UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EVERTON RAFAEL POLINA

Uso de Gerenciamento por Delegação para Controle e Administração de No-breaks

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dra. Liane M. Rockenbach Tarouco
Orientadora

Porto Alegre, maio de 2007.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Polina, Everton Rafael

Uso de Gerenciamento por Delegação para Controle e Administração de No-breaks / Everton Rafael Polina – Porto Alegre: Programa de Pós-Graduação em Computação da UFRGS, 2007.

72 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2007. Orientadora: Liane Margarida Rockenbach Tarouco.

1. Gerência de redes. 2. Gerenciamento distribuído. 3. No-break. 4. Gerenciamento de energia. 5. SNMP. 6. MIB Script. 7. Aplicação de gerenciamento. I. Tarouco, Liane Margarida Rockenbach. II. Título.

AGRADECIMENTOS

A Deus, minha mãe Lourdes, minhas irmãs Kátia e Evelise e minha namorada Viviane. Apoio, interesse, preocupação e até sermão delas valeram.

A meus colegas e amigos que deram apoio e demonstraram preocupação, em especial ao Rodrigo Sanger, Rodrigo Machado e Diego Contessa.

Aos professores Liane e Lisandro que de uma forma ou de outra incentivaram para a realização deste.

A empresa CP Eletrônica que forneceu material de consulta, equipamentos para a realização de testes e apoiou a pesquisa.

Ao Gerson Gabiatti, pela compreensão em diversos momentos, principalmente quando eu precisava me ausentar do trabalho (alias, eram sempre nas piores horas).

A meus colegas de trabalho, que colaboraram mesmo que indiretamente.

Ao Caçula. Sem o apoio moral e a ajuda técnica dele provavelmente nada disso seria realidade.

A todos que de alguma forma ajudaram para a realização deste trabalho ou que apenas contribuíram para minha formação e crescimento.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ANEEL	Agência Nacional de Energia Elétrica
ANSI	American National Standards Institute
ASN.1	Abstract Syntax Notation One
CA	Corrente Alternada
CC	Corrente Contínua
CCITT	Consultative Committee for International Telegraph and Telephone
CMIP	Common Management Information Protocol
CMOT	CMIP Over TCP
DEC	Duração Equivalente de Interrupção por Unidade Consumidora
DISMAN	Distributed Management
DMTF	Distributed Management Task Force
DOD	Department of Defence
FEC	Freqüência Equivalente de Interrupção por Unidade Consumidora
HTTP	HyperText Transfer Protocol
Hz	Hertz
IAB	Internet Architecture Board
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
ISO	International Organization for Standarization
ITU	International Telecommunication Union
JASMIN	Java Script MIB Implementation
MbD	Management by Delegation
MIB	Management Information Base
MLM	Mid Level Manager
OID	Object Identifier
PC	Personal Computer

PDU	Protocol Data Unit
PHP	PHP: Hypertext Preprocessor
RMON	Remote Monitoring
RFC	Request For Comments
SMI	Structure of Management Information
SMX	Script MIB Extensibility
SNMP	Simple Network Management Protocol
SNMPv1	SNMP versão 1
SNMPv2	SNMP versão 2
SNMPv3	SNMP versão 3
TCP	Transport Control Protocol
TLM	Top Level Manager
UDP	User Datagram Protocol
UPS	Uninterruptible Power Supply
URL	Uniform Resource Location
V	Volts
VA	Volt-Ampére
WBEM	Web-Based Enterprise Management
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Atualmente há grande investimento por parte das empresas em sistemas que não podem parar e, conseqüentemente, em sistemas de gerenciamento para perceber a ocorrência de falhas ou prevê-las e evitar que elas aconteçam. Para amenizar os efeitos de falhas no fornecimento de energia elétrica, que são ainda comuns no país e em boa parte do mundo, são utilizados os no-breaks. Esses equipamentos são projetados para garantir o fornecimento de energia por um tempo definido e limitado nos momentos em que a concessionária de energia elétrica interrompe esse fornecimento.

O objetivo principal do presente trabalho é investigar uma solução de gerenciamento de no-breaks que possui como grande diferencial a utilização do paradigma de gerenciamento distribuído por delegação, com o uso do protocolo SNMP, apresentando as vantagens desse paradigma.

Para a distribuição de tarefas, a solução utiliza a MIB Script e uma implementação deste padrão (JASMIN – Java Script MIB Implementation) de forma que toda a interatividade com a ferramenta e a automatização dos processos que compõem a solução foi implementada com o uso das ferramentas livres Apache e PHP.

O trabalho apresenta uma breve descrição sobre o funcionamento dos no-breaks e destaca a UPS-MIB, MIB padrão para gerenciamento desses equipamentos, além de objetos importantes disponibilizados em outras MIBs proprietárias utilizadas para o gerenciamento de no-breaks. Além disso, o presente estudo sugere o uso de ferramentas para que o sistema de gerência seja responsável pelas estações alimentadas pelos no-breaks, ou seja, quando identificado que está próximo o momento do no-break não conseguir mais alimentar a carga, deve-se desligar a carga com segurança para que não ocorra uma perda de dados maior nem haja danos em componentes.

Nota-se que o gerenciamento utilizado neste trabalho, realizado através da ferramenta desenvolvida, facilita a detecção de falhas no sistema de fornecimento de energia, de modo a interagir satisfatoriamente com no-breaks para verificar condições da rede elétrica e tomar decisões pró-ativas com o intuito de manter uma rede de computadores em funcionamento, sem que haja percalços por problemas de falta de energia.

Palavras-Chave: Gerência de redes, gerenciamento distribuído, no-break, gerenciamento de energia, SNMP, MIB Script, aplicação de gerenciamento.

Use of Management by Delegation for Control and Administration of UPS

ABSTRACT

Nowadays most companies are investing in non-stop systems, which leads to investments in management systems to know when a fault occurs or to foresee and avoid them. To reduce the effects of power supply faults (which are common in Brazil and in other parts of the world), people use UPS systems. These equipments are designed to guarantee the power supply for a pre-defined limited time, just in case the power company interrupts the supply.

The main goal of this work is to present the implementation of a management application for UPS systems, which has the differential of using the management-by-delegation paradigm, based on SNMP, showing the advantages of this paradigm.

The application uses Script MIB and an implementation of this standard (JASMIN - Java Script MIB Implementation) for task distribution, in such a way that all the tool interactivity and process automation was done using the free tools Apache and PHP.

The work also presents a brief description about the UPS functioning, and shows UPS-MIB, the standard MIB for UPS management. Additionally the work presents some important objects available in some custom MIBs that are used for UPS management too. This study suggests the use of a set of tools to make the management system responsible for the workstations plugged on the UPS, which means that when the UPS is about to be unable to maintain the load, it should be turned off safely to avoid data losses or hardware faults.

One can note that the management done in this work through the developed tool makes easy the fault detection in the power supply system. This leads to a good interaction with the UPS to verify the conditions of the power network in order to take decisions to maintain the computer network running safely.

Keywords: Network management, distributed management, UPS, power management, SNMP, Script MIB, management application.

1 INTRODUÇÃO

Cada vez com mais freqüência as empresas utilizam sistemas computacionais que não podem parar, seja pelo fator lucro/prejuízo seja pelo fator segurança. Isso significa que as empresas investem cada vez mais para que seus sistemas sejam tolerantes a falhas e que um pequeno detalhe não deixe os sistemas "fora do ar". Há também grande investimento em sistemas de gerenciamento. É cada vez mais importante possuir um sistema capaz de perceber erros instantaneamente ou até mesmo prever que alguns erros acontecerão, principalmente em organizações que possuem o que se chama de “cargas críticas”.

Dentre as causas de falhas nos sistemas computacionais pode-se incluir também as paradas por falta de energia elétrica, ainda comuns em todo o país e em boa parte do mundo. Quando se precisa de sistemas com fornecimento ininterrupto de energia, utiliza-se equipamentos que no Brasil são conhecidos como No-breaks, ou UPS (*Uninterruptible Power Supply*). Esses equipamentos são projetados para garantir o fornecimento de energia por um tempo definido e limitado nos momentos em que a concessionária de energia elétrica interrompe esse fornecimento. Outros equipamentos também utilizados para garantir o fornecimento de energia ao ocorrer falha na energia padrão oferecida pela concessionária são os geradores, que na maioria das vezes são utilizados de forma combinada com os no-breaks, de modo a aumentar bastante a autonomia do sistema de fornecimento de energia..

Obter informações sobre as fontes de alimentação e disponibilizá-las para gerenciamento passa a ser um item importante na rede pois pode-se conseguir, por exemplo, a prevenção de paradas em momentos críticos e o fornecimento de dados necessários para alterações na estrutura de fornecimento/consumo de energia.

Mantendo forte relação com a área de gerência de redes, todos os dispositivos ou recursos gerenciados e as soluções apresentadas foram avaliados e parametrizadas utilizando o protocolo SNMP (*Simple Network Management Protocol*), padrão para gerenciamento de dispositivos na rede (STALLINGS, 1998).

Este trabalho tem como objetivo investigar e apresentar formas e alternativas para o gerenciamento de equipamentos do tipo UPS, estando também preocupado com o gerenciamento da rede computacional alimentada por este sistema de fornecimento de energia. Serão avaliadas ferramentas para controle e registro de informações de dispositivos da rede incluindo a *Host Resources MIB* (GRILLO, 2000).

Ao final deste trabalho será apresentada uma solução para o gerenciamento de um sistema de energia. Esse sistema de energia pode conter um ou mais no-breaks em uma ou mais redes e permite também obter informações e interagir com as estações da rede cuja alimentação depende desse sistema de fornecimento de energia. O sistema de gerenciamento proposto utiliza o conceito de gerenciamento distribuído, com a delegação de tarefas para gerentes espalhados pela rede. O gerenciamento distribuído procura melhorar alguns pontos fracos do tradicional paradigma de gerenciamento centralizado, no qual uma única estação de gerenciamento é responsável pelo controle dos agentes da rede. O paradigma centralizado não é escalonável, definindo assim um limite na gerenciabilidade de uma rede. Outro ponto crítico do gerenciamento centralizado é a disponibilidade do processo de gerenciamento. Se a única estação responsável pelo gerenciamento falhar, a comunicação com os agentes fica comprometida e também todo o gerenciamento. O gerenciamento distribuído busca melhorar a escalabilidade e flexibilidade dos sistemas de gerenciamento, contornando os problemas do paradigma centralizado. O paradigma de gerenciamento distribuído por delegação (*Management by Delegation* – MbD) propõe a delegação de funções de gerenciamento, distribuindo decisões de gerenciamento ao longo da rede gerenciada. Isso pode acontecer através da transferência e controle remoto de *scripts* de gerenciamento (YEMINI; GOLDSZMIDT; YEMINI, 1991) e permite que as funções de gerenciamento fiquem mais próximas das entidades gerenciadas, diminuindo o tráfego de gerenciamento em um único ponto.

Com o objetivo de estruturar o gerenciamento distribuído, o grupo de trabalho DISMAN (IETF, 2005) definiu a MIB Script (LEVI; SCHÖNWÄLDER, 2001) para ser o mecanismo baseado em SNMP que oferece suporte a comunicação entre os gerentes superiores (TLMs - *Top-Level Managers*) e os gerentes intermediários (MLMs - *Mid-Level Managers*). Este é um módulo suportado pelos MLMs e que permite aos TLMs transferir, controlar e obter resultados dos scripts de gerenciamento executados em um MLM. Neste trabalho, são estudados e utilizados os conceitos da MIB Script inclusive com uso de ferramenta desenvolvida no projeto JASMIN (SCHÖNWÄLDER; QUITTEK; KAPPLER, 2000) que implementa seu uso.

O restante do texto está organizado da seguinte maneira: o capítulo 2 apresenta o gerenciamento de redes, conceitos, funcionamento, o protocolo SNMP, sua utilização, seus componentes, o gerenciamento distribuído com os componentes necessários para o funcionamento desse paradigma. O capítulo 3 mostra detalhes e o objetivo da MIB Script, além da ferramenta JASMIN que implementa essa MIB, incluindo os detalhes de instalação e uso da ferramenta. Por sua vez, o capítulo 4 descreve especificamente o no-break, foco de monitoração, e apresenta alguns itens importantes que já foram pesquisados e definidos para a monitoração e gerenciamento desse tipo de equipamento. No capítulo 5 é apresentada a proposta, desenvolvimento e implantação do projeto de uma solução de gerenciamento distribuído. Por fim, o capítulo 6 apresentada uma conclusão a respeito da pesquisa realizada e sobre o projeto desenvolvido, e também sugere trabalhos futuros relacionados a esta dissertação baseados nos resultados obtidos.

2 GERENCIAMENTO DE REDES

Entre os objetivos do gerenciamento de redes estão detectar, armazenar a ocorrência de falhas em arquivos de log, notificar usuários e, sempre que possível, corrigir automaticamente o problema de rede para que a mesma possa continuar operando normalmente. O gerenciamento de falhas envolve a detecção dos sintomas e o isolamento do problema. Procede-se a correção do problema e testa-se a solução em todos os subsistemas envolvidos.

O gerenciamento de redes de forma automatizada é muito importante, e hoje praticamente já não se justifica optar por elementos de rede que não possam ser gerenciados automaticamente. No passado, essa gerência "automática" – que coleta dados e informa ao gerente sobre o problema – era apenas um desejo. Isso ocorria por não existirem muitos equipamentos capazes de fornecer informações a serem gerenciadas e pelo alto custo dos equipamentos aptos a esse fornecimento. No entanto, o que se percebe atualmente é que os benefícios obtidos com a utilização desses equipamentos gerenciáveis são de grande importância além do custo já ter sido reduzido a patamares mais acessíveis.

Entre esses benefícios estão fatos como não parar a produção por um problema que iria ocorrer ou se tornar mais grave dentro de alguns instantes, resolvendo-o imediatamente, ou seja, antes que ele ocorra ou seja humanamente percebido, ou ainda obter informações suficientes sobre um problema ocorrido na rede ou com um equipamento gerenciado a fim de planejar e não permitir a repetição do erro no futuro.

Na década de 1980 ocorreu uma grande expansão no mercado de redes de computadores. Acompanhando a evolução e as novas tecnologias que surgiam, tornava-se cada vez mais evidente a importância das redes de computadores para os modelos de negócios das empresas, onde era visível o grande crescimento da infra-estrutura existente e o surgimento de novas redes. Mas esse crescimento, por muitas vezes desordenado, fez com que o mercado sentisse as grandes desvantagens de se utilizar redes altamente heterogêneas – como era a realidade no momento – e, na maioria das vezes, incompatíveis entre si. Esses pontos viriam a afetar a gerência e o planejamento de crescimento dessas redes. Fazia-se necessário chegar a um consenso no que diz respeito a tecnologias que fossem capazes de gerenciar esses ambientes heterogêneos, com a criação de ferramentas que automatizassem o processo de gerenciamento em ambientes dos mais diversos.

A necessidade de integrar ambientes heterogêneos fez com que o mercado selecionasse naturalmente determinadas tecnologias de interconexão e gerenciamento de

redes. A pilha de protocolos TCP/IP permite uma excelente integração de redes de computadores e domina o mercado de interconexão. Já para o gerenciamento de redes, o SNMP (*Simple Network Management Protocol*) (CASE et al, 1999), protocolo padrão da IETF (*Internet Engineering Task Force*) para o gerenciamento de redes baseadas em tecnologia TCP/IP, tornou-se o mais difundido protocolo de gerenciamento de redes, sobre o qual trabalha a maior parte das aplicações desse gênero.

O gerenciamento de uma rede pode ser classificado em:

- Gerenciamento intrusivo: quando ocorre a inserção de mensagens na rede, aumentando o tráfego, buscando comunicação com os equipamentos gerenciados. Nesse caso o gerenciamento é feito basicamente através de solicitações e respostas.
- Gerenciamento não intrusivo: quando o conteúdo da rede é analisado através da monitoração do enlace, sem colocar tráfego extra de gerenciamento na rede. Esse tipo de gerenciamento possui uma grande limitação que é a necessidade de um único domínio de colisão (TANENBAUM, 2003).

Para ilustrar e ajudar na construção do conceito e verificar a evolução do gerenciamento de redes pode-se pensar em um gerente de redes executando sua tarefa de gerenciamento da sua infra-estrutura, fazendo uso de ferramentas como analisadores de protocolos, por exemplo. Essa forma de trabalho, com a utilização dessa ferramenta para a tarefa de gerenciamento, seria hoje considerada ultrapassada. Se opondo a isso, pode-se imaginar um sistema distribuído no qual os nós gerentes seriam espalhados na rede realizando tarefas de gerenciamento e colaborando para o gerenciamento de toda a rede, com auto polling dos dispositivos gerenciados (os dispositivos gerenciados recebem constantemente requisições a respeito de seu funcionamento ou sobre suas funções). Estações de trabalho gerando e reportando relatórios e gráficos sobre o tráfego e mudanças da topologia da rede também são recursos utilizados no sistema de gerenciamento. Esse exemplo também mostra que o gerenciamento de redes é um serviço que emprega uma variedade de ferramentas, aplicações e dispositivos os quais possibilitam que administradores e gerentes de redes possam monitorar de forma eficiente as redes sob sua responsabilidade.

A estrutura básica nos diversos dispositivos gerenciados – como roteadores, impressoras, serviços – consiste em executar software, tradicionalmente chamados de agentes, que monitoram esses dispositivos, possibilitando o envio de alarmes para estações de gerência quando da ocorrência de determinados problemas. Essas estações de gerenciamento quando recebem tais alarmes podem dar início a certas ações, as quais podem incluir notificação à pessoa responsável pela rede, desligamento de dispositivos, ou mesmo a tentativa de solucionar automaticamente o problema. Essas estações gerentes podem verificar periodicamente o valor de determinadas variáveis relacionadas aos dispositivos monitorados, variáveis essas que são mantidas pelos agentes.

Conforme será visto mais detalhadamente no capítulo 2.1.1, os agentes são módulos de software que organizam as informações coletadas sobre o dispositivo ao qual está alocado, onde normalmente residem, armazenam essas informações em uma base de informações de gerenciamento e as disponibilizam para os gerentes da rede de forma reativa ou pró-ativa, através do protocolo de gerenciamento.

Normalmente é montado um banco de dados no computador que será gerente da rede, contendo informações necessárias para apoiar o diagnóstico e a busca de soluções

para problemas da rede. Isto envolve esforço na identificação, localização e solução das situações de falhas.

Com tudo isso, pode-se observar que muitas vezes não se busca obter em uma rede o máximo de desempenho, por exemplo, mas sim um desempenho adequado para a atividade que se necessita e que esta rede permita a utilização de recursos de gerência da rede – buscando fornecer um serviço mais confiável e seguro. A necessidade de manter a qualidade, a diversificação e a complexidade cada vez maior dos serviços de rede implicam em uma necessidade tão vital quanto o próprio serviço, que é o gerenciamento, da rede, dos dispositivos e dos serviços.

E quando se tem a interligação de diversas redes locais sendo gerenciadas, pode ser que uma rede local esteja funcionando perfeitamente mas sem conexão com as outras redes, e, conseqüentemente, sem conexão com a máquina gerente. Por isso, o ideal é que se tenha, dentro de cada rede local, alguma máquina gerente que permita que a rede local possa ser gerenciada, ou pelo menos tenha suas informações de gerenciamento coletadas, mesmo que estas informações não sejam enviadas instantaneamente a um gerente central. Aplicando-se conceitos de gerenciamento distribuído é possível uma implementação neste sentido, devendo ser utilizado um gerente em diversas máquinas ao longo da rede. O *Remote Monitoring* (RMON) (WALDBUSSER, 1995) permite uma implementação neste sentido, devendo ser implementado em diversas máquinas na rede. É possível ainda que uma estação gerente de sub-rede, envie dados à estação central de gerência apenas em uma situação de falha na rede. Isto contribuiria para redução do tráfego de informações de controle na rede (*overhead*). Maior detalhamento sobre gerenciamento distribuído de redes de computadores será apresentado no capítulo 2.3.

2.1 SNMP

Existem três principais tipos de arquitetura de gerência de redes. A arquitetura de gerenciamento OSI baseada no protocolo CMIP (*Common Management Information Protocol*) (LEINWAND; CONROY, 1996) que se desenvolveu principalmente na área das telecomunicações. O WBEM (*Web-Based Enterprise Management*) do DMTF (*Distributed Management Task Force*) (DMTF, 2006) baseado em arquitetura distribuída e que utilizando XML sobre HTTP para trafegar dados é utilizado para gerenciamento baseado na Web. E a gerência que utiliza o protocolo SNMP (*Simple Network Management Protocol*) sobre rede IP (STALLINGS, 1998).

O SNMP (*Simple Network Management Protocol*) é o protocolo de gerência de redes padrão do IETF (IETF, 2006) e se tornou padrão de fato para o gerenciamento de redes IP. Ele é um protocolo pertencente à camada de aplicação do nível OSI (TANENBAUM, 2003) e utiliza, na camada de transporte, os serviços do protocolo UDP (RFC768) para enviar suas mensagens através da rede IP.

Ao se utilizar o protocolo SNMP para o gerenciamento tem-se basicamente dois personagens na rede que trocam informações entre si, em geral com requisições do tipo cliente-servidor. O gerente SNMP (cliente) realiza basicamente duas operações durante a gerência: a leitura de valores (GET) para o monitoramento do ponto gerenciado e a escrita (SET) onde for possível efetuar a alteração de valores deste ponto. O agente SNMP (servidor) fica então responsável por responder às solicitações de leitura e informar a ocorrência de exceções.

Toda a inteligência do processo fica na estação de gerência, permitindo que o agente seja uma aplicação muito simples e com o mínimo de interferência no dispositivo em que está sendo executado. Para isso, ficam sob responsabilidade do agente apenas as funções de responder às solicitações do gerente e alterar as informações quando solicitada tal operação, além de notificar o gerente no caso de ocorrer alguma exceção. As decisões tomadas na ocorrência de problemas e as funções de relatórios ficam sob responsabilidade do gerente. Mais detalhes sobre o agente e o gerente SNMP são mostrados nos capítulos 2.1.1 e 2.1.2 respectivamente.

A especificação do protocolo SNMPv1, no ano de 1990, está definida na RFC1157 (CASE et al, 1999). Outras duas versões do SNMP foram padronizadas e publicadas pelo IETF. Em 1993, foi definido o SNMP versão 2 (RFC1441) além de SMI (RFC1442) e MIB (RFC1450) para SNMPv2. No ano de 1998 era iniciada a publicação da definição da versão 3 do SNMP, através da RFC2261 que apresenta a arquitetura do protocolo.

Os padrões que definem essa estrutura de gerência de redes são descritos nos seguintes documentos: RFC 1155 – *Structure of Management Information* (SMI), RFC 1156 – *Management Information Base* (MIB) e RFC 1157 – *Simple Network Management Protocol* (SNMP). Mais do que descrever um protocolo para troca de informações de gerenciamento, esses documentos contêm as descrições de como devem ser organizadas essas informações – numa base de dados local ao elemento gerenciado – e como o protocolo SNMP pode ser usado para coletá-las e alterá-las, sendo, para tanto, definidas as mensagens a serem utilizadas e seus campos.

Os agentes normalmente "residem" nos dispositivos gerenciados e mantêm uma base de dados de informações de gerenciamento, chamada MIB (*Management Information Base* – Base de Informações de Gerência). O agente tem a responsabilidade de manter na MIB uma imagem inequívoca do dispositivo gerenciado. O gerente, sempre que necessário, solicita aos agentes informações sobre os dispositivos gerenciados.

Finalmente, as traps são mensagens assíncronas encaminhadas pelos agentes aos gerentes para reportar a ocorrência de eventos mais graves no dispositivo gerenciado. Essas traps carregam mensagens de erros específicas de SNMP ou criadas pelo fabricante do dispositivo.

Para que ocorra a troca de mensagens no protocolo SNMPv1 cinco PDUs (*Protocol Data Unit*) são utilizadas. São elas: *GetRequest*, *GetNextRequest*, *GetResponse*, *SetRequest* e *Trap*. As PDUs correspondem à definição dos formatos empregados pelas entidades de gerência na troca de informações. O gerente utiliza-se de três dessas PDUs para realizar sua função. *GetRequest* e *GetNextRequest* serão utilizadas quando o gerente deseja realizar uma leitura de dados e *SetRequest* é utilizado pelo gerente para solicitar uma escrita, ou seja, a alteração do dado referente a algum objeto. As outras duas PDUs são utilizadas pelo agente. Ele responderá uma solicitação do gerente utilizando a PDU *GetResponse* e tomará a iniciativa de enviar comandos ao gerente através da PDU *Trap*, quando um exceção ocorrer, justificando essa notificação (alarme). A definição do SNMP prevê uma autenticação que deve ser feita ao trocar informações utilizando o protocolo. Essa autenticação é chamada de comunidade (string informada por quem está tomando a iniciativa no envio de informações) e ela pode ser diferente para leitura, escrita e envio de *traps*. Se a comunidade for informada incorretamente, o entendimento entre agente e gerente não acontecerá e a comunicação ignorada.

Com a definição do SNMPv2 surgiram duas novas PDUs (*InformRequest* e *GetBulkRequest*) e a possibilidade de se realizar um gerenciamento distribuído com a comunicação gerente-gerente. A PDU *InformRequest* é utilizada nessa comunicação. Já a PDU *GetBulkRequest* é utilizada pelo gerente quando deseja fazer a solicitação ao agente de uma quantidade maior de dados – e não apenas de um único objeto. A PDU *Trap* no SNMPv2 é denominada *SNMPv2-Trap* e as demais PDUs definidas no SNMPv1 foram mantidas no SNMPv2 (*GetRequest*, *GetNextRequest*, *Response* e *SetRequest*).

No SNMPv3, definido em 5 RFCs (da RFC2271 a RFC2275), o principal objetivo na definição era, aproveitando algumas idéias da segunda versão do protocolo, garantir a segurança durante o processo de gerência, já que a autenticação pela comunidade sempre foi considerada muito fraca para garantir alguma forma de segurança. Mas a versão 3 do SNMP acabou sendo considerada de difícil implementação e esse é um dos motivos que faz com que ela praticamente não seja utilizada atualmente.

2.1.1 Agente SNMP

O agente é um processo executando no nodo gerenciado, responsável pela manutenção das informações de gerência desse nodo. Cada nodo gerenciado pelo SNMP deve possuir um agente e uma base de informações de gerência.

Um agente pode realizar o controle de informações de um nodo remoto, ou seja, um nodo incapaz de se comunicar através de SNMP mas que se comunica de alguma forma e transfere suas informações ao agente SNMP que se encontra em um outro dispositivo. Esse agente é denominado agente *proxy*, responsável por traduzir as informações entre o dispositivo e a aplicação de gerência. Com esse tipo de aplicação, é possível ao sistema de gerência realizar o monitoramento e alteração de dispositivos que se comunicam utilizando outros tipos de protocolos.

Cada nodo gerenciado é visto como um conjunto de variáveis que representam informações referentes ao seu estado atual. Essas variáveis ficam disponíveis ao gerente através de consultas e podem ser alteradas por ele – se assim as variáveis foram definidas. Ao disponibilizar essas variáveis à leitura, o nodo permite seu monitoramento e, ao receber novos valores do gerente, o nodo está sendo controlado. Isso permite ao sistema de gerência identificar, na estrutura da rede, problemas relacionados a nodos e modificá-los de acordo com a necessidade.

O agente também é responsável por notificar o gerente no caso da ocorrência de alguma exceção no nodo gerenciado. Os nodos gerenciados podem apresentar falhas ou comportamentos inadequados e quando o agente identifica que ocorreu um evento significativo, ele imediatamente informa o ocorrido a todas as estações de gerência de sua lista de distribuição de alarmes. Esta operação é efetuada através de uma interrupção (*trap*) e esse *trap* informa o que ocorreu inesperadamente, deixando por conta da estação de gerenciamento a realização de consultas para obter mais detalhes.

Os agentes SNMP podem ser classificados em dois tipos distintos, que diferem entre si pela forma como são implementadas as funcionalidades do protocolo SNMP e pelo modo como são feitas as interações com os dispositivos gerenciados (MAURO, 2005).

O primeiro tipo de agente SNMP é o *agente extensíve*l. Este tipo de agente em geral oferece suporte à MIB-II, e utiliza o SNMP diretamente. Isto significa que possui a implementação de todas as funcionalidades do protocolo SNMP. Para que ele se

comunique com o dispositivo gerenciado, é necessária a implementação de agentes estendidos. Um exemplo de agente extensível é o agente SNMP do sistema operacional Microsoft Windows. Ele não possui suporte a nenhuma MIB, e para que ele responda às requisições de objetos de uma determinada MIB deve haver uma biblioteca adicional que implemente o suporte à MIB. Para a plataforma Linux/Unix pode-se citar o agente Net-SNMP (anteriormente chamado de UCD-SNMP) (NET-SNMP, 2006). Este agente é usado como base para a implementação de uma grande variedade de agentes estendidos e é um dos mais difundidos para o ambiente.

Por sua vez, o agente estendido possui somente funções básicas de comunicação com o dispositivo gerenciado para busca de informações. Este tipo de agente é baseado em um agente principal (extensível), o qual implementa as funções do protocolo SNMP. Dessa forma, o trabalho de resposta às requisições do protocolo SNMP é feito somente pelo agente extensível, ficando para o agente SNMP estendido o trabalho de comunicação com o dispositivo gerenciado e disponibilização das informações de monitoração ao agente extensível.

2.1.2 Gerente SNMP

Um gerente refere-se a uma aplicação de gerência sendo executada em uma estação de gerenciamento. É possível que exista uma ou mais aplicações de gerência em execução em uma mesma estação e todas elas utilizem o protocolo de gerência disponibilizado por essa estação. Essas aplicações são capazes de monitorar os agentes através de requisições de informações contidas na base de informações de gerenciamento e de alterar as características dos nodos gerenciados, informando novos valores ao agente.

Toda a inteligência do processo fica na estação de gerência permitindo que o agente seja uma aplicação muito simples e com o mínimo de interferência no dispositivo em que está sendo executado. Para isso, ficam sob responsabilidade do agente apenas as funções de responder às solicitações do gerente e alterar as informações quando solicitada tal operação, além de notificar o gerente no caso de ocorrer alguma exceção. As decisões tomadas na ocorrência de problemas e as funções de relatórios ficam sob responsabilidade do gerente.

Os gerentes são os responsáveis pela implementação da política que será adotada na gerência. O gerente é acessível à pessoa ou entidade responsável pela administração da gerência de redes. Existem bons aplicativos de gerência disponíveis hoje em dia, sendo que muitos deles utilizam a visualização gráfica dos componentes de rede, de maneira a facilitar a localização dos dispositivos. Alarmes com e-mail, chamadas telefônicas, pagers e outras formas de comunicação com o administrador são comuns nestes aplicativos.

Existem componentes capazes de realizar tanto a tarefa de gerente quanto a de agente. Estes componentes são responsáveis pela construção de uma hierarquia de gerentes no sistema de redes, onde um gerente responsável por um subconjunto da rede pode ser monitorado por outro gerente.

2.2 MIB

MIB (*Management Information Base*) é a base de informações de gerenciamento. É o conjunto de informações contida na MIB que o agente é capaz de responder ao

gerente. Em geral é também codificado um arquivo – chamado arquivo de MIB – no qual são relacionadas essas informações para que o gerente saiba quais são as informações que podem ser solicitadas a um agente e também as informações de alerta (*traps*) que poderão ser enviadas do agente para o gerente.

Para a especificação do arquivo de MIB, surge a necessidade de uma forma padronizada para a codificação dos objetos, para tornar possível a comunicação entre equipamentos produzidos por diferentes fornecedores. Por isso a utilização de uma linguagem de definição de objetos padronizada, juntamente com regras de codificação. A linguagem utilizada para a definição desses objetos é a ASN.1 (*Abstract Syntax Notation One*).

Basicamente a ASN.1 é uma linguagem de declaração de dados primitiva. Ela permite a definição de objetos primitivos e a combinação desses com objetos mais complexos. O SNMP tem ainda algumas convenções léxicas que não são totalmente iguais às que a ASN.1 pura utiliza. A sintaxe de ASN.1 é bem extensa, e o protocolo SNMP utiliza parte dela, ou seja, para definir estruturas de dados SNMP utiliza-se apenas um conjunto simplificado da ASN.1 e a esse subconjunto é dado o nome de SMI (*Structure of Management Information*). As regras de construção das estruturas da MIB são descritas através da SMI cuja estrutura é definida por um conjunto de documentos (ROSE; MCCLOGHRIE, 1990) que sugerem a forma de identificação e agrupamento das informações, as sintaxes e os tipos de dados permitidos.

Constituída por uma estrutura de árvore (Figura 2.1) contendo as variáveis de gerência de um determinado equipamento, a MIB define, para cada variável, um identificador único denominado OID (*Object Identifier*), formado por um inteiro não negativo. O OID oferece uma forma de identificar objetos. Em princípio, todos os objetos definidos em todos os padrões oficiais podem ser exclusivamente identificados. Para localizar esta informação, o identificador da variável que será acessada pelo SNMP é representado com o IP do equipamento em conjunto com o identificador do objeto na árvore MIB.

O OID de um nodo é composto pelo OID do seu pai mais seu próprio identificador relativo. O uso de número nos OIDs dificulta a compreensão dos nodos da MIB e por isso o OID pode ser substituído por um nome: o OID Name. OID e OID Name podem ser expressos conjuntamente. Exemplos: o OID 1.3.6.1.2.1.1 pode ser representado pelo OID Name system. sysUpTime é o OID 1.3.6.1.2.1.1.3 ou system.3.

Para cada objeto são definidos: nome, identificador, sintaxe, descrição e acesso. As instâncias dos objetos são chamadas de variáveis. O nome do objeto (*Object Name*) é composto por uma string de texto curto. O identificador do objeto (OID) é formado por números separados por pontos. A sintaxe (*syntax*) descreve o formato ou valor e define o tipo do objeto. A descrição é uma string que informa o que a variável faz. O acesso é o tipo de controle que se pode ter sobre o objeto que pode ser: somente leitura, leitura e escrita ou não acessível.

O nó raiz da árvore MIB não possui rótulo, lista todas as organizações de padrões importantes no mundo na visão da ISO e possui três subníveis. O nó 0 (ccitt) é administrado pela CCITT (*Consultative Committe for International Telegraph and Telephone*) atual ITU (*International Telecommunication Union*), o nó 1 (iso) é administrado pela ISO (*International Organization for Standartization*) e o nó 2 (joint-iso-ccitt) é administrado em conjunto pela CCITT e pela ISO.

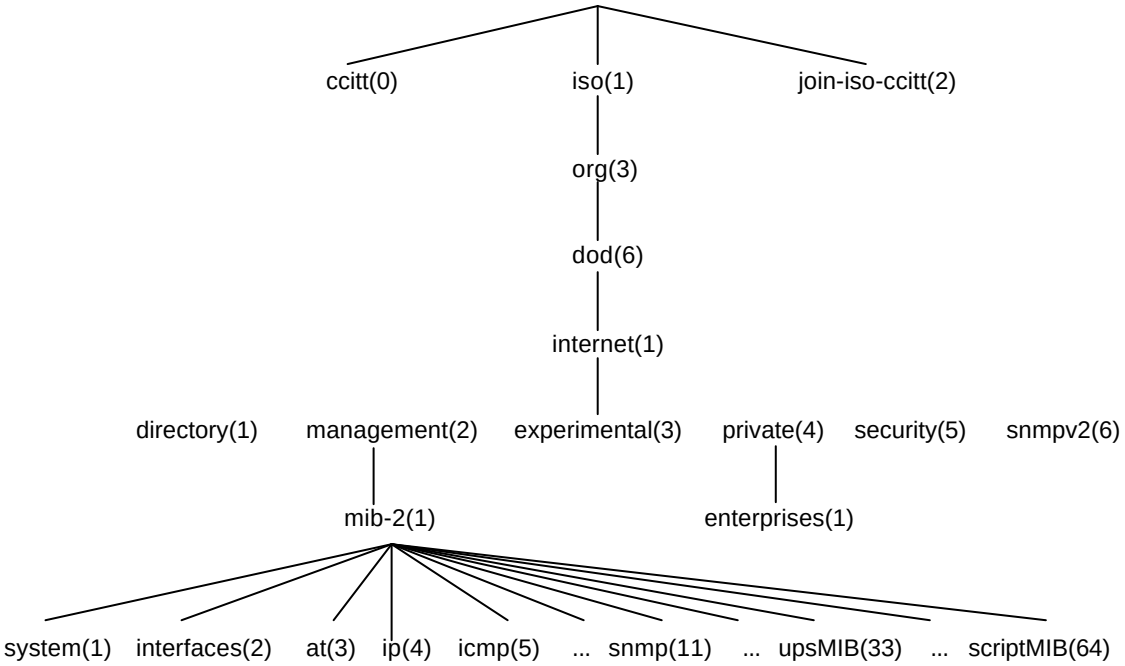


Figura 2.1: Árvore MIB parcial a partir da raiz (TANENBAUM, 2003)

Sob o nó iso ficam os nós *standard* (0), *registration-authority* (1), *member-body* (2) e *identified-organization* (3). Esse último nó org (3) pode ser utilizado por outras instituições. Abaixo do nó org fica o dod (6) que pertence ao departamento de defesa (*Department of Defence*) dos EUA. O DoD definiu um sub-nó para a comunidade internet, que é administrado pela IAB [IAB 2003] e abaixo deste nó temos os nós: *directory* (1), *management* (2), *experimental* (3), *private* (4), *security* (5) e snmpv2 (6).

Sob o nó *management* ficam as informações de gerenciamento e é sob este nó que está o nó da MIB II (mib-2).

Sob o nó *private* fica o nó *enterprises* e sob este nó ficam os nós particulares das empresas (INTERNET ASSIGNED NUMBERS AUTHORITY, 2006).

Muitas MIBs definidas se tornam padrão no gerenciamento a que se propõe. A RFC1066 (MCCLOGHRIE; ROSE, 1998) apresentou a primeira versão da MIB, a MIB I que definiu a base de informações necessárias para gerenciar redes baseadas na pilha de protocolos TCP/IP. A RFC1213 (MCCLOGHRIE; ROSE, 1991) propôs uma segunda MIB, a MIB II (mib-2, apresentada na Figura 2.1). Ela também foi feita para gerenciar redes baseadas na pilha de protocolos TCP/IP e implementa novos objetos em relação a MIB I. É o padrão utilizado atualmente e tornou obsoleta a MIB I. Além destas, há uma grande gama de MIBs para o gerenciamento de redes, dos computadores conectados a uma rede e para dispositivos e equipamentos em geral.

Além das MIBs padronizadas, cada fabricante pode definir uma MIB para descrição de seus equipamentos (MIB proprietária). Estas bases de informações de gerenciamento podem ser baseadas em MIBs padrão ou totalmente personalizadas de acordo com as características do equipamento ou dispositivo gerenciado. Com isso, o que se obtém é um gerenciamento bastante específico, o que pode acarretar uma melhoria na qualidade e quantidade das informações gerenciadas e nas notificações *(trap*s) enviadas ao gerente.

Um exemplo claro de existência de MIB padronizada ocorre no gerenciamento de No-breaks – UPS *(Uninterruptible Power Supply*). A RFC 1628 (CASE, 1994) define a UPS-MIB, uma MIB genérica para monitoração de equipamentos do tipo UPS. Com isso, apesar de muitos fabricantes de equipamentos definirem suas próprias MIBs proprietárias ajustadas a seus equipamentos, muitas delas são parecidas ou fortemente baseadas na UPS-MIB (CASE, 1994).

2.3 Gerenciamento distribuído

Para uso em redes de grande escala, o paradigma tradicional de gerenciamento centralizado alcançou seus limites (GOLDSZMIDT; YEMINI, 1995). Nesse paradigma, uma única estação de gerenciamento é responsável por manter toda a monitoração e controle de agentes que estão distribuídos ao longo da rede.

Com isso, fica estabelecido um limite na gerenciabilidade de uma rede, ou seja, existe um limite que deve ser cuidado e avaliado sobre quantas variáveis podem ser consultadas pela estação central e com qual freqüência isso pode ser realizado (BEN-ARTZI; CHADNA; WARRIER, 1990). Além disso, com o crescimento da rede, o processamento de dados na estação de gerenciamento pode se tornar inviável devido à elevada quantidade de dados que precisam ser coletados e processados.

Outro fator a ser levado em conta contra a utilização do gerenciamento centralizado está relacionado com a disponibilidade do processo de gerenciamento. Como apenas uma única estação é responsável pela monitoração dos agentes, se tal estação falhar ou ficar sobrecarregada, compromete o sistema de gerenciamento. A comunicação com os agentes estará comprometida e uma vez que o gerente que não está mais disponível os agentes ficarão inoperantes, aguardando requisições de consultas. Devido a isso, uma rede de dispositivos pode deixar de ser gerenciada.

O modelo de gerenciamento distribuído por delegação (MbD - *Management by Delegatio*n) é uma alternativa importante para os problemas encontrados no gerenciamento centralizado. Com a sua capacidade de distribuição de operações de gerenciamento para outras entidades participantes do sistema de gerenciamento que podem receber instruções e, executando estas, contribuir no gerenciamento dos dispositivos distribuídos na rede, o MbD fornece uma maior escalabilidade em relação ao número de dispositivos gerenciados, além de uma descentralização de operações que antes eram executadas na única estação de gerenciamento típica do gerenciamento centralizado.

O gerenciamento distribuído contorna problemas do paradigma centralizado e propõe melhorias na escalabilidade e flexibilidade dos sistemas de gerenciamento. A Figura 2.2 mostra esquemas práticos apontando as diferenças entre o gerenciamento centralizado e o gerenciamento distribuído. Essa figura está dividida mostrando do lado esquerdo (A) o esquema de gerenciamento centralizado e do lado direito (B) o gerenciamento distribuído. Nos dois esquemas o primeiro nível (identificado pelo número 1) representa o gerente principal da rede. Mas enquanto no esquema A ele é único na rede, monitorando diretamente os agentes representados por círculos pretos e identificados pelo número 2, no esquema B existem componentes intermediários entre o gerente principal (identificado pelo número 1) e os agentes monitorados (representados por círculos pretos e identificados pelo número 3). Esses componentes intermediários são os gerentes intermediários e não é a toa que na figura eles são representados por círculos divididos ao meio – metade branco e metade preto. Em toda a figura a cor preta

identifica os agentes e a cor branca identifica os gerentes. E é exatamente assim que se comporta o gerente intermediário. Para o gerente principal ele é um agente, recebe informações através do protocolo SNMP e responde ao gerente principal requisições feitas por ele com o objetivo de transferir funções ou obter resultados. E para os agentes finais do processo (identificados pelo número 3) os componentes intermediários (identificados pelo número 2) são gerentes. Fazem consultas SNMP com o objetivo de monitorar a rede da mesma foram como se fosse um gerente único operando na rede.

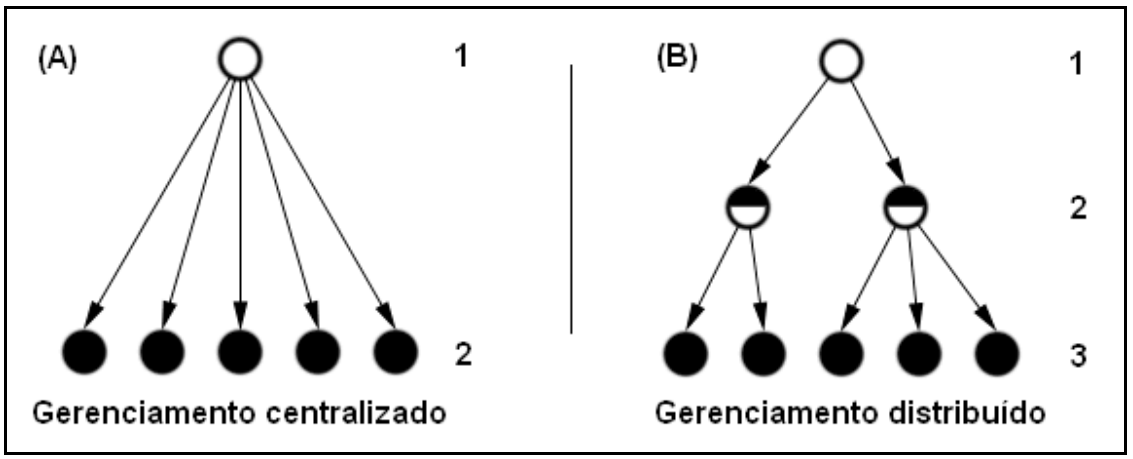


Figura 2.2: Esquema prático comparando gerenciamento centralizado e distribuído (QUITTEK; KAPPLER, 1999)

O paradigma de gerenciamento distribuído por delegação (Figura 2.3) propõe a delegação de funções de gerenciamento para localizações distribuídas, que pode ser realizada pela transferência e controle remoto de *scripts* de gerenciamento (YEMINI; GOLDSZMIDT; YEMINI, 1991). Isso permite que funções de gerenciamento fiquem mais próximas das entidades gerenciadas, diminuindo o tráfego de gerenciamento em um único ponto e distribuindo decisões de gerenciamento ao longo da rede gerenciada.

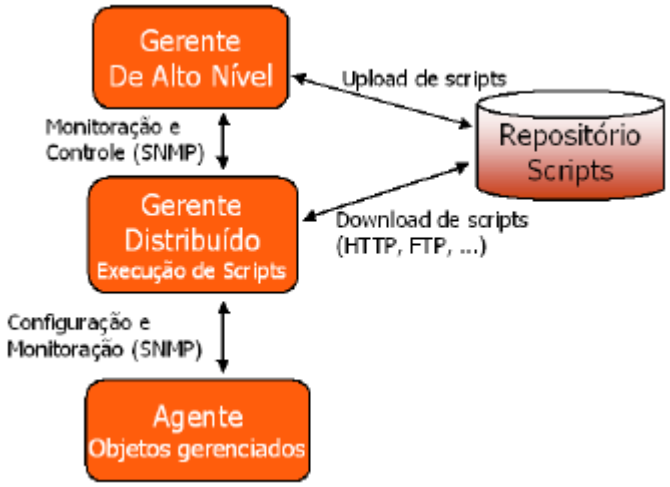


Figura 2.3: Arquitetura MbD do IETF

A abordagem do gerenciamento por delegação tornou-se concreta com a definição da MIB Script (LEVI; SCHÖNWÄLDER, 2001) pelo DISMAN *(Distributed Management Working Grou*p) (IETF, 2005) – um grupo de trabalho do IETF (IETF, 2006). A MIB Script fornece funcionalidades que permitem, através do uso de SNMP,

que *scripts* de gerenciamento sejam transferidos, controlados e o resultado das execuções seja obtido do gerente intermediário remoto.

Seguindo a classificação de gerenciamento distribuído apresentada por Schönwälder et. al. (SCHÖNWÄLDER; QUITTEK; KAPPLER, 2000), as entidades de um sistema MbD podem ser classificadas como gerentes superiores (TLMs - *Top-Level Manager*s), gerentes intermediários (MLMs - *Mid-Level Manager*s) e agentes. TLMs são responsáveis pelo monitoramento e controle de execução de *scripts* delegados para os MLMs, gerentes remotos e distribuídos ao longo da rede. Ou seja, MLMs são os receptores de *scripts* de gerenciamento e aqueles responsáveis pela execução de tais *scripts* como forma de realizar a tarefa de gerenciamento. Para fazer isso, os MLMs fazem a interação com agentes de gerenciamento localizados dentro de dispositivos de rede. Por fim, agentes são responsáveis pela consulta ou alteração dos dados existentes nos dispositivos de rede aos quais estão associados. A comunicação entre um MLM e um agente depende do protocolo de gerenciamento e respectiva versão implementado pelo agente e o correspondente suporte a tal protocolo e versão encontrado nos *scripts* sendo executados no MLM. Por outro lado, a comunicação entre um TLM e um MLM não depende dos *scripts* de gerenciamento. A comunicação necessária para transferir, controlar e obter resultados dos *script*s é totalmente independente do conteúdo dos *script*s.

Várias outras vantagens e características do modelo de gerenciamento por delegação são amplamente cobertos pela literatura sobre gerenciamento de redes (MARTIN-FLATIN, 2002) (KAHANI; BEADLE, 1997).

3 PADRÕES E IMPLEMENTAÇÃO DO GERENCIAMENTO DISTRIBUÍDO

Este capítulo apresenta o estudo feito sobre a MIB Script e sua implementação – o Jasmin – abordando também o uso na prática e a instalação da ferramenta.

A MIB Script foi definida para estruturar o gerenciamento distribuído, criando uma interface SNMP entre os gerentes superiores (TLMs) e os gerentes intermediários ou distribuídos (MLMs).

3.1 MIB Script

A MIB Script (LEVI; SCHÖNWÄLDER, 2001) foi desenvolvida pelo grupo de trabalho *Distributed Network Management* (DISMAN), do IETF, e publicada em maio de 1999 como um *proposed standard*. A MIB Script apresenta um conjunto de objetos que permitem a delegação de *scripts* de gerenciamento para gerentes distribuídos. Ela permite que processos remotos (*scripts*) sejam iniciados, controlados e terminados remotamente através de comandos de um gerente SNMP. Um *script* é um código capaz de ser executado pelo agente remoto que implementa a MIB Script.

A operação da MIB Script resume-se a operações realizadas nas seis tabelas que constituem os Objetos da MIB Script e essas tabelas são brevemente descritas a seguir:

1) **smLangTable**: esta tabela contém as linguagens que podem ser utilizadas para a criação dos scripts que serão executados pelo gerente, ou seja, as linguagens que a implementação da MIB Script é capaz de reconhecer/interpretar e executar. Cada entrada nessa tabela apresenta uma linguagem possível para executar *scripts* e detalhes como versão e revisão da linguagem. A Figura 3.1 apresenta um exemplo de uma consulta a essa tabela na qual os resultados mostraram que o gerente seria capaz de executar *scripts* em duas linguagens: Java e Tcl.

Instance	smLangLanguage	smLangVersion	smLangVendor	smLangRevision	smLangDescr
1	ianaLangJavaByteCode	1.1	enterprises.42	1.1	The Java virtual machine language.
2	ianaLangTcl	8.3	null	8.3.2	The Tool Command Language (Tcl).

Figura 3.1: Exemplo de um consulta à tabela *smLangTable*

Os valores de retorno para os objetos *smLangLanguage* e *smLangVendor* são OIDs que identificam a linguagem e seu fabricante, respectivamente. Nesse exemplo, os retornos para o objeto smLangLanguage foram 1.3.6.1.2.1.73.1 e 1.3.6.1.2.1.73.2. Essas

seqüências representam os objetos conforme representação na IANA-LANGUAGE-MIB (INTERNET ASSIGNED NUMBERS AUTHORITY, 2006a) e representam respectivamente as linguagens Java e Tcl, conforme apresentado na Tabela 3.1. Para o objeto smLangVendor foram obtidos os resultados "enterprises.42" – que identifica a empresa Sun Microsystems em "PRIVATE ENTERPRISE NUMBERS" (INTERNET ASSIGNED NUMBERS AUTHORITY, 2006b) – e "null" que significa fabricante não conhecido ou não identificado.

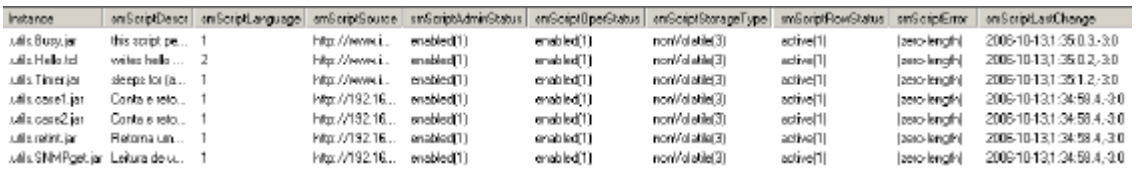
Tabela 3.1: Identificação das linguagens retornadas na consulta exemplo

OID	Identificação na ianaLanguages	Linguagem
1.3.6.1.2.1.73.1	iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).ianaLanguages(73).1	Java
1.3.6.1.2.1.73.2	iso(1).org(3).dod(6).internet(1).mgmt(2).mib-2(1).ianaLanguages(73).2	Tcl

2) **smExtsnTable**: essa tabela apresenta as extensões que a implementação da MIB Script aceita para as linguagens disponíveis (apresentadas na tabela *smLangTable*). É com base na combinação das informações dessas duas primeiras tabelas que o gerente saberá que recursos de linguagens (linguagem com algumas de suas extensões) poderá usufruir para criar seus *scripts.* Da mesma forma que na tabela anterior, cada linha da tabela representa uma extensão disponível que pode ser utilizada para a execução do script. Um pacote Java para uso de comandos SNMP, por exemplo, é uma possível extensão de linguagem (Java) e por conseqüência uma possível entrada dessa tabela.

Essas duas tabelas (*smLangTable* e *smExtsnTable*) só podem ser acessadas para leitura, já que é o agente quem possui controle sobre quais são as linguagens e extensões suportadas.

3) **smScriptTable**: essa tabela mostra todos os *scripts* conhecidos pelo gerente e armazenados localmente. Quando for desenvolvido um novo *script* para ser executado, ele deve ser transferido para o gerente MLM. Isso faz com que o novo *script* seja incluído nessa tabela, ou seja, haverá um linha na tabela que identifica o *script*, suas configurações e outras informações pertinentes para sua criação, manutenção e uso. Quando um *script* é incluído nesta tabela, ele deve obrigatoriamente estar associado a uma linguagem que a implementação é capaz de executar. Dessa forma, quando o *script* for executado o gerente saberá exatamente qual a linguagem deve ser iniciada para executá-lo. A identificação da linguagem do *script* na tabela *smScriptTable* é feita através do código que referencia a linguagem na tabela *smLangTable*, ou seja, para criar um *script* a ser executado na linguagem Java em ambiente que informa conhecer Java e Tcl, conforme exemplo da Figura 3.1, deve-se informar o código de linguagem 1 (um) – *smScriptLanguage* = 1 – pois é este o código (*Instance*) que identifica a linguagem Java na tabela *smLangTable*. A seguir, serão apresentados os passos para criar uma nova entrada na tabela *smScriptTable*, ou seja, para disponibilizar um novo *script* que poderá ser executado e então serão detalhados outros objetos importantes que fazem parte dessa tabela. A Figura 3.2 mostra um resultado parcial de uma consulta exemplo a tabela *smScriptTable*.



Instance	smScriptDescr	smScriptLanguage	smScriptSource	smScriptAdminStatus	smScriptOperStatus	smScriptStorageType	smScriptRowStatus	smScriptError	smScriptLastChange
.ulls.Busy.jar	this script pe...	1	http://www.i...	enabled(1)	enabled(1)	nonVolatile(3)	active(1)	[zero-length]	2006-10-13,1:35:0,3:0
.ulls.Hello.tcl	writes hello ...	2	http://www.i...	enabled(1)	enabled(1)	nonVolatile(3)	active(1)	[zero-length]	2006-10-13,1:35:0,2:0
.ulls.Timer.jar	sleeps for (a...	1	http://www.i...	enabled(1)	enabled(1)	nonVolatile(3)	active(1)	[zero-length]	2006-10-13,1:35:1,2:0
.ulls.case1.jar	Conta e reto...	1	http://192.16...	enabled(1)	enabled(1)	nonVolatile(3)	active(1)	[zero-length]	2006-10-13,1:34:58.4:0
.ulls.case2.jar	Conta e reto...	1	http://192.16...	enabled(1)	enabled(1)	nonVolatile(3)	active(1)	[zero-length]	2006-10-13,1:34:58.4:0
.ulls.retint.jar	Retorna um ...	1	http://192.16...	enabled(1)	enabled(1)	nonVolatile(3)	active(1)	[zero-length]	2006-10-13,1:34:58.4:0
.ulls.SNMPget.jar	Leitura de u...	1	http://192.16...	enabled(1)	enabled(1)	nonVolatile(3)	active(1)	[zero-length]	2006-10-13,1:34:58.4:0

Figura 3.2: Resultado parcial de uma consulta à tabela *smScriptTable*

Para disponibilizar um novo *script* para execução, este deve ser criado, disponibilizado em local acessível ao gerente MLM através de URL e executada uma seqüência de 7 (sete) operações SNMP set-request, conforme será destacado a seguir:

a) Gravar o valor 5 (*createAndWait*) no objeto *smScriptRowStatus* – inicializa uma nova linha na tabela.

b) Gravar o endereço do *script* (URL) no objeto *smScriptSource*.

c) Gravar no objeto *smScriptLanguage* o código da linguagem que deve ser utilizada para executar esse *script* – código da linguagem de acordo com as entradas na tabela *smLangTable*.

d) Gravar a descrição do *script* no objeto *smScriptDescr*.

e) Gravar o valor 3 (*nonVolatile*) no objeto *smScriptStorageType* – faz com que o *script* seja armazenado em memória não volátil.

f) Gravar o valor 1 (*active*) no objeto *smScriptRowStatus* – define que a linha já está definida e pode ser criada.

g) Gravar o valor 1 (*enabled*) no objeto *smScriptAdminStatus* – informa ao sistema que o *script* está liberado para uso.

Após essa seqüência de SETs, deve-se consultar o objeto *smScriptOperStatus*. O resultado dessa consulta deve ser o valor 1 (*enabled*), pois este é o indicativo do sistema que o *script* pode ser utilizado. Se ocorrer qualquer problema na definição do *script* (criação da linha na tabela) o sistema não ajustará o valor desse objeto para "enabled" e incluirá uma descrição da mensagem de erro no objeto *smScriptError*. O motivo mais comum para a não liberação do *script* para execução é a não localização do *script*, ou seja, a URL informada em *smScriptSource* estar incorreta ou inacessível.

Esses passos descrevem a forma como os *scripts* foram definidos e transferidos para o gerente no desenvolvimento deste trabalho, mas não é a única forma de realizar esta operação. Existem duas diferentes formas para transferir um *script* para o gerente. A primeira é chamada de "*pull model*". Requer que o gerente informe a URL de onde o gerente distribuído buscará o *script* e é esta maneira que foi descrita acima. A Segunda forma é chamada de "*push model*" e requer que o gerente envie o *script* para o gerente distribuído através do protocolo SNMP. Esse modelo funciona com o uso da tabela *smCodeTable* (descrita a seguir), permitindo que o gerente escreva o *script* através do envio de uma seqüência de operações set-request do SNMP.

Quando um *script* definido na tabela *smScriptTable* não for mais necessário, ele pode ser eliminado da tabela. Para isso é necessária a execução de uma seqüência de 3 passos, conforme descrito a seguir:

a) Gravar o valor 2 (*disabled*) no objeto *smScriptAdminStatus* – isso informa ao sistema que o *script* está desativado e assegura que nenhuma nova execução que se refere a este *script* será iniciada. O sistema fica aguardando que os *scripts* em execução sejam concluídos.

b) Deve-se consultar o objeto *smScriptOperStatus* e o resultado dessa consulta deve ser o valor 2 (*disabled*), pois este é o indicativo do sistema que o *script* pode ser eliminado, ou seja, não há nenhuma instância desse *script* em execução. Se houver alguma instância em execução o gerente pode forçar que o *script* seja terminado ou aguardar que conclua normalmente. De qualquer forma, o gerente fica monitorando o objeto *smScriptOperStatus* e quando seu resultado for *disabled* realiza a operação seguinte.

c) Gravar o valor 6 (*destroy*) no objeto *smScriptRowStatus* – isso removerá a linha da tabela *smScriptTable* e todos os recursos associados para a execução desse *script*.

4) **smCodeTable**: A definição da MIB Script define a implementação dessa tabela como opcional. Ela é utilizada para armazenar os *scripts* que foram transferidos do gerente através do modelo *push*, ou seja, uso de comandos SNMP para transferir o *script*. Um *script* pode ser dividido em diversas linhas dessa tabela para que seja possível a utilização de *scripts* maiores que o tamanho limite das mensagens SNMP. Além de criar novas linhas nessa tabela também é possível alterar as linhas já existentes, ou seja, utilizar comandos de maneira que eles sejam interpretados e executados na alteração de um *script* já transferido anteriormente.

5) **smLaunchTable**: essa tabela é utilizada para execução e controle dos *scripts*. Nela é descrito o ambiente de execução para cada *script*, incluindo os parâmetros que serão passados e o tempo máximo de vida do *script*. Cada entrada na tabela é chamada de "botão de execução" (*launch button*) que poderá ser usado para iniciar um *script*. Um *script* para ser executado precisa estar listado nesta tabela e para aqui disponibilizá-lo é necessário fazer uma associação com um *script* existente na tabela *smScriptTable*. Isso significa que quando o gerente cria um novo *script* ele deverá primeiramente disponibilizá-lo como um *script* válido na tabela de *scripts* (*smScriptTable*) para depois criar um botão (entrada na tabela *smLaunchTable*) através do qual ele será iniciado. Adicionalmente, para um *script* da tabela *smScriptTable* podem ser disponibilizados diversos botões para execução com ambientes diferentes, variando seus parâmetros, por exemplo. Para isso, ocorre a inclusão do *script* diversas vezes na tabela *smLaunchTable*. Se houver necessidade de executar duas ou mais vezes um mesmo *script* sem variar parâmetros ou outras opções de execução, ou seja, com idêntico ambiente de execução, pode ser incluído um único botão para este *script* e iniciá-lo diversas vezes, como será visto nas opções dos objetos desta tabela.

Para disponibilizar um novo botão para execução de *script*, este deve estar criado e habilitado para uso na tabela *smScriptTable*. Ou seja, conforme visto anteriormente, o objeto *smScriptTable.smScriptOperStatus* deve conter o valor 1 (*enabled*), indicando que o *script* pode ser utilizado. Atendido este requisito, deve ser executada uma seqüência de até 12 (doze) operações SNMP set-request – apresentadas e explicadas abaixo – para deixar o botão disponível.

a) Gravar o valor 5 (*createAndWait*) no objeto *smLaunchRowStatus* – inicializa uma nova linha na tabela.

b) Gravar no objeto *smLaunchScriptOwner* string de identificação do “dono do *script”*.

c) Gravar no objeto *smLaunchScriptName* string de identificação do nome do *script*.

d) Gravar no objeto *smLaunchArgument* os parâmetros que devem ser passados para o *script* no momento de sua execução. Havendo mais de um parâmetro eles devem ser separados por espaço em branco.

e) Gravar no objeto *smLaunchMaxRunning* um valor que indica a quantidade máxima de execuções simultâneas que serão aceitas para esse *script*, ou seja, como visto anteriormente, havendo necessidade de executar diversas vezes um mesmo *script*, basta incluir um único botão para iniciá-lo e utilizar este botão diversas vezes, bastando para isso que o valor de *smLaunchMaxRunning* esteja de acordo com a quantidade de execuções simultâneas desejada.

f) Gravar no objeto *smLaunchMaxCompleted* um valor que indica a quantidade máxima de execuções do *script* já concluídas que serão mantidas para consulta. Na descrição da próxima tabela (*smRunTable*) será visto que nela permanecem as informações sobre os *scripts* mesmo após concluídos para que seja possível consultar resultados, tempo de execução entre outras informações. Sendo assim, ficarão disponíveis para essa consulta, no máximo, o número informado neste objeto. Se for configurado para que fique no máximo n linhas na tabela com resultados de execuções distintas de um *script*, quando for concluída a execução de número n+1 do *script* a entrada que mantinha informações e resultado da execução mais antiga será descartada.

g) Gravar no objeto *smLaunchLifeTime* um valor que será o tempo máximo durante o qual o *script* poderá ser executado. Se o *script* não for concluído no prazo máximo estipulado neste objeto ele automaticamente é encerrado pelo sistema (retornando valor de código de saída 3 = *lifeTimeExceeded*).

h) Gravar no objeto *smLaunchExpireTime* um valor que será o tempo máximo durante o qual o resultado do *script* será mantido (na tabela *smRunTable*) após ter sido concluído.

i) Gravar no objeto *smLaunchRowExpireTime* um valor que será o tempo pelo qual o botão que está sendo inserido na tabela ficará disponível para uso. Passado o tempo definido a linha é eliminada da tabela (a menos que existam entradas na tabela *smRunTable* relacionados a esse *script*). Havendo entradas na tabela *smRunTable* relacionados a esse *script*, ao invés de ser imediatamente eliminada, a linha fica em um estado intermediário, aguardando apenas que não exista mais nenhuma dependência de entradas da tabela *smRunTable* com esse *script*. Para identificar esse estado intermediário o sistema ajusta o objeto *smLaunchOperStatus* para 3 (*expired*). Se for desejado que o botão esteja sempre disponível e não seja removido após algum tempo, deve-se ajustar esse objeto com o valor máximo (2147483647) e assim o contador de tempo estará desligado.

j) Gravar o valor 3 (*nonVolatile*) no objeto *smLaunchStorageType* – faz com que essa entrada da tabela seja mantida em memória não volátil.

k) Gravar o valor 1 (*active*) no objeto *smLaunchRowStatus* – define que a linha (botão de execução) já está definida e pode ser criada.

l) Gravar o valor 1 (*enabled*) no objeto *smLaunchAdminStatus* – informa ao sistema que o botão de execução está liberado para uso.

Após essa seqüência, deve-se consultar o objeto *smLaunchOperStatus*. O resultado dessa consulta deve ser o valor 1 (*enabled*), pois este é o indicativo do sistema de que o botão pode ser utilizado para iniciar a execução do *script*. Toda essa operação apenas deixou o *script* disponível para execução e é essa linha criada na tabela *smLaunchTable*, que permitirá isso. Por esse motivo que cada linha da tabela é chamada de "botão de execução". Com esse botão o mesmo *script* pode ser iniciado diversas vezes, inclusive podem executar em paralelo, ou seja, iniciar o mesmo *script* sem que a execução anterior tenha sido terminada (desde que a configuração utilizada no objeto *smLaunchMaxRunning* permita isso, como explicado anteriormente).

Para iniciar a execução de um *script* é utilizado o objeto *smLaunchStart*. Deve-se fazer um SNMP set-request neste objeto mas há também uma breve seqüência de comandos que devem ser executados nessa tabela *smLaunchTable* conforme descrita a seguir.

Ao escolher o *script* que será executado (escolher uma das linhas da tabela *smLaunchTable*), deve-se consultar o objeto *smLaunchRunIndexNext* dessa tabela e será retornado um valor inteiro que, como o próprio nome indica, é o índice para a próxima execução do botão. Esse índice (valor) deve ser usado para iniciar a execução do *script*, ou seja, deve-se gravar esse valor no objeto *smLaunchStart*. Por exemplo, supondo que uma linha da tabela *smLaunchTable* é o botão para iniciar o *script* "IdentNameResult" e que ao fazer a consulta ao abjeto *smLaunchRunIndexNext* desta linha da tabela o valor retornado nesta consulta foi "8". Será realizada a operação set-request no objeto *smLaunchStart* desta linha utilizando o valor 8. Cada vez que se consulta o objeto *smLaunchRunIndexNext* ele é incrementado o que garante que uma próxima execução desse mesmo *script* utilizará um valor de índice diferente do anterior.

Todo *script* que tenha iniciado sua execução possuirá uma entrada na tabela *smRunTable* que será descrita com mais detalhes a seguir. Essa entrada será identificada através do nome do *script* e de seu índice de execução. Utilizando novamente o exemplo acima, após iniciar a execução do *script* será criada uma entrada na tabela *smRunTable* que será identificada por "IdentNameResult.8". Execuções subseqüentes desse mesmo *script* iriam gerar entradas identificadas por "IdentNameResult.9", "IdentNameResult.10" e assim por diante. Isso garante um identificador único para cada execução do *script*.

Quando um botão de execução definido na tabela *smLaunchTable* não for mais necessário ele pode ser eliminado da tabela e para isso deve-se executar uma seqüência de 3 passos:

a) Gravar o valor 2 (*disabled*) no objeto *smLaunchAdminStatus* – isso informa ao sistema que o botão (entrada na tabela) está desativado, impossibilitando o uso, e assegura que nenhuma nova execução a partir desse botão será iniciada. O sistema fica aguardando que os *scripts* que foram iniciados por este botão, e que permanecem em execução, sejam concluídos.

b) Deve-se consultar o objeto *smLaunchOperStatus* e o resultado dessa consulta deve ser o valor 2 (*disabled*), pois este é o indicativo do sistema de que o botão de execução pode ser eliminado, ou seja, não há em execução nenhum *script* que tenha

sido iniciado por esse botão. Se houver alguma instância em execução o gerente pode forçar que o *script* seja terminado ou aguardar que conclua normalmente. De qualquer forma, o gerente fica monitorando o objeto *smLaunchOperStatus* e quando seu resultado for *disabled* realiza a operação seguinte.

c) Gravar o valor 6 (*destroy*) no objeto *smLaunchRowStatus* – isso removerá o botão de execução (a linha da tabela *smLaunchTable*) e todos os recursos associados para a execução desse *script*.

6) **smRunTable**: essa tabela possui uma entrada para cada *script* que está em execução. Além disso, os *scripts* que já concluíram sua execução também permanecem, por um determinado tempo, com uma entrada nesta tabela para que seja possível verificar que o *script* concluiu sua execução e consultar um possível resultado.

Quando é iniciada a execução de um *script*, automaticamente é colocada uma entrada referente a essa execução nesta tabela. Além disso, o objeto *smRunStartTime* é preenchido com data e hora em que a execução foi iniciada e ao terminar a execução do *script* isso também será registrado – no objeto *smRunEndTime.* Alguns objetos dessa tabela são preenchidos com dados provenientes da tabela *smLaunchTable*, são eles: *smRunArgument*, que contém os parâmetros passados para o *script* no momento de sua execução; *smRunLifeTime*, que possui o valor que representa o tempo máximo durante o qual o *script* pode permanecer em execução; e *smRunExpireTime*, que possui o valor que representa o tempo durante o qual a entrada será mantida nesta tabela após término da execução do *script*. Essas três informações são copiadas dos objetos *smLaunchArgument*, *smLaunchLifeTime* e *smLaunchExpireTime* respectivamente. Duas dessas informações, apesar de pré-definidas, podem ser alteradas: *smRunLifeTime* e *smRunExpireTime.* Ambas podem ser aumentadas ou diminuídas. Em *smRunLifeTime*, se o valor for ajustado para 0 (zero) a execução do *script* será imediatamente abortada. Mas se o valor for ajustado para seu valor máximo permitido (2147483647), então o *script* não terá um limite máximo de tempo para sua execução, permanecendo em execução tanto tempo quanto for necessário, até que o *script* seja finalizado normalmente. Em *smRunExpireTime*, se o valor for ajustado para 0 (zero) esta entrada será eliminada da tabela assim que o *script* tiver terminada a sua execução – assim que objeto *smRunState* tiver o valor 7 (*terminated*).

Um *script* que possui entrada nessa tabela pode ser consultado para conhecer seu estado e até mesmo alterar o estado do *script*. Através do objeto *smRunState* obtém-se o estado do *script* e são 7 os estados possíveis:

1) Inicializando (*initializing*): utilizado para os casos que o *script* foi acionado mas ainda não iniciou sua execução.

2) Executando (*executing*): ao iniciar a execução do *script* seu estado é alterado para *executing*.

3) Suspendendo (*suspending*): esse é um estado intermediário antes de uma execução ser suspensa. Quando solicitada a suspensão de uma execução, o *script* é colocado no estado *suspending*. Se a tentativa de suspender o *script* falhar, seu estado volta a ser *executing*.

4) Suspenso (*suspended*): se a tentativa de suspender a execução de um *script* teve sucesso ele pára e seu estado é alterado para *suspended*.

5) Recomeçando (*resuming*): um *script* suspenso que recebe uma requisição para recomeçar sua execução entra no estado intermediário *resuming*. Após recomeçar a execução o estado do *script* mudará para *executing*. Se o reinício de um *script* suspenso falhar ele volta ao estado *suspended*.

6) Cancelando (*aborting*): é o estado intermediário de um *script* que recebeu a solicitação de cancelamento. Um *script* cancelado não pode ser recomeçado. É isso que o diferencia de um *script* suspenso.

7) Terminado (*terminated*): estado de um *script* que concluiu sua execução.

Conforme visto, os *scripts* podem estar em estados distintos e esses podem ser manipulados. O gerente pode manipular o estado do *script* através do objeto *smRunControl* da tabela *smRunTable*. Um *script* nesta tabela pode ser cancelado, suspenso ou recomeçado. Para isso, basta o gerente gravar neste objeto o valor referente a operação que deseja fazer, conforme a Tabela 3.2:

Tabela 3.2: Operações para controlar a execução de *scripts*

Valor	Operação	Objetivo
1	*abort*	Cancela um *script* em execução
2	*suspend*	Suspende um *script* em execução
3	*resume*	Recomeça um *script* suspenso

Logicamente que essas operações devem ser executadas de acordo com o estado atual do *script* e a MIB Script está preparada para verificar inconsistência nessa solicitação, retornando erro em casos que a operação não é aceita. Solicitar que o *script* seja suspenso ou cancelado, por exemplo, se ele já concluiu sua execução, não faz sentido e são casos em que o erro "valor inconsistente" é retornado. Solicitar a suspensão de uma execução que está suspensa é outro exemplo. A Tabela 3.3 apresenta as operações que são possíveis para cada um dos 7 estados prováveis de uma execução e em que situações será retornado erro.

As operações *abort*, *suspend* e *resume* (cancelar, suspender e recomeçar) também podem ser executadas para manipular um conjunto de *scripts* – ao invés de um *script* único. Se ao invés de ajustar o valor do objeto *smRunControl* da tabela *smRunTable* o gerente ajustar o valor do objeto s*mLaunchControl* da tabela *smLaunchTable*, todos os *scripts* iniciados a partir do mesmo botão de execução serão afetados com o comando executado (cancelar, suspender ou recomeçar), ou seja, todos os objetos *smRunControl* de *scripts* que foram iniciados por essa linha da tabela *smLaunchTable* serão afetados, desde que o valor atual do *smRunState* permita a mudança correspondente do estado como descrito anteriormente e mostrado na Tabela 3.3. A tentativa de ajustar s*mLaunchControl* retorna o erro de "valor inconsistente" somente se todas as tentativas de mudança de estado de *smRunState* retornaram tal erro. Não será retornado erro se pelo menos uma mudança de estado em uma das instâncias aplicáveis do *script* for bem sucedida.

Tabela 3.3: Relação operação x estado do *script*

Estado da execução do *script* (objeto *smRunState*)	Operação solicitada (SNMP SET em *smRunControl*)		
	abort	*Suspend*	*resume*
initializing	Cancela a execução do *script*	Erro: valor inconsistente	Erro: valor inconsistente
executing	Cancela a execução do *script*	Suspende a execução do *script*	Erro: valor inconsistente
suspending	Cancela a execução do *script*	Erro: valor inconsistente	Recomeça a execução do *script*
suspended	Cancela a execução do *script*	Erro: valor inconsistente	Recomeça a execução do *script*
resuming	Cancela a execução do *script*	Erro: valor inconsistente	Erro: valor inconsistente
aborting	Erro: valor inconsistente	Erro: valor inconsistente	Erro: valor inconsistente
terminated	Erro: valor inconsistente	Erro: valor inconsistente	Erro: valor inconsistente

Além dos casos apresentados na Tabela 3.3, a implementação também poderá retornar erro se falhar a tentativa de executar a operação (apesar de estar em uma situação válida).

Outra maneira de se terminar um *script* é ajustando o *smRunLifeTime* para 0 (zero), o que faz com que o sistema encerre a execução do *script* porque o tempo de vida expirou – retornando *lifeTimeExceeded* no código de saída.

Após concluída a execução de um *script*, que pode ser verificado pelo objeto de estado – *smRunState* = 7 (*terminated*) – alguns outros objetos dessa tabela são importantes (além da data e hora de término de execução – *smRunEndTime* – que já foi citado anteriormente).

smRunExitCode: neste objeto será colocado um valor que indica o motivo pelo qual a execução de um *script* terminou. Espera-se que qualquer *script*, na maioria das vezes, tenha sua execução normal, terminando porque suas tarefas foram concluídas com êxito. Nesse caso o código de saída retornado será 1 (*noError*). Mas durante a execução dos *scripts* eles podem ser encerrados por outros motivos, incluindo alguns tipos de erros e isso é representado neste objeto. Outros valores possíveis para esse objeto são apresentados na Tabela 3.4, incluindo o motivo e breve descrição para o valor de retorno. Se o *script* ainda não iniciou a execução ou está executando, o valor deste objeto será 1 (*noError*).

Tabela 3.4: Valores de retorno ao finalizar a execução de *scripts*

Valor e descrição do retorno		Motivo pelo qual o *script* foi encerrado
2	*halted*	solicitação de um gerente
3	*lifeTimeExceeded*	excedeu o tempo limite para execução
4	*noResourcesLeft*	falta de recursos de sistema
5	*languageError*	retornado erro pela linguagem de programação
6	*runtimeError*	erro em tempo de execução
7	*invalidArgument*	parâmetros inválidos – o *script* não foi executado
8	*securityViolation*	violação de segurança
9	*genericError*	Motivo desconhecido ou não especificado

smRunResult: contém o resultado da execução do *script* quando este foi encerrado normalmente, sem erros (*smRunExitCode* = *noError* (1)). Este objeto pode ter valores válidos antes mesmo do *script* ter concluído sua execução pois o *script* poderá estar programado para retornar resultados parciais. Qualquer resultado retornado pelo *script*, seja ele resultado final ou parcial, estará disponível sempre neste objeto.

smRunResultTime: este objeto conterá valor que representa data e hora em que o resultado (*smRunResult*) foi atualizado. Através de consultas a esse objeto pode-se verificar se o resultado parcial foi atualizado.

smRunError: este objeto conterá uma mensagem descrevendo o erro se o *script* não iniciou ou se o *script* terminou de forma inesperado, retornando como código de saída (*smRunExitCode*) o valor 9 (*genericError*).

smRunErrorTime: este objeto conterá valor que representa data e hora em que a mensagem de erro (*smRunError*) foi atualizada.

Todas essas informações apresentadas que detalham o funcionamento da MIB Script são de grande importância pois é com base nelas que a aplicação desenvolvida e apresentada no capítulo 5 vai balizar seu funcionamento.

3.2 JASMIN

O JASMIN (JAva Script Mib ImplementatioN) é um projeto da NEC (NEC C&C RESEARCH LABORATORIES, 2006) e Universidade de Braunschweig (TECHNICAL UNIVERSITY OF BRAUNSCHWEIG, 2006), desenvolvido entre 1999 e 2001, que implementa a MIB Script e é a única implementação conhecida da MIB Script. O principal objetivo do projeto Jasmin era produzir uma implementação para avaliar os conceitos do gerenciamento por delegação – em particular da MIB Script. O agente SNMP do projeto Jasmin foi inicialmente desenvolvido em estações SPARC com Sun Solaris 2.x e em PCs x86 com Linux e considerado portável para outras plataformas UNIX pela forma de instalação.

3.2.1 Instalação do JASMIN

Faz parte da documentação do Jasmin um capítulo que trata detalhadamente a instalação da ferramenta, incluindo requisitos, configurações, uso e como iniciar o agente SNMP com MIB Script. Mesmo assim, a instalação da ferramenta foi a tarefa mais difícil de ser concluída dentre as tarefas realizadas durante a pesquisa e o desenvolvimento deste trabalho. Isso aconteceu devido aos pré-requisitos para a instalação do Jasmin que exige versões específicas de algumas ferramentas. Isso será brevemente descrito agora.

Além do código fonte do Jasmin, disponível no site do projeto (JASMIN, 2000), a instalação possui os seguintes pré-requisitos:

- Java JDK 1.1.x ou JRE 1.1.x
- Net-SNMP 4.2 ou superior
- GNU wget versão 1.5.3
- Ferramentas de desenvolvimento como gcc, awk, lex, yacc e patch, além de, automake (pelo menos versão 1.4) e autoconf (pelo menos versão 2.13).

Sendo assim, conseguir essas versões das ferramentas não foi uma tarefa trivial. Além disso, mesmo com essas versões específicas, utilizá-las em sistemas atuais mostrou-se difícil, sendo que em alguns casos houve incompatibilidades.

Foram realizadas tentativas de instalação conforme o manual de instalação do Jasmin em 6 (seis) diferentes versões e/ou distribuições de Linux (Debian (DEBIAN, 2005), Suse (SUSE, 2005), Mandrake (MANDRIVA, 2005), Red Hat [RED HAT, 2006], Conectiva 10 e Conectiva 6 (MANDRIVA, 2005)).

Por fim, a ferramenta ficou instalada e funcionando na versão 6 da distribuição Conectiva. Isso significa que essa versão atende os requisitos das ferramentas de desenvolvimento.

A versão do wget utilizada foi a 1.5.3 e a versão do Net-SNMP foi a 4.2 (ainda chamado de UCD-SNMP nesta versão). Quanto a versão do Java foi a que precisou de mais tempo de busca da versão adequada e de entendimento das necessidades. Primeiramente foi utilizada a versão 1.1.8 do Java JDK mas ocorriam erros no momento de gerar o código executável do Jasmin. Em outra tentativa, foi utilizado o Java JDK versão 1.2.2 (apesar de os pré-requisitos exigirem versão 1.1 do kit) e a compilação aconteceu com sucesso, gerando o código executável do Jasmin. Mas após concluída a instalação e iniciado o agente SNMP estendido que implementa a MIB Script os *scripts* não eram executados, retornando indicativo de erro na linguagem Java. E de fato, não é possível utilizar o JDK 1.2.2 para a execução dos *scripts*, mas para compilar o código fonte do Jasmin não foi possível utilizar apenas a versão 1.1.8 do JDK. Por fim, o Jasmin ficou instalado após fazer parte da instalação com a versão 1.1.8 e outra parte com a versão 1.2.2 do JDK. Concluída a instalação ficou disponível para uso no sistema novamente a versão 1.1.8 para compilação e execução dos *scripts* criados.

Para uso de comandos SNMP nos *scripts* criados foi necessária a instalação do pacote SNMP AdventNet (ADVENTNET, 2006) que fornece o suporte aos comandos SNMP para a linguagem Java instalada.

3.2.2 Uso do Jasmin

Depois de instalada a ferramenta utilizou-se o tutorial do Jasmin (QUITTEK, 1999) para se ambientar com a ferramenta o que significa principalmente avaliar e executar *scripts* de exemplo. Isso porque o funcionamento do sistema segue as regras estabelecidas pela MIB Script, ou seja, a partir desse momento era utilizado o manual do Jasmin para conhecer detalhes operacionais e a RFC3165 (da MIB Script) para os detalhes funcionais. Mais da metade do tutorial do Jasmin é composto de exemplos e testes sugeridos, o que foi bastante interessante para iniciar o desenvolvimento e criação de novos *scripts.*

Conforme pode ser visto na Figura 3.3 que apresenta a arquitetura do Jasmin, essa implementação suporta múltiplos ambientes de execução para linguagens diferentes através do SMX (*Script MIB Extensibility Protocol*) (SCHÖNWÄLDER; QUITTEK, 1999), que cria uma interface entre o agente e os sistemas de execução. Esse protocolo roda com conexão TCP local, sendo suportado pela maioria das linguagens. Os sistemas de execução são iniciados pelo núcleo do Jasmin e um mecanismo de segurança realiza a autenticação desses sistemas.

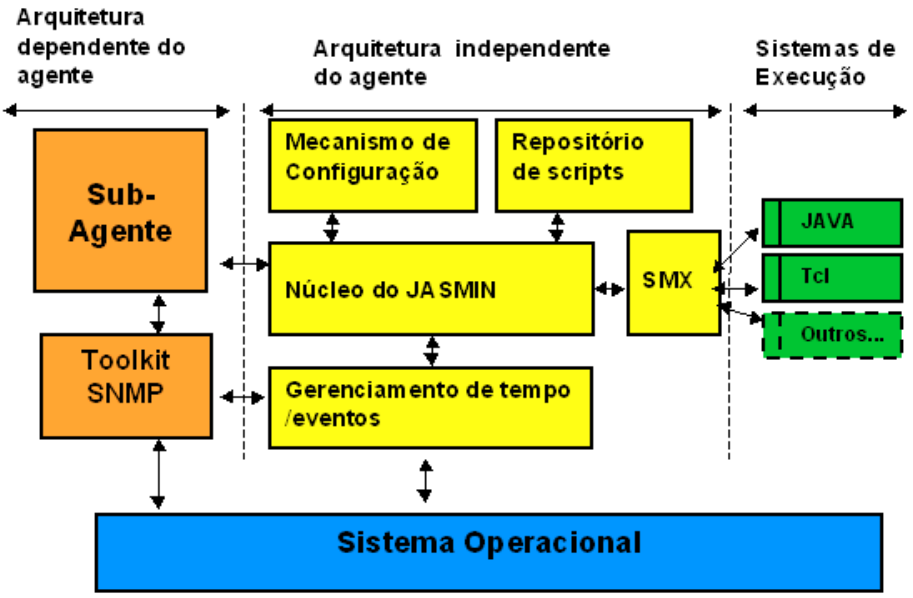


Figura 3.3: Arquitetura do Jasmin

O Jasmin possui dois perfis de segurança. O primeiro perfil é o *operating system security profile*, que especifica quais são os serviços do sistema operacional que são disponibilizados para um sistema de execução. O segundo perfil é o *runtime security profile*, que é dependente do sistema de execução. Nesse perfil, uma *string* contendo as definições de segurança é passada para o sistema de execução via o protocolo SMX e é tarefa do sistema de execução mapear essa *string* para políticas locais de acesso e uso de recursos.

Passadas as “turbulências” da instalação da ferramenta, ela se mostra estável e boa de trabalhar. Ela foi implementada rigorosamente como é especificada a MIB Script e é uma excelente ferramenta para estudo do gerenciamento por delegação, mas com certeza terá que ser atualizada para permitir o uso com sistemas e ferramentas atuais para continuar sendo utilizada e estudada. Desde a implementação do Jasmin ele vem

sendo usado em diversas pesquisas no meio científico como em (STRAUSS, 1999), (VALENTINI; GASPARY, 2003) e (VIANNA et. al, 2007).

O uso do Jasmin se confunde com o uso da MIB Script mas conforme já apresentado, a MIB Script é a definição teórica e o Jasmin a implementação prática. De qualquer forma, apesar de já ter sido detalhado o funcionamento da MIB Script, será revisto agora brevemente o uso da ferramenta com os passos até o momento da execução de um *script* e eventualmente será apresentado algum detalhe relevante sobre a implementação.

A Figura 3.4 apresenta a interação entre os componentes do sistema descrita a seguir.

Um administrador que deseja executar um *script* deve primeiro verificar a tabela *smLangTable*, que lista as linguagens suportadas pela implementação da MIB Script no agente. Além dessa tabela, ele também pode consultar a tabela *smExtsnTable*, que contém as extensões aceitas pelas linguagens instaladas. Este item é importante pois o Jasmin influencia diretamente nessas tabelas, ou seja, é a definição do Jasmin – em sua instalação padrão – que possibilita o uso de duas linguagens: Java e Tcl.

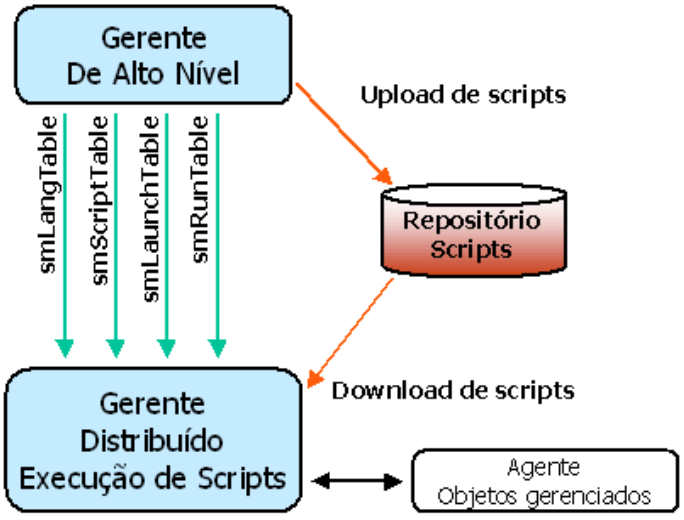


Figura 3.4: Uso do Jasmin

Com base nas informações obtidas nas tabelas de linguagens e extensões, o administrador deve selecionar um *script* apropriado de um repositório para registrá-lo em uma nova linha da tabela *smScriptTable*, que contém a lista com todos os *scripts* conhecidos pelo agente. Esses *scripts* podem ser obtidos através dos modelos *client pull* ou *server push*. O modelo *client pull* requer que apenas um URL seja escrito nessa linha da tabela para que o agente faça o *download* do *script*. Já o modelo *server push* requer que o *script* seja escrito linha a linha na tabela *smCodeTable*, através de PDUs contendo SNMP *SetRequests*. O Jasmin não implementa o modelo *server push* que na MIB Script é definido como opcional. Operações como leitura e deleção de *scripts* também são suportadas por essa tabela.

Para que os *scripts* sejam iniciados é preciso que uma entrada na tabela *smLaunchTable* seja criada. Nessa tabela é descrito o ambiente de execução para cada *script*, incluindo os argumentos que serão passados, o tempo máximo de vida e a identificação do dono do *script*. Várias instâncias de um mesmo *script* podem ser criadas através de uma única entrada nessa tabela e várias entradas dessa tabela podem

apontar para um mesmo *script* na tabela *smScriptTable*. Dessa forma pode-se ter vários *scripts* iguais em execução, mas com argumentos ou permissões diferentes.

Cada *script* disparado possui uma linha criada na tabela *smRunTable* automaticamente. Essa tabela possui a lista de todos os *scripts* que estão executando ou que terminaram a sua execução recentemente. Através dessa tabela o administrador pode controlar os *scripts* em execução, assim como pode obter os resultados intermediários ou finais gerados por eles. O resultado final é mantido na tabela até que o "tempo de vida" da linha expire.

4 GERENCIAMENTO DE NO-BREAKS

De uma forma geral, os sistemas para fornecimento ininterrupto de energia (UPS – *Uninterruptible Power Supply*), conhecidos popularmente no Brasil como *No-breaks* (CP, 2006), possuem como principal função fornecer energia condicionada (estabilizada e filtrada) e sem interrupção para a carga crítica, mesmo durante uma falha da rede comercial.

Conforme mostrado na Figura 4.1, ao receber a energia elétrica da concessionária, o *no-break* transforma esta energia não condicionada, abundante em flutuações, transitórios de tensão e de freqüência, em energia condicionada, onde as características de tensão e freqüência são rigorosamente controladas. Desta forma oferece parâmetros ideais, o que é fundamental para o bom desempenho das cargas sensíveis (críticas).

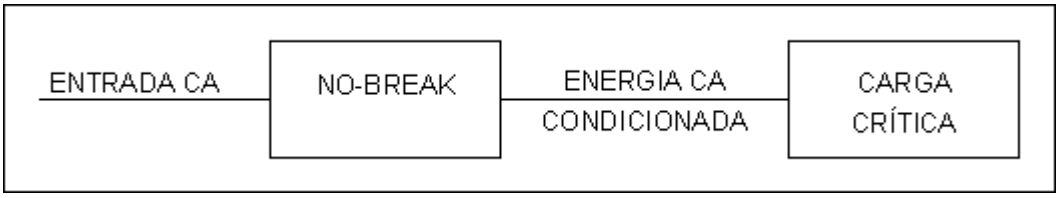


Figura 4.1: Diagrama com no-break alimentando a carga

Transitórios e deformações da forma de onda de tensão, variações de freqüência e mini-interrupções (com duração de até 0,1 segundo) dependem de uma série de fatores, tais como: proximidades de cargas reativas, comutação de cargas na rede, descargas atmosféricas, ruídos, entre outros. Estes fenômenos perturbam a operação e comprometem a confiabilidade dos sistemas computacionais. De acordo com sua magnitude podem, por exemplo, afetar o hardware danificando semicondutores, discos rígidos, cabeças de gravação, etc.

Um sistema *no-break* é composto por circuito retificador/carregador de baterias, banco de baterias, circuito inversor de tensão e chave estática. O circuito retificador/carregador converte tensão alternada em contínua, mantendo o banco de baterias carregado e alimentando o inversor. O banco de baterias armazena energia para alimentar a carga durante falhas da rede elétrica e atua como filtro. O circuito inversor converte tensão contínua proveniente do banco de baterias em tensão alternada para alimentar a carga e a chave estática transfere a carga para a rede em caso de falha no sistema.

Assim como a energia elétrica pode chegar em nossas casas e empresas nas configurações monofásico, bifásico ou trifásico, os no-breaks também possuem essas

características para se adequarem a rede em que serão instalados. E da mesma maneira como é pensado o sistema de energia para suportar a carga em uma residência é pensado o sistema em uma empresa e o no-break apenas acompanha essa filosofia. Em nossas casas, por exemplo, se tivermos poucos equipamentos que serão ligados simultaneamente e de baixa potência, podemos utilizar um sistema monofásico. Mas se a necessidade for maior, precisando de maior potência com diversos equipamentos ligados simultaneamente já será necessário o uso de uma rede bifásica. Grandes empresas comumente utilizam redes trifásicas para atender suas necessidades. Sendo assim, os no-breaks de menor porte (suportando menos carga) em geral são monofásicos ou bifásicos enquanto que os de grande porte são trifásicos.

Na seqüência são apresentados outros termos comuns quando o assunto é no-break e o entendimento desses termos são considerados necessários para o bom entendimento deste trabalho.

A expressão “no-break inteligente” é utilizada para denominar aquele equipamento que possui um controle sobre si mesmo (variáveis e grandezas) e é capaz de se comunicar através de um protocolo pré estabelecido, seja este através da rede TCP/IP, serial RS-232, etc. Contrapondo-se a ele existe o “no-break contato seco” que possui apenas sinalização mecânica de seus alarmes (muitas vezes também transformada em sinalização elétrica e transmitida pela serial, por exemplo) e não possui nenhum controle sobre variáveis importantes no gerenciamento como tensões, correntes, potências, entre outros.

Rede presente e falta de rede: nesses casos o termo rede faz referência a rede elétrica. Sendo assim, “rede presente” indica que existe alimentação na entrada do no-break, ou seja, a concessionária de energia está fornecendo energia elétrica e quando deixar de fornecer utiliza-se o termo “falta de rede”. Nesse caso o no-break passa a consumir carga das baterias.

Níveis da bateria: quando ocorre a falta de energia o no-break supre essa falta retirando energia das baterias para alimentar a carga. A medida que o tempo vai passando e a bateria vai sendo consumida diminui o nível de carga dessa bateria. Existem níveis definidos e devem ser conhecidos além de alguns termos sobre as baterias que são explicados na Tabela 4.1. Além disso, a Figura 4.2 dá uma idéia sobre os níveis definidos e a nomenclatura utilizada para identificar esses níveis – definindo momentos subseqüentes a uma falta de rede (quando a carga é alimentada pelas baterias).

Após análise da Tabela 4.1 podemos concluir que a bateria pode estar sendo carregada, descarregada ou em flutuação e quando ela estiver sendo descarregada, o nível de carga pode ser normal, crítico ou esgotado.

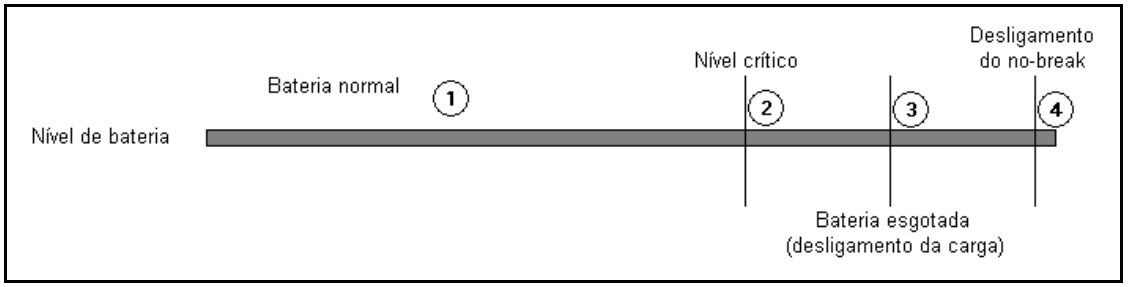


Figura 4.2: Níveis de carga da bateria

Tabela 4.1: Estados da bateria

Situação da baterias	Descrição
Descarregando	Não há rede presente, a carga está sendo alimentada pelas baterias e por isso ela está sendo descarregada
Crítica	O sistema ficou sem energia (e a bateria em descarga) por um tempo próximo do limite máximo. Identificado pelo número 2 na Figura 4.2, indica que em breve a carga não será mais alimentada
Esgotada	Identificado pelo número 3 na Figura 4.2, indica que as baterias não possuem energia para alimentar a carga e a saída do no-break será desligada
Normal	Não há rede presente, a carga está sendo alimentada pelas baterias que ainda não atingiu o nível crítico
Carregando	Depois de utilizar as baterias como fonte de energia, ao retornar a rede na entrada do no-break este passa a carregar a bateria
Flutuação	Ao concluir o processo de carga da bateria (bateria 100% carregada) o no-break mantém essa carga

Bypass (ou operando por bypass) é o termo utilizado quando o no-break está ligado mas, por algum motivo, não está cumprindo suas funções básicas (estabilizar energia, etc). A carga é alimentada com a energia fornecida na entrada, sem nenhum tratamento e nenhuma análise desta e se esta for interrompida a carga será desligada, ou seja, o fornecimento de energia na saída do no-break também será interrompido. E a operação por bypass pode acontecer de forma manual (para uma manutenção, por exemplo) ou de forma automática, quando o no-break detecta em si algum erro que coloca em risco a carga ou o próprio equipamento.

Valores nominais: a Tabela 4.2 apresenta algumas grandezas e explica a definição do seu valor nominal. Em geral o valor nominal define o valor padrão ou o limite para a grandeza.

Tabela 4.2: Detalhes sobre valores nominais

Variável/grandeza	Significado
Tensão Nominal	é a tensão (especificada em (V) Volts) esperada na entrada e/ou na saída do equipamento. Se existir um valor de tensão diferente da nominal e fora da faixa de tolerância ocorre o registro de um alarme (subtensão ou sobretensão).
Freqüência nominal	é a freqüência (em Hz) esperada na entrada e/ou na saída do equipamento. Havendo variação de freqüência ocorre o registro de um alarme.
Potência Nominal	é a carga máxima suportada pelo no-break. Acima desse limite passa a ser indicado alarme de sobrecarga.

4.1 MIB para gerenciamento de No-breaks (UPS-MIB)

Conforme foi apresentado no capítulo 2.1, muitas MIBs são criadas e definidas como padrão para gerenciamento de determinados equipamentos e o mesmo aconteceu com os equipamentos UPS (*Uninterruptible Power Supply*). A UPS MIB define, na RFC1628 (CASE, 1994), uma MIB genérica para a utilização com equipamentos do tipo UPS e pode ser considerada a implementação padrão de uma MIB para UPS. Sendo assim, apesar de muitos fabricantes de equipamentos definirem suas próprias MIBs personalizadas e ajustadas a seus equipamentos, muitas delas são parecidas e fortemente baseadas na UPS MIB.

Em alguns casos, além de criar uma MIB baseada na UPS-MIB (e até por este motivo), os fabricantes de no-breaks também programam seus agentes para responder requisições de objetos da UPS-MIB – além de responder a sua MIB proprietária. Isso torna o agente mais acessível e compatível com outros agentes da rede. Dessa forma, o gerente pode utilizar os objetos da UPS-MIB para monitorar todos os agentes da rede, através dos objetos básicos e comuns, e obter detalhes de alarmes ou informações mais específicas sobre o equipamento através da MIB proprietária fornecida pelo fabricante do equipamento.

Sendo assim, este trabalho utilizará objetos da UPS-MIB na monitoração dos no-breaks e apresenta no capítulo 4.2 os objetos de grande importância na monitoração. Esses objetos são apresentados em (POLINA, 2005) no qual são comparados de forma a verificar se os principais fabricantes do mercado possuem esses objetos em suas MIBs ou respondem a eles através dos objetos da UPS-MIB. Também foi considerado na escolha dos objetos, aqueles que são suportados pela grande maioria dos fabricantes.

Na UPS-MIB estão definidos nove subgrupos que descrevem objetos além de um grupo para as interrupções (*traps*). A seguir é feito breve relato sobre cada um desses grupos.

O grupo *upsObjects* contém os subgrupos que descrevem os objetos apresentados a seguir com uma descrição sobre o subgrupo e suas variáveis.

O grupo ***upsIdent*** tem o objetivo de identificar o *no-break* na rede e isso é feito através dos objetos que informam o nome do fabricante do equipamento (*upsIdentManufacturer*), modelo (*upsIdentModel*), versão do software do no-break (*upsIdentUPSSoftwareVersion*), versão do software do agente (*upsIdentAgentSoftwareVersion*), além de um nome para fácil identificação (*upsIdentName*) e a informação sobre os equipamentos que estão conectados a esse no-break (*upsIdentAttachedDevices*). Essas duas últimas informações são ajustadas pelo gerente do sistema.

O grupo ***upsBattery*** fornece informações sobre o estado do banco de baterias e sobre as condições da energia retirada deste para alimentar a carga. O primeiro objeto deste grupo (*upsBatteryStatus*) informa o status geral do banco de baterias, ou seja, o nível de carga das baterias, que pode estar indicado como normal (*Normal*), baixo (*Low*) ou esgotado (*Depleted*). O objeto seguinte (*upsSecondsOnBattery*) informa o número de segundos transcorridos desde o momento em que ocorreu a falta de energia, ou seja, o no-break não está sendo alimentado e a carga do no-break está sendo alimentada pelas baterias. É possível ainda obter a quantidade de minutos restante estimada (*upsEstimatedMinutesRemaining*) que o banco de baterias manterá a carga, o valor percentual da carga estimada restante no banco de baterias

(*upsEstimatedChargeRemaining*) além dos valores da tensão (*upsBatteryVoltage*) e corrente retirada (*upsBatteryCurrent*) do banco de baterias. Ainda nesse mesmo grupo pode ser informado o valor da temperatura no banco de baterias ou a temperatura ambiente próximo a ele.

O grupo ***upsInput*** informa como e com que qualidade está sendo fornecida a energia externa, pela concessionária, na entrada do *no-break*. Adicionalmente, o primeiro objeto deste grupo é um contador que informa o número de vezes que foi verificada a ausência de energia na entrada do no-break e o segundo objeto informa o número de linhas (fases) na entrada do no-break que também é o número de linhas que existirão na tabela do grupo *upsInput*. Ou seja, para cada fase de entrada existirá uma linha na tabela *upsInputTable* com informações de freqüência (*upsInputFrequency*), tensão (*upsInputVoltage*), corrente (*upsInputCurrent*), e potência real (*upsInputTruePower*).

O grupo ***upsOutput*** apresenta as condições da energia fornecida pelo *no-break* à carga. O primeiro objeto (*upsOutputSource*) informa a fonte de energia (por exemplo: bypass, baterias, etc). É também informada a freqüência de saída (*upsOutputFrequency*) e a quantidades de fases que o equipamento possui – que da mesma forma como ocorre no grupo *upsInput* indica o número de linhas na tabela *upsOutputTable*. Nessa tabela serão disponibilizadas as informações de tensão (*upsOutputVoltage*), corrente (*upsOutputCurrent*), potência real (*upsOutputPower*) e porcentagem de carga presente (*upsOutputPercentLoad*) para cada fase.

O grupo ***upsBypass*** informa estado e condições da fonte alternativa utilizada quando o equipamento está operando no modo *Bypass* (será utilizado o termo *Bypass* em inglês para identificar que a funcionalidade do no-break foi desativada e a saída do no-break que alimenta a carga foi desviada para ser alimentada diretamente pela fonte externa ou rede alternativa). Neste grupo é informado a freqüência (*upsBypassFrequency*) além de tensão (*upsBypassVoltage*), corrente (*upsBypassCurrent*) e potência real transferida para o *bypass* (*upsBypassPower*) para cada uma das fases.

O grupo ***upsAlarm*** informa a quantidade (*upsAlarmsPresent*) e quais são os alarmes que estão ativos. Além disso, são definidos vinte e quatro alarmes como “bem conhecidos” (*upsWellKnownAlarms*) para que se possa verificar o status de cada um deles em particular, sempre que for desejado/necessário. Esses alarmes são apresentados na Tabela 4.3.

O grupo ***upsTest*** apresenta resultados de testes do equipamento e permite que o gerente coloque em ação ou verifique resultados de um teste realizado pelo *no-break*.

O grupo ***upsControl*** oferece ao gerente um domínio do *no-break* podendo, por exemplo, desligá-lo remotamente. O primeiro objeto do grupo é *upsShutdownType* e indica o tipo de desligamento que será feito, ou seja, se será desligado apenas a saída (alimentação da carga) ou todo o sistema do no-break. E é o administrador do sistema (gerente) que define isso, alterando esse objeto como desejar. Através do objeto *upsShutdownAfterDelay* o administrador aciona o desligamento que ocorrerá da forma como definida no objeto anterior e após aguardar o número de segundos especificado neste objeto.

O grupo ***upsConfig*** permite ao gerente configurar os limites do no-break para identificação de operação em condições anormais.

Tabela 4.3: Alarmes definidos na UPS-MIB

Necessidade de trocar o banco de baterias por se encontrar em estado ruim (*upsAlarmBatteryBad*)
Carga sendo alimentada pelo banco de baterias (*upsAlarmOnBattery*)
Banco de baterias em nível crítico (*upsAlarmLowBattery*)
Baterias esgotadas (*upsAlarmDepletedBattery*)
Temperatura atingiu nível intolerável (*upsAlarmTempBad*)
Alimentação na entrada em condição inaceitável (*upsAlarmInputBad*)
Alimentação na saída em condição inaceitável (*upsAlarmOutputBad*)
Sobrecarga na saída do equipamento (*upsAlarmOutputOverload*)
Ativado modo de operação por *bypass* (*upsAlarmOnBypass*)
Problemas com a fonte de energia na rede alternativa (*upsAlarmBypassBad*)
Solicitado desligamento de energia na saída do equipamento (*upsAlarmOutputOffAsRequested*)
Solicitado desligamento do equipamento (*upsAlarmUpsOffAsRequested*)
Falha no sistema de recarga do no-break (*upsAlarmChargerFailed*)
A saída do no-break foi desligada (*upsAlarmUpsOutputOff*)
No-break completamente desligado (*upsAlarmUpsSystemOff*)
Falha no sistema de ventilação do no-break (*upsAlarmFanFailure*)
Falha em um fusível do equipamento (*upsAlarmFuseFailure*)
Falha genérica no equipamento (*upsAlarmGeneralFault*)
Falha detectada no último auto-teste realizado (*upsAlarmDiagnosticTestFailed*)
Falha de comunicação entre o agente e o no-break (*upsAlarmCommunicationsLost*)
Aguardando por rede na entrada para poder alimentar a carga pois não há fonte disponível para fornecer energia na saída (*upsAlarmAwaitingPower*)
Agendado desligamento automático do no-break (*upsAlarmShutdownPending*)
Alimentação para a carga será desligada em menos de 5 segundos (*upsAlarmShutdownImminent*)
Teste em andamento (*upsAlarmTestInProgress*)

O grupo *Traps* tem as definições dos alarmes que serão enviados ao gerente para chamar a atenção para alguma exceção que tenha ocorrido. Normalmente, na ocorrência de alguma exceção, é gerado um alarme para o gerente informando tal fato e o gerente deve realizar consultas para identificar o que ocorreu e o que causou a geração da interrupção.

4.2 Objetos utilizados no processo de gerência

Atualmente é comum encontrar no-breaks que possuem total controle sobre si e capacidade de fornecer detalhes bem específicos a respeito de seu estado e funcionamento. Por isso, as variáveis utilizadas na gerência podem ser previstas para disponibilizar grande quantidade de detalhes, permitindo assim que o gerente obtenha desde informações sobre os alarmes, estado e identificação do equipamento, até valores das grandezas físicas do equipamento como tensão de entrada e saída, corrente, quantidade de carga alimentada, entre outros.

Para os testes da implementação deste trabalho serão utilizados dois no-breaks disponibilizados pela empresa CP Eletrônica S.A., de modelos e configurações distintos. Um equipamento é monofásico, modelo Breakless New e outro é trifásico, modelo Top-DSP. A especificação básica de cada um deles é apresentada na Tabela 4.4. Esses no-breaks possuem agentes SNMP integrados que respondem tanto a uma MIB proprietária (CpUpsMib) da CP Eletrônica quanto a UPS-MIB padrão da RFC1628. A MIB proprietária da empresa possui objetos que detalham melhor os equipamentos e oferecem maior quantidade de informações (POLINA, 2005).

Tabela 4.4: Especificação dos no-breaks utilizados para teste no gerenciamento

Informação	Equipamento	
	Breakless New	Top-DSP
IdentName SNMP	Agent02 - BrkNew	Agent01 - TOPDSP
Modelo	1660AINEW	TOPDSP28300
Potência nominal	6,0KVA	30,0KVA
Configuração	Monofásica	Trifásica
Tensão nominal de entrada	220V	220V
Tensão nominal de saída	120V	220V
Peso	68Kg	475Kg
Dimensões (Alt.xLarg.xProf.)	555 x 295 x 490 mm	1,40 x 0,55 x 0,66 m

Neste capítulo serão apresentados os objetos da UPS-MIB (eventualmente algum objeto específico de algum fabricante) que são considerados de grande importância para utilização na monitoração pelo sistema de gerenciamento que será proposto adiante neste trabalho ou que podem ter grande utilidade para obter informações adicionais a respeito do sistema. Os objetos serão apresentados separados por grupos e juntamente com o objeto e uma breve descrição será apresentada uma possível forma de uso do objeto na monitoração.

Objetos do **grupo Identificação**: provavelmente a informação mais importante desse grupo é o objeto *upsIdentName.* Ele é uma string de 64 bytes que identifica o equipamento e pode ser alterado pelo gerente, ou seja, o administrador ajusta este objeto com um nome ou valor para que ele seja capaz de identificá-lo da maneira como achar melhor. Outras informações que podem ser coletadas neste grupo para melhor identificar o agente são *upsIdentManufacturer* e *upsIdentModel* com a identificação do

fabricante e modelo do equipamento, respectivamente. Essas informações podem também ser importantes em situações de falhas do agente que necessitam de intervenção do fabricante do equipamento para dar o suporte necessário. Nesses casos há uma outra informação que normalmente está presente neste grupo que é o número de série do equipamento (*upsIdentSerialNumber*). Este objeto não faz parte da definição da UPS-MIB mas já está presente na maioria das MIBs proprietárias.

Objetos do **grupo Bateria**: a principal informação deste grupo está no objeto *upsBatteryStatus*, informando o estado atual do banco de baterias. Ele é essencial no momento em que ocorre a interrupção no fornecimento de energia e a alimentação da carga se dá a partir das baterias. Os resultados possíveis para esse objeto são *unknown*(1) quando não é possível determinar o estado da bateria (espera-se que este seja um estado transitório), *batteryNormal*(2) quando a bateria é considerada em condição de suportar a carga, *batteryLow*(3) é indicado quando a bateria estiver descarregada a um nível definido e próximo de seu limite mínimo e *batteryDepleted*(4) indicado quando a bateria chegou em seu nível mínimo e não suporta mais a carga. O estado *batteryLow* é considerado extremamente importante pois é com essa sinalização que deve-se tomar a ação de desligar os equipamentos alimentados por este no-break, ou seja, a bateria está quase totalmente descarregada e em breve não será mais capaz de prover energia para alimentar a carga. Ao chegar no nível *batteryDepleted* o no-break deixa de alimentar a carga (caso ainda exista algum equipamento ligado na saída do no-break) e, dependendo do modelo, pode até se desligar, aguardando o retorno da rede para se ligar novamente.

Nesse grupo também é possível obter tensão e a corrente do banco de baterias através do objetos *upsBatteryVoltage* e *upsBatteryCurrent*, respectivamente, além das informações *upsBatteryTimeOnBattery*, *upsBatteryEstimatedMinutesRemaining* e *upsBatteryEstimatedChargeRemaining* que informam respectivamente, por quanto tempo a bateria já está alimentando a carga, autonomia restante da bateria em minutos e qual a carga da bateria (quanto resta de autonomia) em porcentagem. É comum encontrar equipamentos que informam a autonomia restante apenas em minutos ou apenas em porcentagem.

O banco de baterias do no-break possui uma vida útil muito menor que a do restante do hardware do no-break o que torna comum a prática de troca do banco de baterias, ou seja, quando um conjunto de baterias não consegue mais manter energia para alimentar a carga é dito que o banco de baterias está ruim. Hoje em dia alguns equipamentos já são capazes de detectar isso e informar ao usuário que é necessário trocar o banco de baterias. Por isso, apesar de não existir essa indicação na UPS-MIB, diversos fabricantes já definiram e utilizam os objetos *upsBatteryReplaceIndicator* e *upsBatteryLastReplaceDate* neste grupo. O primeiro objeto é o que irá indicar a necessidade da troca do banco e o segundo mantém a data da última troca realizada.

Objetos do **grupo Entrada**: os objetos desse grupo são de extrema importância mas as anormalidades com eles são monitoradas através de alarmes em outros grupos como por exemplo, tensão de entrada (*upsInputVoltage*) passa a ser um alarme quando chega a nível tão baixo que o equipamento considera “Falta de rede”, gerando o alarme falta de rede e indicando que a carga será alimentada pela bateria. Por isso, os objetos desse grupo são utilizados basicamente na coleta e armazenagem de dados para se construir um histórico desses valores ao longo do tempo, analisando variações. Para consultar e armazenar esses valores deve-se primeiramente obter a quantidade de fases na entrada do equipamento (*upsInputNumPhases*) para depois obter os valores desejados de cada

uma das fases. Nesse grupo pode-se obter os valores de tensão (*upsInputVoltage*), corrente (*upsInputCurrent*), freqüência (*upsInputFrequency*), potência real (*upsInputTruePower*) e potência aparente (*upsInputApparentPower*). Este último valor não está definido na UPS-MIB mas diversos fabricantes já disponibilizam esse valor, principalmente pela maior importância que se dá a cada dia para a razão entre potência real e potência aparente, o chamado fator de potência (MARTINS; GABIATTI; BONAN, 2006).

Objetos do **grupo Saída**: os objetos desse grupo possuem basicamente a mesma importância dos objetos do grupo entrada, ou seja, é interessante manter histórico para avaliar ocorrências e variações e a consulta deve igualmente ser feita por fase que se deseja consultar. Valores que são interessante armazenar para esse fim: freqüência (*upsOutputFrequency*), tensão (*upsOutputVoltage*), corrente (*upsOutputCurrent*), potência real (*upsOutputTruePower*), potência aparente (*upsOutputApparentPower*) e carga (*upsOutputPercentLoad*). Este último valor também pode ser monitorado para se identificar excesso de carga na saída do equipamento (sobrecarga). Ele é obtido em porcentagem e pode retornar um valor de 0 a 200, ou seja, um valor acima de 100 (100%) indica que o no-break está operando com carga maior que sua capacidade nominal e um alarme indicando isso será ativado.

Alarmes conhecidos (pasta *upsWellKnownAlarms* do grupo *upsAlarm*): os objetos dessa pasta são os principais alvos da monitoração SNMP já que nela é possível encontrar os principais alarmes gerados pelo equipamento, com destaque para *upsAlarmOnBattery* que indica quando ocorreu falta de alimentação na entrada (falta de energia) e o no-break está operando em baterias, *upsAlarmLowBattery* sinaliza quando foi atingido nível crítico da bateria, indicando que em breve as baterias não conseguirão mais fornecer energia para a carga, *upsAlarmOutputOverload* indicando sobrecarga, *upsAlarmOutputOffAsRequested* sinaliza que foi solicitado desligamento da saída, *upsAlarmUpsOutputOff* é ativado quando saída for desligada e *upsAlarmCommunicationsLost* indica que ocorreu uma falha de comunicação entre agente e no-break.

Objetos do **grupo Controle** podem ser usados para interagir com o no-break, solicitando ligamento ou desligamento do equipamento. Para desligar o equipamento, primeiramente se configura o tipo de desligamento no objeto *upsShutdownType*, que indicará através do valor 1 (*output*) se será feito o desligamento da saída do equipamento ou se o próprio no-break será desligado – indicado pelo valor 2 (*system*). Em seguida é utilizado o objeto *upsShutdownAfterDelay* para programar o desligamento que pode ser imediato – gravando 0 (zero) neste objeto – ou depois de algum tempo, informando neste objeto a quantidade de segundos que deve esperar para efetivar o desligamento.

Para ligar o no-break utiliza-se o objeto *upsStartupAfterDelay*. De forma análoga ao objeto para desligamento, para ligar o equipamento imediatamente grava-se o valor 0 (zero) no objeto ou grava-se a quantidade de segundos que o agente deve aguardar para ligar o equipamento.

É possível também fazer um "*reboot*" no equipamento. Um único comando que fará um desligamento imediato, espera pelo tempo determinado (gravado no objeto) seguido de um ligamento. Isso é feito através do objeto *upsRebootWithDuration* e esse processo também realiza o desligamento de acordo com o escolhido em *upsShutdownType*.

5 SOLUÇÃO PARA GERENCIAMENTO DISTRIBUÍDO DE ENERGIA

Tudo o que foi apresentado até este momento no trabalho serve de base para o planejamento, implantação e execução da aplicação de gerenciamento. O gerenciamento prevê monitoração e controle de toda a rede, e para isso utilizará o protocolo de gerenciamento SNMP que se tornou padrão de fato para redes TCP/IP. Esse gerenciamento utilizará o conceito padrão de gerentes e agentes SNMP mas ao invés de utilizar o paradigma centralizado, mais comum e conhecido, propõe o gerenciamento distribuído através da delegação de tarefas para gerentes intermediários. A Figura 5.1 apresenta o ambiente de gerenciamento sob o ponto de vista das ferramentas utilizadas. Para a monitoração dos agentes capazes de retornar informações importantes a respeito do no-break ao qual estão acoplados é possível utilizar apenas objetos gerenciáveis definidos na MIB padrão para no-breaks (UPS-MIB). Com isso tudo apresentado, neste capítulo será proposto e mostrado a implementação desse sistema de gerenciamento por delegação.

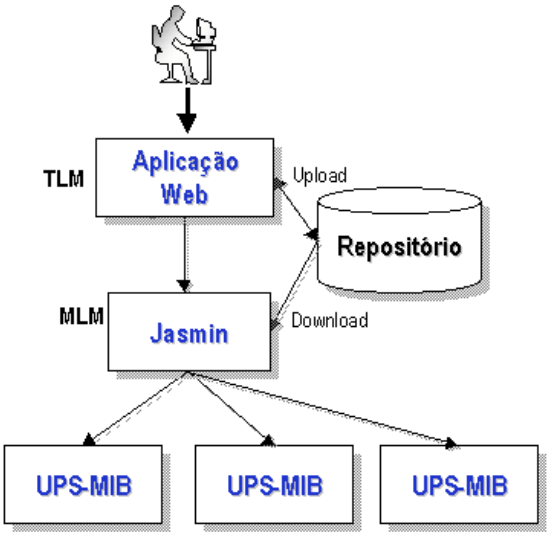


Figura 5.1: Ambiente de gerenciamento

Todas as ferramentas e conceitos apresentados nos capítulos anteriores foram utilizados no desenvolvimento da aplicação de gerenciamento proposta neste trabalho e serão agora apresentadas dentro de suas funcionalidades para a aplicação.

5.1 Estrutura da rede

O sistema de gerenciamento proposto prevê basicamente 4 elementos na rede a ser gerenciada. Dois desses elementos são os no-breaks e os servidores ou estações que são alimentados por esses no-breaks. Isso significa que esses dois elementos já fazem parte da infra-estrutura ou estrutura de rede de uma corporação ou entidade que pretende ter seus dispositivos gerenciados. Os outros elementos desse sistema de gerenciamento são gerentes em dois níveis de subordinação.

Em grandes redes, é previsto e provável que exista mais de um no-break para manter e alimentar os equipamentos da rede. Isso significa que existirão diversos agentes sendo monitorados já que cada no-break contém um agente SNMP. Sendo assim, o gerenciamento deverá ser responsável por diversos agentes (no-breaks) e estes por sua vez são responsáveis por diversos equipamentos, ou seja, são responsáveis por fornecer energia a diversos equipamentos.

Mas mesmo um sistema que prevê o fornecimento de energia para a carga quando a concessionária de energia elétrica não estiver fornecendo pode falhar ou esgotar seus limites de fornecimento de energia alternativa e isso também será monitorado e foi previsto na aplicação de gerenciamento. Como em qualquer sistema de gerenciamento, a preocupação principal é quanto a segurança dos equipamentos e das informações mantidas por eles, ou seja, os equipamentos mais críticos não devem ficar sem alimentação de energia, para que os recursos que eles oferecem estejam disponíveis. No pior caso, se não houver como manter os equipamentos alimentados, eles devem ser desligados de maneira correta e segura, para que mantenham suas informações sem risco de perda ou inconsistência.

Isso significa que além de monitorar e gerenciar os agentes foco do sistema de gerenciamento, ou seja, os no-breaks, o gerenciamento também pretende a interação com os dispositivos fim de uma rede, ou seja, estações de trabalho e de serviços.

Significa que cada grupo de máquinas possui um no-break associado e para cada no-break existirá uma estação de gerenciamento associada (Figura 5.2). Essa estação de gerenciamento será responsável pela monitoração do agente SNMP do no-break e também responsável por ações – no quesito fornecimento de energia – que devam ser tomadas nas máquinas que dependem desse no-break.

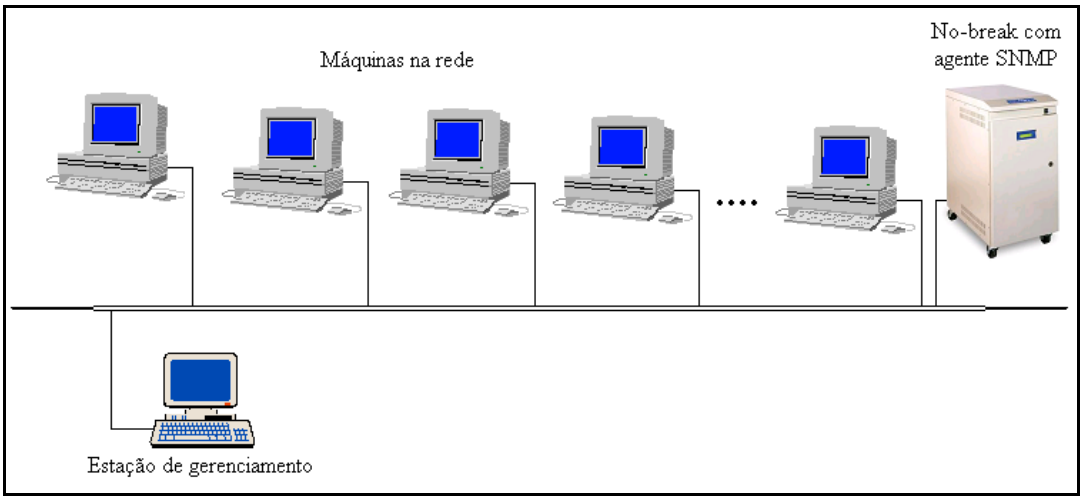


Figura 5.2: Exemplo de estrutura da rede com 1 no-break

Mas essa relação de no-break com estação de gerenciamento, em geral será de N para 1, ou seja, cada no-break está associado para ser gerenciado por uma estação de gerenciamento mas essa estação de gerenciamento pode (e em geral é o que acontece) estar responsável pelo gerenciamento de mais de um no-break, conforme é mostrado na Figura 5.3. Nessa figura, utilizou-se letras e números para identificar estações na rede e no-breaks de forma a identificar quais equipamentos dependem de cada no-break. As estações identificadas pelo número 1 (A1, B1, C1, etc) são alimentadas pela energia mantida pelo no-break 1 (NB1), as estações identificadas pelo número 2 (A2, B2, etc) são alimentadas pela energia mantida pelo no-break 2 (NB2) e assim por diante.

A quantidade de máquinas sendo alimentadas por um no-break pode variar bastante pois vários motivos podem ser levados em conta para que seja definida essa relação.

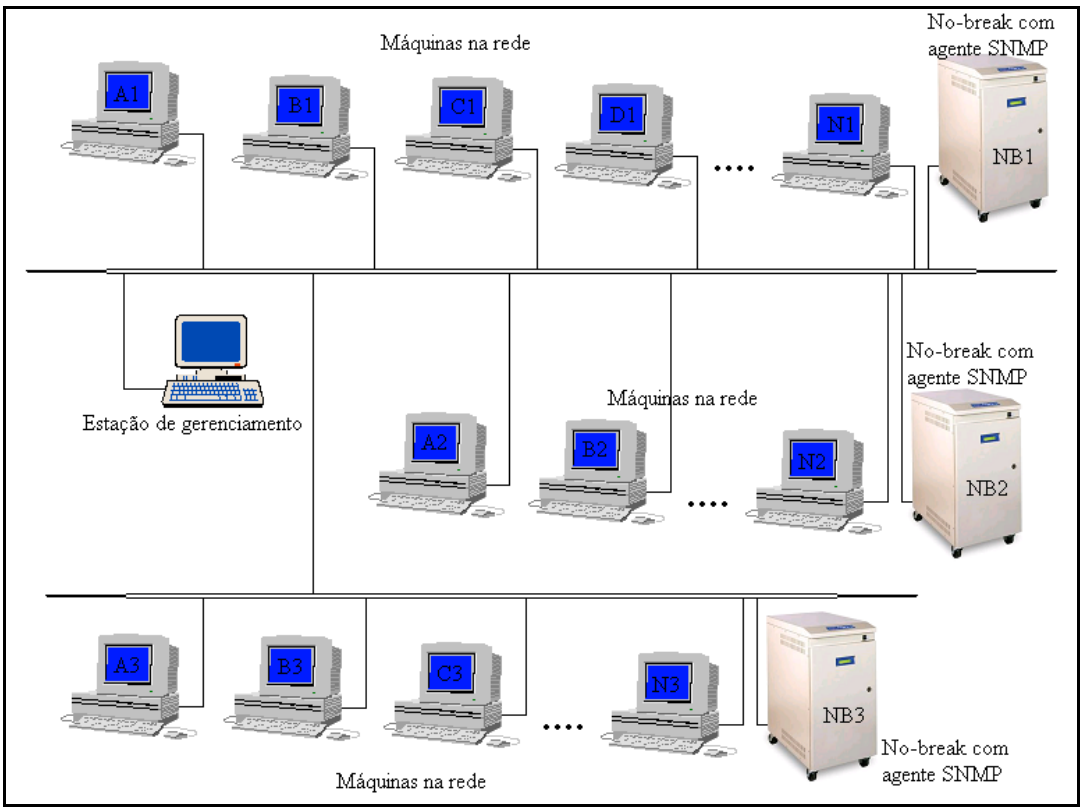


Figura 5.3: Exemplo de estrutura da rede com vários no-breaks

1) Capacidade do no-break: todo no-break é projetado e fabricado para ser capaz de prover energia para um conjunto de carga de acordo com a sua potência. Sendo assim, como existem no mercado equipamentos das mais diversas capacidades (POLINA, 2005), é este o maior motivo para a variação da quantidade de computadores dependentes de um no-break.

2) Autonomia desejada para o no-break e sua carga: todo no-break possui um banco de baterias ligado a ele e é dessas baterias que ele retira energia (CC que é convertida em CA) para alimentar a carga quando ocorre a falha no fornecimento de energia (falta de rede na entrada do no-break). Apesar de esse banco de baterias possuir certa independência do restante do sistema do no-break ele normalmente possui uma potência proporcional a potência do no-break para uma autonomia não muito longa – a menos que se tenha especificado o contrário. Isso quer dizer que em alguns casos ao invés de trocar o banco de

baterias (muitas vezes o no-break não suporta que se altere o tipo de bateria utilizada) ou o no-break para permitir uma autonomia maior para a carga, os responsáveis pela infra-estrutura das empresas podem diminuir a quantidade de equipamentos ligados em um determinado no-break. Diminuindo a carga alimentada pelo no-break, aumenta-se o tempo de autonomia deste, ou seja, a bateria será consumida mais lentamente pois não será necessário retirar muita corrente por unidade de tempo em que a rede principal estiver ausente.

3) Máquinas mais críticas podem ser alimentadas por um subsistema mais robusto e tolerante. Servidores da rede e máquinas consideradas mais importante que outras na rede podem ser alimentadas por no-breaks distintos, especiais ou apenas agrupados em quantidades menores. Por exemplo: a infra-estrutura prevê que, por padrão, em cada no-break serão ligados 100 estações, mas para os servidores a regra é distinta e será utilizado um no-break para cada 5 servidores – ou ainda, alguns mais críticos podem ser ligados sozinho em um no-break, ou com no-breaks redundantes.

E com a grande quantidade de no-breaks e estações de trabalho alimentadas por eles, deve-se aumentar e distribuir ao longo da rede as estações de gerenciamento. Estações que receberão tarefas na forma de *scripts* e executarão essas tarefas para colaborar com o gerenciamento de toda a rede de computadores. Não fazendo essa distribuição, ou seja, mantendo uma única estação de gerenciamento para todos os no-breaks estaríamos utilizando o paradigma centralizado com características e limitações conforme apresentadas no capítulo 2.3.

A Figura 5.4 apresenta uma imagem didática de uma rede completa sendo gerenciada pelo sistema distribuído proposto. A figura está dividida em 4 “níveis”: A, B C e D. No nível A aparece uma única máquina e a representação de uma pessoa. De fato, o que se busca representar nesse nível é que uma pessoa (gerente ou administrador) deve operar uma estação de gerenciamento para que a partir dessa estação todas as políticas de gerenciamento da rede sejam distribuídas para o restante das estações de gerenciamento na rede. Esta estação de gerenciamento do nível A é o gerente TLM responsável por “programar” toda a rede. Em geral essa tarefa precisa ser feita uma única vez (posteriormente apenas ajustes no gerenciamento dessa rede podem ser necessários).

No nível B da Figura 5.4 aparecem outras duas estações de gerenciamento. Essas estações são os gerentes intermediários, ou seja, os gerentes MLM. Nessa figura são apresentadas duas estações para cumprir esse papel mas essa quantidade é apenas um exemplo e atém mesmo nesse exemplo essa quantidade poderia ser menor ou maior. Ou seja, são dois gerentes intermediários para gerenciar cinco agentes mas poderiam ser utilizados três ou mais gerentes intermediários para essa quantidade de agentes ou até mesmo um único gerente já que a quantidade de agentes é pequena. Com esse exemplo e essas colocações também podemos concluir que para uma rede pequena, como a mostrada no exemplo, não existe grande motivo para uso de gerenciamento distribuído – mesmo que possível. Vale lembrar que a figura do exemplo é apenas para tornar mais didática a explicação e que o foco do sistema de gerenciamento é uma grande rede com uma quantidade de agentes muito maior. No exemplo, a estação MLM GI1 (Gerente Intermediário 1) está responsável pela monitoração de três agentes (no-break 1, 2 e 3) enquanto a estação MLM GI2 está responsável pelo gerenciamento de dois agentes (no-break 4 e 5).

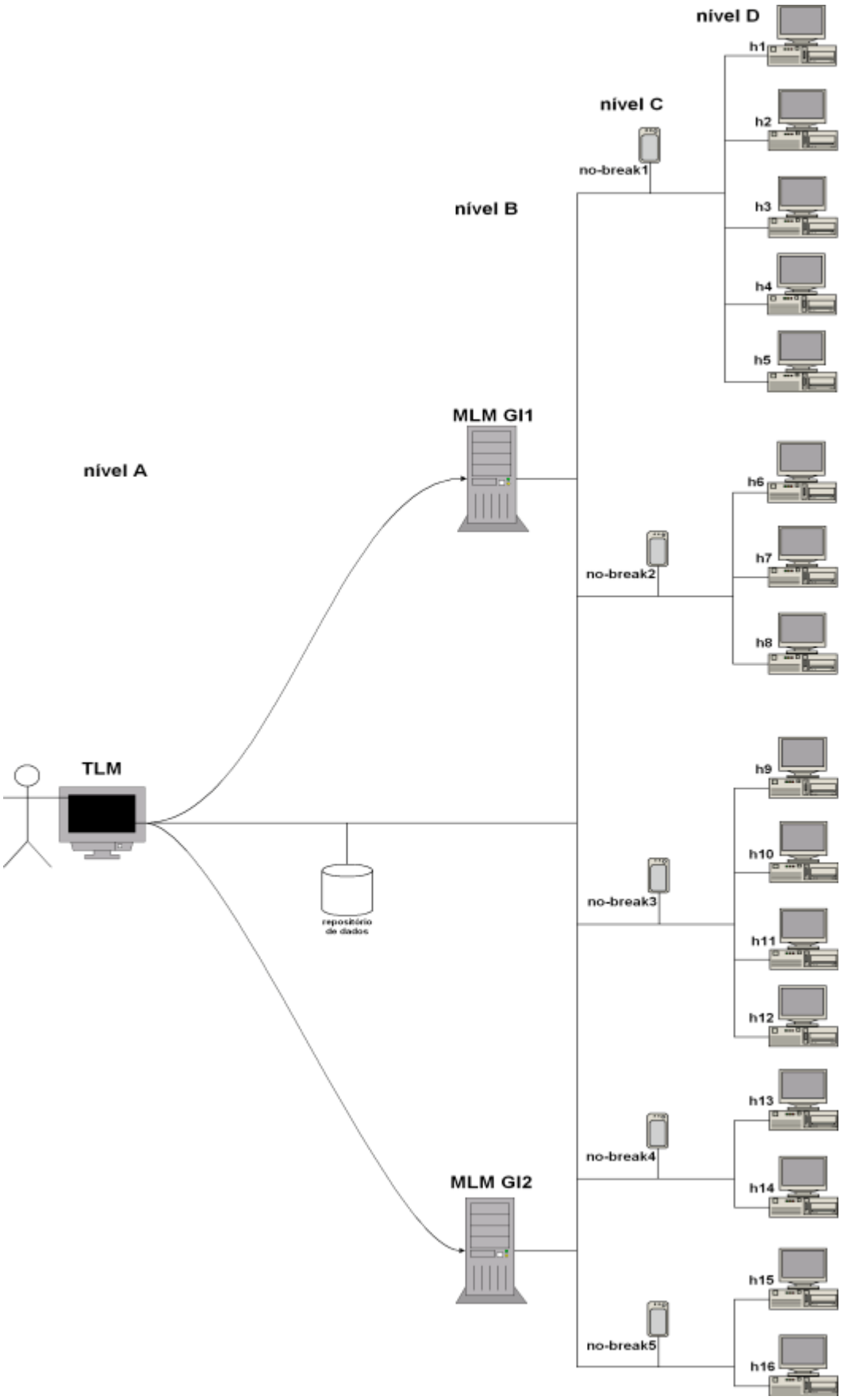


Figura 5.4: Rede com todos os itens sugeridos no gerenciamento

Conforme dito anteriormente, o sistema de gerência têm interesse e preocupação também com a carga dos no-breaks, ou seja, com as estações que são alimentadas por cada um dos no-breaks gerenciados. Sendo assim, e por transição, cada gerente será responsável pela tomada de ações nas estações que são alimentadas por no-breaks pelos quais é responsável. Ou seja, a estação de gerenciamento GI1 é responsável por 12 estações (*host* 1 a 12) enquanto o gerente GI2 é responsável por 4 estações (*host* 13 a 16).

O gerente intermediário (GI1 e GI2) é uma estação que implementa o agente da MIB Script. Isso significa que nela está instalada a ferramenta JASMIN. Esse gerente executará os *scripts* conforme definidos pelo gerente TLM e informados ao gerente MLM através de comandos SNMP.

O nível C da Figura 5.4 apresenta os agentes SNMP que são gerenciados na rede. Neste exemplo aparecem cinco agentes, ou seja, cinco no-breaks que foram numerados para facilitar a explicação e exemplificação da aplicação. Esses agentes poderiam ser quaisquer agentes conhecidos de no-breaks, conforme avaliados em (POLINA, 2005). Mas como verificado em (POLINA, 2005), encontram-se na UPS-MIB os objetos básicos e extremamente necessários para a monitoração de no-breaks, e é por isso que neste trabalho o agente SNMP do no-break será tratado como sendo um agente que mantém dados e responde a requisições de objetos da UPS-MIB (definida na RFC1628). Isso também deixa evidente a importância da UPS-MIB e que mesmo com a inclusão de inúmeros novos recursos nos no-breaks modernos atuais, a MIB definida há mais de uma década continua sendo importante e capaz de ser utilizada em um sistema de gerenciamento. Por outro lado, muitos objetos pesquisados em (POLINA, 2005), são de grande valia, mostrando que é importante continuar o investimento e a evolução de um produto como o no-break (e o agente SNMP do no-break), que cada vez mais se consolida como de extrema importância para uma rede institucional ou corporativa.

No nível D da figura exemplo encontram-se os *hosts* que recebem a energia condicionada fornecida pelos no-breaks, ou seja, energia fornecida pelo equipamento que possui o agente SNMP (objetivo do gerenciamento desse sistema). Todas essas estações (*hosts*) – assim como todos os demais componentes da rede mostrados no exemplo – estão no mesmo barramento de rede, ou seja, fisicamente eles pertencem a mesma rede lógica (rede IP). Mas o fornecimento de energia para essas estações está dividido de forma diferente e identificados os grupos através da sinalização H1, H2, H3 e assim por diante. H1 é o grupo 1 de *hosts* e esses são alimentados pelo no-break 1, H2 é o grupo 2 de *hosts* e esses são alimentados pelo no-break 2, e assim sucessivamente. Isso é importante saber para que seja possível executar ações nas estações corretas quando for necessário interagir com as estações afetadas por uma falha no no-break 2, por exemplo.

5.2 Projeto da aplicação

Neste capítulo será apresentado o projeto que foi proposto e desenvolvido durante a confecção deste trabalho. É o grande objetivo deste trabalho. Com base em todas as informações coletadas e já apresentadas nos capítulos anteriores, foi feita análise verificando possíveis formas como a aplicação poderia ser desenvolvida e como ela poderia ou deveria se comportar. Como já dito anteriormente, o objetivo da aplicação é permitir o gerenciamento de no-breaks de forma que escalabilidade e flexibilidade não sejam problemas substanciais que possam inviabilizar a implantação do projeto. Isso

acontece principalmente pelo fato de se utilizar o paradigma de gerenciamento distribuído, conforme visto. Dessa forma, ao aumentar muito o número de agentes (no-breaks) que precisam ser monitorados deve-se aumentar também o número de gerentes para que o gerenciamento ainda seja viável.

Nas próximas seções serão apresentados o projeto especificamente, incluindo algumas idéias não aproveitadas – justificando o não uso de tal recurso ou técnica, como foi implementado o projeto e a avaliação da ferramenta desenvolvida, buscando validá-la como uma aplicação para uso com o objetivo a que se propõe.

O projeto do sistema de gerenciamento distribuído por delegação pode ser dividido basicamente em seis partes ou também pode-se dizer que são seis itens do projeto que merecem atenção especial. São elas:

1) Gerente principal: é a aplicação principal, de onde todas as políticas de gerenciamento definidas são aplicadas aos outros gerentes. É o chamado TLM da hierarquia de gerenciamento distribuído. Ao iniciar a implantação do sistema de gerenciamento, é nele que tudo começa. A pessoa responsável pelo gerenciamento da rede, operando o gerente principal definirá **o que** deve ser gerenciado, **quando** os agentes devem ser monitorados, **quem** é responsável pela monitoração, **como** o gerenciamento deve funcionar e que **ações** devem ser tomadas em situações que serão pré-determinadas. E são esses cinco itens (o que, quando, quem, como e ações) as outras cinco partes essenciais ao sistema e que recebem atenção especial durante a fase de planejamento e implementação do projeto.
2) Agente: é o equipamento no-break monitorado. A tendência e expectativa é que a quantidade de agentes monitorados seja da ordem de 1 a 5 mil unidades, espalhadas pela rede, distribuídas de acordo com a necessidade de uso maior ou menor de energia e potência em certa localização da rede. Ao analisar o agente gerenciado, também foi necessária a análise sobre os objetos que o agente deveria ser capaz de responder para permitir um gerenciamento adequado para garantir o sucesso da aplicação e apresentar informações relevantes ao processo.
3) Intervalo de gerenciamento ou “quando os agentes devem ser monitorados”: depois de definir quais são os objetos que devem ser monitorados em cada um dos agentes presentes na rede, verificou-se a necessidade de avaliar intervalo entre duas consultas e esse objeto, ou seja, a cada quanto tempo esse objeto deveria ser consultado para verificar sua informação e se com essa consulta alguma ação deve ser tomada. Esse intervalo de tempo é importante pois a informação de cada objeto pode variar com maior freqüência do que outros, ou seja, deve-se consultar alguns objetos com maior freqüência – a cada 10 segundos, por exemplo – enquanto outros podem ser consultados com uma freqüência muito baixa – a cada hora ou a cada 24 horas, por exemplo.
4) Gerentes intermediários (quem executa o gerenciamento): os agentes intermediários são os componentes do sistema que efetivamente ficam responsáveis por monitorar um ou mais agentes na rede, disponibilizar informações previamente determinadas ao gerente principal e tomar iniciativa para executar ações em função do resultado obtido com a monitoração. Esses gerentes intermediários (*Middle Level Manager* - MLM) recebem do gerente principal os *scripts* que devem ser executados. Nesses *scripts* – ou como parâmetros a eles – estarão especificados os agentes e objetos destes que deverão

ser monitorados, o intervalo entre as consultas e também as ações que devem ser executadas de acordo com o resultado obtido na consulta. A todo esse conjunto de informações, estando elas devidamente organizadas de forma que sejam capazes de permitir a definição de todos esses parâmetros (agente, objeto, intervalo, ação), é atribuído o nome de política de gerenciamento. Todas as políticas de gerenciamento são conhecidas pelo administrador e este distribui cada política ao MLM responsável por ela. O MLM é responsável por colocar em prática a política e o TLM monitora o MLM obtendo resultados intermediários, resultados finais ou até mesmo interrompendo a execução de uma política no MLM.

5) O conjunto de todas as políticas relacionadas e verificadas as dependências e influências de umas sobre as outras é que especifica como irá funcionar o gerenciamento. Isso envolve principalmente decisões do administrador na forma como cria as políticas de gerenciamento e quanto automatizado será todo o processo. O administrador pode optar por automatizar totalmente alguns processos e gerar dependência humana em outros – principalmente para tomada de decisões.

6) As ações tomadas em situações pré-determinadas podem ser de extrema importância para o sistema em execução e também para o futuro do sistema. Essas ações não representam apenas atitudes em casos críticos ou negativos mas também o controle e manutenção de histórico, por exemplo, para uma análise da rede ao longo do tempo. Mas as principais ações esperadas no gerenciamento é para que sejam mantidos os serviços de maior prioridade pelo máximo de tempo possível.

5.3 Implementação do projeto

De acordo com o projeto apresentado anteriormente, serão descritas agora todas as etapas envolvidas na implementação do sistema de gerenciamento, detalhando cada um dos níveis propostos.

Para que fique mais didática a apresentação da aplicação, será apresentado primeiramente o nível do gerente principal e na seqüência será apresentado o nível intermediário – a ordem em que eles são utilizados no sistema.

1) Gerente principal (TLM): é responsável pelo cadastramento dos gerentes intermediários (MLMs). No momento em que um gerente intermediário é cadastrado, a aplicação utilizada realiza o contato com a estação gerente (MLM) para verificar se ela possui todos os requisitos necessários para poder receber instruções que a tornam uma estação de gerenciamento. Cumpridas as exigências, a aplicação aceita que o gerente continue o processo de configuração da aplicação. O segundo passo, após informar a estação que conterá o gerente intermediário, consiste em determinar quais são as funções que serão delegadas para esse gerente MLM. Conforme já visto, essas funções serão repassadas na forma de *scripts* que serão executadas por este gerente. Juntamente com a definição dos *scripts* de gerenciamento (tanto pode ser um único *script* sob responsabilidade do gerente MLM como podem ser vários) deverão ser informados os parâmetros que serão utilizados para a execução desses *scripts.* Alguns exemplos de argumentos são IP de agentes monitorados e estações controladas por este gerente, como visto anteriormente. Os *scripts* possíveis de serem executados após repassados aos gerentes intermediários foram pré-determinados e serão apresentados mais adiante neste

trabalho juntamente com uma descrição sobre ele, os parâmetros necessários para a execução de cada um deles e uma idéia do que é possível fazer com cada um dos *scripts.*

2) Gerentes intermediários (MLM): cada gerente intermediário pode ser responsável pela monitoração de um ou mais agentes. Normalmente se trabalha de forma a agrupar agentes por subredes ou pela localização física mais próxima (por sala, andar de prédios ou cidades, por exemplo). Esses gerentes, ao receberem comandos e instruções sobre os *scripts* que terão que executar, armazenam o *script* localmente e, em geral, em memória não volátil. Esses *scripts* ficarão armazenados até o momento que esse gerente MLM receba a instrução de que este *script* especificamente deve ser executado. Cada novo *script* cadastrado recebe uma identificação única que identifica o *script* para ser iniciado, alterado os parâmetros, verificar se o *script* está disponível e apto para ser executado, etc. A instrução de que o *script* deve ser executado, será enviado pelo TLM para o MLM. Especificamente para esta aplicação, por ser baseada e utilizar conceitos da MIB Script, para iniciar um *script* armazenado no gerente MLM, para que este *script*, após iniciado, também seja identificado por um identificador único, o gerente TLM deve consultar o valor de um objeto SNMP presente no gerente MLM e com base no resultado dessa consulta terá o valor do identificador que deve ser utilizado para iniciar a execução do novo *script* e, posteriormente, identificar o *script* em execução, seu status e seus possíveis resultados intermediários.

5.3.1 Desenvolvimento de *scripts* de gerenciamento

A seguir são sugeridos cinco *scripts* para gerenciamento dos no-breaks.

1) *script* para monitorar falta de rede
2) *script* para monitorar autonomia
3) *script* para monitorar bateria crítica
4) *script* para monitorar sobrecarga
5) *script* para monitorar tensão de entrada

Dois deles monitoram os objetos considerados os mais importantes no gerenciamento de no-breaks (falta de rede e bateria crítica). Outro *script*, que monitora a autonomia, é definido para utilizar o procedimento definido como desligamento progressivo, explicado a seguir. Já os *scripts* para monitorar sobrecarga e tensão de entrada foram definidos principalmente para apresentar a idéia, ou seja, outras análises que se pode fazer a partir de dados coletados do no-break.

Foram implementados três desses *scripts* (falta de rede, autonomia e bateria crítica) e sugeridos, com possível forma de funcionamento, os demais.

***Script* para monitorar falta de rede**: recebe como parâmetros quatro campos, sendo eles: 1) IP do agente a ser monitorado; 2) comunidade para leitura de informações do agente; 3) intervalo (em segundos) entre duas consultas SNMP; 4) nome de arquivo texto no qual estão registrados, em estrutura pré-definida (conforme Figura 5.5), IPs de estações, comunidades para interagir com as estações e tempo para desligamento dessas.

Esse *script* monitora o objeto *upsAlarmOnBattery*. Se verificado que o valor do objeto é 0 (zero), significa que a rede está presente. Mas se uma consulta retornar o valor 1 (um) será feito nova consulta a esse mesmo objeto para confirmar a ausência da rede. Ao detectar que não está sendo fornecida energia na entrada do no-break, inicia

automaticamente os *scripts* para monitorar autonomia e para monitorar bateria crítica. Do mesmo modo, ao identificar o retorno da rede, aborta os *scripts* iniciados por ele.

Exemplo de estrutura:

```
{5:IP1,comunidade1;IP2,comunidade2;IP3,comunidade3;...;IPn,comunidaden}
{30:IP1,comunidade1;IP2,comunidade2;...;IPn,comunidaden}
```

A primeira linha indica para desligar os micros após 5 minutos sem rede

A segunda linha indica para desligar os micros após 30 minutos sem rede

Obs.: pode existir um *script* igual a esse mas com interpretação diferente para o primeiro valor da estrutura. Para o novo *script*, ao invés de interpretar o valor como tempo em minutos, será interpretado como autonomia em porcentagem (%), ou seja, os micros serão desligados quando autonomia for 5%, 30%, etc.

Figura 5.5: Exemplo de estrutura especificada no arquivo parâmetro para o *script*

Com o desligamento progressivo é preservada carga das baterias para estações de maior prioridade. A seguir são apresentados dois exemplos que mostram situações reais em que é importante para a empresa afetada pela falta de energia o desligamento progressivo. Nos dois exemplos, o grande objetivo é sempre o contínuo atendimento ao cliente.

Exemplo 1: empresa de tele marketing ou tele atendimento: desliga-se primeiramente as estações utilizadas para funções administrativas, mantendo por mais tempo estações utilizadas pelos atendentes, que interagem diretamente com os clientes.

Exemplo 2: uma agência bancária: desligam estações de uso interno mantendo operacionais os caixas, terminais de auto-atendimento, sistema de segurança e sistema de comunicação (transmissão de dados). Em algumas situações pode-se também desligar parcialmente a rede de atendimento mantendo o fornecimento do serviço mesmo que em menor escala (com maior tempo de espera).

Dados estatísticos sobre falta de energia podem ser utilizados para calcular e definir os tempos para desligamento das estações. A Aneel (ANEEL, 2006) utiliza dois principais indicadores para verificar a qualidade dos serviços oferecidos pelas concessionárias de energia e eles podem ser utilizados para essa definição. Os indicadores são DEC (Duração Equivalente de Interrupção por Unidade Consumidora) que indica o número de horas em média que um consumidor fica sem energia elétrica durante um período, e FEC (Freqüência Equivalente de Interrupção por Unidade Consumidora) que indica quantas vezes, em média, houve interrupção na unidade consumidora (residência, comércio, indústria etc) (ANEEL, 2006b). Esses dados podem ser consultados no site da Aneel por estado, região, concessionária, etc. A razão entre DEC e FEC indica em média o tempo para retornar um fornecimento de energia. A média brasileira deste tempo, no ano de 2005, foi de 1 hora e 20 minutos (ANEEL, 2006c).

***Script* para monitorar autonomia**: esse *script* é iniciado pelo *script* que monitora a falta de rede e pode ser finalizado em duas situações: ou o *script* conclui sua execução, enviando comando de desligamento para cada estação que foi configurada e é concluído

indicando esse resultado, ou o *script* é encerrado pelo mesmo *script* que o chamou, ou seja, o *script* que identificou a ausência de rede e solicitou a monitoração da autonomia (ao perceber que a energia da rede voltou, encerra o *script* que monitora a autonomia).

Recebe como parâmetros três campos: 1) IP do agente a ser monitorado; 2) comunidade para leitura de informações do agente; 3) nome de arquivo texto no qual estão registrados, em estrutura pré-definida, IPs de estações, comunidades para interagir com as estações e tempo para desligamento dessas. O último parâmetro é o mesmo arquivo informado ao *script* que monitora falta de rede, ou seja, o *script* recebe esse arquivo como parâmetro e quando for necessário iniciar os outros *scripts* passa como parâmetro o mesmo arquivo.

Esse *script* faz a leitura do objeto *upsSecondsOnBattery* a cada 30 segundos ou no momento de realizar um desligamento de estações. Se verificado que o valor do objeto (valor lido dividido por 60, para indicar quantidade de minutos) é uma unidade inferior ao valor indicado na estrutura como o momento de desligar estações, passa a fazer a leitura a cada 10 segundos.

Após a confirmação de que o no-break já ficou sem rede durante o tempo informado na estrutura parâmetro, o *script* poderia disparar rotinas para que sejam iniciados os processos de desligamento das máquinas solicitadas.

***Script* para monitorar bateria crítica**: esse *script* é iniciado pelo *script* que monitora a falta de rede e pode ser finalizado em duas situações: ou o *script* conclui sua execução, enviando comando de desligamento para cada estação que foi configurada e é concluído indicando esse resultado, ou o *script* é encerrado pelo mesmo *script* que o chamou, ou seja, o *script* que identificou a ausência de rede e solicitou a monitoração do estado da bateria (ao perceber que a energia da rede voltou, encerra o *script* que monitorava o estado da bateria).

Recebe como parâmetros três campos: 1) IP do agente a ser monitorado; 2) comunidade para leitura de informações do agente; 3) nome de arquivo texto no qual estão registrados, em estrutura pré-definida, IPs de estações, comunidades para interagir com as estações e tempo para desligamento dessas. O último parâmetro é o mesmo arquivo informado ao *script* que monitora falta de rede, ou seja, o *script* recebe esse arquivo como parâmetro e quando for necessário iniciar os outros *scripts* passa como parâmetro o mesmo arquivo.

Esse *script* faz a leitura do objeto *upsAlarmLowBattery* a cada 10 segundos. Se verificado que o valor do objeto é 0 (zero), significa que a bateria ainda possui carga e é capaz de continuar alimentando a carga. Mas se uma consulta retornar o valor 1 (um) será feito imediata e consecutivamente duas novas consultas a esse mesmo objeto para se assegurar que o objeto realmente está informando que o estado da bateria é crítica.

Após a confirmação de que o estado das baterias é crítico, o *script* poderia disparar automática e imediatamente rotinas para que sejam iniciados os processos de desligamento das máquinas cujo IP foi informado nos parâmetros do *script*.

***Script* para monitorar sobrecarga**: recebe como parâmetros três campos: 1) IP do agente a ser monitorado; 2) comunidade SNMP para leitura de informações do agente; 3) nome de arquivo texto no qual estão registrados, em estrutura pré-definida, IPs de estações e comunidades para interagir com as estações.

A estrutura do arquivo texto contendo IPs para serem desligados por esse *script* pode ser parecida com a estrutura utilizada anteriormente (Figura 5.5), sem a informação de autonomia, e apresentando as estações na ordem inversa de prioridade, ou seja, o IP que estiver listado primeiro na lista é a estação de menor prioridade e por isso é a primeira a ser desligada.

Ao identificar o alarme de sobrecarga ativado, aguarda três segundos e faz nova consulta para confirmar se o alarme continua ativo. Essa consulta de confirmação é realizada para que não faça o desligamento em casos de pico de carga, situação que acontece quando os equipamentos são ligados. De qualquer forma, toda ocorrência de sobrecarga deve ser registrada pois mesmo que seja apenas uma sobrecarga temporária, se isso acontecer com muita freqüência a situação da alimentação da rede deve ser reavaliada.

***Script* para monitorar tensão de entrada**: recebe como parâmetros 3 (três) campos: 1) IP do agente a ser monitorado; 2) comunidade SNMP para leitura de informações do agente; 3) intervalo (em segundos) entre duas consultas SNMP.

O *script* faz a leitura do objeto *upsInputNumLines* para descobrir quantas fases tem o equipamento e posteriormente realiza a consulta ao objeto *upsInputVoltage* para obter a tensão em cada uma das fases do equipamento. Os valores lidos são armazenados em base de dados para manter histórico de variação da grandeza.

5.3.2 Desenvolvimento da ferramenta de gerenciamento distribuído

A aplicação desenvolvida basicamente fornece apenas a interface para que o gerente principal seja capaz de realizar as tarefas de associação dos *scripts* disponibilizados para uso do sistema de gerenciamento com os gerentes intermediários responsáveis pelas execuções destes, agentes monitorados e *hosts* da rede que serão manipulados quando o resultado da execução de um *script* assim julgar necessário.

A seguir são apresentadas as etapas utilizadas nessas associações:

Etapa 1: o administrador informa o nome ou IP do gerente intermediário para o qual deseja delegar tarefas, ou seja, repassar os *scripts* que serão executados. Além do IP, são solicitadas 3 (três) strings de comunidade (Figura 5.6). Assim que esses dados forem informados, a aplicação estabelece contato com o gerente intermediário para verificar se ele possui a MIB Script sendo executada pois este é o requisito para que um host possa ser aceito como um gerente intermediário. As três strings de comunidades solicitadas serão utilizadas respectivamente para: 1) leitura de informações do agente que implementa MIB Script; 2) escrita de informações no agente; 3) definir e executar *scripts.*

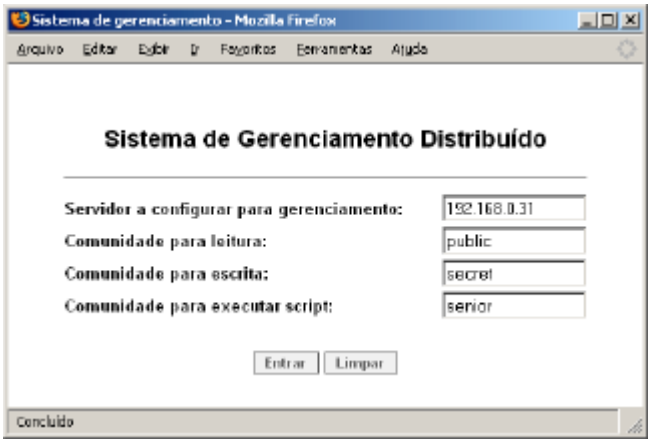


Figura 5.6: Tela inicial do sistema – solicita IP do MLM e 3 strings de comunidade

Após informar esses dados e selecionar o botão “Entrar” (entrar no ambiente que gerencia um MLM), será apresentado o menu principal do sistema (Figura 5.7). É através deste menu que todo o processo de delegação de tarefas para o gerente MLM informado na tela anterior (Figura 5.6) será realizado.

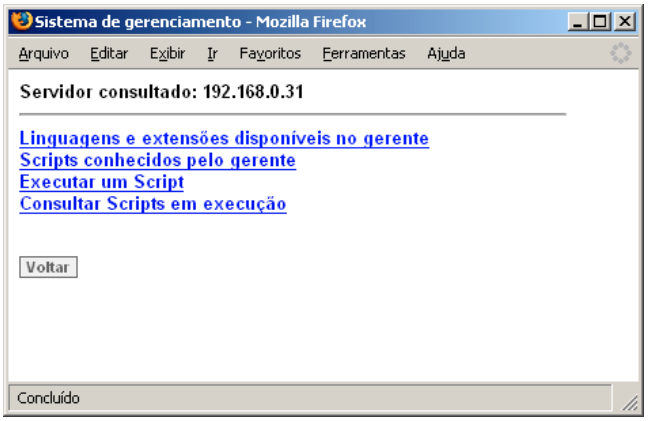


Figura 5.7: Menu principal do sistema

No menu principal é apresentado nome ou IP do gerente intermediário que está sendo configurado e as opções “Linguagens e extensões disponíveis no gerente”, “Scripts conhecidos pelo gerente”, “Executar um Script” e “Consultar Scripts em execução”. Cada uma dessas opções será detalhada a seguir, revisando os passos necessários para a implantação do gerenciamento no gerente intermediário.

A Figura 5.8 mostra um exemplo de tela para quando se escolhe a opção “Linguagens e extensões disponíveis no gerente” no menu principal. As opções ali apresentadas são as linguagens e extensões de linguagens já suportadas pela implementação do Jasmin em sua instalação padrão.

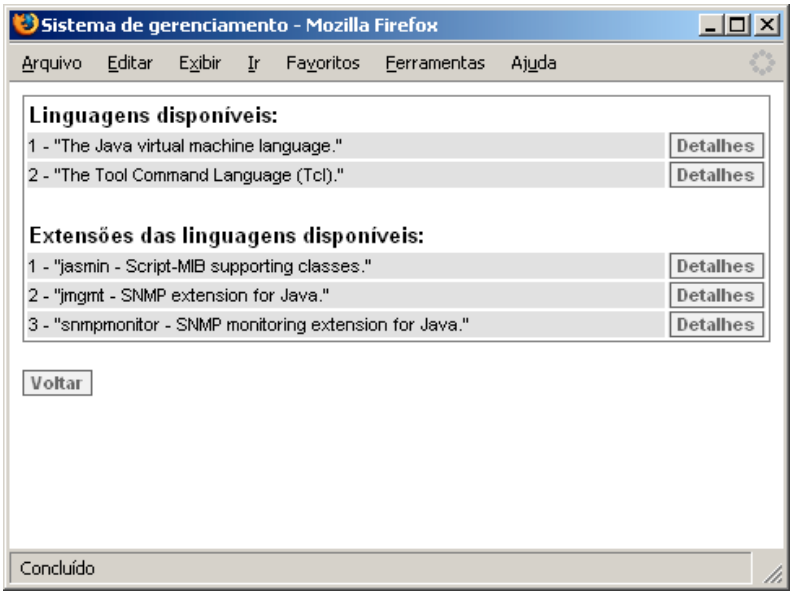


Figura 5.8: Linguagens e extensões instaladas no agente da MIB Script

Etapa 2: o gerente informa quais são os *scripts* que o gerente intermediário deverá utilizar durante seu processo de gerenciamento. Para isso, deverá escolher no menu principal a opção "Scripts conhecidos pelo gerente". Ao selecionar essa opção será apresentada uma tela com todos os *scripts* que já foram disponibilizados ao gerente MLM (Figura 5.9).

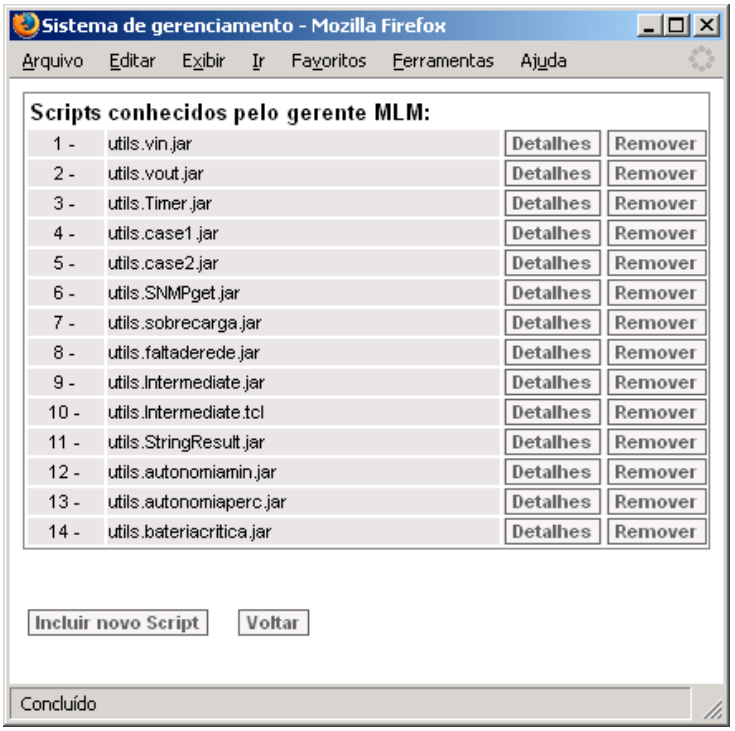


Figura 5.9: *Scripts* conhecidos pelo gerente MLM

Nesta tela, o administrador poderá consultar mais detalhes a respeito do *script* (botão "Detalhes"), apagar um *script* fazendo com que ele seja removido do gerente MLM (botão "Remover") ou ainda incluir novos *scripts* (botão "Incluir novo script"). Ao escolher a opção para incluir novo *script*, será apresentada tela conforme mostrado na

Figura 5.10 e além da *string* identificando a permissão para criar o *script* e a própria identificação (nome) do *script*, deverão ser fornecidas as seguintes informações:

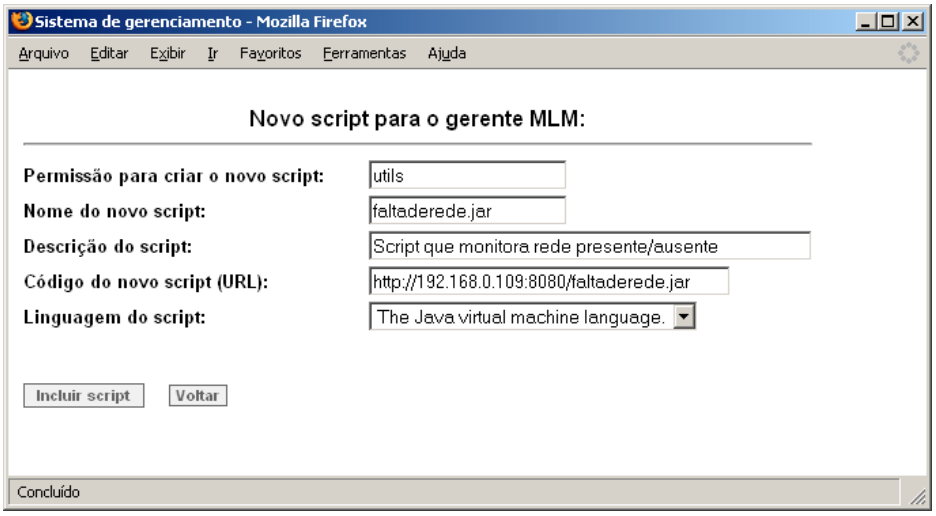


Figura 5.10: Tela de inclusão de novo *script* no gerente MLM

Descrição do script: essa descrição não é utilizada pelo sistema. Ela possui utilidade apenas para o que o administrador do sistema de gerência consiga facilmente obter mais detalhes a respeito do *script* do que simplesmente seu nome. É sugerido que este campo descreva brevemente o uso do *script*, seus possíveis parâmetros e resultado(s) retornado(s).

Código do script (URL): é a localização do *script*, ou seja, URL que o gerente intermediário utilizará para copiar o *script* em questão.

A tarefa de transferir os *scripts* desenvolvidos para um servidor e deixá-los disponíveis para que os gerentes intermediários tenham condições de copiá-los fica sob responsabilidade do administrador. Não foi criada interface para administrar essa tarefa pois ela é considerada trivial e de fácil administração.

Linguagem do script: o administrador escolhe qual linguagem, suportada pela MIB Script, deverá ser utilizada para a execução do *script*. As linguagens apresentadas para que se faça a escolha/seleção são consultadas na tabela *smLangTable* do gerente MLM em questão.

Para cada *script* que o administrador deseja disponibilizar ao gerente intermediário, deve fazer um novo cadastro (Figura 5.10) e incluir as informações acima.

Depois de informar todos esses dados, o gerente deve clicar no botão “Incluir script” para que os dados sejam transferidos ao gerente intermediário. Ao receber os dados o gerente intermediário armazena o *script* localmente e tenta disponibilizá-lo para uso. A aplicação desenvolvida monitora a ação do gerente intermediário e apresentará na tela se o *script* foi criado corretamente e disponibilizado para uso.

Etapa 3: o administrador deverá criar o chamado “ambiente para execução do *script*”, ou seja, para cada *script* que foi disponibilizado na etapa anterior poderá criar um ou mais botões de execução para cada *script*. É provável que um *script* seja executado mais de uma vez com “ambientes diferentes” (executa-se o mesmo *script* mas ocorre variação nos parâmetros recebidos, por exemplo).

Para criar os ambientes de execução, o administrador deverá escolher, no menu principal, a opção “Executar um Script”. Ao selecionar essa opção será apresentada uma tela com todos os ambientes (botões de execução) que foram criados neste gerente e estão disponíveis para execução para um determinado grupo de controle de acesso (Figura 5.11). Esse grupo que estabelece o controle de acesso aos *scripts* pode ser alterado no campo “Listados scripts para execução por”. Nele pode-se escrever o grupo desejado e em seguida clicar no botão “Nova leitura” para que a tabela seja lida novamente (consultado agente MIB Script), apresentando os dados para o novo grupo escolhido.

Figura 5.11: *Scripts* disponíveis para execução no gerente MLM

Nessa tela da Figura 5.11 o administrador pode ainda obter mais detalhes a respeito de um botão de execução – clicando em “Detalhes”, apagar um botão fazendo com que ele não fique mais disponível para execução – selecionando o botão “Remover”, iniciar a execução de um *script* – clicando no botão “Launch”, ou ainda incluir novas linhas na tabela que significa criar novos botões de execução – selecionando o botão “Incluir novo botão de execução de Script”. Ao escolher a opção para incluir novo botão, será apresentada tela conforme mostrado na Figura 5.12 e deverão ser fornecidas as seguintes informações:

Dono do script: deve-se informar essa *string* para identificar se o administrador possui permissão para utilizar o *script*.

Nome do script: identificação do *script* a ser executado. Essa identificação deve ser igual ao nome do *script* que foi incluído na tabela de *scripts* (*smScriptTable*). Para facilitar o preenchimento a aplicação já busca na tabela de *scripts* os nomes possíveis e disponibiliza para escolha do administrador como pode ser visto na Figura 5.13.

Parâmetro(s): nesse campo deve-se colocar o parâmetro que será passado ao *script* quando da sua execução. Havendo mais de um parâmetro eles devem ser separados por um espaço em branco.

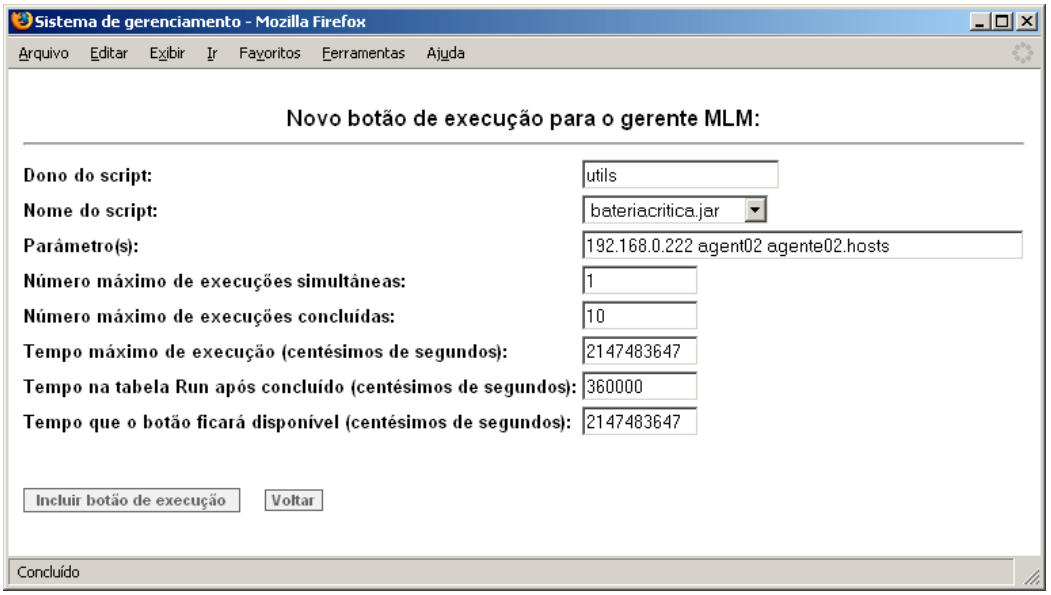


Figura 5.12: Tela de inclusão de novo botão de execução

Número máximo de execuções simultâneas: informar a quantidade máxima que será aceita de execuções simultâneas deste *script* com esta definição de ambiente de execução.

Número máximo de execuções concluídas: define a quantidade máxima de resultados que serão mantidos na tabela *smRunTable*. Esse valor diz respeito especificamente a quantidade de entradas referente ao botão de execução que está sendo configurado.

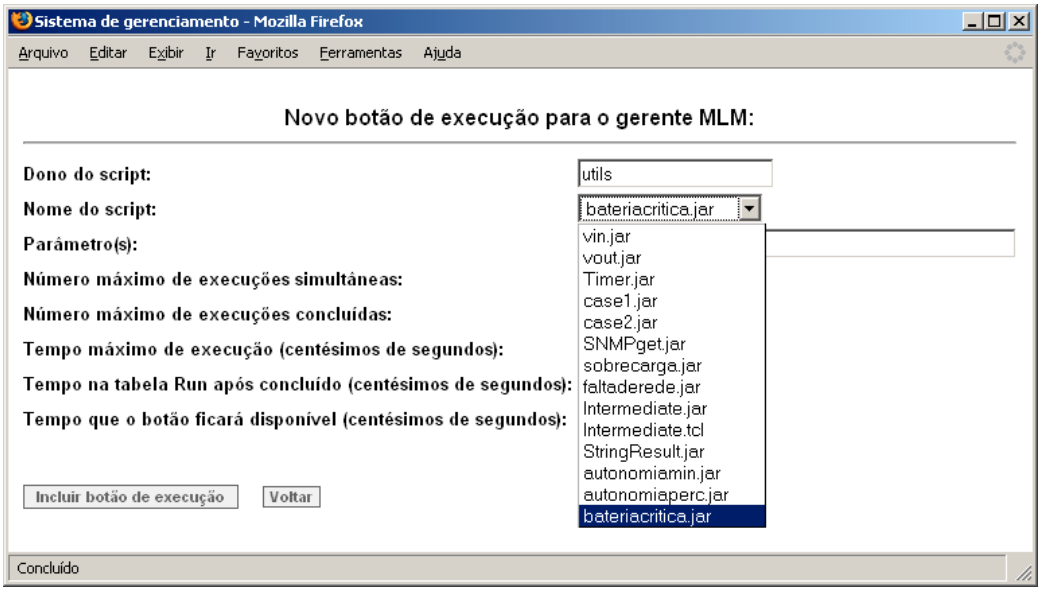


Figura 5.13: Nomes de *scripts* possíveis são preenchidos automaticamente

Tempo máximo de execução: é o tempo máximo (em centésimos de segundos) que cada execução poderá ficar ativa/rodando. Quando é acionado um botão de execução de *script*, ele é iniciado e executa de forma totalmente independente de qualquer outro *script*, ou seja, não há relação de dependência entre dois *scripts*. Sendo assim, esse tempo que é especificado para o *script* ser executado será dado para cada instância do

ambiente que for iniciada. Se o *script* não terminar sua execução normalmente antes do tempo determinado ele será abortado pelo sistema e o código de erro sinalizando que seu tempo para execução esgotou será indicado. Para que um *script* possa ser executado por tempo indeterminado sem ser abortado pelo sistema, esse campo deve ser preenchido com o valor máximo possível definido na MIB Script que é de 2147483647.

Tempo na tabela Run após concluído: tempo máximo (em centésimos de segundos) que cada ambiente, depois de concluída a execução, permanece na tabela *smRunTable* para que seja possível consultar o resultado e outros objetos a respeito da execução.

Tempo que o botão ficará disponível: tempo (em centésimos de segundos) que este botão ficará disponível para ser iniciado um ambiente de execução. Para que o botão não seja removido automaticamente, permanecendo disponível até que o administrador o elimine manualmente, esse campo deve ser preenchido com o valor máximo possível definido na MIB Script que é de 2147483647.

Para cada botão de execução de *script* que o administrador deseja criar no gerente intermediário, deve preencher novo formulário (Figura 5.12) com as informações acima.

Depois de informar todos esses dados, o gerente deve clicar no botão "Incluir botão de execução" para que os dados sejam transferidos ao gerente intermediário. Ao receber os dados o gerente intermediário tenta disponibilizar, com essas informações, um novo botão para iniciar uma execução. A aplicação desenvolvida monitora a ação do gerente intermediário e apresentará na tela se o botão foi criado corretamente e disponibilizado para uso. Se o botão foi criado corretamente e pode ser utilizado, quando a aplicação informar isso disponibilizará na mesma tela a opção de iniciar a execução do *script*.

Etapa 4: o administrador inicia e monitora a execução dos *scripts* desejados. Além de poder iniciar a execução do *script* no momento em que ele é disponibilizado como visto acima, vale lembrar que isso também é possível de ser feito escolhendo, no menu principal, a opção "Executar um Script" e na tela seguinte (Figura 5.11) iniciar a execução do(s) *script(s)* desejado(s).

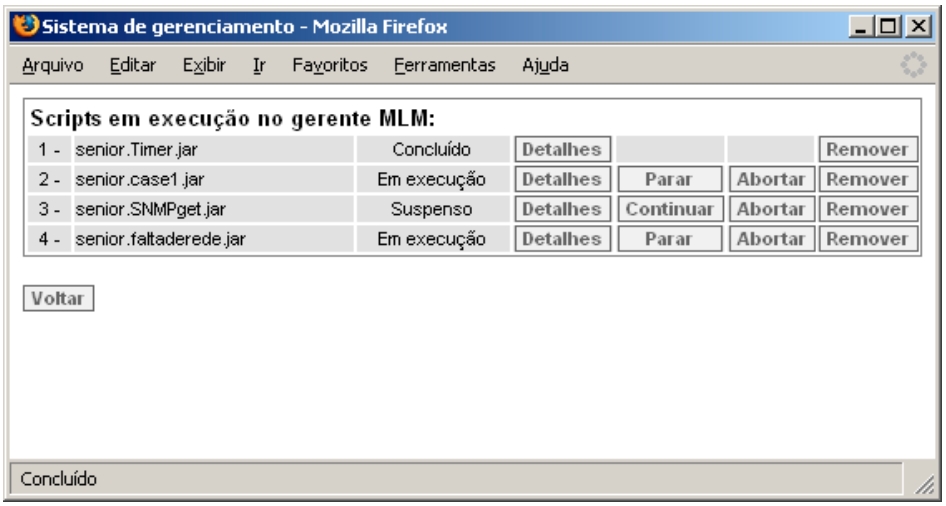


Figura 5.14: *Scripts* em execução no gerente MLM

Cada *script* em execução ou que tenha concluído sua execução recentemente é uma linha da tabela *smRunTable* da MIB Script e eles podem ser consultados e monitorados através da opção "Consultar Scripts em execução" do menu principal do sistema.

Escolhendo essa opção, a tela seguinte (Figura 5.14), apresenta uma linha para cada *script*, indicando seu estado (concluído, suspenso ou em execução) e disponibilizando até quatro botões para interagir com cada um dos *scripts.*

O primeiro botão (“Detalhes”) apresentará todas as informações disponíveis a respeito daquela execução. Quando o *script* estiver concluído, é consultando os detalhes do *script* que será possível verificar o resultado. A Figura 5.15 apresenta os detalhes de um *script* em execução.

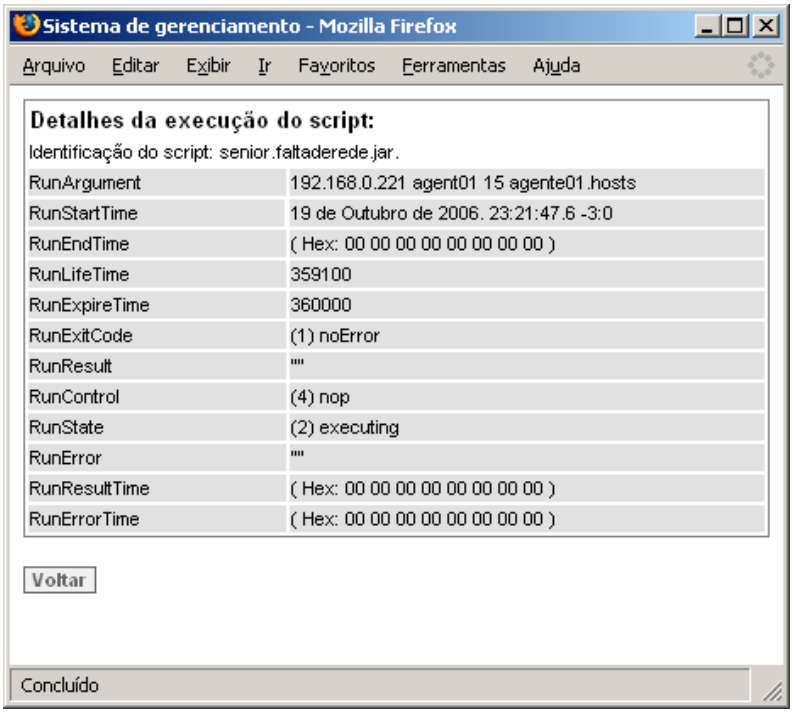


Figura 5.15: Detalhes da execução de um *script*

Os outros botões que podem aparecer ao lado do *script* em execução são “Parar” (interrompe a execução), “Continuar” (continua a execução do *script* interrompido anteriormente), “Abortar” (cancela a execução) e “Remover” (cancela o *script* caso não esteja concluído e elimina a linha da tabela).

5.3.3 Ferramentas utilizadas

Para a implementação do gerente principal (TLM) foi desenvolvida uma aplicação Web capaz de interagir com outros gerentes ou com agentes se isso fosse necessário. As ferramentas utilizadas nessa etapa do desenvolvimento foram um servidor Web e uma linguagem de programação voltada para aplicações Web e com suporte ao protocolo SNMP. Optou-se por utilizar o Apache (APACHE, 2006) como aplicação do servidor Web e a linguagem PHP (PHP, 2006) para o desenvolvimento da aplicação. Diversos aspectos influenciaram para que essas fossem as ferramentas escolhidas para uso nesta implementação. O principal motivo é o dinamismo, ou seja, o PHP permite um desenvolvimento rápido – comparado com a maioria das outras linguagens – e possui boa quantidade de recursos para esse tipo de aplicação. Outro motivo que influenciou para o uso dessas ferramentas é que são ferramentas livres, que podem ser utilizadas sem restrições tanto em ambiente Linux quanto em ambiente Windows. O fato de já ter utilizado essas ferramentas no passado – mesmo que por curto período de tempo e para

aplicações pequenas – também influenciou na tomada de decisão pelo conjunto Apache + PHP.

5.4 Manipulando recursos da rede

Este trabalho avaliou a *Host Resources MIB* (ou apenas *Host MIB*) para, através dela, obter informações a respeito das estações da rede incluindo a possibilidade de desligar as estações quando necessário.

A MIB para gerenciamento de recursos dos hosts possui 5 grupos de informações: Recursos de sistema, Recursos de armazenamento, Recursos de dispositivos, Recursos de software que estão em execução e Recursos de performance de software que estão em execução. A *Host MIB*, por padrão, já está implementada nos agentes SNMP dos sistemas operacionais Windows 2000, XP e 2003. Existem ainda implementações em outros sistemas como no agente SNMP do Digital UNIX (DIGITAL, 2006) além da implementação para o *toolkit* Net-SNMP (Net-SNMP, 2006).

Mas as implementações da *Host MIB* apenas fornecem informações a respeito do *host* e não permitem a interação com o dispositivo em questão. Apesar de existir um objeto definido nessa MIB para se alterar o estado de um processo sendo executado, em implementações que foram testadas o recurso não funciona (ou, em alguns casos, propositadamente não é disponibilizado ao usuário, conforme pode ser visto em (HP, 2006), retornando erro quando se tenta parar um processo e todas as informações encontradas a respeito deste objeto em outras implementações afirmam que ele não é aceito. A intenção era, em momentos críticos nos quais fosse realmente necessário, alterar o estado do processo principal do sistema, forçando o seu desligamento.

Embora nenhuma solução tenha sido colocada em prática, deixando essa lacuna na implantação e passível de se buscar uma solução de acordo com o caso ou o no-break gerenciado, outras alternativas foram avaliadas para substituir o uso da *Host MIB*. Foram revistos alguns dos programas com esse objetivo apresentados em (POLINA, 2005) para verificar se algum deles atendia o requisito desejado sem que fosse necessário o agente SNMP do proprietário do software, o que não aconteceu. Também foi avaliado o agente NetPower sugerido por Pollo em (POLLO, 2002). Ele sugere um agente experimental capaz de desligar os equipamentos total ou parcialmente (apenas alguns periféricos) de acordo com a necessidade e se o equipamento é capaz de realizar tais operações.

6 CONCLUSÃO

O trabalho desenvolvido teve a intenção de apresentar uma solução de gerenciamento tendo os no-breaks como os agentes SNMP desse gerenciamento. Soluções de gerenciamento já são comuns de diversos fabricantes mas o gerenciamento oferecido sempre seguia a linha de gerenciamento centralizado e o administrador da rede poderia também utilizá-lo como distribuído independente (diversos gerentes espalhados pela rede mas sem nenhuma comunicação entre os gerentes). A solução proposta pelo presente trabalho utiliza o paradigma de gerenciamento distribuído por delegação, no qual um gerente principal controla e delega tarefas para os outros gerentes da rede e os gerentes que recebem as tarefas continuam sendo independentes, ou seja, não há exigência de comunicação com o gerente principal para realização de suas funções.

Apesar de ter um custo maior com a necessidade de investir em diversos equipamentos que serão utilizados no gerenciamento, esse investimento se justifica em grandes redes pois diminui ou elimina os problemas que existiriam com a utilização do gerenciamento centralizado.

O sistema desenvolvido e as ferramentas utilizadas são de grande importância principalmente para o meio acadêmico, ou seja, para o estudo das funções e forma de operação do gerenciamento distribuído (MIB Script, especificamente). Comercialmente a aplicação desenvolvida é viável mas as ferramentas utilizadas deixam a desejar. Para que o projeto fosse aceito comercialmente seria necessário atualizar a ferramenta Jasmin para torná-la compatível com sistemas operacionais atuais ou desenvolver nova ferramenta que implementa a MIB Script. Isso porque dificilmente uma empresa aceitaria ter em seu sistema de gerenciamento uma estação rodando sistema operacional que não possui atualizações há mais de um ano – principalmente pela questão da segurança e inviabilidade para suporte técnico.

Verificou-se ainda, com base em análise prévia que realizou a investigação sobre recursos monitoráveis e gerenciáveis de diferentes UPSs, levando em conta também a forma como eles permitem a gerência e uma possível configuração do equipamento, que muitas vezes o mais importante é a forma como as informações coletadas são utilizadas e interpretadas no momento do gerenciamento. Isso significa que o sistema de gerenciamento deve ser bem estruturado e configurado para tomar atitudes corretas em momentos críticos.

Também pôde-se verificar que a pesquisa não está esgotada e o assunto permite uma análise mais aprofundada e ampla, como por exemplo com os itens apresentados a seguir.

Atualização da ferramenta Jasmin ou desenvolvimento de nova ferramenta que implementa a MIB Script. A primeira tarefa é mais fácil pois pode-se trabalhar na versão já existente (Jasmin 1.0) mas a segunda também é bem interessante, principalmente se a nova implementação contemplar também o uso de outros sistemas operacionais.

Desenvolver novos *scripts* que podem ser utilizados pelo sistema de gerenciamento. Os novos *scripts* podem monitorar recursos como temperatura e necessidade de trocar o banco de baterias ou até mesmo desligar o próprio no-break quando não houver mais carga sendo alimentada por ele. Melhorar os *scripts* já existentes, deixando-os mais flexíveis, também pode ser uma alternativa interessante.

Outro recurso interessante para o desenvolvimento de novos *scripts* é prever a monitoração de outros equipamentos como estabilizadores, geradores, analisadores de baterias e até de no-breaks em paralelo (recurso bastante difundido atualmente). No-breaks em paralelo podem, sem nenhum problema, ser monitorados pelos *scripts* apresentados neste trabalho, mas com o desenvolvimento de *scripts* específicos para esse uso o controle sobre o conjunto de no-breaks paralelos pode oferecer maiores benefícios.

Estudar e aperfeiçoar a integração do sistema de gerenciamento com uma aplicação para controle total das estações da rede e que esta aplicação seja, de preferência, um agente SNMP. O ponto de partida pode ser o agente sugerido por Luis Fernando Pollo em (POLLO, 2002).

Criar interface amigável para automatizar ainda mais a tarefa do administrador, coletando os resultados dos *scripts* automaticamente e definindo, com estes e outros dados uma excelente apresentação de dados da rede analisada. Essa interface pode ser tanto no TLM quanto nos MLMs.

REFERÊNCIAS

ADVENTNET. Disponível em: <http://www.adventnet.com/>. Acesso em: jun. 2006.

ANEEL: Agência Nacional de Energia Elétrica. Disponível em: <http://www.aneel.gov.br/>. Acesso em: set. 2006.

ANEEL. **Qualidade do fornecimento de energia melhorou.** Disponível em: <http://www.aneel.gov.br/aplicacoes/noticias/Output_Noticias.cfm?Identidade=909>. Acesso em: set. 2006.

ANEEL. **Qualidade do Serviço**. Disponível em: <http://www.aneel.gov.br/79.htm>. Acesso em: set. 2006.

APACHE Software Foundation. Disponível em: <http://www.apache.org/>. Acesso em: jun. 2006.

BEN-ARTZI, A.; CHADNA, A.; WARRIER, U. Network Management of TCP/IP Networks: present and future. **IEEE Network Magazine**, New York, NY, USA, v.4, n.4, p.35–43, July 1990.

CASE, J. **UPS Management Information Base:** RFC 1628 [S.l.]: Internet Engineering Task Force, 1994.

CASE, J. et al. **A Simple Network Management Protocol (SNMP)**: RFC 1157. [S.l.]: IETF, Network Working Group, 1999.

CP ELETRÔNICA. **Nobreaks - Topologias Principais.** Disponível em: <http://www.cp.com.br/artigos/TextoUps.pdf>. Acesso em: jun. 2006.

DEBIAN. Disponível em: <http://www.debian.org/>. Acesso em: set. 2005.

DMTF **WBEM, Web-Based Enterprise Management, Web Management, xml Management Standards**. Disponível em: <http://www.dmtf.org/standards/standard_wbem.php>. Acesso em: jul. 2006.

DIGITAL. **Host Resources MIB Implementation.** Disponível em: <http://www.cs.arizona.edu/computer.help/policy/DIGITAL_unix/AA-PS2SC-TE_html/netadmin27.html#host_mib_app>. Acesso em: mar. 2006.

GOLDSZMIDT, G.; YEMINI, Y. Distributed Management by Delegation. In: IFIP/IEEE INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 1995. **Proceedings. . .** [S.l.]: IEEE Computer Society, 1995. p.333 – 340.

GRILLO, P.; WALDBUSSER, S. **Host Resources MIB:** RFC 2790. [S.l.]: IETF, Network Working Group, 2000.

HP OpenVMS Systems Documentation. Disponível em: <http://h71000.www7.hp.com/DOC/82final/6530/6530pro_001.html>. Acesso em: set. 2006.

IETF. **Distributed Management (disman) Charter**. Disponível em: <http://www.ietf.org/html.charters/disman-charter.html>. Acesso em: mar. 2005.

IETF: Internet Engineering Task Force. Disponível em: <http://www.ietf.org>. Acesso em: maio 2006.

INTERNET ASSIGNED NUMBERS AUTHORITY. **Private Enterprise Numbers**. Disponível em: <http://www.iana.org/assignments/enterprise-numbers>. Acesso em: jun. 2006.

INTERNET ASSIGNED NUMBERS AUTHORITY. **IANA-LANGUAGE-MIB**. Disponível em: <http://www.iana.org/assignments/ianalanguage-mib>. Acesso em: jul. 2006.

Jasmin - a Script-MIB Implementation. [S.l.]: Technical University of Braunschweig and NEC C&C Research Laboratories, 2000. Disponível em: <http://www.ibr.cs.tu-bs.de/projects/jasmin/>. Acesso em: mar. 2005.

KAHANI, M.; BEADLE, H. Decentralized Approaches for Network Management. **Computer Communications Review**, New York, NY, USA, v.27, n.3, p.36–47, 1997.

LEINWAND, A.; CONROY, K. F. **Network Management:** Practical Perspective. 2nd ed. Boston: Addison Wesley, 1996.

LEVI, D.; SCHÖNWÄLDER, J. **Definitions of Managed Objects for the Delegation of Management Scripts**: RFC 3165. [S.l.]: Internet Engineering Task Force, Network Working Group, 2001.

MANDRIVA. Disponível em: <http://www.mandriva.com/>. Acesso em: fev. 2006.

MARTIN-FLATIN, J. **Web-Based Management of IP Networks and Systems**. Hobo-ken, New Jersey: John Wiley & Sons, 2002. v.1.

MARTINS, A. S.; GABIATTI, G.; BONAN, G. **Entendendo o Fator de Potência**. Disponível em: <http://www.cp.com.br/artigos/fator_de_potencia.pdf>. Acesso em: ago. 2006.

MAURO, D. R.; SCHMIDT, K. J. **Essential SNMP**. 2nd ed. [S.l.]: O'Reilly Media, 2005.

MCCLOGHRIE, K.; ROSE, M. **Management Information Base for Network Management of TCP/IP-based internets:** RFC 1066 [S.l.]: Internet Engineering Task Force, 1988.

MCCLOGHRIE, K.; ROSE, M. **Management Information Base for Network Management of TCP/IP-based internets: MIB-II:** RFC 1213 [S.l.]: Internet Engineering Task Force, 1991.

NEC C&C Research Laboratories. Disponível em: <http://www.ccrle.nec.de/>. Acesso em: maio 2006.

NET-SNMP. Disponível em: <http://www.net-snmp.org/>. Acesso em: ago. 2006.

PHP. Disponível em: <http://www.php.net>. Acesso em: jun. 2006.

POLINA, E.R. **Um estudo sobre gerenciamento de no-breaks**. 2005. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

POLLO, L. F. **Sistema de Gerência de Energia para Redes Locais**. 2002. 144p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

QUITTEK, J. **Jasmin Script Tutorial (Version 1.1)**. 1999. Disponível em: <http://www.ibr.cs.tu-bs.de/projects/jasmin/tutorial.html>. Acesso em: fev. 2006.

QUITTEK, J.; KAPPLER, C. Practical Experiences with Script MIB Applications. **The Simple Times**, [S.l.], v.7, n.2, 1999.

RED HAT. Disponível em: <http://www.redhat.com/>. Acesso em: fev. 2006.

ROSE, M.; MCCLOGHRIE, K. **Structure and Identification of Management Information for TCP/IP-based Internets**: RFC 1155 [S.l.]: Internet Engineering Task Force, 1990.

SCHÖNWÄLDER, J.; QUITTEK, J. **Script MIB Extensibility Protocol Version 1.0.:** RFC 2593. [S.l.]: TU Braunschweig: NEC Europe, 1999.

SCHÖNWÄLDER, J.; QUITTEK, J.; KAPPLER, C. Building Distributed Management Applications with the IETF Script MIB. **IEEE Journal on Selected Areas in Communications**, New York, NY, USA, v.18, n.5, p.702–714, 2000.

SUSE. Disponível em: <http://www.opensuse.org/>. Acesso em: out. 2005.

STALLINGS, W. **SNMP, SNMPv2 and SNMPv3 and RMON 1 and 2**. 3rd ed. Reading: Addison Wesley, 1999.

STRAUSS, F. Script MIB Performance Analysis. **The Simple Times,** [S.l.], v.7, n.2, 1999.

TANENBAUM, A. **Computer Networks.** 4th ed. Upper Saddle River: Prentice Hall, 2003.

TECHNICAL University of Braunschweig. Disponível em: <http://www.ibr.cs.tu-bs.de/>. Acesso em: maio 2006.

VALENTINI, J.; GASPARY, L. P. Monitoração de Disponibilidade e Desempenho de Servidores Críticos usando uma Abordagem Descentralizada. In: ESCOLA REGIONAL DE REDES DE COMPUTADORES, ERRC, 2003, Porto Alegre. **Anais...** [S.l.: s.n.], 2003.

VIANNA, R. L. et al. **An Evaluation of Service Composition Technologies Applied to Network Management**. Submetido e aceito para IFIP/IEEE International Symposium on Integrated Network Management, IM, 2007.

WALDBUSSER, S. **Remote Network Monitoring Management Information Base**: RFC 1757. [S.l.]: IETF, Network Working Group, 1995.

YEMINI, Y.; GOLDSZMIDT, G.; YEMINI, S. Network Management by Delegation. In: INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 1991. **Proceedings...** Amsterdam: North – Holland, 1991. p. 95-107.