de uma aplicação agente SNMP em Java, para o gerenciamento de dispositivos de uma rede LAN, utilizando-se a plataforma de desenvolvimento de aplicações de gerência JDMK da SUN.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em capítulos, conforme é apresentado a seguir.

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

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

O capítulo 3 refere-se à plataforma JDMK, abordando sua funcionalidade, seus benefícios e as ferramentas utilizadas.

O capítulo 4 aborda os MBeans, apresentando as suas funções e qual a sua necessidade no gerenciamento da rede.

O capítulo 5 é voltado ao desenvolvimento do protótipo, onde se apresenta a especificação e implementação do mesmo.

No capítulo 6 é dedicado às conclusões e sugestões de continuidade do trabalho.

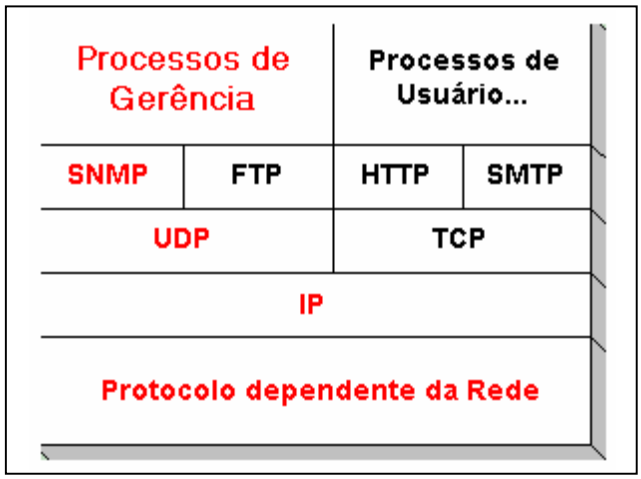
E por fim, o capítulo 7 apresenta as bibliografias utilizadas para todo o desenvolvimento deste trabalho.

2. PROTOCOLO DE GERENCIAMENTO SNMP

O SNMP foi desenvolvido nos anos 80 como resposta para os problemas de gerenciamento em ambientes TCP/IP, envolvendo redes heterogêneas. Inicialmente foi concebido para ser apenas uma solução provisória até o desenvolvimento de um protocolo de gerenciamento mais completo, o CMIP (*Common Management Information Protocol*). Neste contexto, sem um protocolo melhor disponível, o SNMP passou a ser o protocolo mais utilizado ([ODA1998]).

O *Simple Network Management Protocol* (SNMP) é um protocolo da camada de aplicação, como mostrado na Figura 1, desenvolvido para facilitar a troca de informações de gerenciamento entre dispositivos de rede. Estas informações transportadas pelo SNMP (como pacotes por segundo e taxa de erro na rede), permitem aos administradores da rede gerenciar o desempenho da rede de forma remota, encontrando e solucionando os problemas e planejar o crescimento da rede.

FIGURA 1 - PROTOCOLO DE GERENCIAMENTO SNMP



Fonte: [COH2000]

2.1 DESCRIÇÃO DE SNMP

Segundo [ODA1998], o SNMP define uma relação cliente/servidor. O programa cliente (chamado de gerenciador de rede) faz conexões virtuais a um programa servidor

(chamado de agente SNMP) que executa em um dispositivo de rede remoto e disponibiliza informação para o gerente relativo ao estado do dispositivo. O banco de dados, que modela o agente SNMP é denominado *Management Information Base* (MIB) e sua função padrão é controlar valores gerenciados no dispositivo. O SNMP permite a extensão destes valores padrões adicionalmente com valores específicos para um agente particular pelo uso de MIB privados.

Diretivas emitidas pelo gerenciador da rede a um agente SNMP consistem nos identificadores de variáveis de SNMP (chamados identificadores da MIB ou variáveis da MIB) junto com instruções para adquirir o valor do identificador ou fixar o identificador para um novo valor.

Pelo uso de variáveis de MIB privadas, agentes SNMP podem ser incrementadas para gerenciar dispositivos específicos, como *bridges*, *gateways* e roteadores. As definições de variáveis de uma MIB podem suportar um agente em particular. As MIB são escritas num formato padronizado chamado de *Abstract Syntax Notation One* (ASN.1) ([CHI1999]).

2.2 PONTOS POSITIVOS E NEGATIVOS DO SNMP

O SNMP tem vários pontos positivos. Um deles é sua popularidade para a gerência de redes TCP/IP. Agentes SNMP estão disponíveis para vários dispositivos de rede, desde computadores até *bridges*, *modems* ou impressoras ([STA1996]).

Adicionalmente, o SNMP é um protocolo de gerenciamento flexível e extensível. Pode-se estender os agentes SNMP para cobrir dados específicos de dispositivos, pelo uso de arquivos ASN.1. O SNMP pode assumir numerosos trabalhos específicos para classes de dispositivos fornecendo um mecanismo padrão de controle de rede e monitoramento.

Segundo [ODA1996], podem ser identificados alguns pontos fracos do SNMP. Apesar de seu nome, (i.e. *"Simple" Network Management Protocol*), o SNMP é um protocolo relativamente complexo para implementar. Também, o SNMP não é um protocolo muito eficiente. Os modos nos quais são identificadas as variáveis SNMP (como

strings de *byte* onde cada *byte* corresponde a um nodo particular no banco de dados da MIB) conduz desnecessariamente a grandes pacotes de dados PDU (*Protocol Description Unit*), que consomem partes significativas de cada mensagem de SNMP, sobrecarregando a rede de transmissão de dados.

As citações acima não podem ser usadas para diminuir os pontos positivos mencionados no SNMP. Enquanto codificações complicadas frustram programadores, a utilização por parte dos usuários finais não dependem da complexidade dos algoritmos que disponibilizam os dados a eles.

2.3 ELEMENTOS DO SNMP

O SNMP possui uma característica cliente/servidor, onde o cliente seria o gerenciador da rede e o servidor o agente SNMP e a base que modela este agente denomina-se MIB.

2.3.1 AGENTES

No modelo de gerenciamento SNMP, *hosts*, *bridges*, roteadores, *hubs*, etc, devem ser equipados com agentes SNMP para que possam ser gerenciados pela estação de gerenciamento (NMS) através do gerente SNMP. O agente responde a requisições da estação de gerenciamento, que pode ser o envio de informações de gerência ou ações sobre as variáveis do dispositivo onde está.

O funcionamento desta estrutura só é possível graças ao acesso direto à MIB (*Management Information Base*) que o agente possui, pois todas as informações de gerência encontram-se lá ([CHI1999]). Ao receber uma mensagem SNMP do gerente, o agente identifica que operação está sendo requisitada e qual(is) a(s) variável(is) relacionada, e a partir daí executa a operação sobre a MIB; em seguida, monta uma nova mensagem de resposta, que será enviada ao gerente.

À primeira vista, a comunicação do agente com o gerente pode parecer injusta, uma vez que o agente apenas responde ao que lhe é questionado. Mas há momentos em que o agente pode "falar" espontaneamente com o gerente sem que antes seja questionado. Isso ocorre quando o agente detecta, a partir da análise do contexto da MIB, alguma situação

inesperada. Neste momento, o agente gera uma mensagem especial, o Trap, e a envia ao gerente, relatando sobre a situação.

Para o tratamento destas exceções e o envio de Trap, é concedido ao agente um certo poder de decisão, cabendo a ele, a partir da análise do contexto da MIB, decidir se é ou não necessário enviar o Trap ao gerente. Esse poder de decisão é concedido ao agente para que em certas situações, como quando da inicialização do sistema, Trap desnecessários não sejam trafegados pela rede, o que, em se tratando de dezenas de agentes, poderia interferir no desempenho global da rede.

Cabe ao agente um papel fundamental em todo o processo de gerenciamento da rede, acessando e disponibilizando informações de gerência contidas na MIB, além de indicar situações inesperadas de funcionamento do dispositivo que estiver gerenciando através do envio de Trap ao gerente.

2.3.2 GERENTES

A interface entre as aplicações de gerência correntes no NMS e os agentes espalhados pelos dispositivos da rede é o gerente. Cabe ao gerente enviar comandos aos agentes, solicitando informações sobre variáveis de um objeto gerenciado ou modificando o valor de determinada variável.

Os gerentes então processam estas informações colhidas pelos agentes e as repassam à aplicação que as requisitou. A comunicação entre o gerente e as aplicações é possível através da utilização das API do gerente SNMP pelo sistema.

A API (*Application Program Interface*) é um conjunto de funções que fazem o intermédio na execução de comandos entre um programa e outro, de forma a simplificar a um programa o acesso a funções de outro programa e que, no caso do SNMP, fazem o intermédio das execuções entre uma aplicação de gerência e o gerente SNMP [TAN1996].

Quando uma solicitação da aplicação de gerência chega ao gerente, através da API, o gerente mapeia esta solicitação para um ou mais comandos SNMP que, através da troca de mensagens apropriadas, chegarão aos agentes correspondentes. Deve-se ressaltar que este comando SNMP montado pelo gerente pode ser enviado a um outro gerente, que

pode ou não repassá-lo a um agente. Só que a comunicação gerente-gerente só é possível na versão estendida do SNMP, o SNMPv2, e não faz parte do SNMP padrão.

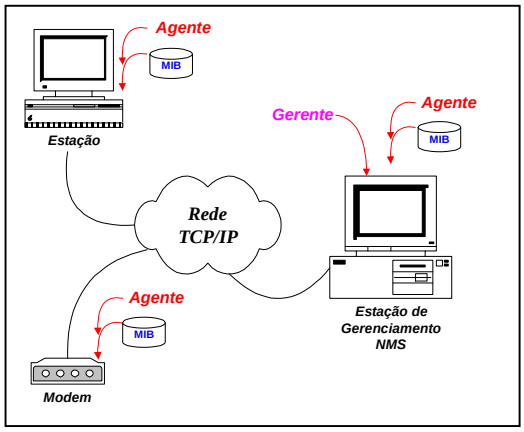
Cabe também ao gerente encaminhar à aplicação de gerência os Trap que porventura sejam enviados pelos agentes. Assim, o software de gerência terá conhecimento da presença de um novo equipamento na rede ou do mal funcionamento de algum dos dispositivos da rede.

Quando da inicialização do software de gerência, nada se sabe sobre a configuração ou funcionamento da rede. Essas informações vão sendo conhecidas através de Trap que são enviados pelos agentes; a partir daí o gerente realiza *polling* [TAN1996] a fim de manter a comunicação com os agentes, possibilitando ao software de gerência mapear, monitorar e controlar a rede.

2.3.3 *MANAGEMENT INFORMATION BASE* (MIB)

Segundo [COH2000], no modelo SNMP, os recursos de uma rede são representados como objetos. Cada objeto é, essencialmente, uma variável que representa um aspecto do dispositivo gerenciado. Todos os objetos (variáveis) são armazenados na *Management Information Base* (MIB).

FIGURA 2 - ELEMENTOS BÁSICOS DO SNMP



Fonte: [COH2000]

2.4 OPERAÇÕES SNMP APLICAVÉIS ÀS VARIÁVEIS DA MIB

No gerenciamento SNMP existem várias operações para a comunicação entre os gerentes e agentes SNMP para obter informações dos dispositivos gerenciados.

2.4.1 GET

O gerente SNMP envia o comando Get a um determinado agente toda vez que necessita recuperar uma informação de gerenciamento específica do objeto gerenciado pelo agente. Estas informações encontram-se na forma básica de variáveis, que por sua vez estão na MIB do elemento de rede gerenciado.

2.4.2. GETNEXT

O comando GetNext assemelha-se ao comando Get, no entanto, enquanto o comando Get solicita ao agente a leitura de determinada instância de uma variável, ao receber um comando GetNext, o agente deve ler a próxima instância disponível, na ordem especificada pela MIB, da(s) variável(is) associada.

Esta operação é especialmente na recuperação de uma tabela de instâncias da MIB cujo tamanho seja desconhecido. Para isto, é inicialmente especificado no comando GetNext o nome da tabela, o que retornará como resposta o primeiro item (neste caso, uma instância de variável) da tabela. Com o nome do primeiro item da tabela, o gerente pode enviar um outro GetNext, desta vez passando a resposta do GetNext anterior como parâmetro, a fim de recuperar o próximo item da tabela, e assim sucessivamente até recuperar todas as instâncias da variável na tabela. Para este caso, se o comando Get fosse utilizado ele falharia, pois o nome da tabela não corresponde a uma instância individual da variável.

2.4.3. SET

A operação Set requisita a um determinado agente a atribuição/alteração do valor de determinada(s) variável(is) de uma MIB. Alguns desenvolvedores, por exemplo, acreditam que este comando não deve retornar um Response. Já outros acham que a

operação Set deve retornar alguma indicação de que a operação foi efetuada. Porém o mais correto seria que após cada operação Set sobre uma variável, uma operação Get fosse efetuada sobre a mesma variável a fim de assegurar que a operação Set foi efetuada.

2.4.4. TRAP

A operação Trap difere de todas as outras. Ela é utilizada por um agente SNMP para notificar de forma assíncrona a um gerente algum evento extraordinário que tenha ocorrido no objeto gerenciado.

Diversos questionamentos são feitos quanto a esta operação. Talvez o maior deles seja: quais eventos devem realmente ser notificados ao gerente? Embora todos concordem que o gerente deva ser informado de alguns eventos significativos, muitos fornecedores de produtos que implementam o SNMP trazem Trap's específicos, muitos deles desnecessários ([STA1996]).

Outro importante questionamento é: Já que a operação Trap não gera um Response, como um agente pode ter certeza de que o gerente o recebeu? Esta preocupação se perde quando a rede está em perfeito funcionamento. Mas e se a máquina estiver, por exemplo, perdendo pacotes? Isso não é fácil de solucionar. Uma possibilidade seria o agente gerar tantos Trap's quanto necessário até que o gerente seja informado do evento. Essa solução, no entanto, aumentaria o tráfego na rede, afetando o seu desempenho (que já está comprometido).

2.4.5. RESPONSES

Como já descrito, sempre que um agente recebe um comando Get, GetNext ou Set ele tenta executar a operação associada e, conseguindo ou não, constrói uma outra mensagem que é enviada ao emissor da requisição. Esta mensagem é a GetResponse. Das operações SNMP, apenas o Trap não gera um Response.

2.5 OBJETOS PADRONIZADOS DE GERENCIAMENTO

O conjunto de objetos que constituem o modelo de informação utilizado por um sistema de gerenciamento através do protocolo SNMP é denominado de "*Management Information Base*" (MIB). Este termo é freqüentemente utilizado para descrever qualquer objeto SNMP ou uma parte dele. No sentido exato, "MIB" simplesmente é uma abstração, como "Banco de dados", que pode ser aplicada para analisar todos os dados de uma rede ou de uma parte dela. Esta nomenclatura casual confunde freqüentemente aos novos usuários ([SOA1995]).

As várias variáveis SNMP da MIB padrão são definidas na RFC-1213, (um das especificações administrativas para o SNMP) ([COH2000]). A MIB padrão inclui vários objetos para medir e monitorar a atividade de *Internet Protocol* (IP), a atividade de TCP, a atividade de *User Datagram Protocol* (UDP), conexões de TCP, interfaces e a descrição do sistema geral. Ambos valores são associados a um nome oficial, como "sysUpTime", e um valor numérico de identificação como "1.3.6.1.2.1.1.3.0", que é o identificador de objeto para "sysUpTime".

2.6 CARACTERÍSTICAS EXTENSÍVEIS

O *Java Management Extensions* (JMX), permite ao usuário estender definições de novas MIB para o sistema. Estas definições normalmente são fornecidas por fabricantes de equipamento de rede, especialmente formatada usando a sintaxe ASN.1. (ASN.1 significa "*Abstract Syntax Notation One*", que é uma linguagem de declaração abstrata adotada pelo SNMP).

A MIB de um dispositivo SNMP é normalmente fixa. Ela é desenvolvida pelos fabricantes de equipamentos de rede (i.e. fabricante de roteadores, de hardware de computador, etc.) e não pode ser estendida ou modificada. Esta extensão de SNMP se refere estritamente ao software gerenciador SNMP ([SUN1999]).

2.7 OBJETOS MIB DO SNMP

Para usar o SNMP efetivamente, usuários precisam estar familiarizados com a Base de Informação do Gerenciador SNMP que define todos os valores de leitura e alteração que o SNMP é capaz. Para começar um administrador tem que conhecer a MIB do SNMP do equipamento ou da rede que estiver gerenciando.

A MIB é organizada em uma estrutura de árvore, semelhante a uma estrutura de diretório de arquivos em disco. O topo do nível SNMP começa com o diretório que contém quatro níveis principais: O nível “mgmt”, que normalmente contém os objetos padrões do SNMP utilizados por todos os dispositivos da rede; o nível “private” que contém os objetos especializados definidos pelos fabricantes de equipamentos de rede; os níveis “extended” e “directory” que são normalmente destituídos de quaisquer dados ou objetos significantes ([SUN1999]).

A estrutura da “árvore” descrita acima é uma parte integrante do padrão SNMP, porém a parte mais significativa da árvore são as “folhas”, objetos da árvore que fornecem os dados de gerência atuais do dispositivo.

2.7.1 OBJETOS DISCRETOS DA MIB

Objetos discretos do SNMP contêm uma parte precisa de dados do gerenciador. Estes objetos são freqüentemente distintos da tabela de itens, somando uma extensão “.0” (ponto-zero) para os nomes. (Se a extensão “.0” é omitida de um objeto SNMP, este sempre será implícito) ([SUN1999]).

Muitos objetos de SNMP são discretos, o operador necessita apenas saber o nome do objeto da MIB e nenhuma outra informação. Objetos discretos representam freqüentemente valores sumários de um dispositivo, particularmente útil por apresentar informação da rede com a finalidade de comparar desempenho de dispositivos de rede. Se a extensão (número de instância) do objeto não é especificada, pode ser assumido como “.0” (ponto-zero).

2.7.2 TABELAS DE OBJETOS DA MIB

Tabela de objetos contêm múltiplas partes de dados do gerenciador. Estes objetos são distintos dos itens “Discrete” adicionando uma extensão “.” (ponto) para os nomes deles que distinguem o valor particular que é referenciado.

A extensão “.” (ponto) se refere ao número da “instância” de um objeto do SNMP. No caso do objeto “Discrete”, este número de instância será zero. No caso da “Tabela” de objetos, este número de instância será o índice na tabela do SNMP.

Tabelas de SNMP são tipos especiais de objetos de SNMP que permitem aglutinar de forma estruturada um conjunto de informações. Por exemplo, SNMP define o objeto “ifDescr” como um objeto padrão do SNMP que indica a descrição de texto de cada interface apoiada por um dispositivo particular. Considerando que podem ser configurados dispositivos de rede com mais de uma interface, este objeto só poderia ser representado como um *array* ([SUN1999]).

Por convenção, objetos SNMP se agrupam sempre em um diretório de “Entrada”, dentro de um objeto com uma “Tabela” (o objeto “ifDescr” objeto descrito acima, reside no diretório “ifEntry” que é armazenado no diretório “ifTable”). Há várias regras para desenvolver objetos SNMP, como segue ([SUN1999]):

a) ao criar no diretório de “Entrada” de uma tabela tem que conter o mesmo número de elementos como outros objetos no mesmo diretório, onde o número de uma instância é o mesmo de todas as entradas. Sempre são considerados objetos da tabela como ordens paralelas de dados;

b) quando criando um novo objeto de “Entrada”, o SNMP requer que um valor tenha sido associado com cada entrada da tabela em uma única mensagem de SNMP (único PDU). Isto significa que, para criar uma fila em uma tabela (pelo comando “set” no SNMP), um valor deve ser especificado para cada elemento na fila;

c) se uma fila da tabela pode ser apagada, o SNMP requer pelo menos que para isso um objeto na entrada tenha um elemento de controle, que é documentado para executar a exclusão da tabela. (Isto só se aplica se uma fila pode ser excluída, que necessariamente não é requerido de uma tabela do SNMP).

2.7.3 TIPOS DE OBJETOS DE UMA MIB

Segundo [SUN1999], os objetos da MIB têm um tipo específico de valores. O SNMP define vários tipos primitivos, como se segue.

2.7.3.1 Tipo de Objetos de Texto

Define-se o tipo “DisplayString” que pode conter informação textual arbitrária (normalmente para um máximo de 256 caracteres). O texto deve conter somente caracteres de impressão. Exemplos deste tipo de objeto incluem valores “sysDescr” e “ifDescr”. Este tipo de objeto é bastante utilizado nas MIB.

2.7.3.2 Tipo de Objetos Contadores

Define-se o tipo “Counter” que é um valor numérico que só pode aumentar. Este é o tipo mais comum de objeto de SNMP na MIB padrão e inclui objetos como “ipInReceives”, “ipOutRequests”, e “snmpInPkts.” Contadores ultrapassam o valor de máximo da variável, mas nunca podem ser negativos.

2.7.3.3 Tipo de Objetos de Medida

Define-se o tipo “Gauge” que é um valor numérico que pode aumentar ou pode diminuir. Este tipo é encontrado em apenas algumas localizações dentro da MIB padrão. Exemplos deste tipo de objeto incluem o objeto “tcpCurrEstab”.

2.7.3.4 Tipo de Objetos Inteiros

Define-se o tipo “Integer” que pode conter valores positivos ou negativos. Este valor normalmente é fornecido por valores de tipo “Counter” ou “Gauge”, mas às vezes é expresso em MIB de equipamentos de fabricantes.

2.7.3.5 Tipo de Objetos Enumerados

Define-se um tipo “Enumerated Value” que associa um campo textual com um valor numérico. Este tipo é bastante comum na MIB padrão e inclui objetos como “ifAdminStatus”, cujo valores enumerados, são “up(1)”, “down(2)”, e “testing(3).” Geralmente os valores enumerados são formatados da seguinte forma “nome(número)”.

2.7.3.6 Tipo de Objetos Tempo

Define-se um tipo “TimeTicks” que representa um tempo decorrido. Este tempo sempre tem uma resolução de um centésimo de segundo, até mesmo quando esta resolução não é usada. Administradores de rede freqüentemente formatam este valor de tempo como “HH:MM:SS:cc” para exibição. Por exemplo, o valor “sysUpTime” indica o tempo decorrido desde que um dispositivo foi reiniciado.

2.7.3.7 Tipo de Objetos Objeto

Define-se um tipo “Object” que pode conter o identificador para outro objeto SNMP. Se o objeto nomeado é compilado na MIB, o nome geralmente é apresentado com o mesmo nome no objeto SNMP. Um exemplo deste tipo de objeto é a variável “sysObjectID” da MIB.

2.7.3.8 Tipo de Objetos Endereço IP

Define-se um tipo “IP address”, que contém o Endereço IP de um dispositivo de rede. Este tipo de objeto é exibido freqüentemente como um endereço de IP convencional. Exemplos deste tipo de objeto incluem “ipRouteDest”, “ipRouteNextHop” e “ipRouteMask”.

2.7.3.9 Tipo de Objetos Endereço Físico

Define-se um tipo “Physical address”, que contém o endereço físico de um dispositivo de rede. Gerentes exibem freqüentemente este tipo de objeto como uma sucessão de valores hexadecimal, precedidos pela palavra-chave “hex:”, e separados através de dois pontos. Exemplos deste tipo de objeto incluem o objeto “ifPhysAddress”.

2.7.3.10 Tipo de Objetos String

Define-se um tipo “String”, que contém uma cadeia de caracteres arbitrários. Quando a cadeia de caracteres contém só caracteres ASCII, os gerentes exibem este valor como uma string de texto. Caso contrário, gerentes exibem este tipo como uma sucessão de valores hexadecimal, precedidos pela palavra-chave “hex:”, e separados através de dois

pontos. Este tipo de valor é incomum, mas ocasionalmente encontrado em MIB proprietárias.

2.7.3.11 Tipo de Objetos Tabela

O tipo “Table” é um objeto que contém entradas da tabela. Este tipo de objeto sempre é um nome intermediário que contém um diretório de “Entrada” e que contém vários objetos da tabela.

2.7.3.12 Tipo de Objetos Auxiliares

O tipo “Branch” é um objeto que contém tipos adicionais, tabelas, ou qualquer tipo de objeto listado acima.

2.8 ACESSO A VALORES DA MIB

Cada objeto SNMP é definido para ter um tipo de acesso “somente de leitura”, “leitura escrita” ou “apenas escrita”. Isso determina se o usuário pode ler o valor de objeto, pode ler e pode escrever o objeto (com um comando “set”) ou só escrever o objeto ([CHI1999]).

Antes que qualquer objeto possa ser lido ou possa ser escrito, o nome comunitário do SNMP deve ser conhecido. Estes nomes comunitários são configurados pelo administrador, e podem ser vistos como senhas necessárias para anexar dados ao SNMP. No sentido exato, nomes comunitários existem para permitir que partes da MIB no SNMP, e subconjuntos de objeto sejam referenciados. Como o termo “Comunitário” é requerido, espera-se que o verdadeiro propósito destes valores sejam identificar comunitariamente entre os objetos SNMP configurados. Porém, é prática comum fazer estes nomes comunitários limitarem o acesso da capacidade do SNMP para usuários sem permissão.

2.9 COMPILANDO OBJETOS DA MIB

Um dos componentes principais de qualquer Gerenciador de SNMP é um “Compilador de MIB” que cria novos objetos da MIB para serem adicionados ao sistema gerenciador.

Quando uma MIB é compilada em um gerente SNMP, são criados novos objetos no gerente que são suportados por agentes da rede. O agente não é afetado pela compilação da MIB (visto que ele já possui seus próprios objetos). O ato de compilar a MIB permite ao gerente saber sobre os objetos especiais que auxiliam o agente, passando a ter acesso sobre estes objetos como parte da configuração do objeto.

Tipicamente, quando uma MIB é compilada no sistema, o gerente cria pastas ou diretórios que correspondem aos objetos. Estas pastas ou diretórios podem ser vistos com um “Navegador MIB” que é uma ferramenta tradicional de gerenciamento de SNMP, incluído em todos sistemas de gerência de rede. Estes objetos novos podem ser acessados ou modificados para afetar o desempenho de um agente remoto.

Os objetos da MIB são documentados em ASN.1 que é o idioma de declaração de tipo abstrato adotado pelo SNMP. O usuário obterá definições ASN.1 para uma nova parte de equipamento de rede ou um novo agente de SNMP, transferirá este arquivo de definições para o sistema gerenciador de rede ([SUN1999]).

2.10 O SNMP SOBRE A CAMADA DE TRANSPORTE

O protocolo SNMP foi desenvolvido para rodar sobre a camada de transporte, na camada de aplicação da pilha de protocolo TCP/IP. A maioria das implementações do SNMP utilizam o *User Datagram Protocol* (UDP) como protocolo de transporte. O UDP é um protocolo não-confiável, não garantindo a entrega, a ordem ou a proteção contra duplicação das mensagens ([ODA1998]).

O SNMP utiliza o UDP, pois foi desenvolvido para funcionar sobre um serviço de transporte sem conexão. A maior razão para isto é o desempenho. Utilizando um serviço orientado à conexão, como o TCP, a execução de cada operação necessitaria de uma prévia conexão, gerando um *overhead* significativo, o que é inaceitável na atividade de gerência

onde as informações devem ser trocadas entre as entidades da forma mais rápida possível, sem afetar o desempenho da transmissão dos demais dados.

Segmentos UDP são transmitidos em datagramas IP. O cabeçalho UDP inclui os campos de origem e destino, um *checksum* opcional que protege o cabeçalho UDP e os dados de usuário (em caso do *checksum* ser violado, a *Protocol Description Unit* (PDU) é descartada).

Duas portas UDP são associadas às operações SNMP. As operações Get, GetNext, GetBulk, Inform e Set utilizam a porta 161. Por ser acionada em casos de exceção, a operação Trap têm reservada para si a porta 162 ([TAN1996]).

Como o UDP é um protocolo não-confiável, é possível que mensagens SNMP sejam perdidas. O SNMP por si só não fornece garantia sobre a entrega das mensagens. As ações a serem tomadas quando da perda de uma mensagem SNMP não são abordadas pelo padrão. No entanto, algumas considerações podem ser feitas sobre a perda de mensagens SNMP.

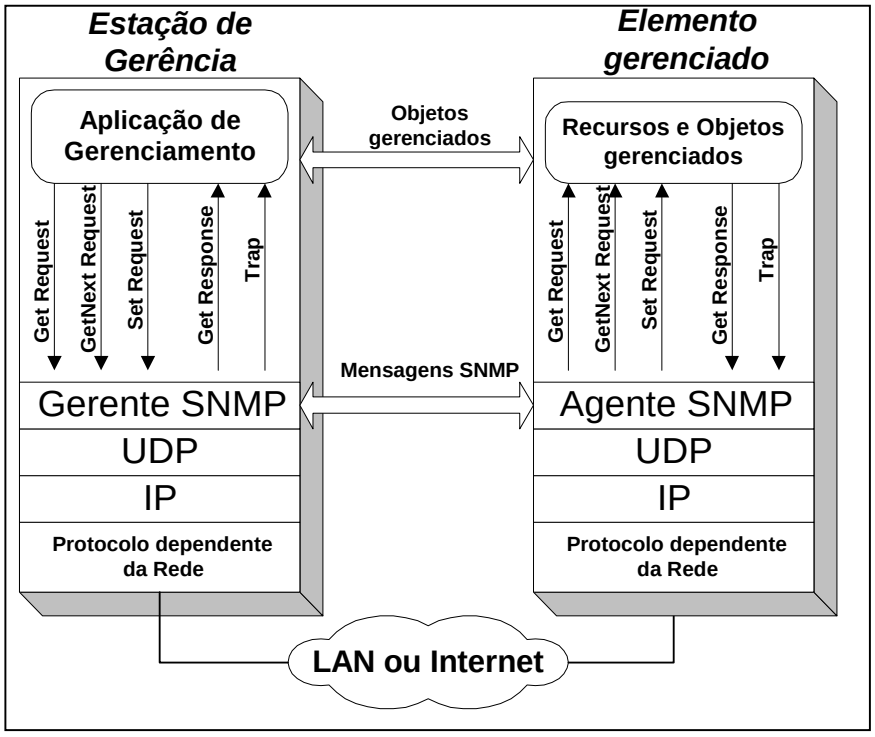
No caso de uma mensagem de Get, GetNext ou GetBulk, a estação de gerenciamento deve assumir que esta mensagem (ou o GetResponse correspondente) tenha sido perdida se não receber resposta num certo período de tempo. A estação de gerenciamento então pode repetir a mensagem uma ou mais vezes até que obtenha a resposta. Desde que um único *request-id* (que identifica a mensagem) seja utilizado para cada operação distinta, não haverá dificuldade em identificar mensagens duplicadas.

No caso de uma mensagem de Set, a recuperação deve provavelmente envolver o teste no objeto associado à operação através de uma Get sobre este mesmo objeto a fim de determinar se a operação Set foi ou não efetivada.

Segundo [COH2000], no SNMP o gerente executa o gerenciamento através da utilização do protocolo SNMP, que é implementado no topo das camadas UDP, IP e do protocolo dependente do tipo de rede (Ethernet, FDDI, X.25, por exemplo).

A figura 3 ilustra o contexto do protocolo SNMP na pilha de protocolo TCP/IP, utilizando o UDP como protocolo de conexão.

FIGURA 3 - PROTOCOLO SNMP SOBRE A CAMADA DE TRANSPORTE



O SNMP requer alguns serviços da camada de transporte para que suas mensagens sejam transmitidas entre as entidades de gerenciamento. Isso independe da implementação da camada de transporte.

2.10.1. SERVIÇOS EXIGIDOS PELO SNMP

Rodando na camada de aplicação da arquitetura TCP/IP, como mostrado na Figura 3, o SNMP encontra-se acima das camadas de Transporte e Rede. Para que o SNMP possa funcionar adequadamente, estas camadas (Transporte e Rede) devem fornecer ao menos cinco serviços que são de vital importância à transmissão das mensagens SNMP através da rede ([COH2000]):

a) roteamento -A camada de Rede é quem fornece as funções de roteamento, as quais aperfeiçoam toda a utilidade do gerenciamento da rede, dando a possibilidade de “re-rotear” os pacotes em torno de áreas da rede com falhas. Isso permite que o gerenciamento da rede continue operando mesmo após falha em alguma(s) máquina(s) na rede;

b) independência do meio - Permite ao SNMP gerenciar diferentes tipos de elementos da rede, de forma independente. Caso o SNMP fosse construído sobre a camada de enlace, estaria preso a esta implementação de enlace específica, desse modo o SNMP não poderia gerenciar um outro dispositivo que implementasse outro protocolo de enlace, o que limitaria o gerenciamento da rede;

c) fragmentação e Remontagem - Este serviço está relacionado com a independência do meio. A mensagem SNMP pode ser fragmentada em pacotes, transmitida pelo meio e remontada no destino. O protocolo IP permite que os pacotes SNMP trafeguem pelo meio com diferentes tamanhos. A Fragmentação e a Remontagem reduzem a robustez geral do gerenciamento da rede, pois se qualquer fragmento for perdido pelo caminho, toda a operação irá falhar;

d) *checksum* ponto-a-ponto - O *checksum* ponto-a-ponto é um serviço de checagem de erros fornecido pela camada de transporte que permite a uma entidade conferir a integridade dos dados recebidos através da rede, aumentando a confiabilidade da transmissão;

e) multiplexação/demultiplexação - Os serviços de Multiplexação e Demultiplexação são fornecidos pelo protocolo de transporte. Estes serviços facilitam em muito os possíveis relacionamentos de gerenciamento com o SNMP, permitindo que várias aplicações SNMP possam utilizar serviços da camada de transporte.

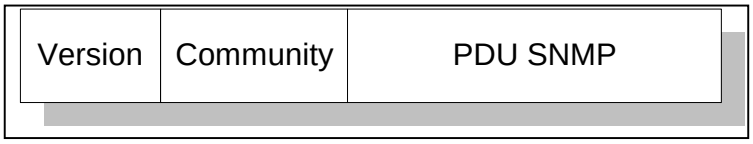
2.10.2. FORMATO DA MENSAGEM SNMP

No SNMP, as informações são trocadas entre os gerentes e agentes na forma de mensagens. Cada mensagem possui duas partes, um cabeçalho e uma *Protocol Data Unit* (PDU). O cabeçalho inclui um número de versão (*version*) que indica a versão do SNMP e um nome de comunidade (*community*). O nome de comunidade possui duas funções. Primeiro, o nome de comunidade define um dispositivo de acesso para um grupo de NMS ([ODA1998]).

Segundo aqueles dispositivos cujo nome de comunidade são desconhecidos são excluídos de operações SNMP. Os gerentes também podem utilizar o nome de comunidade

como uma forma de autenticação. O campo PDU pode conter qualquer um dos cinco tipos de PDU utilizados pelo SNMP (SNMP PDU). Esta estrutura pode ser visualizada na Figura 4.

FIGURA 4 - FORMATO DA MENSAGEM SNMP



Fonte: [ODA1998]

As PDU GetRequest PDU, GetNextRequest PDU e SetRequest PDU são definidas no SNMP com o mesmo formato da GetResponse PDU, com os campos *error-status* e *error-index* com valor zero. Essas PDU possuem os seguintes campos ([ODA1998]):

a) *version* - Indica a versão do protocolo SNMP utilizado;

b) *community* - O nome de comunidade atua como uma senha para autenticar a mensagem SNMP;

c) SNMP PDU - É a unidade de dados de protocolo (*Protocol Data Unit* - PDU) utilizada pelo SNMP; contém os dados referentes a operação desejada (Get, GetNext etc). Sua forma mais geral inclui os seguintes campos:

- PDU *type* - indica o tipo de PDU, neste caso pode ser uma GetRequest PDU, uma GetNextRequest PDU, uma SetRequest PDU ou uma GetResponse PDU;

- *request*-id - Usado para identificar o request; essa mesma identificação será utilizada na resposta a esta mensagem;

- *error-status* - É um sinalizador utilizado para indicar que uma situação inesperada ou erro ocorreu no processamento da mensagem; seus valores possíveis são:

i. noError (0) - Indica que não houve qualquer tipo de erro no processamento da mensagem;

ii. tooBig (1) - Se o tamanho da mensagem GetResponse gerada exceder uma limitação local, então o campo error-status assume este valor. Neste caso, o valor do campo error-index deve ser zero;

iii. noSuchName (2) - Indica que o valor de alguma variável da lista de variáveis (*variablebindings*) não está disponível na MIB em questão. Neste caso, o valor do campo error-index indica em qual variável da lista ocorreu o problema;

iv. badValue (3) - Significa que o valor para uma determinada variável da lista não está de acordo com a linguagem ASN.1; ou que o tipo, tamanho ou valor é inconsistente. Assim como no caso anterior, o valor do campo error-index indica em qual variável da lista ocorreu o problema;

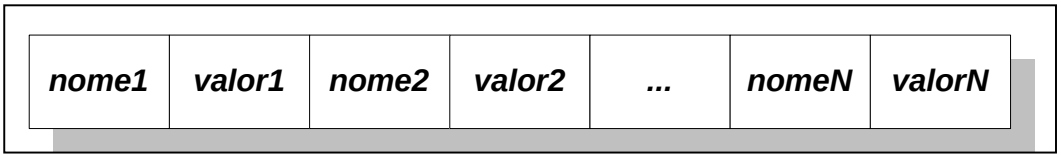
v. readOnly (4) - Indica que determinada variável da lista só pode ser lida e não alterada. Neste caso, esse tipo de código de erro só é retornado após uma operação de Set sobre alguma variável. O valor do campo error-index indica em qual variável da lista ocorreu o problema;

vi. genErr (5) - Se o valor de uma determinada variável da lista não puder ser identificado ou alterado por outras razões que não as já citadas, então o campo error-status assume o valor genErr ou simplesmente 5. Neste caso, o valor do campo error-index indica em qual variável da lista ocorreu o problema;

d) *error-index* - Quando o campo *error-status* é diferente de zero, este campo fornece uma informação adicional indicando qual variável da lista de variáveis causou a exceção (ou erro);

e) *variablebindings* - Uma lista de nomes de variáveis e seus respectivos valores (em alguns casos, como no GetRequest PDU, esses valores são nulos). A estrutura deste campo é mostrada na Figura 5.

FIGURA 5 - ESTRUTURA DO CAMPO VARIABLEBINDINGS



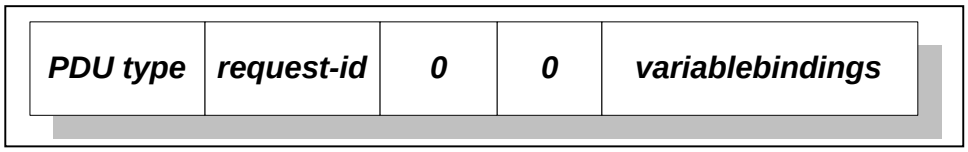
Fonte: [ODA1998]

Por se tratar de um caso particular de mensagem, indicando uma situação inesperada, a Trap PDU possui uma estrutura diferente das demais PDU utilizadas pelo SNMP.

2.10.2.1. GetRequest PDU

A GetRequest PDU é utilizada pela estação de gerência SNMP para executar a operação Get, requisitando ao agente SNMP a leitura de variáveis da MIB da máquina onde ele se encontra. A entidade emissora inclui os seguintes campos nesta PDU:

FIGURA 6 - FORMATO DA GETREQUEST PDU



Fonte: [ODA1998]

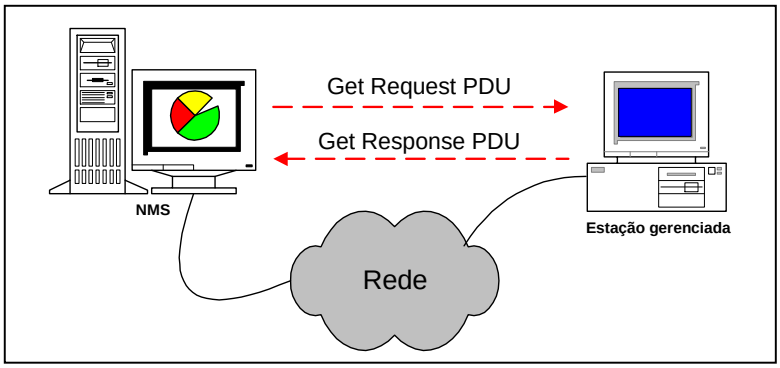
a) PDU *type* - Este campo indica que se trata de um GetRequest PDU;

b) *request-id* - A entidade emissora associa a este campo um número que identifica unicamente este *request*. O *request-id* possibilita à aplicação SNMP relacionar uma mensagem de response recebida a uma mensagem de *request* enviada, pois o response trará o mesmo valor do *request-id*. Além disso, o *request-id* permite ao agente identificar PDU duplicadas geradas em função de falhas no serviço de transporte;

c) *variablebindings* - Uma lista de instâncias de objetos (variáveis) dos quais são requisitados os valores.

Percebe-se na Figura 6 que na GetRequest PDU os campos *error-status* e *error-index* possuem o valor zero.

FIGURA 7 - PASSAGEM DA GETREQUEST PDU



Fonte: [COH2000]

A entidade SNMP receptora responde a uma GetRequest PDU com uma GetResponse PDU contendo o mesmo valor do *request-id*, como está mostrando na figura 7. Caso a entidade receptora da GetRequest PDU possa fornecer o valor de todas as variáveis listadas no campo *variablebindings*, então é montada uma GetResponse PDU contendo este campo acrescentado do respectivo valor de cada variável. Se o valor de apenas uma variável não puder ser informado, então nenhum dos valores das outras variáveis são retornados. As condições de erro que podem acontecer são ([ODA1998]):

a) para uma variável referenciada no campo *variablebindings* que não seja encontrada uma correspondente na MIB em questão; ou a variável referenciada pode ser um tipo agregado e desta forma não há uma valor agregado a esta instância. Neste caso, a entidade receptora retorna uma GetResponse PDU com o campo *error-status* indicando *noSuchName* e com o valor do campo error-index indicando que variável da lista de variáveis (*variablebindings*) causou o problema, ou seja, se a terceira variável da lista de variáveis não estiver disponível para a operação Get, então o campo *error-index* da GetResponse PDU possui o valor 3;

b) a entidade receptora pode fornecer o valor de todas as variáveis da lista, mas o tamanho resultante da GetResponse PDU pode exceder a limitação local. Neste caso, a entidade receptora envia à entidade emissora uma GetResponse PDU com o campo *error-status* indicando *tooBig*;

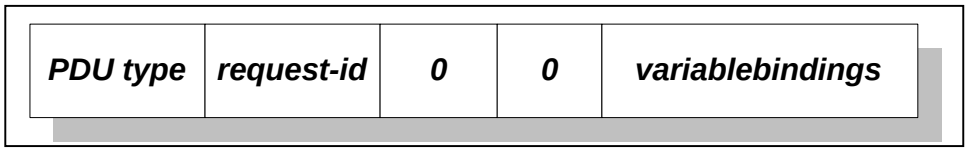
c) a entidade receptora pode não fornecer o valor de uma determinada variável por alguma outra razão. Neste caso, a entidade receptora retorna uma GetResponse PDU com o campo *error-status* indicando *genErr* e com o valor do campo *error-index* indicando que variável da lista de variáveis causou o erro.

Se nenhum dos casos anterior se aplica, então a entidade receptora envia para a entidade da qual originou a GetRequest PDU uma GetResponse PDU com o campo *error-status* indicando *noError* e com o valor do campo *error-index* com valor zero.

2.10.2.2. GetNextRequest PDU

A GetNextRequest PDU é utilizada pela estação de gerência SNMP para executar a operação GetNext, requisitando ao agente SNMP a leitura da próxima variável na ordem lexicográfica da MIB da máquina onde ele se encontra. A GetNextRequest PDU é praticamente idêntica à GetRequest PDU, possuindo o mesmo mecanismo de troca e a mesma estrutura, como mostrado na Figura 8.

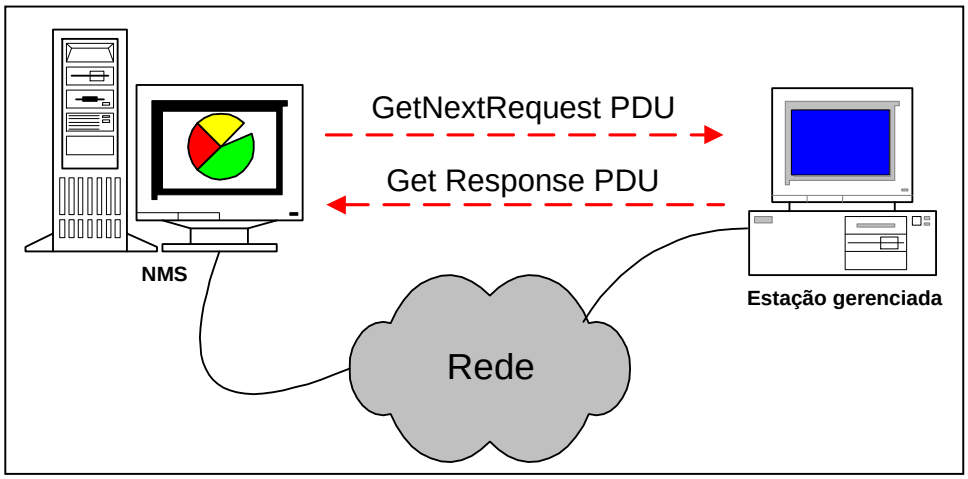
FIGURA 8 - FORMATO DA GETNEXTREQUEST PDU.



Fonte: [ODA1998]

A única diferença é o seguinte: numa GetRequest PDU cada variável descrita na lista de variáveis (*variablebindings*) deve ter seu valor retornado. Na GetNextRequest PDU, para cada variável, é retornado o valor da instância que é a próxima na ordem segundo a definição da MIB. Assim como a GetRequest PDU, a GetNextRequest PDU é chamada ou todos os valores de todas as variáveis da lista (*variablebindings*) são retornados ou nenhum deles é, conforme mostrado na figura 9.

FIGURA 9 - PASSAGEM DA GETNEXTREQUEST PDU



Fonte: [COH2000]

Apesar de parecer pouco significante, esta diferença entre a GetRequest PDU e a GetNextRequest PDU possui grandes implicações. A fim de entender essas implicações vamos analisar algumas possibilidades:

- suponha-se que uma estação de gerência (NMS) deseje recuperar os valores de todas as variáveis simples do grupo *UDP* de uma determinada MIB. A NMS então envia ao gerente responsável por gerenciar esta MIB uma GetRequest PDU na seguinte forma:

GetRequest (*udpInDatagrams.0*, *udpNoPorts.0*, *udpInErrors.0*,

udpOutDatagrams.0)

Se a MIB em questão suportar todas as variáveis relacionadas na lista, então uma GetResponse PDU deverá ser retornada com os respectivos valores das variáveis referenciadas na GetRequest PDU:

GetResponse ((*udpIndatagrams.0* = 100),(*udpNoPorts.0* = 1),

(*udpInErrors.0* = 2) ,(*udpOutDatagrams.0* = 200)) ,

Onde 100, 1, 2 e 200 são os valores das respectivas variáveis requisitadas no GetRequest. Entretanto, se não for possível retornar o valor de qualquer das variáveis, então

a GetResponse PDU é descartada e outra GetResponse PDU é construída e enviada ao gerente da NMS com o código de erro indicando *noSuchName*. Para assegurar que todos os valores disponíveis sejam informados, utilizando-se a GetRequest PDU, a estação de gerenciamento teria que enviar quatro destas PDU separadas, cada uma requisitando o valor de uma variável específica. Agora consideremos a utilização da GetNextRequest PDU na recuperação dos valores das variáveis:

GetNextRequest (*udpIndatagrams.0*, *udpNoPorts.0*, *udpInErrors.0*,

udpOutDatagrams.0)

Neste caso, o agente irá retornar o valor da próxima instância do objeto (variável) para cada identificador da lista de variáveis. Suponha agora que todas as variáveis da lista sejam suportadas pela MIB em questão. O identificador (Object Identifier) para a variável *udpInErrors*, por exemplo, é 1.3.6.1.2.1.7.3. A próxima instância para esta mesma variável na ordem da MIB é identificada por udpInErrors.0 ou 1.3.6.1.2.1.7.3.0. Da mesma forma, a próxima instância de *udpNoPorts* (1.3.6.1.2.1.7.2) é *udpNoPorts.0* (1.3.6.1.2.1.7.2.0), e assim para as outras variáveis. Assim, se todos os valores estão disponíveis, o agente irá retornar uma GetResponse PDU na seguinte forma:

GetResponse ((*udpIndatagrams.0* = 100),(*udpNoPorts.0* = 1),

(*udpInErrors.0* = 2) ,(*udpOutDatagrams.0* = 200)),

A qual é a mesma retornada com a utilização da GetRequest PDU. Agora, suponhamos que o valor de *udpNoPorts* não esteja disponível e que a mesma GetNextRequest PDU seja utilizada. Neste caso, a resposta do agente viria na seguinte forma:

GetResponse ((*udpIndatagrams.0* = 100),(*udpInErrors.0* = 2),

(*udpInErrors.0* = 2) ,(*udpOutDatagrams.0* = 200))

Neste caso, o identificador da variável *udpNoPorts.0* não é um identificador válido para a MIB em questão. Portanto, o agente retorna o valor da próxima instância na ordem, a qual neste caso é *udpInErrors.0*. Podemos perceber que a utilização da

GetNextRequest PDU nos permite um melhor desempenho na recuperação de um conjunto de valores de variáveis quando há a possibilidade de algum destes valores não estarem disponíveis.

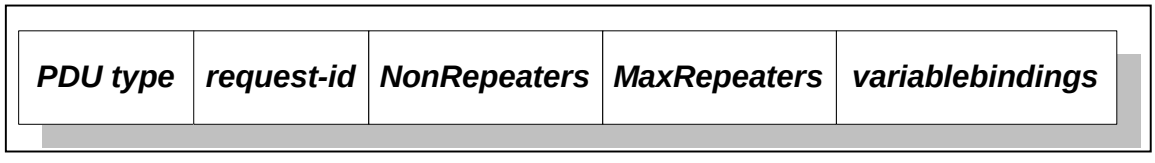
A utilização da operação GetNext é particularmente interessante na recuperação de uma seqüência de instâncias de variáveis. No entanto, esta operação ainda possui o inconveniente de se ter que enviar sucessivas GetNextRequest PDU, pois este tipo de PDU não permite recuperar um número grande de instâncias.

2.10.2.3. GetBulkRequest PDU

A GetBulk PDU é utilizada por um gerente SNMP para executar a operação GetBulk, uma das melhores características adicionadas ao SNMP padrão, que permite que um gerente SNMP resgate uma grande quantidade de informações de gerenciamento de uma determinada MIB. A GetBulkRequest PDU permite a um gerente SNMP requisitar que a resposta (GetResponse PDU) seja tão grande quanto possível dentro da restrição de tamanho.

A GetBulkRequest PDU segue o mesmo princípio da GetNextRequest PDU, recuperando sempre a próxima instância, na ordenação da MIB, de cada variável da lista de variáveis (*variablebindings*). A diferença é que, com a GetBulkRequest PDU, é possível especificar quantas próximas instâncias na ordem devem ser retornadas na mesma GetResponse PDU.

FIGURA 10 - FORMATO DA GETBULKREQUEST PDU



Fonte: [ODA1998]

Como mostrado na Figura 10, a GetBulkRequest PDU utiliza um formato diferente das demais PDU, com os seguintes campos:

a) PDU *type*, *request-id* e *variablesbindings* - Estes campos possuem na GetBulkRequest PDU a mesma função que possuem nas demais PDU (GetRequest PDU, GetNextRequest PDU, SetRequest PDU, Response PDU, InformRequest PDU e Trap PDU);

b) *Nonrepeaters* - Especifica o total das primeiras variáveis da lista de variáveis (*variablebindings*) das quais apenas a próxima instância na ordem deve ser retornada;

c) *Max-repetitions* - Especifica o número de sucessores, também na ordem, que devem ser retornados para o resto (*total_de_variáveis_da_lista* - *Nonrepeaters*) das variáveis da lista de variáveis.

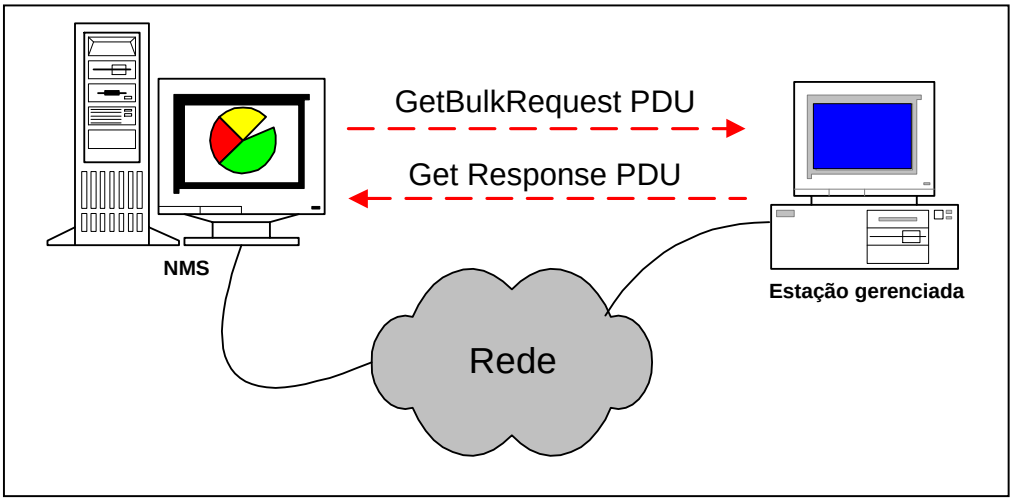
Após executar a operação GetBulk o agente deve construir uma GetResponse PDU, contendo as instâncias das variáveis requisitadas e seus respectivos valores. No entanto, pode acontecer de a GetResponse PDU não retornar todos os pares (instância, valor) requisitados na lista (podendo até retornar nenhum dos pares). Isso pode ocorrer por três razões:

a) se o tamanho da mensagem que encapsula a GetResponse PDU exceder a limitação local de tamanho ou exceder o tamanho máximo de uma mensagem de *request* (definido pelo protocolo), então a GetResponse é gerada com um número menor de pares no campo *variablebindings*. Ou seja, a GetResponse PDU é gerada inserindo-se os pares até que o tamanho máximo seja alcançado;

b) se todos os pares subseqüentes descritos na lista de variáveis possuírem o valor *endOfMibView* (significando que não existe um sucessor da variável na ordem da MIB), o campo *variablebindings* pode ser truncado neste ponto, retornando nenhum dos pares requisitados;

c) se o processamento de uma GetBulkRequest PDU necessitar de uma grande quantidade de tempo de processamento pelo agente, então o agente pode particionar o processamento, retornando resultados parciais.

FIGURA 11- PASSAGEM DA GETBULKREQUEST PDU



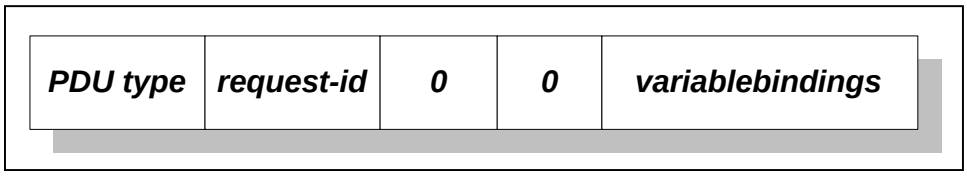
Fonte: [COH2000]

Se o processamento de alguma variável falhar, por qualquer motivo que não seja *endOfMibView*, então nenhum dos pares requisitados é retornado. Em vez disso, o agente envia ao gerente uma GetResponse PDU com o campo *error-status* indicando *genErr* e o valor do campo *error-index* apontando para a variável da lista que gerou o problema. O código de erro *tooBig* nunca é retornado por um agente SNMP pois, como já foi dito, quando a mensagem de resposta é muito grande nem todas as variáveis requisitadas são retornadas na mesma PDU, diminuindo seu tamanho.

2.10.2.4. SetRequest PDU

A SetRequest PDU é utilizada por um gerente SNMP na realização da operação Set, indicando ao agente que o valor de determinada(s) variável(is) deve(m) ser alterado(s). Este tipo de PDU possui o mesmo formato da GetRequest PDU. No entanto, é utilizada para escrever uma variável na MIB, ao invés de lê-la. Diferente da GetRequest PDU e da GetNextRequest PDU, cujo valor das variáveis referenciadas na lista (*variablebindings*) têm valor zero, neste tipo de PDU o valor de cada instância da lista representa o valor a ser modificado na MIB em questão.

FIGURA 12 - ESTRUTURA DA SETREQUEST PDU



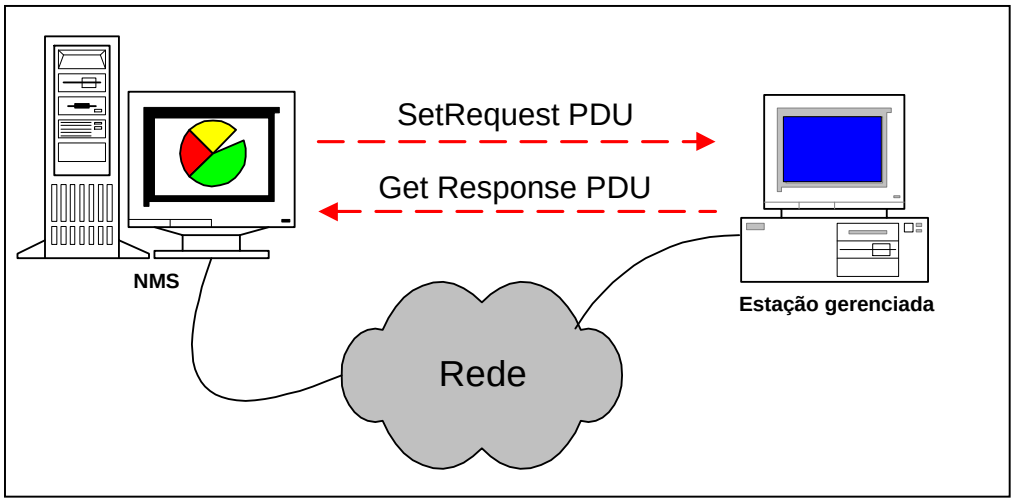
Fonte: [ODA1998]

Quando um agente recebe uma SetRequest PDU ele efetua, quando possível, a operação Set, modificando o valor das variáveis correspondentes, e retorna ao gerente da NMS emissora uma GetResponse PDU contendo o mesmo *request-id*. Assim como a operação Get, a operação Set é solicitada ou todas as variáveis são modificadas/atualizadas ou nenhuma delas o é.

Se um agente que recebeu uma SetRequest PDU puder alterar o valor de todas as variáveis especificadas na lista (*variablebindings*), então ele constrói uma GetResponse PDU incluindo a mesma lista de variáveis, com seus respectivos valores atualizados, enviando-a ao gerente da NMS emissora. Se ao menos um dos valores não puder ser modificado, então nenhum valor é retornado e nem escrito na MIB em questão. As mesmas condições de erro usadas na GetRequest PDU podem ser retornadas (*noSuchName*, *tooBig*, *genErr*).

Outro erro que pode ocorrer durante a operação Set é o *badValue*. Esta condição de erro ocorre quando algum dos pares (variável, valor) contidos na SetRequest PDU estiver inconsistente no que diz respeito ao tipo, tamanho ou valor. Quando algum erro ocorre o agente descarta a GetResponse PDU e constrói uma nova com o campo *status-error* indicando o tipo de erro que ocorreu durante a operação e com o campo *index-error* apontando para a variável da lista que ocasionou o problema.

FIGURA 13 - PASSAGEM DA SETREQUEST PDU



Fonte: [COH2000]

Um curioso código de erro que pode ser retornado por um agente durante a execução da operação Set é o *readOnly*, que praticamente não é implementado. O problema é que o *readOnly* significa que uma determinada variável na MIB em questão cujo valor deveria ser modificado possui *status* de apenas leitura (*read only*), não podendo portanto ser alterado. No entanto, quando um gerente pede a um agente para modificar uma variável que seja do tipo *read only*, o agente não podendo executar esta operação, constrói e envia uma GetResponse PDU com o código de erro indicando *noSuchName* e não *readOnly*.

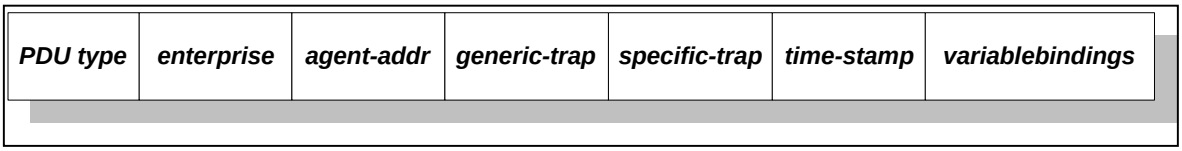
O grande problema desta confusão é o fato de que quando um gerente recebe uma GetResponse PDU de uma operação de Set com o código de erro indicando *noSuchName*, ele não tem como saber se a instância da variável realmente não foi encontrada na MIB (o que justifica o *noSuchName*) ou se o problema, na verdade, é que a instância da variável possui *status read only* na MIB em questão.

Neste caso, para ter certeza da origem do problema, o gerente teria que enviar uma GetRequest PDU para a variável que causou o problema a fim de verificar se realmente a variável não existe na MIB ou se ela apenas possui o *status* de *read only*.

2.10.2.5. Trap PDU

A Trap PDU é utilizada pelo agente SNMP na execução da operação Trap, notificando de forma assíncrona eventos extraordinários que tenham ocorrido no objeto gerenciado. A Trap PDU possui uma estrutura diferente das outras PDU como mostrada na Figura 14.

FIGURA 14 - ESTRUTURA DA TRAP PDU - SNMP



Fonte: [ODA1998]

a) *PDU type* - indica o tipo de PDU, neste caso indica que trata-se de uma Trap PDU;

b) *enterprise* - indica o tipo do objeto que gerou o Trap; este campo é preenchido a partir do *sysObjectID*;

c) *agent-addr* - endereço do objeto que gerou o Trap;

d) *generic-trap* - indica o tipo do Trap; os valores possíveis para este campo são:

- *coldStart (0)* - Significa que a entidade emissora do Trap está sendo inicializada, de tal modo que a configuração do agente ou da entidade de protocolo podem ser alteradas;

- *warmStart (1)* - Significa que a entidade emissora do Trap está sendo reinicializada, de tal modo que nenhuma configuração do agente nem da entidade é alterada;

- *linkDown (2)* - Significa que a entidade emissora do Trap reconheceu que o estado de alguma de suas linhas de comunicação representadas na configuração do agente está disponível;

- *linkUp (3)* - Significa que a entidade emissora do Trap reconheceu que o estado de uma de suas linhas de comunicação representadas na configuração do agente está disponível;

- *authentication-Failure (4)* - Significa que a entidade emissora do Trap foi o destinatário de uma mensagem de protocolo que não estava corretamente autenticada. Quando da implementação do gerente, esta propriedade de enviar este tipo de Trap deve ser configurável, podendo em determinados casos ser desativada;

- *egpNeighborLoss (5)* - Significa que um vizinho EGP (*Exterior Gateway Protocol*) da entidade emissora do Trap foi o mesmo EGP de quem a entidade emissora marcou a queda e cujo relacionamento foi perdido;

- *enterprise-Specific (6)* - Significa que a entidade emissora do Trap reconhece que ocorreu algum evento avançado específico ocorreu;

e) *specific-trap* - possui o código específico do Trap;

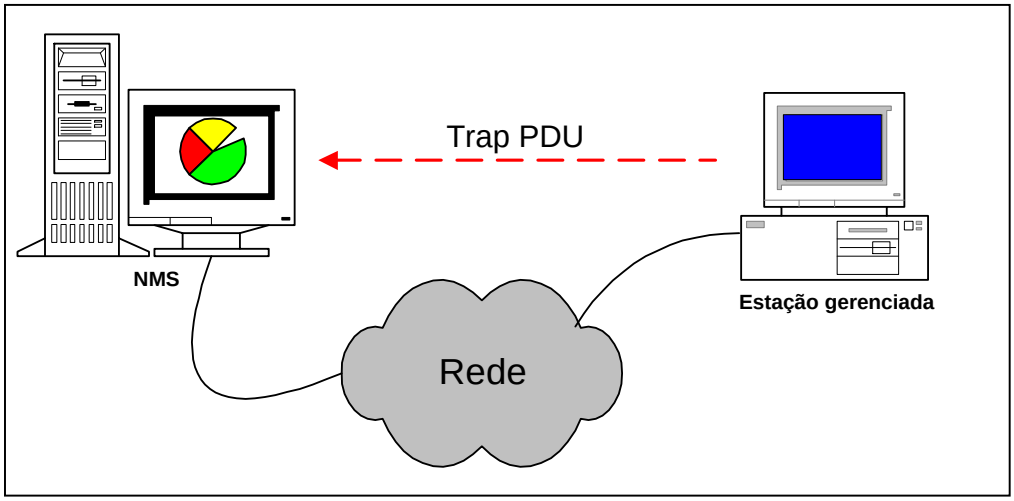
f) *time-stamp* - armazena o tempo decorrido entre a última reinicialização da entidade que gerou o Trap e a geração do Trap; contém o valor de *sysUpTime*;

g) *variablebindings* - Uma lista de nomes de variáveis e seus respectivos valores.

Toda vez que uma máquina da rede é inicializada, o agente responsável deve enviar ao gerente um Trap do tipo *coldStart (0)*, indicando que a máquina "entrou" na rede.

Outra característica que difere a Trap PDU das outras PDU é o fato de que ela não necessita de uma resposta, como mostra a Figura 15.

FIGURA 15 - PASSAGEM DA TRAP PDU

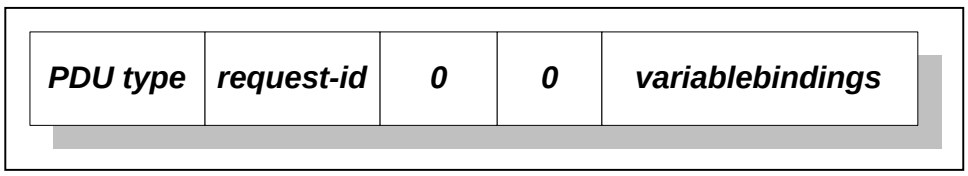


Fonte: [COH2000]

2.10.2.5.1. Trap PDU-v2

A Trap PDU-v2 segue as mesmas regras da Trap PDU utilizada na versão básica do SNMP, mas com um formato diferente. Conforme a Figura 16, este formato é o mesmo de todas as outras PDU utilizadas no SNMP, exceto a GetBulkRequest PDU, o que facilita o processamento das mensagens pelo agente SNMP.

FIGURA 16 - ESTRUTURA DA TRAP PDU - SNMPV2.



Fonte: [ODA1998]

2.10.2.6. InformRequest PDU

A InformRequest PDU é enviada por um agente SNMP a outro agente SNMP, fornecendo informações de gerenciamento ao último agente. Esta PDU inclui o campo *variablebindings* com os mesmos elementos da Trap PDU-v2, como mostra a Figura 17.

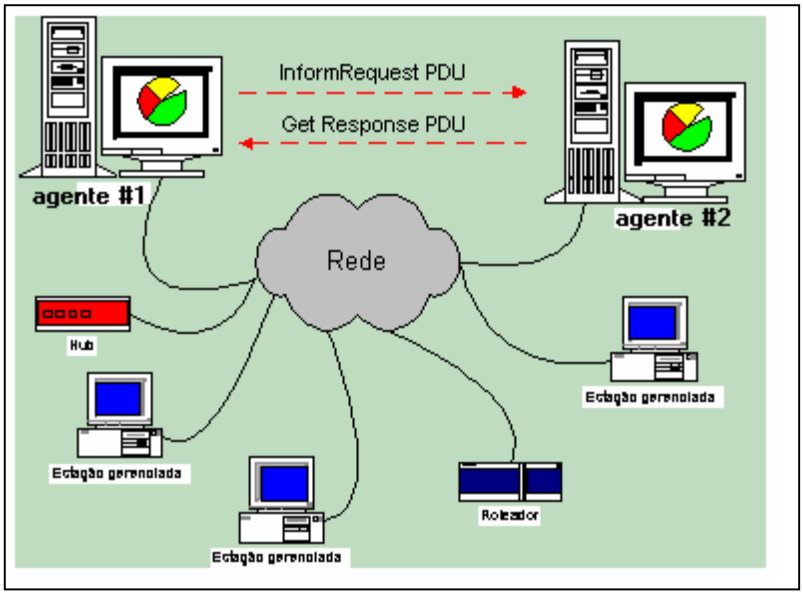
FIGURA 17 - ESTRUTURA DA INFORMREQUEST PDU

PDU type	request-id	0	0	variablebindings

Fonte: [ODA1998]

Ao receber uma InformRequest PDU, o agente determina o tamanho da mensagem de resposta que encapsula uma GetResponse PDU com os mesmos valores dos campos *request-id*, *error-status*, *error-index* e *variablebindings* da InformRequest PDU recebida. Se este tamanho exceder a limitação local de tamanho ou exceder o tamanho máximo da mensagem de resposta, então uma GetResponse PDU é construída com o código de erro (*error-status*) indicando *tooBig*, o índice de erro (*error-index*) com valor zero e com a lista vazia de variáveis (*variablebindings*).

FIGURA 18 - PASSAGEM DA INFORMREQUEST PDU



Fonte: [SUN1999]

Caso o tamanho da mensagem seja aceitável, seu conteúdo é passado para a aplicação destino e é construída e enviada para o agente origem uma GetResponse PDU com o código de erro indicando *noError* e o índice de erro com valor zero.

2.10.2.7. GetResponse PDU

O formato da GetResponse PDU é idêntico a GetRequest PDU a não ser pela indicação do tipo (PDU *Type*). Uma GetResponse PDU é gerada por uma entidade SNMP sempre que esta recebe uma GetRequest PDU, GetNextRequest PDU ou SetRequestPDU, como descrito anteriormente.

2.11 COMPONENTES ESSENCIAIS DO GERENCIADOR

Segundo [ODA1998], todos os gerentes SNMP contêm vários componentes básicos como segue.

2.11.1 FUNÇÕES DE ALARME

Os gerentes SNMP possuem a funcionalidade de monitorar os objetos da MIB e de responder com algum tipo de notificação quando estes mesmos são violados. Isto faz com que constantemente seja testada a integridade de uma rede. Inerente a esta função está a habilidade de determinar quais dispositivos estão respondendo e quais dispositivos não estão respondendo.

2.11.2 MONITORAMENTO DE FUNÇÕES

Os gerentes SNMP podem monitorar continuamente um valor SNMP e verificar este valor a intervalos periódicos para ver a tendência de uma rede. Este tipo de função pode ser usado para determinar a carga de utilização de uma rede durante um certo período.

2.11.3 FUNÇÕES DE RECEPÇÃO

Os gerentes SNMP podem receber mensagens do SNMP emitidas espontaneamente por dispositivos de rede. Mensagens espontâneas SNMP são uma parte importante do padrão e permitem aos dispositivos informar problemas. Em função das mensagens espontâneas serem assíncronas e fora do controle do gerente da rede, os gerentes SNMP requerem algum tipo de filtro de mensagem para eliminar mensagens indesejadas ou não pertinentes.

2.11.4 CONFIGURANDO FERRAMENTAS DO GERENCIADOR

Os gerentes SNMP possuem uma configuração de ferramentas. O tipo mais tradicional de ferramenta de gerenciamento SNMP é um browser de MIB que permite um usuário inspecionar o MIB cujo objeto seja um dispositivo em particular. Esta é freqüentemente a interface principal para fixar valores de SNMP de um agente e aplicar mudanças na rede.

2.11.5 COMPILADOR DE MIB

O compilador de MIB tem a funcionalidade de adicionar novos objetos na MIB para os equipamentos de rede, dependendo do recurso requerido que a rede está necessitando.

3. A PLATAFORMA JDMK

O JDMK é uma solução baseada em Java para construir e distribuir inteligência de gerenciamento em sistemas, aplicações e dispositivos de rede. Para o desenvolvimento ele utiliza classes de Java e ferramentas que simplificam o desenvolvimento de agentes dinâmicos e extensíveis ([SUN1999]).

Um agente é uma aplicação que oferece um ou mais serviços. Os agentes agem como assistentes invisíveis, o alertam para problemas potenciais ao longo da rede ou executam tarefas de gerenciamento sem intervenção humana. Agentes são inteligentes, autônomos e dinâmicos. Eles podem carregar serviços de gerenciamento do servidor de rede e podem incluir novos serviços para que fiquem disponíveis. Isto permite que um novo serviço de gerenciamento pode ser iniciado a qualquer hora.

3.1 DIFERENÇAS ENTRE O JDMK E AS DEMAIS PLATAFORMAS

Segundo [SUN1999], o gerenciamento de rede é executado através de aplicações de gerência. Estas aplicações de gerência são auxiliadas por agentes, que agem como os intérpretes e filtros e enviam comandos aos elementos de rede, controlando e armazenando informações sobre os elementos da rede. As aplicações de gerência controlam os meios que estes agentes operam, pois estes contêm pequena inteligência de gerenciamento e podem executar apenas operações básicas de gerenciamento de rede.

Desenvolvedores têm que escolher uma única tecnologia de gerenciamento para várias existentes no mercado. Em alguns casos, os desenvolvedores podem implementar gerenciamento de múltiplas tecnologias para suprir uma cobertura mais ampla dos mercados potenciais, como por exemplo, SNMP e CMIP.

O JDMK é uma plataforma que oferece uma instrumentação uniforme por administrar sistemas, aplicações e sistemas de gerenciamento de rede. Também é a plataforma chave, na qual habilita a rede de *service-driven*. A rede *service-driven* é uma nova aproximação para a rede que computa e se concentra nos serviços que serão utilizados. Estes serviços variam dos serviços de baixo nível que administram relações

entre dispositivos de rede para os serviços "*value-added*" que podem ser fornecidos aos usuários finais.

Com o JDMK, os serviços são diretamente incorporados nos agentes. São concedidas permissões aos agentes para executar o gerenciamento da rede e a sua distribuição.

Assim, o JDMK abre a porta para novos tipos de aplicações de gerenciamento que podem ser criadas, desenvolvidas, adicionadas ou eliminadas em tempo de execução.

3.2 BENEFÍCIOS DO JDMK

Segundo [SUN1999], o JDMK apóia protocolos de gerenciamento, como o SNMP. Também usa um mecanismo dinâmico baseado na Internet que permite que serviços novos sejam carregados do servidor de rede em tempo de execução. Não só são implementados serviços dentro de dispositivos, mas também pode ser baseados em redes, armazenados em arquivos como simples páginas na rede da mesma maneira como tecnologias baseadas em *applets* Java.

Podem ser criados serviços e carregados quando necessário. A funcionalidade de gerenciamento autônomo ainda é outra característica chave do JDMK, permitindo administrar uma grande base instalada.

3.3 COMO OPERA

O JDMK opera com o JMX (*Java Management Extensions*) para administrar seus recursos. Para assegurar que um recurso é um recurso administrativo, é necessário ter certeza que se adere às especificações do JMX, que fornece uma especificação para implementar recursos administrativos. São implementados recursos administrativos como Gerenciadores MBeans.

Um MBean é um recurso de gerenciamento em conformidade com o padrão JMX, que pode ser qualquer recurso que se deseja administrar. Por exemplo, uma aplicação, uma implementação de um serviço, um dispositivo, um usuário e assim sucessivamente.

O JDMK é o primeiro produto que implementa a especificação pública para o JMX. O JMX define uma arquitetura, padrões de especificação, API's, serviços para aplicação e gerenciamento de rede, tudo dentro de uma única especificação.

Uma das maiores vantagens do JDMK é que permite administrar recursos de uma localização remota. Assim, pode-se manipular objetos sobre os MBeans em um agente remoto:

a) buscando um valor de atributo;

b) alterando um valor de atributo.

Pode-se ter todo o acesso remoto do JMX aos recursos administrativos:

a) chamando uma operação;

b) recebendo uma notificação espontânea emitida por um recurso administrativo.

O gerente também pode adicionar um novo recurso JMX remotamente por:

a) usando classes Java existentes já carregadas no agente;

b) usando classes novas carregadas em uma localização arbitrária.

O JDMK gradativamente simplifica o desenvolvimento de aplicações de gerenciamento fornecendo comunicação de multi-protocolo entre o lado do gerente e do agente. Podem ser gerenciados dispositivos ou aplicações diretamente por um *browser* ou por uma aplicação SNMP existente. Protocolos adaptadores (por exemplo, HTTP/TCP, RMI, SNMP) permitem ao desenvolvedor adaptar as aplicações para o ambiente de comunicação local.

Para criar novos agentes, usa-se um gerador de objeto de gerenciamento (proxygen) que vem acompanhado com o JDMK. Com esta ferramenta, pode-se criar aplicações com módulos gerados de acesso remoto que gerenciam no auxílio da

comunicação do protocolo. Pode-se criar um objeto de procura para cada objeto JMX que se deseja administrar remotamente.

O JDMK também oferece um *toolkit* para SNMP, para o desenvolvimento de agentes e gerentes, inclusive o compilador de *mibgen* que encapsula o SNMP nas MIB geradas na forma de MBeans que implementam as MIB.

3.4 FERRAMENTAS AGREGADAS

O JDMK possui ferramentas que contribuem na especificação do JMX. Ele oferece um mecanismo baseado na Web para automaticamente utilizar a propagação de serviços de gerenciamento pela rede para os agentes.

Novos serviços podem ser adicionados e velhos serviços podem ser removidos dependendo das exigências do gerenciador. Os serviços fornecidos incluem o seguinte: monitoramento, timer, mlet e filtro ([SUN1999]).

Ferramentas para criação de serviços:

a) um gerenciador de gerador de objeto (proxygen) que auxilia os desenvolvedores a criar as próprias tecnologias específicas baseadas em Java para serviços de gerenciamento;

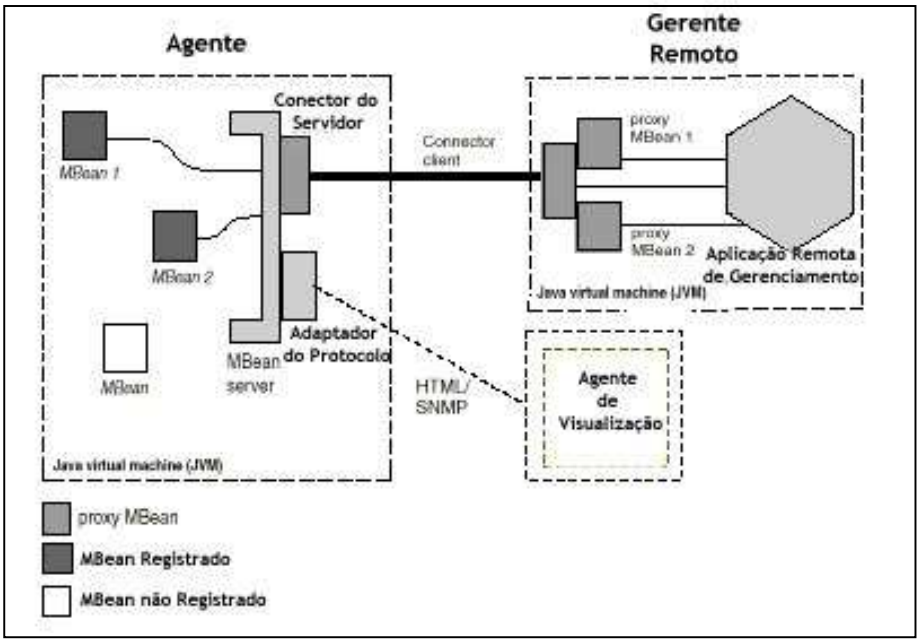
b) um compilador de MIB SNMP (mibgen) com tecnologia baseada em Java que incorpora as entradas e saídas dos MBeans. Esta ferramenta implementa as MIB, habilitando o agente JDMK a ser gerenciado por um gerente SNMP.

3.4.1 CONCEITOS CHAVES

Segundo ([SUN1999]), há cinco palavras-chave para definir o gerenciamento de rede na plataforma JDMK, no qual são as seguintes: servidor de MBean, MBeans, adaptador, conectores e proxy MBeans.

Segue abaixo os conceitos chaves para o JDMK:

FIGURA 19 – CONCEITOS CHAVES PARA O JDMK



Fonte: SUN[1999]

4. MBEANS

O nível de instrumentação JMX oferece padrões para implementar recursos de gerenciamento JMX. A instrumentação de um determinado recurso de gerenciamento é fornecido por um ou mais Java Beans de gerenciamento ou MBeans. Um MBean é um objeto Java que trabalha com certos padrões de especificação para expor atributos e operações ([SUN1999]).

Estes atributos e operações habilitam qualquer agente JDMK para reconhecer e gerenciar o MBean. Os padrões de especificação para MBeans dão ao desenvolvedor controle total dos recursos, dispositivos ou aplicações que serão gerenciados. Por exemplo, padrões de atributo que permitem fazer a distinção entre as propriedades de somente leitura ou leitura e escrita em um MBean. MBeans implementam uma interface de gerenciamento que define os atributos e operações acessíveis para gerenciamento.

MBeans podem gerar e podem propagar eventos espontâneos de notificação para aplicações de gerenciamento remoto. MBeans devem ser desenvolvidos de acordo com as regras especificadas pelo *Java Management Extensions* (JMX).

A arquitetura de JMX permite fazer distintas operações em um agente, passando pelo seu servidor de MBean, tais como ([ODA1998]):

a) listar MBeans;

b) habilitar MBean para ter acesso localmente ou remotamente;

c) criar novos MBeans;

d) executar operações definidas pelo MBeans;

e) receber notificações emitidas por MBeans;

f) mostrar a interface de gerenciamento de um MBean;

g) permitir desenvolver um gerente genérico;

h) fornecer uma visão do MBean para um protocolo específico.

Um novo MBean pode ser administrado assim que seja registrado dentro do servidor de MBean. Um MBean pode ser criado e pode ser registrado por:

a) outro objeto dentro do agente;

b) uma aplicação de gerenciamento remoto que conectou ao agente.

Os serviços de gerenciamento fornecidos com o JDMK são implementados com MBeans. Também podem ser escrito os MBeans que representam objetos que se deseja administrar. Estes objetos poderiam incluir os recursos atuais para todos aqueles que queiram administrar e utilizar os serviços de administração de recursos.

O JDMK não impõe nenhuma restrição quanto ao local onde foi compilado e foram armazenadas as classes MBean. Elas podem ser armazenadas em qualquer localização especificada no CLASSPATH, variável de ambiente do agente, ou em um local remoto. Pode-se desenvolver dois tipos de MBeans ([SUN1999]):

a) MBean Padrão;

b) MBean Dinâmico.

4.1 MBEAN PADRÃO

MBeans padrão são objetos Java em conformidade com padrões de especificação derivados do modelo de componentes JavaBeans. Os MBeans padrão lhe permitem definir sua interface de gerenciamento como uma Interface de Java. Os métodos para leitura e escrita e os atributos do MBean que podem chamar suas operações são descritos nesta interface de gerenciamento.

São definidos MBeans padrão estáticos, aqueles em que os elementos são definidos na interface de gerenciamento no momento da compilação e não podem ser alterados em tempo de execução. Eles são o tipo mais rápido e mais fácil de MBeans para

implementar quando se deseja criar novos MBeans de gerenciamento de recursos em que a estrutura de dados é previamente definida.

4.2 MBEAN DINÂMICO

MBeans dinâmicos permitem a uma aplicação de gerenciamento descobrir a interface de gerenciamento de um recurso em tempo de execução, ele oferece um modo simples para criar um novo MBean utilizando recursos de MBeans padrões. MBeans dinâmicos são incorporados nas estruturas de gerenciamento que mudam com o passar do tempo [SUN1999].

4.3 SERVIDOR MBEAN

O servidor de MBean é um registro para os recursos de gerenciamento do JMX nos quais são expostas as operações de gerenciamento de um agente. Ele oferece um gerenciamento de serviços para manipular os recursos JMX de gerenciamento através de protocolos independentes. Todas as operações de gerenciamento executadas nestes recursos são feitos pela *interface* do servidor de gerenciamento MBean. Os objetos podem ser criados e gerenciados dentro do servidor de MBean, assim, tornando-os visível para aplicações de gerenciamento.

O servidor de MBean inclui uma biblioteca de agente que pode reutilizar serviços na forma de objetos de gerenciamento, quando especificado pelo JMX. Estes serviços incluem mlet, timer, monitoramento e filtro.

Os objetos podem ser registrados no servidor de MBean:

a) uma própria aplicação de agente;

b) uma aplicação de gerenciamento remoto (por um conector ou um adaptador de protocolo).

Qualquer recurso de gerenciamento JMX que queira ter acesso a operações de gerenciamento deve ser registrado no servidor de MBean. Este MBean precisa ser

identificado com um nome de objeto. O gerente e o agente usam este nome de objeto para identificar qual objeto irá executar uma operação de gerenciamento. É possível ter múltiplos servidores MBean dentro do mesmo agente JDMK.

4.4 PROXY MBEANS

Segundo [SUN1999], um Proxy MBean representa a visão de gerente do MBean gerenciado onde o gerente foi desenvolvido usando o servidor de Interface MBean remoto. Esta interface faz a funcionalidade do servidor de MBean disponível na aplicação de gerenciamento remoto e deixa para o gerente interagir diretamente com o servidor de MBean do agente.

O gerente tem acesso a um MBean executando operações no Proxy MBean, onde estes são propagados no MBean. Estas operações podem ser:

a) buscar e fixar atributos;

b) executar operações;

c) receber notificações espontâneas emitidas pelo MBean ao gerente remoto de aplicação.

O JDMK possui um compilador chamado proxygen que pode ser utilizado para gerar um proxy MBean a partir de um MBean. Proxy MBeans são ideais quando as interfaces de gerenciamento não precisam ser alteradas em tempo real, ou seja, onde os atributos e operações disponíveis são estáveis.

4.4.1 PROXY MBEANS GENÉRICO (PMG)

Uma Proxy MBean Genérico age da mesma maneira como um Proxy MBean. Porém, o PMG faz com que ele utilize o compilador de proxygen como se fosse o Proxy MBeans, mas simplesmente criando uma instância nova de uma Proxy MBean Genérica.

Proxy MBeans Genéricos são ideais quando deseja-se executar operações em um MBean e quando sabe-se que os atributos e operações daquele MBean mudarão. Isto

faz com que o PMG seja a solução perfeita para sistemas de gerenciamento dinâmicos em que as propriedades de um MBean podem mudar em tempo de execução.

Segundo [SUN1999], o PMG pode ser usado para administrar qualquer MBean padrão ou MBean dinâmico.

4.5 CONECTORES E ADAPTADORES DE PROTOCOLOS

Conectores e adaptadores de protocolos permitem a um agente ter acesso e administrar aplicações de administração remotamente. Conectores conectam aplicações de administração com o servidor de MBean e permitem aos MBeans serem instanciados e administrados remotamente. Todo conector contém a mesma interface de administração remota, mas utilizam um protocolo diferente para o qual permite aplicações de administração remota. Embora o agente tenha um protocolo específico, podem ser conectados um ou mais clientes ao servidor de MBean ([SUN1999]).

O JDMK contém conectores para a maioria dos protocolos principais da internet (por exemplo, HTTP/TCP e RMI).

Um conector está composto de duas partes:

a) um servidor de conexão no lado de agente;

b) um cliente de conexão no lado de gerente.

A interface do servidor de MBean remoto é implementada pelo cliente de conexão. Conectores permitem transpor aplicações de administração sobre um local no gerente, oferecendo o mesmo nível de funcionalidade ao agente.

Protocolos adaptadores têm apenas um componente de servidor e contém a conexão entre uma aplicação de administração remota e um agente, através de um protocolo de rede específico. Do lado do gerente, pode-se ver e administrar qualquer MBean. Isto pode ser feito tendo acesso à interface do servidor de MBean diretamente através de um determinado protocolo.

Um cliente do adaptador SNMP é um gerente SNMP. Além disso, o JDMK fornece um *toolkit* de SNMP que lhe permite gerar MBeans de acordo com MIB padrão ou proprietárias, habilitando novos agentes baseados em tecnologia que podem ser integrados em infra-estruturas de redes existentes, incluindo produtos de administração baseados em SNMP.

Conectores e adaptadores de protocolo habilitam aplicações de administração para:

a) procurar MBeans existente pelos nomes de objetos;

b) buscar ou acrescentar atributos de instâncias de MBean existentes;

c) executar operações em instâncias de MBean existentes;

d) instanciar um MBean e registrar a instância de um novo MBean;

e) receber notificações de MBeans conhecidas (com exceção do adaptador HTML).

Segundo [SUN1999], porém, um agente pode incluir qualquer número de conectores ou adaptadores de protocolo, possibilitam a administração remota para muitos gerentes por protocolos diferentes. *Proxys* são utilizados em MBeans ao lado de gerentes para executar operações de administração nos MBeans correspondentes aos agentes.

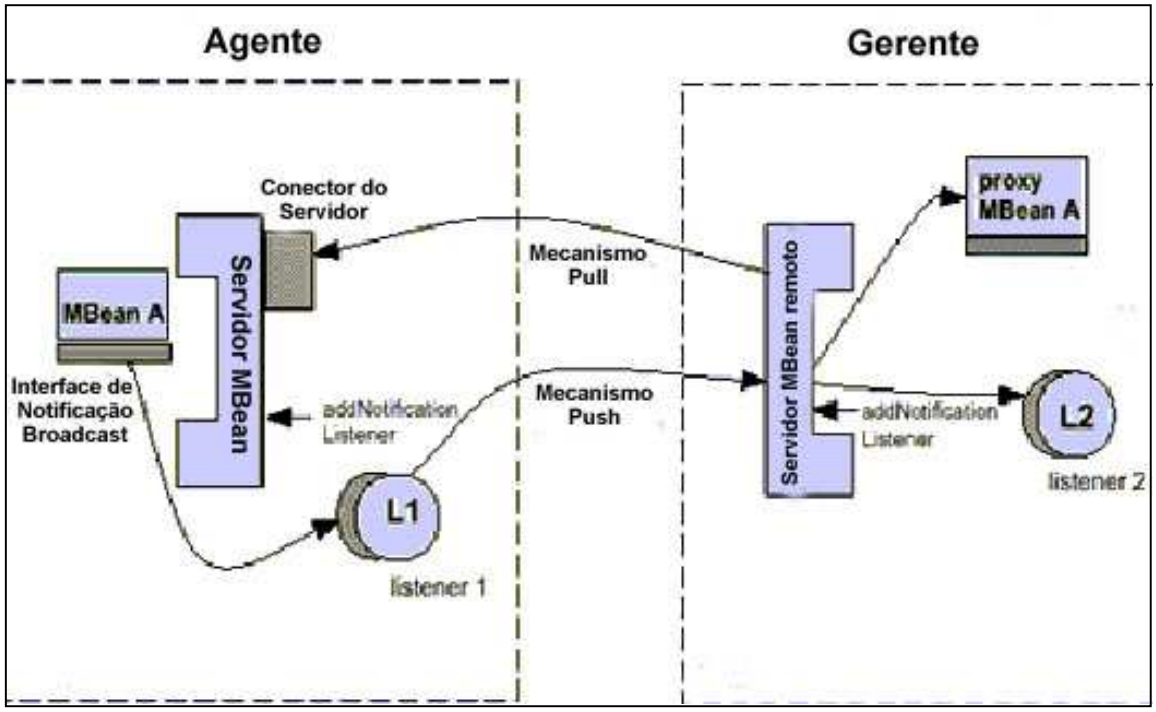
Estes MBeans são registrados com o servidor de MBean que é independente de protocolo. Nem conectores nem adaptadores de protocolo são inicializados automaticamente quando são criados; é necessário inicializá-lo manualmente.

4.6 MODELO DE NOTIFICAÇÃO DO JMX

A interface de administração de um MBean permite a um agente de MBean executar controle e operações de configuração nos recursos de administração. O modelo de notificação do JMX permite para os MBeans enviar notificações de *broadcast* para o

administrador, aplicações de administração e outros objetos registrados que ficarem esperando uma notificação de um *broadcast* de um MBean.

FIGURA 20 - MECANISMO DE NOTIFICAÇÃO REMOTA



Fonte: [SUN1999]

Também pode ser adicionado um ouvinte no servidor de MBean remoto ou em um Proxy MBean. Uma maneira para se adicionar um ouvinte para o gerente pode ser chamando a operação de *addNotificationListener* apropriada para *Proxys* MBean. Isto vai transmitir uma chamada para o agente e um ouvinte de protocolo dependente, fazendo com que o agente verifique a chamada recebida, criando no servidor de MBean (Figura 20). No protocolo dependente, o ouvinte gerado é administrado pelo servidor. Está é a principal rotina, de enviar notificações ao longo da rede tanto para o agente como para o gerente, que usam um protocolo especificado.

4.7 O MECANISMO PUSH

Pode ser optado em utilizar o mecanismo Push para enviar notificações de um agente para um gerente. Quando os conectores utilizam o mecanismo Push o servidor de

conexão reenvia as notificações para o cliente de conexão assim que eles sejam emitidos pelos MBeans.

Quando um MBean, ao lado do agente, emite uma notificação, o HTTP ou o adaptador RMI remete esta notificação para o cliente de conexão ao gerente.

4.8 O MECANISMO PULL

Pode ser optado em utilizar o mecanismo Pull quando seja necessário enviar notificações de um agente para um gerente. Isto pode acontecer quando o agente está impossibilitado de conectar-se a um gerente. Por exemplo, se há um *firewall* entre o agente e o gerente.

Os Pull acontecem quando o servidor de conexão do HTTP/RMI do agente é instruído pelo gerente para tentar buscar notificações, assim, o gerente decide quando e com que freqüência quer recobrar as notificações. Por exemplo, o Pull poderia tentar adquirir notificações a cada trinta segundos ou a todo minuto.

5. CRIANDO UM AGENTE SNMP

Para desenvolvimento da aplicação do Agente SNMP, no gerenciamento de dispositivos de rede, foi utilizada a plataforma JDMK. Para a utilização do JDMK é necessário que se converta a MIB SNMP a ser utilizada em um formato específico requerido pela ferramenta: um conjunto de MBeans. Os MBeans são objetos Java (Java Beans de gerenciamento) que contém as propriedades dos objetos do modelo de informação definido na MIB SNMP.

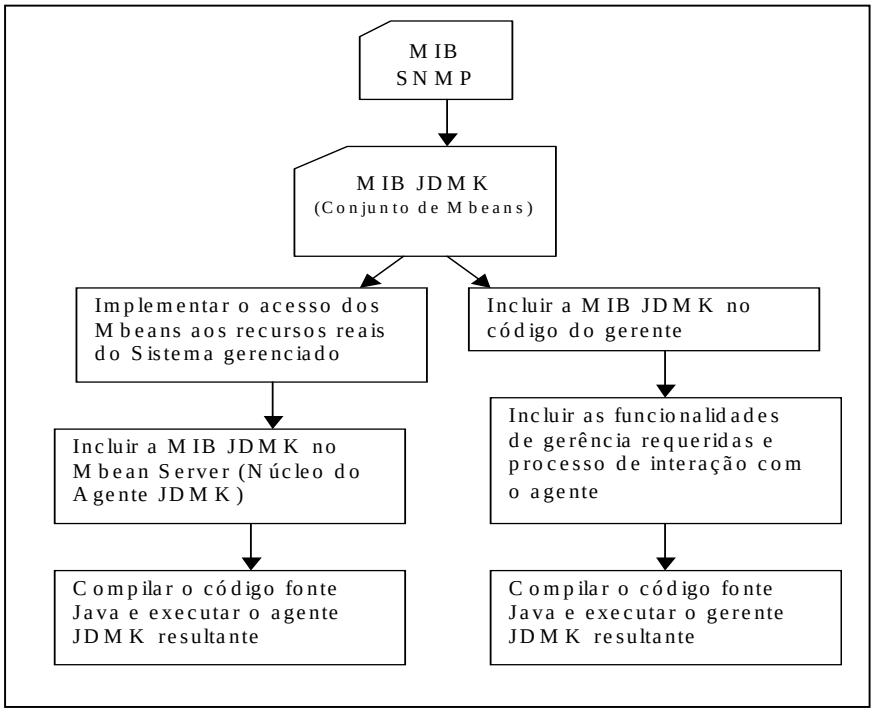
Para gerar o código dos MBeans que representam uma MIB é necessário utilizar a ferramenta Mibgen. A ferramenta Mibgen cria a estrutura de get e set utilizada para ler ou escrever variáveis internas.

O SNMP interage com uma MIB personalizada para implementar um agente compatível com SNMPv1 e SNMPv2. Também fornece os mecanismos para enviar Traps e implementar o controle de acesso comunitário.

5.1 PROCESSO DE DESENVOLVIMENTO DE UMA MIB

O processo de implementação de uma MIB SNMP para ser utilizada pelo ambiente de desenvolvimento de aplicações de gerência JDMK é composto por um grupo de procedimentos bem específicos, como pode ser visto na Figura 21.

FIGURA 21 – PROCESSO DE DESENVOLVIMENTO DE UMA MIB



Nas próximas sessões será discutido e detalhado cada um dos procedimentos envolvidos no processo de desenvolvimento de uma aplicação agente utilizando o JDMK como foi representado esquematicamente na Figura 21 (a aplicação gerente é desenvolvida de forma equivalente).

Será descrito a seguir o processo de compilação da MIB para convertê-la para o formato requerido pelo agente JDMK. Neste processo será utilizado o subconjunto da MIB-II definida por RFC1213.

5.1.1 GERADOR DE MIB MBEANS

Para obter o conjunto de MBeans constituintes da MIB JDMK a partir da MIB SNMP utiliza-se a ferramenta Mibgen.

Para executar a ferramenta de Mibgen, após instalar a Plataforma JDMK, deve-se estar no diretório onde está o seu arquivo de especificação da MIB e executar o seguinte comando:

$mibgen -d. mib_II_TCC.txt

Isto gerará os seguintes arquivos no diretório atual:

a) o MBean (por herança) para a MIB:

- RFC1213_MIB.Java.

b) o MBean e sua classe metadata para o grupo **SNMP**:

- Snmp.Java;
- SnmpMBean.java;
- SnmpMeta.Java.

c) o MBean e sua classe metadata para o grupo **Sistema**:

- System.Java;
- SystemMBean.java;
- SystemMeta.java.

d) o MBean e sua classe metadata para o grupo **Interfaces**:

- Interfaces.Java;
- InterfacesMBean.Java;
- InterfacesMeta.Java.

e) a classe que representa uma tabela de interface, e o MBean que representa entradas na tabela:

- TableIfTable.Java;
- IfEntry.Java;
- IfEntryMBean.Java;
- IfEntryMeta.Java.

f) classes que representam tipos enumerados que são utilizados nestes grupos e entradas:

- EnumSnmpEnableAuthenTraps.Java;
- EnumIfOperStatus.Java;
- EnumIfAdminStatus.Java;
- EnumIfType.Java.

g) tabela de OID para representar a definição de nome de todas as variáveis de MIB:

- RFC1213_MIBOidTable.java

O MBean com o nome da MIB é apenas a classe administrativa central para administrar os outros MBeans que foram implementados na MIB. Todos os outros MBeans contêm as variáveis SNMP como atributos da interface de administração ([SUN1999]). A ferramenta Mibgen gera MBeans padrões para a MIB, assim são implementados métodos individuais como get e set para cada um dos atributos em função das suas permissões de acesso.É necessário que sejam implementados os métodos get e set para ler e escrever dados.

Considerando que o SNMP não apóia ações em MIB, as únicas operações nestes MBeans são implementações de checagem do set SNMP solicitando variáveis que possam ser alteradas. Operações podem ser adicionadas, mas o gerente SNMP não poderá ter acesso. Porém, outros gerentes poderão chamar estas operações se estiverem conectados por outro protocolo.

5.2 IMPLEMENTANDO OS RECURSOS DA MIB

A implementação da MIB implica no desenvolvimento de um conjunto de classes que implementam as interfaces JAVA geradas como resultado da compilação da MIB SNMP através do Mibgen.

O trabalho só implementa uma parte dos métodos de acesso aos atributos contidos no RFC1213. Estas implementações são contidas nas classes com o sufixo de “Impl”. Essas classes foram geradas através do Mibgen de forma que podem ser alteradas para sua personalização.

Segue abaixo um apanhado da implementação mostrado no agente:

a) InterfaceImpl.java - adiciona uma nova entrada de notificação para o objeto de IfTable, então esta cria mais duas novas entradas da tabela com variáveis que os adiciona na tabela, associando com:

- TableEntryListenerImpl.java – cria o *listener* para a entrada da tabela de notificações de eliminação: imprime os valores da entrada na nova tabela;
- IfEntryImpl.java – implementa a tabela de entrada e fornece um método interno para troca de variável de OperStatus que gera um trap. Este método não está exposto na interface do MBean, assim está só disponível no código desta aplicação de agente.

b) SnmpImpl.java - inicializa e implementa variáveis do grupo SNMP, muitas destas são variáveis estáticas do agente SNMP. Assim chama-se os métodos de get do objeto adaptador SNMP para retornar as informações;

c) SystemImpl.java - inicializa as variáveis de grupo de sistema.

O SnmpImpl.java e arquivos do SystemImpl.java fornecem código que pode ser utilizado para implementar posteriormente estes grupos SNMP comuns.

5.2.1 COMPILANDO O MBEANS E AGENTES

Para compilar os arquivos .java, o classpath têm que conter o diretório atual (.), deve-se criar uma pasta patchfiles na pasta atual e copiar o arquivo RFC1213_MIB.java criado para a pasta patchfiles. Em seguida executa-se o seguinte comando:

*$ javac -classpath classpath -d . *.java*

5.3 O ADAPTADOR DE PROTOCOLO SNMP

Com a MIB implementada na forma de MBeans, a aplicação de agente necessita de um adaptador de protocolo SNMP para funcionar como um agente SNMP. Considerando-se que o adaptador do protocolo SNMP também é um MBean, os mesmo MBean também pode ser criado e iniciado dinamicamente em seu agente por um gerente conectado ou pelo próprio adaptador HTML. Conforme quadro 01, segue abaixo o código do Agente SNMP no qual foi implementado o adaptador de protocolo SMNP.

QUADRO 01 - APLICAÇÃO DO AGENTE SNMP

```
public class Agent {
  static SnmpAdaptadorServer snmpAdaptador = null;
  public static void main(String args[]) {
          MBeanServer server;
          ObjectName htmlObjName;
          ObjectName snmpObjName;
          ObjectName mibObjName;
          ObjectName trapGeneratorObjName;
          int htmlPort = 8082;
          int snmpPort = 8085;
          [...]
          try {
                  server = MBeanServerFactory.createMBeanServer();
                  String domain = server.getDefaultDomain();
                  // Cria um adaptador HTML
                  //
                  htmlObjName = new ObjectName( domain +
```

```
":class=HtmlAdaptadorServer,protocol=html,port=" + htmlPort);
            HtmlAdaptadorServer htmlAdaptador = new  HtmlAdaptadorServer(htmlPort);
            server.registerMBean(htmlAdaptador, htmlObjName);
            htmlAdaptador.start();
            // Cria um adaptador SNMP
            //
            snmpObjName = new ObjectName(domain +
            ":class=SnmpAdaptadorServer,protocol=snmp,port=" + snmpPort);
            snmpAdaptador = new SnmpAdaptadorServer(snmpPort);
            server.registerMBean(snmpAdaptador, snmpObjName);
            snmpAdaptador.start();
                  // Envia um trap SNMP  (porta utilizada = snmpPort+1)
                  // Trap's sao definidas no arquivo ACL
                  //
                  snmpAdaptador.setTrapPort(new Integer(snmpPort+1));
                  snmpAdaptador.sendV1Trap(0, 0, null);
                  // Cria o MIB-II (RFC 1213) e adiciona no MBean Server
                  //
                  mibObjName = new ObjectName("snmp:class=RFC1213_MIB");
                  RFC1213_MIB mib2 = new RFC1213_MIB();
                  // O MBean ira registrar todo o grupo e tabela de entradas do
        MBeans
                  server.registerMBean(mib2, mibObjName);
                  // Une o  adaptador SNMP para a MIB
                  mib2.setSnmpAdaptadorName(snmpObjName);
                  [...]
                  } catch (Exception e) {
                  e.printStackTrace();
                  }
                  }
                  // Chama a referencia get no objeto adaptador SNMP
                  static public SnmpAdaptadorServer getSnmpAdaptador() {
                  return snmpAdaptador;
        }
}
```

5.3.1 EXECUTANDO O ADAPTADOR SNMP

O adaptador SNMP é executado da mesma forma que se executa o adaptador *Hipertext Markup Language* (HTML). Primeiramente cria-se um nome de objeto para o MBean, então instancia-se a classe com um construtor que permite especificar uma porta não padrão, registra-se o MBean como servidor MBean e então executa-se o adaptador para ativá-lo.

Como valor padrão, o protocolo SNMP utiliza a porta 161. Considerando que outras aplicações podem estar usando este porta, o agente utiliza a porta 8085. Quando conecta-se a este agente o gerente SNMP precisará especificar este número de porta.

5.3.2 CRIANDO A MIB

A aplicação do agente cria e administra uma MIB, o subconjunto de MIB-II. Para isso instancia-se o correspondente RFC1213_MIB com os MBeans que foram gerados pela ferramenta Mibgen.

O processo de registro deixa instanciar o MBean e registrar outros MBeans que representam os grupos da MIB e as entradas de suas tabelas. A configuração de todos estes MBeans, ao término do registro, compõe a representação da MIB na forma de MBeans.

5.3.3 LIGANDO MBEANS DA MIB

O MBean principal de uma MIB deve ser ligado explicitamente à instância do adaptador SNMP. O adaptador SNMP não interage com qualquer outro MBean nem com o servidor de MBean ([SUN1999]).

Segundo ([SUN1999]), depois que uma MIB for instanciada, deve-se fixar os atributos do SNMPAdaptadorName no MBean principal para que seja ligado ao adaptador SNMP. Este método pode ser chamado diretamente de setSnmpAdaptadorName, se o MBean da MIB foi registrado no servidor de MBean. Deste modo, o adaptador SNMP terá uma referência de toda as MIB que tem que exibir.

No processo que liga o adaptador SNMP e a raiz OID da MIB, o adaptador usa este OID para determinar quais variáveis são implementadas na MIB correspondente. Para que o adaptador SNMP solucione um pedido em um determinado OID, as raízes OID de todas as MIB devem ser distintas. Isto implica que nenhuma raiz OID pode ser igual a outro ou pode ser uma sub-cadeia de outro.

Embora o adaptador SNMP possa ser registrado no servidor de MBean, o adaptador só faz MIB visíveis para gerentes SNMP. O gerente SNMP está limitado por seu protocolo, pois não pode tirar total proveito de um agente JDMK pela MIB básica, e não tem acesso a quaisquer outros MBeans.

5.3.4 TENDO ACESSO AO MBEAN DA MIB

Uma vez que o MBean que representa uma MIB foi instanciado e ligou-se com o adaptador SNMP, a MIB passa a ser acessada através do adaptador SNMP. Gerentes SNMP podem enviar pedidos para operar os conteúdos da MIB. O adaptador SNMP interpreta os pedidos de administração SNMP, executa a operação no MBean

correspondente e retorna a resposta SNMP para o gerente. O adaptador de protocolo SNMP é compatível com SNMPv1 e SNMPv2, com a exceção de InformRequest.

A vantagem de ter um agente SNMP num JDMK é que se pode usar os outros protocolos de comunicações para interagir com MIB e administrar o adaptador SNMP. No agente, a MIB que foi registrada pode ser visualizada pelo gerente da rede os seus MBeans customizados em um navegador de rede pelo adaptador de protocolo HTML.

Se o agente incluir outros conectores nas aplicações de administração, poderiam ser conectados ao agente e também poderiam administrar a MIB e o adaptador SNMP. Um gerente não-SNMP pode instanciar novos objetos MIB e ligá-los ao adaptador SNMP e operar os atributos fornecidos nas operações da MIB.

Os gerentes não-SNMP podem operar as variáveis de uma MIB, podem adquirir e podem fixar valores embora qualquer gerente SNMP também passa ter este tipo de acesso pelo adaptador SNMP.

5.3.5 ADMINISTRANDO O ADAPTADOR SNMP

Gerentes não-SNMP também podem controlar o agente SNMP pelo MBean do adaptador SNMP. Como as outras comunicações do MBean, a porta e outros atributos podem ser modificados pelo adaptador SNMP quando ele estiver parado. Pode-se também adquirir informação sobre seu estado, parado ou reiniciado para controlar quando estiver *on-line*. São definidos estes atributos administrativos e operações na interface de CommunicatorServerMBean.

O Servidor do adaptador SNMP também implementa o SnmpAdaptadorServerMBean que conecta para definir sua informação operacional. O grupo SNMP de MIB-II define certas variáveis estáticas que os agentes SNMP tem que mostrar. Por exemplo, o adaptador SNMP fornece métodos para getSnmpInPkts e getSnmpOutBadValues. Gerentes não-SNMP podem ler estas variáveis como atributos do adaptador SNMP MBean.

Segundo [SUN1999], o adaptador SNMP também fornece outra informação operacional que não é disponível aos gerentes SNMP. Por exemplo, o *ActiveClientCount* e *ServedClientCount* no qual são relatório de atributos de somente leitura do gerente SNMP de conexões para este agente. Atributos de leitura e escrita do *BufferSize* que permite mudar

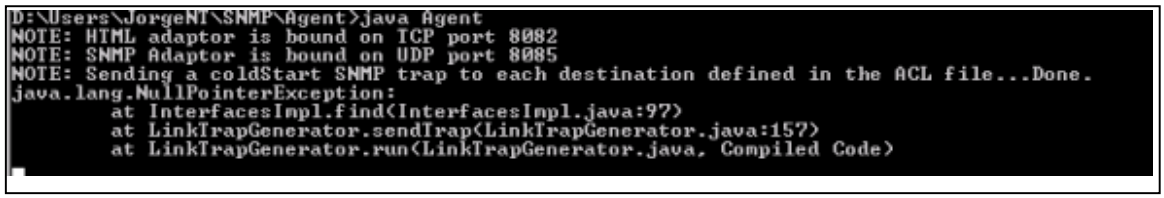
o tamanho da mensagem do *buffer* quando o adaptador estiver parado. O adaptador MBean também fornece operações para enviar Traps ou implementar sua própria segurança.

5.4 EXECUTANDO O AGENTE SNMP

Depois de construir o Agente SNMP descrito anteriormente, para executar o agente deve-se executar o seguinte comando na sua pasta de implementação, conforme a Figura 22:

$ java Agent

FIGURA 22 – EXECUTANDO O AGENTE SNMP



Agora esta criado o acesso ao adaptador de HTML deste agente apontando um navegador de rede para a seguinte URL: http://localhost:8082. Pelo adaptador HTML, pode-se ver os MBeans que representam a MIB (Figura 23):

FIGURA 23 – ACESSANDO O AGENTE VIA PROTOCOLO HTML

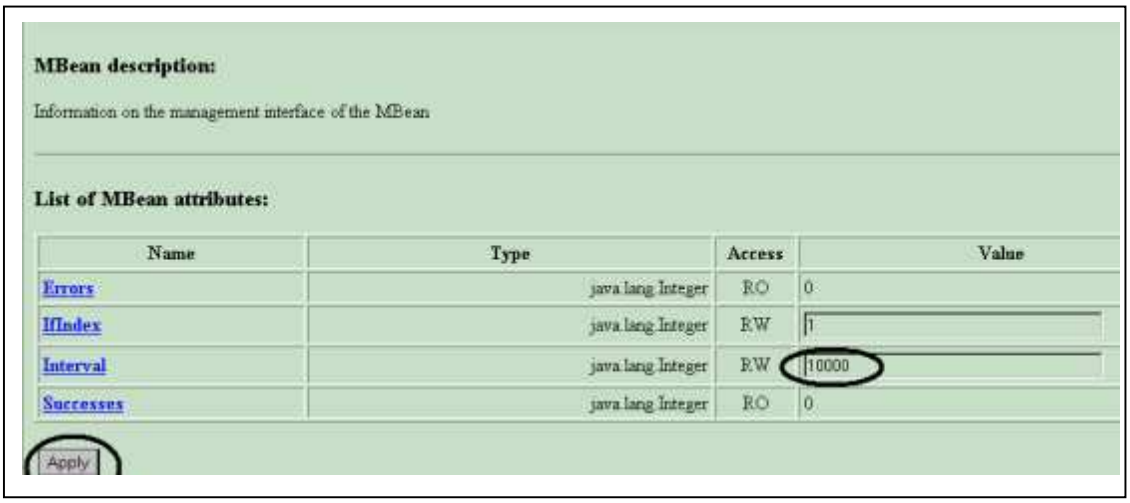
a) a classe RFC1213_MIB MBean do domínio SNMP é a classe principal da MIB. Nela há um nome e a informação sobre o adaptador SNMP para o qual o MBean é conectado;

b) o domínio RFC1213_MIB contém os MBeans de cada grupo;

c) o domínio de ifTable contém as entradas das tabelas de interface.

Em quaisquer destes MBeans, pode-se escrever novos valores nos campos de texto de atributos fornecidos e então pode-se selecionar o botão de “Apply”. Esta configuração fixa a variável correspondente ao SNMP, e depois disso, os gerentes SNMP verão o novo valor.

FIGURA 24 – INTERAGINDO COM O AGENTE SNMP



5.4.1 ENVIANDO TRAPS

A aplicação de agente SNMP também demonstra como o agente pode enviar Traps. Personalizando a classe IfEntryImpl pode-se fornecer um método que troca a variável de *IfOperStatus* e enviar um Trap.

Alterando o código gerado, pode-se fazer com que uma entidade do agente troque o estado de operação e assim a variável de MIB será atualizada e um Trap será enviada aos gerentes SNMP.

O quadro 02 contém o código para enviar um trap pelo adaptador SNMP.

QUADRO 02 - ENVIA UMA TRAP NA CLASSE DE IFENTRYIMPL

```
public void switchifOperStatus() {
   // implementa a troca das chamadas indiretas sendTrap
   [...]
}
// Metodo de chamada apos a varivel estiver trocada
// Pode ser chamado com o padrao ==2 (up) ou 3 (down ou testing)
public void sendTrap(int generic) {
            SnmpAdaptadorServer snmpAdaptador = null;
            // Devolve  a referencia do protocolo SNMP atraves
            // do metodo statico do Agente ou da classe StandAloneSnmpAgent
            snmpAdaptador = Agent.getSnmpAdaptador();
            if (snmpAdaptador == null) {
              SNMPAdaptador = StandAloneSnmpAgent.getSnmpAdaptador();
            }
if (snmpAdaptador == null) {
return;
}
            Vector varBindList = new Vector();
            // IfIndex  e a entrada da tabela IF
            SnmpOid oid1 = new SnmpOid("1.3.6.1.2.1.2.2.1.1." + IfIndex);
            SnmpInt value1 = new SnmpInt(IfIndex);
            SnmpVarBind varBind1 = new SnmpVarBind(oid1, (SnmpValue)
             value1);
            varBindList.addElement(varBind1);
            try {
              SNMPAdaptador.sendV1Trap(generic, 0, varBindList);
            } catch (Exception e) {
              e.printStackTrace();
            }
}
```

Considerando que o método de sendTrap é executado em uma linha diferente, ele precisa adquirir uma referência para a instância do adaptador SNMP, onde são chamados de métodos estáticos derivados da nossa implementação do agente. Este código é específico destes agentes e é apenas um exemplo de como restaurar esta informação.

Considerando que nosso programa não tem nenhum status real de operação, então criou-se a classe LinkTrapGenerator que trocará o estado periodicamente. É um MBean que contém uma *thread* com repetição contínua. O período de intervalo entre um trap e outro e o índice da entrada da tabela pode ser modificado pelos atributos do MBean.

QUADRO 03 - A THREAD DO LINK TRAP GENERATOR

```
public void run() {
        while (true) {
                try {
                        sleep(interval);
                } catch (Exception e) {
                        e.printStackTrace();
                }
                sendTrap();
        }
}
public void sendTrap() {
        // recebe a entrada do status de quem sera trocado
        IfEntryImpl ifEntryImpl = InterfacesImpl.find(ifIndex);
        if (ifEntryImpl == null) {
                errors++;
                return;
        }
        ifEntryImpl.switchifOperStatus();
        successes++;
}
```

Para iniciar o gerador de Traps, a aplicação cria um MBean LinkTrapGenerator, conforme quadro 03. Durante o momento em que ele está sendo registrado, o MBean inicia a *thread* e envia um Trap num intervalo padrão de 2 segundos, conforme Quadro 04.

QUADRO 04 - INICIANDO UM TRAP GENERATOR

```
// Cria um LinkTrapGenerator (especifica o ifIndex no nome do objeto)
//
String trapGeneratorClass = "LinkTrapGenerator";
int ifIndex = 1;
trapGeneratorObjName = new ObjectName("trapGenerator:class=" +
        trapGeneratorClass + ",ifIndex=" + ifIndex);
server.createMBean(trapGeneratorClass, trapGeneratorObjName);
```

Para que um adaptador SNMP envie Traps aos gerentes remotos, é necessário definir o grupo de Traps no arquivo de lista de controle de acesso (ACL). Este grupo de definições comunitárias lista todos os *hosts* para os quais o agente enviará todos os Traps ([SUN1999]). Uma definição comunitária associa um nome comunitário com uma lista de hosts, especificado pelo *hostname* ou pelo endereço de IP. Todos os *hosts* receberão os Traps em um pacote identificado pelo nome comunitário.

Segue no quadro 05 um exemplo modelo da especificação do arquivo ACL:

QUADRO 05 – ESPECIFICAÇÃO DO ARQUIVO ACL

```
acl = {
            {
            communities = public
            access = read-only
            managers = yourmanager
            }
            {
            communities = private
            access = read-write
            managers = yourmanager
            }
            }

trap = {
            {
            trap-community = public
            hosts = yourmanager
            }
}
```

São enviados Traps na porta especificada pelo atributo de TrapPort do MBean SnmpAdaptadorServer. No agente foi fixada a porta trap para 8086, mas isto pode ser alterado durante a implementação da MIB ou aplicação de administração.

Se o arquivo de ACL não foi definido ou se o grupo de trap estiver vazio, os Traps serão enviados para o localhost

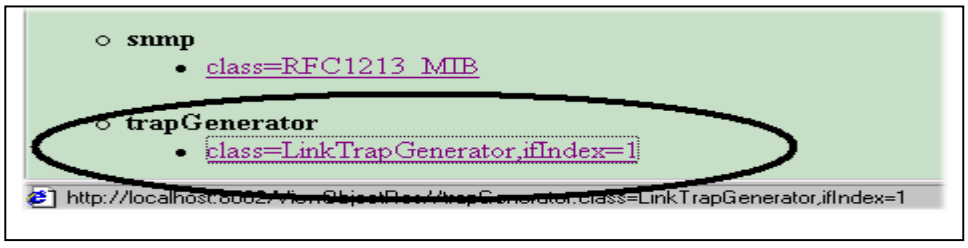
5.4.2 INTERAGINDO COM O GERADOR DE TRAPS

Após criar o MBean que envia Traps para o adaptador SNMP, segue abaixo uma interação com o gerador de Traps:

a) acessando o adaptador HTML deste agente utilizando um navegador HTML para a seguinte URL:

- http://localhost:8082 /;
- Clique no Class=LinkTrapGenerator,ifIndex=1;

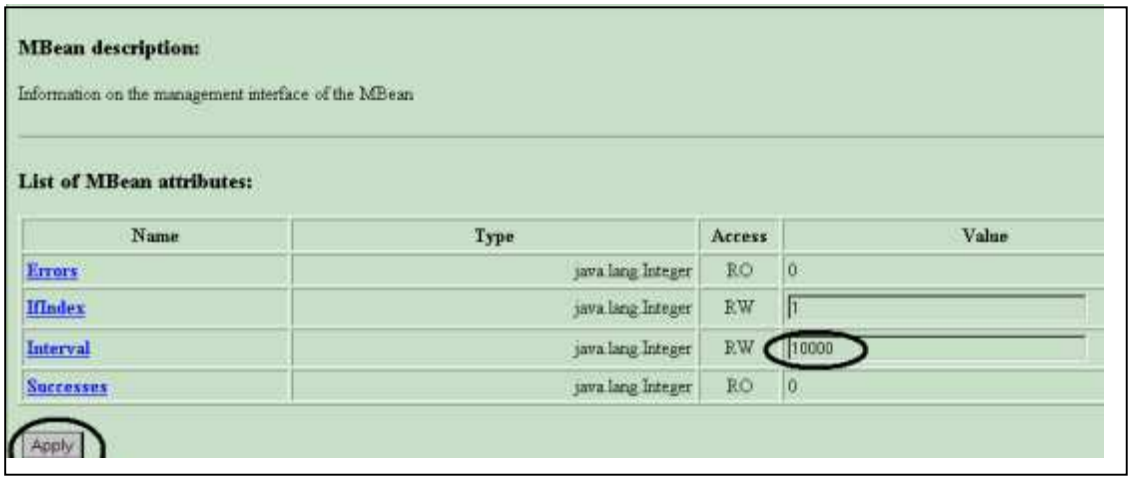
FIGURA 25 – INTERAGINDO COM O GERADOR DE TRAPS



- Pelo adaptador de HTML, pode-se ver o MBean que representa o objeto do gerador de Traps, podendo modificar seus atributos para alterar a entrada da tabela em que se esta operando e mudar o intervalo entre as Traps.

b) mude o intervalo da trap para 10000 de forma que as Traps sejam enviadas a cada 10 segundos;

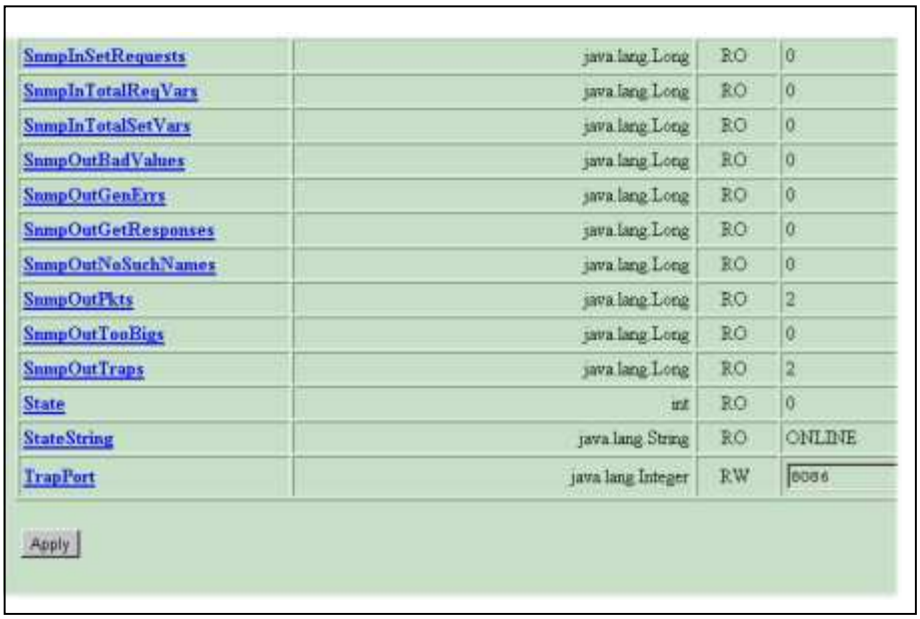
FIGURA 26 – INTERAGINDO COM O GERADOR DE TRAPS II



c) volte para a página do agente e clique no ifEntry.ifIndex=1 ´do MBean ifTable. Configure o período de recarga para 10, e clique no botão “Reload”. O gerador de trap deveria trocar o valor da variável de IfOperStatus, isto acontece devido na implementação da entrada da tabela que envia um trap quando este estado dele é alterado;

d) volte para a página do agente e clique no name=Snmp no MBean RFC1213_MIB. Role a tela para baixo e veja o SnmpOutPkts e variáveis do SNMPOutTraps. Estas variáveis deveriam ser as únicas com valores diferentes de zero, se nenhum gerente conectou ao agente SNMP. O grupo SNMP apresenta informação sobre o adaptador SNMP e pode-se ver quantos Traps foram enviados desde que o agente foi iniciado.

FIGURA 27 – RESULTADO DO ENVIO DE TRAPS NO AGENTE SNMP



5.5 LISTAS DE CONTROLE DE ACESSO (ACL)

Para o adaptador SNMP, o JDMK fornece o controle de acesso baseado no endereço IP e na máquina de host dos gerentes. Informações sobre acessos corretos são armazenadas nas máquinas hosts em um arquivo de ACL ([SUN1999]).

O arquivo de ACL também define os hosts dos gerentes para os quais os agentes enviarão os Traps. Quando um trap é enviado, o agente enviará isto a todos os *hosts* listados nas definições de trap do arquivo de ACL.

Para habilitar o controle de acesso e os Traps no adaptador SNMP, assegure-se que um arquivo de ACL existe quando qualquer agente seja iniciado. O arquivo de ACL deve ser nomeado jdmk.acl e deve ser localizado no diretório de configuração.

No Windows NT o arquivo ACL deve estar no seguinte caminho, Dir\SUNWjdmk\jdmk4.0\JDKversion\etc\conf \

$ java -classpath classpath -Djdmk.acl.file=TCCACL Agente

Se um arquivo de ACL existir, será aplicado o acesso restrito definindo para todos os gerentes ou servidores *proxy* que têm acesso ao agente por seu adaptador SNMP. Se o arquivo de ACL não existir quando os agentes forem inicializados, a todos os gerentes é concedido acesso completo ao agente pelo adaptador SNMP.

5.5.1 FORMATO DO ARQUIVO ACL

Segundo [SUN1999], um arquivo ACL contém um grupo de acl que define ao gerente os direitos de acesso e um grupo de trap que definem quais *hosts* poderão enviar Traps.

5.5.2 FORMATO DO GRUPO ACL

O grupo de acl contém um ou mais listas de configurações de comunidade.

QUADRO 06 – FORMATO ARQUIVO ACL

```
acl = {
list1
list2
  ...
listN
}
```

Cada lista tem o formato exibido no quadro 07:

QUADRO 07 – FORMATO ARQUIVO ACL

```
{
        communities = communityList
        access = accessRights
        managers = hostList
}
```

O communityList é uma lista comunitária SNMP que nomeia para qual controle de acesso se aplicará. A nomeação comunitária nesta lista está separada através de vírgulas.

O *accessRights* especifica os direitos a serem concedidos a todos os gerentes que executam nas máquinas especificadas nos itens dos gerentes. Há dois possíveis valores: leitura e escrita ou somente de leitura.

O artigo de *hostList* especifica as máquinas *hosts* dos gerentes que são concedidos os direitos de acesso. O *hostList* é uma lista de hosts separada por vírgulas, em cada um dos quais podem ser expressados como:

a)Um nome *host*;

b)Um endereço IP;

c)Uma máscara de sub rede.

5.5.3 FORMATO DO GRUPO DE TRAP

O grupo de trap especifica os hosts para os quais o agente pode enviar Traps. Este grupo contém um ou mais definições de comunidade de trap.

QUADRO 08 – ESPECIFICAÇÃO DOS HOSTS PARA ACESSO

```
trap = {
community1
community2
  ...
communityN
}
```

Cada definição é a associação entre uma configuração dos hosts e a cadeia de Traps enviadas no SNMP, de acordo com o formato no quadro 09:

QUADRO 09 – DEFINIÇÃO DOS HOSTS

```
{
        trap-community = trapCommunityString
        hosts = trapInterestHostList
}
```

O item de trapCommunityString especifica a cadeia SNMP que será incluído nas Traps enviadas aos hosts especificados nos itens hosts.

O item de trapInterestHostList especifica uma lista separada por vírgulas dos hosts, onde cada *host* deve ser identificado por seu nome ou endereço de IP completo.

6. CONCLUSÃO

Hoje, as redes de computadores baseadas no protocolo TCP/IP, como por exemplo, a Internet, crescem assustadoramente. Isto se deve ao fato deste protocolo ser de fácil implementação e manutenção, além de permitir a interligação de LAN através de outras WAN, com um desempenho considerável.

Atualmente é possível, inclusive, ter-se uma implementação de rede local utilizando o protocolo TCP/IP, sem conectá-la a outras redes, caracterizando uma rede Intranet, hoje muito utilizada na implementação de LAN em empresas ([ODA1998]).

Porém é necessário monitorar e controlar o funcionamento, crescimento e parâmetros dessas redes, garantindo o seu correto desempenho. A necessidade de gerenciamento de redes é evidente e tende a crescer à medida que as redes se tornam maiores e mais complexas. Como essas redes estão cada vez mais heterogêneas, uma padronização do protocolo de gerência garante que estas sejam gerenciadas de maneira uniforme.

Entre as atividades básicas do gerenciamento de redes estão a detecção e correção de falhas, em um tempo mínimo, e no estabelecimento de procedimentos para a previsão de problemas futuros. Por exemplo, monitorando linhas cujo tráfego esteja aumentando ou roteadores que estejam se sobrecarregando, é possível tomar medidas que evitem o colapso da rede, como a reconfiguração das rotas ou troca do roteador por um modelo mais adequado.

O protocolo de gerência SNMP (*Simple Network Management Protocol*) constitui atualmente um padrão operacional “de facto”, e grande parte do seu sucesso se deve a sua simplicidade. Outro aspecto importante é a sua capacidade de gerenciar redes heterogêneas constituídas de diferentes tecnologias, protocolos e sistemas operacionais. Dessa forma, o SNMP é capaz de gerenciar redes Ethernet, Token Ring que conectem PCs, Apple Machintosh, estações SUN e outros tipos de computadores ([COH2000]).

No entanto, o SNMP possui algumas limitações. A maior delas é o fato de não ser apropriado para o gerenciamento de redes muito grandes, devido à limitação de desempenho da estratégia de *pooling* utilizada pelo gerente SNMP na comunicação com os agentes SNMP ([CHI1999]). O *Simple Network Management Protocol* é um excelente

protocolo para gerenciar dispositivos em uma LAN. A maior parte das LAN disponibilizam a largura de banda necessária para que um gerente SNMP possa efetuar o *pooling* nos vários agentes da rede sem que isso afete significativamente o desempenho da mesma. Enquanto que numa WAN a largura de banda é massivamente utilizada, além de ser lenta.

Assim, o desenvolvimento de ferramentas de gerência que implementem o protocolo SNMP torna-se bastante atrativo. Como mostrado neste trabalho, hoje já contamos com plataformas, como o próprio JDMK, que permitem desenvolver essas ferramentas com uma maior independência do protocolo, tornando o processo de concepção menos complexo.

A plataforma JDMK em conjunto com a linguagem Java tornou-se possível criar um agente SNMP que possa operar em sistemas operacionais diferentes, como por exemplo, Unix e Windows NT, fazendo com que fossem alcançados os objetivos propostos. A linguagem Java tornou possível a idéia de criar um agente que tivesse características Web e pudesse ser acessado em qualquer local que se tenha acesso a Internet.

Umas das maiores dificuldades foi utilizar uma plataforma na qual, a mesma era recente no mercado, onde tinha-se pouca bibliografia e pouco se sabia dela, o JDMK ainda é novo no Brasil, sendo que, grande parte das dúvidas que vieram a aparecer, foram sanadas via correio eletrônico, por usuários, principalmente dos Estados Unidos e Alemanha, onde concentra-se grande parte dos desenvolvedores que utilizam esta plataforma.

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

Outra tendência do SNMP é sua migração para a Web, pois a conveniência do gerenciamento neste ambiente possui um apelo muito forte ([SUN1999]). Uma ferramenta de gerenciamento baseado em Web que utilizaria o navegador familiar ao usuário para apresentar as informações de gerenciamento, tornando a tarefa de gerência bem mais amigável do que uma linha de comando. Uma vez que esta ferramenta possa ser utilizada em qualquer PC que possua um navegador, as informações de gerenciamento passam a estar acessíveis de qualquer local. Esta portabilidade permite que apenas com um

navegador Web e um ponto de acesso remoto, o gerente possa monitorar e controlar a rede a qualquer hora, em qualquer lugar, enquanto viaja ou até mesmo de casa.

Por fim, devemos relembrar que com uma estrutura de gerenciamento adequada e a utilização das facilidades advindas disto, poderemos garantir o funcionamento adequado dos serviços de rede.

Para trabalhos futuros é interessante implementar mais recursos da MIB utilizada ou então utilizar a mesma plataforma JDMK para a criação de gerentes SNMP e ainda incrementar o gerenciamento HTML (MIB-*Browser*) para torná-lo uma aplicação mais amigável para a gerência.

7. BIBLIOGRAFIA

[CHI1999] CHISANE, Fabrício. **Gerenciamento de redes SNMP**. 02/2000. Endereço eletrônico: http://www.inf.ufgrs.br/~chisane

[COH2000] COHEN, Yoran. **SNMP simple network management protocol**. 02/2000. Endereço eletrônico: http://www-dos.uniinc.msk.ru/tech1/1995/snmp/snmp.htm

[FUR1998] FURLAN, José Davi. **Modelagem de objetos através da UML :** the unified modeling language. São Paulo : Makron Books, 1998.

[LIU1999] LIU, Narayanan J. **Enterprise Java Developer´s Guide**. United States of America : McGraw-Hill, 1999.

[MAF1999] MAFINSKI, André. **Protótipo de software de gerência SNMP para o ambiente Windows NT**. Universidade Regional de Blumenau : Trabalho de Conclusão de Curso de Ciências da Computação - Bacharelado, 1999.

[NAU1996] NAUGHTON, Patrick. **Dominando o java.** São Paulo : Makron Books, 1996.

[NET1990] NETO, Vicente. **Redes de dados, teleprocessamento e gerência de redes**. São Paulo : Érica, 1990.

[ODA1998] ODA, Cybelle Suemi. Gerenciamento de redes de computadores. Endereço Eletrônico: http://www.gt-er.cg.org.br/operacoes/gerencia-redes

[SOA1995] SOARES, Luiz Fernando G. **Redes de computadores : LANs, MANs e WANs às redes ATM**. Rio de Janeiro : Campus, 1995.

[STA1996] STALLINGS, W. **SNMP, SNMP v2 and RMON.** First Edition. Addison Wesley, 1996.

[SUN1999] SUN, Microsystems. **Java Dynamic Management Kit 4.0 Tutorial**. 02/2000. Endereço eletrônico: http://docs.sun.com

[TAN1996] TANEMBAUM, A. S. **Computer networks.** Third Edition. Prentice Hall, 1996.

[VOG1999] VOGEL, Andreas. **Programming with Enterprise JavaBeans**. New York : Wiley, 1999.

[WAL1997] WALNUM, Clayton. **Java em exemplos**. São Paulo : Makron Books, 1997.

[WIN1993] WINBLAD, Ann L et al. **Software orientado ao objeto**. São Paulo : Makron Books, 1993.

8. ANEXOS

8.1 ANEXO 1

Código fonte do Agente em Java.

```
/*
 * @(#)Arquivo      Agente.java
 * @(#)Autor        Jorge Lucas
 * @(#)Data         06/2000
 *
 */

import javax.management.ObjectName;
import javax.management.MBeanServer;
import javax.management.MBeanServerFactory;

import com.sun.jdmk.Trace;
import com.sun.jdmk.comm.HtmlAdaptorServer;
import com.sun.jdmk.comm.SnmpAdaptorServer;


public class Agente {

    static SnmpAdaptorServer snmpAdaptor = null;

    public static void main(String args[]) {

        MBeanServer server;
        ObjectName htmlObjName;
        ObjectName snmpObjName;
        ObjectName mibObjName;
        ObjectName trapGeneratorObjName;
        int htmlPort = 8082;
        int snmpPort = 161;


        try {
            Trace.parseTraceProperties();
            Trace.send(Trace.LEVEL_TRACE, Trace.INFO_MISC, "Agente",
"main", "Caminho padrao");
        } catch(java.io.IOException e) {
            e.printStackTrace();
        }

        try {
            server = MBeanServerFactory.createMBeanServer();
            String domain = server.getDefaultDomain();

            // Cria e inicia o adaptador HTML.
            //
```

```
            htmlObjName = new ObjectName(domain +
":class=HtmlAdaptorServer,protocol=html,port=" + htmlPort);
            Trace.send(Trace.LEVEL_TRACE, Trace.INFO_MISC, "Agente",
"main", "Adicionando adaptador HTML para o servidor MBean como o nomme
\n\t" + htmlObjName);
            java.lang.System.out.println("NOTA: Adaptador HTML esta
conectado na porta TCP" + htmlPort);
            HtmlAdaptorServer htmlAdaptor = new
HtmlAdaptorServer(htmlPort);
            server.registerMBean(htmlAdaptor, htmlObjName);
            htmlAdaptor.start();

            //
            // Codigo especifico para o SNMP:
            //

            // Cria e inicia o adaptador SNMP.
            // Especifica a porta de uso na Constructor.
            // Caso queria utiliar a porta(161) comentar a proxima linha
            // snmpPort = 8085;
            //
            snmpPort = 8085;
            snmpObjName = new ObjectName(domain +
":class=SnmpAdaptorServer,protocol=snmp,port=" + snmpPort);
            Trace.send(Trace.LEVEL_TRACE, Trace.INFO_MISC, "Agente",
"main", "Adicionando adaptador SNMP para o servidor MBean como o nomme
\n\t" + snmpObjName);
            java.lang.System.out.println("NOTa: Adaptador SNMP esta
conectado na porta UDP " + snmpPort);
            snmpAdaptor = new SnmpAdaptorServer(snmpPort);
            server.registerMBean(snmpAdaptor, snmpObjName);
            snmpAdaptor.start();

            // Inicializa a porta trap SNMP.
            // Utiliza a porta = snmpPort+1.
            //
            java.lang.System.out.print("NOTa: Enviando um trap SNMP para
cada destino definido no arquivo ACL ...");
            snmpAdaptor.setTrapPort(new Integer(snmpPort+1));
            snmpAdaptor.sendV1Trap(0, 0, null);
            java.lang.System.out.println("Done.");

            // Cria a MIBCreate II (RFC 1213) e adiciona no Servidor
MBean.
            //
            mibObjName= new ObjectName("snmp:class=RFC1213_MIB");
            Trace.send(Trace.LEVEL_TRACE, Trace.INFO_MISC, "Agente",
"main", "Adicionando RFC1213-MIB para o Servidor MBean como o nome\n\t" +
mibObjName);
            RFC1213_MIB mib2 = new RFC1213_MIB();
            server.registerMBean(mib2, mibObjName);

            // Conecta o adaptador SNMP na MIB para fazer o acesso da MIB
através adaptador do protocolo SNMP.
            // Se o passo nao acontecer, a MIB ainda estara ativa no
Agente JDMK:
            // Os objetos serao acessados atraves do HTML e nao SNMP.
```

```
            //
            mib2.setSnmpAdaptorName(snmpObjName);

            // Cria um LinkTrapGenerator.
            // Especifica o ifIndex para usar no object name.
            //
            String trapGeneratorClass = "LinkTrapGenerator";
            int ifIndex = 1;
            trapGeneratorObjName = new ObjectName("trapGenerator" +
":class=" + trapGeneratorClass + ",ifIndex=" + ifIndex);
            Trace.send(Trace.LEVEL_TRACE, Trace.INFO_MISC, "Agente",
"main",
                       "Adicionando " + trapGeneratorClass + " para o
Servidor MBean com o nome \n\t" + trapGeneratorObjName);
            server.createMBean(trapGeneratorClass, trapGeneratorObjName);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    static public SnmpAdaptorServer getSnmpAdaptor() {
        return snmpAdaptor;
    }
}
```

8.2 ANEXO 2

Código fonte do arquivo de permissão ACL

```
#
# @(#)jdmk.acl
#
#
# accesso: pode ter acesso somente para "read-only" ou "read-write"
#
# gerents que podem acessar:
#    - hostname: bla bla bla
#    - Endereco IP: 123.456.789.12
#    - Mascara de Sub-rede: 123.255.255.255
#


acl = {
 {
 communities = public, private
 access = read-only
 managers = 100.100.100.10
 }
 {
 communities = public
 access = read-write
```

```
 managers = 100.100.100.10
 }
}

trap = {
  {
  trap-community = public
  hosts = 100.100.100.10
  }
  {
  trap-community = private
  hosts = 100.100.100.10
  }
}
```