UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Eduardo Coelho Cerqueira

IMPLEMENTAÇÃO DE UM SOFTWARE DISCRIMINADOR DE REPASSE DE EVENTOS PARA A ARQUITETURA INTERNET

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dra. Elizabeth Sueli Specialski

Florianópolis, fevereiro de 2003

IMPLEMENTAÇÃO DE UM SOFTWARE DISCRIMINADOR DE REPASSE DE EVENTOS PARA A ARQUITETURA INTERNET

Eduardo Coelho Cerqueira

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Dr. Fernando A. O. Gauthier
Coordenador do Curso

Banca Examinadora

_____

Prof. Dra. Elizabeth S. Specialski
Orientadora

_____

Prof. Dr. João Bosco da Mota Alves

_____

Prof. Dra. Silvia Modesto Nassar

A meus pais Luciano e Ana e
meu irmão Leonardo

AGRADECIMENTOS

A Deus.

A Nossa Senhora de Nazaré (padroeira dos paraenses) por todas as realizações na minha vida.

Ao meu amado pai e irmão e a minha amada mãe pelos esforços financeiros, companheirismo, conselhos, confiança e amizade.

As tias, tios e primos que sempre me apoiaram, em especial para minha querida Rosinha, Teté, Celinha, Marcinha, Azize, Eliana, Bassalo, Ronaldo, Claudão, Inocêncio, Antero, Jô e Pedro.

À minha orientadora, Profª Drª Elizabeth Sueli Specialski, pelo excelente trabalho realizado, pelo extremo profissionalismo e capacidade de trabalho, por todos os conselhos que muito ajudaram na minha vida.

Ao meu grande mestre Prof. Dr. Bosco da Mota Alves e a minha querida Profª Drª Silvia Modesto Nassar, que sempre me ajudaram na minha formação acadêmica e pessoal.

Agradeço também a todos os paraenses residentes em Florianópolis, pelo companheirismo e amizade.

Ao meu grande amigo e irmão Neto que sempre foi meu companheiro, para todas as horas.

Aos amigos e irmãos do coração Peri e Gordinho por sempre me acolherem e dividirem comigo momentos maravilhosos da minha vida.

Aos amigos do coração, Iuri, Suzana, Hugo, Helaine, Jana, Hilton, Serginho, Mateus, Muchiba, Draka, Rhoden, Hélio, Samarone, Cabeça, Feio, Fabinho, Marquinho, Magá, Evaristo, Jaime, Tércio, Jailson, Negão, João Lord, Paulo Gatuno e muitos outros mais que não consigo citar agora.

A todos os amigos e funcionário da UFSC e NPD, pela acolhida e informações passadas, em especial a Edson Melo, Kathia Juca, Gerson, Júlio, Vinícius e Richard.

A todos os amigos e alunos da Fesurv, especialmente meus “filhos” Fernando e Hugo, ao grande Maxwell, o AAJ Cresio11, Goiatuba, Rogério e muitos outros.

SUMÁRIO

Excluído: xii
Excluído: xiii
Excluído: xiv
Excluído: xv

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Abreviaturas

ASN.1 - *Abstract Syntax Notation. One*

BER - *Basic Encoding Rules*

CCITT - *International Consultative Committee on Telegraphy and Telephony*

CMIP – *Common Management Information Protocol*

CMOT – CMIP *over* TCP/IP

EGP – *External Gateway Protocol*

IAB - *Internet Architecture Board*

ICMP - *Internet Control Message Protocol*

IEC - *International Electrotechnical Commission*

IEEE – *Institute of Electrical and Electronics Engineers*

IESG - *Internet Engineering Steering Group*

IETF – *Internet Engineering Task Force*

IP – *Internet Protocol*

ISO - *International Organization for Standardization*

ISOC - *Internet Society*

MIB - *Management Information Base*

NMS - *Network Management Station*

OID – *Object Identifier*

OSI – *Open System Interconnection*

PDU – *Protocol Data Unit*

PING - *Packet Internet Groper*

RFC – *Request for Comments*

RMON – *Remote Network Monitoring*

SGMP - *Simple Gateway Monitoring Protocol*

SMI – *Structure of Management Information*

SMP – *Simple Management Protocol*

SNMP – *Simple Network Management Protocol*

TCP/IP - *Transmission Control Protocol / Internet Protocol*

UDP – *User Datagram Protocol*

RESUMO

Este trabalho apresenta um *Software* Discriminador de Repasse de Eventos (SDRE) construído em Java, aplicando as funcionalidades do modelo OSI em ambientes SNMP. É feita uma revisão bibliográfica das duas arquiteturas de gerência de redes, buscando adequar a norma OSI/ITU-T X-734 à Internet e são apresentadas motivações que demonstram esforços atuais nesta área. Esta aplicação é simples e funcional, enfatizando a técnica de relatório de eventos, filtrando os mesmos e podendo selecionar várias estações de gerenciamento destinatárias, como forma de descentralizar e melhorar o desempenho da arquitetura de gerência de redes Internet. Por fim são apresentados os resultados dos testes efetuados, a avaliação e a validação do *software*, bem como o ambiente simulado e de produção onde foi implantado o discriminador.

ABSTRACT

This research presents a Event Report Discriminator Software (SDRE) constructed in Java, applying the functionalities of OSI model in environments SNMP. A bibliographical revision of the two architectures of management network is made, having searched to adjust the OSI/ITU-T recommendation X-734 into to Internet and the motivations demonstrate the current efforts in this area. This application is simple and functional, besides emphasizing the technique of event report, filtering and being able to select some management stations as final destination, as form to decentralize and to improve the performance of the Internet network management architecture. Finally the results of the effected tests, the evaluation and the validation of software are presented, as well as the simulated and production environment where the discriminator was implanted.

1. Introdução

Com o aumento da complexidade, quantidade de computadores, surgimento de novos dispositivos, serviços e da importância da comunicação de dados em redes de computadores, tornou-se necessário o gerenciamento das mesmas. Anos atrás, tinha-se ambientes com poucas máquinas onde era possível ao administrador gerenciá-las somente com a sua experiência, sem a utilização de recursos para o controle e monitoração do ambiente.

Atualmente, as redes de computadores podem possuir diversos dispositivos (*switches*, *hubs*, roteadores, servidores, entre outros) que são indispensáveis para a comunicação de informações nas organizações que precisam estar não somente em execução, mas funcionando perfeitamente para garantir aos seus usuários a maior disponibilidade possível dos serviços oferecidos (DOUGLAS & SCHMIDT, 2001).

O mau funcionamento dos recursos, causado por uma má administração, pode acarretar prejuízos enormes para as organizações, diminuindo os benefícios providos pela rede. Um ponto importante é o fato da sub ou super-utilização dos equipamentos por falta de conhecimento do administrador das informações que trafegam na rede. Por exemplo, um *switch* pode estar sobrecarregado enquanto outro encontra-se sub-utilizado, sendo que o mesmo poderia ser usado para balancear o tráfego.

Devido aos problemas encontrados, referentes à ausência do gerenciamento da rede, se percebeu, então, a necessidade de ferramentas e dispositivos para auxiliar nesta tarefa. Com isso, foram criados os sistemas de gerência de rede, que ajudam administradores a monitorar e controlar seus ambientes.

Os sistemas de gerenciamento foram criados inicialmente por fabricantes que desenvolviam seus *softwares* e *hardware* proprietários, onde as comunicações somente eram possíveis entre equipamentos do mesmo fornecedor. Com isso, as organizações eram forçadas a adquirir componentes de um mesmo fabricante, para garantir a interoperabilidade entre eles. (VENÂNCIO NETO, 2001).

Na tentativa de solucionar estas dificuldades, iniciaram-se pesquisas para a criação de arquiteturas padronizadas para o gerenciamento de rede, tornando possível a comunicação entre equipamentos de diferentes fabricantes (independente do fornecedor).

As arquiteturas de redes mais difundidas são a Internet e a OSI (*Organization System Interconnection*) da ISO (*International Organization for Standardization*). A primeira, atualmente mais utilizada, padronizou o SNMP (*Simple Network Management Protocol*) como protocolo de gerenciamento, já a segunda, mais complexa e pouco usada, utiliza o protocolo CMIP (*Common Management Information Protocol*) para a troca de informações entre os elementos envolvidos no gerenciamento (STALLINGS, 1999).

Alguns conceitos importantes na arquitetura SNMP são: gerente (NMS – *Network Management Station*/estação de gerenciamento de rede), agente e base de informação de gerenciamento (MIB - *Management Information Base*).

O modelo SNMP é utilizado basicamente para a monitoração da rede, dando escassa ênfase ao controle (por motivos de segurança) e ao envio de *traps* (notificações assíncronas enviadas pelo agente quando algum limite pré-estabelecido é ultrapassado). As informações são coletadas em uma MIB ligada ao agente e representam o estado dos objetos gerenciados (STALLINGS, 1999).

O SNMP é tradicionalmente utilizado de forma centralizada, onde geralmente um gerente é responsável por coletar, modificar e analisar informações sobre os dispositivos gerenciados. Já o CMIP é considerado distribuído, pois comumente adota-se mais de uma estação de gerenciamento para realizar essas tarefas (VENÂNCIO NETO, 2001). Além do mais, uma quantidade reduzida de gerentes pode ser destinatário dos *traps* emitidos pelo agente, onde o mesmo alarme é recebido por todas as NMS, que ficam responsáveis pelas ações a serem tomadas, aumentando o *overhead* na rede e o processamento da estação de gerenciamento. As informações são redundantes, por exemplo, a notificação referente ao desempenho deveria ser enviada somente ao(s) gerente(s) responsável(is) por aquela função e não para todos os outros. A Fig. 1.1 ilustra um cenário onde um mesmo *trap* é enviado para todas as NMS configuradas no agente.

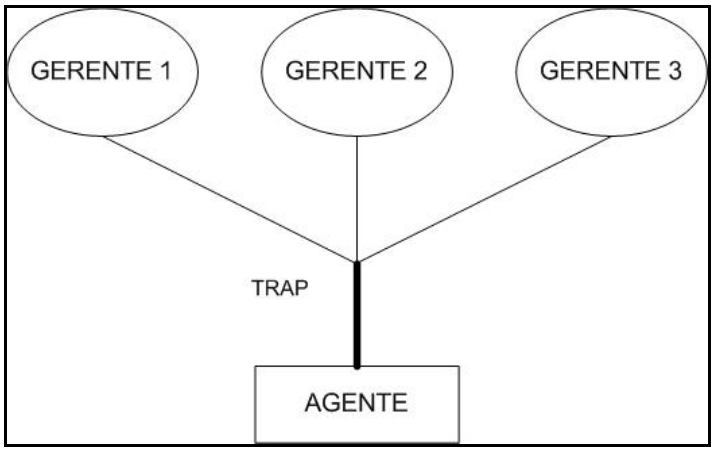

Figura 1.1 - Cenário de envio de *trap*s para todas as NMS

Em ambientes com uma quantidade razoável de dispositivos e baixo fluxo de dados, esta arquitetura teria um grau de eficiência satisfatória. Entretanto, em um ambiente com tráfego excessivo de pacotes, grande quantidade de *hosts*, dispositivos de interconexão e serviços, teria seu grau de vulnerabilidade exponencialmente maior, devido a diversos fatores, como violação de dados, contaminação por vírus, acessos a *sites* indesejados, entre outros. Dessa forma, um NMS não seria capaz de solucionar problemas em tempo hábil, comprometendo a integridade do ambiente.

Uma solução para este problema seria a adoção de mais de um gerente e distribuí-los em máquinas diferentes, concedendo mais desempenho, autonomia e independência a cada um, além de atividades de gerência simultânea e pró-ativa. Disso vê-se que, enquanto uma NMS soluciona um problema de gargalo, outra poderia estar detectando a ocorrência de tentativas de invasão.

Entretanto, como característica atual do SNMP, todos os NMS são destinatários de todos os tipos de *traps*, aumentando drasticamente o consumo da largura de banda da rede e *overhead* nas estações, já que alguns gerentes recebem alarmes inúteis para as suas tarefas de gerenciamento. Para solucionar este problema, algum esquema de filtragem de relatório de eventos deve ser adicionado ao ambiente, evitando que os *traps* sejam enviados indiscriminadamente para todos os gerentes. Assim, cada *trap* poderia ser encaminhado ao(s) gerente(s), de acordo com seu(s) domínio(s) de gerência, proporcionando um melhor aproveitamento da banda disponível. Por exemplo, a notificação referente à segurança da rede seria enviada somente para a NMS responsável por esta tarefa.

No gerenciamento de redes OSI, este problema é solucionado com eficácia através da utilização de um objeto denominado Discriminador de Repasse de Eventos[1] (VENÂNCIO NETO, 2001). Neste contexto, um elemento com características semelhantes deve ser incorporado a ambientes gerenciados através do SNMP, com modificações suficientes para adequar-se às características específicas deste protocolo[2].

Com isso, as taxas de tráfego de pacotes na rede e a carga de trabalho em cada estação de gerenciamento são diminuídas, já que cada NMS deverá receber apenas *traps* selecionados de acordo com seu domínio de gerência. A Fig. 1.2 apresenta um ambiente onde o agente envia um alarme para o dispositivo (discriminador) capaz de realizar a filtragem e escolher o(s) destinatário(s) para o(s) *trap*(s).

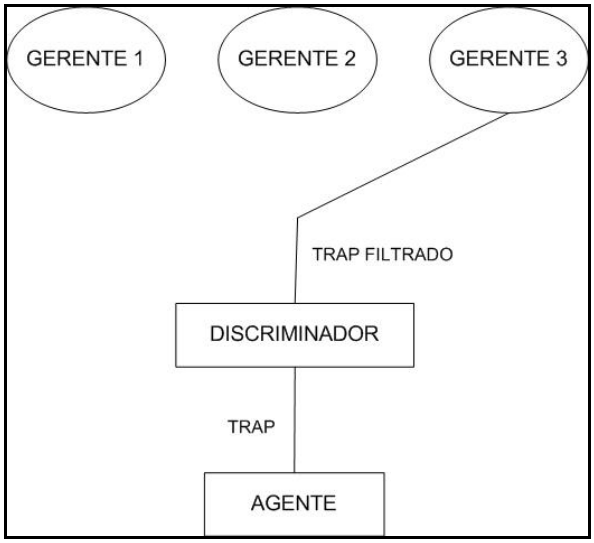

Figura 1.1 - A utilização de um discriminador para filtrar *traps* e selecionar destinatários

1.1. Motivação

O entusiasmo para esse estudo surgiu devido aos problemas encontrados no modelo atual do SNMP, descritos na seção anterior, referentes a ambientes de grande porte gerenciados de forma centralizada.

Alguns autores como FERIDUN et al (1999), ZAPF et al (1999), LI et al (2001), VENÂNCIO NETO et al (2002) realizaram pesquisas com intuito de descentralizar o modelo SNMP no sentido de melhorar o desempenho, a escalabilidade, a eficiência da rede e de outras funcionalidades importantes. Os três primeiros autores utilizaram

[1] Apresentado no capítulo 4
[2] Detalhadamente descrito no capítulo 3

Agentes Móveis e Java e o último enfatiza a técnica de relatório de eventos e Java para conseguir o objetivo. Todos os estudos obtiveram resultados satisfatórios, conseguindo uma forma de descentralizar a arquitetura SNMP.

Além destes, grandes empresas como CS CARE, HP, IBM e SUN também reconheceram os benefícios da ênfase à técnica de relatório de eventos, disponibilizando no mercado versões de seus *softwares* de gerenciamento já com esta funcionalidade incorporada. Contudo, estes são executados no NMS e, desta forma, não existindo o impedimento da recepção dos *traps*, ou seja, uma maneira para que a seleção das mensagens seja feita antes de chegar ao gerente final. Isto contribui para o aumento de pacotes na rede e para o consumo de recursos da estação de gerenciamento. Existe também um custo para aquisição dos *softwares*. Seus produtos estão brevemente descritos na seção seguinte.

1.2. Trabalhos Correlatos

Na tentativa de obter um melhor embasamento foram pesquisadas soluções existentes que possuam um sistema de tratamento de eventos no modelo SNMP. Entre as encontradas, se destacam: HP Open View, Sun Net Manager, Implementação de um Discriminador de Repasse de Eventos para o Ambiente SNMP e *Trap* Console.

1.2.1. HP Open View

Este *software* de gerenciamento de rede é produzido pela Hewlett Packard e de acordo com (DOUGLAS & SCHMIDT, 2001) utiliza o *deamon ovtrapd* para tratamento dos *traps* recebidos. Sendo possível configurar eventos a partir de interfaces gráficas para executar ações que variam desde mostrar uma mensagem no monitor do NMS, até ignorar as notificações recebidas ou enviar o evento para outras NMS que possuem o *Open View*.

1.2.2. Sun Net Manager

Desenvolvido pela Sun Microsystems, atualmente este *software* de gerência de rede encontra-se na versão 2.3. Possui uma ferramenta para tratamento dos *traps* recebido pela NMS e, para isso, utiliza o *trap deamon*, que traduz os relatórios de eventos recebidos para o formato *SunNet Manager Trap*. Com isso, é capaz determinar se o alarme deve ou não ser descartado e especifica um tipo de prioridade para o repasse

(*hight, medium e low*) para, posteriormente, encaminhar o *trap* para uma ou mais NMS (SUN, 2002).

1.2.3. *Trap* Console

É desenvolvido pela CS Care Inc, encontra-se na versão 1.4, utilizou-se Java em sua implementação (possibilitando "rodar" independente de plataforma), requer um espaço em disco de aproximadamente 1 (um) MB e precisa do JVM (*Java Virtual Machine*) presente no equipamento.

O *Trap* Console é uma aplicação servidora para o gerenciamento de *traps* SNMPv1 e SNMPv2, tornando possível o manuseio de relatório de eventos encaminhados pelos agentes. Pode ser usado em conjunto com outras aplicações de gerenciamento de redes, sendo capaz de selecionar (filtrar) os *traps*, de especificar destinatários, de notificar via e-mail ou *sockets*, entre outras, com isso, diminuindo o tráfego de informações.

1.2.4. Implementação de um Discriminador de Repasse de Eventos para o Ambiente SNMP

Desenvolvido como um protótipo resultante de um trabalho de dissertação de mestrado de Venâncio Neto (VENÂNCIO NETO, 2001), possibilita a existência de mais de um gerente na rede, além de evitar que os *traps* emitidos pelos agentes sejam encaminhados para todos os gerentes indiscriminadamente, sem a necessidade de alteração no código do agente.

Neste trabalho, o objeto discriminador fica localizado entre o agente e o gerente. Isso proporciona uma forma transparente para os elementos envolvidos. O protótipo foi desenvolvido usando a linguagem Java e uma API gratuita desenvolvida pela AdventNet. Porém há limitações em relação à inexistência de uma interface gráfica, de uma base de dados, de um método eficaz de busca e re-direcionamento dos *traps*. Não aceita solicitações simultâneas (*multithreads*), não possui flexibilidade para a inserção, alteração ou exclusão de gerentes e dos *traps*.

1.3. Objetivos

1.3.1. Objetivo Geral

O objetivo geral dessa dissertação é a criação, implementação e validação de um *Software* Discriminador de Repasse de Eventos (SDRE) com o intuito de descentralizar a arquitetura de gerenciamento de redes IP, enfatizando a técnica de recepção de relatórios de eventos. O SDRE receberá as notificações, depois fará uma filtragem das mensagens definindo a ação a ser tomada de acordo com uma base de dados e o envio ou não do(s) *trap*(*s*) a gerente(s) pré-definido(s) de acordo com as áreas funcionais de gerência definidas pela ISO (desempenho, falha, configuração, segurança e contabilização).

1.3.2. Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Construir um protótipo com as funcionalidades do Discriminador de Repasse de Eventos OSI adequadas às necessidades dos ambientes SNMP;
- Desenvolver uma base de regras e método de busca eficaz;
- Avaliar o desempenho do Software Discriminador de Repasse de Eventos, através de um teste de carga;
- Implementar interfaces gráficas simples e funcionais para configuração da ferramenta, como inclusão de regras e novos gerentes, bem como exclusão e consulta destas.

1.4. Estrutura do Trabalho

O presente trabalho está organizado em sete capítulos. O primeiro capítulo descreve uma introdução ao tema, a motivação que levou à elaboração da pesquisa, os objetivos gerais e específicos e trabalhos correlatos. Já o segundo capítulo, apresenta as cinco áreas funcionais do gerenciamento OSI. O terceiro capítulo descreve a arquitetura de gerenciamento de redes internet, onde está detalhado o conceito, os objetivos, os componentes, a monitoração, o controle, as versões existentes do protocolo e como são representados, a MIB e SMI (*Structure of Management Information*), a monitoração remota (RMON), as mensagens e operações SNMP, como funciona o gerenciamento centralizado, a questão da segurança e interoperabilidade e os *softwares* de gerência de

rede. O quarto capítulo aborda considerações sobre o discriminador de repasse de eventos. O quinto capítulo, mostra a solução proposta, bem como a arquitetura interna e externa do *Software* Discriminador de Repasse de Eventos, sua implementação, validação e resultados. O sexto capítulo apresenta a conclusão e sugestões para trabalhos futuros. No último capítulo são discriminadas as referências bibliográficas e a bibliografia consultada. O apêndice I possui algumas versões das RFCs (*Request for Comments*) relacionadas com o SNMP e o apêndice II a especificação formal das unidades de dados de protocolo (PDUs) em ASN.1 do SNMPv1. Por último, o apêndice III mostra os *traps* enviados pelos dispositivos gerenciáveis e discriminados pelo SDRE.

2. Áreas funcionais do Gerenciamento OSI

O gerenciamento de redes de computadores é de vital importância para as organizações que utilizam redes, pois através de seu uso é possível obter maior disponibilidade, desempenho, eficiência, segurança e satisfação dos usuários. Um problema encontrado, atualmente, se refere ao uso de ferramentas de *software/hardware* proprietários que dificultam esse gerenciamento e apresentam um custo elevado, sujeitando as organizações e os administradores de redes a ficarem atrelados somente aos equipamentos e serviços fornecidos pelos fabricantes, dos quais inicialmente foram adquiridos, conforme se refere abaixo:

Com o intuito de resolver problemas de interoperabilidades, foram desenvolvidos padrões de gerenciamento que permitem aos sistemas computacionais e equipamentos de redes se comunicarem, independentemente de fabricante. As atividades de gerenciamento foram classificadas em cinco áreas específicas, conhecidas coletivamente como Áreas Funcionais de Gerência, e consistindo no Gerenciamento de Falhas; Gerenciamento de Desempenho; Gerenciamento de Configuração; Gerenciamento de Contabilização e Gerenciamento de Segurança (SPECIALSKI, 2002, VERONEZ, 2000).

2.1. Gerenciamento de Configuração

O Gerenciamento de Configuração envolve manutenção e monitoração da estrutura física e lógica da rede. Corresponde ao conjunto de processos que exercem o controle sobre os objetos gerenciados, identificando-os, coletando e fornecendo dados sobre os mesmos a fim de dar suporte a funções, que segundo VERONEZ (2000) são:

- Atribuir valores iniciais aos parâmetros do sistema;
- Iniciar e terminar operações sobre objetos gerenciáveis;
- Alterar e configurar o sistema;
- Associar nomes a conjuntos de objetos gerenciados.

Nesse tipo de gerenciamento alguns recursos (servidores, *hubs, switches*) podem ser usados com diferentes funções (por exemplo, um computador pode atuar tanto como estação de trabalho quanto *gateway*) e serviços (um mesmo equipamento pode prover diferentes serviços). Isso fica na responsabilidade do administrador da rede, portanto a escolha dos recursos, das funções e dos serviços, dependendo de sua necessidade.

2.2. Gerenciamento de Falhas

O Gerenciamento de Falhas é responsável pela manutenção e monitoração do estado de cada um dos objetos gerenciados e pelas ações necessárias ao restabelecimento ou isolamento das unidades com problemas. As informações coletadas podem ser usadas para indicar os elementos de redes que estão funcionando, os que operam precariamente ou permanecem fora de operação (VERONEZ, 2000).

É importante ressaltar que falha não é o mesmo que erro. Falha é uma condição anormal cuja recuperação exige ação de gerenciamento (SPECIALSKI, 2002). Consiste na detecção de um problema, sua solução e posterior correção. Uma alternativa para reduzir o prejuízo com a falha de algum equipamento ou dispositivo é o uso de *hardware* ou *software* de tolerância a falhas. O Gerenciamento de Falhas provê facilidades para o Gerenciamento de Desempenho (LOUREIRO et al., 2002). De acordo com STALLINGS (1999), deve estar preparado para:

- Determinar o componente exato que falhou;
- Isolar o resto da rede da falha, sem prejudicar o funcionamento;
- Reconfigurar ou modificar a rede para minimizar o impacto da operação sem o componente que falhou;
- Reparar ou trocar o componente com problemas para restaurar a rede ao seu estado anterior.

2.3. Gerenciamento de Desempenho

O Gerenciamento de Desempenho envolve a coleta e interpretações das medições periódicas dos indicadores de desempenho, identificando gargalos, verificando se o tráfego é excessivo e fazendo prognóstico do desempenho futuro da rede (VERONEZ, 2000).

A função do gerenciamento de desempenho é coletar e analisar dados estatísticos com o intuito de monitorar e corrigir o comportamento e o melhoramento da rede (por exemplo, distribuindo a quantidade de tráfego por diferentes *switches*), dos elementos da rede ou outros equipamentos, e auxiliar no planejamento, manutenção e medida de qualidade (LOUREIRO et al., 2002).

Para solucionar problemas referentes ao desempenho, o gerente precisa associar métricas e valores apropriados aos recursos de rede, para que possam fornecer

indicadores de diferentes níveis de desempenho (por exemplo, analisar as informações sobre a quantidade de pacotes que chegam ou saem de um equipamento, com isso determinando horários de picos existente na rede). Com essas informações o administrador pode ter um controle satisfatório sobre o funcionamento da rede. Outro ponto importante é a documentação da configuração da rede, pois permite que o administrador tenha um maior número de opções na solução de problemas.

2.4. Gerenciamento de Contabilização

O Gerenciamento de Contabilização é responsável pela manutenção e monitoração dos custos dos recursos consumidos na rede, estabelecendo métricas, quotas e podendo gerar tarifas para cobrança (por exemplo, a utilização da impressora pelo usuário X implicará cobrança de um valor referente ao uso da mesma).

As informações coletadas pelo gerenciamento de Contabilização podem auxiliar na melhoria do desempenho e no melhor aproveitamento dos elementos da rede, evitando uma super ou sub-utilização de recursos e equipamentos. (SPECIALSKI, 2002).

Outra utilidade do Gerenciamento de Contabilidade é determinar se a utilização dos recursos da rede está aumentando rapidamente com o crescimento, caso afirmativo, deve-se iniciar a tomada de ações ou rever as políticas adotadas.

2.5. Gerenciamento de Segurança

LOUREIRO et al. (2002) relata que o Gerenciamento de Segurança trata da proteção das informações estratégicas da rede, procurando agregar aos dispositivos de acesso ao sistema, controles de acesso aos usuários e notificando possíveis problemas de segurança. Segundo SPECIALSKI (2002), isso pode envolver:

- A geração, a distribuição e o armazenamento de chaves de criptografia;
- Manutenção e distribuição de senhas e informações de controle de acesso;
- Monitoração e controle de acesso de toda ou parte da rede e às informações obtidas dos nodos;
- Coleta, armazenamento e exame de registros de auditoria e *logs* de segurança.

2.6. Considerações Finais

Na maioria das redes, as cinco áreas funcionais de gerenciamento têm sido aplicadas na gerência reativa, emitindo alarmes e eventos, somente depois de apresentar problemas. Porém isso está propenso a mudanças, considerando que já existem trabalhos e ferramentas onde é aplicada a gerência pró-ativa para realizar o gerenciamento das redes de computadores. Situações que caracterizam estados que possam vir a degradar os serviços oferecidos, podem ser identificadas a fim de evitar que ocorram problemas mais graves. Por exemplo, quando a capacidade de armazenamento do disco rígido do servidor de *e-mail* alcançar um limite de 80 %, enviar uma mensagem ao responsável pelo suporte da rede para trocar ou expandir essa capacidade antes que prejudique o funcionamento do mesmo. (CLEMENTI & CARVALHO, 1999).

No entanto, esta gerência pró-ativa requer a utilização de mensagens geradas pelos agentes a fim de informar aos gerentes sobre a ocorrência de eventos. No próximo capítulo apresenta-se os fundamentos da arquitetura de gerenciamento SNMP, onde são evidenciadas as principais dificuldades para a adoção da pró-atividade.

3. A Arquitetura de Gerenciamento de Redes Internet – SNMP

Este capítulo trata da Arquitetura de Gerenciamento de Redes Internet, e será mostrado com detalhes o protocolo SNMP, incluindo seus conceitos, objetivos da arquitetura, componentes básicos, funções de gerenciamento (monitoração e controle), como os dados são representados, versões existentes do protocolo, estrutura de informações de gerenciamento (SMI), base de informações de gerenciamento (MIB), monitoração remota (RMON), mensagens e operações suportadas pelo protocolo e a interoperabilidade entre as duas primeiras versões do protocolo (SNMPv1 e SNMPv2).

Desde que foi desenvolvido em 1988, o SNMP tornou-se o padrão de gerenciamento adotado para atender à necessidade cada vez maior de administradores e fabricantes de equipamentos de interconectividade para redes IP, pois é uma solução simples que requer um código pequeno para a implementação, e os fabricantes podem, facilmente, adequá-lo em seus produtos (STALLINGS,1999).

O protocolo SNMP atua no nível de aplicação da pilha TCP/IP e usa o UDP (*User Datagram Protocol*) como protocolo de transporte para passagem de dados entre os elementos envolvidos. Não é orientado a conexão, ou seja, nenhuma conexão é estabelecida entre os mesmos. É baseado no modelo agente/gerente e é conhecido como simples, pois o agente necessita de um *software* mínimo. A maioria do esforço computacional fica por conta do gerente. O SNMP habilita administradores de rede a gerenciar a performance da rede, encontrar e solucionar problemas e planejar o crescimento de maneira eficaz (VENÂNCIO NETO, 2001).

Os comandos para tomada de alguma ação ou somente monitoração são enviados do gerente para o agente. O agente pode estar localizado em *hubs*, *switchs*, *gateways*, *host*s. Resumindo, um agente pode ser instalado em qualquer dispositivo capaz de suportar o SNMP. Quando a solicitação chega ao dispositivo gerenciado, este responde para o gerente o que foi solicitado ou em alguns casos, pode retornar mensagens de erro. O agente também é capaz de enviar informação não solicitada (*traps*) para a NMS, se algum limite pré-definido for excedido. A Fig. 3.1**,** mostra como é feita a comunicação e troca de mensagem entre as entidades gerente e agente no SNMP.

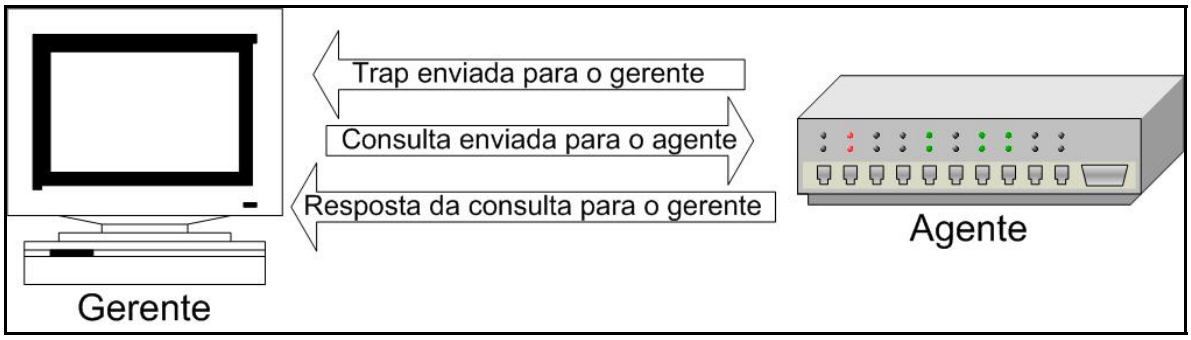

Figura 3.1 - Troca de mensagem entre gerente e agente SNMP

O protocolo SNMP é encontrado em três versões: SNMPv1 (primeira versão), SNMPv2 (segunda versão) e SNMPv3 (terceira versão). Cada uma com suas funções e funcionalidades[3].

3.1. Finalidades da Arquitetura

Segundo BATES (2002), "o objetivo de uma arquitetura de gerenciamento é deixar o sistema computacional o mais disponível possível". O modelo SNMP tenta minimizar o número e a complexidade de funções realizadas pelos agentes gerenciados. Os quatros motivos que tornam esta arquitetura atraente são (CASE, 1990):

1. O custo de desenvolvimento do *software* agente necessário para suportar o protocolo é relativamente reduzido;
2. O grau de funcionalidade suportado remotamente é proporcionalmente aumentado, à medida que se eleva o uso dos recursos internet na tarefa de gerenciamento;
3. A quantidade de funções de gerenciamento, que são suportadas remotamente, é proporcionalmente aumentada, à medida que são impostas algumas restrições sobre forma e sofisticação de ferramentas de gerência;
4. Conjuntos simplificados de funções de gerenciamento são facilmente entendidos e usados pelos desenvolvedores de ferramentas de gerenciamento de redes.

Um outro objetivo do protocolo é que o paradigma funcional para monitoração e controle seja suficientemente extensível para acomodar aspectos adicionais, e possivelmente não previstos da operação e gerenciamento de redes (SPECIALSKI, 2002).

[3] O item 3.7 apresenta uma descrição mais detalhada

O terceiro objetivo é que a arquitetura seja o quanto possível independente da arquitetura e dos mecanismos hospedeiros e *gateways* particulares (SPECIALSKI, 2002).

3.2. Componentes Básicos da Arquitetura SNMP

Para que seja possível o gerenciamento, utilizando o modelo SNMP é preciso que se tenha os seguintes componentes, segundo os autores VENÂNCIO NETO (2001) e VERONEZ (2000):

- Um ou mais nodos gerenciados, cada um contendo uma entidade de processamento denominada agente;
- Pelo menos uma estação de gerenciamento contendo uma ou mais entidades de processamento denominadas aplicações de gerenciamento (gerentes);
- Opcionalmente entidades de processamento capazes de agir tanto no papel de gerente como de agente, chamadas de entidades de duplo comportamento;
- Informação de gerenciamento em cada nó gerenciado, que descreve a configuração, estado, estatística e que controla as ações do nó gerenciado;
- Um protocolo de gerenciamento, o qual os agentes e gerentes utilizam para trocar mensagens de gerenciamento, no caso o SNMP.

A Fig. 3.2 ilustra um exemplo dos componentes básicos da arquitetura SNMP, onde se tem uma aplicação de gerenciamento (gerente), quatro elementos gerenciados (agentes), a MIB com as informações sobre os nós e o protocolo usado para a troca de informações é o SNMP.

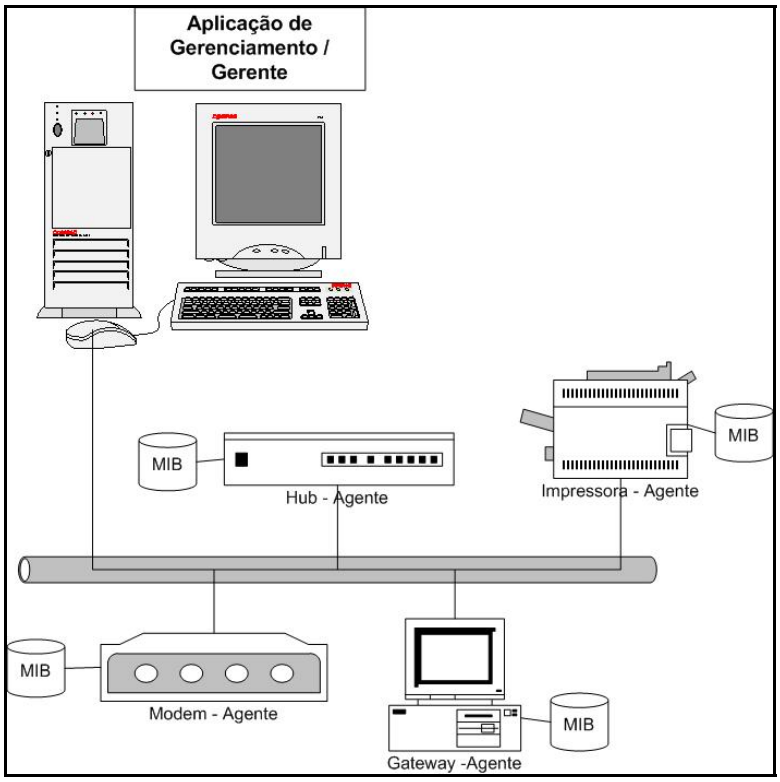

Figura 3.2 - Componentes básicos do SNMP

No mundo SNMP, existem dois tipos de entidades: gerentes e agentes. Segundo (DOUGLAS & SCHMIDT, 2001), o gerente é um tipo de servidor executando algum tipo de sistema de *software* que pode lidar com tarefas de gerenciamento de uma rede. Os gerentes ou NMS (*Network Management Station*) são responsáveis pela operação de *polling* (consultar e alterar informações em um agente) e por receber *traps* (informa que algo aconteceu) dos agentes (objetos gerenciados) de rede.

O agente é um de *software* executado nos dispositivos gerenciados da rede (AIZMAN, 1999). Pode ser programado (um *deamon*, usado em Unix) ou incorporado ao sistema operacional (encontrado em roteadores CISCO). A implementação de agentes SNMP em produtos auxilia os administradores de rede, pois possibilitam o acesso às informações relevantes ao ambiente monitorado, podendo ajudar em importantes decisões. No caso do agente perceber uma anormalidade, ele pode enviar um *trap* para que o gerente seja alertado do “problema”. Um exemplo prático seria o caso em que um agente fornece informações de gerenciamento sobre o estado das interfaces do roteador, a quantidade de bits trafegando em cada porta, se existem interfaces paradas, a velocidade de transmissão, entre outros. Segundo (PARNES et al.,

1999) uma dificuldade na utilização de *traps* é que eles precisam ser configurados (pré-definidos) pelo gerente, mas podem ser importantes para a gerência pró-ativa.

O gerente monitora a rede, solicitando aos agentes seus estados e características, porém a eficiência é aumentada quando os objetos gerenciados enviam *traps* como forma de gerência pró-ativa, evitando o consumo de recursos na rede e no NMS.

Tanto os *polling* quanto os *traps* podem acontecer simultaneamente. Não existem restrições sobre quando o gerente pode consultar o agente e nem sobre quando o agente enviará um *trap*. O tempo referente à consulta do gerente ou o limite para envio de informações dependerá do administrador e da necessidade da rede.

Os objetos representam os recursos da rede, onde cada um é uma variável que representa uma informação sobre o agente. A coleção dos objetos é conhecida como Base de Informações de Gerenciamento (MIB – *Management Information Base*) (STALLINGS, 1999). Esta base pode ser considerada um banco de dados de objetos gerenciados que o próprio agente rastreia e assim todo o tipo de informações requerido pelo gerente é definido por esta MIB.

A Estrutura de Informação de Gerenciamento (SMI - *Structure of Management Information*) é um método usado no SNMP para definir objetos gerenciados e os respectivos comportamentos. As regras para informação de gerenciamento são definidas usando *Abstract Syntax Notation One* (ASN.1).

O SNMP permite ao administrador da rede monitorar ou controlar os dispositivos gerenciados. O primeiro caso está relacionado com a observação e análise, isto é, realiza uma operação de leitura. Já o segundo caso se preocupa com a modificação do valor de um objeto gerenciado, ou seja, a escrita.

3.3. Monitoração da Rede

A monitoração é uma das principais utilizações do SNMP e está envolvida com a tarefa de verificação das informações importantes para o gerenciamento. As informações disponibilizadas para a monitoração podem ser classificadas segundo, STALLINGS (1999) em três categorias: estáticas, dinâmicas e estatísticas.

- Estática: estas informações caracterizam a configuração atual e os elementos. Por exemplo, o número e identificação de portas em um roteador (onde os dados não são trocados freqüentemente);

- Dinâmica: estão relacionadas com os eventos na rede, como a transmissão de pacote na rede, onde os valores são mutáveis;
- Estatística: pode ser derivada das informações dinâmicas, como a média de pacotes transmitidos por uma unidade em um sistema final.

Existem duas técnicas usadas na monitoração de dispositivos gerenciados: *polling* e *event-reporting* (*traps*). A primeira é conhecida como *request-response* e é realizada entre o gerente e agente. O gerente envia uma solicitação com informações do que ele deseja saber e, o agente responde com os dados contidos na MIB, o gerente pode usar o *polling* para aprender/estudar informações dos objetos gerenciados, com isso podendo fazer melhorias e prognósticos na rede.

O *event-reporting* é sempre de iniciativa do agente para o gerente (que recebe a informação). Um agente pode gerar relatórios periódicos sobre as condições de seu estado (STALLINGS, 1999). A notificação pode ser programada pelo gerente, pode existir uma periodicidade ou, então, o agente ser configurado para quando atingir um limiar avisar o acontecido, por exemplo, a capacidade de armazenamento chegou e um patamar de 80%.

A Fig. 3.3 mostra como é feita a monitoração entre gerente e agente no SNMP, partindo de uma operação de consulta do gerente, exigindo uma resposta do agente ou então de um *trap* enviado pelo objeto gerenciado sem nenhuma solicitação da estação de gerenciamento.

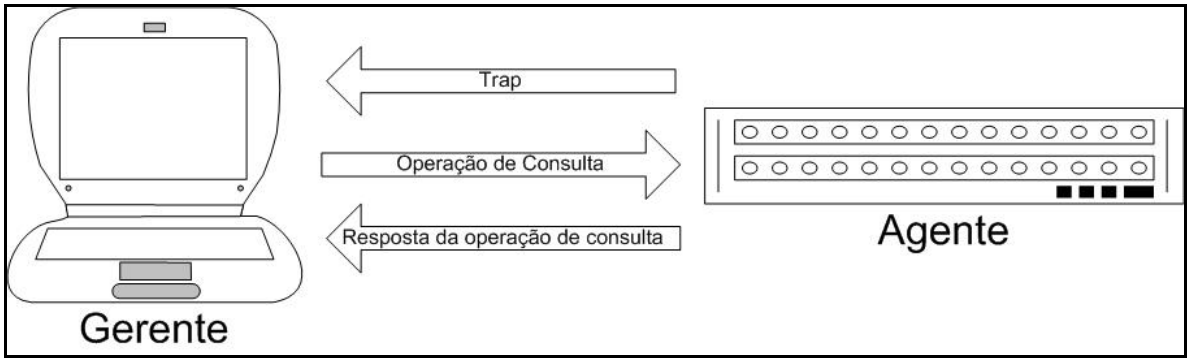

Figura 3.3 - Monitoração da rede

As duas técnicas são empregadas em sistemas de gerência de redes, porém dependendo da situação o foco é alterado. No caso da gerência de telecomunicações é mais usado o *event-report* (de forma pró-ativa), enquanto que o SNMP usa mais o

método de *polling* (não é apropriado para redes grandes, pois tem limitações de performance e segundo, CHEIKHROUHOU & LABETOULLE (2000) pode causar um aumento no tráfego intolerável). Já o modelo OSI usa os dois, sem nenhuma distinção.

Existem fatores que determinam a técnica a ser usada, e depende das necessidades do administrador e do ambiente. Entre alguns elementos têm-se os citados por STALLINGS (1999):

- A quantidade de tráfego gerada por cada método;
- Robustez em situações críticas;
- A demora em notificar o gerente;
- A quantidade de processamento nos dispositivos gerenciados;
- As aplicações de monitoração de rede suportadas;
- As considerações referidas no caso de um equipamento falhar antes de enviar a notificação.

De posse desses fatores, cabe ao gerente/administrador da rede determinar as necessidades de como resolver os problemas e que método é capaz de supri-las sem prejudicar o desempenho da rede. Uma monitoração contínua é essencial para saber o estado do objeto gerenciável (JIAO et al., 2000).

Em relação ao *polling* deve ser avaliada a freqüência levando em consideração três aspectos: o agente/CPU dos dispositivos, o consumo de largura de banda e os tipos de valores que estão sendo solicitados. Por exemplo, alguns valores recebidos podem ser medidos de 10 (dez) em 10 (dez) minutos, pois em alguns ambientes de rede é um desperdício verificar com intervalo de poucos segundos (só acarretaria um tráfego maior). Se o administrador já tem os valores de “picos” e as tendências, deve ficar atento para não congestionar a rede (DOUGLAS & SCHMIDT, 2001). E FLATIN, (1999) complementa que muitas das informações coletadas com o *polling* são redundantes e só acarretam no aumento de pacotes na rede, como por exemplo, o nome do contato.

3.4. Controle da Rede

Enquanto o monitoramento trata da leitura das variáveis, o controle está relacionado com a modificação de parâmetros e execução de ações em dispositivos SNMP e para isso é preciso ter permissão. Dentre as áreas funcionais da gerência de

rede, a configuração e a segurança estão mais voltadas para o controle, enquanto as demais para monitoração.

Como visto no item 2.1**,** a gerência de configuração está relacionada com a inicialização, manutenção e desligamento de parte ou de toda a rede. Pode especificar valores para os atributos e os recursos gerenciáveis, por exemplo, a interface número dois do roteador precisa ser desabilitada, pois está prejudicando o desempenho do mesmo. O gerenciamento de configuração inclui as seguintes funções (STALLINGS, 1999):

- Definição de informações de configuração;
- Atribuição e alteração de valores dos atributos;
- Definição e modificação de relacionamentos;
- Inicialização e terminação de operações na rede;
- Distribuição de *software*;
- Exame das variáveis e relacionamentos;
- Relatórios do *status* da configuração.

O gerente envia uma mensagem para que o agente mude o atributo ou realize alguma ação, tendo como conseqüência uma operação de escrita. É preciso ter cuidado com esse tipo de operação, pois pode modificar completamente o desempenho da rede. No caso de uma alteração errada ser executada, por exemplo, desligar o *switch* principal. A Fig. 3.4 ilustra como é feito o controle em uma rede gerenciada.

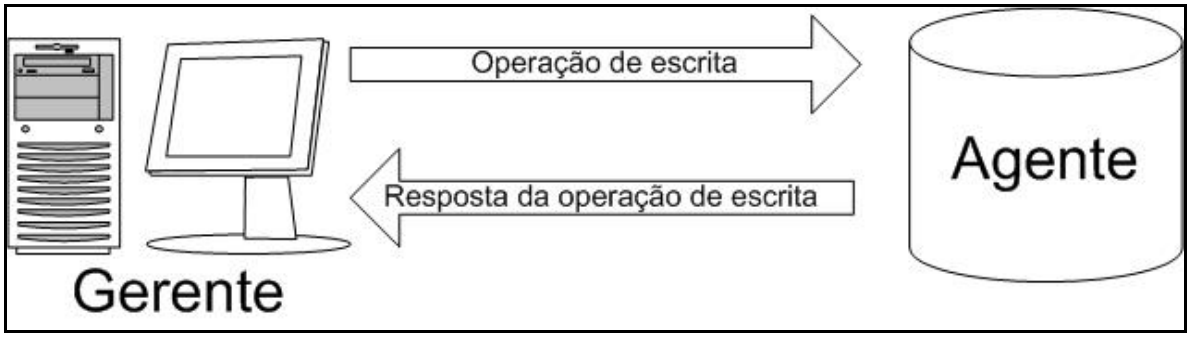

Figura 3.4 - Controle em uma rede gerenciada

3.5. SNMP e Representação de Informações de Gerenciamento

No SNMP, as informações de gerenciamento são representadas usando um subconjunto de sintaxe chamado ASN.1 (*Abstract Syntax Notation One*). A estrutura de

informação de gerenciamento (SMI) descreve como os objetos gerenciados na MIB são definidos.

3.5.1. SNMPv1 (versão 1)

A primeira versão do protocolo SNMP foi formulada no final da década de 80 para resolver problemas de gerenciamento em curto prazo e, oficialmente publicada em 1990. A versão atual do protocolo SNMP, definida na RFC 1157 e é um padrão completo da IETF (*Internet Engineering Task Force*) (DOUGLAS & SCHMIDT, 2001). Uma das dificuldades encontradas diz respeito à segurança, esta baseada em comunidades, que são *strings* de texto puro que permitem qualquer aplicativo, fundamentado em SNMP, conhecido da comunidade e, que esta tenha acesso à informação de gerenciamento de um dispositivo. Os administradores de redes que têm dispositivos gerenciáveis devem ficar atentos com as *strings* da comunidade, pois se um *hacker* tiver acesso a essa informação a rede pode se tornar vulnerável e sofrer conseqüências prejudiciais. Em geral, existem três tipos de comunidade no SNMPv1: *read-only*, *read-write* e *trap*.

Devido a deficiência em relação à segurança, muitos fabricantes decidiram não implementar o comando *set* (não permitindo escrita), com isso reduzindo o SNMP somente à tarefa de monitoração. A versão 3 (três) do protocolo tenta corrigir essas falhas, com isso dando mais confiabilidade ao ambiente de rede de computadores.

3.5.2. SNMPv2 (versão2)

Na tentativa de solucionar problemas com a primeira versão (SNMPv1) e quando se percebeu que o padrão de gerenciamento OSI demoraria a ser implementado, começou-se a desenvolver uma versão conhecida como SNMPv2 (STALLINGS, 1999).

O resultado, foi a publicação de um conjunto de RFCs (RFC 1902 - RFC 1908) datadas de 1996, onde os aspectos importantes de segurança foram deixados de lado, porém melhorias foram feitas na arquitetura: código de erros mais detalhados, definição de uma nova estrutura de informações de gerenciamento, a inclusão de duas novas PDUs (*GetBulkRequest* PDU e *InformRequest* PDU), entre outras (SPECIALSKI, 2002).

Apesar das vantagens apresentadas é pouquíssimo utilizado, sua complexidade implica dificuldades de implementação e não foi bem recebido pela comunidade de

gerência (SPECIALSKI, 2002).

3.5.3. SNMPv3 (versão 3)

A segurança tem sido um ponto fraco nas duas primeiras versões do SNMP, onde é utilizado o conceito de *string* de comunidades como senha para autenticação, que nada mais é do que um texto plano trocado entre o gerente e o agente e não representa nenhuma segurança. Uma pessoa mal intencionada pode capturar essa senha e de posse disso recuperar informações de dispositivos, modificar configurações e até mesmo paralisar a rede.

A terceira versão do SNMP (SNMPv3) está voltada para questões de segurança, não houve alteração no protocolo, nem existem novas operações e tem suporte para as versões anteriores. De acordo com (OMARI; BOUTABA; CHERKAOUI, 1999) as propriedades de segurança abordadas são:

- Autenticação: Permite a um agente verificar se uma solicitação está vindo de um gerente autorizado e a integridade do seu conteúdo;
- Criptografia: Permite que gerentes e agentes criptografarem suas mensagens para evitar invasão de terceiros;
- Controle de Acesso: Torna possível configurar agentes para oferecerem diferentes níveis de acesso a diferentes gerentes.

Uma característica importante é a segurança no pacote SNMPv3 e a outra é no caso do comando *GetBulk* que solicita para o agente devolver valores de uma só vez, reduzindo o tráfego gerado pelo excesso de *polling* criado por múltiplos comandos *GetNext*, com isso melhorando a performance da rede (BATES, 2002).

3.5.4. Estrutura de Informação de Gerenciamento (SMI)

A SMI especifica uma metodologia para definição da informação de gerenciamento contida na MIB e usa um subconjunto de tipos de dados ASN.1. Evita tipos de dados complexo para simplificar a tarefa de implementação e para garantir a interoperabilidade. Existem duas versões: SMIv1 (versão 1) e SMIv2 (versão 2). A primeira versão define com exatidão como os objetos são nomeados e especifica os respectivos tipos de dados associados. Já a segunda fornece otimizações para o SNMPv2. Conforme DOUGLAS & SCHMIDT (2001), a definição de objetos gerenciados no SMIv1 pode ser fragmentada em três atributos:

1. Identificador do objeto (OID): define com exclusividade um objeto gerenciado;
2. Tipo e Sintaxe: o tipo de dados de um objeto gerenciado é definido por meio de um subconjunto da ASN.1, que é uma forma de especificar o modo como os dados são representados e transmitidos entre os gerentes e agentes. A vantagem é o fato de que a notação é independente do sistema operacional da máquina;
3. Codificação: antes de ser transmitido, um objeto gerenciado é codificado em uma *string* de *octetos* por meio do método *Basic Encoding Rules* (BER). O BER define o modo de codificação e decodificação dos objetos para que sejam transmitidos através de um meio de transmissão.

Os nomes para todos os tipos de objetos contidos na MIB são definidos explicitamente na MIB padrão Internet ou em outros documentos que seguem as convenções de nomeação definidas na SMI.

Cada instância de tipo de objeto definida na MIB é identificada, nas operações SNMP, por um nome único chamado nome de variável. Geralmente, o nome de uma variável SNMP é um *OBJECT IDENTIFIER* (OID) da forma x.y, onde x é o nome de um tipo de objeto não agregado definido na MIB e y é um fragmento de *OBJECT IDENTIFIER* que, de uma forma específica para o tipo de objeto nomeado, identifica a instância desejada (SPECIALSKI, 2002). Um OID é formado por uma seqüência de inteiros baseada nos nós da árvore, separada por pontos “.”, por exemplo, se o administrador deseja saber a quantidade de *octetos* de entrada em um determinado dispositivo gerenciável é preciso informar a seqüência 1.3.6.1.2.1.2.2.1.10.

Na árvore de registro de tipos de objetos o nó posicionado no início da árvore é denominado raiz. Tudo que tiver filhos será uma sub-árvore e o que não tiver será chamado de folha ou nó terminal. A Fig. 3.5 apresenta a árvore de registro utilizada para nomeação de objetos definidos na MIB.

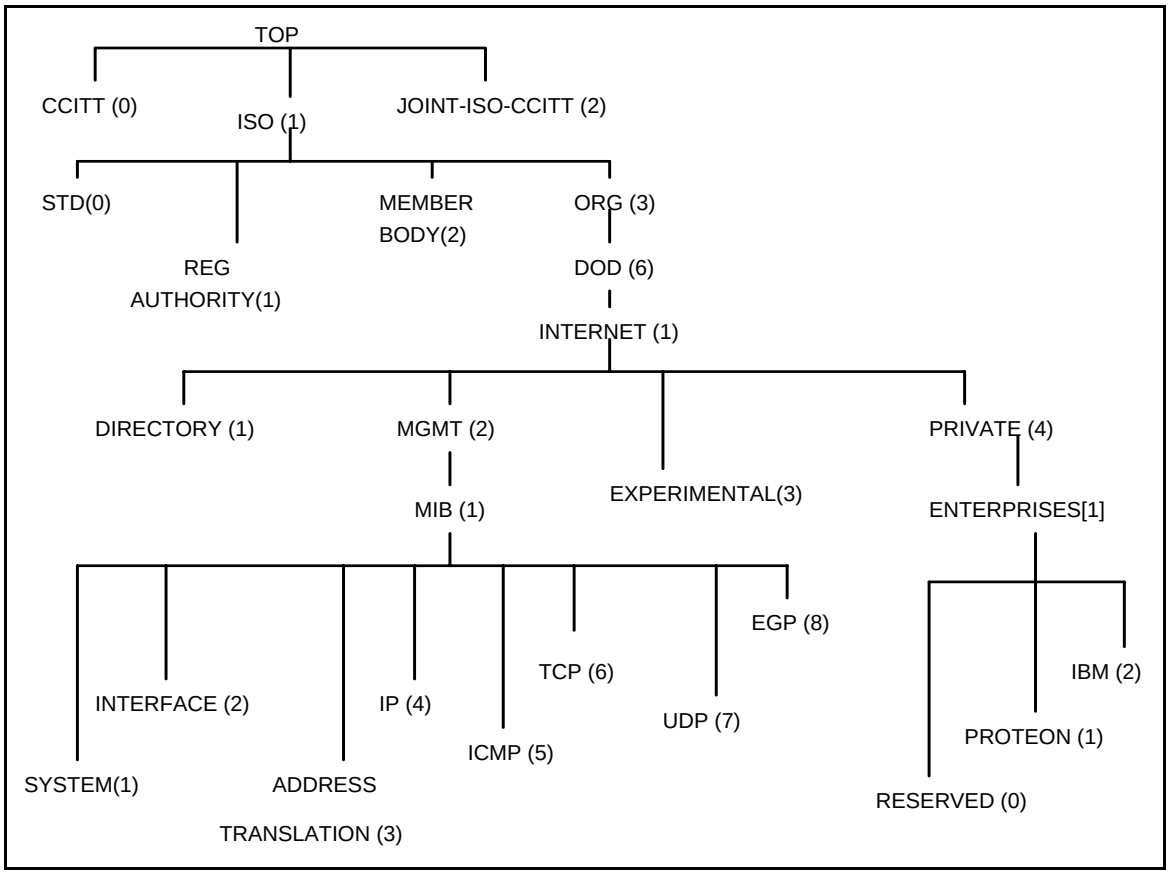

Figura 3.5 - Árvore de registro utilizada para nomeação de objetos definidos na MIB (SPECIALSKI, 2002)

A SMI identifica os tipos de dados que podem ser usado na construção de uma MIB, por exemplo, inteiro, e a forma que os recursos podem ser representados e nomeados. A codificação de dados descreve como as informações associadas com um objeto gerenciado são formatadas para a transmissão.

3.5.5. Base de Informação de Gerenciamento – MIB

A MIB é uma coleção estruturada de objetos gerenciados. O conhecimento dos objetos existentes em uma MIB e de como usar essas informações são importantes para o gerenciamento de redes. Cada elemento gerenciável do sistema mantém uma MIB que reflete o estado dos recursos gerenciados naquele nodo (SPECIALSKI, 2002).

Atualmente existem duas versões da MIB: MIB I e MIB II. A segunda é uma extensão da primeira, com a adição de objetos, grupos e funcionalidades. Os objetos da MIB-II (definida na RFC 1213) são subdivididos nos seguintes grupos mostrados por STALLINGS (1999):

- *System* (1): informações gerais sobre o sistema;
- *Interfaces* (2): informações sobre cada uma das interfaces do sistema para a sub-rede;
- *At* (*address translation; deprecated*) (3): descreve a tabela de translação de endereços para mapeamento de endereços internet para endereços de sub-rede;
- *Ip* (4): informação relativa a experiências de implementação e execução do protocolo IP no sistema;
- *Icmp* (5): informação relativa a experiências de implementação e execução do protocolo ICMP no sistema;
- *Tcp* (6): informação relativa a experiências de implementação e execução do protocolo TCP no sistema;
- *Udp* (7): informação relativa a experiências de implementação e execução do protocolo UDP no sistema;
- *Egp* (8): informação relativa a experiências de implementação e execução do protocolo EGP no sistema;
- *dot3* (*transmission*) (10): fornece informações sobre esquemas de transmissão e protocolos de acesso em cada interface do sistema;
- *snmp* (11): informação relativa a experiências de implementação e execução do protocolo SNMP no sistema.

A idéia de se organizar em grupos é porque os objetos são organizados de acordo com as funções das entidades gerenciadas. Também fornecem um guia para implementadores de agentes saberem identificar quais objetos devem ser implementados. A Fig. 3.6 ilustra a sub-árvore da MIB II.

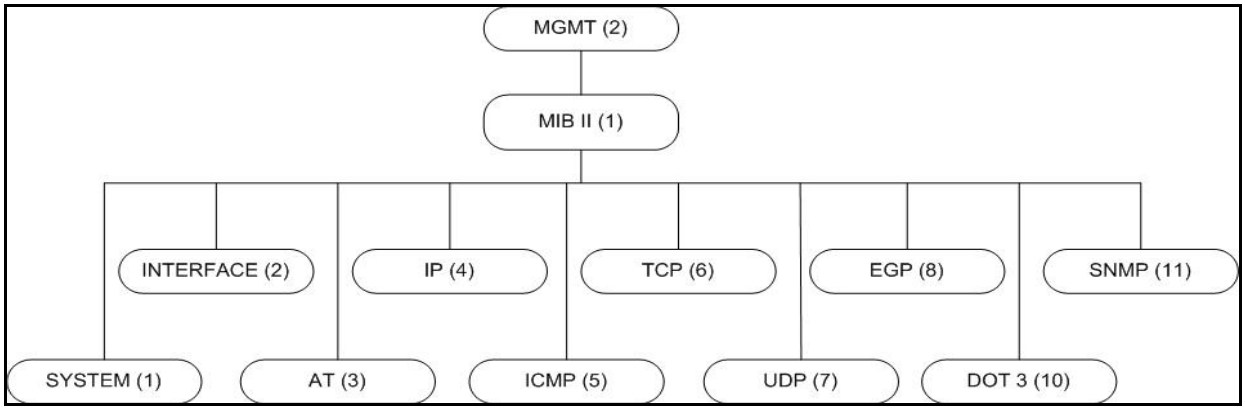

Figura 3.6 – Sub-árvore da MIB II

3.5.6. Monitoração Remota – RMON

Somente com as informações coletadas na MIB, fica difícil para o administrador saber o tráfego existente na rede, pois os dados são referentes aos dispositivos gerenciados e não à rede toda. Outro problema encontrado é a técnica de *polling*, que obriga ao gerente a ficar solicitando informações dos agentes periodicamente, podendo causar uma sobrecarga na rede.

A tecnologia RMON consiste na presença de um monitor instalado na rede que se deseja estudar, coletando informações e, eventualmente, enviando notificações sobre a ocorrência de eventos. O monitor pode ser tanto um dispositivo dedicado à captura de dados e sua análise, como também pode estar implantado em estações de trabalho, servidores, roteadores, *hubs*, por exemplo (SPECIALSKI, 2002). Tipicamente, opera na rede em modo promíscuo (recebendo todos os dados que trafegam pela rede), verificando cada pacote e inclusive podendo armazenar as informações para uma análise posterior. Além disso, melhora o desempenho da rede, haja vista que o gerente não precisa coletar constantemente informações dos agentes.

De acordo com STALLINGS (1999), a RMON representa uma das principais adições para a arquitetura SNMP é uma extensão da MIB Internet. Existem duas versões: a RMONv1 que é definida na RFC 1757 e uma versão otimizada do padrão denominada RMONv2 que é definida na RFC 2021.

A RMONv1 fornece ao gerente dados estatísticos sobre uma LAN ou WAN inteira, no nível de pacotes. Já a segunda versão aprimora a primeira ao fornecer dados estatísticos no nível de rede e de aplicativos, que podem ser obtidos de várias maneiras; uma delas seria colocar um RMON em cada segmento de rede a ser monitorado (DOUGLAS & SCHMIDT, 2001).

STALLINGS (1999) explica que a RMON foi projetada para atingir os seguintes objetivos:

- Operação *off-line*: o monitor coleta e armazena estatísticas que podem ser recuperadas pela estação gerente a qualquer momento;
- Monitoração pró-ativa: o monitor está constantemente rodando diagnósticos e armazenando informações. Se ocorrer alguma anormalidade pode notificar o gerente para tomar alguma decisão, possivelmente usando o diagnóstico;
- Detecção e registro de problemas: o monitor pode ser configurado para checar

constantemente determinadas condições na rede e quando ocorrer notificar o gerente ou armazenar o *log*;

- Resumo dos dados: o monitor pode realizar análises específicas dos dados coletados, ajudando a estação de gerenciamento na sua tarefa;
- Múltiplos gerentes: o monitor é capaz de suportar várias estações gerentes, que podem executar diferentes funções e prover capacidades de gerência para unidades diferentes.

3.5.7. Mensagens SNMP

Como mencionado por DOUGLAS & SCHMIDT (2001), o SNMP usa o protocolo UDP para o transporte de informações entre as entidades. Utiliza a porta 161 do UDP para enviar e receber solicitações e a 162 para *traps*. Os números das portas são usados pelos dispositivos que possuem o SNMP, porém alguns fabricantes mudam essas configurações em seus equipamentos proprietários.

As mensagens trocadas entre as estações de gerenciamento e os agentes no SNMP possuem: um número indicando a versão, um nome da comunidade usada na troca e uma unidade de dados do protocolo (PDU- *Protocol Data Unit*) (AUGUSTO NETO, 2001).

A mensagem do SNMP contém duas partes: cabeçalho e a PDU. A Fig. 3.7 ilustra o formato básico de uma mensagem SNMPv1.

Figura 3.7 - Formato básico de uma mensagem SNMPv1

Segundo STALLINGS (1999) o cabeçalho da mensagem é composto por dois campos:

- Número da Versão: indica qual versão do protocolo SNMP está sendo usada;
- Nome da Comunidade: é uma forma de confiabilidade entre o gerente e o agente, ou seja, uma espécie de senha. Um agente é configurado com três nomes de comunidades: *read-only*, *read-write* e *trap*.

A Fig. 3.8 mostra como são definidos os campos de mensagem SNMP as PDUs *GetRequest, GetNextRequest, SetRequest, SetResponse e Traps* presentes no SNMPv1.

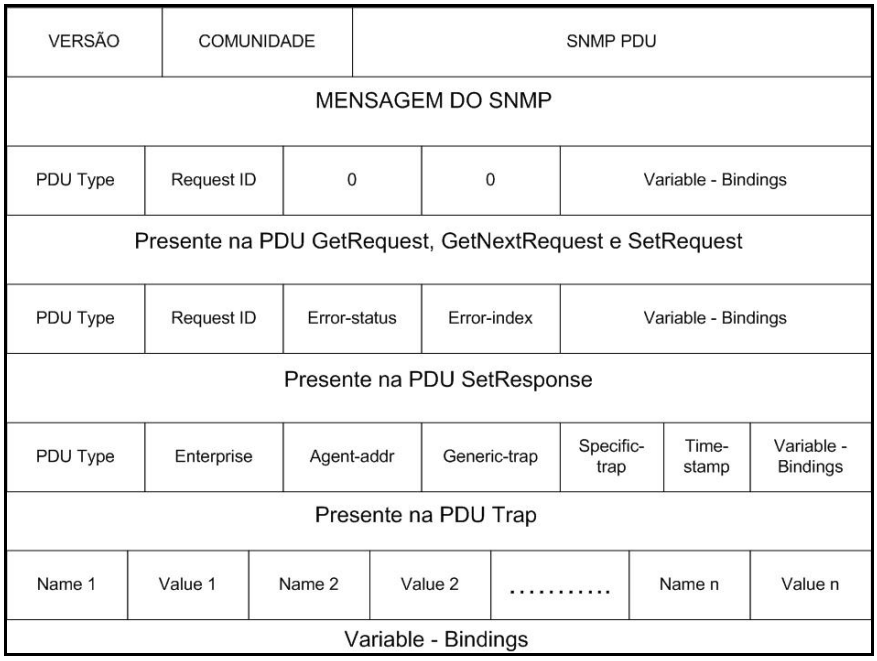

Figura 3.8 - Definição dos campos de mensagens SNMPv1

A significação de cada campo é:

- *PDU type*: informa o tipo de PDU;
- *Request ID*: associa solicitações SNMP com respostas;
- *Error-status*: indica que algo aconteceu enquanto processava a requisição.
- *Error-index*: associa um erro com uma instância de um objeto particular.
- *Variable Bindings*: Uma lista dos nomes das variáveis e seus correspondentes valores;
- *Enterprise*: tipo de objeto que gerou o *trap*, baseado no *SysObjetcID*;
- *Agent-addr*: endereço do objeto gerador do *trap*;
- *Generic-trap*: um dos seis tipos de *traps* genéricos;
- *Specific-trap*: código do *trap* específico;
- *Time-stamp*: é a hora da geração do *trap*.

3.5.8. Procedimento para transmissão recebimento de mensagens SNMPv1

Em STALLINGS (1999) é explicado que para a geração e transmissão de PDUs entre entidades SNMP são usados os seguintes passos:

1. A PDU é construída usando ASN.1;
2. Em seguida a PDU é passada para o serviço de autenticação junto com o endereço de transporte de origem e destino e o nome da comunidade;
3. Depois a entidade de protocolo constrói a mensagem, contendo o campo versão, nome da comunidade e o resultado do passo anterior;
4. Então o novo objeto ASN.1 é codificado usando as regras básicas de codificação (BER) e enviado para o serviço de transporte.

Na hora em que a entidade SNMP recebe uma mensagem, realiza as etapas a seguir (STALLINGS, 1999):

1. Primeiramente é feita uma verificação na sintaxe da mensagem e, se estiver tudo certo, a mensagem é aceita; senão, a mensagem é descartada;
2. Depois, a entidade SNMP verifica o número da versão do protocolo e, no caso de ser incompatível, descarta a mensagem;
3. Então a entidade de protocolo transmite o nome da comunidade, a parte da mensagem referente a PDU e o endereço de transporte da origem e destino para o serviço de autenticação;
    a. Se a autenticação não tiver sucesso, então o serviço de autenticação gera um *trap* e descarta a mensagem;
    b. Senão, o serviço de autenticação retorna uma PDU em forma de um objeto ASN.1;
4. Por fim, a entidade de protocolo executa uma verificação na sintaxe da PDU e, no caso de falha, a mensagem é descartada, senão, a PDU é processada.

Tanto na transmissão quanto no recebimento de mensagens SNMPv1 o serviço de autenticação age somente no sentido de verificar o nome da comunidade, se é compatível ou não com o informado pelo usuário/administrador.

3.5.9. Operações do SNMP

Este item descreve o significado de cada uma das operações suportadas pelo protocolo SNMP (versão 1, 2 e 3) e como funcionam. Para melhor compreensão, o apêndice 2 apresenta como algumas PDUs são construídas em ASN.1.

3.5.9.1. GetRequest PDU

Essa PDU é solicitada de um gerente para um agente, que recebe e devolve o valor da instância do objeto referenciado pelo OID contido na PDU. A resposta é devolvida usando a *GetResponse* PDU. É usado nas três versões do protocolo SNMP.

Junto da solicitação são enviadas as *Variable Bindings* para o agente saber o que a entidade de gerenciamento está querendo. É uma lista de objetos da MIB que podem ser consideradas como os pares OID=valor que facilita para o gerente selecionar as informações necessárias quando o receptor preencher a solicitação e retornar a resposta (DOUGLAS & SCHMIDT, 2001).

Esta é uma operação atômica, onde todos ou nenhum valor é retornado. Se pelo menos um valor não pode ser informado por algum motivo ou apresentar erro, então nenhuma variável é retornada.

3.5.9.2. GetNextRequest-PDU

A operação *Get* é usada para recuperar um único objeto da MIB de cada vez. O gerenciamento dessa maneira pode ser uma perda de tempo. Uma solução encontrada é a PDU *GetNextRequest* que permite recuperar mais de um objeto de um dispositivo em um intervalo de tempo.

Esta PDU é usada nas três versões do protocolo. É realizada uma solicitação do valor de uma ou um conjunto de variáveis que seguem a ordem lexiográfica àquelas informadas na solicitação (SPECIALSKI, 2002).

3.5.9.3. GetBulkRequest-PDU

O *GetBulkRequest* é um dos benefícios do SNMPv2 e é usado também no SNMPv3. Permite que o gerente recupere uma grande quantidade de informações de uma só vez do agente. Usa os mesmos princípios do *GetNextRequest*, a seleção é dada sempre na próxima instância do objeto em ordem lexicográfica, porém é possível selecionar múltiplos acessos em ordem lexicográfica. Outra vantagem também se refere

ao fato de que nas operações anteriores, se o agente não puder retornar todas as respostas solicitadas, enviará uma mensagem de erro sem dados. Já no *GetBulk* o agente é instruído para informar o máximo de repostas possíveis, inclusive incompletas (DOUGLAS & SCHMIDT, 2001). A Fig. 3.9 apresenta como é realizada a seqüência de solicitações da operação *GetBulk*.

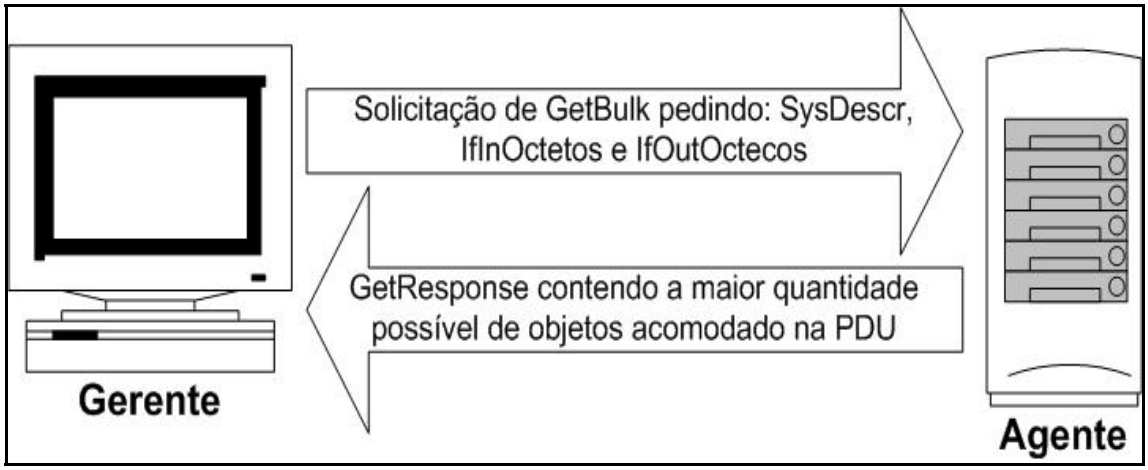

Figura 3.9 - Seqüência de solicitações de GetBulk

3.5.9.4. SetRequest-PDU

É enviada pelo gerente para o agente quando o mesmo deseja fazer uma modificação (escrever) no valor do objeto gerenciado. Está presente nas três versões do protocolo. Os objetos na MIB definidos como *read-write* ou *write-only* podem ser alterados, usando essa operação(STALLINGS, 1999). É uma operação atômica, onde todos os valores são alterados ou nenhum é alterado.

3.5.9.5. Trap PDU

Até agora foram apresentadas todas as operações que são de iniciativa do gerente para o agente. No caso da *trap* PDU é ao contrário, é um meio do agente informar à estação de gerenciamento que aconteceu alguma anormalidade na rede.

A configuração necessária para que isso aconteça deve ser feita no próprio agente pelo administrador da rede. O destino do *trap* é geralmente o endereço IP do gerente e não existe confirmação da mensagem.

Outra definição usada por DOUGLAS & SCHMIDT (2001) é que os *traps* são um método para que um agente envie a uma estação de monitoramento uma notificação

assíncrona sobre as condições que o monitor deve conhecer. São enviados pelo UDP na porta 162.

Pelo fato do SNMP usar o UDP e algumas vezes os *traps* não chegarem no seu objetivo, não os torna inúteis. É uma boa solução para redes com um grande tráfego, pois dessa forma evita aos gerentes ficarem periodicamente, coletando informações de seus agentes, gerando um *overhead* na rede e nos gerentes.

As informações enviadas através dos *traps* são úteis para a gerência pró-ativa, pois possibilitam que o(s) gerente(s) seja(m) notificado(s) de alguma anormalidade no(s) agente(s). De posse desta(s) informação(ões), podem executar ações para tentar solucionar ou minimizar o(s) problema(s).

Outro fator importante é que quando o *trap* chega no gerente, ele precisa ser interpretado para poder diagnosticar o problema. São classificados em duas categorias: *traps* genéricos e *traps* específicos das empresas. Existem sete tipos de *traps* genéricos numerados de zero a seis como será apresentado adiante. Os *traps* específicos proporcionam aos fabricantes o uso das vantagens do *trap* (criando de acordo com suas necessidades e bem mais específicos).

Segundo DOUGLAS & SCHMIDT (2001), os *traps* podem relatar algumas situações:

- Uma interface de rede no dispositivo (onde está o agente em execução) foi paralisada;
- Uma interface de rede no dispositivo (onde está o agente em execução) foi reativada;
- Uma chamada recebida em um *rack* modem não conseguiu estabelecer uma conexão com o *modem*;
- A ventoinha em um computador ou roteador está com defeito.

Os sete tipos de *traps* citados por (CASE, 1990) são:

- *coldStart* (0): significa que a entidade SNMP que enviou a mensagem está se reinicializando para que a configuração no agente ou a implementação na entidade de protocolo seja alterada;
- *warmStart* (1): significa que a entidade SNMP que enviou a mensagem está se reinicializando, porém não especificamente por causa da alteração da configuração do agente ou implementação na entidade de protocolo;

- *linkDown* (2): significa uma falha em um do *links* de comunicação do agente;
- *linkUp* (3): significa que a entidade de protocolo de envio reconhece que um *link* de comunicação representado na configuração do agente voltou a estar ativo;
- *autenticationFailure* (4): significa que a entidade de protocolo é o destinatário de uma mensagem de protocolo que não está apropriadamente autenticada. Enquanto as implementações do SNMP devem ser capazes de gerar este *trap*, devem também ser capazes de suprimir a emissão dos mesmos através de mecanismo específico de implementação;
- *egpNeighborLoss* (5): significa que um vizinho do EGP ficou inativo;
- *enterpreseSpecific* (6): significa que a entidade que enviou protocolo reconhece que algum evento *enterprise-específico* (proprietário do fabricante) ocorreu.

O SNMPv2 define *traps* de uma maneira um pouco diferente, enquanto que na primeira versão são definidas como *TRAP-TYPE* na MIB, na segunda são definidas como *NOTIFICATION-TYPE*. E os *traps* no SNMPv3 são os mesmos do SNMPv2 com a inclusão dos recursos de autenticação e privacidade (DOUGLAS & SCHMIDT, 2001). A Fig. 3.10 demonstra a geração de *trap*.

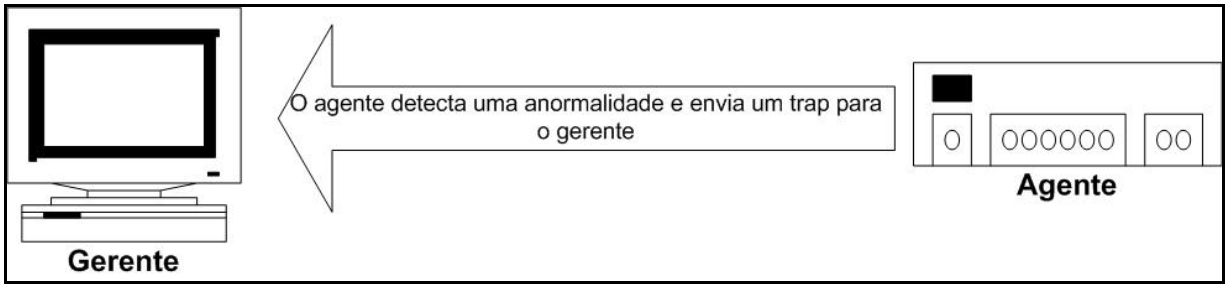

Figura 3.10 - Geração de *trap*

3.5.9.6. InformRequest PDU

O *InformRequest* PDU possui o mesmo formato que a *GetRequest* PDU e é usado a partir da segunda versão do SNMP (SNMPv2). Permite a comunicação entre estações de gerenciamento, quando se tem mais de uma estação gerente na rede. O envio de um *InformRequest* necessita que o receptor confirme o recebimento da mensagem (NATACLE, 1998). Também pode ser usado para o envio de *traps*, nesse caso o agente

receberá um aviso confirmando a chegada, podendo inclusive reenviar no caso de não receber a confirmação. A dificuldade em usar o *inform* para o envio de *traps* é que ele precisa de confirmação e, dependendo da rede, pode gerar *overhead*, pois o tráfego será aumentado.

3.5.9.7. Report

Como apresentado em DOUGLAS & SCHMIDT (2001), esta operação foi definida na segunda versão do SNMP, mas nunca foi implementada e agora faz parte da especificação do SNMPv3. Seu objetivo é permitir a comunicação entre os mecanismos do SNMP, principalmente para relatar problemas relacionados ao processamento de mensagens.

3.6. Gerenciamento Centralizado do SNMP

Segundo DUARTE et al (2002), a arquitetura para gerência de redes IP, tradicionalmente, adota um dispositivo central para as atividades de monitoração e controle, realizando, assim, estas tarefas através do protocolo SNMP junto aos agentes distribuídos. Esta centralização, dependendo da complexidade do ambiente, pode sobrecarregar da rede e do NMS.

Outra definição usada por VENÂNCIO NETO et al (2002) indica que, geralmente, uma única estação de gerência é adotada, concentrando, assim, todas as operações referentes ao gerenciamento da rede (monitoração e controle) e a recepção dos relatórios de eventos (*traps/informs*). Esta concentração pode implicar perda de desempenho do NMS e gargalos na rede.

A Fig. 3.11 elucida um possível ambiente de gerenciamento de rede centralizado, onde se tem um gerente que administra todas as mensagens SNMP na rede. Com isso, dependendo do fluxo de informações, pode haver uma sobrecarga no mesmo.

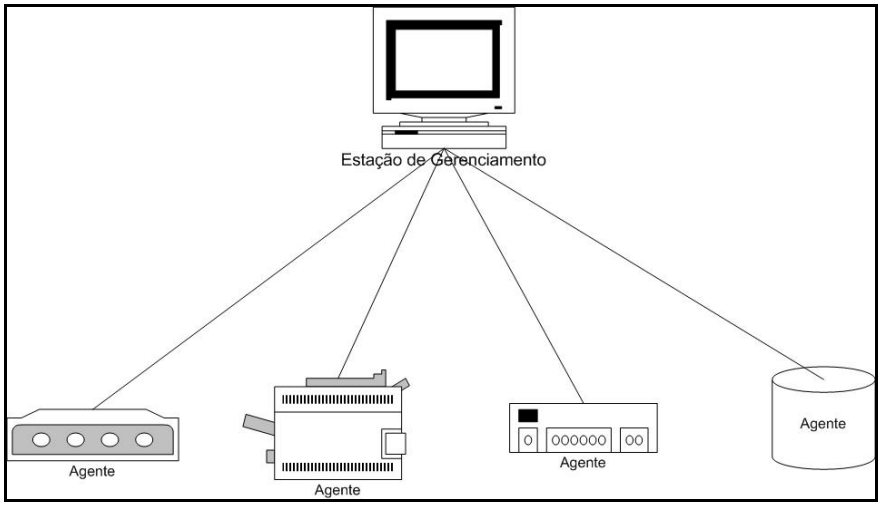

Figura 3.11 - Possível ambiente de gerenciamento de redes centralizado

3.7. Interoperabilidade no SNMPv1 e SNMPv2

Uma das dificuldades encontradas na transição do protocolo SNMPv1 e SNMPv2 é em relação à incompatibilidade. A idéia era que a evolução fosse bem recebida, porém existiam duas diferenças (o formato das mensagens e as operações suportadas pelos protocolos) que influenciaram na aceitação da transição dos protocolos. Em relação ao formato das mensagens implementadas na segunda versão apresentam-se incompatibilidades, porque utilizam cabeçalhos e formatos de PDUs diferentes da primeira versão.

Na tentativa de resolver esses problemas foram propostas duas soluções que segundo STALLINGS (1999) são: agente *proxy* e gerente bilíngüe.

3.7.1. Agente proxy

Um agente SNMPv2 pode ser implementado e configurado para atuar como um agente *proxy*, permitindo a comunicação entre as duas versões. O agente *proxy* recebe as PDUs vindas do gerente SNMPv2, as converte para PDUs SNMPv1 e depois envia para o agente SNMPv1. Deve se seguir as seguintes regras na hora de enviar a mensagem do gerente para o agente (STALLINGS, 1999):

- As PDUs *GetReques*, *GetNextReuqest* e *SetRequest* não são alteradas;
- A PDU *GetBulkRequest* é convertida para *GetNextRequest* com as mesmas listas *variable-binding*.

Quando a mensagem partir do agente SNMPv1 para o gerente SNMPv2, primeiramente deve ser mapeada pelo agente *proxy* SNMPv2 e depois enviada para o NMS SNMPv2. A conversão segue as seguintes regras: (STALLINGS, 1999):

- A *GetResponse* PDU não é alterada;
- A *trap* PDU é convertida para SNMPv2;

3.7.2. Sistemas de Gerenciamento de Redes Bilíngüe

Uma outra alternativa encontrada para a coexistência é o gerente bilíngüe, que suporta ambas versões do protocolo (SNMPv1 e SNMPv2). Quando uma aplicação de gerenciamento deseja enviar informações para um agente, o gerente bilíngüe pode escolher qual versão usar, baseado em informações armazenadas em um banco de dados local que indica qual protocolo é usado por cada agente.

A Fig. 3.12 ilustra como é feita essa transição entre gerente SNMPv2 e agentes SNMPv1 e 2 em um sistema de gerenciamento de redes bilíngüe.

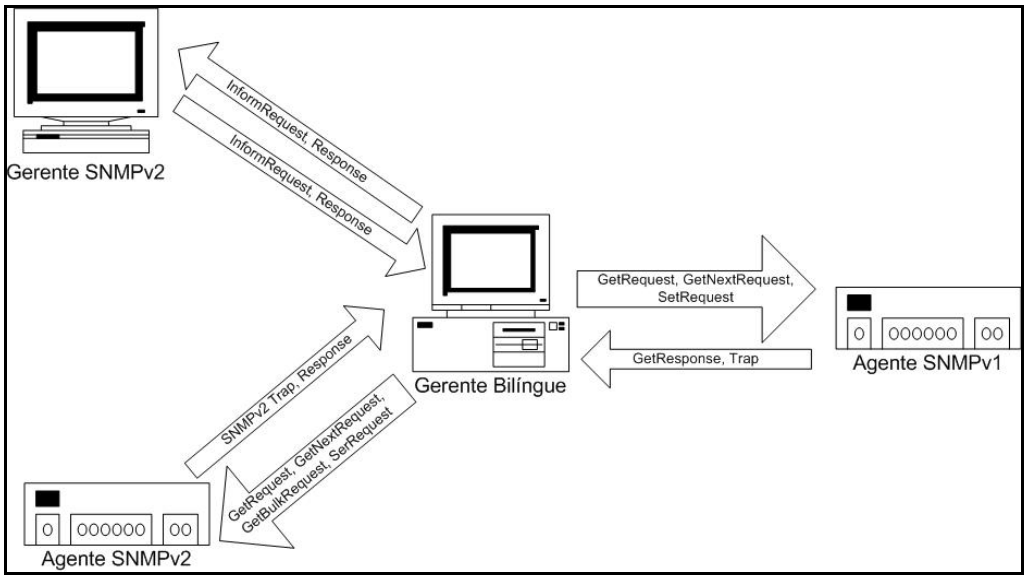

Figura 3.12 - Sistema Bilíngüe

3.8. *Software* para Gerência de Redes

O *software* para gerenciamento de redes é de fundamental importância para obter informações e tomar atitudes sobre o funcionamento de seu ambiente. Fica geralmente localizado na estação gerente (em alguns casos um *host* pode ter função de NMS e agente ao mesmo tempo), é responsável pelas operações realizadas nos agentes e também pelo recebimento das notificações.

Existem diversos *softwares* comerciais ou gratuitos que proporcionam ao usuário um gerenciamento satisfatório, porém a escolha dependerá da quantidade de recurso financeiro que se está disposto a despender, grau de dificuldade de instalação e manuseio e as necessidades da rede.

Vários *softwares*, no mercado, proporcionam uma interface amigável e comandos que permitem executar algumas ou todas as operações do SNMP. O responsável pela rede pode configurar essas ferramentas de acordo com o seu ambiente. Por exemplo, as informações podem ser mostradas em forma gráfica, de modo a facilitar a compreensão dos dados (DOUGLAS & SCHMIDT, 2001).

A tabela 5.1 mostra alguns fabricantes, *softwares* de gerenciamento de redes, sistemas operacionais que suportam e o custo, aproximadamente dos pacotes de acordo com IBM (2002), SUN (2002), MURPHY (2002).

Tabela 5.1 - *Softwares* de gerenciamento de rede

Fabricante	***Software* de Gerência**	**Sistema Operacional**	**Custo U$**
Cisco System, Inc	Cisco Works Blue Internetworks Status Monitor	TME/10 NetView OS/390 e Solve: Netmaster	14.995,00
Hewlett-Packard	Hp OpenView Network Node Manager	HP-UX, SunOS e Solaris	15.750,00
IpSwitch, Inc	WhatsUp Gold	Windows 95/98/NT/2000	795,00
Tobias Oetikere e outros	MRTG	Unix, Linux e Windows	Grátis
IBM	Net View	AIX, Linux, Solaris, Windows NT/2000	N/A
Sun	Solstice Domain Manager 2.3	Solaris	11.084,00

3.9. Considerações Finais

Como a própria nomenclatura diz, o SNMP é um protocolo simples de gerenciamento de redes. Atualmente, existem três versões (SNMPv1, SNMPv2, SNMPv3), onde cada nova versão tenta corrigir problemas e implementa novas funções. Um dos grandes problemas encontrado nas primeiras versões está relacionado com a

segurança, pois a autenticação está relacionada somente com o nome da comunidade, agindo como uma senha que o gerente passa em texto plano (sem nenhuma criptografia) para o agente, permitindo que pessoas não autorizadas “peguem” essas mensagens em benefício próprio ou prejudiquem o funcionamento normal da rede.

A arquitetura SNMP pode ser considerada como centralizada, pois geralmente um gerente é responsável por um ou mais agente. A técnica mais utilizada é a monitoração e não o controle. As mensagens podem ser enviadas dos gerentes para os agentes (*polling*) ou ao contrário (*trap*). Apesar de consumir mais recursos da rede, a técnica de *polling* tem mais ênfase do que a *trap*, que segundo JIAO et al. (2000) pode ser mais eficiente dependendo da situação. O *event report* pode trazer informações precisas e detalhadas sobre os eventos ocorridos, além de proporcionar a gerência pró-ativa.

O modelo SNMP não está preparado para gerenciar o ambiente computacional de forma pró-ativa, pois para obter dados sobre seus agentes o NMS utiliza a técnica de *polling*. Porém, se houvesse melhorias no *event report* seria possível realizar a gerência pró-ativa, onde o gerente receberia informações precisas de anormalidades ocorridas em seus agentes e executaria ações para não comprometer o bom funcionamento do ambiente.

A idéia de se ter somente um gerente responsável por uma rede não é uma boa solução. Para redes com pouco tráfego e poucos equipamentos gerenciados pode ser até uma solução viável, mas para ambientes com muitos computadores, dispositivos gerenciáveis, tráfego “pesado”, que prescinde de informações, não é considerada uma solução adequada, pois o excesso de *polling* pode sobrecarregar o gerente e causar um *overhead* na rede.

O gerenciamento distribuído é uma excelente proposta para o gerenciamento de redes de computadores. Uma solução para questões apresentadas é dividir o sistema de gerenciamento em módulos de gerência menores, onde cada um responderia por funções específicas. Por exemplo, cada gerente poderia ser responsável por uma ou mais áreas funcionais da gerência de redes OSI, com isso diminuindo a quantidade de tráfego em somente uma estação de gerenciamento. Nesse caso, uma solução é dar uma importância maior à técnica de *traps*, que é bastante usada na gerência de telecomunicações.

Existem algumas soluções proprietárias para essa questão do uso de *traps* (em relação à prioridade e destinatários), porém, na sua maioria, são soluções comerciais e não nativas do SNMP. Já no ambiente OSI existe uma ferramenta nativa para filtrar alarmes e definir o(s) gerente(s) que deve(m) receber as notificações. Esta abordagem será apresentada no próximo capítulo.

4. Discriminador de Repasse de Eventos

Neste capítulo será detalhado o discriminador de repasse de eventos presente na arquitetura de gerenciamento OSI e definido na norma X-734 ISO/IEC. Primeiramente, será mostrado o escopo da norma, depois as definições importantes, os objetivos, o modelo para função de gerenciamento de relatórios de eventos, o modelo genérico de relatório de eventos, a função de gerenciamento de relatórios de eventos, os atributos do discriminador de repasse de eventos, os pacotes de programação, os serviços existentes e, por fim, a conclusão.

4.1. Escopo

Esta recomendação/padrão internacional define uma função de gerenciamento de sistema que pode ser usada por um processo de aplicação em um ambiente de gerenciamento centralizado ou distribuído, para interagir com os propósitos de um sistema de gerenciamento, como definido pelo CCITT (*International Consultative Committee on Telegrahy and* Telephony) Rec. X.700 | ISO/IEC (*International Organization for Standardization / International Electrotechnical Commission*) 7498-4. Define-se a função de gerenciamento de repasse de eventos, serviços e duas unidades funcionais (unidade funcional de gerenciamento de relatório de eventos e unidade funcional de gerenciamento da monitoração de relatório de eventos). O quadro 1 apresenta o que esta recomendação/padrão internacional contém ou não segundo a ITU (1993):

Quadro 4.1 - Conteúdo e não conteúdo da norma

Contém na Norma	Não Contém na Norma
Estabelecer requisitos do usuário para a função de gerenciamento de repasse de eventos Estabelecer modelos que relatam os serviços providos pela função para requisitos do usuário	Definir a natureza de qualquer implementação destinada a oferecer função de gerenciamento de repasse de eventos Especificar a maneira pela qual o gerenciamento é consumado pelo usuário da função de repasse de eventos
Definir os serviços providos pela função	Definir a natureza de qualquer interação a qual resulte no uso da função de repasse de eventos

Especificar o protocolo necessário para prover os serviços	Especificar os serviços necessários para o estabelecimento, comunicado normal ou não de uma associação de gerenciamento
Definir o relacionamento entre os serviços e as operações e notificações da SMI	Especificar os requisitos para autorização da utilização da função de repasse de eventos para qualquer atividade associada
Definir relacionamentos entre outras funções de gerenciamento	Definir os objetos gerenciáveis destinados ao gerenciamento de equipamentos com protocolos proprietários
Especificar requisitos para adaptação	

4.2. Definições

Serão apresentadas algumas definições importantes de elementos envolvidos no processo da função de gerenciamento de repasse de eventos (ITU, 1993, SPECIALSKI et al, 1993):

- Discriminador: O objeto discriminador estabelece condições que devem ser satisfeitas antes de permitir que um objeto de entrada seja repassado. O repasse de um objeto também pode estar associado a um pacote de programação;
- Discriminador de Repasse de Eventos: é um discriminador que toma suas atitudes, conforme os relatórios de eventos potenciais;
- Função de Gerenciamento de Repasse de Eventos: É uma função, incluindo a definição de uma classe de objetos de suporte ao gerenciamento, que permite um gerente controlar a transmissão de Relatórios de Eventos de objetos gerenciados independentes da definição de objetos gerenciados;
- Relatório de Evento Potencial: Notificação enviada pelos objetos gerenciados e entregue ao discriminador, onde o mesmo analisa a mensagem e, de acordo com a sua configuração, envia ou não para algum gerente (pode ser ou não transformada em um relatório de eventos).

4.3. Finalidades do Discriminador

Esta função tem como objetivo, de acordo com (ITU, 1993, SPECIALSKI et al, 1993):

- Definir um serviço de controle de relatório de evento que permita ser flexível no sentido de selecionar qual(is) relatório(s) de evento(s) deve(m) ser enviado(s) a um sistema de gerenciamento particular;
- Especificar qual(is) destinatário(s) deve(m) receber determinados relatórios de eventos;
- Especificar um mecanismo para controlar o repasse de relatórios de eventos que seja possível, por exemplo, suspender ou reiniciar seu encaminhamento;
- Possibilitar que um sistema externo de gerência modifique as condições, usadas nos relatórios de eventos;
- Possibilitar a designação de endereços alternativos de *backup* para enviar relatórios de eventos quando o endereço primário não estiver disponível.

4.4. Modelo para a Função de Gerenciamento de Relatórios de Eventos

As necessidades funcionais descritas, anteriormente, tratam do comportamento do sistema, podendo ser reduzida a uma necessidade básica no comportamento do sistema, o que possibilita especificar condições a serem satisfeitas por um relatório de evento potencial, emitido por um objeto gerenciado particular, para ser enviado a endereços de destinatários específicos (ITU, 1993).

4.5. Modelo de gerenciamento de Relatório de Eventos

O modelo de gerenciamento de relatório de eventos descreve os componentes conceituais, utilizados no processamento local de relatórios de eventos potenciais e na emissão desses relatórios (SPECIALSKI et al., 1993). Este também mostra o controle, o relatório de eventos e a recuperação das mensagens.

A função conceitual de pré-processamento de eventos recebe notificações locais dos objetos gerenciados e transforma em relatórios de eventos potenciais, onde os mesmos são encaminhados para todos os discriminadores de repasse de eventos, presentes no sistema aberto local. Cada relatório de evento potencial é um objeto de entrada de um discriminador, que seleciona quais devem ser enviados ou não para um destino particular, em um período de tempo, e não são visíveis para os demais componentes do sistema. O discriminador, também, especifica se o relatório será enviado em modo confirmado, ou não-confirmado.

Cada discriminador de repasse de eventos deve determinar um intervalo de duração no qual os relatórios de eventos serão selecionados e enviados adiante.

Um importante componente é o construtor discriminador (*discriminator construct*) que especifica quais características e condições um relatório de evento potencial deve ter para ser repassado para seu(s) destinatário(s) (ITU, 1993).

Por ser um objeto gerenciado, o discriminador de repasse de eventos é capaz de emitir notificações, onde as mesmas são processadas como relatório de eventos potencial por todos os discriminadores de repasse de eventos (isso o inclui).

A Fig. 4.1 demonstra o modelo de gerenciamento de relatório de eventos.

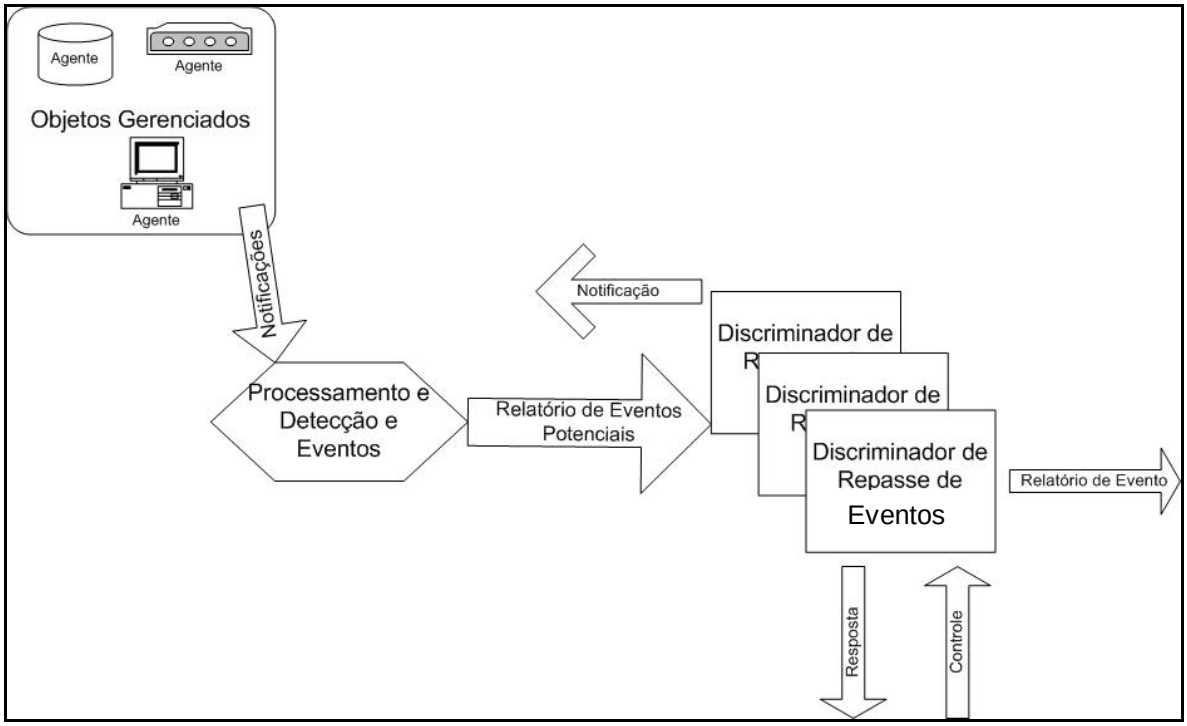

Figura 4.1- Modelo de gerenciamento de relatório de eventos

4.6. Função de gerenciamento de relatório de eventos

Esta função permite um sistema aberto controlar o discriminador e o encaminhamento de relatórios de eventos de outros sistemas abertos. Os relatórios de eventos são frutos de uma notificação da ocorrência de um evento em um objeto gerenciado. Essa função é capaz de identificar o(s) destinatário(s) de cada relatório de evento, de permitir que a discriminação e o encaminhamento possam ser inicializadas, terminadas, suspensas ou reinicializadas através de seu estado administrativo e que os atributos possam ser lidos e modificados.

Como explicado no ITU (1993), o gerenciamento de relatório de eventos é responsável por:

- Iniciar o encaminhamento de eventos;
- Terminar o encaminhamento de eventos;
- Suspender o encaminhamento de eventos;
- Reiniciar o encaminhamento de eventos;
- Modificar condições de encaminhamento de eventos;
- Restaurar condições de encaminhamento de eventos.

4.7. Discriminador

O discriminador é uma superclasse básica de objetos que pode ser especializado em subclasses para especificar classes de objetos de suporte à gerência, que permitam o controle de várias funções do sistema de gerenciamento de sistemas (ITU, 1993, SPECIALSKI et al., 1993). É capaz da fazer uma seleção do que será enviado a diante.

As condições especificadas pelo discriminador são (ITU, 1993):

- Identificar os pacotes de programação que determinam quando a emissão deve ocorrer;
- Adotar critérios de discriminação;
- Estados administrativos, de utilização e operacional do discriminador;
- Condições específicas para cada subclasse particular do objeto discriminador.

Como descrito no item anterior, o objeto gerenciado discriminador permite a um outro sistema exercer controle sobre as operações e notificações.O discriminador possui estados administrativos, operacional, utilização, status de disponibilidade e outros atributos.

4.7.1. Construtor Discriminador

O construtor discriminador está presente no discriminador e é responsável por selecionar quais objetos de entrada serão repassados (age como um filtro). É um conjunto de uma ou mais asserções sobre a presença de valores de atributos. No caso de apresentar mais de uma asserção, esta deve ser agrupada, usando os operadores lógicos (*AND, OR*). Quando o teste na asserção do valor de atributo for falso (*FALSE*), significa que a asserção está presente no valor do atributo contido no construtor discriminador e ausente no objeto de entrada.

Para realizar o serviço de filtragem, o construtor de discriminador pode testar condições específicas de igualdade e desigualdade, a presença de atributos e a ausência de qualquer uma dessas condições (SPECIALSKI et al., 1993).

Se um construtor de discriminador for vazio, então serão considerados como verdadeiros (*TRUE*) todos os atributos dos objetos de entrada do discriminador, pois não será possível especificar as condições para filtragem.

4.7.2. Atributos do discriminador

A classe discriminador apresenta alguns atributos que serão detalhados (ITU, 1993):

- Identificador do Discriminador (*Discriminator ID*): este atributo é usado para identificar uma única instância do discriminador;
- Construtor Discriminador (*Discriminator Construct*): especifica os testes sobre as informações que devem ser processadas pelo discriminador;
- Estado Administrativo (*Administrative State*): representa o estado administrativo do discriminador e pode assumir os valores: bloqueado (*locked*) e desbloqueado (*unlocked*);
- Estado Operacional (*Operarational State*): representa a capacidade operacional do discriminador em executar suas funções e pode assumir os valores: habilitado (*enable*) e desabilitado (*disable*);

Se um discriminador estiver com o estado administrativo desbloqueado, o sistema aberto pode encaminhar relatórios de eventos para o destino especificado. A mudança de estado administrativo é resultado da ação de um sistema de gerenciamento ou de uma atividade local, e, quando isso ocorrer, o discriminador deve emitir uma notificação. Quando o discriminado encontra-se no estado operacional habilitado, significa dizer que está em operação e, caso contrário, está inoperante.

4.7.3. Pacotes de Programação

Esses pacotes permitem que os discriminadores tenham a habilidade de trocar automaticamente condições, de *reporting-on* para *reporting-off* e, no caso de não estarem presentes no discriminador, será considerado a condição de *reporting-on*. Segundo (SPECIALSKI et al., 1993, ITU, 1993) são definidos três pacotes de programação e somente um pode ser associado a qualquer instância do discriminador:

1) Pacote de Programação Diária (*Daily Scheduling Package*): esta opção permite a programação de relatórios com uma periodicidade de vinte e quatro horas. Possui um atributo chamado de intervalos do dia (*intervals of day*), que define uma lista de intervalos de tempo (início e fim dos intervalos de tempo do dia). Os discriminadores que tiverem com a condição *reporting-on* receberão a notificação já os que tiverem em *reporting-off,* não serão notificados por representarem os intervalos excluídos. O valor padrão é um único intervalo de vinte e quatro horas;
2) Pacote de Programação Semanal (*Weekly Scheduling Packge*): esta opção permite a programação de relatórios com a periodicidade de uma semana. Possui os seguintes atributos: StartTime (horário do início), StopTime (horário do fim) e WeekMask (máscara semanal) que é formada por DaysOfWeek (dias da semana) e IntervlsOfDay (intervalos do dia);
3) Pacote de Programação Externa (*External Schedular Sheduling Package*): permite que a programação de relatórios seja definida por um objeto gerenciado, por um programador externo que tem a capacidade de modificar as condições do discriminador através de mensagens. Possui um atributo chamado SchedularName (nome do programador), que especifica o nome do objeto gerenciado e o programador.

4.7.4. Discriminador de Repasse de Eventos

O discriminador de repasse de eventos permite a especificação das condições que devem ser atendidas pelos relatórios de eventos potenciais, antes que estes sejam enviados a um destino particular. Esta é uma subclasse da classe objeto discriminador, e que possui alguns atributos herdados da classe discriminador[4] e outros mais que são:

- Endereço Destino (*Destination adress*): especifica o(s) endereço(s) primário, que pode ser uma entidade de aplicação ou um grupo de endereços;
- Listas de Endereços *Backup* (*Backup adress list*): é uma lista ordenada de endereços *backups* para serem usados no caso de falha do endereço primário;
- Endereço Ativo (*Active adress*): especifica o endereço da Entidade de Aplicação para onde os eventos serão enviados pelo discriminador;

[4] Apresentado no item 4.8

4.8. Serviços

Os serviços (operações) que podem ser aplicados a cada instância de um discriminador de repasse de eventos são:

- Criação de Relatório de Repasse de Eventos;
- Eliminação de Relatório de Repasse de Eventos;
- Modificação de valores de atributos;
- Suspensão e Retomada de atividade de discriminação.

4.9. Considerações Finais

A arquitetura de gerenciamento Internet usa um sistema de gerência de rede centralizado, onde o gerente fica responsável pela operação de *polling*. Caso descentralize, cada gerente é capaz de detectar problemas com mais rapidez e, com isso, tentar resolver as falhas de forma mais eficiente (THOTTAN & JI, 1998).

O modelo OSI, através do discriminador de repasse de eventos, permite a descentralização das atividades de gerenciamento. Além disso, possui a facilidade do agente escolher os destinatários e o que enviar para os gerentes.

A proposta apresentada no próximo capítulo é empregar o discriminador de repasse de eventos no modelo SNMP, possibilitando filtrar os *traps* que serão enviados e qual destinatário receberá a notificação (de acordo com a sua área funcional).

5. Solução Proposta

Este capítulo apresenta a proposta de um modelo para descentralização da arquitetura de gerenciamento Internet, a metodologia empregada, a arquitetura do sistema, tanto interna quanto externa e os ambientes de testes usados para implementação, validação e resultados.

5.1. Modelo

O modelo proposto é composto por 3 (três) entidades básicas: o agente SNMP, o gerente SNMP e o *Software* Discriminador de Repasse de Eventos (SDRE). Este último age de forma transparente (como um intermediador) entre as duas outras, sendo visto como um gerente pelo agente, e como um agente pelo gerente SNMP. A Fig. 5.1 ilustra o modelo proposto, onde os agentes enviam notificações para um ou mais SDRE, que são responsáveis pela filtragem dos *traps* e o posterior repasse ou não para o(s) gerente(s) responsável(is).

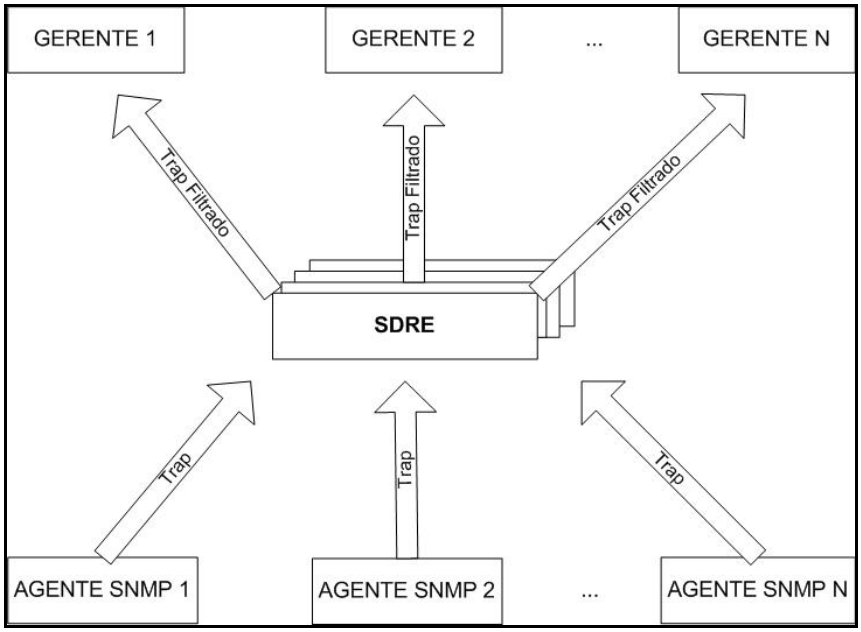

Figura 5.1 - Modelo proposto

Através da implementação do SDRE, conseguiu-se um melhor desempenho em ambientes com diversos equipamentos de interconectividade, *hosts*, serviços e servidores, pois houve redução na quantidade de *traps* trafegando pela rede (diminuindo o consumo da largura de banda), *overhead* na(s) NMS(s) e os gerentes conseguem responder as tomadas de decisões com menor tempo, pois são informados somente de

relatórios de eventos relacionados a sua área de atuação. O *software* possui as seguintes características e funcionalidades:

- Interface gráfica de fácil entendimento e funcional;
- Independência de plataforma (Ex: Windows, Linux, Unix);
- Base de regras para a filtragem e direcionamento dos *traps*;
- Flexibilidade para inserção de regras para discriminação;
- Transparência para os componentes SNMP;
- Facilidade de implementação; e
- Desempenho satisfatório.

O ideal seria que o discriminador estivesse incorporado no agente de cada dispositivo gerenciável, conforme ilustrado na Fig. 5.2. Entretanto, é impossível alterar tal código, pois o mesmo já vem programado de fábrica, pelo fabricante. Logo, a solução apresentada é útil para provar a viabilidade da pesquisa e também pode ser aplicada em ambientes, atualmente, em execução, onde não é possível a troca de dispositivos. Para isso, precisa ser instalado o SDRE e com poucas configurações tanto nele como nos agentes.

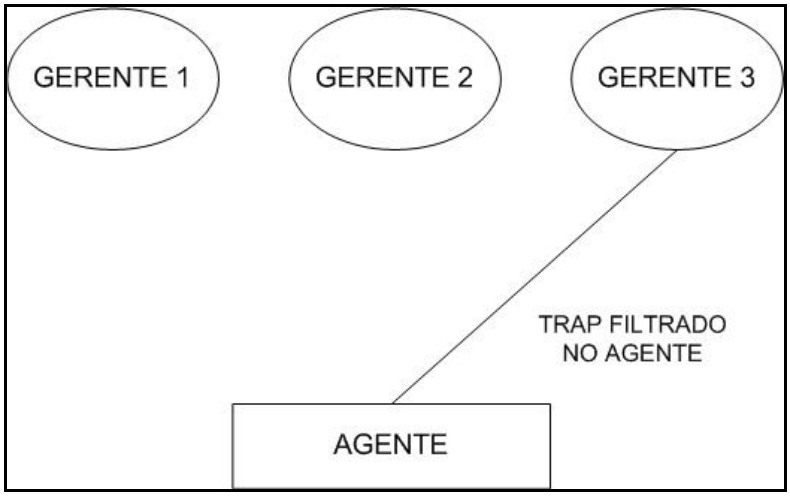

Figura 5.2 - Modelo onde o agente já possui um discriminador implementado

5.2. Metodologia

Para a realização do trabalho, foi preciso fazer uma revisão bibliográfica do conteúdo, estudando, principalmente, o SNMP, a norma OSI X-734 e as pesquisas realizadas, envolvendo o assunto. Além disso, foi necessário:

- Implementação de até 5 (cinco) gerentes, estes são destinatários dos *traps* discriminados, com o intuito de validar a descentralização;

- Implementação de vários agentes gerenciáveis que notificam o SDRE quando algum limite pré-estabelecido é ultrapassado;
- Implementação de um *software* (SDRE) com as funcionalidades de um Discriminador de Repasse de Eventos, adaptadas ao ambiente SNMP, tendo como função a recepção e posterior encaminhamento de *traps*, de acordo com regras pré-estabelecidas;
- Implementação de um módulo para geração dos *traps* em quantidade pré-definida, objetivando o teste do SDRE em altas taxas de envio de alarmes;
- Implementação de um *software* receptor dos *traps* para imprimir os conteúdos destes, contabilizar tempos e a quantidade recebida;
- Instalação da linguagem de programação JAVA J2SDK 1.4 e API AdventNet 1.3.3; e
- Análise do desempenho do SDRE, através de um teste de carga.

É utilizada a linguagem de programação Java, desenvolvida pela Sun Microsystems, para a implementação do *software*. Diversos autores e fabricantes de sistema de gerenciamento utilizaram Java para o desenvolvimento de aplicações, envolvendo SNMP, obtendo êxito e desempenho satisfatório (VELLOSO et al, 2002, LI et al, 2001, GUIAGOUSSOU, 2001). E segundo VENÂNCIO NETO et al (2002), BATES (2002), FERIDUM et al (1999) os motivos que tornam esta linguagem atraente são:

- Independência de plataforma;
- Linguagem de programação altamente difundida no mundo;
- Robustez;
- Existência de diversas API's abertas SNMP;
- Desempenho satisfatório;
- Documentação de fácil acesso;

5.3. Arquitetura Sistema

O presente trabalho visou a construção, implementação e validação de um *Software* Discriminador de Repasse de Eventos que utiliza as funcionalidades da arquitetura OSI em ambientes SNMP, tendo como objetivo descentralizar o modelo

Internet, enfatizar a técnica de Relatório de Eventos e proporcionar uma maior eficiência no gerenciamento pró-ativo. Apesar de se propor um modelo de gerência distribuído, com a presença de um ou mais SDRE (dependendo da necessidade do administrador) na rede, o modelo é capaz de enviar *traps* para diversos gerentes distribuídos, a adoção do *software* elucida a visão de um modelo centralizado para os dispositivos gerenciados. Porém uma importante diferença é a capacidade de enviar notificações para uma grande quantidade de NMS.

O SDRE utiliza a versão 1 do protocolo SNMP e para um melhor entendimento do funcionamento do mesmo, optou-se por dividir a explicação em duas partes: Arquitetura Interna e Externa. A Fig. 5.3 ilustra o funcionamento do mesmo, onde tem-se:

- 1: Dispositivos gerenciáveis configurados para enviar notificações;
- 2: As notificações antes de passarem pelo SDRE são consideradas Relatórios de Eventos Potenciais e somente depois de submetidas às regras podem se tornarem ou não Relatórios de Evento de fato;
- 3: Ao receber o *trap* o SDRE verifica os valores do campos *Generic* e *Specific Trap*;
- 4: Os dados são comparados com as valores residentes na base de dados;
- 5: As informações sobre o repasse do *trap* extraídas são repassadas;
- 6: O sistema pode gerar um mensagem de erro ou descartar o *trap* caso essa opção tenha sido habilitada pelo administrador;
- 7: A mesma PDU pode ser um objeto de entrada para outro discriminador;
- 8: Após o refinamento, a notificação é considerada um Relatório de Evento; e
- 9: São o(s) gerente(s) que irão receber o(s) *trap*(s).

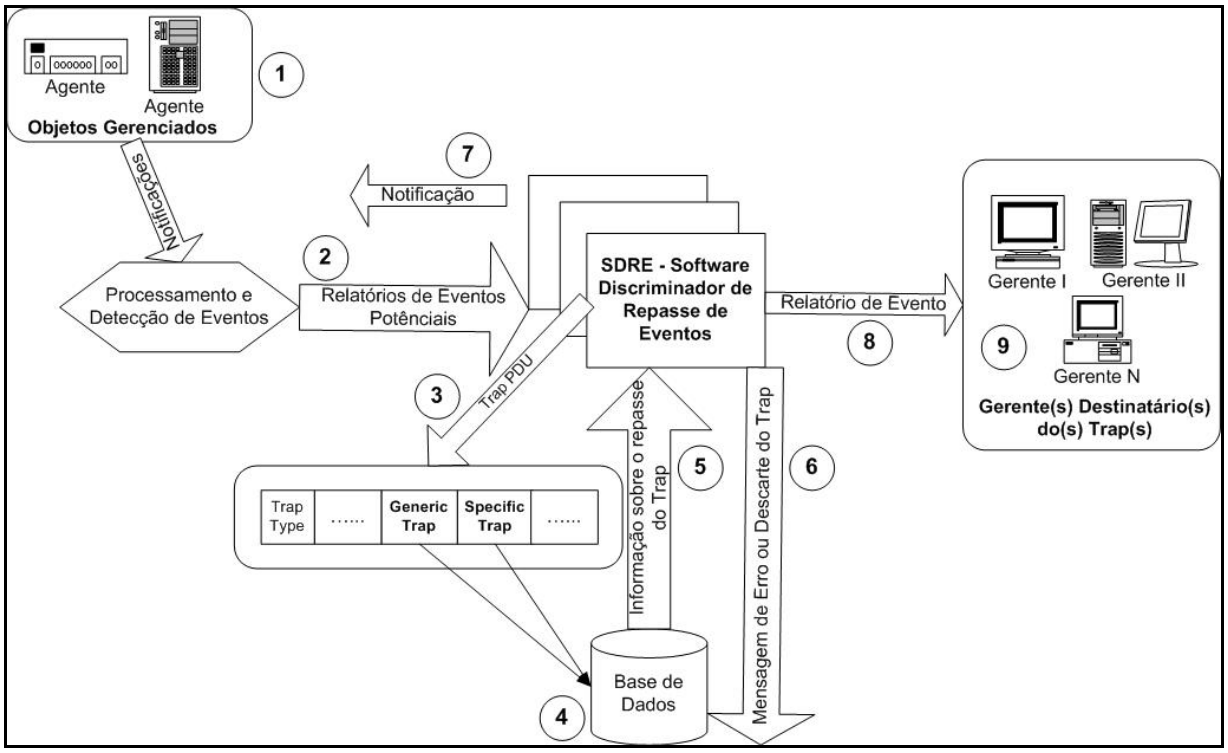

Figura 5.3 - Funcionamento do Sistema

5.3.1. Arquitetura Interna

Este item apresenta o funcionamento interno do SDRE desenvolvido em Java, tal como, a criação e relacionamento das classes, a pesquisa realizada pela base de regras na memória e a API utilizada para o desenvolvimento do mesmo.

Para a elaboração desta arquitetura foi preciso o estudo aprofundado e instalação da biblioteca AdventNet SNMP API 1.3.3, disponibilizada, gratuitamente, pela empresa AdventNet (ADVENTNET, 2002), possibilitando operações sobre objetos gerenciáveis SNMP. Com isso, foi possível obter informações sobre os campos da PDU *Trap* bem como reconstruir a mesma para um posterior reenvio ao(s) gerente(s).

5.3.1.1. Implementação da Classe Projeto

Esta classe foi implementada na linguagem Java por motivos já apresentados e, ainda, por suportar o protocolo de gerenciamento de rede SNMP, através da importação da biblioteca AdventNet SNMP 1.3.3. API, suporta *multithreads* para acesso simultâneo, possibilitando respostas rápidas e aumento na performance. O grande obstáculo encontrado está na dificuldade de manipulação dos objetos e métodos, que são de difícil compreensão e, ainda as diversas variáveis, do tipo *private* e que só podem

ser acessadas pela própria classe, não possibilitando serem utilizadas em outras situações.

O objetivo desta classe é receber a PDU *Trap* cujo padrão é utilizar a porta 162 (sem dificuldades pode ser modificada), verificar se os campos estão preenchidos corretamente e se a comunidade é válida. Posteriormente, através de operações do tipo *get*, conseguir obter os valores dos campos *generic trap type* e *specific trap type*, e realizar uma comparação com a base de dados residentes em memória, através de lista ligada e de acordo com essas informações executar uma ação que pode ser:

- Descartar *Trap*: Quando não existe nenhum gerente cadastrado para receber a notificação ou se a opção "descartar *trap*" tiver sido ativada;
- Apresentar uma mensagem de erro: Quando houver algum problema no tratamento dos dados é impresso na tela uma mensagem de erro e o *trap* é descartado;
- Repassar o *Trap* para um ou mais gerentes: Quando os valores apresentados nos dois campos correspondem a um ou mais gerentes que deve(m) receber relatório de eventos; e
- Repassar para outro SDRE: O *trap* deve ser encaminhado a outro SDRE para uma seleção mais específica.

Nas duas últimas situações, a PDU é modificada e, através do método *SnmpTarget*, o pacote é reenviado na porta solicitada (162) com os campos alterados para os respectivos destinatários. Devido a existência do método não é necessário a criação de um *socket* para enviar a PDU, a própria classe já se encarrega desta função de forma otimizada. Além disso, para o controle e validação de desempenho foi utilizado um contador com o intuito de verificar se a quantidade de *traps* recebidos é a mesma do resultado da soma de enviados e descartados.

Para facilitar o entendimento o quadro 5.1 apresenta parte do código da classe e seus respectivos comentários:

Quadro 5.1 - Código da classe Projeto

```
if (pdu.getCommand() == api.TRP_REQ_MSG) { //obtem os valores
      d = pdu.getTrapType(); // armazena o valor do trap genérico
      h = pdu.getSpecificType();// armazena o valor do trap específico
      a = Integer.toString(d); // transforma o número inteiro para
string
      b = Integer.toString(h); // transforma o número inteiro para
string

      dh.load(); // executa a classe DataHandler
      ips = dh.search(a,b); //Procura o numero Generico e Especifico
```

```
do trap
      System.out.println("generic: "+d+" specific: "+h); // Imprime na
tela para controle o numero Generico Especifico do Trap
      cont++;
      System.out.println("Qtd:"+cont); // Imprime a quantidade de
traps
      i=0;

      // O conteúdo do while é responsável por enviar o trap recebido
para os vários gerentes correspondentes.
     // A classe SnmpTarget se encarrega de enviar um datagrama para o
host e porta especificados nos métodos setTargetHost() e
setTargetPort(), não havendo a necessidade de se criar um
DatagramSocket para fazer o envio.

     while(!ips[i].equals("\n")){ // faz a busca até final do arquivo

     try {  // Usa o método SnmpTarget para enviar o trap com os
valores específicos

     target = new SnmpTarget(); //Cria o objeto target da classe
SnmpTarget

     target.setTargetHost(ips[i]); // O(s) host(s) que irão receber
o(s) trap(s)

     target.setTargetPort(162); // Porta por onde será enviada o trap

// Dados da API
     String[] IdList = new String[pdu.getVariableBindings().size()];
                for (int j=0; j<pdu.getVariableBindings().size(); j++)
                   IdList[j] = pdu.getObjectID(j).toString();
                target.setObjectIDList(IdList);

// Armazena o valor do agente que enviou o trap

     String IpOrigem = pdu.getRemoteHost();

// Dados da API
     SnmpVar[] snmpVar = new
SnmpVar[pdu.getVariableBindings().size()];
                for (int j=0; j<pdu.getVariableBindings().size(); j++)
                   snmpVar[j] = pdu.getVariable(j);

// Reenvia o trap com os dados necessários: Enterprise, Agent Address,
Generic Trap, Specific Trap, Time Stamp e Variable Binding;

     target.snmpSendTrap(pdu.getEnterprise(),IpOrigem , d, h,
pdu.getUpTime(), snmpVar);

// Tempo para enviar o trap em ms
     Thread.sleep(5);

     } catch (Exception e) {
                System.err.println("Erro Enviando a
Trap:"+e.getMessage());
             }
```

```
            System.out.println(ips[i++]); // Imprime o(s) endereço(s)
IP(s) que irão receber os Traps
    }
  }
     else
        System.err.println("Nenhuma PDU Trap foi recebida.");
```

5.3.1.1. Implementação da Classe DataHandler

A classe DataHandler é responsável pela manipulação das informações, pela criação do arquivo “trap.dat” e “trap.new”, pela inserção dos gerentes que irão receber as notificações, pela criação da lista encadeada, pelo método de busca e das condições para exclusão das informações.

Os campos do arquivo trap.dat são: *generic* (valor do *trap* genérico), *specific* (valor do *trap* genérico), IP (endereço IP dos gerentes destinatários do *trap*) e descarte (ativando esse campo as notificações são desconsideradas). Ao inicializar o SDRE os dados pertencentes a este arquivo são colocados na memória de forma aleatória (lista encadeada), segundo DEITEL (2001), obtendo um acesso rápido e grande desempenho para esse tipo de situação. Os dados são armazenados em forma de *string* e para a manipulação precisam ser “quebrados”, utilizando métodos específicos da classe Java.io. O quadro 5.2 ilustra parte do código usado.

Quadro 5.2 - Parte do código da classe DataHandler

```
// Grava nos dados no arquivo trap.dat
    public boolean write(String generic, String specific, String ip,
String descarte){
        try{
//Escrever no arquivo os campos
            FileWriter out = new FileWriter(fileName, true);
//Escrever
            out.write(generic, 0, generic.length());
            out.write(' ');
            out.write(specific, 0, specific.length());
            out.write(' ');
            out.write(ip, 0, ip.length());
            out.write(' ');
            out.write(descarte,0,descarte.length());
            out.write('\n');
            out.close();
        }catch(IOException e){// Se houver falha apresenta mensagem de
erro
            System.out.println("Fim de Arquivo");
            return false;
        }
        return true;
```

Para facilitar o trabalho do administrador/usuário do sistema, a exclusão pode ser feita através de quinze diferentes condições:

- 1 - Seleção do campo “Número do Trap Genérico”: exclui todos os *traps* cujos campos “Número do Trap Genérico” correspondem ao valor informado;
- 2 - Seleção do campo “Número do Trap Genérico” e “Número do Trap Específico”: exclui todos os *traps* correspondentes à asserção dos dois campos;
- 3 – Seleção do campo “Número do Trap Genérico” e “Número do Trap Específico” e “IP do Gerente”: exclui todos os *traps* correspondentes à combinação dos três campos;
- 4 – Seleção do campo “Número do Trap Genérico” e “Número do Trap Específico” e “IP do Gerente” e “Descartar”: exclui todos os *traps* correspondentes à combinação dos quatro campos;
- 5 - Seleção do campo “Número do Trap Específico” e “IP do Gerente” e “Descartar”: exclui todos os *traps* correspondentes à combinação dos três campos;
- 6 - Seleção do campo “IP do Gerente” e “Descartar”: exclui todos os *traps* correspondentes à combinação dos dois campos;
- 7 - Seleção do campo “Descartar”: exclui todos os *traps* que possuem a opção de descartar;
- 8 - Seleção do campo “Número do Trap Genérico” e “IP do Gerente” e “Descartar”: exclui todos os *traps* correspondentes à combinação dos três campos;
- 9 - Seleção do campo “Número do Trap Genérico” e “Descartar”: exclui todos os *traps* correspondentes à combinação dos dois campos;
- 10 - Seleção do campo “Número do Trap Genérico” e “Número do Trap Específico” e “Descartar”: exclui todos os *traps* correspondentes à combinação dos três campos;
- 11 - Seleção do campo “Número do Trap Específico” e “IP do Gerente” e “Descartar”: exclui todos os *traps* correspondentes à combinação dos três campos;
- 12 - Seleção do campo “Número do Trap Específico” e “Descartar”: exclui todos os *traps* correspondentes à combinação dos dois campos;
- 13 - Seleção do campo “Número do Trap Genérico” e “IP do Gerente”: exclui todos os *traps* correspondentes à combinação dos dois campos;

- 14 - Seleção do campo “Número do Trap Genérico” e “IP do Gerente”: exclui todos os *traps* correspondentes à combinação dos dois campos; e
- 15 - Seleção do campo “IP do Gerente”: exclui todos os *traps* que correspondem ao valor informado.

O quadro 5.3 apresenta parte do código utilizada para exclusão dos campos do arquivo trap.dat.

Quadro 5.3 – Parte do código da classe DataHandler

```
// Condições para exclusão do(s) campo(s) do arquivo trap.dat

// Condição 1.
    if((del.txfntrapGen.isEnabled()) &&
(!(del.txfntrapSpec.isEnabled())) &&
(!(del.txfiptraps.isEnabled()))&&(del.btdescarte.isSelected()==false))
      {
             if((this.generic.equals(del.txfntrapGen.getText())))
                 { log = true;}
       }

// Condição 15
if(!(del.txfntrapGen.isEnabled()) && (!(del.txfntrapSpec.isEnabled()))
&& ((del.txfiptraps.isEnabled()))
&&(del.btdescarte.isSelected()==false))
      {
                if((this.ip.equals(del.txfiptraps.getText())))
                       { log = true;}
      }

   if(log == false) //Nessa condição os dados são gravados
      {
       d.write(this.generic,this.specific,this.ip,this.descarte);
      }
        log = false; // a variável recebe novamente o valor falso
        }
```

5.3.1.1. Implementação das Classes SDRE, MenuForm, DeletaTrap, InterTraps, DataConsul

Estas classes estão relacionadas com a criação dos quadros (*frames*), disposição física de cada texto, botão, caixa de texto, tamanho e execução de outras classes. São usadas diversas variáveis e objetos, com funções específicas e foram importadas classes nativas da linguagem Java.

A classe principal (*main*) é a SDRE cuja finalidade é a exibição da mesma, conforme apresentada no quadro 5.4. A *MenuForm* é responsável pela criação do *Menu*, assim distribuído: Discriminador, Cadastra, Excluir, Consultar e Sair. Já a *DeletaTrap* tem como finalidade a construção do *frame* e dos objetos pertencentes a opção

"Exclusão dos Traps e Gerentes". A *InterTraps* constrói a janela e os dados de "Cadastra Traps e Gerentes". Em *DataConsul* é realizada uma busca dos dados no arquivo trap.dat e exposto na tela, em forma de tabela.

Quadro 5.4 – Código da classe SDRE

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
import java.lang.*;
public class SDRE extends JFrame  // Classe SDRE
{
   private Color colorValues[] = { Color.black, Color.blue, Color.red,
Color.green , Color.gray };  // Cores da Tela
   static Dimension screenSize; //Tamanho da tela

   public SDRE()
   {
      super( "SDRE - Software Discriminador de Repasse de Eventos" );
// Título
      MenuForm bar = new MenuForm();  // Executa o menubar - MenuForm
      this.setJMenuBar( bar );  // seta menubar para o JFrame
      this.getContentPane().setBackground( Color.gray );
      screenSize = Toolkit.getDefaultToolkit().getScreenSize();
      this.setSize( 640, 480 ); // A tela terá o tamanho 640 X 480
      Dimension frameSize = getSize();
      setLocation(screenSize.width/2-frameSize.width/2,
      screenSize.height/2-frameSize.height/2);
      this.show();
   }

   public static void main( String args[] )
   {
      SDRE app = new SDRE();
      app.addWindowListener
      (
         new WindowAdapter()
         {
            public void windowClosing( WindowEvent e )
            {
               System.exit( 0 );
            }
         }
      );
   }
}
```

5.3.1. Arquitetura Externa

Neste item é detalhado o funcionamento externo do SDRE o qual consiste em janelas, botões, campos e funções do mesmo. Ao executar o *software*, aparecerá a tela inicial apresentada na Fig. 5.4 em tamanho 640 X 480 *pixel*, com o título Software Discriminador de Repasse de Eventos e cinco opções no *menu*, que são: Discriminador, Cadastrar, Excluir, Consultar e Sair.

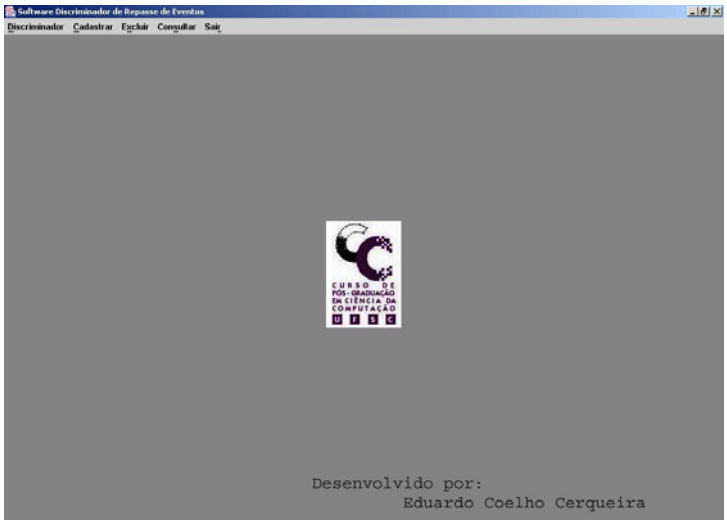

Figura 5.4 - Tela Inicial

Ao ser clicado com o *mouse* a opção “Discriminador”, usando a tecla de atalho “Alt + D” ou “Tab” aparecerá o *sub-menu* “Iniciar” e se o mesmo for acionado, o SDRE começa a receber os *traps* e executar as funções descritas anteriormente.

Em “Cadastrar”, tem-se um *sub-menu* “Cadastrar Traps e Gerentes” e, ao ser selecionado, aparecerá a tela ilustrada na Fig. 5.5 com informações como:

- Número do Trap Genérico: O usuário deve informar o número do *trap* genérico;
- Número do Trap Específico: O usuário deve informar o número do *trap* específico;
- IP do Gerente: O endereço IP do gerente que receberá o *trap* correspondente às informações citadas anteriormente;
- Descartar Trap: Situações onde o *trap* não deve ser encaminhado, então o mesmo é descartado;
- Botão Cancelar: Cancela a operação e volta à tela anterior;
- Botão OK: Os dados serão gravados no arquivo trap.dat;
- Botão Limpar: Limpa os campos preenchidos;

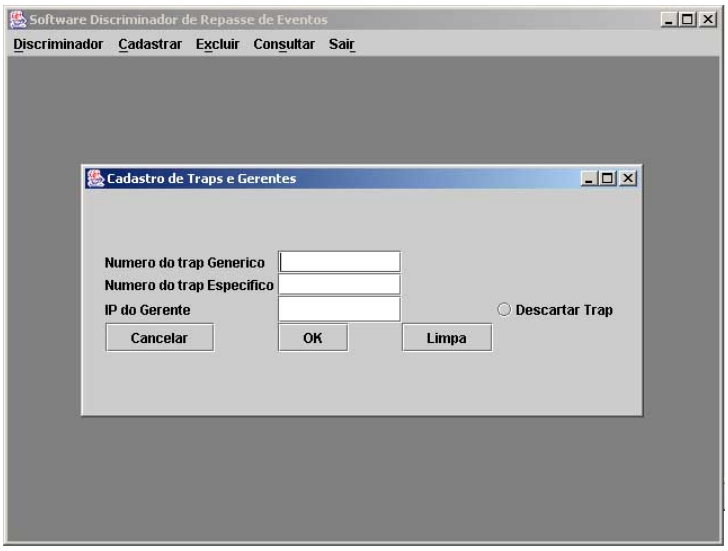

Figura 5.5 - Janela Cadastra Traps e Gerentes

A opção “Excluir” do *menu* apresenta a tela mostrada na Fig. 5.6, possibilitando a exclusão dos campos do arquivo trap.dat e possui quinze diferentes tipos de remoção[5]. A janela chamada “Deletar Traps e Gerentes” possui as seguintes funcionalidades:

- *RadioButton* Número do Trap Genérico: ao ser acionado, permite ao usuário informar o número do *trap* genérico;
- *RadioButton* Número do Trap Específico: quando selecionado, possibilita o digitar número do *trap* específico;
- *RadioButton* IP do Gerente: ao ser clicado, admite que o endereço IP do gerente que receberá o *trap* seja preenchido;
- Descartar Trap: Descarta o *trap* com as informações mencionadas antes;
- Botão Cancelar: Cancela a operação e volta a tela anterior;
- Botão OK: Elimina o(s) dado(s) selecionado(s) do arquivo trap.dat;
- Botão Limpar: Limpa os campos preenchidos.

[5] Apresentado no item 5.3.1.1

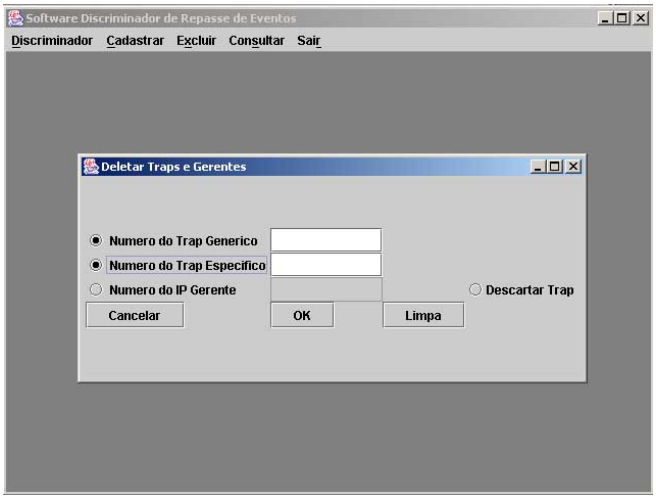

Figura 5.6 - Excluir Traps e Gerentes

A opção “Consultar” imprime na tela os dados em forma de tabela, pertencentes ao arquivo trap.dat. Permite que o usuário do *software* tenha um controle das informações contidas no arquivo. A coluna da tabela é formada pelos campos: Genérico, Específico, IP e Descarte. A Fig 5.7 mostra a janela Relatório dos Traps. O último título da janela principal é “Sair” que ao ser escolhido termina o SDRE.

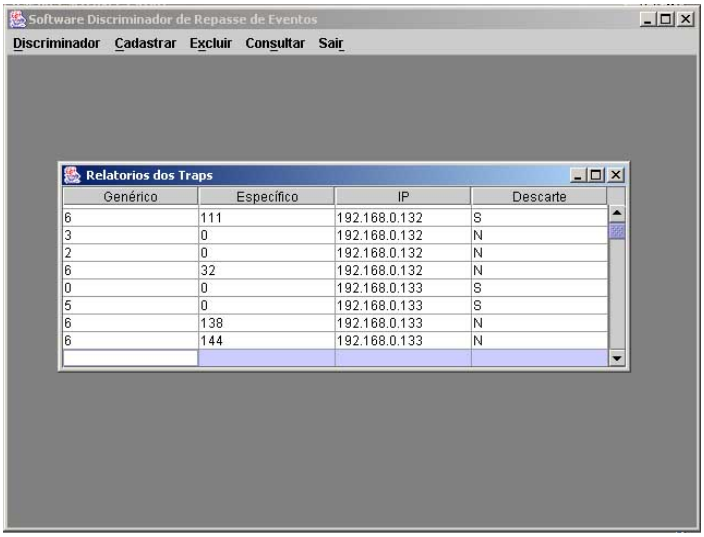

Figura 5.7 - Relatório dos Traps

5.4. Implantação, Validação e Resultado

O presente trabalho foi organizado em duas etapas: a primeira realizada em um ambiente de teste, onde se buscou gerar *traps* em uma rede experimental, no Laboratório de Redes de Computadores e Sistemas Distribuídos – LRCSD, da Fundação do Ensino Superior de Rio Verde – Fesurv, a fim de simular possíveis situações reais e,

uma grande quantidade de notificações como forma de validar o SDRE. A segunda utilizou um ambiente em produção do NPD (Núcleo de Processamento de Dados) da UFSC (Universidade Federal de Santa Catarina), concentrando diversos equipamentos gerenciáveis; vários fabricantes diferentes; e um alto tráfego de pacotes.

Na tentativa de verificar a eficácia do *software,* utilizou-se a fórmula matemática a seguir: **∑ TE = ∑ TR = ∑ TD**, onde ∑ TE representa o(s) número(s) de *trap(s)* enviado(s) pelo(s) dispositivo(s) de rede, ∑ TR equivale ao número de *trap(s)* recebido(s) pelo SDRE e ∑ TD significa número de *trap(s)* discriminado(s).

Em todas as situações usadas, o *software* foi utilizado simultaneamente com outras aplicações da máquina, não sendo o(s) *host*(s) de uso exclusivo, por exemplo, eram executados editores de texto, navegadores *web*, antivírus, aplicações gráficas e etc.

5.4.1. Ambiente de Testes Simulado

Para o primeiro ambiente no LRCSD – Fesurv foi montada uma rede de computadores para gerar uma grande quantidade de alarmes e, verificar o comportamento do *software*. O cenário apresentado na Fig. 5.8 é composto pelos seguintes sistemas operacionais e equipamentos:

- Sistema Operacional Linux RedHat 7.3 para a arquitetura I 386;
- Sistema Operacional Linux Slackware 8.1 para a arquitetura I 386;
- Sistema Operacional Windows 2000 Profession SP 3;
- 10 (dez) microcomputadores Pentium III Intel, velocidade de 1.0 GHz, 128 (cento e vinte e oito) Mb de memória RAM, disco rígido com capacidade de 20 (vinte) Gb;
- 2 (dois) servidores Pentium IV Intel, velocidade de 1.4 GHz, 256 (duzentos e cinqüenta e seis) Mb de memória RAM, disco rígido com capacidade de 40 (vinte) Gb; e
- 2 (dois) *Switches* Gerenciáveis.

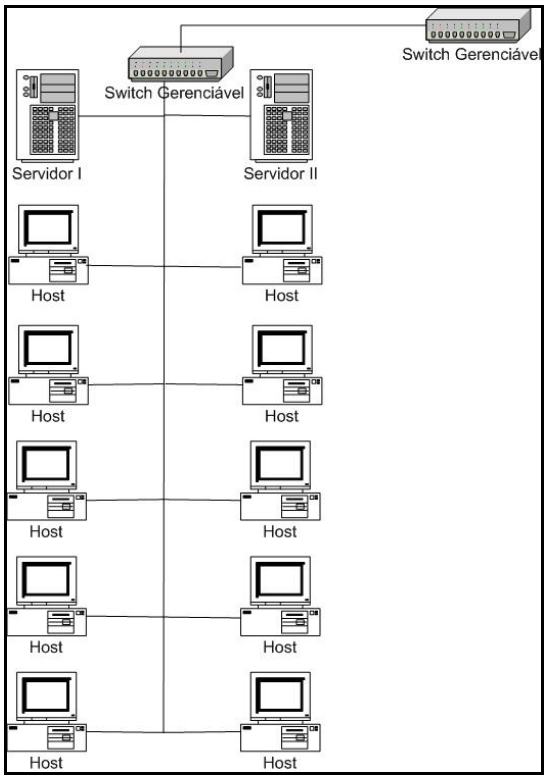

Figura 5.8 - Ambiente de Teste LRCSD / Fesurv

Para conseguir simular uma grande quantidade de *traps* enviados pelos dispositivos gerenciáveis, foi implementado um *software* em Java, utilizando a API AdventNet 1.3.3, onde o mesmo constrói e envia uma PDU *Trap* para um endereço IP com informações especificadas pelo usuário, como mostra parte do código no quadro 5.5. Foi utilizado um contador como forma de contabilizar a quantidade de notificações enviadas para ser aplicada na fórmula matemática descrita no item 5.4.

Quadro 5.5 – Parte do código da classe sendtrap

```
// Os valores que devem ser informados para a criação e envio do trap

    String usage = "sendtrap [-c community] [-p port] -m MIB_files
host enterprise agent-addr generic-trap specific-trap timeticks [OID
value]";
    String options[] = { "-c", "-m", "-p"};
    String values[] = { null,null, null};

// Parte do código

    target.setTargetHost( opt.remArgs[0] );  //indica o nome do
destinatáio
    target.setTargetPort( 162 );  // indica a porta a ser enviada


// Parte do código
```

```
    try {  // Envia o pacote

       for (k=0; k<10; k++) // envia 10 pacotes de uma vez
       {
// Envia o trap com os valores informados

        target.setObjectIDList(oids);
        target.snmpSendTrap(opt.remArgs[1], opt.remArgs[2], trap_type,
                specific_type, time, var_values);

        // Envia a cada 5 ms
        Thread.sleep(5);}

    } catch (Exception e) {
        System.err.println("Error Sending Trap: "+e.getMessage());
    }
```

Para verificar se os *traps* foram discriminados de forma correta, utilizou-se dois *softwares free* (grátis) para o recebimento dos mesmos, onde cada um possuía uma variável, que armazenava a quantidade de notificações recebidas, que são: Trap Receiver versão 5.0 da Network Computing Technologies (TRAP RECEIVER, 2002) e o TrapReceiver construído a partir da API AdventNet 1.3.3. Antes de iniciar os testes, a variável que contabilizava a chegada dos relatórios de eventos era zerada.

Foram discriminados pelo SDRE um total de 31.430 (trinta e um mil quatrocentos e trinta) *traps* no período de 02/12/2002 a 20/12/2002, os mesmos foram enviados por diferentes *hosts* e *switchs* para um número de até 4 (quatro) discriminadores e repassados a uma quantidade de 1 (um) a 5 (cinco) gerentes. Dependendo da associação dos valores dos campos *Generic* e *Especifc Trap* com os gerentes destinatários, algumas notificações foram taxadas como “Descartar Trap” pelo administrador e foram descartadas. Também em algumas situações o *software* analisava a PDU e encaminhava para outro discriminador, como uma forma mais hierárquica de filtro. A tabela 5.1 faz um resumo dos *traps* enviados pelos equipamentos gerenciados, recebidos pelo SDRE e repassados para os gerentes. Os significados dos campos são:

- Data: Data do experimento;
- Q H: Quantidade de horas;
- Q T R S: Quantidade de *traps* recebido(s) pelo(s) discriminador(es) simultaneamente;
- Q D G: Quantidade de dispositivos gerenciados;
- Q SDRE: Quantidade de SDRE na rede;

- G D: Gerente(s) destinatário(s) dos *traps*; e
- Q T T: Quantidade total de *traps* discriminados.

Tabela 5.2 – Resumo das informações obtidas

Data	Q H	Q T R S	Q D G	Q SDRE	GD	Q T T
02/12/02	14	1	1	1	I	18
03/12/02	14	4	4	1	I	72
04/12/02	14	40	4	1	I	720
05/12/02	14	60	6	1	I	1080
06/12/02	14	80	8	1	I	1440
07/12/02	4	80	8	1	I	400
09/12/02	14	80	8	1	I e II	1440
10/12/02	14	80	8	1	I e II	1440
11/12/02	14	80	8	1	I e II	1440
12/12/02	14	70	7	2	I, II e III	2520
13/12/02	14	70	7	2	I, II e III	2520
14/12/02	4	70	7	2	I, II e III	700
16/12/02	14	60	6	3	I, II, III e IV	3240
17/12/02	14	50	5	4	I, II, III e IV	3600
18/12/02	14	50	5	4	I, II, III e IV	3600
19/12/02	14	50	5	4	I, II, III, IV e V	3600
20/12/02	14	50	5	4	I, II, III, IV e V	3600

Para analisar a eficiência do SDRE, em situações onde ocorra o recebimento de *traps* simultâneos, configurou-se cada *software* responsável pelo envio de alarmes num mesmo período de tempo de 10 (dez) notificações para o discriminador a cada 5 (cinco) ms.

A base de dados utilizada era composta por até 500 (quinhentos) números de *traps* distintos e o mesmo poderia ser enviado para um ou mais gerentes. O quadro 5.6 demonstra os dias e a quantidade de informações contidas no arquivo trap.dat.

Quadro 5.6 – Base de Dados

Data	Qtd. Traps na Base de Dados
02 – 07/12/02	100
09 – 14/12/02	300
16 – 20/12/02	500

O(s) gerente(s) destinatário(s) das informações foi(ram) escolhido(s) de acordo com a sua área de atuação e somente recebiam informações necessárias a ele(s). Por exemplo, o gerente de segurança é informado quando alguma tentativa de invasão ocorre na rede, sem necessitar saber se estão acontecendo problemas no desempenho. Para o teste, primeiramente, um gerente era notificado, posteriormente, utilizou-se dois, três, quatro e cinco gerentes destinatários. Observou-se que o SDRE suporta uma grande quantidade de destinatários dos Relatórios de e Eventos, demonstrando-se que a forma descentralizada do modelo não limita o sistema a uma quantidade pequena de NMS para o recebimento dos pacotes, além de possuir um filtro eficaz de informações.

A utilização de vários discriminadores na rede é de opção do administrador. Nos testes, o uso de 1 (um) SDRE não apresentou perda de desempenho, porém optou-se pela utilização de até 4 (quatro) como forma de criar uma hierarquia de discriminadores, aliviando o tráfego e aumentando a tolerância à falhas (caso houvesse uma falha).

5.4.2. Ambiente de Produção

Com o propósito de obter resultados mais próximos da realidade, foi utilizada a infra estrutura da rede UFSC (Universidade Federal de Santa Catarina), PoP-SC (Ponto de Presença em Santa Catarina da Rede Nacional de Pesquisa), RCT2 (Rede Catarinense de Tecnologia), RNP (Rede Nacional de Pesquisa) e RMAV-FLN (Rede Metropolitana de Alta Velocidade – Florianópolis), por se tratar de um ambiente em produção, com uma grande quantidade de tráfego e com ocorrência de pacotes com erros e. Os dispositivos gerenciáveis utilizados foram:

- um *Lan Switch* IBM 8271;
- um Roteador Cisco 7500; e
- um Switch IBM 8265.

A Fig. 5.9 ilustra o ambiente de produção onde foi aplicado o SDRE.

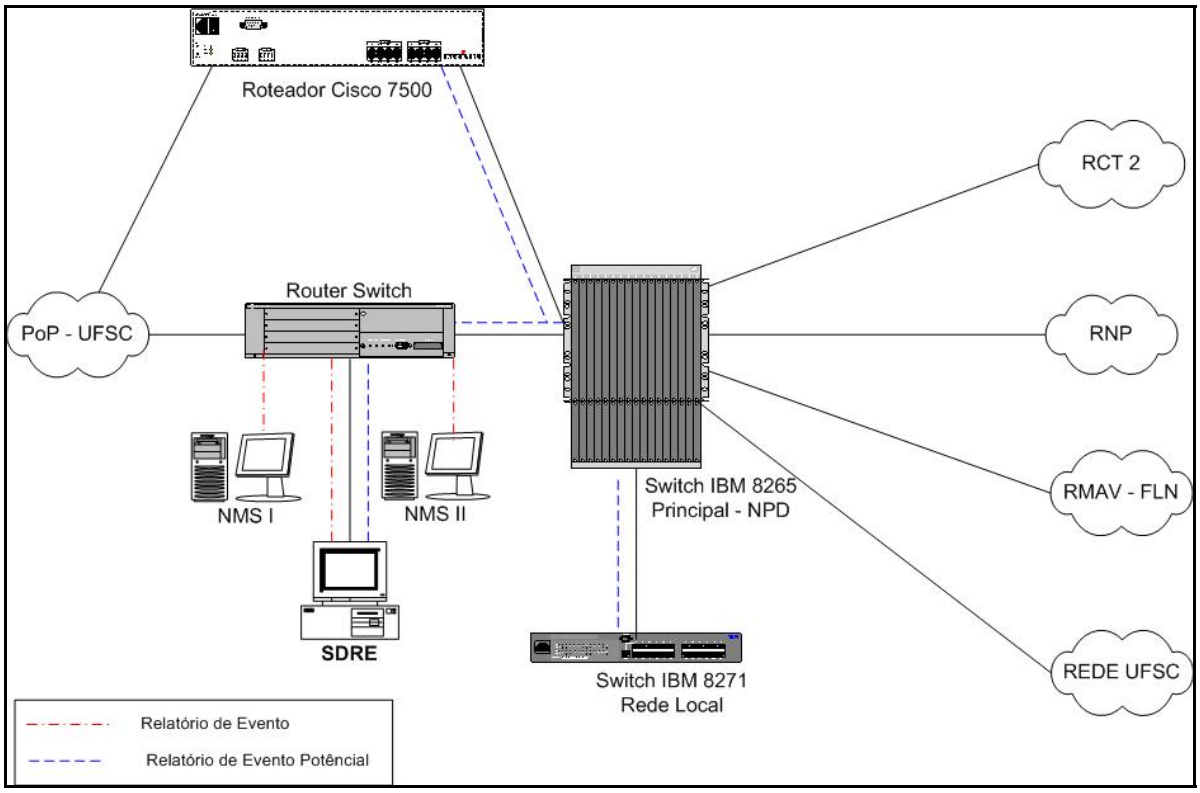

Figura 5.9 - Ambiente de Produção – NPD / UFSC

Os testes foram realizados no período de 20/01/2003 a 27/01/2003, o *host* hospedeiro do SDRE foi um computador Pentium Celeron 400 MHz, 64 MB de memória RAM, disco rígido 10 MB, com o sistema operacional Windows 98. A implementação do *software* discriminador exigiu um mínimo de configuração, isto é, apenas o requisito de que a máquina, onde o mesmo foi instalado, possuísse JVM (*Java Virtual Machine*) e que os agentes fossem configurados para enviar informações para ele.

Nos dois primeiros dias, todas as notificações eram filtradas e enviadas para a NMS-I. Analisando a origem dos alarmes, percebeu-se uma grande quantidade de *traps* referentes à segurança. Então, optou-se pela utilização de mais outra estação de gerenciamento, a NMS-II, onde a mesma era responsável pelo recebimento dos Relatórios de Eventos específicos sobre segurança, podendo, com isso, solucionar problemas de forma mais produtiva e eficiente.

Os gerentes possuíam o sistema operacional Windows 2000 *Profession* e o *software* de gerência WhatsUp Gold 6.0 da Ipswitch. Não houve a necessidade de

utilizar outro SDRE, pois um único conseguiu atender às necessidades esperadas pelo administrador.

O *software* ficou em execução a partir das 17:00 h do dia 20 até 19:00 h do dia 27. Por diversas vezes existiram situações onde a máquina hospedeira precisou ser reiniciada. Por se tratar de um ambiente em execução, algumas alterações não puderam ser realizadas.

Houve 1832 notificações enviadas pelos agentes e tratadas pelo SDRE. Todas foram encaminhadas para seu(s) destinatário(s) final(is) ou descartadas. A base de dados utilizada continha 80 tipos de *traps* diferentes, onde cada um poderia ser descartado ou repassado para uma ou mais NMS. O quadro 5.7 mostra esses resultados, onde:

- Q T R S: Quantidade de *traps* recebido(s) pelo(s) discriminador(es) simultaneamente;
- Q D G: Quantidade de dispositivos gerenciados;
- Q SDRE: Quantidade de SDRE na rede;
- G D: Gerente(s) destinatário(s) dos *traps*; e
- Q T T: Quantidade total de *traps* discriminados.

Quadro 5.7 – Resultados no Ambiente de produção

Q T R S	Q D G	Q SDRE	G D	QTT
1832	3	1	1 e/ou 2	1832

6. Conclusões e Trabalhos Futuros

Este trabalho teve como objetivo o desenvolvimento, a implementação e a validação de um *Software* Discriminador de Repasse de Eventos (SDRE) para a arquitetura Internet, utilizando os conceitos existentes na OSI e descritos na norma X-734 da ITU-T. Através de sua aplicação, foi possível obter uma flexibilidade para inserção de diversos gerentes destinatários dos *traps*, não havendo uma limitação, nesse sentido, como é encontrada atualmente.

Outro ponto importante foi a ênfase aos relatórios de eventos, pois depois de algum tempo coletando e analisando informações na rede, o administrador consegue traçar um perfil do ambiente e saber, por exemplo, os horários de "picos" e as deficiências. A utilização do SDRE diminuiu substancialmente a necessidade de constantemente consumir recursos e tempo com informações desnecessárias (operações de *get*), permitindo que os agentes fossem configurados para enviarem notificações, quando um limite pré-estabelecido fosse ultrapassado.

A solução utilizada propiciou um melhoramento na técnica de *event-report* adotada atualmente pelo modelo SNMP, além de assegurar um melhor desempenho na gerência pró-ativa.

Devido à dificuldade de alteração do código do agente e pelo fato de existirem ambientes em execução, a utilização do *software* discriminador aumenta o desempenho e a flexibilidade da arquitetura SNMP, onde o mesmo é responsável por receber todas as notificações dos agentes, analisá-las e, de acordo com sua base de regras, repassar ou não para a(s) NMS. Com um mínimo de configuração, o discriminador pode ser aplicado em qualquer versão do protocolo (atualmente SNMPv1) e ambiente de rede.

A interface gráfica é simples e funcional, onde o usuário do SDRE não precisa ter um grau de conhecimento elevado para utilizá-la, além disso, foram aplicadas técnicas (*multithreads* e lista encadeada) com o intuito de conseguir um alto desempenho. O discriminador pode ser usado em qualquer sistema operacional que possua JMV e sua base de dados fica armazenada em um arquivo "trap.dat" facilitando seu transporte.

A possibilidade de distribuir a carga de gerenciamento entre diversos gerentes e a filtragem dos *traps* é de grande valia, pois permite que problemas sejam solucionados de maneira mais rápida e eficaz. Além disso, consiste em uma solução para descentralizar a arquitetura SNMP.

Os resultados alcançados com os testes foram satisfatórios e demonstraram que o SDRE pode ser aplicado em qualquer ambiente de rede, sem perda de performance. O teste de carga realizado indica o uso do *software* em situações onde o número de *traps* recebido, simultaneamente, em cada módulo, não ultrapasse duzentos, pois haverá redução do desempenho. Um exemplo da vantagem do discriminador foi encontrado na segunda fase, onde o *Switch* IBM 8271 estava configurado para enviar mensagens a três NMS diferentes, com uma limitação de no máximo sete destinatários, onde os gerentes recebiam sempre o mesmo conjunto de notificações. Com a implementação do SDRE, todos os alarmes eram enviados a ele, filtrados e repassados para as estações de gerenciamento de acordo com a atividade de cada uma. Com isso, diminuiu o tráfego desnecessário de pacotes e as NMS passaram a receber informações de seu interesse, deixando de existir limitação em relação à quantidade de gerentes na rede.

Com base nos estudos realizados e resultados obtidos, neste trabalho, pode-se recomendar alguns trabalhos futuros:

- A construção e implantação de uma MIB Discriminadora, para que os *traps* sejam discriminados nos próprios agentes;
- A implementação de interfaces Web, para que o usuário do SDRE possa manusear o *software* via Internet;
- A utilização de um banco de dados para armazenar as regras e comparar o desempenho com o modelo atual;
- A possibilidade do SDRE suportar os protocolos SNMPv2 e SNMPv3, visando abranger todas as versões.

7. Referências Bibliográficas

ADVENTNET. **AdventNet SNMP API**. Disponível em: http://www.adventnet.com/products/snmp/. Acessado em: 15 de setembro de 2002.

AIZMAN, Alex. **Multi-Management: application-centric approach**. IFIP/IEEE. 6th International Symposium on Integrated Network Management, USA, Boston, Vol. 06, p. 871 – 884, maio de 1999.

ANDREY, Laurent; FESTOR, Oliver; NATAF, Emmanuel et al. **A Java – Based TNM Development and Experimentation Environment**. IEEE Journal on Selected Areas in Communications, VOL.18 , NO.5, p. 664 – 675, maio de 2000.

ANEROUSIS, Nikolaos. **A Distributed Computing Environment for Building Scalable Management Services**. IFIP/IEEE. 6th International Symposium on Integrated Network Management, USA, Boston, Vol. 06, p. 547 – 562, maio de 1999.

BATES, Regis. **Network Management SNMP**. EUA: McGraw-Hill, 2002. p. 575 – 595.

CASE, Jeffrey; FEDOR, Mark; SCHOFFSTALL, Martin et al., J. **Simple Network Management Protocol**. RFC 1157, maio de 1990, 35 p.

CHEIKHROUHOU, M; LABETOULLE, J. **An Efficient Polling Layer for SNMP**. Network Operations and management Symposium, Vol. 07, p. 477 - 490, setembro de 2000.

CLEMENTI, S.; CARVALHO, T. M. B. **Métodos para Especificação e Implementação de Solução de Gerenciamento**. Anais do 17º Simpósio Brasileiro de Redes de Computadores, Salvador BA, 1999.

CS CARE. **Trap Console Hand Book**. EUA, maio de 2002.

DEITEL, Harvey; DEITEL, Paul. **Java, Como Programar**. Porto Alegre: Editora Bookman, 2001, 3° edição. 1201 p.

DOUGLAS, Mauro; SCHMIDT, Kevin**. SNMP Essencial**. Rio de Janeiro: Editora Campus, 2001. 316 p.

DUARTE, Fátima; LOUREIRO, Antonio; MURTA, Cristina; et al. **On The Scalability and Performance Impact of Centralized LAN Management**. VII Workshop de Gerência de Redes e Serviços de Telecomunicações. 20° Simpósio Brasileiro de Redes de Computadores. Rio de Janeiro, p 24-30, maio de 2002.

FERIDUM, M; KASTELEIJN, W; KRAUSE, J. **Distributed Management with Mobile Components**. IFIP/IEEE. 6th International Symposium on Integrated Network Management, USA, Boston, Vol. 06, p. 547 – 562, maio de 1999.

FLATIN, J. **Push vs. Pull in Web-Based Network Management**. IFIP/IEEE 6th International Symposium on Integrated Network Management, USA, Boston, Vol. 06, p. 3 – 18, maio de 1999.

GUIAGOUSSOU, M; BOUTABA, R; KADOCH, M. **A Java API for Advanced Faults Management**. IFIP/IEEE International Symposium on Integrated Network Management, Vol. 08, p. 483 - 498, maio de 2001.

IBM. **Tilovi NetView**. Disponível em: http://www.tivoli.com/products/index/netview/. Acessado em: 25 de julho de 2002.

IETF. **The Internet Engineering Task Force**. Disponível em: http://ietf.org. Acessado em: 24 de julho de 2002.

ITU. **Event Management Function**: Recommendation X.734. 1993. 18 p.

JIAO, Jia; NAQVI, Shamim; RAZ, Danny; et al. **Toward Efficient Monitoring**. IEEE Journal on Selected Areas in Communications, VOL.18 , NO.5, p. 723 – 732, maio de 2000.

KIDSTON, David; ROBINSON, John. **Distributed Network Management for Coalition Deployments**. U.S. Government. MILCOM 2000, vol.1, p. 460-464, setembro de 2000.

LI, H; YANG, S; XI, H; et al. **System Designs for Adaptive, Distributed Network Monitoring and Control**. IFIP/IEEE International Symposium on Integrated Network Management, Vol. 08, p. 77 - 90, maio de 2001.

LOUREIRO, Antônio; NOGUEIRA, José; RUIZ, Linnyer; et al. **Redes de Sensores Sem Fio**. Anais do XXII Congresso da Sociedade Brasileira de Computação. XXI JAI – Livro Texto, Florianópolis - SC, v. 2, p. 193-232, julho de 2002.

McCloghrie, Keith; ROSE, Marshall. **Structure and Identification of Management Information for TCP/IP-based Internets**. RFC 1066, agosto de 1988. 89 p.

MURPHY, Jeff. **Networking Monitoring Software**. Disponível em: http://netman.cit.buffalo.edu/nm-bin/showprods.cgi?db=MONITORING. Acessado em: 28 de julho de 2002

NATACLE, Bob. **WinSNMP v2.0 – Evolution of industry-standard API**. The Simple Times. v 3, n 1, 7-18 p, março de 1998.

OMARI, Salima; BOUTABA, Raouf; CHERKAOUI, Omar. **Policies in SNMPv3-based Management**. IFIP/IEEE International Symposium on Integrated Network Management, USA, Boston, Vol. 06, pp. 797 – 811, maio de 1999.

PANTON,Tim; ARKESTEIJN, Birgit. **Westhawk's SNMP Stack in Java**. The Simple Times. Vol. 9, n 1, 11-14 p.

PARNES, P; SYNNES, K; SCHEFSTRON, D. **Real-Time Control and Management of Distributed Applications using IP- Multicast**. IFIP/IEEE International Symposium on Integrated Network Management, USA, Boston, Vol. 06, p. 901 – 914, maio de 1999.

RHODEN, Guilherme. **Detecção de Intrusões em Backbones de Redes de Computadores através da Análise de Comportamento com SNMP**. Dissertação (Mestrado em Ciência da Computação). Pós-Graduação em Ciência da Computação. Universidade Federal de Santa Catarina - UFSC. Florianópolis – SC. 2002.

RISO, Bernardo; SPECIALSKI, Elizabeth; FILHO, Jair; et al. **Arquitetura OSI de Gerenciamento. Sociedade Brasileira para Interconexão de Sistemas Abertos. Gerenciamento de Redes – Uma Abordagem de Sistemas Abertos**. São Paulo: Makron Books do Brasil, 1993, p. 13-34.

ROSE, Marshall; McCloghrie, Keith. **Structure and Identification of Management Information for TCP/IP-based Internets**. RFC 1065, agosto de 1988. 21 p.

ROSE, Marshall. **Management Information Base for Network Management of TCP/IP based internets**: MIB-II. RFC 1158, maio de 1990. 133 p.

ROSE, Marshall; McCloghrie, Keith. **Structure and Identification of Management Information for TCP/IP-based Internets**. RFC 1155, maio de 1990. 22 p.

SPECIALSKI, Elizabeth. **Gerência de Redes de Computadores e de Telecomunicações**. Apostila disponível no http://notes.ufsc.br/aplic/beth.nsf/ccae10dff955ca3f0425694b0078fbce/FILE/ApostPos2002.zip. Acessado em: julho de 2002.

SPECIALSKI, Elizabeth; MEDEIROS, Manoel; SOUSA JÚNIOR, Rafael; et al. **Funções de Gerenciamento de Relatório de Evento. Sociedade Brasileira para**

Interconexão de Sistemas Abertos. Gerenciamento de Redes – Uma Abordagem de Sistemas Abertos. São Paulo: Makron Books do Brasil, 1993, p. 69- 78.

STALLINGS, Willian. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2**. EUA: Addison Wesley, 1999. 3° edição. 616 p.

SUN. **SunNet Manager**. Disponível em: http://www.sun.com/products-n-solutions/nep/software/sm/datasheet.html. Acessado em: 25 de julho de 2002.

TRAP RECEIVER. **Trap Receiver Software**. Disponível em http://www.ncomtech.com/download.htm. Acessado em: 05 de setembro de 2002.

THOTTAN, Marina; JI, Chuanyi. **Proactive Anomaly Detection Using Distributed Intelligent Agents**. IEEE Network, Vol. 12, p. 21 - 27, setembro 1998.

VELLOSO, Pedro; RUBINSTEIN, Marcelo; DUARTE Otto. **Uma Ferramenta de Gerenciamento de Redes Baseada em Agentes Móveis**. Congresso Brasileiro de Automática, Natal – RN, setembro de 2002, 6 p.

VENÂNCIO NETO, Augusto. **Implementação de um Discriminador de Repasse de Eventos para o Ambiente SNMP**. Dissertação (Mestrado em Ciência da Computação). Pós-Graduação em Ciência da Computação. Universidade Federal de Santa Catarina - UFSC. Florianópolis – SC. 2001.

VENÂNCIO NETO, Augusto; SPECIALSKI, Elizabeth; CERQUEIRA, Eduardo et al. **SNMP Model Decentralization Through Utilization of Distributed Multi-Managers with Emphasis in Evert Report**. The 2002 International Conference on Security and Management. Las Vegas, US, junho de 2002.

VERONEZ, Cleverson. **Gerência de Desempenho do Tráfego em Redes Utilizando Baseline Bayesiana**. Dissertação (Mestrado em Ciência da Computação). Pós-

Graduação em Ciência da Computação. Universidade Federal de Santa Catarina - UFSC. Florianópolis – SC. 2000.

ZAPF, M; HERRMANN, K.; GEIHS, K. **Decentralized SNMP Management with Mobile Agents**. IFIP/IEEE International Symposium on Integrated Network Management, Boston, USA, Vol. 06, p. 623 - 635, maio de 1999.

Apêndice 1

Este apêndice apresenta algumas versões das RFCs relacionadas com o SNMP de acordo com (PANTON & ARKESTEIJN, 2001).

SMIv1 Data Definition Language
Full Standards:

- RFC 1155 - Structure of Management Information
- RFC 1212 - Concise MIB Definitions

Informational:

- RFC 1215 - A Convention for Defining Traps

SMIv2 Data Definition Language

Full Standards:

- RFC 2578 - Structure of Management Information
- RFC 2579 - Textual Conventions
- RFC 2580 - Conformance Statements

SNMPv1 Protocol

Full Standards:

- RFC 1157 - Simple Network Management Protocol

Proposed Standards:

- RFC 1418 - SNMP over OSI
- RFC 1419 - SNMP over AppleTalk
- RFC 1420 - SNMP over IPX

SNMPv2 Protocol

Draft Standards:

- RFC 1905 - Protocol Operations for SNMPv2
- RFC 1906 - Transport Mappings for SNMPv2
- RFC 1907 - MIB for SNMPv2

Experimental:

- RFC 1901 - Community-based SNMPv2
- RFC 1909 - Administrative Infrastructure
- RFC 1910 - User-based Security Model

SNMPv3 Protocol

Draft Standards:

- RFC 2571 - Architecture for SNMP Frameworks
- RFC 2572 - Message Processing and Dispatching
- RFC 2573 - SNMP Applications
- RFC 2574 - User-based Security Model
- RFC 2575 - View-based Access Control Model
- RFC 1905 - Protocol Operations for SNMPv2
- RFC 1906 - Transport Mappings for SNMPv2
- RFC 1907 - MIB for SNMPv2

Proposed Standards:

- RFC 2576 - Coexistence between SNMP Versions

Informational:

- RFC 2570 - Introduction to SNMPv3

Experimental:

- RFC 2786 - Diffie-Helman USM Key Management

SNMP Agent Extensibility

Draft Standards:

- RFC 2741 - AgentX Protocol Version 1
- RFC 2742 - AgentX MIB

SMIv1 MIB Modules

Full Standards:

- RFC 1213 - Management Information Base II
- RFC 1643 - Ethernet-Like Interface Types MIB

Draft Standards:

- RFC 1493 - Bridge MIB
- RFC 1559 - DECnet phase IV MIB

Proposed Standards:

- RFC 1285 - FDDI Interface Type (SMT 6.2) MIB
- RFC 1381 - X.25 LAPB MIB
- RFC 1382 - X.25 Packet Layer MIB
- RFC 1414 - Identification MIB

- RFC 1461 - X.25 Multiprotocol Interconnect MIB
- RFC 1471 - PPP Link Control Protocol MIB
- RFC 1472 - PPP Security Protocols MIB
- RFC 1473 - PPP IP NCP MIB
- RFC 1474 - PPP Bridge NCP MIB
- RFC 1512 - FDDI Interface Type (SMT 7.3) MIB
- RFC 1513 - RMON Token Ring Extensions MIB
- RFC 1525 - Source Routing Bridge MIB
- RFC 1742 - AppleTalk MIB

SMIv2 MIB Modules

Full Standards:

- RFC 2819 - Remote Network Monitoring MIB

Draft Standards:

- RFC 1657 - BGP version 4 MIB
- RFC 1658 - Character Device MIB
- RFC 1659 - RS-232 Interface Type MIB
- RFC 1660 - Parallel Printer Interface Type MIB
- RFC 1694 - SMDS Interface Type MIB
- RFC 1724 - RIP version 2 MIB
- RFC 1748 - IEEE 802.5 Interface Type MIB
- RFC 1850 - OSPF version 2 MIB
- RFC 1907 - SNMPv2 MIB
- RFC 2115 - Frame Relay DTE Interface Type MIB
- RFC 2571 - SNMP Framework MIB
- RFC 2572 - SNMPv3 MPD MIB
- RFC 2573 - SNMP Applications MIBs
- RFC 2574 - SNMPv3 USM MIB
- RFC 2575 - SNMP VACM MIB
- RFC 2790 - Host Resources MIB
- RFC 2863 - Interfaces Group MIB

Proposed Standards:

- RFC 1666 - SNA NAU MIB

- RFC 1696 - Modem MIB
- RFC 1697 - RDBMS MIB
- RFC 1747 - SNA Data Link Control MIB
- RFC 1749 - 802.5 Station Source Routing MIB
- RFC 1759 - Printer MIB
- RFC 2006 - Internet Protocol Mobility MIB
- RFC 2011 - Internet Protocol MIB
- RFC 2012 - Transmission Control Protocol MIB
- RFC 2013 - User Datagram Protocol MIB
- RFC 2020 - IEEE 802.12 Interfaces MIB
- RFC 2021 - RMON Version 2 MIB
- RFC 2024 - Data Link Switching MIB
- RFC 2051 - APPC MIB
- RFC 2096 - IP Forwarding Table MIB
- RFC 2108 - IEEE 802.3 Repeater MIB
- RFC 2127 - ISDN MIB
- RFC 2128 - Dial Control MIB
- RFC 2206 - Resource Reservation Protocol MIB
- RFC 2213 - Integrated Services MIB
- RFC 2214 - Guaranteed Service MIB
- RFC 2232 - Dependent LU Requester MIB
- RFC 2238 - High Performance Routing MIB
- RFC 2266 - IEEE 802.12 Repeater MIB
- RFC 2287 - System-Level Application Mgmt MIB
- RFC 2320 - Classical IP and ARP over ATM MIB
- RFC 2417 - Multicast over UNI 3.0/3.1 / ATM MIB
- RFC 2452 - IPv6 UDP MIB
- RFC 2454 - IPv6 TCP MIB
- RFC 2455 - APPN MIB
- RFC 2456 - APPN Trap MIB
- RFC 2457 - APPN Extended Border Node MIB
- RFC 2465 - IPv6 Textual Conventions and MIB

- RFC 2466 - ICMPv6 MIB
- RFC 2493 - 15 Minute Performance History TCs
- RFC 2494 - DS0, DS0 Bundle Interface Type MIB
- RFC 2495 - DS1, E1, DS2, E2 Interface Type MIB
- RFC 2496 - DS3/E3 Interface Type MIB
- RFC 2512 - Accounting MIB for ATM Networks
- RFC 2513 - Accounting Control MIB
- RFC 2514 - ATM Textual Conventions and OIDs
- RFC 2515 - ATM MIB
- RFC 2558 - SONET/SDH Interface Type MIB
- RFC 2561 - TN3270E MIB
- RFC 2562 - TN3270E Response Time MIB
- RFC 2564 - Application Management MIB
- RFC 2576 - SNMP Community MIB
- RFC 2584 - APPN/HPR in IP Networks
- RFC 2591 - Scheduling MIB
- RFC 2594 - WWW Services MIB
- RFC 2605 - Directory Server MIB
- RFC 2613 - RMON for Switched Networks MIB
- RFC 2618 - RADIUS Authentication Client MIB
- RFC 2619 - RADIUS Authentication Server MIB
- RFC 2667 - IP Tunnel MIB
- RFC 2662 - ADSL Line MIB
- RFC 2665 - Ethernet-Like Interface Types MIB
- RFC 2668 - IEEE 802.3 MAU MIB
- RFC 2669 - DOCSIS Cable Device MIB
- RFC 2670 - DOCSIS RF Interface MIB
- RFC 2677 - Next Hop Resolution Protocol MIB
- RFC 2720 - Traffic Flow Measurement Meter MIB
- RFC 2737 - Entity MIB
- RFC 2742 - AgentX MIB
- RFC 2787 - Virtual Router Redundancy Proto. MIB

- RFC 2788 - Network Services Monitoring MIB
- RFC 2789 - Mail Monitoring MIB
- RFC 2873 - Fibre Channel Fabric Element MIB
- RFC 2851 - Internet Network Address TCs
- RFC 2856 - High Capacity Data Type TCs
- RFC 2864 - Interfaces Group Inverted Stack MIB
- RFC 2895 - RMON Protocol Identifier Reference
- RFC 2925 - Ping, Traceroute, Lookup MIBs
- RFC 2932 - IPv4 Multicast Routing MIB
- RFC 2933 - IGMP MIB
- RFC 2940 - COPS Client MIB
- RFC 2954 - Frame Relay Service MIB
- RFC 2955 - Frame Relay / ATM PVC MIB
- RFC 2959 - Real-Time Transport Protocol MIB
- RFC 2981 - Event MIB
- RFC 2982 - Expression MIB
- RFC 3014 - Notification Log MIB
- RFC 3019 - Multicast Listener Discovery MIB
- RFC 3020 - Frame Relay UNI/NNI Multilink MIB
- RFC 3055 - PSTN/Internet Interworking MIB
- RFC 3083 - DOCSIS Baseline Privacy Interface MIB
- RFC 3144 - RMON Interface Monitoring MIB
- RFC 3165 - Scripting MIB

Informational:

- RFC 1628 - Uninterruptible Power Supply MIB
- RFC 2620 - RADIUS Accounting Client MIB
- RFC 2621 - RADIUS Accounting Server MIB
- RFC 2666 - Ethernet Chip Set Identifiers
- RFC 2707 - Print Job Monitoring MIB
- RFC 2896 - RMON Protocol Identifier Macros
- RFC 2922 - Physical Topology MIB

Experimental:

- RFC 2758 - SLA Performance Monitoring MIB
- RFC 2786 - Diffie-Helman USM Key MIB
- RFC 2934 - IPv4 PIM MIB

Related Documents

Informational:

- RFC 1270 - SNMP Communication Services
- RFC 1321 - MD5 Message-Digest Algorithm
- RFC 1470 - Network Management Tool Catalog
- RFC 2039 - Applicability of Standard MIBs toWWW Server Management
- RFC 2962 - SNMP Application Level Gateway for

Payload Address Translation

- RFC 2975 - Introduction to Accounting Management
- RFC 3052 - Service Management Architectures Issues and Review
- RFC 3216 - SMIng Objectives

Experimental:

- RFC 1187 - Bulk Table Retrieval with the SNMP
- RFC 1224 - Techniques for Managing Asynchronously Generated Alerts
- RFC 1238 - CLNS MIB
- RFC 1592 - SNMP Distributed Program Interface
- RFC 1792 - TCP/IPX Connection MIB Specification
- RFC 3139 - Requirements for Configuration Management of IP-based Networks
- RFC 3179 - Script MIB Extensibility Protocol 1.1
- RFC 3198 - Terminology for Policy-Based Management

Apêndice 2

Este apêndice descreve como as PDUs são descritas em ANS.1.

```
RFC1157-SNMP DEFINITIONS ::= BEGIN

    IMPORTS

        ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks

            FROM RFC1155-SMI;

        Message ::=SEQUENCE {version INTEGER {version-1(0)},community
OCTET STRING, data ANY}

    -- protocol data units

        PDUs ::= CHOICE {get-request GetRequest-PDU,

                get-next-request GetNextRequest-PDU,

                get-response GetResponse-PDU,

                set-request SetRequest-PDU,

                trap Trap-PDU}

--- PDUs

GetRequest-PDU       ::= [0] IMPLICIT PDU

GetNextRequest-PDU ::= [1] IMPLICIT PDU

GetResponse-PDU     ::= [2] IMPLICIT PDU

SetRequest-PDU :      := [3] IMPLICIT PDU

PDU :: = SEQUENCE (request-id INTEGER,
```

```
                error-status INTEGER {
                                noError (0),
                                tooBig (1),
                                noSuchName (2),
                                badValue (3),
                                readOnly (4),
                                genErros (5),
                error-index INTEGER,
                variable-binding VarBindings}
Trap=PDU :: [4] IMPLICIT SEQUENCE {
                enterprise OBJECT IDENTIFIER
                agente-addr NetworkAddress,
                generic-trap INTEGER {
                        coldStart (0),
                        warmStart (1),
                        linkDown (2),
                        linkup (3),
                        authenticationFailure (4),
                        egpNeighborLoss (5),
```

entrepriseSpecific (6)}

specifc-trap INTEGER,

time-stamp TimeTicks,

variable-bindings (VarBindList}

varBind:: = SEQUENCE {name ObjectName, value ObjectSyntax}

varBindList ::= SEQUENCE OF VarBind

END

Apêndice 3

Software Discriminador de Repasse de Eventos (SDRE)
LRCSD (Laboratório de Redes de Computadores e Sistemas Distribuídos) – Fesurv

- Q T R S: Quantidade de *traps* recebido(s) pelo(s) discriminador(es) simultaneamente;
- Q D G: Quantidade de dispositivos gerenciados;
- Q SDRE: Quantidade de SDRE na rede;
- G D: Gerente(s) destinatário(s) dos *traps*; e
- Q T T: Quantidade total de *traps* discriminados.

Data 02/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	1	1	1	I	
8:00	1	1	1	I	
9:00	1	1	1	I	
10:00	1	1	1	I	
11:00	1	1	1	I	
12:00	1	1	1	I	
13:00	1	1	1	I	
14:00	1	1	1	I	
15:00	1	1	1	I	
16:00	1	1	1	I	
17:00	1	1	1	I	
18:00	1	1	1	I	
19:00	1	1	1	I	
20:00	1	1	1	I	
21:00	1	1	1	I	
22:00	1	1	1	I	
23:00	1	1	1	I	
24:00	1	1	1	I	18
Data 03/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	4	4	1	I	
8:00	4	4	1	I	
9:00	4	4	1	I	
10:00	4	4	1	I	
11:00	4	4	1	I	
12:00	4	4	1	I	
13:00	4	4	1	I	
14:00	4	4	1	I	
15:00	4	4	1	I	
16:00	4	4	1	I	
17:00	4	4	1	I	
18:00	4	4	1	I	

19:00	4	4	1	I	
20:00	4	4	1	I	
21:00	4	4	1	I	
22:00	4	4	1	I	
23:00	4	4	1	I	
24:00	4	4	1	I	72
Data 04/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	40	4	1	I	
8:00	40	4	1	I	
9:00	40	4	1	I	
10:00	40	4	1	I	
11:00	40	4	1	I	
12:00	40	4	1	I	
13:00	40	4	1	I	
14:00	40	4	1	I	
15:00	40	4	1	I	
16:00	40	4	1	I	
17:00	40	4	1	I	
18:00	40	4	1	I	
19:00	40	4	1	I	
20:00	40	4	1	I	
21:00	40	4	1	I	
22:00	40	4	1	I	
23:00	40	4	1	I	
24:00	40	4	1	I	720
Data 05/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	60	6	1	I	
8:00	60	6	1	I	
9:00	60	6	1	I	
10:00	60	6	1	I	
11:00	60	6	1	I	
12:00	60	6	1	I	
13:00	60	6	1	I	
14:00	60	6	1	I	
15:00	60	6	1	I	
16:00	60	6	1	I	
17:00	60	6	1	I	
18:00	60	6	1	I	
19:00	60	6	1	I	
20:00	60	6	1	I	
21:00	60	6	1	I	
22:00	60	6	1	I	
23:00	60	6	1	I	
24:00	60	6	1	I	1080

Data 06/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	80	8	1	I	
8:00	80	8	1	I	
9:00	80	8	1	I	
10:00	80	8	1	I	
11:00	80	8	1	I	
12:00	80	8	1	I	
13:00	80	8	1	I	
14:00	80	8	1	I	
15:00	80	8	1	I	
16:00	80	8	1	I	
17:00	80	8	1	I	
18:00	80	8	1	I	
19:00	80	8	1	I	
20:00	80	8	1	I	
21:00	80	8	1	I	
22:00	80	8	1	I	
23:00	80	8	1	I	
24:00	80	8	1	I	1440
Data 07/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
8:00	80	8	1	I	
9:00	80	8	1	I	
10:00	80	8	1	I	
11:00	80	8	1	I	
12:00	80	8	1	I	400
Data 09/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	80	8	1	I e II	
8:00	80	8	1	I e II	
9:00	80	8	1	I e II	
10:00	80	8	1	I e II	
11:00	80	8	1	I e II	
12:00	80	8	1	I e II	
13:00	80	8	1	I e II	
14:00	80	8	1	I e II	
15:00	80	8	1	I e II	
16:00	80	8	1	I e II	
17:00	80	8	1	I e II	
18:00	80	8	1	I e II	
19:00	80	8	1	I e II	
20:00	80	8	1	I e II	
21:00	80	8	1	I e II	
22:00	80	8	1	I e II	
23:00	80	8	1	I e II	

24:00	80	8	1	I e II	1440
Data 10/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	80	8	1	I e II	
8:00	80	8	1	I e II	
9:00	80	8	1	I e II	
10:00	80	8	1	I e II	
11:00	80	8	1	I e II	
12:00	80	8	1	I e II	
13:00	80	8	1	I e II	
14:00	80	8	1	I e II	
15:00	80	8	1	I e II	
16:00	80	8	1	I e II	
17:00	80	8	1	I e II	
18:00	80	8	1	I e II	
19:00	80	8	1	I e II	
20:00	80	8	1	I e II	
21:00	80	8	1	I e II	
22:00	80	8	1	I e II	
23:00	80	8	1	I e II	
24:00	80	8	1	I e II	1440
Data 11/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	80	8	1	I e II	
8:00	80	8	1	I e II	
9:00	80	8	1	I e II	
10:00	80	8	1	I e II	
11:00	80	8	1	I e II	
12:00	80	8	1	I e II	
13:00	80	8	1	I e II	
14:00	80	8	1	I e II	
15:00	80	8	1	I e II	
16:00	80	8	1	I e II	
17:00	80	8	1	I e II	
18:00	80	8	1	I e II	
19:00	80	8	1	I e II	
20:00	80	8	1	I e II	
21:00	80	8	1	I e II	
22:00	80	8	1	I e II	
23:00	80	8	1	I e II	
24:00	80	8	1	I e II	1440
Data 12/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	70	7	2	I, II e III	
8:00	70	7	2	I, II e III	
9:00	70	7	2	I, II e III	

10:00	70	7	2	I, II e III	
11:00	70	7	2	I, II e III	
12:00	70	7	2	I, II e III	
13:00	70	7	2	I, II e III	
14:00	70	7	2	I, II e III	
15:00	70	7	2	I, II e III	
16:00	70	7	2	I, II e III	
17:00	70	7	2	I, II e III	
18:00	70	7	2	I, II e III	
19:00	70	7	2	I, II e III	
20:00	70	7	2	I, II e III	
21:00	70	7	2	I, II e III	
22:00	70	7	2	I, II e III	
23:00	70	7	2	I, II e III	
24:00	70	7	2	I, II e III	2520
Data 13/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	70	7	2	I, II e III	
8:00	70	7	2	I, II e III	
9:00	70	7	2	I, II e III	
10:00	70	7	2	I, II e III	
11:00	70	7	2	I, II e III	
12:00	70	7	2	I, II e III	
13:00	70	7	2	I, II e III	
14:00	70	7	2	I, II e III	
15:00	70	7	2	I, II e III	
16:00	70	7	2	I, II e III	
17:00	70	7	2	I, II e III	
18:00	70	7	2	I, II e III	
19:00	70	7	2	I, II e III	
20:00	70	7	2	I, II e III	
21:00	70	7	2	I, II e III	
22:00	70	7	2	I, II e III	
23:00	70	7	2	I, II e III	
24:00	70	7	2	I, II e III	2520
Data 14/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
8:00	70	7	2	I, II e III	
9:00	70	7	2	I, II e III	
10:00	70	7	2	I, II e III	
11:00	70	7	2	I, II e III	
12:00	70	7	2	I, II e III	700
Data 16/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	60	6	3	I, II, III e IV	
8:00	60	6	3	I, II, III e IV	

9:00	60	6	3	I, II, III e IV	
10:00	60	6	3	I, II, III e IV	
11:00	60	6	3	I, II, III e IV	
12:00	60	6	3	I, II, III e IV	
13:00	60	6	3	I, II, III e IV	
14:00	60	6	3	I, II, III e IV	
15:00	60	6	3	I, II, III e IV	
16:00	60	6	3	I, II, III e IV	
17:00	60	6	3	I, II, III e IV	
18:00	60	6	3	I, II, III e IV	
19:00	60	6	3	I, II, III e IV	
20:00	60	6	3	I, II, III e IV	
21:00	60	6	3	I, II, III e IV	
22:00	60	6	3	I, II, III e IV	
23:00	60	6	3	I, II, III e IV	
24:00	60	6	3	I, II, III e IV	3240
Data 17/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	50	5	4	I, II, III e IV	
8:00	50	5	4	I, II, III e IV	
9:00	50	5	4	I, II, III e IV	
10:00	50	5	4	I, II, III e IV	
11:00	50	5	4	I, II, III e IV	
12:00	50	5	4	I, II, III e IV	
13:00	50	5	4	I, II, III e IV	
14:00	50	5	4	I, II, III e IV	
15:00	50	5	4	I, II, III e IV	
16:00	50	5	4	I, II, III e IV	
17:00	50	5	4	I, II, III e IV	
18:00	50	5	4	I, II, III e IV	
19:00	50	5	4	I, II, III e IV	
20:00	50	5	4	I, II, III e IV	
21:00	50	5	4	I, II, III e IV	
22:00	50	5	4	I, II, III e IV	
23:00	50	5	4	I, II, III e IV	
24:00	50	5	4	I, II, III e IV	3600
Data 18/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	50	5	4	I, II, III e IV	
8:00	50	5	4	I, II, III e IV	
9:00	50	5	4	I, II, III e IV	
10:00	50	5	4	I, II, III e IV	
11:00	50	5	4	I, II, III e IV	
12:00	50	5	4	I, II, III e IV	
13:00	50	5	4	I, II, III e IV	
14:00	50	5	4	I, II, III e IV	

15:00	50	5	4	I, II, III e IV	
16:00	50	5	4	I, II, III e IV	
17:00	50	5	4	I, II, III e IV	
18:00	50	5	4	I, II, III e IV	
19:00	50	5	4	I, II, III e IV	
20:00	50	5	4	I, II, III e IV	
21:00	50	5	4	I, II, III e IV	
22:00	50	5	4	I, II, III e IV	
23:00	50	5	4	I, II, III e IV	
24:00	50	5	4	I, II, III e IV	3600
Data 19/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	50	5	4	I, II, III, IV e V	
8:00	50	5	4	I, II, III, IV e V	
9:00	50	5	4	I, II, III, IV e V	
10:00	50	5	4	I, II, III, IV e V	
11:00	50	5	4	I, II, III, IV e V	
12:00	50	5	4	I, II, III, IV e V	
13:00	50	5	4	I, II, III, IV e V	
14:00	50	5	4	I, II, III, IV e V	
15:00	50	5	4	I, II, III, IV e V	
16:00	50	5	4	I, II, III, IV e V	
17:00	50	5	4	I, II, III, IV e V	
18:00	50	5	4	I, II, III, IV e V	
19:00	50	5	4	I, II, III, IV e V	
20:00	50	5	4	I, II, III, IV e V	
21:00	50	5	4	I, II, III, IV e V	
22:00	50	5	4	I, II, III, IV e V	
23:00	50	5	4	I, II, III, IV e V	
24:00	50	5	4	I, II, III, IV e V	3600
Data 20/12/02					
Hora	**Q T R S**	**Q D G**	**Q SDRE**	**G D**	**Q T T**
7:00	50	5	4	I, II, III, IV e V	
8:00	50	5	4	I, II, III, IV e V	
9:00	50	5	4	I, II, III, IV e V	
10:00	50	5	4	I, II, III, IV e V	
11:00	50	5	4	I, II, III, IV e V	
12:00	50	5	4	I, II, III, IV e V	
13:00	50	5	4	I, II, III, IV e V	
14:00	50	5	4	I, II, III, IV e V	
15:00	50	5	4	I, II, III, IV e V	
16:00	50	5	4	I, II, III, IV e V	
17:00	50	5	4	I, II, III, IV e V	
18:00	50	5	4	I, II, III, IV e V	
19:00	50	5	4	I, II, III, IV e V	
20:00	50	5	4	I, II, III, IV e V	

21:00	50	5	4	I, II, III, IV e V	
22:00	50	5	4	I, II, III, IV e V	
23:00	50	5	4	I, II, III, IV e V	
24:00	50	5	4	I, II, III, IV e V	3600

Quantidade Total de Traps: 31430

Base de Dados de Traps:
02 – 07/12: 100 traps distintos;
09 – 14/12: 300 traps distintos;
16 – 20/12: 500 traps distintos;

Gerentes:
I – Gerente I;
II – Gerente II;
III – Gerente III;
IV – Gerente IV;
V – Gerente V;