

*Ajuda para os Administradores de Sistemas e de Redes*

# SNMP

*Essencial*



O'REILLY®

Douglas R. Mauro & Kevin J. Schmidt

# Sumário

<i>Prefácio</i> .....	IX
<b>1 O que é SNMP? .....</b>	1
Monitoração e gerenciamento de rede .....	1
Versões de RFCs e SNMP .....	4
Gerenciadores e agentes .....	5
Estrutura de informações de gerenciamento e MIBs .....	6
Gerenciamento de hosts .....	7
Introdução sucinta ao Remote Monitoring (RMON) .....	8
Como obter mais informações .....	8
<b>2 Examinando o SNMP .....</b>	10
SNMP e UDP .....	10
Comunidades de SNMP .....	13
Structure of Management Information .....	14
Extensões para a SMI na Versão 2 .....	25
Exame minucioso da MIB-II .....	28
Operações do SNMP .....	29
Considerações adicionais sobre o gerenciamento de hosts .....	42
Considerações adicionais sobre a monitoração do sistema .....	43
<b>3 Arquiteturas de NMS .....</b>	45
Considerações sobre o hardware .....	45
Arquiteturas de NMS .....	47
Um passo à frente .....	51
<b>4 Hardware compatível com o SNMP .....</b>	53
O que realmente significa ser compatível com o SNMP? .....	53
Meu dispositivo é compatível com o SNMP? .....	54
Upgrade de hardware .....	56
Em resumo .....	57
Um passo à frente .....	57
<b>5 Software de gerenciamento de rede .....</b>	59
Agentes de SNMP .....	60
Suítes de NMS .....	62
Gerenciadores de componentes (gerenciamento específico do fornecedor) .....	65
Análise de tendências .....	67
Software de suporte .....	69

<b>6</b>	<b><i>Configurando a NMS</i></b>	<b>72</b>
	Network Node Manager do OpenView, da HP	72
	SNMPc Enterprise Edition da Castle Rock	85
<b>7</b>	<b><i>Configurando agentes do SNMP</i></b>	<b>88</b>
	Configurações de parâmetros	88
	Questões de segurança	90
	Considerações sobre a configuração de agentes	92
<b>8</b>	<b><i>Polling e definição</i></b>	<b>110</b>
	Recuperando um valor de uma MIB individual	110
	Recuperando diversos valores de MIB	117
	Definindo um valor de MIB	120
	Respostas de erro	123
<b>9</b>	<b><i>Polling e limiares</i></b>	<b>124</b>
	Polling interna	126
	Polling externa	133
<b>10</b>	<b><i>Traps</i></b>	<b>153</b>
	Noções básicas sobre traps	153
	Recebendo traps	154
	Exibição das categorias de eventos	162
	Browser de alarme	163
	Criando eventos no OpenView	164
	Enviando traps	169
<b>11</b>	<b><i>Agentes extensíveis de SNMP</i></b>	<b>179</b>
	Net-SNMP	180
	SystemEDGE	185
	Agente extensível do OpenView	189
<b>12</b>	<b><i>Adaptando o SNMP a seu ambiente</i></b>	<b>202</b>
	Programa de criação de traps gerais	202
	Quem está entrando em minha máquina?(I-Am-in)	203
	Eliminando arquivos do núcleo	205
	Verificação de disco Veritas	209
	Verificador de espaço de disco	214
	Monitor de Portas	225
<b>13</b>	<b><i>MRTG</i></b>	<b>230</b>
	Usando o MRTG	231
	Exibindo gráficos	236
	Representação gráfica de outros objetos	237
	Outros aplicativos de obtenção de dados	242
	Atenção	244
	Como obter ajuda	244

A	<i>Usando octetos de entrada e saída</i>	245
B	<i>Considerações adicionais sobre o NNM da OpenView</i>	253
C	<i>Ferramentas do Net-SNMP</i>	263
D	<i>RFCs do SNMP</i>	275
E	<i>Suporte para a linguagem Perl no SNMP</i>	281
F	<i>SNMPv3</i>	287
	<i>Índice</i>	299

## O que é SNMP?

Na complexa rede de roteadores, switches e servidores dos dias atuais, talvez pareça aterrorizante gerenciar todos os dispositivos existentes em sua rede e certificar-se de que estejam não somente em execução, como também funcionando perfeitamente. É exatamente nesse momento que o *Simple Network Management Protocol* (SNMP) pode entrar em ação. O SNMP foi lançado em 1988 para atender à necessidade cada vez maior de um padrão para gerenciar os dispositivos de IP (*Internet Protocol*). O SNMP oferece aos usuários um conjunto “simples” de operações que permitem o gerenciamento remoto desses dispositivos.

Este livro é dedicado aos administradores de sistema que desejam começar a utilizar o SNMP para gerenciar os respectivos servidores ou roteadores, mas não têm conhecimento sobre o assunto nem sabem como fazê-lo. Tentamos apresentar as informações básicas do que significa o SNMP e como ele funciona; além disso, ensinamos a utilizar o SNMP na prática, por meio de algumas ferramentas amplamente disponíveis. Acima de tudo, queremos tornar esse livro prático – um livro que ajuda a rastrear o que a rede está fazendo.

### Monitoração e gerenciamento de rede

O núcleo do SNMP é um conjunto simples de operações (e das informações obtidas por essas operações) que permitem ao administrador modificar o estado de alguns dispositivos baseados em SNMP. Por exemplo, é possível utilizar o SNMP para encerrar uma interface em um roteador ou verificar a velocidade operacional de uma interface de Ethernet. O SNMP pode até monitorar a temperatura de um comutador e avisar quando ela estiver muito alta.

Geralmente, o SNMP é associado ao gerenciamento de roteadores, mas é importante saber que ele pode ser utilizado para gerenciar vários tipos de dispositivos. Embora o antecessor do SNMP, o *Simple Gateway Management Protocol* (SGMP), tenha sido desenvolvido para gerenciar roteadores da Internet, é possível utilizar o SNMP para gerenciar sistemas Unix, Windows, impressoras, racks de modem, fontes de energia e muito mais. É possível gerenciar qualquer |

dispositivo executando um software que permita a recuperação de informações de SNMP. Isso inclui não apenas dispositivos físicos, mas também software, como servidores da Web e bancos de dados.

Outro aspecto do gerenciamento de rede é a monitoração da *rede*; ou seja, monitorar uma rede inteira em vez de componentes individuais, como roteadores, hosts e outros dispositivos. O *Remote Network Monitoring* (RMON) foi desenvolvido para ajudar a entender o funcionamento da própria rede, e como os dispositivos individuais afetam a rede como um todo. É possível utilizá-lo para monitorar não somente o tráfego de LAN (*Local Area Network*), como também as interfaces de WAN (*Wide Area Network*). Discutiremos o RMON com detalhes, mais adiante neste capítulo e no Capítulo 2.

Antes de avançar mais, examinemos o cenário anterior e posterior, que mostra que o SNMP faz diferença em uma organização.

## *Antes e depois do SNMP*

Vamos supor que você tenha uma rede com 100 equipamentos executando diversos sistemas operacionais. Algumas máquinas são servidores de arquivo, outras são servidores de impressão, uma delas executa um software que verifica transações de cartões de crédito (presumivelmente, em um sistema de emissão de pedidos baseado na Web), e os demais equipamentos são estações de trabalho pessoais. Além disso, existem vários comutadores e roteadores que ajudam a manter o funcionamento da rede física. Um circuito T1 conecta a empresa à Internet global e existe uma conexão privada com o sistema de verificação de cartões de crédito.

O que acontece quando um dos servidores de arquivos é derrubado? Se isso ocorrer no meio da semana, é provável que os usuários desse servidor percebam e o administrador pertinente seja chamado para corrigir o problema. Mas e se isso acontecer quando todos já tiverem saído, inclusive os administradores, ou durante o final de semana?

E se a conexão privada com o sistema de verificação de cartões de crédito cair às 10 horas da manhã de uma sexta-feira, e não for restaurada até segunda-feira de manhã? Se o problema foi um defeito de hardware e não puder ser corrigido trocando uma placa ou substituindo o roteador, talvez se percam milhares de dólares em vendas nos sites da Web por um motivo tão ínfimo. Da mesma forma, se o circuito T1 para a Internet cair, isso poderia prejudicar o volume de vendas gerado pelas pessoas acessando o site da Web e colocando pedidos.

Evidentemente, esses são problemas sérios – que certamente podem afetar a sobrevivência de uma empresa. Nesse momento, o SNMP entra em ação. Em vez de esperar que alguém perceba que alguma coisa está errada e localizar o responsável pela correção do problema (o que talvez não aconteça até segunda-feira de manhã, se o problema ocorrer no fim de semana), o SNMP permite monitorar a rede constantemente, mesmo que você não esteja presente. Por exemplo, ele observará se está aumentando o número de pacotes defeituosos entrando em uma das interfaces de roteador, informando que o roteador está pres-

tes a falhar. Você pode configurar uma notificação automática quando a falha for iminente, para lhe permitir corrigir o roteador antes da interrupção real.

Você também pode configurar para ser avisado se o processador de cartões de crédito estiver com problemas – você poderá até corrigir o problema de sua casa. E se nada der errado, você voltará ao escritório na segunda-feira de manhã sabendo que não encontrará surpresas.

Na realidade, não haverá um reconhecimento glorioso ao corrigir problemas com antecedência, mas você e sua diretoria descansarão em paz. Não sabemos como transformar tudo isso em um salário mais alto – às vezes, é melhor ser aquele cara que entra em cena, de repente, e conserta o que deu errado no meio de uma crise, em vez de ser o que impede que a crise ocorra. Entretanto, o SNMP não permite manter registros que comprovem que a rede está executando confiavelmente e que informem quando você tomou uma medida para evitar a crise.

## *Considerações sobre recursos humanos*

A implementação de um sistema de gerenciamento de rede pode significar uma expansão da equipe para lidar com o aumento da carga de manutenção e operação de um ambiente desse tipo. Ao mesmo tempo, a inclusão dessa modalidade de monitoração deve reduzir, na maioria dos casos, a carga de trabalho da equipe de administração do sistema. Será necessário:

- Uma equipe para manter a estação de gerenciamento. Isso inclui assegurar que essa estação esteja configurada para lidar corretamente com os eventos de dispositivos com recurso de SNMP.
- Uma equipe para manter os dispositivos com recurso de SNMP. Isso inclui garantir que as estações de trabalho e os servidores consigam se comunicar com a estação de gerenciamento.
- Uma equipe para observar e corrigir a rede. Geralmente, esse grupo é denominado *Network Operations Center* (NOC) e trabalha 24 horas por dia, 7 dias por semana. Uma alternativa para esse grupo de 24/7 é implementar o plantão rotativo, no qual uma pessoa recebe chamadas o tempo todo, mas não está necessariamente presente no escritório. O plantão funciona somente em ambiente de rede de pequeno porte, em que uma interrupção na operação da rede possa aguardar que alguém se dirija até o escritório para corrigir o problema.

Não é possível prever a quantidade de integrantes da equipe necessária para manter um sistema de gerenciamento. O tamanho da equipe varia em função do porte e da complexidade da rede gerenciada. Alguns provedores de backbone maiores da Internet dispõem de 70 pessoas ou mais em seus NOCs, enquanto outros têm apenas uma.

## Versões de RFCs e SNMP

A Internet Engineering Task Force (IETF) é responsável pela definição de protocolos padrão que controlam o tráfego na Internet, incluindo o SNMP. A IETF publica *Requests for Comments* (RFCs), que são especificações para os diversos protocolos existentes no mundo do IP. Primeiramente, os documentos se submetem ao rastreamento de padrões como padrões *sugeridos*, depois passam para o status de *draft*. Quando um *draft* final é ocasionalmente aprovado, a RFC recebe o status de *standard* (*padrão*) – se bem que existem bem menos padrões totalmente aprovados do que você imagina.

Duas outras designações de rastreamento de padrões, *histórico* e *experimental*, definem (respectivamente) um documento substituído por uma RFC mais recente e um documento que ainda não está pronto para se tornar um padrão. A lista a seguir engloba todas as versões atuais de SNMP e o status emitido pela IETF para cada uma (consulte o Apêndice D para obter uma lista completa das RFCs do SNMP):

- **SNMP Version 1 (SNMPv1)** – é a versão padrão atual do protocolo SNMP, definida na RFC 1157 e é um padrão completo da IETF. A segurança do SNMPv1 baseia-se em *comunidades*, que não são nada mais do que senhas: strings de texto puro que permitem que qualquer aplicativo baseado em SNMP (que reconheça a string) tenha acesso a informações de gerenciamento de um dispositivo. Geralmente, existem três comunidades no SNMPv1: *read-only*, *read-write* e *trap*.
- **SNMP Version 2 (SNMPv2)** – é freqüentemente citado como SNMPv2 baseado em strings de comunidade. Essa versão do SNMP tem a denominação técnica SNMPv2c, mas, neste livro, citaremos essa versão apenas como SNMPv2. Ela está definida na RFC 1905, RFC 1906 e RFC 1907, é uma IETF com status experimental. Embora seja experimental, alguns fornecedores já começaram a aceitá-la na prática.
- **SNMP Version 3 (SNMPv3)** – será a próxima versão do protocolo a alcançar o status completo da IETF. No momento, é um padrão sugerido, definido na RFC 1905, RFC 1906, RFC 1907, RFC 2571, RFC 2572, RFC 2573, RFC 2574 e RFC 2575, que inclui suporte para autenticação rigorosa e comunicação privativa entre as entidades gerenciadas. O Apêndice F apresenta uma introdução ao SNMPv3 e analisa a configuração do agente do SNMPv3 para o Net-SNMP e Cisco. As informações desse apêndice fornecem ao administrador de sistema ou de rede o conhecimento prático necessário para começar a utilizar o SNMPv3, à medida que o produto ganha aceitação no mundo do gerenciamento de rede.

O site oficial de RFCs é o <http://www.ietf.org/rfc.html>. Entretanto, um dos maiores problemas relacionados às RFCs é localizar a que você deseja. Talvez seja um pouco mais fácil navegar pelo índice de RFCs da Ohio State University (Universidade Estadual de Ohio – <http://www.cis.ohio-state.edu/services/rfc/index.html>).

## *Gerenciadores e agentes*

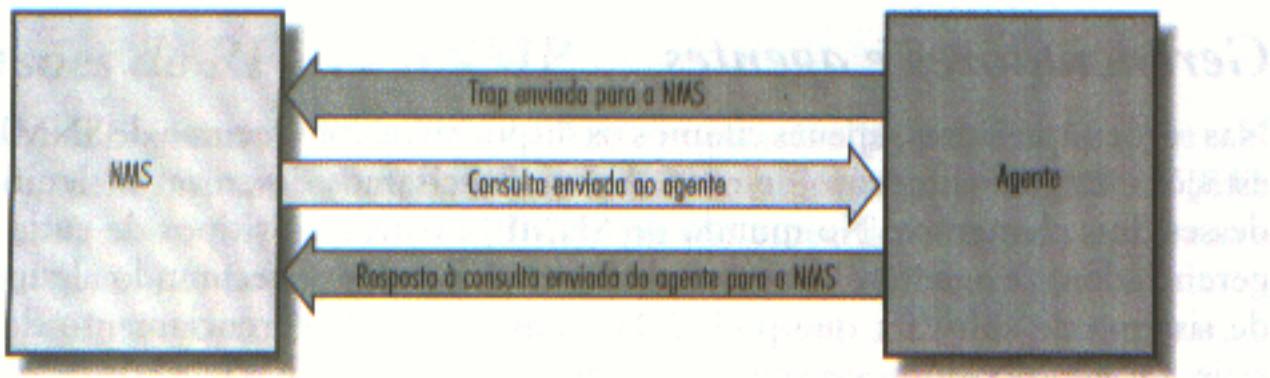
Nas seções anteriores, apenas citamos os dispositivos com recurso de SNMP e as estações de gerenciamento de rede. Agora, precisamos descrever o significado desses dois elementos. No mundo do SNMP, existem dois tipos de entidades: gerenciadores e agentes. Um gerenciador é um servidor executando algum tipo de sistema de software que pode lidar com tarefas de gerenciamento de uma rede. Os gerenciadores costumam ser chamados NMSs (*Network Management Stations* – estações de gerenciamento de rede)\*. Uma NMS é responsável pela operação de polling e por receber traps de agentes na rede.

Uma *poll*, no contexto de gerenciamento de rede, é a operação de consultar informações em um agente (roteador, comutador, servidor Unix etc.). Essas informações podem ser utilizadas posteriormente para detectar se ocorreu algum tipo de evento desastroso. Uma *trap* é um método utilizado por um agente para informar à NMS que algo aconteceu. As traps são enviadas de modo assíncrono, não em resposta a consultas da NMS. Além disso, a NMS é responsável por executar uma ação\*\* baseada nas informações recebidas do agente. Por exemplo, quando um circuito T1 de conexão à Internet é derrubado, o roteador pode enviar uma trap à NMS. Por sua vez, a NMS pode reagir, talvez com um aviso que lhe permita saber que algo aconteceu.

A segunda entidade, o *agente*, é a peça de software executada nos dispositivos da rede gerenciados por você. Pode ser um programa separado (um *daemon*, na terminologia do Unix) ou pode ser incorporado ao sistema operacional (por exemplo, o IOS da Cisco em um roteador, ou o sistema operacional de baixo nível que controla um *no-break*). Atualmente, a maioria dos dispositivos de IP é fornecida com alguma modalidade de agente de SNMP interno. O fato de os fornecedores desejarem implementar agentes em alguns de seus produtos facilita ainda mais o serviço do administrador de sistema ou do gerente da rede. O agente fornece à NMS informações de gerenciamento, rastreando os diversos aspectos operacionais dos dispositivos. Por exemplo, o agente em um roteador pode rastrear o estado de cada uma das respectivas interfaces: quais estão funcionando, quais estão paradas etc. A NMS pode consultar o status de cada interface em um roteador e reagir adequadamente, se algumas delas estiverem paradas. Quando o agente percebe algo errado, pode enviar uma trap à NMS. Essa trap se origina no agente e é enviada à NMS, onde é tratada adequadamente. Alguns dispositivos enviarão uma trap correspondente de “all clear”, quando ocorrer uma transição de um estado defeituoso para um estado correto. Esse recurso pode ser prático ao determinar quando uma situação problemática foi solucionada. A Figura 1-1 mostra o relacionamento existente entre a NMS e um agente.

\* Consulte o Capítulo 5 para ler uma discussão com os prós e contras de algumas aplicações populares de NMS.

\*\* Lembre-se de que a NMS está configurada para executar tal ação.



**Figura 1-1 Relacionamento entre uma NMS e um agente**

É importante lembrar que as polls e traps podem acontecer simultaneamente. Não há restrições sobre quando a NMS pode consultar o agente nem sobre quando um agente pode enviar uma trap.

## Estrutura de informações de gerenciamento e MIBs

A SMI (*Structure of Management Information* – estrutura de informações de gerenciamento) é um método para definir objetos gerenciados e os respectivos comportamentos. Um agente possui uma lista dos objetos por ele rastreados. Esse tipo de objeto é o status operacional de uma interface de roteador (por exemplo, *em funcionamento*, *parada* ou *em teste*). Essa lista define coletivamente as informações que a NMS pode utilizar para detectar o funcionamento geral do dispositivo em que o agente reside.

A MIB (*Management Information Base* – base de informações de gerenciamento) pode ser considerada um banco de dados de objetos gerenciados que o agente rastreia. Todo tipo de informações sobre status ou estatísticas acessado pela NMS é definida em uma MIB. A SMI é um método para definir objetos gerenciados, enquanto a MIB é a definição (por meio da sintaxe da SMI) dos próprios objetos. Como um dicionário, que mostra como pronunciar uma palavra e, em seguida, apresenta o significado ou a definição dessa mesma palavra, uma MIB define um nome em texto de um objeto gerenciado e explica o seu significado. O Capítulo 2 descreve as MIBs e a SMI com mais detalhes.

Um agente pode implementar várias MIBs, mas todos os agentes implementam uma MIB específica denominada *MIB-II*\* (RFC 1213). Esse padrão define variáveis para elementos como dados estatísticos de uma interface (velocidades da interface, MTU, octetos\*\* enviados e recebidos etc.), assim como alguns outros aspectos relacionados ao próprio sistema (localização do sistema, contato do sistema etc.). O principal objetivo da MIB-II é fornecer informações gerais sobre gerenciamento de TCP/IP e não engloba todo item possível que um fornecedor gerenciaria em um dispositivo específico.

\*MIB-I é a versão original dessa MIB, mas ela não é mais consultada desde que a MIB-II a implementou.

6 | \*\* Um octeto é formado por 8 bits, que é a unidade fundamental de transferência em redes TCP/IP.

Que outros tipos de informações seriam úteis? Primeiramente, existem vários padrões em draft e sugeridos, desenvolvidos para ajudar a gerenciar aspectos como o *frame relay* (protocolo de comutação de pacotes), o ATM (*Asynchronous Transfer Mode*), a FDDI (*Fiber Distributed Data Interface*) e serviços (correio, DNS [*Domain Name System*] etc.). Exemplos dessas MIBs e dos respectivos números de RFC seriam:

- ATM MIB (RFC 2515)
- Frame Relay DTE Interface Type MIB (RFC 2115)
- BGP Version 4 MIB (RFC 1657)
- RDBMS MIB (RFC 1697)
- RADIUS Authentication Server MIB (RFC 2619)
- Mail Monitoring MIB (RFC 2249)
- DNS Server MIB (RFC 1611)

Mas tudo isso não é tudo, há ainda a explicação do porque os fornecedores e pessoas comuns podem definir variáveis de MIB para uso próprio.\* Por exemplo, suponha um vendedor lançando um novo roteador no mercado. O agente incorporado ao roteador responderá às solicitações da NMS (ou enviará traps para a NMS) relacionadas às variáveis definidas pelo padrão MIB-II; provavelmente, também implementará MIBs para os tipos de interface por ele fornecidas (por exemplo, RFC 2515 para o ATM e RFC 2115 para o Frame Relay). Além disso, o roteador pode ter alguns recursos novos significativos que compensam monitorar, mas que não estão cobertos por qualquer MIB padrão. Conseqüentemente, o fornecedor define uma MIB exclusiva (às vezes citada como MIB proprietária) que implementa objetos gerenciados para as informações de status e estatísticas do novo roteador.



O simples carregamento de uma nova MIB em uma NMS não permite necessariamente a recuperação dos dados/valores/objetos etc. definidos nessa MIB. Basta carregar as MIBs suportadas pelos agentes consultados (por exemplo, *snmpget*, *snmpwalk*). Você pode carregar MIBs adicionais para um futuro suporte de dispositivos, mas não entre em pânico quando o dispositivo não responder (e possivelmente retornar erros) a essas MIBs ainda sem suporte.

## *Gerenciamento de hosts*

O gerenciamento de recursos do host (espaço de disco, utilização da memória etc.) é um aparte importante do gerenciamento de rede. A diferença entre a ad-

\* Este assunto será discutido em mais detalhes no próximo capítulo.

ministração dos sistemas tradicionais e o gerenciamento de rede foi desaparecendo aos poucos nos últimos dez anos e agora nem existe mais. Como afirma a Sun Microsystems, "A rede é o próprio computador". Se seu servidor da Web ou de correio estiver parado, não importa se os roteadores estão funcionando corretamente – você ainda receberá chamados. A *Host Resources MIB* (RFC 2790) define um conjunto de objetos para ajudar a gerenciar aspectos críticos dos sistemas Unix e Windows.\*

Alguns dos objetos aceitos pela Host Resources MIB incluem capacidade de disco, número de usuários do sistema, número de processos em execução e softwares atualmente instalados. No mundo atual do comércio eletrônico, um número cada vez maior de pessoas utiliza os sites da Web voltados para serviços. Assegurar o funcionamento correto dos servidores de back-end é tão importante quanto monitorar os roteadores e outros dispositivos de comunicação.

Infelizmente, algumas implementações de agentes para essas plataformas não incluem essa MIB, porque ela não é necessária.

## *Introdução sucinta ao Remote Monitoring (RMON)*

O *Remote Monitoring Version 1* (RMONv1 ou RMON) é definido na RFC 2819; uma versão otimizada do padrão, denominada RMON Version 2 (RMONv2), é definida na RFC 2021. O RMONv1 oferece à NMS dados estatísticos sobre uma LAN ou WAN inteira, no nível de pacotes. O RMONv2 aprimora o RMONv1 ao fornecer dados estatísticos no nível de rede e de aplicativos, que podem ser obtidos de várias maneiras. Um modo é colocar uma prova do RMON em cada segmento de rede a ser monitorado. Alguns roteadores da Cisco dispõem de recursos limitados de RMON incorporados, de modo que é possível a utilização de sua funcionalidade para executar atividades menos complexas de RMON. De modo semelhante, alguns comutadores da 3Com implementam a especificação completa de RMON e podem ser utilizados como provas de RMON com todos os recursos.

A RMON MIB foi elaborada para permitir que uma verdadeira investigação de RMON seja executada off-line, visando a obter dados estatísticos sobre a rede sendo observada, sem que a NMS faça consultas constantes. Mais tarde, a NMS pode solicitar os dados estatísticos obtidos na investigação. Outro recurso que a maioria das investigações implementa é a possibilidade de definir limiares para várias condições de erro e, quando um limiar é ultrapassado, pode alertar à NMS por meio de uma trap do SNMP. você encontrará detalhes mais técnicos sobre o RMON no próximo capítulo.

## *Como obter mais informações*

Talvez pareça desanimador dominar o SNMP. As RFCs oferecem uma definição oficial do protocolo, mas são escritas para desenvolvedores de software, não

\* Qualquer sistema operacional executando um agente do SNMP pode implementar a Host Resources, que não está restrita aos agentes em execução nos sistemas Unix e Windows.

para administradores de rede, e talvez seja difícil obter as informações necessárias. Felizmente, existem alguns recursos on-line disponíveis. O Web site mais relevante é o Network Management Server, localizado na Universidade de Buffalo (<http://netman.cit.buffalo.edu>), que contém links úteis para outros sites que fornecem informações semelhantes, assim como uma lista de produtos de gerenciamento de rede (<http://netman.cit.buffalo.edu/Products.html>) que engloba fornecedores de software e de hardware; esse site também possui revisões de produtos, é um excelente ponto de partida para a pesquisa de informações sobre gerenciamento de rede e pode ser uma ferramenta muito útil para determinar que tipos de hardware e software existem atualmente. Outros dois Web sites excelentes são o SimpleWeb (<http://wwwsnmp.cs.utwente.nl>) e o SNMP Link (<http://www.SNMPLink.org>). *The Simple Times*, uma publicação on-line dedicada ao SNMP e a gerenciamento de rede, também é útil. Você encontrará a edição atual e todas as anteriores, em <http://www.simple-times.org>.

Outro recurso eficiente é o Usenet News. O newsgroup que a maioria das pessoas freqüenta é o *comp.dcom.net-management*. Outro newsgroup interessante é o *comp.protocols.snmp*.

Grupos como esses promovem uma comunidade de compartilhamento de informações, permitindo que os profissionais experientes interajam com as pessoas que não têm muito conhecimento sobre SNMP ou sobre gerenciamento de rede.

Para saber se um fornecedor específico tem equipamentos compatíveis com SNMP, a *Internet Assigned Numbers Authority* (IANA) compilou uma lista dos arquivos de MIBs proprietárias, fornecidas por diversos fornecedores. Você encontrará essa lista em <ftp://ftp.iana.org/mib/>. Também existe uma FAQ (perguntas freqüentes) sobre o SNMP disponível em <http://www.faqs.org/faqs/snmp-faq/part1/> e outra em <http://www.faqs.org/faqs/snmp-faq/part2/>.

## Examinando o SNMP

Neste capítulo, analisaremos o SNMP minuciosamente. Quando você terminá-lo, deverá saber como o SNMP envia e recebe informações, o que significam exatamente as comunidades do SNMP e como ler arquivos de MIB. Além disso, examinaremos com detalhes as três MIBs apresentadas no Capítulo 1: MIB-II, Host Resources e RMON.

### SNMP e UDP

O SNMP usa o *User Datagram Protocol* (UDP) como protocolo de transporte para passagem de dados entre gerenciadores e agentes. O UDP, definido na RFC 768, foi escolhido por meio do *Transmission Control Protocol* (TCP) por não ter conexão; isto é, nenhuma conexão ponto-a-ponto é estabelecida entre o agente e a NMS quando os *datagramas* (pacotes) são transferidos de um lado para o outro. Esse aspecto do UDP torna-o não confiável, uma vez que não existe confirmação de datagramas perdidos no nível do protocolo. O próprio aplicativo do SNMP se encarregar de detectar se ocorreu perda de datagramas e retransmite-os, se necessário. Em geral, isso é feito com um simples *timeout* (tempo de espera). A NMS envia uma solicitação de UDP para um agente e aguarda uma resposta. O intervalo de tempo em que a NMS espera depende da configuração na NMS. Ser o timeout for decorrido e a NMS não receber resposta do agente, será presumida a perda do pacote e a NMS retransmite a solicitação. O número de retransmissões de pacotes pela NMS também é configurável.

Pelo menos no que diz respeito às solicitações regulares de informações, a natureza não confiável do UDP não é um grande problema. Na pior das hipóteses, a estação de gerenciamento emite uma solicitação e nunca recebe uma resposta. Para as traps, a situação é um pouco diferente. Se um agente envia uma trap e ela nunca chega, a NMS não sabe se essa trap sequer foi enviada. Nem o próprio agente detecta a necessidade de reenvio da trap, por que a NMS não é obrigada a enviar uma resposta para o agente, confirmado o recebimento da trap.

O reverso da natureza incerta do UDP é o baixo overhead, de modo que o impacto sobre o desempenho da rede é menor. O SNMP foi implementado

através do TCP; entretanto, essa implementação está mais voltada para as situações de casos especiais em que alguém está desenvolvendo um agente para um equipamento proprietário. Em uma rede gerenciada e muito congestionada, o SNMP por meio do TCP é uma péssima idéia. É importante entender que o TCP não é um mágico e que o SNMP foi elaborado para trabalhar com redes enfrentando problemas – se sua rede nunca falhasse, não seria necessário monitorá-la. Quando uma rede está falhando, um protocolo que tenta obter os dados, mas desiste quando não consegue, é certamente uma opção de design melhor do que um protocolo que inunda a rede com retransmissões, na tentativa de obter credibilidade.

O SNMP usa a porta 161 do UDP para enviar e receber solicitações e a porta 162 para receber traps de dispositivos gerenciados. Todo dispositivo que implementa o SNMP deve utilizar esses números de porta como defaults, mas alguns fornecedores permitem modificar as portas defaults na configuração do agente. Se esses defaults forem modificados, a NMS deve ser informada sobre essas alterações para consultar o dispositivo por meio das portas corretas.

A Figura 2-1 mostra o conjunto de protocolos TCP/IP, que é a base de toda a comunicação por TCP/IP. Atualmente, todo dispositivo para comunicação na Internet (como os sistemas Windows NT, servidores Unix, roteadores Cisco etc.) deve utilizar esse conjunto de protocolos. Esse modelo é geralmente citado como uma pilha de protocolos, porque cada camada usa as informações da camada posicionada imediatamente abaixo.

Quando uma NMS ou um agente precisa executar uma função de SNMP (como uma solicitação ou uma trap), ocorrem os seguintes eventos na pilha de protocolos:

### *Aplicativo*

Primeiramente, o aplicativo de SNMP (NMS ou agente) determina o que fará. Por exemplo, ele pode enviar uma solicitação de SNMP para um agente, pode enviar uma resposta a uma solicitação de SNMP (enviada a partir do agente) ou enviar uma trap para uma NMS. A camada do aplicativo fornece serviços para um usuário final, como um operador solicitando informações de status de uma porta em um comutador de Ethernet.

### *UDP*

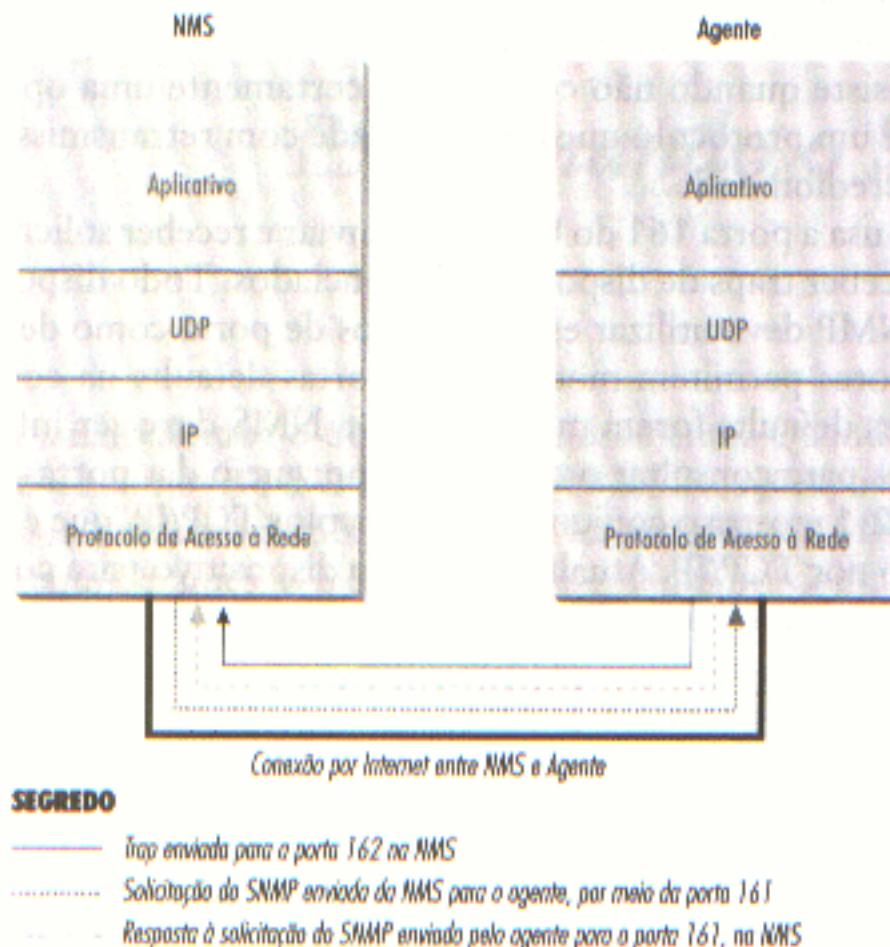
A camada seguinte, UDP, permite a comunicação entre dois hosts. O cabeçalho do UDP contém, entre outras informações, a porta de destino do dispositivo para o qual ela está enviando uma solicitação ou uma trap. Essa porta de destino será a 161 (consulta) ou 162 (trap).

### *IP*

A camada IP tenta fornecer o pacote de SNMP ao destino almejado, conforme especificado pelo respectivo endereço IP.

## *Medium Access Control (MAC)*

O último evento que deve ocorrer para um pacote de SNMP alcançar seu destino é ser controlado na rede física, onde pode ser direcionado para o destino final.



*Figura 2-1 Modelo de comunicação por TCP/IP e o SNMP*

A camada MAC é formada pelos verdadeiros drivers de hardware e de dispositivo que colocam seus dados em um cabeamento físico, como um cartão Ethernet. Essa camada também é responsável por receber os pacotes da rede física e reenviá-los para a pilha de protocolos, para que sejam processados pela camada do aplicativo (SNMP, nesse caso).

Essa interação entre os aplicativos de SNMP e a rede não é diferente da que existe entre duas pessoas que se correspondem por carta. Ambas têm mensagens que precisam ser trocadas entre si. Vamos supor que você precise escrever uma carta para sua correspondente perguntando se ela gostaria de visitá-lo no verão. Ao decidir enviar o convite, você atuou como o aplicativo de SNMP. Preencher o envelope com o endereço de sua correspondente equivale à função da camada de UDP, que registra a porta de destino do pacote no cabeçalho do UDP; nesse caso, é o endereço de sua correspondente. Colocar o selo no envelope e inserir esse envelope na caixa de correio para o carteiro recolher corresponde à função da camada de IP. A última ação acontece quando o carteiro recolhe a carta. A partir daí, a carta será enviada ao destino final, a caixa de correio de sua correspondente. A camada de MAC de uma rede de computadores equivale às caminhonetes e

aviões do correio que transportam a carta. Quando sua correspondente receber essa carta, passará pelo mesmo processo para lhe enviar uma resposta.

## Comunidades de SNMP

O SNMPv1 e SNMPv2 usam o conceito de comunidades para definir uma confiabilidade entre gerenciadores e agentes. Um agente é configurado com três nomes de comunidade: *read-only*, *read-write* e *trap*. Os nomes de comunidades são basicamente senhas; não existe diferença entre uma string de comunidade e a senha que você usa para acessar sua conta no computador. As três strings de comunidade controlam tipos diferentes de atividades. Como o próprio nome sugere, a string de comunidade *read-only* permite ler valores de dados, sem modificá-los. Por exemplo, essa string permite ler o número de pacotes transferidos por meio das portas existentes no roteador mas não permite redefinir os contadores. A comunidade *read-write* pode ler e modificar valores de dados; com a string de comunidade *read-write*, é possível ler os contadores, redefinir seus valores e até as interfaces ou executar outras ações que modifiquem a configuração do roteador. Por último, a string de comunidade *trap* permite receber traps (notificações assíncrona) do agente.

A maioria dos fornecedores entrega seus equipamentos com strings de comunidade defaults, geralmente *public* para a comunidade *read-only* e *private* para a comunidade *read-write*. É importante modificar esses defaults antes de instalar o dispositivo definitivamente na rede. (Talvez você esteja cansado de ouvir essa recomendação porque repetimos essa história várias vezes, mas é absolutamente imprescindível.) Ao configurar um agente de SNMP, convém configurar o destino de sua trap, que é o endereço para o qual o agente enviará todas as traps por ele geradas. Além disso, como as strings de comunidade do SNMP são enviadas em texto comum, é possível configurar um agente para enviar uma trap de falha de autenticação de SNMP quando alguém tentar consultar um dispositivo com uma string de comunidade incorreta. Entre outros aspectos, as traps de falha de autenticação podem ser muito úteis ao determinar quando um invasor estiver tentando acessar sua rede.

Como as strings de comunidade são basicamente senhas, para selecioná-las use as mesmas regras aplicadas às senhas de usuário do Unix ou NT: nenhuma palavra do dicionário, nomes de esposas etc. Geralmente, é uma boa idéia usar uma string alfanumérica com combinação de maiúsculas/minúsculas. Conforme mencionado anteriormente, o problema com a autenticação do SNMP é que as strings de comunidade são enviadas em texto simples, o que facilita a interceptação dessas strings e o uso indevido contra você. O SNMPv3 trata dessa questão ao permitir, entre outros aspectos, autenticação e comunicação segura entre os dispositivos do SNMP.

Existem meios de reduzir os riscos de ataque. Os firewalls ou filtros de IP minimizam a chance de alguém prejudicar um dispositivo gerenciado em uma rede, atacando-o por meio do SNMP. Você pode configurar o firewall para aceitar o tráfego do UDP somente de uma lista de hosts conhecidos. Por exemplo, é

possível o tráfego do UDP na porta 161 (solicitações do SNMP) em sua rede somente quando procedente de uma de suas estações de gerenciamento de rede.

O mesmo é válido para as traps; você pode configurar o roteador para aceitar o tráfego de UDP pela porta 162 para sua NMS somente se procedente de um dos hosts sendo monitorados. Os firewalls não são totalmente eficientes mas precauções simples, como estas, reduzem bastante seus riscos.



É importante saber que se alguém tiver acesso de leitura-gravação a qualquer um de seus dispositivos do SNMP, poderá controlá-los por meio do SNMP (por exemplo, poderá definir as interfaces do roteador, desabilitar portas ou até modificar as tabelas de direcionamento). Uma maneira de proteger as strings de comunidade é usar uma *Virtual Private Network* (VPN) para garantir que o tráfego da rede seja codificado. Outro método é mudar freqüentemente as strings de comunidade. Em uma rede pequena, não é difícil modificar essas strings, mas em uma rede que ocupa quarteirões da cidade ou muito mais, e possui dezenas (ou centenas ou milhares) de hosts gerenciados, pode ser problemático mudar as strings de comunidade. Uma solução fácil é escrever um script simples em Perl que use o SNMP para alterar essas strings em seus dispositivos.

## *Structure of Management Information*

Até agora, empregamos a expressão “informações de gerenciamento” no contexto de parâmetros operacionais de dispositivos com recurso de SNMP. Entretanto, falamos muito pouco sobre o que as informações de gerenciamento realmente contêm ou como são representadas. O primeiro passo para entender que tipo de informação um dispositivo pode fornecer é conhecer como esses dados são representados no contexto do SNMP. A *Structure of Management Information Version 1* (SMIv1, RFC 1155) faz exatamente isso: define com exatidão como os objetos gerenciados<sup>\*</sup> são nomeados e especifica os respectivos tipos de dados associados. A *Structure of Management Information Version 2* (SMIv2, RFC 2578) fornece otimizações para o SNMPv2. Começaremos nossa discussão com a SMIv1 e abordaremos a SMIv2 na seção seguinte.

A definição de objetos gerenciados pode ser fragmentada em três atributos:

### *Nome*

O *nome ou identificador de objeto* (OID – Object Identifier) define com exclusividade um objeto gerenciado. Geralmente, os nomes são exibidos em

\* No restante deste livro, a expressão “informações de gerenciamento” será citada como “objetos gerenciados”. De modo semelhante, um único fragmento de informação de gerenciamento (como o status operacional da interface de um roteador) será conhecido como um “objeto gerenciado”.

dois formatos: numérico e o “legível pelo ser humano”. Em ambos os casos, os nomes são longos e inconvenientes. Nos aplicativos de SNMP, você aprende a navegar corretamente pelo namespace.

### *Tipo e sintaxe*

O tipo de dado de um objeto gerenciado é definido por meio de um subconjunto da *Abstract Syntax Notation One* (ASN.1). A ASN.1 é um meio de especificar o modo como os dados são representados e transmitidos entre gerenciadores e agentes, no contexto do SNMP. A vantagem em relação à ASN.1 é o fato de que a notação independe da máquina. Isso significa que um PC executando o Windows NT pode ser comunicar com uma máquina do Sun SPARC sem considerar aspectos como o seqüenciamento de bytes.

### *Codificação*

*Uma única instância de um objeto gerenciado é codificada em uma string de octetos por meio do método Basic Encoding Rules (BER).* O método BER define o modo de codificação e decodificação dos objetos para que sejam transmitidos através de um meio de transporte, como a Ethernet.

## *Nomeando OIDs*

Os objetos gerenciados são organizados em uma hierarquia em árvore. Essa estrutura é base do esquema de atribuição de nomes do SNMP. Um ID de objeto é formado por uma seqüência de inteiros baseada nos nós da árvore, separada por pontos (.). Embora exista uma forma legível ao ser humano mais amistosa do que uma string de números, essa forma não é nada mais do que uma seqüência de nomes separados por pontos, cada qual representando um nó da árvore. Assim, é possível utilizar os próprios números ou uma seqüência de nomes que representam os números. A Figura 2-2 mostra os primeiros níveis dessa árvore. (Excluímos de propósito algumas ramificações da árvore que não nos interessam aqui.)

Na árvore de objetos, o nó posicionado no início da árvore é denominado *raiz*, tudo o que tiver filhos será uma *subárvore* e tudo o que não tiver filhos será chamado *nó de folha*. Por exemplo, a raiz na Figura 2-2, o ponto inicial da árvore, é denominada “Root-Node” (Nó-raiz). A respectiva subárvore é formada por *ccitt(0)*, *iso(1)* e *joint(2)*. Nessa ilustração, *iso(1)* é o único nó que contém uma subárvore; os outros dois nós são nós de folha. *ccitt(0)* e *joint(2)* não pertencem ao SNMP e não serão discutidos neste livro.\*

---

\* A subárvore *ccitt* é administrada pelo International Telegraph and Telephone Consultative Committee (CCITT); a subárvore *joint* é administrada conjuntamente pela International Organization for Standardization (ISSO) e CCITT. Como mencionado anteriormente, nenhuma dessas ramificações está relacionada ao SNMP.

No restante deste livro, abordaremos a subárvore *iso(1).org(3).dod(6).internet(1)*,\* representada na forma de OID como 1.3.6.1 ou como *iso.org.dod.internet*. Cada objeto gerenciado possui uma OID numérica e nome em texto associado. A notação decimal de pontos é o modo de representação interna de um objeto gerenciado dentro de um agente; o nome em texto, como um nome de domínio IP, evita a necessidade de memorizar strings de inteiros longas e cansativas.

A ramificação *directory* não é usada no momento. A ramificação *management*, ou *mgmt*, define um conjunto padrão de objetos de gerenciamento da Internet. A ramificação *experimental* é reservada para fins de teste e pesquisa.

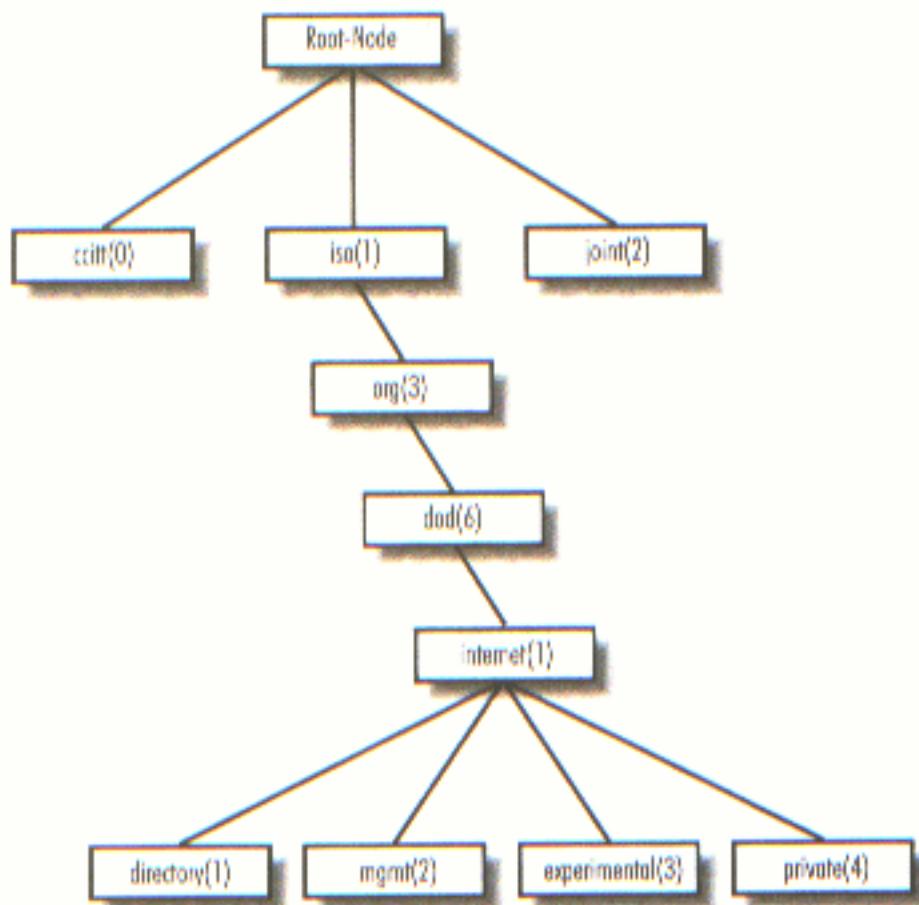


Figura 2-2 Árvore de objetos da SMI

Os objetos na ramificação *private* são definidos unilateralmente, o que significa que os indivíduos e as organizações são responsáveis pela definição dos objetos dessa ramificação. Examine a definição da subárvore *internet* e de quatro de suas subárvore:

```
internet      OBJECT IDENTIFIER ::= { iso org{3} dod{6} 1 }
directory     OBJECT IDENTIFIER ::= { internet 1 }
mgmt         OBJECT IDENTIFIER ::= { internet 2 }
experimental   OBJECT IDENTIFIER ::= { internet 3 }
private        OBJECT IDENTIFIER ::= { internet 4 }
```

A primeira linha declara *internet* como a OID 1.3.6.1, que está definida como uma subárvore de *iso.org.dod* ou 1.3.6 ( ::= é um operador de definição). As quatro últimas declarações são semelhantes, mas definem as outras ramificações que pertencem à ramificação *internet*. Para a ramificação *directory*, a notação { *internet* 1 } informa que ela faz parte da ramificação *internet* e que a OID é 1.3.6.1.1. A OID de *mgmt* é 1.3.6.1.2, e assim por diante.

No momento, existe uma única ramificação na subárvore *private*, usada para permitir que os fornecedores de hardware e software definam seus objetos privados para qualquer tipo de hardware ou software que o SNMP deva gerenciar. A respectiva definição da SMI é:

```
enterprises OBJECT IDENTIFIER ::= { private 1 }
```

A Internet Assigned Numbers Authority (IANA) gerencia atualmente todas as atribuições de números de empresa privada para indivíduos, instituições, organizações, empresas etc.\* É possível obter uma lista de todos os números atuais de empresas privadas em <ftp://ftp.isi.edu/in-notes/iana/assignments/enterprise-numbers>. Por exemplo, o número de empresa privada da Cisco Systems é 9, e a OID base para o respectivo espaço de objetos é definida como *iso.org.dod.internet.private.enterprises.cisco* ou 1.3.6.1.4.1.9. A Cisco pode criar uma ramificação *private*, se necessário. Geralmente, empresas como a Cisco, que fabrica equipamentos de rede, definirem seus próprios objetos *private enterprise* (empresa privada). Isso permite um conjunto mais completo de informações de gerenciamento do que o obtido com base no conjunto padrão de objetos gerenciados definidos na ramificação *mgmt*.

As empresas não são as únicas que podem registrar seus próprios números de empresas privadas. Qualquer um pode fazê-lo. O formulário baseado na Web para registrar números de empresas privadas pode ser encontrado em <http://www.isi.edu/cgi-bin/iana/enterprise.pl>. Depois que você preencher o formulário, que solicita informações como o nome da organização e informações de contatos, sua solicitação deve ser aprovada em uma semana. Por que você desejaria registrar um número exclusivo? Quando você dominar o SNMP, descobrirá aspectos a serem monitorados que não constam em qualquer MIB, pública ou privada. Com um número de empresa exclusivo, você pode criar sua MIB privada, que permite monitorar exatamente o que você deseja. Seja astuto ao estender os agentes para que procurem as informações necessárias.

## Definindo OIDs

O atributo SYNTAX oferece definições de objetos gerenciados por meio de um subconjunto do ASN.1. A SMIV1 define vários tipos de dados primordiais para o gerenciamento de redes e dispositivos de rede. É importante lembrar que esses tipos de dados são apenas um meio de definir o tipo de informação que um objeto gerenciado pode conter. Os tipos aqui discutidos são semelhantes àqueles que você definiria em uma linguagem de programação de computador, como a C. A Tabela 2-1 lista os tipos de dados aceitos na SMIV1.

\* O termo “empresa privada” será usado neste livro inteiro em relação à ramificação *enterprises*.

Tabela 2-1 Tipos de dados da SMIv1

Tipo de dado	Descrição
INTEGER	Geralmente, um número de 32 bits usado para especificar tipos numerados no contexto de um único objeto gerenciado. Por exemplo, o status operacional da interface de um roteador pode ser <i>up</i> (em funcionamento), <i>down</i> (parado) ou <i>testing</i> (em teste). Com os tipos numerados, 1 representaria o status <i>em funcionamento</i> , 2 seria o status <i>parado</i> e 3, o status <i>em teste</i> . O valor zero (0) não deve ser usado como um tipo numerado, de acordo com a RFC 1155.
OCTET STRING	Uma string de zero ou mais octetos (conhecidos mais comumente como bytes) geralmente utilizada para representar strings de texto, mas usada ocasionalmente para representar endereços físicos.
Counter	Um número de 32 bits com o valor mínimo de 0 e máximo de $2^{32} - 1$ (4.294.967.295). Quando o valor máximo é alcançado, volta ao zero e inicia novamente. É basicamente utilizado para rastrear informações, como o número de octetos enviados e recebidos em uma interface ou o número de erros e descartes encontrados em uma interface. Um Counter aumenta monotonamente, no sentido de que seus valores nunca devem diminuir durante a operação normal. Quando um agente é reinicializado, todos os valores do Counter devem ser definidos com zero. São utilizados deltas para determinar se é possível declarar algo útil nas sucessivas consultas de valores do Counter. Um delta é computado ao consultar um Counter pelo menos duas vezes em uma linha e ao tirar a diferença entre os resultados das consultas durante um período de tempo.
OBJECT IDENTIFIER	Uma string decimal de pontos que representa um objeto gerenciado na árvore de objetos. Por exemplo, 1.3.6.1.4.1.9 representa a OID de empresa privada da Cisco Systems.
NULL	Atualmente, sem uso no SNMP.
SEQUENCE	Define listas que contêm zero ou mais tipos diferentes de dados do ASN.1.
SEQUENCE OF	Define um objeto gerenciado formado por uma SEQUENCE de tipos do ASN.1.
IpAddress	Representa um endereço do IPv4 de 32 bits. Nem a SMIv1 nem a SMIv2 discute endereços do IPv6 de 128 bits; esse problema será tratado pelo grupo de trabalho da SMI Next Generation (SMING) da IETF (consulte <a href="http://www.ietf.org/html.charters/smning-charter.html">http://www.ietf.org/html.charters/smning-charter.html</a> ).
NetworkAddress	Idêntico ao tipo IpAddress, mas pode representar tipos diferentes de endereços de rede.

Gauge	Um número de 32 bits com valor mínimo de 0 e máximo de $2^{32} - 1$ (4.294.967.295). Ao contrário de um Counter, um Gauge pode aumentar e diminuir aleatoriamente, mas sem ultrapassar o valor máximo. A velocidade da interface em um roteador é medida com um Gauge.
TimeTicks	Um número de 32 bits com valor mínimo de 0 e máximo de $2^{32} - 1$ (4.294.967.295). TimeTicks mede o tempo em centésimos de segundo. O tempo de funcionamento em um dispositivo é medido com esse tipo de dado.
Opaque	Permite o armazenamento de qualquer codificação do ASN.1 em uma OCTET STRING.

O objetivo de todos esses tipos de objetos é definir objetos gerenciados. No Capítulo 1, mencionamos que a MIB é um agrupamento lógico de objetos gerenciados à medida que pertencem a uma tarefa deg1 específica, a um vendedor específico etc. A MIB pode ser considerada uma especificação que define os objetos gerenciados que um fornecedor ou dispositivo aceita. Por exemplo, a Cisco tem literalmente centenas de MIBs definidas para sua vasta linha de produtos. Como exemplo, seu dispositivo Catalyst tem uma MIB separada de seu roteador da série 7000. Ambos os dispositivos possuem características diferentes que exigem capacidades de gerenciamento distintas. Geralmente, as MIBs específicas do fornecedor são distribuídas como arquivos de texto comuns, que podem ser verificados (ou até modificados) com um editor de texto padrão, como o *vi*.



A maioria dos produtos modernos de NMS mantém uma forma compacta de todas as MIBs que definem o conjunto de objetos gerenciados para todos os diversos tipos de dispositivos que gerenciam. Geralmente, os administradores de NMS compilam a MIB de um fornecedor em um formato que a NMS pode utilizar. Após carregar ou compilar uma MIB, os administradores podem fazer referência aos objetos gerenciados usando a ID de objeto numérica ou legível pelo ser humano.

É importante saber ler e entender os arquivos de MIB. O exemplo a seguir é uma versão reduzida da MIB-II (tudo o que estiver precedido por – é um comentário):

RFC1213-MIB DEFINITIONS ::= BEGIN

IMPORTS

    mgmt, NetworkAddress, IpAddress, Counter, Gauge,  
    TimeTicks  
        FROM RFC1155-SMI

OBJECT-TYPE

    FROM RFC 1212;

```
mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }
```

- groups in MIB-II

```
system      OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }
at          OBJECT IDENTIFIER ::= { mib-2 3 }
ip          OBJECT IDENTIFIER ::= { mib-2 4 }
icmp        OBJECT IDENTIFIER ::= { mib-2 5 }
tcp          OBJECT IDENTIFIER ::= { mib-2 6 }
udp          OBJECT IDENTIFIER ::= { mib-2 7 }
egp          OBJECT IDENTIFIER ::= { mib-2 8 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp        OBJECT IDENTIFIER ::= { mib-2 11 }
```

- Tabela Interfaces

- A tabela Interfaces contém informações sobre as interfaces da entidade. Cada interface é considerada associada a uma 'subnetwork' (sub-rede) Esse termo não deve ser confundido
  - com 'subnet' (sub-rede), que se refere a um esquema de particionamento de endereços, usado no conjunto de protocolos de Internet.

```
ifTable OBJECT-TYPE
```

```
SYNTAX SEQUENCE OF IfEntry
ACCESS not-accessible
STATUS mandatory
```

```
DESCRIPTION
```

```
"Uma lista de entradas da interface. O nº de entradas é especificado pelo valor de ifNumber."
```

```
::= { interfaces 2 }
```

```
ifEntry OBJECT-TYPE
```

```
SYNTAX IfEntry
ACCESS not-accessible
STATUS mandatory
```

```
DESCRIPTION
```

```
"Uma entrada de interface contendo objetos existentes na camada da sub-rede e abaixo de uma interface
```

específica."

```
INDEX { ifIndex }
 ::= { ifTable 1 }
```

```
IfEntry ::=
```

```
SEQUENCE {
    ifIndex
```

```
        INTEGER,
ifDescr
        DisplayString,
ifType
        INTEGER,
ifMtu
        INTEGER,
ifSpeed
        Gauge,
ifPhysAddress
        PhysAddress,
ifAdminStatus
        INTEGER,
ifOperStatus
        INTEGER,
ifLastChange
        TimeTicks,
ifInOctets
        Counter,
ifInUcastPkts
        Counter,
ifInNUcastPkts
        Counter,
ifInDiscards
        Counter,
ifInErrors
        Counter,
ifInUnknownProtos
        Counter,
ifOutOctets
        Counter,
ifOutUcastPkts
        Counter,
ifOutNUcastPkts
        Counter,
ifOutDiscards

        Counter,
ifOutErrors
        Counter,
ifOutQLen
        Gauge,
ifSpecific
        OBJECT IDENTIFIER
}

ifIndex OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
```

```

STATUS mandatory
DESCRIPTION
    "Um valor único para cada interface, que varia entre
    1 e o valor de ifNumber. O valor de cada interface
    deve permanecer constante pelo menos a partir de uma
    reinicialização do sistema de gerenciamento de rede
    da entidade até a reinicialização seguinte."
::= { ifEntry 1 }

ifDescr OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..255))
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Uma string de texto contendo informações sobre a
        interface. Essa string deve incluir o nome do
        fabricante, o nome do produto e a versão
        da interface do hardware."
::= { ifEntry 2 }

END

```

A primeira linha desse arquivo define o nome da MIB, nesse caso RFC1213-MIB. (A RFC 1213 é a que define a MIB-II; algumas das MIBs citadas são definidas por RFCs). O formato dessa definição é sempre o mesmo. A seção IMPORTS da MIB é citada ocasionalmente como a seção de ligação, e permite importar tipos de dados e OIDs de outros arquivos de MIB, por meio da cláusula IMPORTS. Essa MIB importa os seguintes itens da RFC1155-SMI (A RFC 1155 define a SMIv1, discutida anteriormente neste capítulo):

- mgmt
- NetworkAddress
- IpAddress
- Counter
- Gauge
- TimeTicks

Também é importante o OBJECT-TYPE da RFC 1212, a Concise MIB Definition, que define como os arquivos de MIB são escritos. Cada grupo de itens importados por meio da cláusula IMPORTS usa uma cláusula FROM para definir o arquivo de MIB em que os objetos são obtidos

As OIDs utilizadas no restante da MIB vêm depois da seção de ligação. Esse grupo de linhas configura o nível superior da subárvore *mib-2*, definida como *mgmt* seguida por .1. Vimos anteriormente que *mgmt* equivalia a 1.3.6.1.2. Por conseguinte, a *mib-2* é equivalente a 1.3.6.1.2.1. De modo semelhante, o grupo *interfaces* em *mib-2* é definido como { *mib-2* 2 } ou como 1.3.6.1.2.1.2.

Após a definição das OIDs, chegamos às definições dos objetos reais. Toda definição de objeto tem o seguinte formato:

```
<name> OBJECT-TYPE
  SYNTAX <datatype>
  ACCESS <read-only, read-write, write-only ou not-accessible>
  STATUS <mandatory, optional ou obsolete>
  DESCRIPTION
    "Descrição em texto explicando esse objeto gerenciado específico."
 ::= { <OID exclusiva que define este objeto> }
```

O primeiro objeto gerenciado no subconjunto da definição da MIB-II é *ifTable*, que representa uma tabela de interfaces de rede em um dispositivo gerenciado (observe que os nomes dos objetos são definidos com maiúsculas/minúsculas, com a primeira letra minúscula). Eis a definição desse objeto com a notação do ASN.1:

```
ifTable OBJECT-TYPE
  SYNTAX SEQUENCE OF IfEntry
  ACCESS not-accessible
  STATUS mandatory
  DESCRIPTION
    "Uma lista de entradas de interface. O número de entradas é
determinado
    pelo valor de ifNumber."
 ::= { interfaces 2 }
```

A SYNTAX de *ifTable* é SEQUENCE OF *IfEntry*. Isso significa que *ifTable* é uma tabela contendo as colunas definidas em *IfEntry*. O objeto está not-accessible, o que indica que não é possível consultar em um agente o valor desse objeto. O status é mandatory, indicando que um agente deve implementar este objeto de acordo com a especificação da MIB-II. A DESCRIPTION descreve este objeto com exatidão. A OID única é 1.3.6.1.2.1.2.2 ou *iso.org.dod.internet.mgmt.interfaces.2*.

Examinemos agora a definição de SEQUENCE do arquivo de MIB apresentada anteriormente nesta seção, usada com a SEQUENCE OF tipo na definição de *ifTable*:

```
IfEntry ::=
  SEQUENCE {
    ifIndex
      INTEGER,
    ifDescr
      DisplayString,
    ifType
      INTEGER,
    ifMtu
      INTEGER,
```

```
    .  
    .  
    .  
ifSpecific  
    OBJECT IDENTIFIER  
}
```

O nome da seqüência (*IfEntry*) é formado por maiúsculas/minúsculas, mas a primeira letra é maiúscula, ao contrário da definição de objeto em *ifTable*. É assim que um nome de seqüência é definido. Uma seqüência é apenas uma lista de colunas de objetos e os respectivos tipos de dados da SMI, que define uma tabela de conceitos. Nesse caso, esperamos encontrar variáveis definidas por *ifIndex*, *ifDescr*, *ifType* etc. Essa tabela pode conter qualquer número de linhas; o agente se encarrega de gerenciar as linhas residentes na tabela. A NMS pode adicionar linhas à tabela. Esta operação será discutida mais adiante, na seção “Operação de set”.

Uma vez que já existe a *IfEntry* para especificar o que encontraremos em qualquer linha da tabela, podemos reexaminar a definição da própria *ifEntry* (as linhas reais da tabela):

```
ifEntry OBJECT-TYPE  
    SYNTAX  IfEntry  
    ACCESS  not-accessible  
    STATUS  mandatory  
    DESCRIPTION  
        "Uma entrada de interface contendo objetos da camada da sub-rede  
        e abaixo para uma interface específica."  
    INDEX  { ifIndex }  
    ::= { ifTable 1 }
```

*ifEntry* define uma linha específica em *ifTable*. A definição é quase idêntica à de *ifTable*, exceto pelo fato de termos inserido uma nova cláusula, INDEX. O índice é uma chave exclusiva, usada para definir uma linha individual em *ifTable*. O agente verifica se o índice é único no contexto da tabela. Se um roteador tiver seis interfaces, *ifTable* terá seis linhas. A OID de *ifEntry* é 1.3.6.1.2.1.2.2.1 ou iso.org.dod.internet.mgmt.interfaces.ifTable.ifEntry. O índice de *ifEntry* é *ifIndex*, definido como:

```
ifIndex OBJECT-TYPE  
    SYNTAX  INTEGER  
    ACCESS  read-only  
    STATUS  mandatory  
    DESCRIPTION  
        "Um valor único para cada interface, que varia entre 1 e  
        o valor de ifNumber. O valor de cada interface deve permanecer  
        constante pelo menos de uma reinicialização do sistema de  
        gerenciamento de rede da entidade até a reinicialização  
        seguinte."  
    ::= { ifEntry 1 }
```

O objeto *ifIndex* é *read-only*, o que significa que podemos ver seu valor, mas não modificá-lo. O último objeto definido por nossa MIB é *ifDescr*, que é uma descrição textual da interface representada por essa linha específica em *ifTable*. O exemplo de nossa MIB termina com a cláusula *END*, que marca o final da MIB. Nos arquivos da verdadeira MIB-II, cada objeto listado na seqüência de *IfEntry* tem uma definição de objeto exclusiva. Nesta versão da MIB, listamos apenas duas delas, por questão de economia de espaço.

## Extensões para a SMI na Versão 2

A SMIv2 estende a árvore de objetos da SMI ao adicionar a ramificação *snmpV2* à subárvore *internet*, ao incluir diversos tipos de dados novos e ao efetuar algumas outras alterações. A Figura 2-3 mostra como os objetos de *snmpV2* se enquadram no cenário maior; a OID dessa nova ramificação é 1.3.6.1.6.3.1.1 ou *iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBObjects*. A SMIv2 também define alguns tipos de dados novos, resumidos na Tabela 2-2.

Tabela 2-2 Novos tipos de dados para a SMIv2

Tipo de dado	Descrição
<i>Integer32</i>	Idêntico a INTEGER.
<i>Counter32</i>	Idêntico a Counter.
<i>Gauge32</i>	Idêntico a Gauge.
<i>Unsigned32</i>	Representa valores decimais no intervalo de 0 a $2^{32} - 1$ , inclusive.
<i>Counter64</i>	Semelhante a Counter32, mas o valor máximo é 18.446.744.073.709.551.615. Counter64 é perfeito para as situações em que um Counter32 pode retornar a 0 em pouco tempo.
<i>BITS</i>	Uma lista de bits não negativos.

A definição de um objeto na SMIv2 mudou um pouco em relação à SMIv1. Existem alguns campos opcionais novos, o que concede mais controle sobre o modo de acesso a um objeto, permitindo aumentar uma tabela com mais colunas e apresentar descrições mais adequadas. Eis a sintaxe de uma definição de objeto para a SMIv2. As partes modificadas estão em negrito:

```
<name> OBJECT-TYPE  
  SYNTAX <datatype>  
  UnitsParts <Optional, vide abaixo>  
  MAX-ACCESS <Vide abaixo>  
  STATUS <Vide abaixo>  
  DESCRIPTION
```

"Descrição em texto explicando este objeto gerenciado específico."  
**AUGMENTS { <nome da tabela> }**  
**::= { <OID única que define este objeto> }**

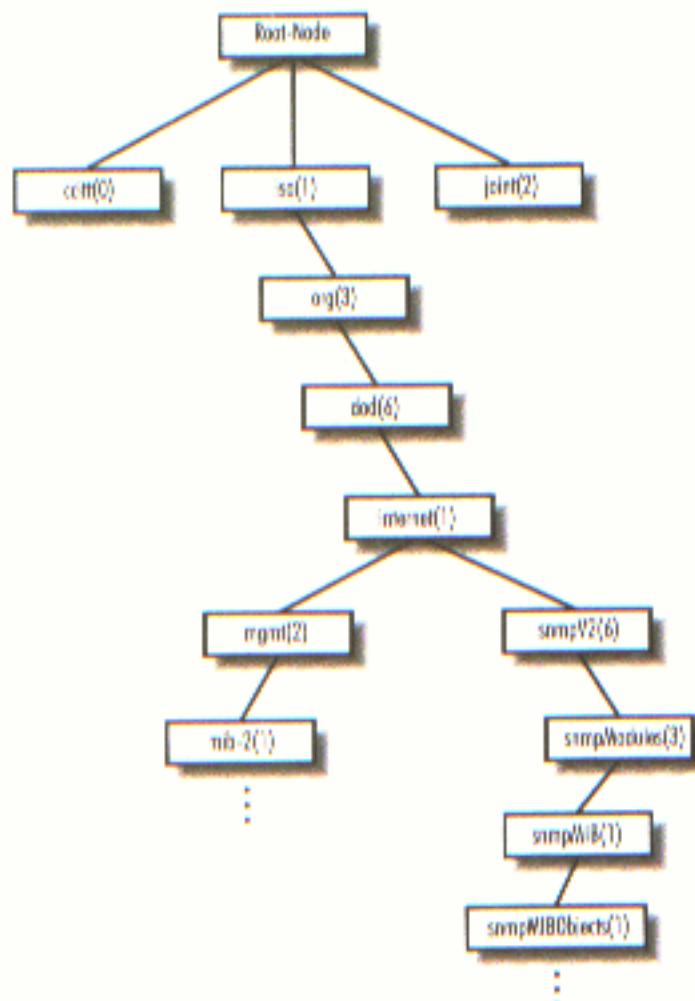


Figura 2-3 Árvore de registros da SMIv2 para o SNMPv2

A Tabela 2-3 descreve resumidamente os aperfeiçoamentos introduzidos na definição de objeto efetuados na SMIv2.

Tabela 2-3 Aprimoramentos na Definição de Objeto da SMIv2

Aprimoramento na Definição de Objeto	Descrição
UnitsParts	Uma descrição em texto das unidades (por exemplo, segundos, milissegundos etc.) suadas para representar o objeto.
MAX-ACCESS	O ACCESS de um OBJECT-TYPE pode ser MAX-ACCESS no SNMPv2. As opções válidas para MAX-ACCESS são read-only, read-write, read-create, not-accessible e accessible-for-notify.
STATUS	Esta cláusula foi estendida para aceitar as palavras-chave current, obsolete e deprecated. current no SNMPv2 equivale a mandatory em uma MIB do SNMPv1.

Tabela 2-3 Continuação

Aprimoramento na Definição de Objeto	Descrição
AUGMENTS	Em alguns casos, é útil adicionar uma coluna a uma tabela já existente. A cláusula AUGMENTS permite estender uma tabela, adicionando uma ou mais colunas, representadas por alguns outros objetos. Essa cláusula requer o nome da tabela em que o objeto será incluído.

SMIv2 define um novo tipo de trap denominado NOTIFICATION-TYPE, que discutiremos mais adiante, na seção “Notificação do SNMP”. A SMIv2 também apresenta novas convenções de texto que permitem a criação de objetos gerenciados por meios mais abstratos. A RFC 2579 define as convenções de texto usadas no SNMPv2, listadas na Tabela 2-4.

Tabela 2-4 Convenções de texto para a SMIv2

Convenção	Descrição
DisplayString	Uma seqüência de caracteres ASCII do NVT. Uma DisplayString não pode ter mais de 255 caracteres.
PhysAddress	Um endereço no nível físico ou de mídia, representado como uma OCTET STRING.
MacAddress	Define o endereço de acesso à mídia para a IEEE 802 (o padrão para as redes de área local) em ordem canônica <sup>a</sup> . (Na linguagem cotidiana, significa o endereço de Ethernet.) Esse endereço é representado com seis octetos.
TruthValue	Define os valores Booleanos, true e false.
TestAndIncr	Usada para impedir que duas estações de gerenciamento modifiquem o mesmo objeto gerenciado, simultaneamente.
AutonomousType	Uma OID utilizada para definir uma subárvore com definições adicionais relacionadas à MIB.
VariablePointer	Um ponteiro para a instância de um objeto específico, como a ifDescr da interface 3. Nesse caso, VariablePointer seria a OID ifDescr.3.
RowPointer	Um ponteiro para uma linha em tabela. Por exemplo, ifIndex.3 indica a terceira linha na ifTable.
RowStatus	Usada para gerenciar a criação e eliminação de linhas de uma tabela, uma vez que o SNMP não pode fazer isso por meio do próprio protocolo. RowStatus pode rastrear o estado de uma linha de uma tabela, assim como receber comandos para a criação e exclusão de linhas. Essa convenção de texto é elaborada para preservar a integridade da tabela quando mais de um gerenciador estiver atualizando linhas. Os tipos numerados a seguir definem os comandos e as variáveis de estado: active(1), notInService(2), notReady(3), createAndGo(4), createAndWait(5) e destroy(6).

Tabela 2-4 Continuação

Convenção	Descrição
TimeStamp	Mede o tempo decorrido entre o tempo de funcionamento do sistema de um dispositivo e algum evento ou ocorrência.
TimeInterval	Mede um intervalo de tempo em centésimos de segundo. TimeInterval pode usar qualquer valor inteiro, de 0-2147483647.
DateAndTime	Uma OCTET STRING usada para representar informações de data e hora.
StorageType	Define o tipo de memória que um agente usa. Os possíveis valores são other(1), volatile(2), nonVolatile(3), permanent(4) e readOnly(5).
TDomain	Denota um tipo de serviço de transporte.
TAddress	Denota o endereço do serviço de transporte. TAddress é definido de 1-255 octetos de comprimento.

\* Ordem canônica significa que o endereço deve ser representado com o bit menos significativo em primeiro lugar.

## Exame minucioso da MIB-II

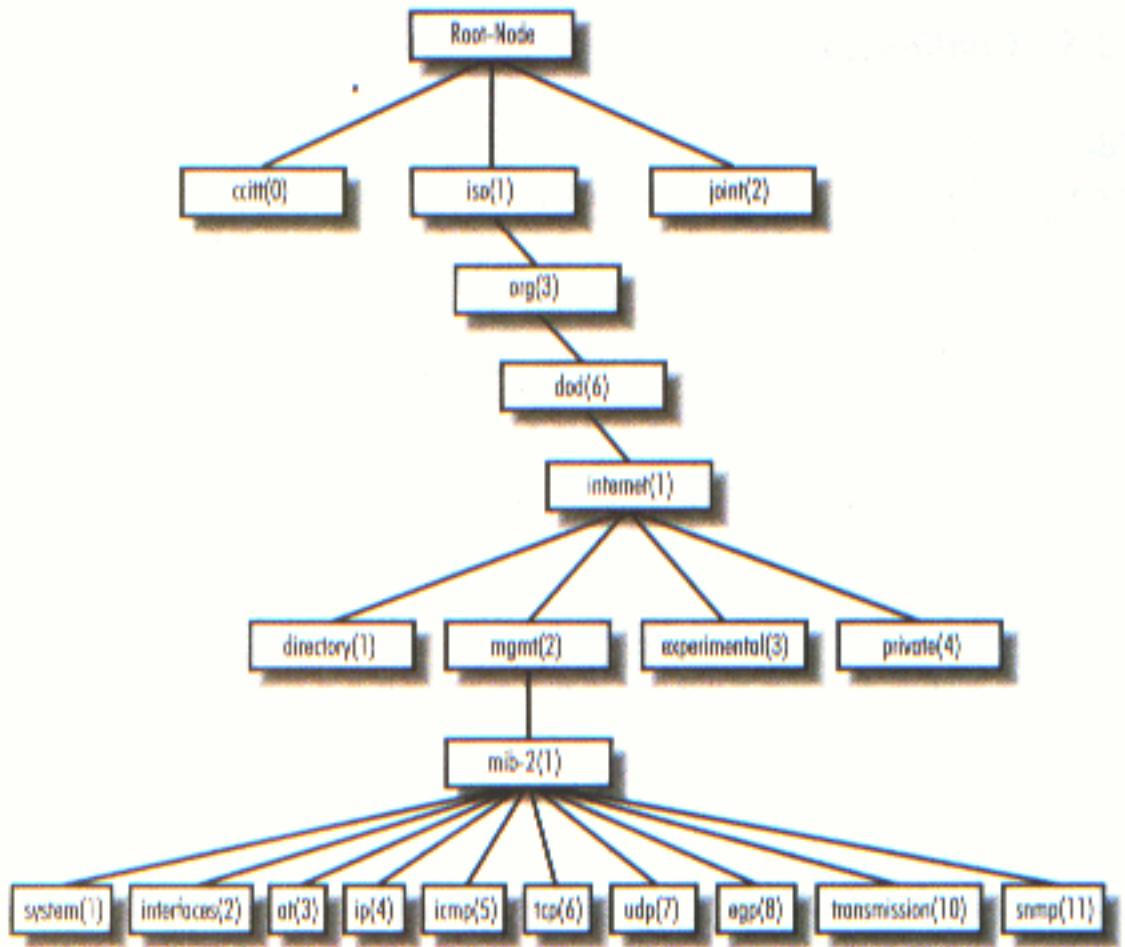
A MIB-II é um grupo de gerenciamento muito importante, porque cada dispositivo com suporte para o SNMP também deve aceitar a MIB-II. Por conseguinte, usaremos objetos da MIB-II nos exemplos deste livro. Não descreveremos com detalhes cada objeto na MIB; definiremos apenas as subárvore. A seção da RFC1213-MIB que define as OIDs básicas da subárvore *mib-2* é parecida com esta:

```

mib-2      OBJECT IDENTIFIER ::= { mgmt 1 }
system     OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces OBJECT IDENTIFIER ::= { mib-2 2 }
at         OBJECT IDENTIFIER ::= { mib-2 3 }
ip         OBJECT IDENTIFIER ::= { mib-2 4 }
icmp       OBJECT IDENTIFIER ::= { mib-2 5 }
tcp        OBJECT IDENTIFIER ::= { mib-2 6 }
udp        OBJECT IDENTIFIER ::= { mib-2 7 }
egp        OBJECT IDENTIFIER ::= { mib-2 8 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp       OBJECT IDENTIFIER ::= { mib-2 11 }

```

A *mib-2* é definida como *iso.org.dod.internet.mgmt.1* ou *1.3.6.1.2.1*. A partir daqui, é possível ver que o grupo *system* é *mib-2 1* ou *1.3.6.1.2.1.1*, e assim por diante. A Figura 2-4 mostra a subárvore MIB-II da ramificação *mgmt*.



*Figura 2-4 Subárvore MIB-II*

A Tabela 2-5 descreve resumidamente cada grupo de gerenciamento definido na MIB-II. Não explicaremos com detalhes cada grupo porque você mesmo pode obter a RFC 1213 e ler a MIB.

*Tabela 2-5 Descrição sucinta dos grupos da MIB-II*

Nome da subárvore	OID	Descrição
<i>system</i>	<i>1.3.6.1.2.1.1</i>	Define uma lista de objetos pertencentes à operação do sistema, como o tempo de funcionamento, contato e nome do sistema.
<i>interfaces</i>	<i>1.3.6.1.2.1.2</i>	Rastreia o status de cada interface em uma entidade gerenciada. O grupo <i>interfaces</i> monitora as interfaces em funcionamento ou inativas e rastreia aspectos, como octetos enviados e recebidos, erros e eliminações etc.
<i>at</i>	<i>1.3.6.1.2.1.3</i>	O grupo <i>address translation (at)</i> é fornecido somente para manter a compatibilidade com versões anteriores e, provavelmente, será retirado da MIB-III.
<i>ip</i>	<i>1.3.6.1.2.1.4</i>	Rastreia os diversos aspectos do IP, incluindo o roteamento do IP.

Tabela 2-5 Continuação

Nome da subárvore	OID	Descrição
<i>icmp</i>	1.3.6.1.2.1.5	Rastreia aspectos como erros do ICMP, exclusões etc.
<i>tcp</i>	1.3.6.1.2.1.6	Rastreia, entre outros aspectos, o estado das conexões do TCP (como <i>closed</i> , <i>listen</i> , <i>synSent</i> etc.).
<i>udp</i>	1.3.6.1.2.1.7	Rastreia dados estatísticos do UDP, datagramas internos e externos etc.
<i>egp</i>	1.3.6.1.2.1.8	Rastreia diversos dados estatísticos sobre o EGP e mantém uma tabela de vizinhos do EGP.
<i>transmission</i>	1.3.6.1.2.1.10	Não existem atualmente objetos definidos para este grupo, mas outras MIBs específicas de mídia são definidas por meio desta subárvore.
<i>snmp</i>	1.3.6.1.2.1.11	Avalie o desempenho da implementação básica do SNMP na entidade gerenciada e rastreia aspectos, como o número de pacotes de SNMP enviados e recebidos.

## Operações do SNMP

Discutimos o modo como o SNMP organiza informações, mas ainda não abordamos como realmente podemos obter informações de gerenciamento. Agora, examinaremos o funcionamento efetivo do SNMP.

*Protocol Data Unit* (PDU) é o formato de mensagem que os gerenciadores e agentes utilizam para enviar e receber informações. Existe um formato PDU padrão para cada uma das seguintes operações do SNMP:

- *get*
- *get-next*
- *get-bulk* (SNMPv2 e SNMPv3)
- *set*
- *get-response*
- *trap*
- *notification* (SNMPv2 e SNMPv3)
- *inform* (SNMPv2 e SNMPv3)
- *report* (SNMPv2 e SNMPv3)

Examinemos cada uma dessas operações.

## Operação de get

A solicitação de *get* é iniciada pela NMS, que a envia para o agente. O agente recebe a solicitação e a processa para obter o máximo proveito. Alguns dispositivos com carga excessiva, como os roteadores, talvez não consigam responder à solicitação e precisarão excluí-la. Se o agente conseguir obter as informações solicitadas, retornará um *get-response* para a NMS, onde é processado. Esse processo é ilustrado na Figura 2-5.

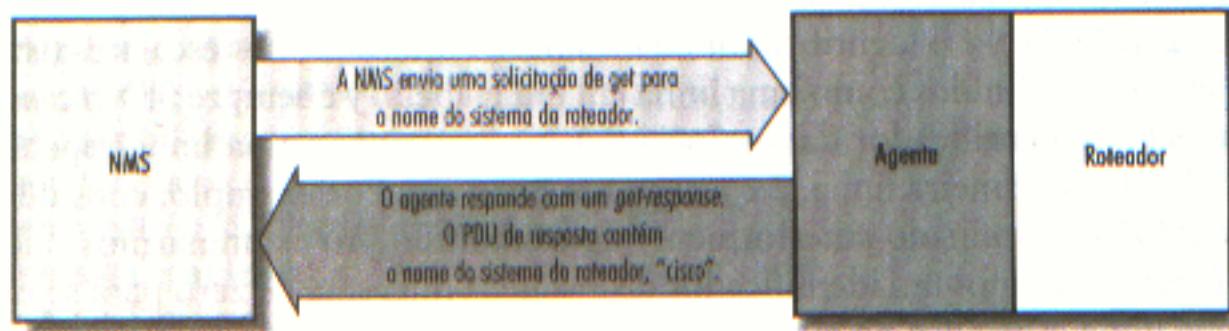


Figura 2-5 Seqüência de solicitações de *get*

Como o agente sabe o que a NMS estava procurando? Um dos itens na solicitação do *get* é uma *vinculação de variáveis*, ou varbind, que é uma lista de objetos da MIB que permite que o receptor de uma solicitação veja o que o emissor deseja saber. As vinculações de variáveis podem ser consideradas como os pares *OID=valor* que facilitam para o emissor (a NMS, nesse caso) selecionar as informações necessárias quando o receptor preencher a solicitação e retornar a resposta. Vejamos essa operação em ação:

```
$ snmpget cisco.ora.com public .1.3.6.1.2.1.1.6.0  
system.sysLocation.0 = ""
```



Todos os comandos Unix apresentados neste capítulo procedem do pacote do agente do Net-SNMP (anteriormente, o projeto UCD-SNMP), um agente do Unix e Windows NT disponível gratuitamente. O Capítulo 5 informa um URL em que é possível fazer o download do pacote. Os comandos desse pacote estão resumidos no Apêndice C.

Neste exemplo, existem várias ocorrências. Primeiro, estamos executando um comando em um host Unix. O comando é chamado *snmpget* e sua finalidade principal é facilitar a obtenção de dados de gerenciamento por meio de uma solicitação de *get*. Estamos especificando três argumentos na linha de comando: o nome do dispositivo a ser consultado (*cisco.ora.com*), a string da comunidade *read-only* (*public*) e a OID a ser obtida (*.1.3.6.1.2.1.1.6.0*). Se reexaminarmos a Tabela 2-5, veremos que *1.3.6.1.2.1.1* é o grupo *system*, mas existem mais dois inteiros no final da OID: *.6* e *.0*. Na realidade, *.6* é a variá-

vel da MIB que desejamos consultar; o nome legível pelo ser humano é *sysLocation*. Nesse caso, gostaríamos de ver a localização do sistema definida no roteador Cisco. Como é possível constatar pela resposta (*system.sysLocation.0 = “”*), no momento, a localização do sistema nesse roteador não está definida. Além disso, observe que a resposta de *snmpget* está no formato de vinculação de variáveis, *OID=valor*.

Existe mais um aspecto a ser analisado. Por que a variável da MIB tem *.0* inserido no final? No SNMP, os objetos da MIB são definidos pela convenção *x.y*, onde *x* é a verdadeira OID do objeto gerenciado (em nosso exemplo, *1.3.6.1.2.1.1.6*) e *y* é o identificador da instância. Para os objetos escalares (isto é, os objetos não definidos como uma linha em uma tabela) *y* é sempre 0. No caso de uma tabela, o identificador da instância permite selecionar uma linha específica da tabela; 1 é a primeira linha, 2 é a segunda linha etc. Por exemplo, considere o objeto *ifTable* examinado anteriormente neste capítulo. Ao examinarmos valores em *ifTable*, usariammos um identificador de instância diferente de zero para selecionar uma linha específica na tabela (neste caso, uma interface de rede específica).



Os aplicativos de NMS gráfica, que abrangem a maioria dos pacotes comerciais, não usam programas de linha de comando para recuperar informações de gerenciamento. Usamos esses comandos para dar uma idéia de como funcionam os comandos de recuperação e o que eles costumam retornar. As informações que uma NMS gráfica recupera e o respectivo processo de recuperação são idênticos a esses programas de linha de comando; a NMS apenas permite formular consultas e exibe os resultados por meio de uma GUI mais adequada.

O comando *get* serve para recuperar um único objeto da MIB de cada vez. Entretanto, tentar gerenciar dessa maneira pode ser uma perda de tempo. Nesse momento, o comando *get-next* entra em cena e permite recuperar mais de um objeto de um dispositivo em um intervalo de tempo.

### *Operação de get-next*

A operação de *get-next* permite emitir uma seqüência de comandos para recuperar um grupo de valores de uma MIB. Em outras palavras, para cada objeto MIB a ser recuperado, são gerados separadamente uma solicitação de *get-next* e um *get-response*.

O comando *get-next* atravessa uma subárvore em ordem lexicográfica. Como uma OID é uma seqüência de inteiros, é fácil para um agente iniciar na raiz da árvore de objetos da respectiva SMI e descer até encontrar a OID que está procurando. Quando a NMS receber uma resposta do agente ao comando *get-next* recém emitido, ela enviará outro comando *get-next* e continuará repetindo esse comando até que o agente retorne um erro, significando que o final da MIB foi alcançado e não há mais objetos a serem obtidos.

Se examinarmos outro exemplo, constataremos esse comportamento em ação. Dessa vez, usaremos um comando denominado *snmpwalk*, que apenas facilita o procedimento *get-next*. Esse comando é chamado como o comando *snmpget*, exceto pelo fato de que agora especificamos a ramificação na qual iniciar (nesse caso, o grupo *system*):

```
$ snmpwalk cisco.ora.com public system
system.sysDescr.0 = "Cisco Internetwork Operating System Software
..IOS (tm) 2500 Software (C2500-I-L), Version 11.2(5), RELEASE
SOFTWARE (fc1)..Copyright (c) 1986-1997 by cisco Systems, Inc...
Compiled Mon 31-Mar-97 19:53 by ckralik"
system.sysObjectID.0 = OID: enterprises.9.1.19
system.sysUpTime.0 = Timeticks: (27210723) 3 days, 3:35:07.23
system.sysContact.0 = ""
system.sysName.0 = "cisco.ora.com"
system.sysLocation.0 = ""
system.sysServices.0 = 6
```

A seqüência *get-next* retorna sete variáveis de MIB. Cada um desses objetos faz parte do grupo *system* conforme definido na RFC 1213. Vemos a ID do objeto sistema, o tempo de funcionamento do sistema, a pessoa de contato etc.

Depois que você examina alguns objetos, como *get-next* sabe qual objeto deve examinar em seguida? *get-next* baseia-se no conceito da ordenação lexicográfica da árvore de objetos da MIB. Essa ordem é mais simples porque cada nó na árvore recebe a atribuição de um número. Para entender o que isso significa, iniciemos na raiz da árvore e desçamos até o nó do sistema.

Para alcançar o grupo *system* (OID 1.3.6.1.2.1.1), iniciamos na raiz da árvore de objetos e desemos. A Figura 2-6 mostra a progressão lógica da raiz da árvore até o grupo *system*. Em cada nó na árvore, visitamos a ramificação do menor número. Assim, quando estivermos no nó da raiz, começaremos a visitar o *ccitt*. Esse nó não possui nós subordinados, e passamos para o nó *iso*. Como o nó *iso* tem um nó filho, passamos para esse nó, *org*. O processo continua até alcançarmos o nó *system*. Como cada ramificação é formada por inteiros ascendentes (*ccitt(0)* *iso(1)* *join(2)*, por exemplo), o agente não enfrenta problemas para atravessar essa estrutura em árvore até alcançar o grupo *system(1)*. Se precisássemos continuar esse percurso, iríamos até o *system.1* (*system.sysLocation*), *system.2* e os outros objetos no grupo *system*. Em seguida, iríamos até *interfaces(2)* e assim por diante.

## Operação de *get-bulk*

O SNMPv2 define a operação de *get-bulk*, que permite que um aplicativo de gerenciamento recupere uma grande seção de uma tabela, de uma só vez. A operação do *get* padrão pode tentar recuperar mais de um objeto MIB de uma vez, mas os tamanhos de mensagens são limitados pelos recursos do agente. Se o agente não puder retornar todas as respostas solicitadas, retornará uma mensagem de erro sem dados. A operação de *get-bulk*, por outro lado, instrui o agente

a retornar o máximo da resposta possível. Isso significa que são possíveis respostas incompletas. Ao emitir um comando *get-bulk*, é necessário definir dois campos: “nonrepeaters” e “max-repetitions”. O campo “nonrepeaters” informa ao comando *get-bulk* que é possível recuperar os primeiros N objetos com uma simples operação de *get-next*. O campo “max-repetitions” instrui o comando *get-bulk* a tentar até M operações *get-next* para recuperar os demais objetos. A Figura 2-7 mostra a seqüência do comando *get-bulk*.

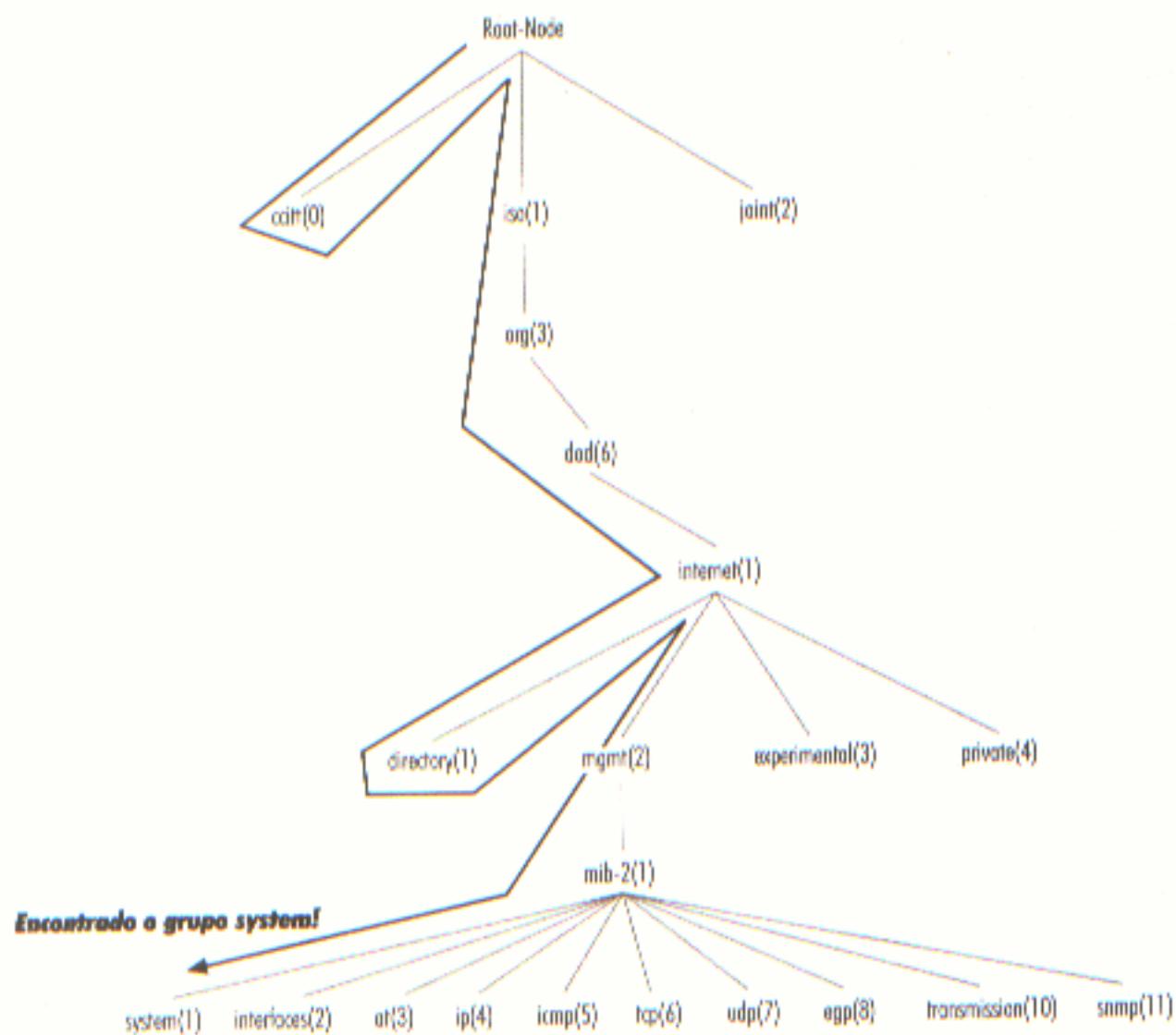


Figura 2-6 Árvore MIB

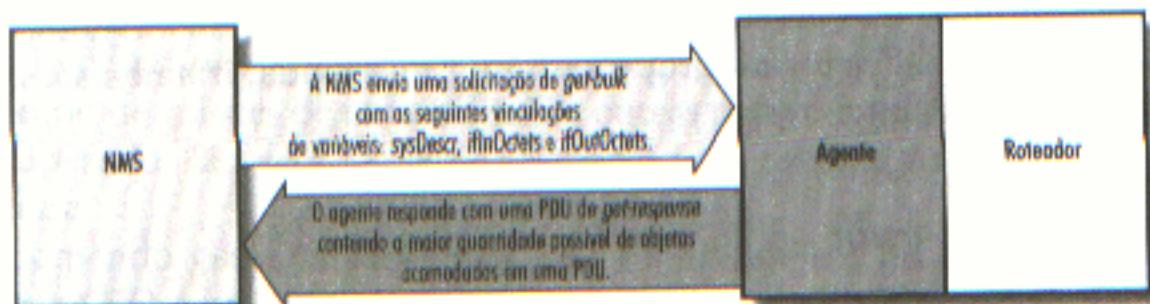


Figura 2-7 Seqüência de solicitações de *get-bulk*

Na Figura 2-7, estamos solicitando três vinculações: *sysDescr*, *ifInOctets* e *ifOutOctets*. O número total de vinculações de variáveis solicitadas é determi-

nado pela fórmula  $N + (M * R)$ , onde  $N$  é o número de “nonrepeaters” (isto é, objetos escalares na solicitação – nesse caso 1, porque *sysDescr* é o único objeto escalar),  $M$  é o máximo de repetições (nesse caso, definimos aleatoriamente o nº 3) e  $R$  é o número de objetos não-escalares na solicitação (nesse caso, 2, porque *ifInOctets* e *ifOutOctets* são não-escalares). Usando os números deste exemplo, temos  $1 + (3 * 2) = 7$ , que é o número total de vinculações de variáveis que pode ser retornado por essa solicitação de *get-bulk*.

O pacote do Net-SNMP dispõe de um comando para emitir consultas de *get-bulk*. Se executarmos esse comando usando todos os parâmetros discutidos anteriormente, ele ficará assim:

```
$ snmpbulkget -v2c -B 1 3 linux.ora.com public sysDescr ifInOctets
ifOutOctets
system.sysDescr.0 = "Linux Linux 2.2.5-15 #3 Thu May 27 19:33:18 EDT 1999
i686"
interfaces.ifTable.ifEntry.ifInOctets.1 = 70840
interfaces.ifTable.ifEntry.ifOutOctets.1 = 70840
interfaces.ifTable.ifEntry.ifInOctets.2 = 143548020
interfaces.ifTable.ifEntry.ifOutOctets.2 = 111725152
interfaces.ifTable.ifEntry.ifInOctets.3 = 0
interfaces.ifTable.ifEntry.ifOutOctets.3 = 0
```

Como *get-bulk* é um comando do SNMPv2, é necessário instruir o *snmpgetbulk* a utilizar uma PDU do SNMPv2 com a opção *-v2c*. Os “nonrepeaters” e “max-repetitions” são definidos com a opção *-B 1 3*, que define os “nonrepeaters” com 1 e “max-repetitions” com 3. Observe que o comando retornou sete vinculações de variáveis: uma para *sysDescr* e três para cada *ifInOctets* e *ifOutOctets*.

## Operação de *set*

O comando *set* é utilizado para modificar o valor de um objeto gerenciado ou para criar uma nova linha em uma tabela. Os objetos definidos MIB como *read-write* ou *write-only* podem ser alterados ou criados com esse comando. Uma NMS pode definir mais de um objeto de cada vez.

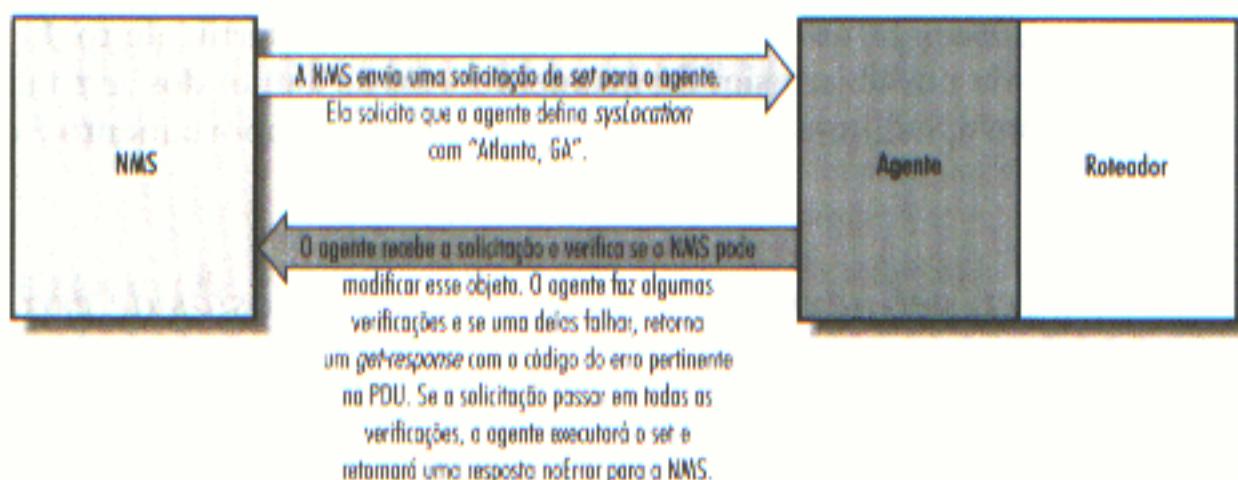


Figura 2-8 Seqüência de solicitações de *set*

A Figura 2-8 mostra a seqüência de solicitações de *set*; esse comando é semelhante aos outros que vimos até agora mas, na realidade, o comando está modificando algo na configuração do dispositivo, em vez de apenas recuperar uma resposta a consulta. Se examinarmos um exemplo de um *set* real, você verá o comando em ação. O exemplo a seguir consulta a variável *sysLocation*, depois define-a com um valor:

```
$ snmpget cisco.ora.com public system.sysLocation.0  
system.sysLocation.0 = ""  
$ snmpset cisco.ora.com private system.sysLocation.0 s "Atlanta, GA"  
system.sysLocation.0 = "Atlanta, GA"  
$ snmpget cisco.ora.com public system.sysLocation.0  
system.sysLocation.0 = "Atlanta, GA"
```

O primeiro comando é o conhecido *get*, que exibe o valor atual de *sysLocation*. Em um dos exemplos anteriores, vimos que ele estava sem definição; isso ainda ocorre. O segundo comando é o *snmpset*. Para esse comando, fornecemos o nome do host, a string de comunidade *read-write* (*private*) e a variável a ser definida (*system.sysLocation.0*), juntamente com o novo valor (*s "Atlanta, GA"*). O *s* informa ao *snmpset* que *desejamos definir o valor de sysLocation com uma string*; e "Atlanta, GA" é o novo valor. Como descobrir se *sysLocation* *exige um valor de string*? A definição de *sysLocation* na RFC 1213 é parecida com a seguinte:

```
sysLocation OBJECT-TYPE  
    SYNTAX  DisplayString (SIZE (0..255))  
    ACCESS  read-write  
    STATUS   mandatory  
    DESCRIPTION  
        "A localização física deste nó (por exemplo, 'cabine telefônica,  
        3º andar')."  
    ::= { system 6 }
```

A SYNTAX para *sysLocation* é *DisplayString (SIZE (0..255))*, que significa que é uma string com, no máximo, 255 caracteres. O comando *snmpset* obtém êxito e informa o novo valor de *sysLocation*. Mas apenas para confirmar, executamos um último *snmpget*, que informa que o *set* realmente surtiu efeito. É possível definir mais de um objeto simultaneamente, mas se algum dos *sets* falhar, todos falharão (isto é, nenhum valor é modificado). Esse comportamento é o almejado.

### *Operação de get, get-next, get-bulk e set para respostas de erros*

As respostas de erros ajudam a confirmar se uma solicitação de *get* ou *set* foi processada corretamente pelo agente. As operações de *get*, *get-next* e *set* podem retornar as respostas de erro apresentadas na Tabela 2-6. O status de cada erro aparece entre parênteses.

Tabela 2-6 Mensagens de erro do SNMPv1

Mensagem de Erro do SNMPv1	Descrição
noError(0)	Solicitação executada sem problemas.
tooBig(1)	Resposta à solicitação muito grande para acomodar em uma única resposta.
noSuchName(2)	Um agente foi solicitado a obter ou definir uma OID não encontrada; por exemplo, a OID não existe.
badValue(3)	Objeto <i>read-write</i> ou <i>write-only</i> definido com valor inconsistente.
readOnly(4)	Erro geralmente não utilizado. Erro equivalente: <i>noSuchName</i> .
0	Este é um erro genérico. Se ocorrer um erro para o qual não existe uma mensagem anterior adequada, é emitido um erro <i>genError</i> .

As mensagens de erro do SNMPv1 não são muito abrangentes. Na tentativa de corrigir esse problema, o SNMPv2 define respostas de erro adicionais, válidas para as operações de *get*, *set*, *get-next* e *get-bulk*, desde que o agente e a NMS tenham suporte para o SNMPv2. Essas respostas adicionais estão listadas na Tabela 2-7.

Tabela 2-7 Mensagens de erro do SNMPv2

Mensagem de Erro do SNMPv2	Descrição
noAccess(6)	Foi emitido um <i>set</i> para uma variável inacessível. Isso geralmente ocorre quando a variável tem um tipo de ACCESS not-accessible.
wrongType(7)	Um objeto foi definido com um tipo diferente daquele especificado em sua definição. Esse erro ocorrerá se você tentar definir um objeto que é do tipo0 INTEGER como se fosse uma string, por exemplo.
wrongLength(8)	Objeto definido com um valor diferente daquele que ele exige. Por exemplo, é possível definir uma string com um tamanho máximo de caracteres. Esse erro ocorre se você tentar definir um objeto string com um valor que ultrapasse o tamanho máximo.
wrongEncoding(9)	Foi emitida uma operação de <i>set</i> usando uma codificação incorreta para o objeto sendo definido.
wrongValue(10)	Variável definida com um valor não reconhecido. Pode ocorrer quando um <i>read-write</i> está definido como um valor numerado, e você tenta defini-lo com um valor que não consta nos tipos numerados.

Tabela 2-7 Continuação

Mensagem de Erro do SNMPv2	Descrição
noCreation(11)	Você tentou definir uma variável não existente ou criar uma variável que não existe na MIB.
inconsistentValue	Uma variável da MIB encontra-se em um estado inconsistente e não aceita quaisquer solicitações de <i>set</i> .
resourceUnavailable(13)	Nenhum recurso do sistema disponível para executar um <i>set</i> .
commitFailed(14)	Erro genérico para definir falhas.
undoFailed(15)	Um <i>set</i> falhou e o agente não conseguiu anular todos os <i>sets</i> anteriores até o ponto da falha.
authorizationError(16)	Um comando do SNMP não pôde ser autenticado; em outras palavras, alguém forneceu uma string de comunidade incorreta.
notWritable(17)	Uma variável não aceitará um <i>set</i> , embora devesse.
inconsistentName(18)	Você tentou definir uma variável, mas essa tentativa falhou porque a variável se encontrava em um tipo de estado inconsistente.

## Traps do SNMP

Uma trap é um meio de um agente informar à NMS que aconteceu algo errado. Na seção “Gerenciadores e agentes” do Capítulo 1, examinamos o conceito de traps em termos gerais; agora, analisaremos as traps com mais detalhes. A Figura 2-9 mostra a seqüência da geração da trap.

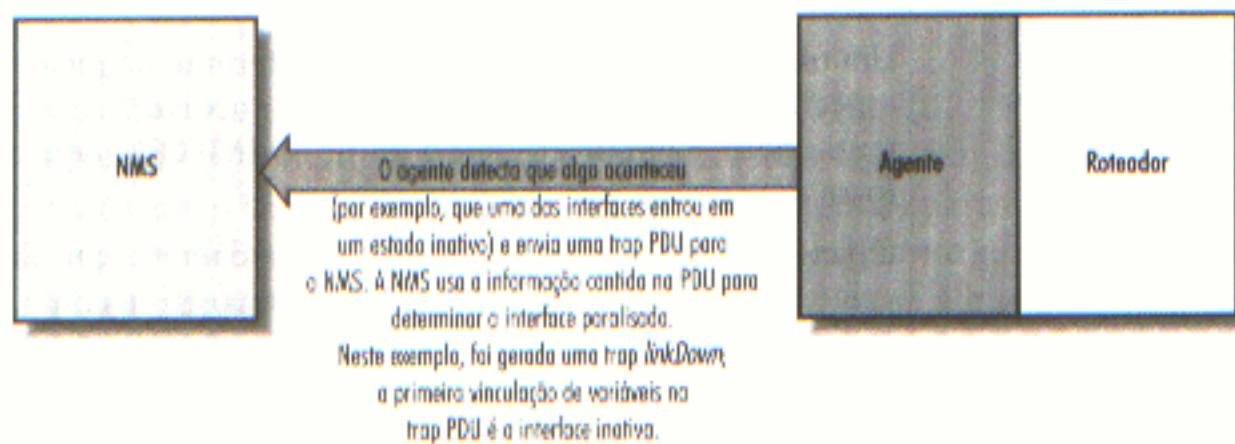


Figura 2-9 Geração de trap

A trap se origina no agente e é enviada para o destino configurado no próprio agente. Geralmente, o destino da trap é o endereço IP da NMS. Nenhuma confirmação é enviada da NMS para o agente, de modo que o agente não sabe se a trap alcançou a NMS. Como o SNMP usa o UDP e as traps são elaboradas para

informar os problemas ocorridos na rede, os traps estão propensas principalmente à perda e a não alcançar seus destinos.

Entretanto, o fato de que as traps podem desaparecer não as torna menos úteis; em um ambiente bem planejado, elas fazem parte do gerenciamento da rede. É mais recomendável que seu equipamento tente informar que algo está errado, mesmo que a mensagem nunca alcance você, do que simplesmente desistir e deixar você adivinhando o que pode ter acontecido. Eis algumas situações que uma trap pode relatar:

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

Quando uma NMS recebe uma trap, ela deve saber interpretar essa trap; isto é, a NMS deve saber o significado da trap e como interpretar as informações que a trap transmite. Uma trap é identificada pela primeira vez pelo número de trap genérica. Existem sete números de trap genérica (0-6), apresentados na Tabela 2-8. A trap genérica 6 é uma categoria especial de caça-tudo para as traps “específicas de empresa”, definidas pelos fornecedores ou usuários, classificadas fora das seis categorias de traps genéricas. As traps específicas de empresa são também identificadas por um ID de empresa (isto é, um ID de objeto existente em algum lugar na ramificação *enterprises* da árvore da MIB, iso.org.dod.internet.private.enterprises) e um número de trap específica selecionado pela empresa que definiu a trap. Assim, o ID de objeto de uma trap específica de empresa é o *ID da empresa.número da trap específica*. Por exemplo, ao definir traps especiais para suas MIBs privadas, a Cisco insere todas elas na árvore da MIBs específica da empresa(iso.org.dod.internet.private.enterprises.cisco). Como veremos no Capítulo 10, você pode definir suas próprias traps específicas da empresa; a única exigência é que você deve registrar o número de sua empresa junto à IANA.

Geralmente, uma trap é empacotada com informações. Como você previa, essas informações estão na forma de objetos MIB e seus valores; conforme mencionado anteriormente, esses pares de objeto-valor são conhecidos como vinculações de variáveis. Para as traps genéricas 0 a 5, o conteúdo da trap geralmente é incorporado ao software da NMS ou ao receptor da trap. As vinculações de variáveis contidas por uma trap específica de empresa são determinadas por quem definiu a trap. Por exemplo, se um modem em um rack de modem falhar, o agente do rack pode enviar uma trap para a NMS informando a falha. Muito provavelmente, a trap será específica de empresa, definida pelo fabricante do rack; o conteúdo da trap fica a critério do fabricante, mas, provavelmente, conterá informações suficientes para permitir identificar com exatidão o que falhou (Por exemplo, a posição da placa do modem no rack e o canal nessa placa).

Tabela 2-8 Traps genéricas

Nome e N° da Trap Genérica	Definição
<i>coldStart</i> (0)	Indica que o agente foi reinicializado. Todas as variáveis de gerenciamento serão redefinidas; especificamente, Counters e Gauges serão redefinidas com zero (0). Um aspecto positivo da trap <i>coldStart</i> é a possibilidade de usá-la para detectar quando um novo hardware for adicionado à rede. Quando um dispositivo é ligado, ele envia essa trap para seu destino de trap. Se o destino da trap estiver definido corretamente (por exemplo, para o endereço IP de sua NMS), a NMS poderá receber a trap e reconhecer se é necessário gerenciar o dispositivo.
<i>warmStart</i> (1)	Indica que o agente reinicializou a si mesmo. Nenhuma das variáveis de gerenciamento será redefinida.
<i>linkDown</i> (2)	Enviada quando uma interface em um dispositivo é paralisada. A primeira vinculação de variáveis identifica a interface inativa.
<i>linkUp</i> (3)	Enviada quando uma interface em um dispositivo é reativada. A primeira vinculação de variáveis identifica a interface reativada.
<i>authenticationFailure</i> (4)	Indica que alguém tentou consultar o agente com uma string de comunidade incorreta; útil para detectar se alguém estiver tentando obter acesso não autorizado a um dos dispositivos.
<i>egpNeighborLoss</i> (5)	Indica que um vizinho do <i>Exterior Gateway Protocol (EGP)</i> ficou inativo.
<i>enterpriseSpecific</i> (6)	Indica que a trap é específica de empresa. Os fornecedores e usuários do SNMP definem suas próprias traps na ramificação de empresa privada da árvore de objetos da SMI. Para processar essa trap corretamente, a NMS deve decodificar o número da trap específica que faz parte da mensagem do SNMP.

No Capítulo 1, mencionamos que a RFC 1697 é a MIB do RDBMS. Uma das traps definidas por essa MIB é a *rdbmsOutOfSpace*:

```
rdbmsOutOfSpace TRAP-TYPE
ENTERPRISE rdbmsTraps
VARIABLES { rdbmsSrvInfoDiskOutOfSpaces }
DESCRIPTION
```

"Uma trap *rdbmsOutOfSpace* significa que um dos servidores de banco de dados gerenciados por este agente não pode alocar espaço para um dos bancos de dados gerenciados por este agente. Tenha cuidado para não congestionar a rede com essas traps."

A empresa é *rdbmsTraps* e o nº da trap específica é 2. Essa trap tem uma vinculação de variáveis, *rdbmsSrvInfoDiskOutOfSpaces*. Se examinarmos outros locais na MIB, descobriremos que essa variável é um objeto escalar. Sua definição é:

```
rdbmsSrvInfoDiskOutOfSpaces OBJECT-TYPE
  SYNTAX Counter
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "Nº total de tentativas sem êxito do servidor no sentido de obter
     o espaço de disco necessário, desde a inicialização do servidor.
     Esse aspecto seria verificado por um agente ao receber uma trap
     rdbmsOutOfSpace."
 ::= { rdbmsSrvInfoEntry 9 }
```

A DESCRIPTION deste objeto indica por que a observação sobre ter cuidado para não congestionar a rede (no texto de DESCRIPTION para o TRAP-TYPE) é tão importante. Sempre que o RDBMS não pode alocar espaço para o banco de dados, o agente enviar uma trap. Um banco de dados ocupado (e cheio) poderia terminar enviando essa trap milhares de vezes por dia.

Alguns fornecedores de RDBMS no mercado, como o Oracle, fornecem um agente de SNMP com os respectivos mecanismos de banco de dados. Geralmente, agentes dessa natureza dispõem de uma funcionalidade muito além da fornecida pela MIB do RDBMS.

## Notificação do SNMP

Em um esforço de padronizar o formato PDU das traps do SNMPv1 (lembre-se de que as traps do SNMPv1 têm um formato de PDU diferente de get e set), o SNMPv2 define um NOTIFICATION-TYPE. O formato de PDU para o NOTIFICATION-TYPE é idêntico ao do get e set. A RFC 2863 redefine o tipo de notificação genérica, *linkDown*, assim:

```
linkDown NOTIFICATION-TYPE
  OBJECTS { ifIndex, ifAdminStatus, ifOperStatus }
  STATUS current
  DESCRIPTION
    "Uma trap linkDown significa que a entidade do SNMPv2, atuando como
     um agente, detectou que o objeto ifOperStatus object de um de seus
     links de comunicação saiu do estado inativo e entrou em outro
     estado (mas não no estado notPresent). Esse outro estado é indicado
     pelo valor incluído de ifOperStatus."
 ::= { snmpTraps 3 }
```

A lista de vinculações é denominada OBJECTS em vez de VARIABLES, mas ocorreram poucas alterações. O primeiro objeto é a interface específica (*ifI-*

*ndex*) que migrou da condição de *linkDown* para outra. A OID dessa trap é 1.3.6.1.6.3.1.1.5.3 ou *iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBObjects.snmpTraps.linkDown*.

## Mecanismo *inform* do SNMP

Finalmente, o SNMPv2 fornece um mecanismo *inform*, que permite a comunicação entre gerenciadores. Essa operação pode ser útil quando for necessária mais de uma NMS na rede. Quando um *inform* é enviado de uma NMS para outra, o receptor envia uma resposta para o emissor confirmado o recebimento do evento. Esse comportamento é semelhante aos das solicitações de *get* e *set*. Observe que é possível utilizar um *inform* do SNMP para enviar traps do SNMPv2 para uma NMS. Se você usar um *inform* para esse objetivo, o agente será notificado quando a NMS receber a trap.

## Operação de *report* do SNMP

A operação de *report* foi definida na versão draft do SNMPv2, mas nunca foi implementada. Agora, faz parte da especificação do SNMPv3 e deve permitir que a comunicação entre os mecanismos do SNMP (principalmente para relatar os problemas relacionados ao processamento de mensagens do SNMP).

## Considerações adicionais sobre o gerenciamento de hosts

O gerenciamento dos hosts é uma parte importante do gerenciamento da rede. Você poderia até supor que a Host Resources MIB faria parte de todo agente do SNMP baseado em host, mas esse não é o caso. Alguns agentes do SNMP implementam essa MIB e outros não. Alguns deles avançam ainda mais e implementam extensões proprietárias baseadas nessa MIB. Isso ocorre principalmente devido ao fato de que essa MIB foi elaborada para funcionar como uma estrutura básica descendente para o gerenciamento de hosts, especialmente para fomentar um desenvolvimento abrangente.

A Host Resources MIB define os sete grupos a seguir:

host	OBJECT IDENTIFIER ::= { mib-2 25 }
hrSystem	OBJECT IDENTIFIER ::= { host 1 }
hrStorage	OBJECT IDENTIFIER ::= { host 2 }
hrDevice	OBJECT IDENTIFIER ::= { host 3 }
hrSWRun	OBJECT IDENTIFIER ::= { host 4 }
hrSWRunPerf	OBJECT IDENTIFIER ::= { host 5 }
hrSWininstalled	OBJECT IDENTIFIER ::= { host 6 }

O OID do host é 1.3.6.1.2.1.25 (*iso.org.dod.internet.mgmt.mib-2.host*).

Os seis grupos restantes definem diversos objetos que fornecem informações sobre o sistema.

O grupo *hrSystem* (1.3.6.1.2.1.25.1) define objetos pertencentes ao próprio sistema, que abrangem tempo de funcionamento, data do sistema, usuários e processos do sistema.

Os grupos *hrDevice* (1.3.6.1.2.1.25.3) e *hrStorage* (1.3.6.1.2.1.25.2) definem objetos pertencentes aos sistemas de arquivo e ao armazenamento do sistema, como a memória global do sistema, utilização de disco e porcentagem não-ociosa da CPU. São muito úteis porque podem ser utilizados para gerenciar as partições de disco existentes no host. Você também pode utilizá-los para procurar erros em um dispositivo de disco específico.

Os grupos *hrSWRun* (1.3.6.1.2.1.25.4), *hrSWRunPerf* (1.3.6.1.2.1.25.5) e *hrSWInstalled* (1.3.6.1.2.1.25.6) definem objetos que representam alguns aspectos do software em execução ou instalado no sistema. Com esses grupos, é possível determinar o sistema operacional em execução no host, assim como os programas que o host está executando no momento. O grupo *hrSWInstalled* pode ser usado para rastrear os pacotes de software instalados.

Como é possível observar, a Host Resources MIB oferece alguns objetos necessários ao gerenciamento de sistema que podem ser usados por praticamente todos os que precisam gerenciar sistemas críticos.

## *Considerações adicionais sobre a monitoração do sistema*

Um tratamento mais abrangente o RMON está além do escopo deste livro, mas compensa abordar os grupos que formam o RMONv1. Geralmente as provas do RMON são dispositivos independentes que acompanham o tráfego nos segmentos da rede à qual estão acoplados. Alguns fornecedores implementam pelo menos algum tipo de prova de RMON em seus roteadores, hubs ou comutadores. O Capítulo 9 apresenta um exemplo de como configurar o RMON em um roteador Cisco.

A RMON MIB define os dez grupos a seguir:

rmon	OBJECT IDENTIFIER ::= { mib-2 16 }
statistics	OBJECT IDENTIFIER ::= { rmon 1 }
history	OBJECT IDENTIFIER ::= { rmon 2 }
alarm	OBJECT IDENTIFIER ::= { rmon 3 }
hosts	OBJECT IDENTIFIER ::= { rmon 4 }
hostTopN	OBJECT IDENTIFIER ::= { rmon 5 }
matrix	OBJECT IDENTIFIER ::= { rmon 6 }
filter	OBJECT IDENTIFIER ::= { rmon 7 }
capture	OBJECT IDENTIFIER ::= { rmon 8 }
event	OBJECT IDENTIFIER ::= { rmon 9 }

O RMONv1 fornece dados estatísticos no nível de pacote sobre uma LAN ou WAN inteira. O OID do *rmon* é 1.3.6.1.2.1.16 (iso.org.dod.internet.mgmt.mib-2.rmon). O RMONv1 é formado por nove grupos:

*statistics* (1.3.6.1.2.1.16.1)

Contém dados estatísticos sobre todas as interfaces Ethernet monitoradas pela prova

*history* (1.3.6.1.2.1.16.2)

Registra amostras periódicas de dados estatísticos do grupo *statistics*

*alarm* (1.3.6.1.2.1.16.3)

Permite que um usuário configure um intervalo de polling e um limiar para qualquer objeto registrado pela prova do RMON

*hosts* (1.3.6.1.2.1.16.4)

Registra dados estatísticos sobre o tráfego de cada host na rede

*hostTopN* (1.3.6.1.2.1.16.5)

Contém dados estatísticos do host utilizados para gerar relatórios sobre os hosts que encabeçam uma lista ordenada por um parâmetro existente na tabela de hosts

*matrix* (1.3.6.1.2.1.16.6 )

Armazena informações de erro e utilização para os conjuntos de dois endereços

*filter* (1.3.6.1.2.1.16.7)

Faz a correspondência de pacotes com base em uma equação de filtro; quando um pacote encontra um filtro correspondente, esse filtro pode ser obtido ou um evento pode ser gerado

*capture* (1.3.6.1.2.1.16.8)

Permite a captação dos pacotes se corresponderem a um filtro do grupo de filtros

*event* (1.3.6.1.2.1.16.9)

Controla a definição de eventos do RMON

O RMONv2 otimiza o RMONv1 ao fornecer a obtenção de dados estatísticos no nível de aplicativo e rede. Como o único exemplo de RMON deste livro usa o RMONv1, pararemos aqui e não entraremos no RMONv2. Contudo, recomendamos que você leia a RFC 2021 para conhecer as otimizações implementadas por esta versão do RMON para a monitoração de redes.

# 3

## *Arquiteturas de NMS*

Uma vez que você já conhece os conceitos básicos do modo de comunicação entre as estações de gerenciamento (NMSs) e os agentes, apresentaremos o conceito de uma arquitetura de gerenciamento de redes. Antes de desenvolver o gerenciamento do SNMP, você deve investir esforços no desenvolvimento de um planejamento coerente. Se você simplesmente instalar o software da NMS software em algumas de suas máquinas de desktop favoritas, provavelmente terminará com algo que não funciona muito bem. Arquitetura de NMS significa um plano que ajuda a utilizar as NMSs com eficiência para gerenciar a rede. Um componente importante do gerenciamento de rede é a seleção do hardware adequado (como uma plataforma apropriada em que executar a NMS) e a certeza de que suas estações de gerenciamento estão localizadas de modo a permitir a observação eficaz dos dispositivos na rede.

### *Considerações sobre o hardware*

Gerenciar uma rede relativamente grande requer uma NMS com um poder de computação considerável. Nos atuais ambientes em rede complexos, o tamanho das redes pode variar desde alguns nós até milhares de nós. O processo de polling e recebimento de traps de centenas ou milhares de entidades gerenciadas pode sacrificar o melhor dos equipamentos de hardware. O fornecedor de sua NMS pode ajudá-lo a determinar o tipo de hardware adequado para gerenciar sua rede. A maioria dos fornecedores tem fórmulas para calcular a quantidade de RAM necessária para obter o nível de desempenho almejado, em função das exigências de sua rede. Geralmente, isso engloba os dispositivos a serem pesquisados, a quantidade de informações solicitadas a cada dispositivo e o intervalo de polling dessas informações. O software a ser executado também é importante. Produtos de NMS, como o OpenView, são aplicativos complexos e pesados; para executar seus próprios scripts com a linguagem Perl, você pode utilizar uma plataforma de gerenciamento muito mais compacta.

Pode-se dizer algo mais útil do que “Pergunte ao fornecedor”? Sim. Primeiramente, embora estejamos acostumados a imaginar o software da NMS exigir-

do uma estação de trabalho com uma média de recursos instalados ou um PC altamente sofisticado, o hardware de desktop avançou tanto nos últimos dois anos, que qualquer PC moderno pode executar esse software. Em termos específicos, ao consultar as recomendações de alguns fornecedores, encontramos sugestões para uso de um PC com uma CPU mínima de 300 MHz, 128 MB de memória e 500 MB de espaço de disco. As exigências para estações de trabalho da Sun SPARC e HP são semelhantes.

Examinemos cada uma dessas exigências:

#### *CPU de 300 MHz*

Esse requisito está dentro da faixa de qualquer sistema desktop atual, mas provavelmente você não poderá remanejar seu velho equipamento da aposentadoria para usá-lo como uma estação de gerenciamento.

#### *128 MB de memória*

Talvez você precise adicionar memória a qualquer PC em uso; as estações de trabalho da Sun e HP são fornecidas com configurações de memória mais generosas. Honestamente, os fornecedores tendem a superestimar os requisitos de memória, de modo que não será um problema fazer um upgrade para 256 MB. Felizmente, a RAM tem um preço muito acessível no momento. (Os preços de memória flutuam de um dia para o outro, mas encontramos recentemente módulos DIMM de 256 MB por menos de 100 dólares.)

#### *500 MB de espaço de disco*

Provavelmente, essa recomendação é baseada na quantidade de espaço necessário para armazenar o software e não em relação ao espaço necessário para os arquivos de log, dados de tendências de longo prazo, etc. Mas, novamente, espaço é muito barato hoje em dia e mesquinharia é contraproducente.

Examinemos um pouco mais detalhadamente como o conjunto de dados de longo prazo afeta as exigências de espaço de disco em seu sistema. Primeiramente, perceba que alguns produtos dispõem de recursos mínimos de coleta de dados enquanto outros existem unicamente para a finalidade de obtenção de dados (como o MRTG). Uma coleta de dados eficiente depende até certo ponto do produto de NMS selecionado. Portanto, antes de escolher um produto de software, pondere sobre as exigências de coleta de dados. Você precisa fazer análise de tendências de longo prazo? Nesse caso, isso afetará a escolha do software e o hardware de execução.

Como um ponto de partida, vamos supor que você tenha 1.000 nós, que precise coletar dados a cada minuto e que está coletando 1 KB de dados por nó. Isso significa 1 MB por minuto, 1.4 GB por dia – você preencheria um disco de 40 GB em aproximadamente um mês. Isso beira o extravagante. Mas examinemos as premissas:

- Certamente, coletar dados a cada minuto é algo excessivo; coletar a cada 10 minutos deve bastar. Agora, seu disco de 40 GB armazenará quase 1 ano de dados.
- Mil nós não representam uma rede tão grande. Mas você deseja realmente armazenar dados de tendências dos PCs de todos os usuários? Grande parte desse livro ensina a controlar o volume de dados coletados. Em vez de 1.000 nós, contemos primeiramente as interfaces. E não levemos em consideração os sistemas de desktop – nosso interesse é somente os dados de tendência para nosso backbone de rede: principais servidores, roteadores, comutadores etc. Até em uma rede de porte médio, isso significa provavelmente 100 ou 200 interfaces.
- A quantidade de dados obtidos por interface depende de alguns fatores, e o formato dos dados não é o menos relevante. O status de uma interface pode ser “em funcionamento” ou “paralisada” – questão de um único bit. Se os dados estiverem sendo armazenados em uma estrutura de dados binários, podem ser representados por um único bit. Mas se você estiver usando o *syslog* para armazenar seus dados de log e escrevendo scripts em Perl para fazer análise de tendências, os registros do *syslog* terão aproximadamente 80 bytes, mesmo que você esteja armazenando somente um bit de informações. Os mecanismos de armazenamento de dados variam do *syslog* a esquemas de bancos de dados mirabolantes – obviamente, você precisa conhecer o que está usando e como suas exigências de armazenamento serão afetadas. Além disso, você precisa saber o volume de informações que será realmente mantido em cada interface. Para rastrear apenas o número de octetos de entrada e saída de cada interface, desde que você esteja armazenando esses dados com eficiência, seu disco de 40 GB pode facilmente durar quase um século.

Francamente, é difícil saber quais serão suas necessidades de armazenamento quando essas exigências permeiam duas a três ordens de magnitude. Mas a lição aprendida é a seguinte: nenhum fornecedor pode saber quais serão suas necessidades de armazenamento. Um gigabyte deve ser muito para dados de log em uma rede relativamente grande, se você estiver armazenando dados somente para um subconjunto razoável dessa rede, sem consultas freqüentes e sem salvar muitos dados. Mas isso abrange muitas variáveis e você é o único a controlar. Entretanto, lembre-se de que quanto maior o volume de dados coletados, tanto mais tempo e recursos da CPU serão necessários para filtrar todos esses dados e gerar resultados significativos. Não importa se você está usando um software dispendioso de análise de tendências ou alguns scripts domésticos – o processamento de muitos dados é dispendioso. Pelo menos em termos de coleta de dados de longo prazo, é melhor pecar por insuficiência do que por excesso.

## Arquiteturas de NMS

Antes de comprar todo o seu equipamento, compensa investir algum tempo para elaborar uma arquitetura para sua rede que a torne mais gerenciável. A ar-

quitetura mais simples possui uma única estação de gerenciamento, responsável pela rede inteira, como mostra a Figura 3-1.

A rede ilustrada na Figura 3-1 tem três locais: Nova York, Atlanta e São José. A NMS em Nova York é responsável por gerenciar não somente a parte da rede de Nova York, como também as de Atlanta e São José. As traps enviadas de qualquer dispositivo em Atlanta ou São José precisam atravessar a Internet para alcançar a NMS em Nova York.

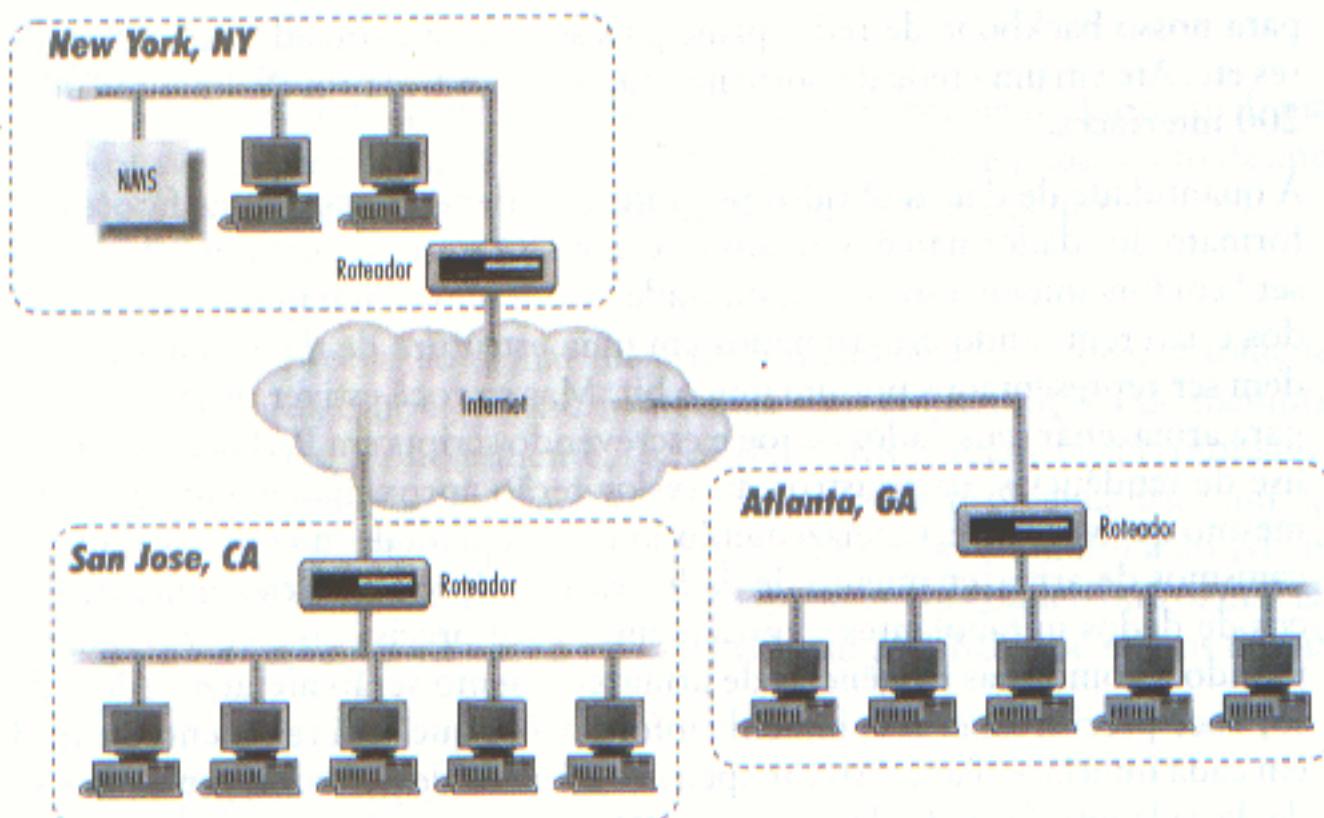


Figura 3-1 Arquitetura de uma NMS individual

O mesmo se aplica aos dispositivos de polling em São José e Atlanta: a NMS em Nova York precisa enviar suas solicitações pela Internet para alcançar esses locais remotos. Para as redes pequenas, uma arquitetura como essa pode funcionar bem. Contudo, quando a rede crescer até o ponto em que uma única NMS não possa mais gerenciar todos os aspectos, essa arquitetura se tornará um grande problema. A NMS em Nova York pode se tornar lenta em suas consultas a locais remotos, principalmente porque há muitos aspectos a serem gerenciados. Com isso, quando surgirem problemas em um local remoto, talvez eles não sejam percebidos durante algum tempo. No pior caso, sequer serão percebidos.

Também compensa ponderar sobre as equipes. Com uma única NMS, a equipe de suas operações básicas estaria acompanhando o funcionamento da rede em Nova York. Mas geralmente os problemas exigem a presença de alguém no local, para tomar uma decisão. Isso requer alguém em Atlanta e São José, adicionalmente à coordenação pertinente. Talvez você não precise de um administrador de rede em tempo integral, mas precisará de alguém que saiba o que fazer quando um roteador falhar.

Quando sua rede crescer até o ponto em que uma única NMS não possa mais gerenciar tudo, essa é a hora de migrar para uma arquitetura de NMSs distribuídas. A idéia que respalda essa arquitetura é simples: use duas ou mais estações de gerenciamento e aloque-as o mais próximas possível dos nós que gerenciam. No caso de nossa rede de três locais, teríamos uma NMS em cada local. A Figura 3-2 mostra a inclusão de duas NMSs na rede.

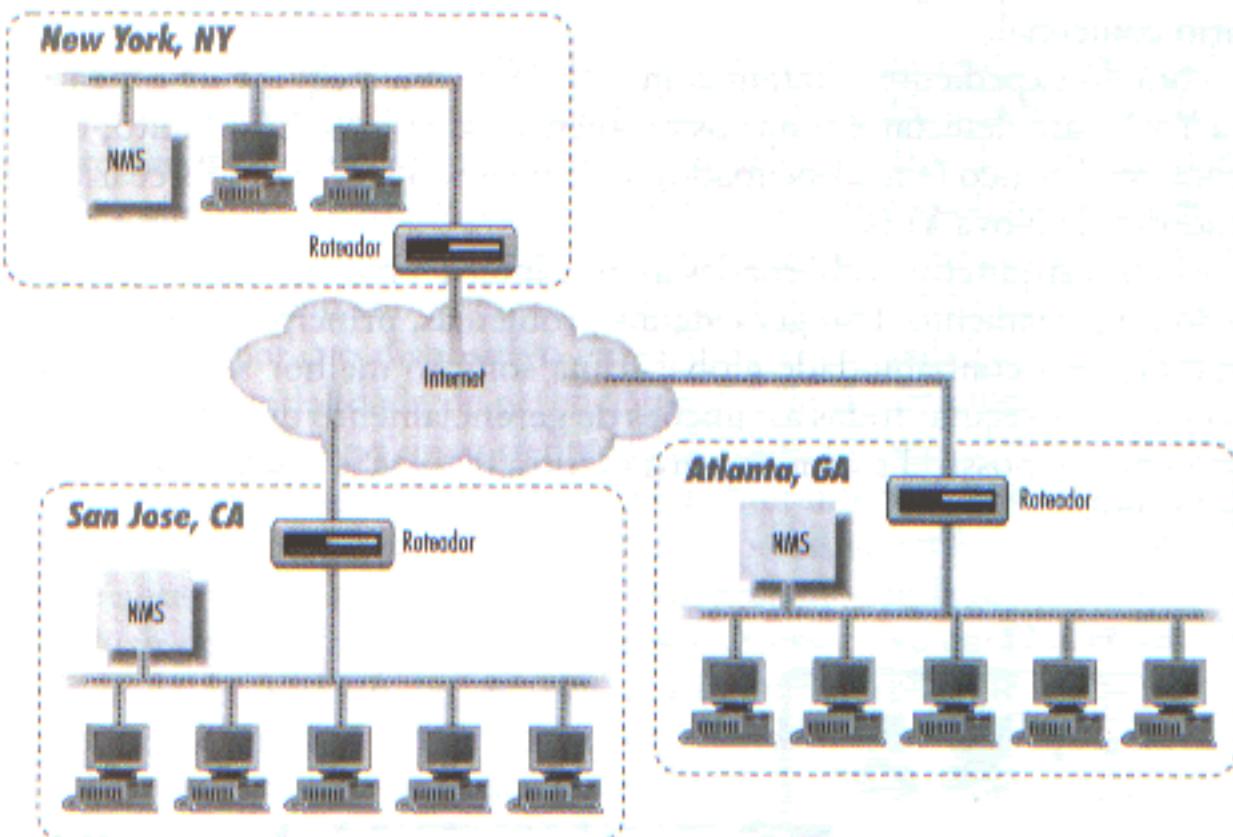


Figura 3-2 Arquitetura de NMSs distribuídas

Essa arquitetura tem algumas vantagens, dentre elas, a flexibilidade. Com a nova arquitetura, as NMSs em Atlanta e São José podem atuar como estações de gerenciamento independentes, cada qual com uma equipe totalmente auto-suficiente ou podem enviar eventos para a NMS em Nova York. Se as NMSs remotas enviarem todos os eventos para a NMS de Nova York, não haverá necessidade de uma equipe de operações adicional em Atlanta e São José. Inicialmente, parece que voltamos à situação da Figura 3-1, mas isso realmente não acontece. A maioria dos produtos de NMS fornece algum tipo de interface cliente para visualizar os eventos ocorrendo atualmente na NMS (traps recebidas, respostas a consultas etc.). Como a NMS que envia eventos para Nova York já detectou o problema, basta deixar a NMS de Nova York percebê-lo para tratá-lo adequadamente. A NMS de Nova York não precisava utilizar recursos importantes para consultar a rede remota e descobrir a ocorrência de um problema.

A outra vantagem é que você pode alocar equipe de operações em Atlanta e São José para gerenciar cada uma dessas localidades remotas, se necessário. Se Nova York perder a conectividade com a Internet, os eventos enviados de Atlanta ou São José não seriam detectados em Nova York. Com uma equipe de opera-

ções em Atlanta e São José, e as NMSs nessas localidades funcionando no modo independente, não seria tão crítica uma interrupção das operações na rede em Nova York. A equipe do local remoto continuaria funcionando como se nada tivesse acontecido.

Outra possibilidade nessa arquitetura é um modo híbrido: você fornece atendimento ao centro de operações em Nova York 24 horas por dia, 7 dias por semana, mas as operações de Atlanta e São José funcionam somente durante o horário comercial.

Fora do expediente, contam com a NMS e com a equipe de operações de Nova York para detectar e tratar os problemas ocorridos. Entretanto, no horário comercial crítico (e mais ocupado), Atlanta e São José não sobrecarregam os operadores de Nova York.

As duas arquiteturas abordadas usam a Internet para enviar e receber o tráfego do gerenciamento. Isso gera alguns problemas, principalmente no tocante à segurança e à confiabilidade global. Uma solução melhor seria utilizar links privados para executar todas as funções de gerenciamento de rede. A Figura 3-3 mostra como é possível estender a arquitetura de NMSs distribuídas para utilizar esses links.

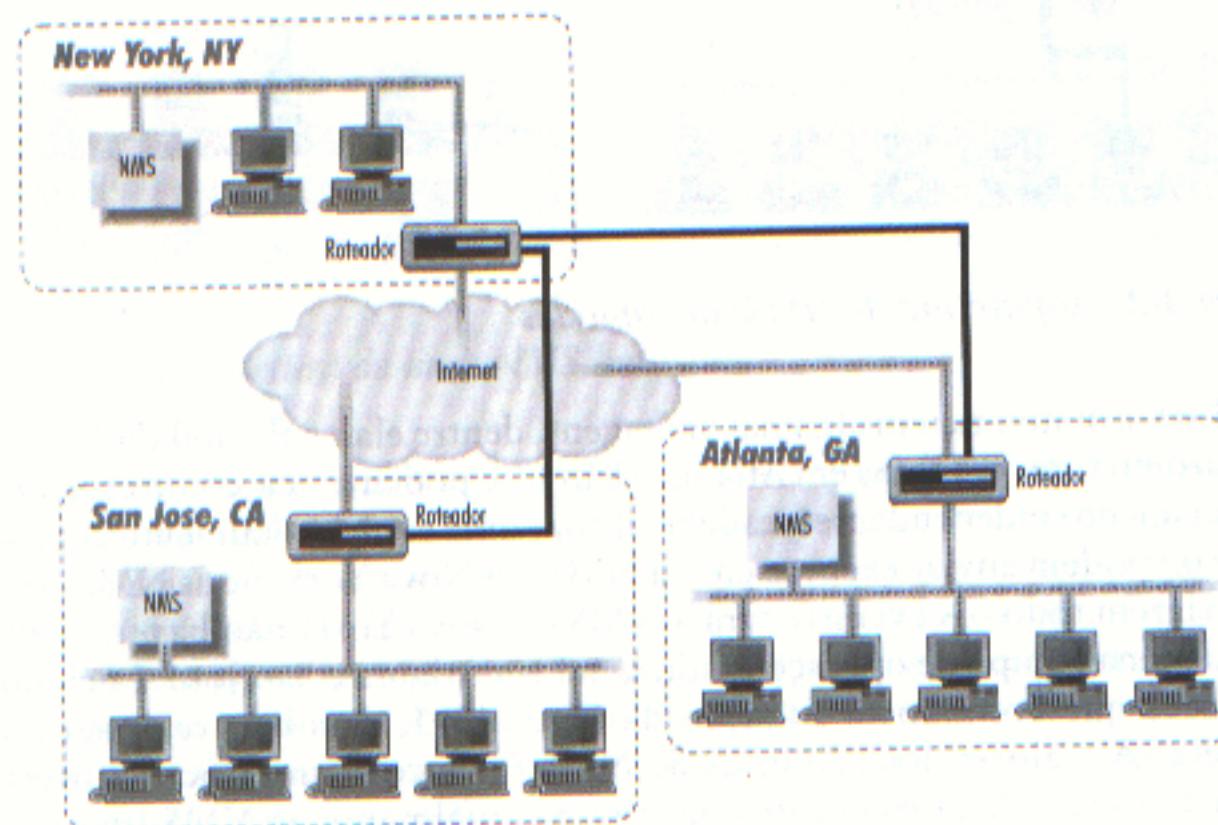


Figura 3-3 Usando links privados para o gerenciamento de rede

Vamos supor que o roteador de Nova York seja o principal roteador da rede. Estabeleçemos links privados (mas não necessariamente de alta velocidade) entre São José e Nova York, e entre Nova York e Atlanta. Isso significa que São José não conseguirá alcançar Nova York, mas alcançará Atlanta via Nova York. Atlanta usará Nova York para alcançar São José, também. Os links privados (destacados pelas conexões mais espessas de um roteador para o outro) são

basicamente dedicados ao tráfego do gerenciamento, embora pudéssemos aproveitá-los em outras aplicações. O uso de links privados tem outra vantagem: nossas strings de comunidade nunca são enviadas pela Internet. O uso de links de rede privados para o gerenciamento de rede funciona igualmente bem na arquitetura de NMS individual. Evidentemente, se a rede de sua empresa consiste totalmente em links privados e suas conexões da Internet são dedicadas somente ao tráfego externo, usar links privados no tráfego do gerenciamento seria um desperdício.

O último item a ser mencionado é o conceito de polling orientada por traps. Isso não tem nada a ver com a arquitetura de NMS, mas pode ajudar a aliviar a pressão do gerenciamento da NMS. A idéia por trás da polling orientada por traps é simples: a NMS recebe uma trap e inicia uma poll (consulta/verificação) ao dispositivo que gerou a trap. O objetivo desse cenário é detectar se existe realmente um problema no dispositivo, e permitir que a NMS ignore (ou aloque alguns recursos para) o dispositivo em operação normal. Se uma organização conta com essa modalidade de gerenciamento, deve implementá-la de modo a praticamente excluir a polling orientada por traps. Ou seja, deve evitar a polling de dispositivos em intervalos regulares para obter informações de status. Em vez disso, as estações de gerenciamento devem apenas aguardar o recebimento de uma trap antes de consultar um dispositivo. Essa modalidade de gerenciamento pode reduzir consideravelmente os recursos necessários a uma NMS para gerenciar uma rede. Entretanto, há uma desvantagem: as traps podem desaparecer na rede e nunca alcançar a NMS. Essa é a realidade da natureza sem conexão do UDP e da natureza imperfeita das redes.

## Um passo à frente

O gerenciamento de rede baseado na Web engloba o uso do *HyperText Transport Protocol* (HTTP) e da *Common Gateway Interface* (CGI) para gerenciar as entidades em rede. Funciona incorporando um servidor da Web em um dispositivo compatível com o SNMP, aliado ao mecanismo da CGI para converter as solicitações ao estilo SNMP (de uma NMS baseada na Web) em operações reais de SNMP, e vice-versa. É possível incorporar servidores da Web em dispositivos dessa natureza a um custo financeiro e operacional muito baixo.

A Figura 3-4 é um diagrama simplificado da interação entre uma NMS baseada na Web e um dispositivo gerenciado. A aplicação da CGI preenche a lacuna entre o aplicativo de gerenciamento e o mecanismo do SNMP. Em alguns casos, o aplicativo de gerenciamento pode ser um conjunto de *applets* Java transferidos para o navegador da Web e executados no gerenciador baseado na Web. As versões atuais do OpenView são fornecidas com uma GUI baseada na Web.

O gerenciamento de rede baseado na Web poderia evitar ou, pelo menos, reduzir a necessidade do software tradicional de NMS, que pode ter uma aquisição, configuração e manutenção dispendiosas. Grande parte dos principais fornecedores de NMS atuais oferece suporte somente para algumas versões conhecidas do Unix e só recentemente começaram a ter suporte para o Windows | 51

9x/NT/2000, limitando com isso suas opções de sistemas operacionais. Entretanto, com uma NMS baseada na Web, essas duas preocupações desaparecem. Em sua grande maioria, a tecnologia de navegadores da Web é gratuita e o Netscape Communications (atualmente, AOL Time Warner) aceita alguns tipos de Unix, assim como as plataformas Wintel e Apple.

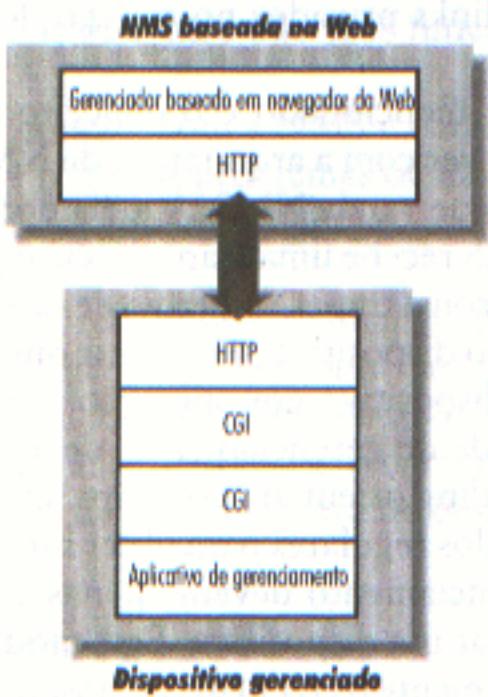


Figura 3-4 Gerenciamento de rede baseado na Web

O gerenciamento de rede baseado na Web não deve ser percebido como uma panacéia. É uma boa idéia, mas levará algum tempo para que os fornecedores adotem essa tecnologia e migrem para a integração de seus produtos atuais, com base na Web. Também existe a questão da padronização ou da falta de padronização. O consórcio *Web-Based Enterprise Management* (WBEM) trata essa questão ao definir um padrão para o gerenciamento baseado na Web. Líderes do setor, como a Cisco e BMC Software, estão entre os primeiros fundadores do WBEM. Você obterá mais informações sobre essa iniciativa na página da Web da Distributed Management Task Force, <http://www.dmtf.org/wbem/>.

# 4

## *Hardware compatível com o SNMP*

Saber se existem dispositivos gerenciáveis por SNMP é um bom ponto de partida para o gerenciamento *Zen* de redes. Antes de examinarmos como é possível determinar se o que você já possui é gerenciável, discutiremos sucintamente o que torna um dispositivo compatível com o SNMP.

Os fornecedores não são obrigados a implementar todas as MIBs que o SNMP fornece,\* mas os dispositivos gerenciáveis por SNMP devem aceitar no mínimo a MIB-II. Também cabe aos fornecedores implementar algumas das MIBs mais úteis, assim como as próprias MIBs privadas, uma vez que a possibilidade de gerenciar um produto que usa efetivamente o SNMP é um ponto de venda cada vez mais importante.

### *O que realmente significa ser compatível com o SNMP?*

Alguns fornecedores afirmam que seus produtos são compatíveis com SNMP. Em termos gerais, isso procede. Mas o que eles realmente querem dizer é que seus produtos aceitam um conjunto de operações de SNMP, assim como a MIB-II. Quanto à compatibilidade com o SNMPv1, as operações aceitas englobam:

- *get*
- *get-next*
- *set*
- *get-response*
- *trap*

\* Você encontrará alguns exemplos dessas MIBs padrão no Capítulo 1.

Além disso, se o produto é compatível com o SNMPv2 e SNMPv3, deve oferecer suporte para as seguintes operações:

- *get-bulk*
- *inform*
- *notification*
- *report*

Os fornecedores podem oferecer suporte para o SNMPv1, SNMPv2, SNMPv3 ou para os três. Um agente de SNMP que aceita duas versões do SNMP é denominado “bilingüe”. Recentemente, esse recurso estava restrito aos dispositivos com suporte para o SNMPv1 e SNMPv2. Agora, um dispositivo pode aceitar as três versões, o que tecnicamente o torna um trilingüe. É possível para um agente falar todas as versões do SNMP porque a SMIv2 é um superconjunto da SMIv1 e a SMIv2 é usada na maior parte das vezes, com o SNMPv3.

Entretanto, o suporte para essas operações é apenas uma peça do quebra-cabeça de fornecer um produto gerenciável. Outra peça é oferecer uma MIB privada suficientemente abrangente para fornecer aos gerentes de rede as informações necessárias para gerenciar suas redes com inteligência. Nos atuais ambientes de rede complexos, não compensa adquirir um equipamento com MIB privada mínima ou implementada com deficiência. Por exemplo, é importante medir a temperatura ambiente dentro de dispositivos como roteadores, hubs e comutadores. A Cisco e outros fornecedores liberam essas informações por meio de suas MIBs privadas; outros fornecedores não. Se você estiver em processo de aquisição de um roteador sofisticado, convém examinar as MIBs provadas dos fornecedores para descobrir quais deles fornecem informações mais relevantes.

Outro fator que afeta o suporte de MIB do fornecedor é a definição do produto. A Concord Communications (fornecedores de um agente de SNMP para o Unix e Windows) provavelmente não aceitará a RS-232 MIB (RFC 1659), porque seus produtos são destinados a fornecer informações de gerenciamento de sistemas e aplicativos. A 3Com, por outro lado, implementou essa MIB para a respectiva linha de Hubs de Dupla Velocidade, porque esses hubs possuem portas para RS-232.

## ***Meu dispositivo é compatível com o SNMP?***

A documentação de seu produto deve ajudar a determinar a compatibilidade do hardware ou software com o SNMP. Consulte também seu representante de vendas ou a central de atendimento ao cliente, se aplicável. Outra maneira de saber se um produto é compatível com o SNMP é fazer uma consulta do *snmpget* no dispositivo em questão.\*

\* Com esse método, podemos tentar adivinhar uma string de comunidade. Em nosso caso, tentamos *public* ou *private*. Se não obtivermos uma resposta, pode significar que erramos em nossa adivinhação ou que o agente não está configurado.

É fácil emitir um *get* de diagnóstico em qualquer dispositivo. O modo mais comum de fazer isso é localizar um host do Unix que possua o comando binário *snmpget* instalado.\* Existem algumas variações desse comando; consulte a página do manual ou o administrador do sistema para obter ajuda. A variável mais fácil de consultar é a *sysDescr*, que fornece uma descrição do sistema sendo consultado. Examine o que acontece quando você usa o comando *snmpget* do Net-SNMP para verificar a *sysDescr* em um host comum do Linux:

```
$ snmpget linuxserver.ora.com public system.sysDescr.0  
system.sysDescr.0 = "Linux version 2.0.34 (root@porky.redhat.com)  
(gcc version 2.7.2.3) #1 Fri May 8 16:05:57 EDT 1998"
```

A resposta do *linuxserver.ora.com* é característica da maioria dos dispositivos gerenciados. Entretanto, observe que não há nada consagrado na descrição real; o texto recuperado varia de um fornecedor para o outro. Emitir um *snmpget* em um roteador Cisco 2503 deve retornar algo assim:

```
$ snmpget orarouter.ora.com public system.sysDescr.0  
system.sysDescr.0 = "Cisco Internetwork Operating System Software  
..IOS (tm) 2500 Software (C2500-I-L), Version 11.2(5), RELEASE  
SOFTWARE (fc1)..Copyright (c) 1986-1997 by cisco Systems, Inc...  
Compiled Mon 31-Mar-97 19:53 by ckralik"
```

A descrição do sistema desse roteador informa que ele está executando a Versão 11.2(5) do Cisco IOS. Geralmente, esse tipo de informação é inútil, mas, pelo menos, informa que o dispositivo está executando um agente do SNMP. Veja o que acontece quando algo dá errado:

```
$ snmpget linuxserver.ora.com public system.sysDescr.0  
Timeout: No Response from linuxserver.ora.com.
```

Essa mensagem indica que o comando *snmpget* do Net-SNMP não recebeu uma resposta do *linuxserver.ora.com*. Algumas peças podem estar erradas, uma das quais é que não existe um agente de SNMP em execução no host de destino. Entretanto, também é possível que o *linuxserver* tenha sido derrubado, que exista algum tipo de problema na rede ou que tudo esteja executando corretamente, mas você não esteja usando a string de comunidade correta. Além disso, é possível que o dispositivo sendo consultado tenha recursos de SNMP, mas o agente de SNMP não seja habilitado até que você mesmo o configure.

Se você supõe que tem um equipamento gerenciável mas não tem certeza, convém saber que a maioria dos fornecedores despacha seus produtos com as strings de comunidade *read* e *write* definidas com *public* e *private*, respectivamente. (As ferramentas do Net-SNMP utilizadas aqui usam *private* como predefinição para ambas as strings de comunidade.\*\*)

\* O Capítulo 7 aborda a instalação do agente Net-SNMP e do kit de ferramentas que inclui utilitários como o *snmpget*.

\*\* Como nossos agentes usam a string de comunidade *public* e a string predefinida no Net-SNMP é *private*, é necessário especificar uma string de comunidade *public* na linha de comando.

Assim que você confirmar que o dispositivo sendo testado é gerenciável por SNMP, mude imediatamente as strings de comunidade. Deixá-las definidas com valores conhecidos, como *public* e *private*, é um sério problema de segurança.

Quando ficar confirmado que seu dispositivo tem suporte para SNMP, você poderá ir mais além e verificar se esse dispositivo aceita a Versão 2. Uma maneira eficiente de fazer isso é emitir uma solicitação que só possa ser respondida por um agente da Versão 2, como a solicitação de *bulk-get*. Você pode utilizar o comando *snmpbulkget* demonstrado no Capítulo 2 para fazer essa solicitação:

```
$ snmpbulkget -v2c -B 1 3 linux.ora.com public sysDescr ifInOctets ifOutOctets
system.sysDescr.0 = "Linux linux 2.2.5-15 #3 Thu May 27 19:33:18 EDT 1999 i686"
interfaces.ifTable.ifEntry.ifInOctets.1 = 70840
interfaces.ifTable.ifEntry.ifOutOctets.1 = 70840
interfaces.ifTable.ifEntry.ifInOctets.2 = 143548020
interfaces.ifTable.ifEntry.ifOutOctets.2 = 111725152
interfaces.ifTable.ifEntry.ifInOctets.3 = 0
interfaces.ifTable.ifEntry.ifOutOctets.3 = 0
```

Agora, sabemos que o *linux.ora.com* tem suporte para o SNMPv2 – em termos específicos, o v2c. Podemos avançar e tentar confirmar a presença de suporte para a Versão 3? Para a Versão 3, é melhor consultar a documentação do fornecedor. A maioria dos fornecedores ainda não oferece suporte para a Versão 3 e prevemos que essa adoção seja muito lenta – alguns fornecedores têm suporte somente para a Versão 1.

## Upgrade de hardware

Após confirmar se você possui ou não dispositivos do SNMP na rede, talvez este seja o momento de fazer um upgrade! É possível que você descubra que alguns dispositivos que você deseja gerenciar não têm suporte para o SNMP. Existem duas maneiras de fazer o upgrade: você pode aposentar o equipamento existente e comprar outro hardware mais moderno, mais gerenciável ou pode fazer um upgrade do firmware de seu equipamento (se oferecido pelo fornecedor) para uma versão que aceite o SNMP. Contudo, alguns fornecedores se oferecerão para comprar o equipamento mais antigo ou até dar um desconto pela troca por um equipamento do concorrente.

Evidentemente, a atualização de seu equipamento pode não ser necessária. Se você possui aplicativos de software usados para gerenciar equipamentos não compatíveis com o SNMP, e esses aplicativos funcionam, não há necessidade de um upgrade. Se você domina bem os scripts e deseja conhecer mais detalhes sobre o SNMP, descobrirá que é possível escrever scripts que permitem utilizar o SNMP para monitorar aplicativos sem suporte para SNMP, usando wrapper/scripts. Para obter um exemplo, leia o Capítulo 12.

Seja qual for o método adotado, entenda que o SNMP existe para oferecer um modo coerente de gerenciar equipamentos em rede. Se você está gerencian-

do atualmente sua rede com diversas ferramentas de gerenciamento herdadas, cada qual aceitando alguns dispositivos de fornecedores específicos, o SNMP é uma alternativa a mais. Talvez você se sinta seguro com as ferramentas antigas – mas será cada vez mais prático utilizar o SNMP para oferecer uma abordagem de gerenciamento de rede uniforme.

## Em resumo

É provável que você esteja comprando dispositivos compatíveis com o SNMP, há anos, sem ter conhecimento do assunto. À medida que o SNMP tem se popularizado, tem sido incorporado a um número cada vez maior de dispositivos. A compatibilidade do SNMP tornou-se um verdadeiro ponto de venda para a maioria dos fornecedores.

Talvez nem seja necessário dizer, mas a maioria dos dispositivos de rede aceita o SNMP, inclusive roteadores, bridges, hubs, servidores e PCs de mesa.\* Contudo, vários outros tipos de equipamentos também são gerenciáveis via SNMP, incluindo os no-breaks, unidades de condicionamento de ar e outras partes importantes de sua infra-estrutura. Após identificar os roteadores e hubs compatíveis com o SNMP, fique de olhos abertos em relação aos outros dispositivos que devam ser gerenciados. Embora o SNMP seja eficiente ao gerenciar sua rede, hosts, hubs e roteadores, ele não está limitado a um ambiente de operação em rede somente.

## Um passo à frente

A Internet Engineering Task Force (IETF) está em processo de definição de uma tecnologia de rastreamento de padrões para favorecer a possibilidade de estender o agente do SNMP (AgentX). Conforme definido anteriormente, um agente de SNMP é um software residente em um dispositivo gerenciado, que responde a solicitações do SNMP e gera traps assíncronas. Você encontrará informações sobre a possibilidade de extensão dos agentes na RFC 2741 e no site da Web do AgentX, <http://www.scguild.com/agentx/>. A necessidade do AgentX surge da impossibilidade de adicionar e remover objetos da MIB enquanto um agente estiver em execução; em outras palavras, a falta de um método padrão de estender a funcionalidade de um agente. O *SNMP Multiplexing Protocol* (SMUX, RFC 1227) foi uma tentativa anterior de fornecer padronização para a possibilidade de extensão de agentes, mas o protocolo foi considerado deficiente e, desde então, foi abandonado.

A Figura 4-1 é uma visão geral da arquitetura do AgentX. Com o AgentX, o agente consiste em uma única entidade de processamento, denominada *agente*

\* Os hubs, comutadores e roteadores mais simples e elaborados para uso doméstico provavelmente não terão suporte para SNMP. Geralmente, os hubs e comutadores com suporte para SNMP são anunciados como “gerenciáveis” e, normalmente, custam muito mais caro. Quanto aos roteadores, leia a documentação atentamente.

*principal* e nenhuma ou mais entidades de processamento denominadas *subagentes*. O agente principal e os subagentes podem residir no mesmo dispositivo ou podem se comunicar por meio de um dispositivo proxy. O agente principal se comunica com a NMS, quase como um agente de SNMP comum. Os subagentes têm acesso direto à MIB, enquanto o agente principal não tem. Consequentemente, os subagentes executam funções de gerenciamento em variáveis gerenciadas, depois transmitem essas informações ao agente principal por meio do protocolo AgentX, que não é baseado no SNMP.

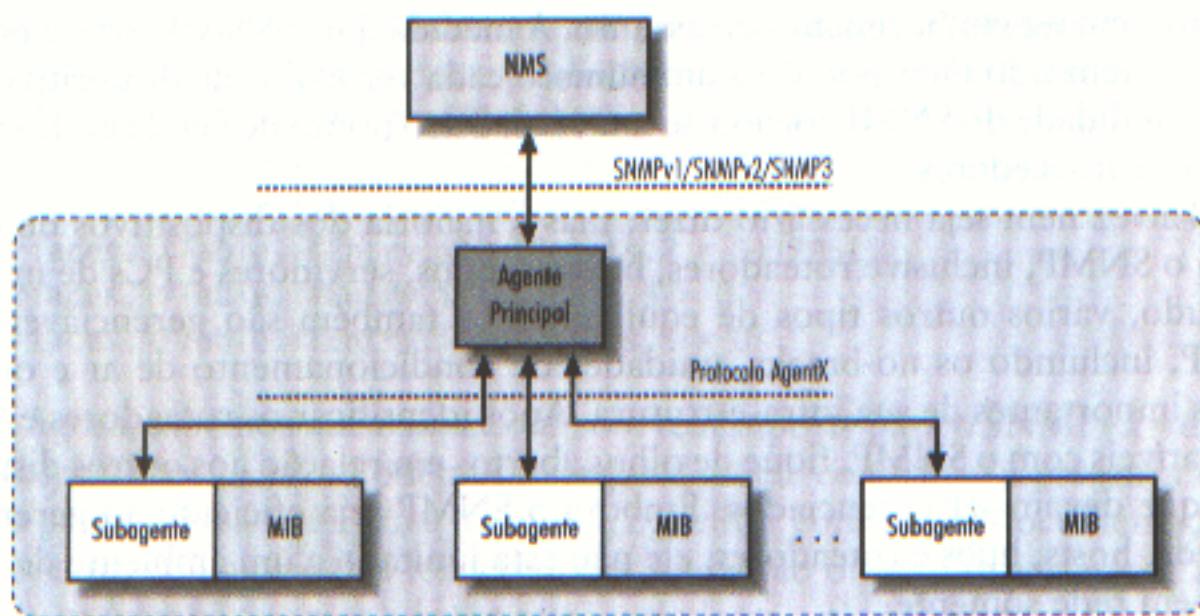


Figura 4-1 Arquitetura do AgentX

Sem uma abordagem padronizada para a possibilidade de extensão, é muito difícil para os fornecedores rastrear as possíveis extensões para os agentes das diversas plataformas que aceitam. O AgentX tenta endereçar essa questão, oferecendo aos fornecedores uma interface consistente para estender agentes. Também define o conceito de regiões da MIB ou conjuntos de variáveis gerenciadas. Um subagente é responsável por registrar essas MIBs junto a um único agente principal. Na prática, isso significa que os fornecedores terão um subagente para cada MIB que implementarem; por exemplo, um subagente de RMON, um subagente de MIB-II, um subagente de Host Resources e outros. Isso ajuda aos fornecedores porque oferece uma maneira de adicionar e remover instâncias de MIB de um agente, sem prejudicar a real operação entre uma NMS e o agente.

## *Software de gerenciamento de rede*

Existem alguns pacotes de software de SNMP disponíveis, desde bibliotecas de programação que permitem criar utilitários exclusivos (usando a linguagem Perl, C/C++ ou Java) até plataformas completas e dispendiosas de gerenciamento de rede. Este capítulo apresenta algumas vantagens e desvantagens de alguns dos pacotes mais usados. Isso não somente lhe dará uma idéia dos pacotes existentes, como também ajudará a decidir o que é adequado para você (entretanto, lembre-se de que essas vantagens e desvantagens representam apenas opiniões nossas). Sempre que possível, as soluções de fonte aberta e os produtos comercializados.

O software de gerenciamento é enquadrado em cinco categorias

- Agentes de SNMP
- Suítes de NMS
- Gerenciadores de componentes (gerenciamento específico do fornecedor)
- Software de análise de tendências
- Software de suporte

Infelizmente, especificar as suas necessidades não é tão simples quanto selecionar um programa de cada categoria. Se a sua rede é pequena e você deseja criar ferramentas exclusivas, provavelmente não necessita de uma suíte complexa de NMS. Se você precisa ou não de um software de análise de tendências depende, naturalmente, de seu interesse em analisar as tendências relacionadas à utilização de sua rede. Os produtos disponíveis dependem, em parte, das plataformas em que você está interessado. O mínimo que você conseguirá é um agente de SNMP em um dispositivo e algum software que recupere um valor nesse dispositivo (usando um *get* de SNMP). Embora sendo o mínimo, é suficiente para começar a trabalhar e você pode obter o software gratuitamente.

Este capítulo apresenta uma pesquisa abrangente de alguns produtos importantes em cada uma dessas categorias. Como existem muitos pacotes do que é possível abordar neste livro, consulte o Network Management Server (<http://net-man.cit.buffalo.edu/Products.html>) para obter listas de produtos de gerenciamento de rede.

## Agentes de SNMP

Como explicamos no Capítulo 1, um agente é um software que controla todas as comunicações de SNMP com e de qualquer dispositivo compatível com o SNMP. Em alguns dispositivos, como os roteadores da Cisco, o software do agente é incorporado ao próprio dispositivo e não requer instalação. Em outras plataformas, como o Windows NT, talvez seja necessário instalar o agente como parte de um pacote de software adicional.

Antes de examinar os tipos de agentes necessários a você, pesquise os tipos de dispositivos existentes na rede e os tipos de informações que você deseja receber de cada um. Alguns agentes são muito básicos e retornam apenas um volume limitado de informações, enquanto outros podem retornar muitas informações. Para começar, descubra se você precisa receber informações procedentes de servidores (Unix, Windows NT, etc.) ou de dispositivos de rede (roteadores, comutadores, etc.). Geralmente, os dispositivos do tipo rede externa fornecem mais informações do que seus parceiros de servidor. Por outro lado, os dispositivos de rede não são facilmente estendidos (talvez nem seja possível), em parte porque o hardware da rede geralmente não dispõe de um ambiente operacional baseado em disco.\* Isso impede que o usuário final acesse o agente para fazer modificações ou estendê-lo. O restante desta seção fornece informações sobre alguns dos pacotes de software atualmente disponíveis para serem utilizados como agentes de SNMP.



Certifique-se de saber que tipo de software está em execução nos servidores de sua rede (sistemas de correio, pacotes de contabilidade etc.). Alguns aplicativos não ouvirão nem responderão a solicitações do SNMP, mas enviarão traps. As traps podem ser muito úteis para monitorar alguns desses aplicativos. Além disso, existem aplicativos de varredura de vírus, logins remotos (pcAnywhere) e os UPSs que enviarão traps informativas quando for detectado um erro. Procure este recurso na próxima vez em que você adquirir qualquer pacote ou suíte de software.

### HP Extensible SNMP Agent

<http://www.openview.hp.com>

**Plataformas** Solaris, HP-UX

**Vantagens** Inclui um programa *snmptrap* e um agente HP que oferece funcionalidade adicional (em sua maioria para os sistemas HP). O agente é extensível usando um subconjunto do ASN.1.

**Desvantagens** Custo por dispositivo. É necessário rastrear vários daemons.

\* Leia o Capítulo 11 para obter uma abordagem dos agentes extensíveis.

**Plataformas** Solaris

**Vantagens** Disponível gratuitamente para as versões mais recentes do Solaris. Empacotado com o Solaris (Versões 2.6 e acima). O agente é extensível.

**Desvantagens** Mínimo; tem suporte somente para a MIB-II.

---

### Concord SystemEDGE

<http://www.empire.com>

**Plataformas** Alguns tipos de Unix, Windows NT

**Vantagens** Fornece informações muito detalhadas sobre o sistema (CPU, espaço de disco, sistemas de arquivos, aplicativos instalados etc.). Integra-se ao serviço de SNMP do Windows NT. Rastreador de registros no Unix e NT. O agente é totalmente extensível. Funciona com o pacote Network Health da Concord e com a suíte TREND da Trinagy.

**Desvantagens** Pode ser dispendioso, exceto se adquirido em grande quantidade.

---

### Microsoft

<http://www.microsoft.com>

**Plataformas** Windows 9x/NT/2000

**Vantagens** Incorporado ao kernel do sistema operacional. Pode ser controlado pelos serviços do NT.

**Desvantagens** Atende apenas aos requisitos mínimos de um agente compatível com o SNMP. Instale o service pack mais recente após instalar o software.

---

### Net-SNMP\*

<http://net-snmp.sourceforge.net>

**Plataformas** Alguns tipos de Unix, Windows 9x/NT

**Vantagens** Grátis e bastante robusto. Facilmente extensível usando scripts de shell ou de Perl. Inclui um daemon de trap.

**Desvantagens** A documentação é mínima, o que significa que pode ser difícil para os principiantes fazê-lo executar como desejam.

---

\* Antigo projeto UCD-SNMP.

**Plataformas** Unix, Windows NT

**Vantagens** Kit de ferramentas eficiente para escrever um agente, se essa for a funcionalidade que você procura.

**Desvantagens** Não se integra ao Windows SNMP Service. Em sua grande maioria, um produto de kit de ferramentas; requer muito trabalho para ser útil.

## Suites de NMS

Empregamos o termo “suite” no contexto de um pacote de software que empacota vários aplicativos em um único produto prático. Nesta seção, discutiremos o software da NMS, que é uma das partes mais importantes do cenário de gerenciamento de rede. Sem esse pacote, o software do agente na seção anterior é praticamente inútil. Os produtos de NMS permitem uma visão geral em rede de seus servidores, roteadores, comutadores etc. Na maioria dos casos, essa visão é uma representação gráfica da rede, com vários rótulos e ícones interessantes. Esses pacotes são bastante configuráveis e funcionam em quase todos os ambientes de rede. Entretanto, aliados a essa liberdade, vem uma etiqueta de preço alto e um processo de configuração confuso. Alguns produtos estão mais voltados para o lado rede do gerenciamento (por exemplo, dispositivos como roteadores, hubs e comutadores). Outros avançam um pouco mais e permitem personalizar agentes de servidores e estações de trabalho para se integrarem sem problemas às suas NMSs. Lembre-se de que os pacotes maiores são destinados às redes mais complexas e de grande porte, e exigem muito treinamento. Procure pesquisar os pacotes antes de comprá-los; se possível, obtenha versões demo. O restante desta seção lista alguns dos pacotes mais comuns de NMS.

---

### HP OpenView NNM

<http://www.openview.hp.com>

**Plataformas** Solaris, HP-UX, Windows NT/2000

**Vantagens** Excelente suite de SNMP para empresa de médio e grande porte. Embora possa ser complexo, é gerenciável com pouca ajuda do suporte da OpenView. Possui um belo mapa gráfico e sistema de monitoração de eventos. Pode fazer análise de tendências de dados de histórico. O preço parece justo e é possível um desconto, obtendo-se uma licença para um número limitado de nós gerenciados.

**Desvantagens** Não existem muitos plug-ins disponíveis de aplicativos de terceiros.

---

<b>Plataformas</b>	Solaris, HP-UX, Windows NT/2000
<b>Vantagens</b>	Se você é uma das empresas listadas pela publicação Fortune 500 e procura implementar o OpenView em esteróides, o ITO é o produto adequado para você. Esse produto é voltado para o usuário. Mapas, eventos e muito mais podem ser exibidos ou omitidos de acordo com o perfil do usuário. O sistema de eventos é mais parecido com uma bilheteria. Estão disponíveis diversos “plug-ins inteligentes” de terceiros.
<b>Desvantagens</b>	O preço pode ser muito alto. Elaborado para empresas de grande porte. Não existem muitas pessoas preparadas para essa implementação, sem treinamento ou sem ajuda de consultoria.

---

<b>Tivoli Netview</b>	<a href="http://www.tivoli.com/products/index/netview/">http://www.tivoli.com/products/index/netview/</a>
<b>Plataformas</b>	OS/390, Solaris, AIX, Digital UNIX, Windows NT (Intel e Alpha)
<b>Vantagens</b>	Uma solução de gerenciamento de redes verdadeiramente distribuídas, que pode detectar problemas na origem, antes de afetarem os usuários.
<b>Desvantagens</b>	É um sistema de gerenciamento muito pesado que exige alto investimento e muitos recursos para a implementação e o funcionamento.

---

<b>Castle Rock SNMPc</b>	<a href="http://www.castlerock.com">http://www.castlerock.com</a>
<b>Plataformas</b>	Windows 98/NT/2000
<b>Vantagens</b>	Excelente para as empresas de pequeno e médio porte. Contém tudo o que você precisa para ter uma NMS em funcionamento e em execução, em seu ambiente operacional. O preço é bastante razoável e inclui os recursos.
<b>Desvantagens</b>	O mapa da rede poderia ser um pouco mais elaborado. Não oferece uma representação realística de sua rede.

---

<b>BMC</b>	<a href="http://www.bmc.com">http://www.bmc.com</a>
<b>Plataformas</b>	Várias plataformas, inclusive Unix e Windows NT
<b>Vantagens</b>	A BMC desenvolveu bases de conhecimento para gerenciar a maioria dos aspectos da empresa, incluindo redes, bancos de dados e servidores
<b>Desvantagens</b>	Os módulos de conhecimento são úteis mas proprietários. O custo tende a permanecer alto. Não usa o SNMP como linguagem nativa.

## Computer Associates Unicenter TNG Framework

<http://www.cai.com>

**Plataformas** Unix, Windows NT/2000

**Vantagens** Pode ajudar a gerenciar a empresa de IT completa – tudo, desde o gerenciamento de redes tradicionais até o sistema de banco de dados Oracle.

**Desvantagens** Outro sistema de gerenciamento pesado, cuja implementação pode consumir muito tempo, recursos e dinheiro.

---

## Veritas NerveCenter

<http://www.veritas.com>

**Plataformas** Solaris, HP-UX, Windows NT

**Vantagens** Usa modelos de comportamento (máquinas de estado finito) para modelar situações de redes do mundo real. O NerveCenter é elaborado para ser um mecanismo de polling independente ou para ser usado em conjunto com o mapa gráfico do OpenView. As subrotinas da linguagem Perl podem ser compiladas no mecanismo de polling para uso posterior.

**Desvantagens** Consome mais esforços de manutenção do que o OpenView e seu funcionamento tende a ser mais complexo.

---

## OpenRiver

<http://www.riversoft.com>

**Plataformas** Solaris

**Vantagens** A RiverSoft, empresa desenvolvedora do OpenRiver, orgulha-se em afirmar que sua NMS fornece “gerenciamento de rede sem intervenção”. Também oferecem um verdadeiro reconhecimento de redes das camadas 2 e 3. Apesar dos surpreendentes recursos do produto, o preço é bastante razoável.

**Desvantagens** Disponível atualmente para o Solaris (embora a RiverSoft esteja planejando uma versão Windows NT).

---

## GxSNMP

<http://www.gxsnmp.org>

**Plataformas** Qualquer plataforma Unix com um compilador ANSI C e o kit de ferramentas GTK/GDK instalado

**Vantagens** Esta NMS gratuito é fornecido com diversos recursos excelentes, como uma ferramenta de mapeamento (não de reconhecimento automático) e integração com a linguagem SQL.

**Desvantagens** Este projeto ainda está engatinhando (mas há vários recursos planejados que o tornarão uma solução robusta de NMS).

**Plataformas** A maioria das plataformas Unix, Windows NT

**Vantagens** O Tkined é uma plataforma gratuita e extensível de gerenciamento de redes, que oferece um mapa da rede e ferramentas para detectar redes de IP. Além disso, gerencia dispositivos com padrões compatíveis e não-compatíveis com o SNMP (*ping, traceroute* etc.). O Tcl é usado para estender e adicionar funcionalidade ao Tkined.

**Desvantagens** É necessário conhecer o Tcl para estender este pacote.

**Plataformas** Qualquer plataforma com suporte para o Java

**Vantagens** OpenNMS é uma tentativa de fornecer aos usuários um serviço totalmente aberto e um produto de gerenciamento de rede. É escrito em Java e liberado sob a GNU Public License (GPL). Tem suporte para reconhecimento de redes e polling distribuída, entre outros recursos.

**Desvantagens** Este projeto ainda está engatinhando.

## *Gerenciadores de componentes (gerenciamento específico do fornecedor)*

Esses pacotes de software estão voltados para determinado tipo de fornecedor ou função; por exemplo, um gerenciador de componentes pode ser um produto para gerenciar um rack de modem. Antes de adquirir um pacote assim, analise detalhadamente seu ambiente atual, se existe alguma probabilidade de crescimento e quais fornecedores você costuma utilizar ou provavelmente usará no futuro. Como alguns desses produtos são específicos do fornecedor, é fácil comprar algo que se tornará menos útil do que você esperava. Por exemplo, o CiscoView (parte da suíte CiscoWorks) é uma excelente peça de software; faz muitas coisas interessantes, como exibir a parte traseira dos roteadores. Mas se você comprar alguns dos dispositivos da Nortel meses depois da instalação desse produto, ele não conseguirá apresentar uma visão unificada de sua rede. Alguns pacotes realmente permitem gerenciar os equipamentos de seus concorrentes; por exemplo, um gerenciador de componentes que monitora comutadores pode lidar com os comutadores de concorrentes. Antes de comprar qualquer um desses produtos, pesquise sua rede e faça indagações sobre os recursos dos produtos. O restante desta seção listará alguns gerenciadores de componentes disponíveis.

---

## Sun Management Center

<http://www.sun.com/symon/>

**Plataformas** Solaris, Windows (camada Console)

**Vantagens** Fornece um único ponto de gerenciamento para todos os servidores, desktop e sistemas de armazenamento da Sun, para o ambiente operacional, aplicativos e serviços de centrais de dados da Solaris. Este produto é dimensionável para milhares de sistemas em uma única plataforma de gerenciamento unificável e se integra facilmente às principais plataformas de terceiros, oferecendo mais flexibilidade. Também pode obter desempenho do sistema em tempo real e oferece uma suíte gratuita de diagnóstico de hardware (plug-in) que detecta falhas de hardware.

**Desvantagens** Embora possa gerenciar e monitorar outros fornecedores, essa possibilidade não é muito fácil.

---

## CiscoWorks 2000

<http://www.cisco.com>

**Plataformas** Solaris, HP-UX, AIX, Windows NT para alguns

**Vantagens** Esta suíte permite fazer tudo, desde o controle de versão em seus arquivos de configuração até gráficos de latência e imagens detalhadas das partes traseiras dos dispositivos.

**Desvantagens** Os mapas são um pouco confusos. Não gera um instantâneo muito amistoso de sua rede e demora muito para retornar as configurações para os dispositivos. Seria ótimo se a operação de restauração fosse tão fácil quanto a de backup.

---

## 3Com Total Control

<http://www.3com.com>

**Plataformas** Solaris, Windows 9x

**Vantagens** Permite que o usuário visualize o status de um rack de modem, exibindo uma imagem do próprio rack físico – mostrando tudo, desde os parafusos até o logo. O usuário pode rearrumar placas individuais ou o chassi inteiro, entre outras possibilidades. É um produto muito engenhoso e pode ser útil ao tentar rastrear problemas do equipamento.

**Desvantagens** Como anteriormente, o chassi Total Control da 3Com nunca pode ter até 336 modems, pode ser difícil começar por aí (é necessário consultar o status de todos os modems na rack). O tempo de inicialização pode ser bastante impactado pela velocidade da rede entre você e o chassi em questão.

---

**Plataformas** Unix, Windows NT

**Vantagens** Ferramenta muito eficiente para gerenciar equipamento Cabletron, e está começando a incluir a possibilidade de gerenciar equipamentos de outros fornecedores.

**Desvantagens** Configuração e manutenção complexas. Adequados a lojas que precisam de uma plataforma sofisticada.

## Análise de tendências

Diante dos tantos problemas da rede, é válido ter algum tipo de registro histórico para se ter uma idéia de quando algo começou a dar errado. Isso permite retornar e rever o que aconteceu antes da ocorrência de um problema, e possivelmente impedi-lo de se repetir. Para ser pró-ativo em relação ao diagnóstico de problemas antes de ocorrerem, é fundamental saber o que significa "normal" em uma rede – você precisa de um conjunto de dados estatísticos de linha de base que revele como sua rede se comporta normalmente. Embora diversos pacotes maiores façam alguns relatórios de tendências, a sua utilização pode ser difícil e complexa. Talvez sequer ofereçam o tipo de informação necessária. Quando você constatar o que um sistema dedicado de análise de tendências pode fazer, descobrirá por que compensa, em termos de tempo, esforços e dinheiro, integrar um desses sistemas a seu esquema de monitoração de rede.

Se seu ambiente precisa de uma monitoração rigorosa, considere a obtenção de provas de RMON, que são uma inclusão excelente aos pacotes de análise de tendências, porque a maioria desses pacotes pode utilizar o tipo de dados obtidos por essas provas. O restante desta seção listará alguns pacotes de análise de tendências.

**Plataformas** Solaris, HP-UX, Windows NT

**Vantagens** Gráficos profissionais, baseados na Web. Oferece ao usuário a possibilidade de fazer o download e exibir relatórios em PDF. Você pode procurar na Web relatórios mais detalhados. Excelente gerenciamento de usuários que permite limitar os usuários a visualizarem o que é realmente necessário. A Concord oferece um "Network Health Checkup" (check-up da rede) gratuito. Este programa também pode interagir com provas e dispositivos baseados em servidor para gerar uma análise completa de tendências de sua rede e servidores.

**Desvantagens** É possível que algumas pessoas se surpreendam com o preço. O licenciamento é concedido por elemento.

---

**Trinagy\* TREND**

<http://www.desktalk.com>

**Plataformas** Unix, Windows 9x/NT

**Vantagens** Excelente produto para o planejamento de capacidades. Externamente, o Trinagy aceita previsões de 30, 60 e 90 dias, entre outros cálculos. O visualizador de relatórios é escrito em Java, o que o torna utilizável na maioria das plataformas. A arquitetura de relatórios é aberta, no sentido de que é possível criar relatórios exclusivos.

**Desvantagens** Requer duas semanas de treinamento para executar e administrar. O esquema de cálculo de preços é semelhante ao do eHealth, porque o tamanho do banco de dados depende da quantidade de dispositivos consultados, do tempo de permanência dos dados, etc. Você pode obter mais por menos, ao ajustar os intervalos de polling e os tempos de retenção.

---

**MRTG**

<http://www.mrtg.org>

**Plataformas** A maioria das plataformas do Unix, Windows NT

**Vantagens** Gratuito, configuração e utilização fáceis, muito bem documentado. Além dos dispositivos de polling existentes na rede, o MRTG pode receber entrada de fontes não compatíveis com o SNMP.

**Desvantagens** É necessário instalar vários pacotes, o que pode ser difícil em algumas plataformas. Por exemplo, o MRTG precisa de um módulo Perl de SNMP específico para executar todas as suas tarefas de polling. Não é muito escalável.

---

**Cricket**

<http://cricket.sourceforge.net>

**Plataformas** A maioria das plataformas do Unix

**Vantagens** Excelente ferramenta que entra em cena onde o MRTG se despede. Usa o RRDTTool, versão da próxima geração do MRTG.

**Desvantagens** O Cricket tem thread (encadeamento) simples, de modo que a coleta de dados de uma rede de tamanho médio pode ser rápida, principalmente se você obtém dados de utilização freqüentemente. Por isso, é muito eficiente, e você não enfrentará quaisquer problemas por bastante tempo.

**Plataformas** Unix, Windows NT

**Vantagens** Muito flexível e é fornecido com excelente reporting para uso imediato.

**Desvantagens** Requer conhecimento sólido de gerenciamento e programação de redes (Perl) para personalizá-lo para fazer qualquer tarefa além dos recursos fornecidos.

## Software de suporte

Software de suporte é uma mala que contém todo o tipo de recursos utilizados em conjunto com os pacotes de software listados anteriormente. Alguns desses pacotes devem ser usados para escrever aplicativos de SDNM independentes. O restante desta seção descreve vários pacotes de software de suporte. A maioria deles é distribuída gratuitamente e podem ser utilizados com pouca ou nenhuma experiência anterior.

---

**Perl** <http://www.perl.com>  
<http://www.perl.org>

**Plataformas** Unix, Windows NT, Mac OS

**Vantagens** A *Practical Extraction and Report Language* (Perl) é uma linguagem versátil de criação de scripts, ferramenta preferida dos administradores de sistema e engenheiros de rede, entre outros profissionais. O MRTG e Cricket usam a Perl para executar suas atividades dos bastidores.

**Desvantagens** Algumas pessoas afirmam que a Perl não tem desvantagens. A reclamação mais comum em relação à linguagem é o fato de ser interpretada e não compilada, como a linguagem de programação C.

---

**SNMP Support for Perl** <http://www.switch.ch/misc/leinen/snmp/perl/>  
<http://www.cpan.org>

**Plataformas** Unix, Windows NT, Mac OS

**Vantagens** Fornece sub-rotinas fáceis de usar, que dão acesso às funções básicas do SNMP. Amplamente testada, por ser o mecanismo fundamental de SNMP no pacote MRTG .

**Desvantagens** Não parece ter muita divulgação no mercado.

---

**WILMA** [http://ftp.ldv.e-technik.tu-muenchen.de/dist/WILMA/INDEX.html](ftp://ftp.ldv.e-technik.tu-muenchen.de/dist/WILMA/INDEX.html)

**Plataformas** A maioria das plataformas do Unix

**Vantagens** Contém as funções básicas do SNMP, assim como um compilador de MIBs e navegador.

**Desvantagens** As funções poderiam ser um pouco mais dinâmicas e amistosas.

---

**Net-SNMP C Library**

<http://net-snmp.sourceforge.net>

**Plataformas** Unix, Windows 9x/NT

**Vantagens** Esta biblioteca pode ser usada para desenvolver suas próprias aplicações do SNMP. A biblioteca é muito fácil de usar, depois que você a entende. O aspecto positivo do pacote é a oferta de um código-fonte para comandos como *snmpget*, *snmpset* e *snmpwalk*, que pode ser usado para constatar como essas operações são executadas.

**Desvantagens** A documentação sobre como utilizar a biblioteca é deficiente e chega a ser inexistente.

---

**Net-SNMP Perl Module**

<http://www.cpan.org/authors/id/GSM/>

**Plataformas** Unix, Windows 9x/NT

**Vantagens** Esta biblioteca fornece funcionalidade idêntica à do Net-SNMP C, exceto no Perl.

**Desvantagens** Durante a instalação, este módulo precisa ter acesso à biblioteca do Net-SNMP C para funcionar corretamente.

---

**A3Com**

<http://www.kernel.org/software/A3Com/>

**Plataformas** Unix, Windows NT

**Vantagens** Um conjunto simples de módulos que podem ser usados para gerenciar o SuperStack II 3900/9300 da 3Com e os comutadores da LAN CoreBuilder 3500. Pode ser um bom começo para o gerenciamento em um orçamento.

**Desvantagens** Funcionalidade limitada.

---

**SNMP++**

<http://rosegarden.external.hp.com/snmp++/>

**Plataformas** Unix (Linux, HP-UX e Solaris), Windows

**Vantagens** Se você precisa usar o C++ para o desenvolvimento de aplicações do SNMP, este é o pacote indicado. Você pode criar aplicações

poderosas com mínimo esforço de programação. Esta biblioteca pode ser lançada na comunidade de origens abertas e tem distribuição gratuita.

**Desvantagens** Requer conhecimento de C++.

---

Netcool

<http://www.micromuse.com>

**Plataformas** Unix, Windows NT

**Vantagens** Um mecanismo de correlação de eventos, usado para reduzir os eventos de gerenciamento que as plataformas tradicionais de NMS tendem a gerar, apresentando ao usuário final somente o que esse usuário precisa saber para corrigir os problemas da rede. Elaborado para receber eventos de NMSs, como o OpenView ou NerveCenter, mas receber eventos de praticamente todo tipo de fonte de gerenciamento. A Micromuse comercializa provas que servir de interface para tudo, desde as conhecidas plataformas de NMS até equipamentos de centrais telefônicas.

**Desvantagens** Requer um pouco de configuração inicial (mas depois disso, a utilização e manutenção são fáceis).

---

Network Computing Technologies Trap Receiver

<http://www.ncomtech.com>

**Plataformas** Windows 95/NT

**Vantagens** Fácil de usar e pode ser configurada para executar ações por ocasião de recebimentos de traps.

**Desvantagens** Não é executado em qualquer tipo de Unix.

# 6

## Configurando a NMS

Após selecionar alguns softwares para utilizar em seu ambiente, chegou o momento de discutir como instalar e executá-lo. Neste capítulo, examinaremos minuciosamente alguns pacotes de NMS. Embora tenhamos listado alguns pacotes no Capítulo 5, analisaremos apenas alguns deles aqui, e usaremos esses pacotes em exemplos no restante deste livro. Esses exemplos devem permitir tirar o máximo proveito de outros pacotes de gerenciamento de rede baseado no SNMP em funcionamento e em execução, com bem menos esforço.

### *Network Node Manager do OpenView, da HP*

O Network Node Manager (NNM) é um software licenciado. O pacote dispõe de um recurso denominado “Instant-On” que permite utilizar o produto por tempo limitado (60 dias) enquanto você aguarda a chegada da licença real. Nesse período, você estará restrito a uma licença de 250 nós gerenciados, mas os recursos do produto não estão limitados, em hipótese alguma. Quando você instalar o produto, a licença do Instant-On será habilitada, por default.



Verifique os scripts do OpenView localizados no diretório *bin* do OpenView (normalmente, */opt/OV/bin*). Um grupo particularmente importante de scripts define variáveis de ambiente que permitem percorrer com muito mais facilidade a estrutura de diretórios do OpenView. Esses scripts são denominados *ov.envvars.csh*, *ov.envvars.sh* etc. (isto é, *ov.envvars* seguido do nome da shell sendo utilizada). Quando você executar o script adequado à sua shell, ele definirá variáveis de ambiente, como *\$OV\_BIN*, *\$OV\_MAN* e *\$OV\_TMP*, que indicam os diretórios *bin*, *man* e *tmp* do OpenView. Assim, é possível alternar facilmente para o diretório contendo as páginas do manual do OpenView, com o comando *cd \$OV\_MAN*. Essas variáveis de ambiente serão utilizadas neste livro inteiro e em toda a documentação do OpenView.

## Executando o NNM

Para iniciar a GUI do OpenView em uma máquina do Unix, defina sua variável de ambiente DISPLAY e execute o comando `$OV_BIN/ovw`. Esse procedimento iniciará o NNM do OpenView. Se o NNM de seu sistema executou qualquer detecção, os nós encontrados devem aparecer no ícone Internet (nível superior). Se ocorrerem problemas ao iniciar o NNM, execute o comando `$OV_BIN/ovstatus -c` e, em seguida, `$OV_BIN/ovstart` ou `$OV_BIN/ovstop`, respectivamente, para iniciar ou interrompê-lo. Por default, o NNM instala os scripts necessários para iniciar os respectivos daemons quando a máquina inicializar. O OpenView executará todas as suas funções em segundo plano, mesmo que você não esteja executando quaisquer mapas. Isso significa que você precisa manter uma cópia do NNM em execução no console, o tempo todo, nem é necessário iniciá-lo explicitamente quando a máquina reiniciar.

Quando a GUI for inicializada, apresentará um mapa de alto nível clicável. Esse mapa, denominado mapa Root, oferece uma visão do nível inicial de sua rede. O mapa exibe sua rede sem apresentar cada detalhe de uma só vez. Para obter mais informações sobre qualquer item exibido, seja uma subrede ou um nó individual, clique no item em questão. Você pode fazer uma expansão para ver qualquer nível de detalhamento desejado – por exemplo, você pode examinar uma placa de interface em um nó específico. Quanto mais detalhes você desejar, tanto mais deverá clicar. A Figura 6-1 mostra um mapa característico do NNM.

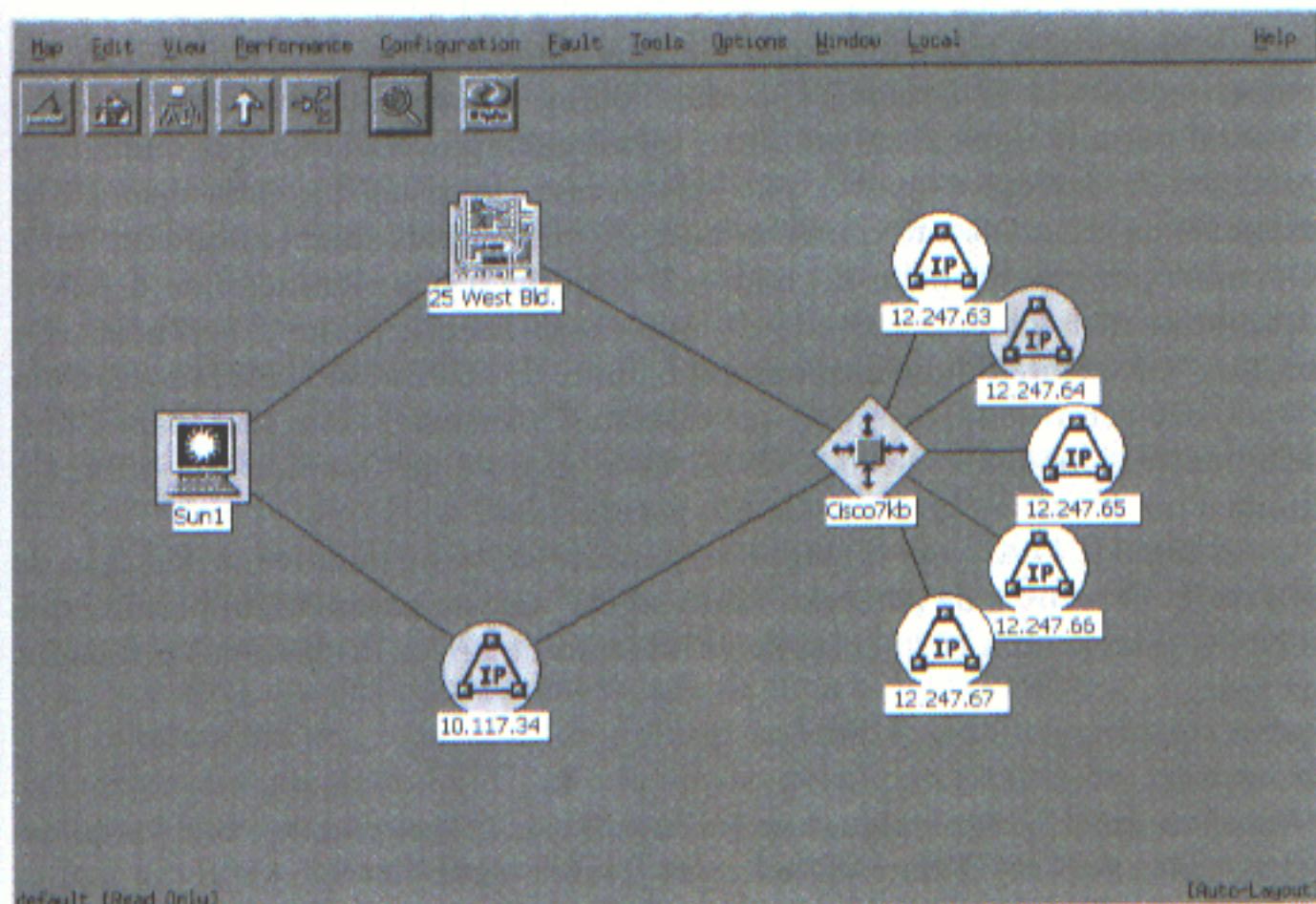


Figura 6-1 Um mapa característico do NNM

A barra de menus (ver Figura 6-2) permite percorrer o mapa com mais facilidade. Existem opções como fechar o NNM (botão mais à esquerda), saltar imediatamente para o mapa Home (segundo a partir da esquerda),\* o mapa Root (terceiro à esquerda), o mapa pai ou mapa anterior (quarto à esquerda) ou o navegador rápido.\*\* Também existe um botão que permite deslocar a perspectiva do mapa ou aplicar mais zoom em parte desse mapa.

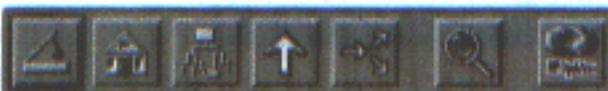


Figura 6-2 Barra de menus do NNM do OpenView



Para evitar que você adoeça ao examinar a rede que você acabou de descobrir, lembre-se de que é possível adicionar algumas adaptações rápidas e fáceis, que transformarão essa panacéia de nomes, números e ícones em uma perspectiva coordenada da rede.

### Processo netmon

O processo daemon do NNM (*netmon*) inicia automaticamente quando o sistema inicializa e é responsável por revelar os nós de sua rede, além de algumas outras tarefas. No menu do NNM, vá para “Options → Network Polling Configurations: IP”. É exibida uma janela semelhante à da Figura 6-3.

A Figura 6-3 mostra a seção General do assistente de configuração. As outras seções são IP Discovery, Status Polling e Secondary Failures. A seção General permite especificar um filtro (neste exemplo, NOUSERS) que controla o processo de detecção – talvez não seja necessário visualizar cada dispositivo existente na rede. Discutiremos a criação de filtros, mais adiante neste capítulo. Optamos por revelar além do limite da licença, o que significa que o NNM descobrirá mais objetos na rede do que nossa licença permite gerenciar. Os objetos “excedentes” (que ultrapassam o limite da licença) são colocados em um estado não gerenciado, de modo que é possível visualizá-los em seus mapas, mas sem controlá-los por meio do NNM. Essa opção é útil quando sua licença o limita a um número específico de nós gerenciados.

A seção IP Discovery (Figura 6-4) permite ativar ou desativar a detecção de nós de IP. O recurso de detecção “auto adjust” (ajuste automático) permite que o NNM calcule uma freqüência de verificação de novos dispositivos instalados na rede.

\* Você pode definir qualquer mapa como seu mapa Home. Quando você descobrir o mapa que deseja utilizar, entre em “Map → Submap → Set This Submap as Home”.

\*\* Este é um mapa especial em que é possível colocar os objetos que devem ser observados freqüentemente. Permite acessar os objetos rapidamente, sem precisar localizá-los pesquisando o mapa da rede.

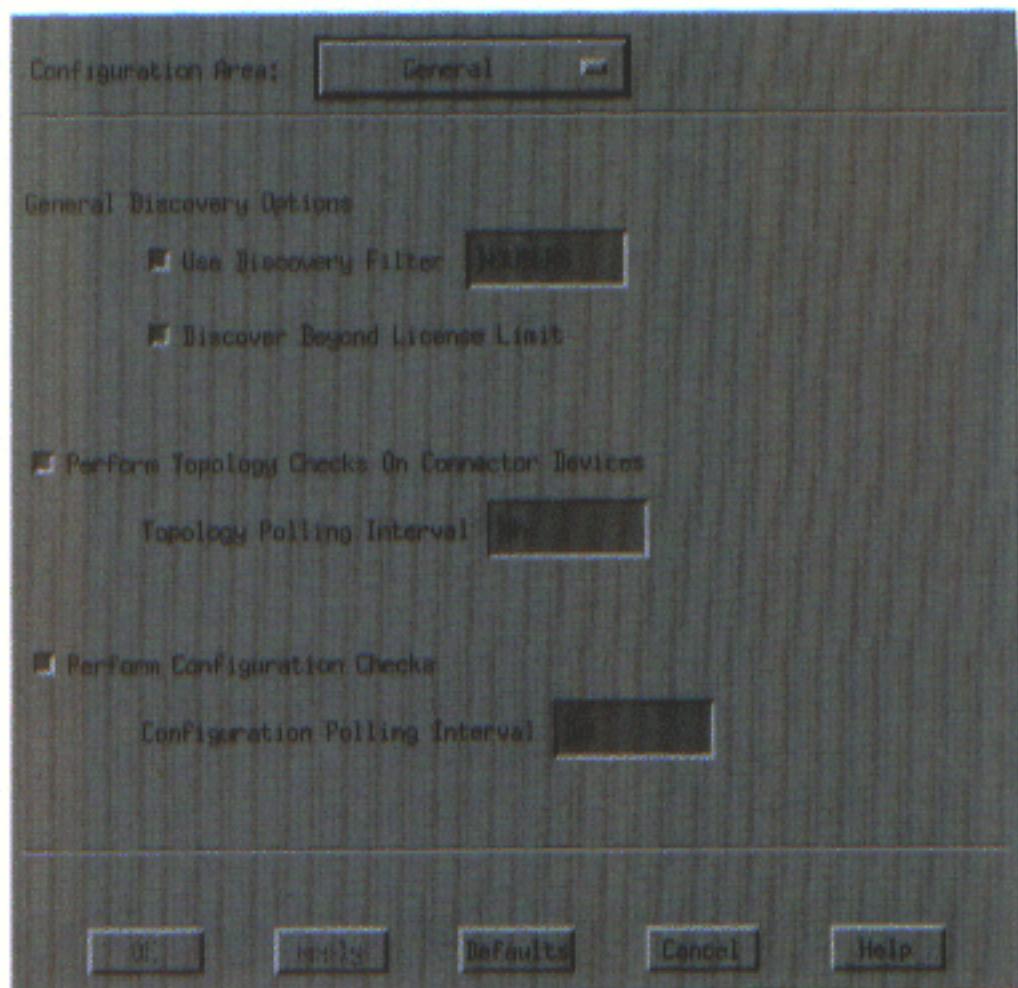


Figura 6-3 Opções de configuração de polling da rede, da seção General do OpenView

Quanto maior a quantidade de dispositivos novos detectados, tanto mais freqüente será a polling; se não forem encontrados quaisquer dispositivos novos, o processo será retardado, esperando ocasionalmente um dia (1d) para verificar a presença de dispositivos novos. Se você não aprova a idéia de variar os intervalos de pesquisa (ou talvez mais dentro da realidade, se você acha que verificar a presença de novos dispositivos na rede consumirá mais recursos do que você gostaria, seja em sua estação de gerenciamento de rede, seja na própria rede), especifique um intervalo de detecção fixo. Finalmente, o botão "Discover Level-2 Objects" instrui o NNM a descobrir e relatar os dispositivos pertencentes à segunda camada do modelo de rede OSI. Essa categoria inclui elementos como hubs e comutadores não gerenciados, alguns dispositivos AppleTalk e outros.

A Figura 6-5 mostra a seção de configuração Status Polling. Aqui, é possível ativar ou desativar a polling de status e excluir os nós inativos ou inacessíveis por um período de tempo especificado. Este exemplo está configurado para excluir nós após um período de inatividade de uma semana (1w).

Evidentemente, as opções de polling de DHCP são úteis principalmente em ambientes que usam DHCP. Essas opções permitem estabelecer um relacionamento entre o comportamento da polling e os endereços IP. Você pode especificar um filtro que selecione endereços atribuídos pelo DHCP. A partir de então, é possível definir um tempo, após o qual o *netmon* excluirá os endereços DHCP sem resposta dos respectivos mapas em sua rede.

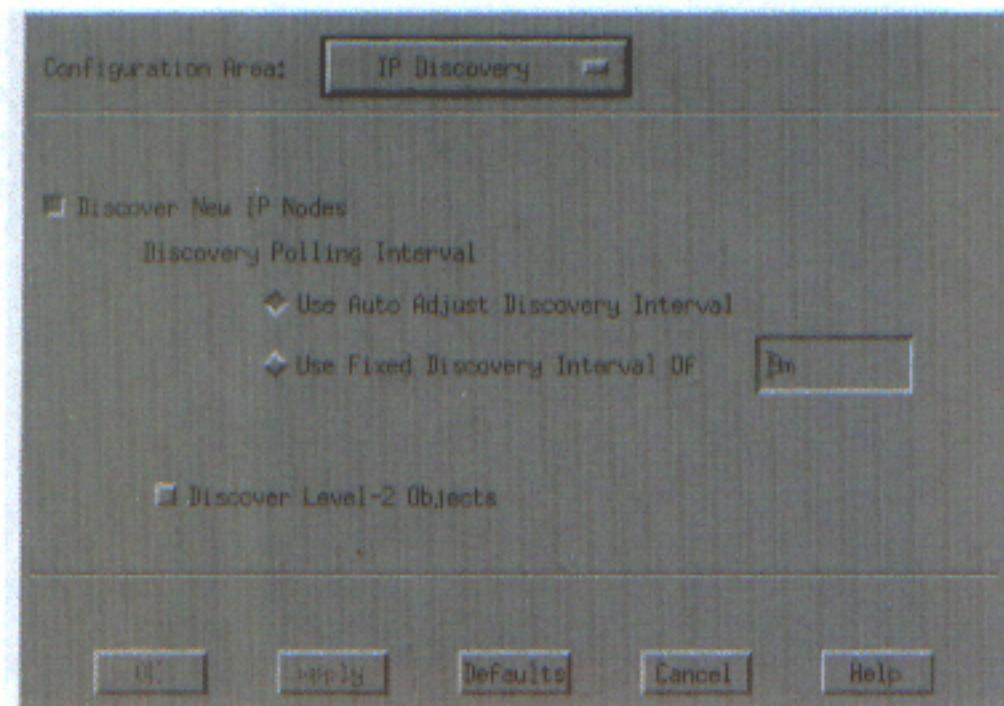


Figura 6-4 Opções de configuração de polling de rede, da seção IP Discovery no OpenView

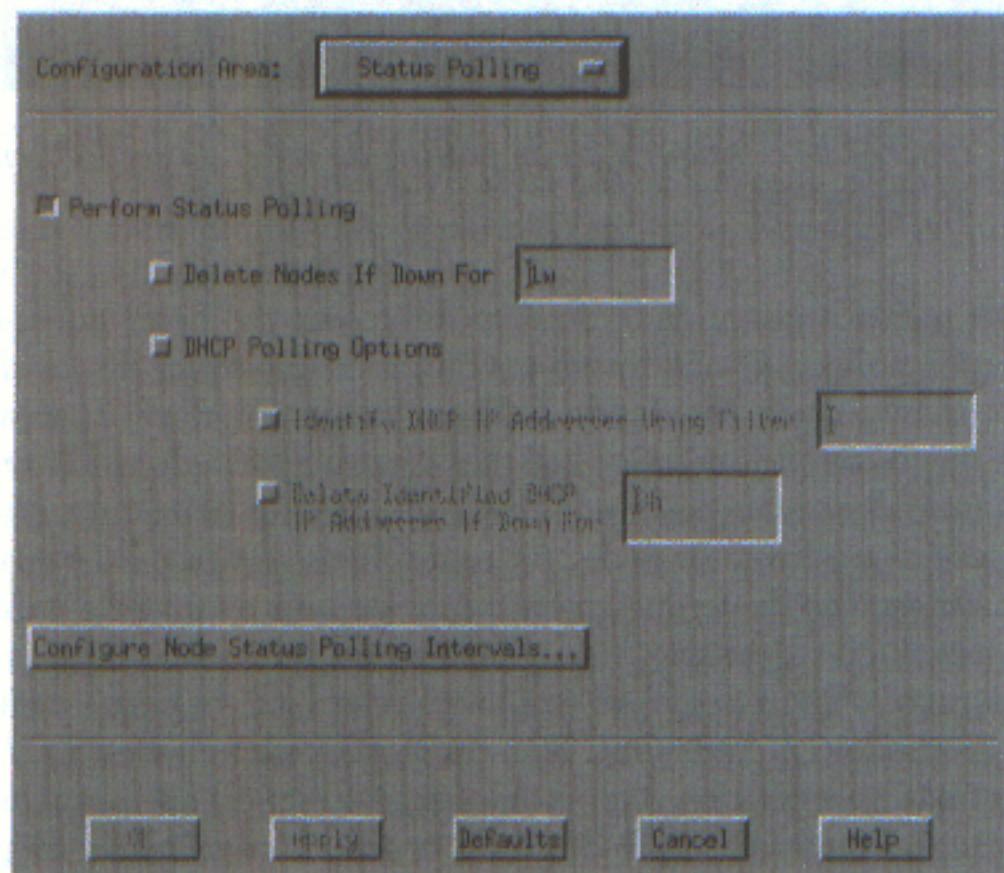


Figura 6-5 Opções de configuração de polling de rede, da seção Status Polling do OpenView

Se um dispositivo permanecer inativo por um período de tempo específico, o *netmon* desassociará o nó e o endereço IP. O motivo desse comportamento é simples: em um ambiente de DHCP, o desaparecimento de um endereço IP geralmente indica que o nó recebeu um novo endereço IP de um servidor de DHCP.

Nesse caso, continuar pesquisando o endereço antigo é um desperdício de esforços e, possivelmente, um ledo engano porque o endereço pode estar reatribuído a outro host.

Finalmente, a seção de configuração Secondary Failures, mostrada na Figura 6-6, permite informar ao pesquisador como reagir a detectar uma falha secundária. Isso ocorre quando um nó além de um dispositivo com falha está inacessível; por exemplo, quando um roteador fica inativo, tornando inacessível o servidor de arquivos conectado por meio de uma das interfaces do roteador. Nessa seção de configuração, é possível declarar a exibição/omissão de alarmes para as falhas secundárias. Se você preferir suprimi-los, poderá configurar um filtro que identifique os nós importantes da rede, que não sejam suprimidos mesmo se considerados com falhas secundárias.

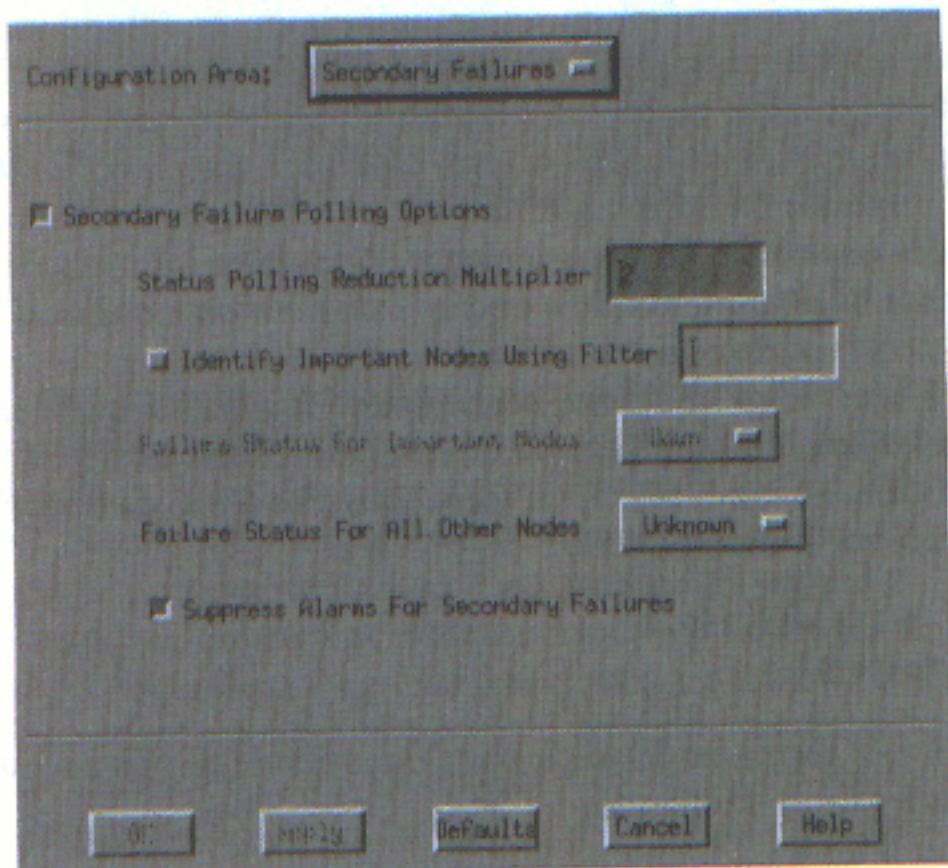


Figura 6-6 Opções de configuração de polling de rede, da seção Secondary Failures do OpenView

Assim que seu mapa estiver em funcionamento, você observará que nada é detectado. Inicialmente, o *netmon* não revelará nada além do segmento de rede ao qual sua NMS está acoplada. Se sua NMS tiver um endereço IP 24.92.32.12, você não descobrirá seus dispositivos em 123.67.34.0. O NNM encontrará os roteadores adjacentes e os respectivos segmentos, desde que sejam compatíveis com o SNMP, e os colocará em um estado não gerenciado (cor esmaecida) no mapa.\* Isso indica que tudo o que estiver dentro e sob este ícone não será verificado nem revelado.

\* No NNM, entre em "Help → Display Legend", para obter uma lista dos ícones e suas cores.

Selecionar o ícone e entrar em “Edit → Manage Objects” instrui o NNM a começar a gerenciar essa rede e permite que o *netmon* comece a revelar os nós existentes em seu interior. Sempre que necessário, você pode encerrar o gerenciamento de nós, clicando em “UnManage” em vez de em “Manage”.

Se os roteadores não apresentarem quaisquer redes adjacentes, experimente testá-los com “Fault → Test IP/TCP/SNMP”. Adicione o nome de seu roteador, clique em “Restart” e examine o tipo de resultados retornados. Se você receber “OK except for SNMP” (OK, exceto quanto ao SNMP), revise o Capítulo 7 e leia a próxima, sobre a configuração de nomes de comunidade padrão no OpenView.

Além disso, o *netmon* permite especificar um arquivo-semente que ajude a descobrir objetos mais rapidamente. O arquivo-semente contém endereços IP individuais, faixas de endereços IP ou nomes de domínio que limitem o âmbito dos hosts revelados. É possível criar o arquivo-semente em qualquer editor de texto – basta colocar um endereço e um nome de host em cada linha. Às vezes, faz mais sentido colocar os endereços de seus gateways no arquivo-semente, uma vez que os gateways mantêm tabelas ARP para a rede. O *netmon* revelará posteriormente todos os outros nós da rede, evitando, assim, a necessidade de adicionar todos os host ao arquivo-semente. Para obter informações mais pertinentes, consulte na documentação a opção *-s* do *netmon* e os Local Registration Files (LRF – arquivos de registro locais).

O NNM tem outro utilitário, denominado *loadhosts*, que permite um nó de cada vez ao mapa. Eis um exemplo de como adicionar hosts, de forma relativamente aleatória, ao mapa do OpenView. Observe o uso da opção *-m*, que define a subrede com 255.255.255.0:

```
$ loadhosts -m 255.255.255.0  
10.1.1.12 gwrouter1
```

Quando você terminar de adicionar a quantidade de nós necessária, pressione Ctrl-d para sair do comando.

## Configurando intervalos de polling

A página SNMP Configuration está localizada fora da tela principal, em “Options → SNMP Configuration”. É exibida uma janela semelhante à da Figura 6-7, com quatro seções: Specific Nodes, IP Address Wildcards, Default e a seção de entrada (que não aparece por questões de visualização). Cada seção possui as mesmas áreas gerais: Node ou IP Address, Get Community, Set Community, Proxy (se aplicável), Timeout, Retry, Port e Polling. A área Default, posicionada infelizmente no final da tela, configura o comportamento padrão para o SNMP na rede – ou seja, o comportamento (strings de comunidade etc.) de todos os host não listados como “nós específicos” ou que correspondem a um dos caracteres curingas. A seção Specific Nodes permite especificar as exceções, com base em cada nó. IP Address Wildcards permite configurar propriedades para uma faixa de endereços. Esse recurso é útil principalmente para as redes que possuem

diferentes nomes de comunidade de *get* e *set*.<sup>\*</sup> Todas as áreas permitem especificar um tempo limite em segundos e um valor de novas tentativas. O campo Port permite inserir outro número de porta (a porta default é a 161). Polling é a frequência de verificação de seus nós.

The screenshot shows the 'SNMP Configuration' window with three main sections:

- Specific Nodes:** A table with one row for node 208.166.230.1. It has columns for Node, Get Community, Set Community, Proxy, Timeout, Retry, Port, and Polling. Values: Get Community is 'netrepublic'; Set Community is empty; Proxy is '<none>'; Timeout is 0.9; Retry is 2; Port is 161; Polling is 2m.
- IP Wildcard:** A table with two rows. The first row has IP 203.128.4.\* with Get Community 'public' and Set Community empty. The second row has IP 202.101.231.\* with Get Community 'public' and Set Community empty. Both rows have Proxy '<none>', Timeout 2.0, Retry 3, Port 161, and Polling 6m.
- Default:** A table with two rows. The first row is 'Default' with Get Community empty, Set Community empty, and Proxy 'Default'. The second row is 'Global Default' with Get Community 'public' and Set Community empty. Both rows have Timeout 0.6, Retry 2, Port 161, and Polling 4.30m.

Figura 6-7 Página SNMP Configuration do OpenView

É importante entender como funcionam os tempos limite e as novas tentativas. Se examinarmos Specific Nodes, veremos um Timeout (tempo limite) de 0,9 segundos e uma Retry (nova tentativa) especificada com o valor 2 para 208.166.230.1. Se o OpenView não receber uma resposta em 0,9 segundo, tentará novamente (a primeira tentativa) e aguardará 1,8 segundos. Se continuar sem qualquer resposta, dobrará novamente o período de tempo de espera (tempo limite) para 3,6 segundos (a segunda tentativa); se nada for retornado, ele declarará o nó inacessível e pintará esse nó com vermelho no mapa do NNM. Com esses valores de Timeout e Retry, são necessários cerca de 6 segundos para identificar um nó inatingível.

Imagine o que aconteceria se ocorre um Timeout de 4 segundos e uma Retry de 5. Na quinta tentativa, estaríamos esperando 128 segundos e o processo total demoraria 252 segundos. Isso significa mais de 4 minutos! Para um dispositivo de missão crítica, 4 minutos representam um tempo enorme para uma falha permanecer desapercebida.

Este exemplo mostra que você deve ser criterioso em suas definições de Timeout e Retry – principalmente na seção Default, porque essas definições são aplicáveis à maior parte de sua rede. Definições muito altas de Timeout e Retry e períodos de polling extremamente curtos farão com que o *netmon* fique muito lento; será necessário começar novamente antes que o poller (pesquisador) veri-

\* Esses nomes de comunidade são utilizados em partes diferentes do NNM. Por exemplo, ao pesquisar um objeto com o *xnmbsweser*, você não precisar inserir (ou lembrar) a string de comunidade se esse objeto (ou a respectiva rede) estiver definido nas configurações do SNMP.

fique todos os seus dispositivos.\* Esse é um problema freqüente quando existem vários nós, redes lentas, pouco tempo de polling e números altos de Timeout e Retry.\*\* Quando um sistema entra em retardamento, demora muito tempo para descobrir problemas ocorridos nos dispositivos que ele está monitorando no momento, assim como para detectar novos dispositivos. Em alguns casos, é possível que o NNM sequer detecte problemas nos dispositivos paralisados! Se os valores de Timeout e Retry estiverem definidos inadequadamente, não será possível localizar os problemas nem reagir às interrupções.

Ficar para trás pode ser muito frustrante. Recomendamos iniciar o período de Polling com um valor alto e continuar trabalhando até as operações se normalizarem. Dez a vinte minutos é um bom ponto de partida para o período de Polling. Na fase de teste inicial, é sempre possível definir um intervalo cura para os servidores de teste, etc.

### *Algumas considerações sobre as cores do mapa do NNM*

Nesse momento, a detecção deve estar ocorrendo e você deve estar começando a ver alguns objetos novos em seu mapa. Você deve ver uma correlação entre as cores desses objetos e as cores de Event Categories (categorias de eventos) do NNM (consulte o Capítulo 10 para obter mais informações sobre Event Categories). Se um dispositivo estiver acessível via *ping*, sua cor será verde. Se não for possível alcançar o dispositivo, ele se tornará vermelho. Se algo “subordinado” ao dispositivo falhar, o dispositivo se tornará verde esmaecido, indicando que o dispositivo em si está funcionando, mas algo subordinado entrou em um status não normal. Por exemplo, um roteador pode estar funcionando mas um servidor da Web na LAN de suporte pode ter falhado. A origem do status de um objeto como esse é Compound ou Propagated. (Os outros tipos de origem de status são Symbol e Object.) A origem de status Compound é um excelente método para saber se existe um problema em um nível inferior, e ainda manter o controle de tudo. Esse status chama a sua atenção para o problema e permite começar a expandir até alcançar o objeto aprisionado.

É sempre divertido desligar ou desconectar uma máquina e observar o ícone dessa máquina se avermelhando no mapa. Pode ser uma boa maneira de mostrar a importância do novo sistema de gerenciamento a seu chefe. Você também pode aprender a ludibriar e a fazer o OpenView perder um dispositivo, mesmo que desconectado. Com um intervalo de polling relativamente longo, é fácil desconectar um dispositivo e reconectá-lo antes de o OpenView detectar a presença do dispositivo. Quando o OpenView der conta da situação, o nó estará no lugar e parecendo perfeito. Os intervalos longos de polling facilitam a perda dessas falhas temporárias. Os intervalos mais curtos de polling reduzem a pro-

\* Lembre-se de que o mapa da maioria dos NNMs é pesquisada com os conhecidos *pings* e não por SNMP.

\*\* Procure na página do manual do *netmon* a opção *-a*, principalmente perto de *-a12*. Você pode tentar executar o *netmon* com *-a!?*, que listará as opções *-a* válidas. Se você encontrar quaisquer números negativos em *netmon.trace* após executar o *netmon -a12*, seu sistema estará retardando.

babilidade de o OpenView deixar passar alguma coisa, e aumentam a chance de o *netmon* ficar para trás e, por sua vez, ignorar outras falhas. Use etapas gradativas de modo a não derrubar nem sobrecarregar o *netmon* nem a rede.

## Usando filtros do OpenView

Seu mapa pode incluir alguns dispositivos de que você não necessita, que não deseja nem são relevantes. Por exemplo, você pode não querer pesquisar nem gerenciar PCs dos usuários, principalmente se existirem muitos usuários e uma licença limitada. Talvez compense ignorar esses dispositivos do usuário para abrir mais compartimentos para o gerenciamento de servidores, roteadores, comutadores e outros dispositivos mais importantes. O *netmon* dispõe de um mecanismo de filtragem que permite controlar com exatidão os dispositivos gerenciados. Esse mecanismo permite expurgar os dispositivos indesejados, limpa seus mapas e pode reduzir o volume de tráfego de gerenciamento na rede.

Neste livro, avisamos várias vezes que uma polling incorreta em sua rede pode gerar um tráfego muito volumoso de gerenciamento. Isso ocorre quando as pessoas ou os programas usam os intervalos de polling default, que são extremamente rápidos para a manipulação da rede ou dos dispositivos da rede. Por exemplo, um sistema de gerenciamento poderia pesquisar cada nó em sua rede 10.1.0.0 – possivelmente, milhares deles – a cada dois minutos. A polling pode consistir em solicitações de *get* ou *set* do SNMP, *pings* isolados ou ambos. O NNM do OpenView usa uma combinação dessas possibilidades para detectar e um nó está funcionando e em execução. A filtragem evita que você tenha que selecionar entre diversos nós sem uso e reduz a carga na rede. O uso de um filtro permite manter os nós críticos da rede na visualização e permite que você pesquise os dispositivos desejados e ignore aqueles nos quais você não tem interesse. A última coisa que poderia lhe acontecer seria você receber uma notificação sempre que um usuário desligasse o PC, ao término do expediente.

Os filtros também ajudam o gerenciamento de rede, permitindo excluir usuários do DHCP da detecção e polling da rede. O DHCP e BOOTP são utilizados em alguns ambientes para gerenciar grandes grupos de endereços IP. Embora sejam úteis, esses protocolos podem transformar o gerenciamento da rede em um pesadelo, uma vez que geralmente é difícil saber o que está ocorrendo quando os endereços estão sendo atribuídos, desalocados e reciclados.

Em meu ambiente, utilizamos apenas o DHCP para nossos usuários. Todos os servidores e impressoras possuem endereços IP codificados por hardware. Em nossa configuração, podemos especificar todos os clientes do DHCP e, em seguida, declarar que desejamos tudo *exceto* esses clientes em nossa detecção, mapas etc. O exemplo a seguir deve permitir que a maioria dos usuários trabalhe com uma filtragem eficaz. Invista algum tempo examinando o manual “A Guide to Scalability and Distribution for Network Node Manager” do OpenView, para obter informações mais minuciosas sobre a filtragem. O arquivo de filtro default, localizado em \$OV\_CONF/C, está dividido em três seções:

- Sets
- Filters
- FilterExpressions

Além disso, as linhas que começam com // são comentários. Os comentários após os caracteres // podem aparecer em qualquer lugar; algumas das outras instruções possuem os próprios campos de comentários internos.

A seção Sets permite inserir nós individuais em um grupo. Esse recurso pode ser prático para separar os usuários com base em suas posições geográficas, por exemplo. Em seguida, você pode utilizar esses grupos ou qualquer combinação de endereços IP para especificar seus Filtros, que também são agrupados por nome. A partir de então, é possível utilizar todos esses agrupamentos e combiná-los em FilterExpressions. Se isso lhe parece confuso, é mesmo! Os filtros podem ser muito confusos, principalmente ao incluir uma sintaxe complexa e uma lógica não tão lógica (&&, ||, etc.). A sintaxe básica para definir Sets, Filters e FilterExpressions é parecida com a seguinte:

```
nome "comentários ou descrição" { conteúdo }
```

Toda definição contém um nome, seguido por comentários exibidos entre aspas e, em seguida, o comando entre chaves. Nossa filtro default,\* denominado filters, está localizado em \$OV\_CONF/C e parece com o seguinte:

```
// as linhas que começam com // são consideradas COMENTÁRIOS e ignoradas!
// Início dos Filtros MyCompanyName
```

```
Sets {

    dialupusers "DialUp Users" { "dialup100", "dialup101", \
                  "dialup102" }

}

Filters {

    ALLIPRouters "All IP Routers" { isRouter }

    SinatraUsers "All Users in the Sinatra Plant" { \
        ("IP Address" ~ 199.127.4.50-254) || \
        ("IP Address" ~ 199.127.5.50-254) || \
        ("IP Address" ~ 199.127.6.50-254) }

    MarkelUsers "All Users in the Markel Plant" { \
        ("IP Address" ~ 172.247.63.17-42) }

    DialAccess "All DialAccess Users" { "IP Hostname" in dialupusers }
```

\* Seu filtro pode ser muito diferente. O que apresentamos aqui foi reduzido para aliviar a complexidade de escrever um filtro.

```
}

FilterExpressions
{
    ALLUSERS "All Users" { SinatraUsers || MarkelUsers || DialAccess }

    NOUSERS "No Users" { !ALLUSERS }
}
```

Agora, vamos dividir esse arquivo em partes para saber o que ele faz.

### Sets

Primeiramente, definimos um Set\* denominado dialupusers, que contém os nomes de host (do DNS) que nossos usuários de discagem receberão ao discarem para nossa instalação. Esses são exemplos perfeitos de atividades que não desejamos gerenciar nem monitorar em nosso ambiente do OpenView.

### Filters

A seção Filters é a única não opcional. Definimos quatro filtros: ALLIPRouters, SinatraUsers, MarkelUsers e DialAccess. O primeiro filtro instrui a detecção de nós que possuem o valor de campo `isRouter`. O OpenView pode definir o atributo objeto de um dispositivo gerenciado com valores como `isRouter`, `isHub`, `isNode` etc.\*\* Esses atributos podem ser utilizados em expressões de Filtro para facilitar ainda mais a filtragem por grupos de objetos gerenciados, em vez de por faixas de endereços IP, por exemplo.

Os dois filtros seguintes especificam faixas de endereços IP. O filtro `SinatraUsers` é o mais complexo dos dois. Nesse filtro, especificamos três faixas de endereços IP, cada qual separada por símbolos OR lógicos (`||`). A primeira faixa (`"IP Address" - 199.127.6.50-254`) informa que, se o endereço IP estiver na faixa 199.127.6.50-199.127.6.254, deverá ser filtrado e ignorado. Se não estiver nessa faixa, o filtro examinará a faixa seguinte para se o endereço consta na faixa. Se não constar, o filtro examinará a última faixa de IP. Se o endereço IP não estiver em nenhuma das três faixas, o filtro permitirá a sua exibição e subsequente gerenciamento pelo NNM. A maioria dos programadores deve conhecer os outros operadores lógicos: `&&` representa um AND (E) lógico e `!` representa um NOT (NÃO) lógico.

O último filtro, `DialAccess`, permite excluir todos os sistemas que possuem um nome de host listado no `dialupusers`, definido no início do arquivo.

### FilterExpressions

A seção seguinte, `FilterExpressions`, permite combinar os filtros definidos anteriormente com a lógica adicional. Você pode utilizar uma `FilterExpression` em

\* Esses Sets não estão relacionados com a operação do `snmpset`, que conhecemos.

\*\* Procure na área `$OV_FIELDS` uma lista de campos.

qualquer local em que você usaria um Filtro. Imagine o seguinte: você cria expressões complexas usando Filters que, por sua vez, podem utilizar Sets nas partes contents das respectivas expressões. Em seguida, pode utilizar FilterExpressions para criar expressões mais simples, ainda que mais robustas. Em nosso caso, colocamos todos os filtros citados anteriormente em uma FilterExpression denominada ALLUSERS. Como nosso mapa do NNM deverá conter dispositivos não-usuários, definiremos um grupo denominado NOUSERS e instruiremos que ignore todos os dispositivos do tipo usuário como comando !ALLUSERS.

Como você pode ver, FilterExpressions também podem ajudar a tornar os elementos mais legíveis. Quando você terminar de configurar seu arquivo de filtro, use o programa \$OV\_BIN/ovfiltercheck para verificar a sintaxe de seu novo filtro. Se ocorrerem problemas, o programa informará para que você os corrija.

Após a definição de nossos filtros, podemos aplicá-los usando o comando *ovtopofix* ou o menu de configuração de polling mostrado na Figura 6-3.

Para remover nós de um mapa, use \$OV\_BIN/ovtopofix -f FILTER\_NAME. Vamos supor que alguém tenha criado um novo âmbito do DHCP sem informar a você e, repentinamente, todos os novos usuários agora estão no mapa. Você pode editar o arquivo de filtros, criar um novo grupo com a faixa de endereços IP do novo âmbito do DHCP, adicioná-lo à FilterExpression ALLUSERS, executar o *ovfiltercheck* e, se não ocorrerem erros, execute o \$OV\_BIN/ovtopofix -f NOUSERS para atualizar o mapa simultaneamente. Em seguida, interrompa e reinicie o *netmon* – caso contrário, ele continuará detectando esses nós indesejados, usando o filtro antigo. Eu mesmo executo o *ovtopofix* todos os meses para excluir alguns nós aleatórios.

## Carregando MIBs no OpenView

Antes de continuar explorando o NNM do OpenView, carregue algumas MIBs específicas do fornecedor.\* Isso o ajudará posteriormente, quando você começar a interagir (fazendo polling, grafando etc.), mas com dispositivos compatíveis com o SNMP. Vá para “Options → Load/Unload MIBs: SNMP”. Será exibida uma janela em que é possível adicionar MIBs específicas de fornecedores ao banco de dados. Como alternativa, você pode executar o comando \$OV\_BIN/ xnmloadmib e pode ignorar a necessidade de percorrer o NNM imediatamente.

Terminamos nossa *tour* resumida da configuração do OpenView. É impossível apresentar uma introdução completa à configuração do OpenView neste capítulo, de modo que tentamos fornecer um levantamento dos aspectos mais importantes de sua execução. Nada substitui a documentação e as páginas do manual que acompanham o produto.

\* Algumas plataformas e ambientes consideram o carregamento de uma MIB como uma compilação dessa MIB.

## *SNMPc Enterprise Edition da Castle Rock*

Terminaremos o capítulo com uma descrição sucinta do SNMPc, Versão 5.0, da Castle Rock, executado no Windows NT/2000. O SNMPc é um produto mais simples que o OpenView em diversos aspectos. Entretanto, embora mais simples, não quer dizer que não possui recursos. É também mais barato que o OpenView, o que o torna ideal para os estabelecimentos que não dispõem de muito dinheiro para investir em uma plataforma NMS, mas precisam do apoio e respaldo oferecidos por um produto comercial.

É fácil instalar o SNMPc. O instalador solicita o número da licença e um dispositivo-semente da detecção. O dispositivo-semente é semelhante ao arquivo-semente do *netmon* do OpenView. No caso do SNMPc, recomendamos fornecer o endereço IP (ou nome do host) do gateway do sistema, porque esse dispositivo pode ser usado para detectar outros segmentos da rede. A omissão do dispositivo-semente da detecção não impedirá que o SNMPc faça a descoberta, mas o limitará aos dispositivos da rede aos quais esteja diretamente conectado.

### *Mapa do SNMPc*

Quando o SNMPc estiver em funcionamento e em execução, você verá os dispositivos por ele detectados na visão do mapa Root. A Figura 6-8 mostra a barra de botões principais. O botão da extremidade direita (a casa) acessa o nível mais alto no mapa. As ferramentas de zoom permitem aproximar e afastar o mapa, aumentando ou diminuindo o nível de detalhamento exibido. Também é possível acessar o submapa Root, selecionando “Map View → Root submap” no menu View.



Figura 6-8 Barra de botões principais do SNMPc

### *Descoberta e filtros*

Quando você acabar de examinar seus mapas, deverá começar a ajustar os parâmetros da polling. Vá para “Config → Discovery Agents”. Será exibido um menu parecido com o da Figura 6-9. Ao examinar as guias do menu, percebe-se logo que você poderá configurar suas Seeds (sementes), Communities (comunidades) e Filters (filtros) aqui. Os filtros do SNMPc equivalem aos do OpenView, mas são bem mais simples.

A guia General permite controlar o comportamento da polling e da descoberta do SNMPc. A caixa de seleção para ativar e desativar a descoberta é auto-explicativa. A caixa de seleção “Enable Status Polling” determina se o SNMPc verificará (com o *ping*) periodicamente os nós na rede para detectar se estão respondendo ou não. Por default, todos os nós são verificados a cada 10

a 30 segundos. Para modificar esses valores predefinidos, você pode editar as propriedades de cada dispositivo (um a um), selecionar e destacar vários dispositivos (usando a tecla Ctrl) ou utilizar a ferramenta de seleção de objetos. É possível carregar essa ferramenta, usando o terceiro botão a partir da esquerda na barra de botões principais, ou acessando “View → Selection Tool”. A caixa de seleção “Discover Ping Nodes” permite especificar se você deseja revelar os dispositivos que possuem uma entidade de IP ou IPX mas não têm um agente de SNMP. “Discover IPX Nodes” permite revelar os dispositivos de IPX. O SNMPC também verificará se um dispositivo aceita diversos protocolos, como o SMTP, HTTP etc.

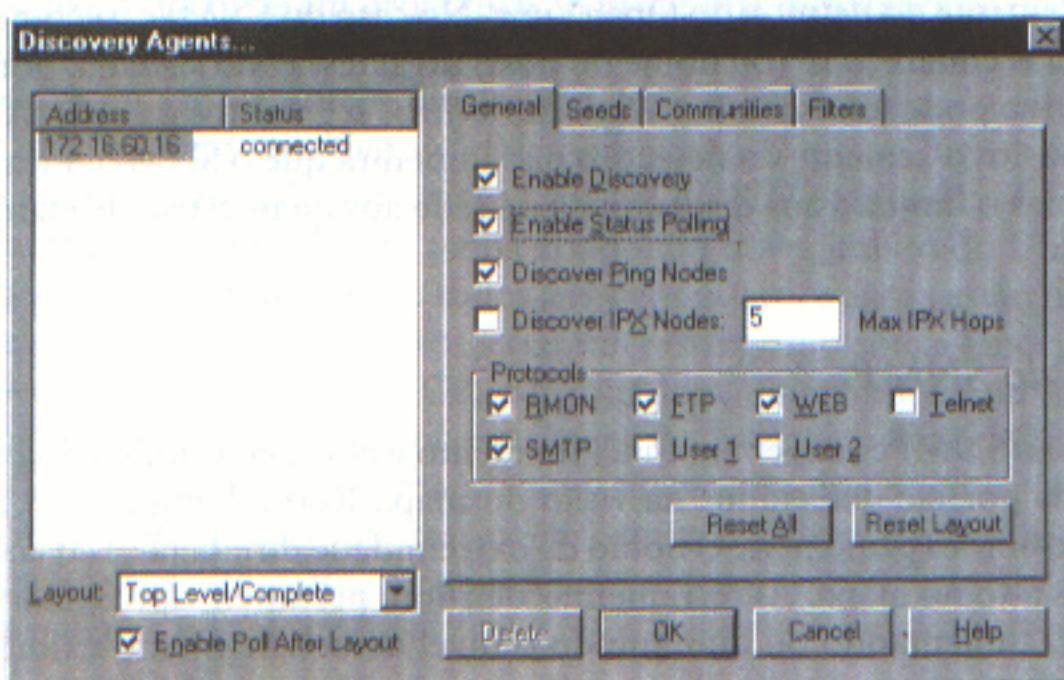


Figura 6-9 Menu Discovery Agents do SNMPC

Esse recurso permite configurar itens de menu personalizados com base nos serviços que o dispositivo está executando. A seção **Protocols** da guia **General** permite especificar os protocolos para os quais o SNMPC testará.

A guia **Seeds** permite especificar os dispositivos do SNMP que acompanharão o processo de descoberta. Essa guia também permite determinar mais de um endereço IP semente. (Lembre-se de que você foi solicitado a fornecer um dispositivo de endereço semente ao instalar o produto.)

A guia **Communities** permite especificar strings de comunidade para sua rede. Você pode indicar vários nomes de comunidade; o SNMPC experimentará os diversos nomes de comunidade ao revelar os nós de seu sistema. Assim que detectar a comunidade correta para um dispositivo específico, o SNMPC inserirá a string de comunidade no atributo “Get Community” do dispositivo específico. Isso significa apenas que o dispositivo recém-descoberto será gravado com a respectiva string de comunidade.

A última guia, **Filters**, permite excluir alguns endereços IP da descoberta. É possível especificar endereços individuais ou utilizar um asterisco (\*) como um curinga para indicar redes completas.

## *Carregando MIBs no SNMPc*

Assim como qualquer produto de gerenciamento de rede razoavelmente abrangente, o SNMPc pode carregar e compilar novas MIBs. Para isso, selecione “Config → MIB Database” na barra de menus principais. Essa janela permite especificar o caminho para o arquivo da MIB e fornece informações completas sobre o status da compilação etc. Clique no botão “Help” para obter mais informações sobre a compilação de MIBs.

O SNMPc é uma NMS compacta, que oferece alguns recursos adicionais, como relatório de tendências. Uma discussão mais detalhada sobre a instalação está além do escopo deste livro. O sistema de Ajuda on-line, que acompanha o SNMPc, é muito eficiente e recomendamos que você o aproveite ao máximo.

# Configurando agentes do SNMP

Nesse momento, você já deve saber o que é um agente do SNMP: nada mais do que um software residente no dispositivo a ser monitorado, e que responde às solicitações da NMS e gera traps. Este capítulo discute a configuração de agentes. Inicialmente, são definidos alguns parâmetros de configuração padrão, comuns a todos os agentes do SNMP, e, em seguida, o capítulo aborda alguns parâmetros avançados com os quais você pode se deparar ao configurar seus equipamentos. A maior parte deste capítulo discorre sobre a configuração de alguns dispositivos comuns, levando em consideração questões de segurança.

## Configurações de parâmetros

Todos os dispositivos do SNMP compartilham os seguintes parâmetros comuns configuráveis:

- *sysLocation*
- *sysContact*
- *sysName*
- Strings de comunidades de acesso leitura-gravação e somente leitura (e frequentemente, uma string de comunidade de trap)
- Destino da trap

O parâmetro *sysLocation* é a localização física do dispositivo sendo monitorado e sua definição na RFC 1213 é:

```
sysLocation OBJECT-TYPE
  SYNTAX  DisplayString (SIZE (0..255))
  ACCESS  read-write
  STATUS  mandatory
  DESCRIPTION
    "A localização física deste nó (por exemplo, 'cabine telefônica,
     3º andar')."
  ::= { system 6 }
```

Como é possível constatar, a SYNTAX é *DisplayString*, o que significa que pode ser uma string de caracteres ASCII; o tamanho é declarado com, no máximo, 255 caracteres. Esse objeto específico é útil para determinar a localização de um dispositivo. Esse tipo de informações práticas é fundamental em uma rede grande, principalmente no caso de uma rede de área estendida. Se um comutador tornar-se mal comportado, será possível encontrar a localização física desse comutador. Infelizmente, *sysLocation* geralmente não é definido quando o dispositivo é instalado e ainda mais freqüentemente, não é modificado quando o dispositivo é movido. Informações incertas são piores do que nenhuma informação; portanto, seja disciplinado e mantenha seus dispositivos atualizados.

A definição de *sysContact* da RFC 1213 é semelhante à de *sysLocation*:

```
sysContact OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "A identificação em texto da pessoa de contato para este nó gerenciado,
         juntamente com as informações sobre como contatar essa pessoa."
    ::= { system 4 }
```

*sysContact* é uma *DisplayString* e está evidente a finalidade desse uso: identificar o contato principal para o dispositivo em questão. É importante definir esse objeto com um valor adequado porque pode ser útil para a equipe de operações determinar quem deverá ser contatado se ocorrer uma falha catastrófica. Você também pode utilizá-lo para assegurar que você será avisado, se for responsável por um dispositivo específico, quando alguém precisar paralisar seu dispositivo a título de manutenção ou reparo. Assim como em *sysLocation*, mantenha essa informação atualizada à medida que a equipe mudar. Não é raro encontrar dispositivos cujo *sysContact* é alguém que saiu da empresa há muitos anos.

*sysName* deve ser definido com um FDQN (*fully-qualified domain name* – nome de domínio totalmente qualificado) para o dispositivo gerenciado. Em outras palavras, trata-se do nome do host associado ao endereço IP do dispositivo gerenciado. A definição na RFC 1213 é a seguinte:

```
sysName OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
        "Um nome atribuído em termos administrativos a este nó gerenciado.
         Por convenção, trata-se do nome de domínio totalmente qualificado
         do nó."
    ::= { system 5 }
```

**Sup** Os parâmetros `read-only` e `read-write` são as strings de comunidade para o acesso `read-only` e `read-write`. Observe que `sysLocation`, `sysContact` e `sysName` têm valores de `ACCESS` definidos como `read-write`. Com a string de comunidade `read-write` adequada, é possível modificar a definição desses objetos e de muitos outros mais importantes. Em última análise, não será um grande problema se alguém fizer seu roteador mentir sobre sua localização – provavelmente, você já sabe que ele não está localizado na Antártica.

Entretanto, quem pode fazer isso também consegue burlar suas tabelas de direcionamento e praticar outras ações muito mais prejudiciais. Certamente, quem tiver somente a string de comunidade `read-only` poderá descobrir outras informações sobre sua rede do que você revelaria a uma pessoa estranha. A definição das strings de comunidade é muito importante para manter um ambiente protegido. A maioria dos dispositivos é fornecida com strings de comunidade default, muito conhecidas. Não pense que você pode deixar a definição das strings de comunidade para depois.

Os parâmetros de destino da trap especificam os endereços para os quais as traps são enviadas. Não há nenhuma mágica aqui – como as traps são notificações assíncronas geradas por seus dispositivos, o agente precisa saber quem deve receber a notificação. Alguns dispositivos têm suporte para traps de falha de autenticação, geradas se alguém tentar acessá-los com strings de comunidade incorretas. Esse recurso é muito útil, porque permite detectar as tentativas de violação de seus dispositivos. Alguns dispositivos também aceitam a possibilidade de incluir uma string de comunidade nas traps; é possível configurar a estação de gerenciamento de rede para responder somente às traps que contêm a string de comunidade correta.

Vários dispositivos têm desvios adicionais para os parâmetros de acesso e de traps. Por exemplo, os dispositivos Cisco permitem criar strings de comunidade diferentes para partes distintas da MIB – é possível utilizar essa opção para permitir a definição de algumas variáveis, mas não outras. Alguns fornecedores permitem impor restrições nos hosts que podem fazer solicitações de SNMP. Ou seja, o dispositivo responderá somente às solicitações de determinados endereços IP, independentemente da string de comunidade.

A faixa de opções de configuração que provavelmente você encontrará estará limitada apenas pela criatividade dos fornecedores, de modo que é obviamente impossível descrever tudo o que você poderia encontrar. A seção “Considerações sobre a configuração de agentes”, apresentada mais adiante neste capítulo, dá uma idéia de como alguns agentes implementam os parâmetros de configuração padrão e faz uma análise resumida dos outros recursos possivelmente disponíveis.

## Questões de segurança

O Capítulo 2 discute as questões de segurança relacionadas ao SNMPv1 e SNMPv2. Evidentemente, o maior problema é que as strings de comunidade `read-only` e `read-write` são enviadas como strings de texto explícitas; o agente ou a

NMS não aplica qualquer codificação. Por conseguinte, as strings de comunidade estão disponíveis para quem tiver acesso a um *sniffer* de pacotes. Certamente, isso significa quase todos na rede com um PC e a possibilidade de fazer download de softwares amplamente disponíveis. Isso o incomoda? Deveria.

Obviamente, é necessário ter com as strings de comunidade as mesmas precauções que você teria com as senhas do superuser ou do administrador. Escolha strings de comunidade difíceis de adivinhar. As strings alfanuméricas com letras maiúsculas/minúsculas são escolhas eficientes de strings de comunidade; não use palavras de dicionários.

Embora alguém com a string de comunidade read-only não possa fazer tanto estrago quanto alguém com a string read-write, você deve tomar as mesmas precauções nos dois casos. Não se esqueça de modificar suas strings de comunidade – as maioria dos dispositivos são fornecidos com strings de comunidade pré-configuradas e extremamente fáceis de se adivinhar.

Isso não soluciona os problemas com os *sniffers* de pacotes. Quando estiver configurando um agente, convém limitar os dispositivos que podem fazer solicitações de SNMP (presumindo que o agente de seu sistema permite fazer essa restrição). Dessa forma, mesmo que alguém obtenha as strings de comunidade, precisará burlar o endereço IP de uma de suas estações de gerenciamento para fazer qualquer estrago.

Naturalmente, algumas pessoas sabem como trapacear endereços IP e não convém pressupor que você possa confiar em seus funcionários. Uma solução melhor para o problema seria impedir que os pacotes de SNMP fiquem visíveis nas conexões de rede externas e nas partes da rede em que você não deseja que eles apareçam. Isso requer a configuração de seus roteadores e firewalls com listas de acesso que bloqueiem os pacotes de SNMP do mundo exterior (o que pode incluir partes de sua rede). Se você não confia nos usuários da rede, convém configurar uma rede administrativa separada para ser utilizada para consultas de SNMP e em outras operações de gerenciamento. Esse procedimento é dispendioso e radical – é difícil imaginar a extensão de uma rede além dos principais roteadores e servidores – mas pode ser o que sua situação assim o exija.

Para utilizar o SNMP para monitorar sua rede em casa, seja muito cuidadoso. Você não quer que suas strings de comunidade viajem pela Internet pública em uma forma não codificada. Para usar as ferramentas do SNMP diretamente de casa, instale o software VPN ou alguma forma de canalização, para manter privado o tráfego de seu SNMP. Uma abordagem mais adequada para a monitoração em casa é utilizar uma interface de web; ao usar o SSL, é possível impedir que os outros visualizem seus gráficos de utilização. (Nenhum produto de gerenciamento de rede que conhecemos aceita o SSL fora da caixa; mas permitem a integração com servidores externos, como o Apache, que têm suporte para SSL).

O SNMPv3 (discutido no Apêndice F) corrige a maioria dos problemas de segurança; em particular, assegura que as strings de comunidade estejam sempre codificadas. Infelizmente, no momento, existem raras implementações do SNMPv3. Está evidente a direção que você deseja seguir, mas você ainda não pode chegar a algum lugar.

## *Considerações sobre a configuração de agentes*

Nas seções a seguir, examinaremos a configuração de alguns agentes de SNMP conhecidos. Escolhemos os dispositivos encontrados em quase todas as redes modernas (PCs x86, Servidores do Unix, roteadores, no-breaks etc.).

O objetivo dessa discussão não é demonstrar como seu agente específico está configurado – isso não seria prático, devido às centenas de dispositivos e fornecedores existentes. Nossa finalidade é dar uma idéia de quais são as opções comuns e as etapas gerais de configuração de um agente.

### *Agente do Windows 95/98*

Nesta seção, examinaremos a configuração do SNMP para o agente do Windows 95/98, usando o System Policy Editor (Editor de diretivas do sistema) do Windows. Todas as configurações estão armazenadas no registro, de modo que você também pode fazer alterações na configuração usando o *regedit*, mas há menos probabilidade de erro se você usar o Editor de diretivas do sistema. É importante observar que o Windows 95, 98 e NT têm as mesmas entradas de SNMP no registro; portanto, a configuração desses sistemas é semelhante. Também compensa observar que o agente de SNMP da Microsoft não é muito robusto, embora seja adequado para se obter apenas a funcionalidade básica do SNMP. Existem outros agentes disponíveis; o SystemEDGE da Concord e o SNMPC da Castle Rock têm suporte para os sistemas operacionais da Microsoft.



A menos que você domine totalmente a edição do Registro, recomendamos enfaticamente que você use o Editor de diretivas do sistema para efetuar alterações na configuração dos agentes. Configurações incorretas no registro podem resultar problemas sérios no sistema. Considerese avisado.

O Editor de diretivas do sistema Windows é fornecido com o Resource Kit do Windows 95/98 e deve ser instalado para permitir a configuração do agente de SNMP. Na primeira execução, o Editor de diretivas do sistema solicitará um arquivo *.adm*. Selecione C:\WINDOWS\INF\ADMIN.ADM para esse arquivo. Selecione “File Open Registry” e clique duas vezes no ícone Local Computer. Na guia Policies, clique nos sinais de mais até você alcançar Network e, em seguida, SNMP. Esse procedimento deve abrir quatro itens de configuração de agente de SNMP. A Figura 7-1 traz a janela exibida. Para ativar uma opção, coloque uma marca ao lado. Quando terminar, clique em “OK” e, em seguida, em “File Save” na tela principal. Se você não seguir essas etapas, a configuração não será gravada no registro.

As configurações de “Communities” permitem definir as strings de comunidade. Marque a caixa e clique em “Show” na seção inferior. Esse procedimen-

to abre outra janela apresentando suas strings de comunidade. Para criar uma nova comunidade, clique em “Add” e insira a string. Repita as etapas, se necessário, para seu site. Se essa opção ficar desmarcada ou se estiver marcada mas sem nomes de comunidade listados, o agente responderá a todas as solicitações de SNMP recebidas. O próximo item de caixa de seleção, “Permitted managers”, especifica as NMSs que podem acessar esse agente. Você pode identificar suas estações de gerenciamento por endereços de IPX, endereços IP ou nomes DNS. Por exemplo, você pode utilizar esse item para restringir o acesso do SNMP a uma NMS específica.

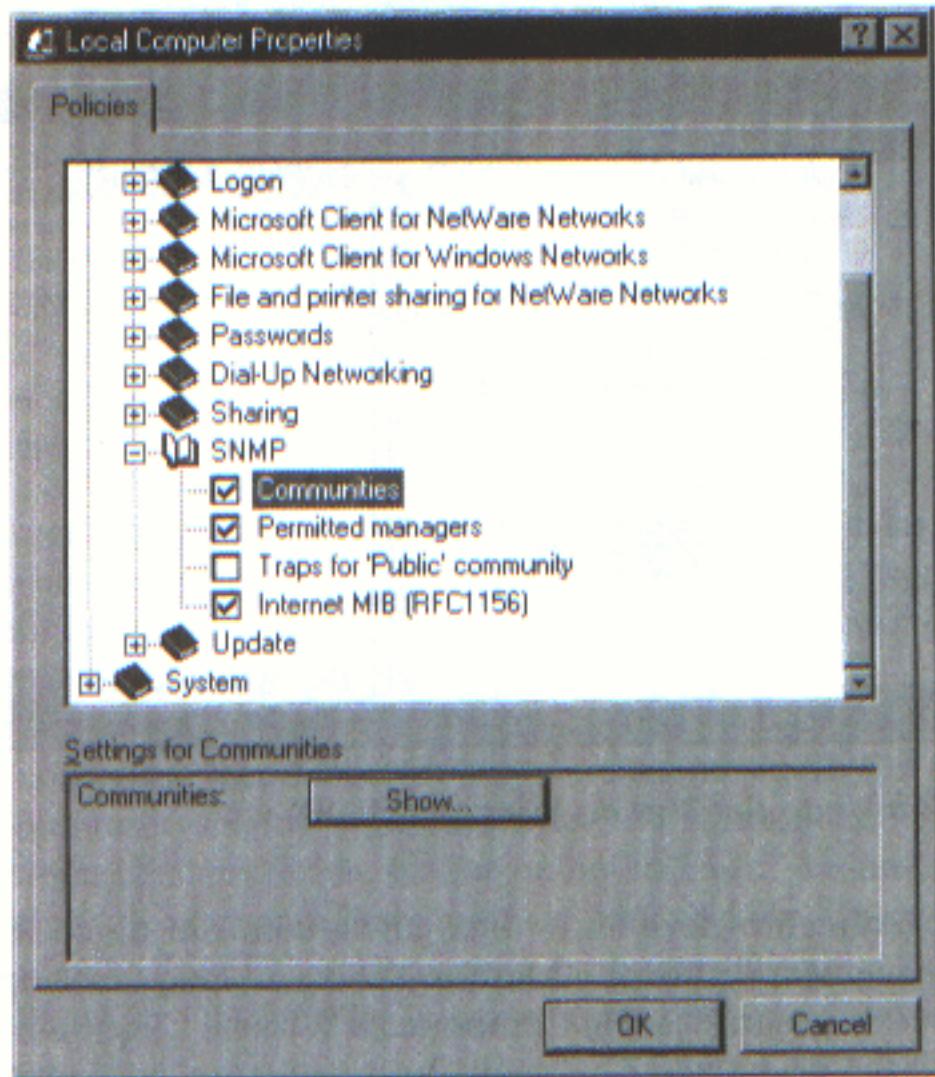


Figura 7-1 Editor de diretivas do sistema Windows 95/98

Se a caixa “Permitted managers” estiver desmarcada ou marcada, mas sem entradas, o agente responderá a todas as solicitações, independentemente da origem. Marcar a opção “Traps for ‘Public’ community” permite indicar até cinco NMSs para receber traps. A última configuração, “Internet MIB (RFC1156)”, permite definir os objetos Contact Name (*sysContact*) e Location (*sysLocation*).

Salve as alterações com “File Save” no menu principal do Editor de diretivas do sistema. A Figura 7-2 mostra as entradas no Editor do Registro, depois que você utilizar o Editor de diretivas para defini-las.

## Agente do Windows NT 4.0 e Windows 2000

Para configurar o serviço do SNMP no Windows NT 4.0 e 2000, comece no Control Panel e clique duas vezes no ícone Network. Clique na guia Services, selecione “SNMP Service” e pressione o botão “Properties”. Se a opção “SNMP Service” não estiver listada, é necessário adicioná-la. Pressione o botão “Add” e selecione “SNMP Service” na lista de serviços. Essa lista solicitará o disco do sistema Windows NT; portanto, deixe-o à mão. Para o Windows 2000, entre no Control Panel e clique em “Add/Remove Programs”. Quando a janela se abrir, clique em “Add/Remove Windows Components” e selecione “Management and Monitoring Tools”.

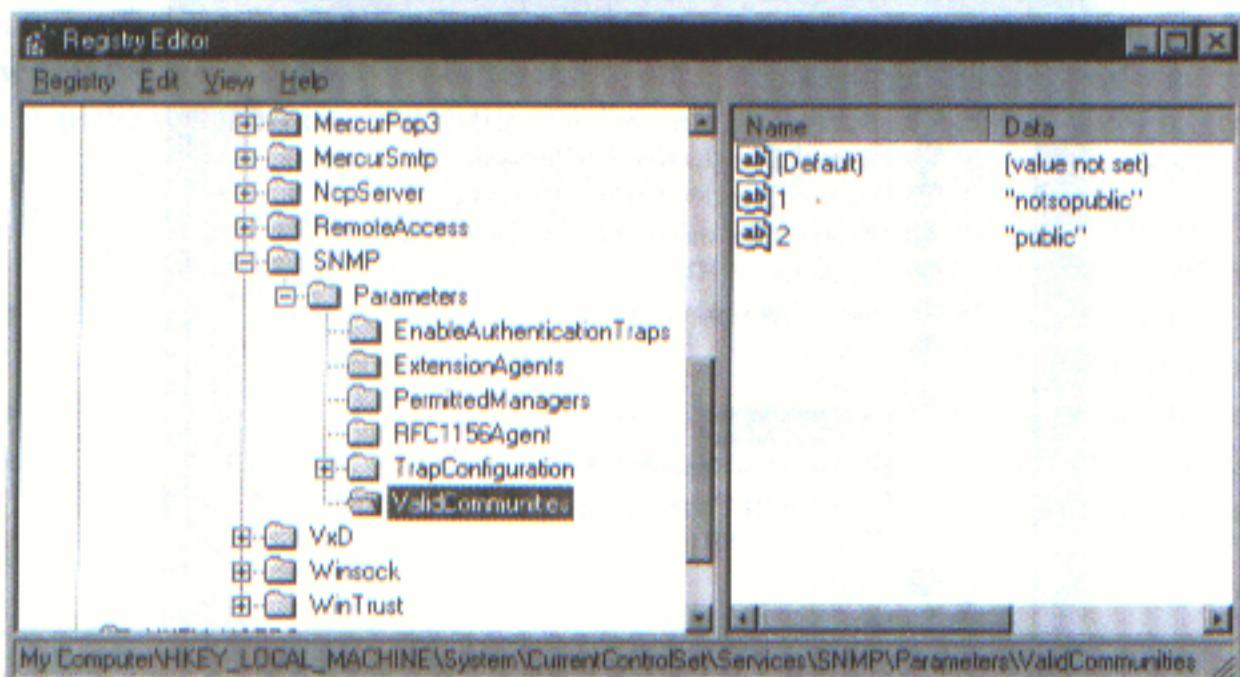


Figura 7-2 Editor do Registro do Windows 95/98

Esse procedimento deve abrir uma janela com um único item, “Simple Network Management Protocol”. Marque a caixa ao lado dessa opção e clique em “OK”. Você retornará à janela Components Wizard. Clique em “Next” para iniciar a instalação do serviço de SNMP. Provavelmente, você precisará do CD do Windows 2000.

Após instalar o serviço de SNMP ou selecioná-lo na lista de serviços instalados, será exibida uma nova janela, dividida em três guias: Agent, Traps e Security. Na guia Agent, você deve configurar os objetos Contact (*sysContact*), Location (*sysLocation*) e Service (*sysServices*). Ainda não tínhamos mencionado o objeto *sysServices*; a RFC 1213 define esse objeto conforme mostrado a seguir:

```
sysServices OBJECT-TYPE
  SYNTAX  INTEGER (0..127)
  ACCESS  read-only
  STATUS   mandatory
  DESCRIPTION
```

"Um valor que indica o conjunto de serviços oferecidos por

essa entidade. O valor é um somatório que usa inicialmente o valor zero. A partir de então, para cada camada, L, no intervalo de 1 a 7, para a qual este nó executa transações,  $2^L$  elevado a  $(L - 1)$  é adicionado à soma. Por exemplo, um nó que executa basicamente funções de direcionamento teria um valor de 4 ( $2^{(3-1)}$ ). Ao contrário, um nó que é um host oferecendo serviços de aplicativo teria um valor de 72 ( $2^{(4-1)} + 2^{(7-1)}$ ). Observe que no contexto do conjunto de protocolos da Internet, os valores devem ser calculados adequadamente:

#### funcionalidade da camada

- 1 física (por exemplo, repetidoras)
- 2 link de dados/sub-rede (por exemplo, bridges)
- 3 internet (por exemplo, gateways de IP)
- 4 ponto a ponto (por exemplo, hosts de IP)
- 7 aplicativos (por exemplo, transmissores de correio)

Para os sistemas incluindo protocolos OSI, as camadas 5 e 6 também podem ser consideradas."

A guia Agent fornece uma caixa de seleção para cada uma das sete camadas ISO que o objeto *sysServices* representa. O texto da *DESCRIPTION* na RFC apresenta uma definição sucinta de cada camada. Se necessário, marque cada serviço oferecido por sua máquina do NT.

Quando terminar de configurar a guia Agent, selecione a guia Traps; essa guia permite configurar a comunidade em que o agente de SNMP enviará traps. Na caixa "Community Name", digite o nome da comunidade (que distingue maiúsculas/minúsculas) preferida. Clique no botão "Add" à esquerda e adicione até cinco destinos de trap para esse nome de comunidade. Os destinos de trap podem ser endereços de IPX, endereços IP ou nomes DNS.

Agora, clique na guia Security. A parte inicial dessa guia oferece uma opção para enviar traps de erro de autenticação. É conveniente marcar essa caixa porque pode ajudá-lo a detectar invasores. A caixa "Accepted Community Names" lista todos os nomes de comunidade aos quais o agente responderá. Clique em "Add" o nome da comunidade preferido. É importante configurar essas comunidades porque uma pessoa com a string de comunidade correta pode invadir seu sistema. Se você deixar essa caixa em branco, o agente responderá a todas as solicitações. A metade inferior do menu Security permite especificar se o agente aceitará pacotes de SNMP de qualquer host ou somente de uma lista especificada. Para criar uma lista, o que recomendamos enfaticamente, clique em "Only Accept SNMP Packets from These Hosts" e use o botão "Add" para adicionar os nomes de host ou endereços de suas estações de monitoração. As opções dos hosts são idênticas às aquelas dos destinos de trap; são aceitos endereços de IPX, endereços IP e nomes DNS.

Por último, clique em “OK” para salvar as alterações e atualizar o Registro do Windows. Se, em algum momento, você cometer um erro, clique em “Cancel”, para anular o processo de configuração; nenhuma alteração será efetuada no registro.

## Agente do HP OpenView para o HP-UX e Solaris

Um único arquivo de configuração de texto controla os parâmetros desse agente; geralmente, o arquivo é denominado `/etc/SnmpAgent.d/snmpd.conf` ou `/etc/snmpd.conf` nos sistemas mais antigos. Não é necessário editar esse arquivo para o agente funcionar normalmente. Se você o editar, deverá interromper e reiniciar o agente principal executando o script `SnmpMaster`, primeiro com um `stop` e, em seguida, com um `start`:

```
$ /sbin/init.d/SnmpMaster stop  
$ /sbin/init.d/SnmpMaster start
```

### Configuração simples

O arquivo de configuração a seguir configura o agente para responder às solicitações de `get` usando o nome de comunidade `public` e às solicitações de `set` usando o nome de comunidade `private`. Não há restrições impostas sobre as MIBs a serem consultadas nem sobre os hosts que podem fazer as consultas. Essa configuração não possui segurança, porque as strings de comunidade estão definidas com defaults habitualmente utilizados e são amplamente conhecidas. O agente do OpenView envia, por default, traps de falha de autenticação, de modo que não é necessário habilitar essas traps no arquivo de configuração.

```
get-community-name: public  
set-community-name: private  
trap-dest: 127.0.0.1  
contact: B.Gates  
location: 12 Pyramid – Egypt
```

A configuração mais simples é editar o arquivo e inserir nomes de comunidade mais razoáveis nas duas primeiras linhas. Na realidade, não podemos fazer essa afirmação: os nomes de comunidade são basicamente senhas. Para selecionar nomes de comunidades, use as mesmas regras aplicáveis à seleção da senha da raiz. Defina sempre o host da trap de destino (`trap-dest`) como endereço IP do host que receberá a trap.

O exemplo a seguir configura vários nomes de comunidade diferentes:

```
get-community-name: public  
get-community-name: media  
set-community-name: hushed  
set-community-name: veryprivate  
set-community-name: shhhh
```

Criamos duas comunidades de *get* (read-only) e três comunidades de *set* (read-write). Essas comunidades podem ser usadas conforme a necessidade. (Na vida real, teríamos escolhido nomes mais obscuros.) Por exemplo, você poderia conceder ao grupo operations de Nova York acesso de comunidade *public* e ao grupo operations de Atlanta acesso de comunidade *media*. As demais comunidades de *set* também podem ser subdivididas entre os vários administradores e outras equipes que precisam de acesso leitura-gravação.

## Configuração avançada

A configuração de diversas strings de comunidade não parece muito útil e, em si mesma, não o é. Mas você pode avançar nesse conceito e criar várias comunidades, cada qual consistindo em alguns hosts específicos e pode acessar somente alguns dos objetos gerenciados pelo SNMP. O exemplo a seguir permite que o host 10.123.56.25 emita *gets* usando o nome de comunidade *comname* e *sets* usando o nome de comunidade *private*. O host 10.123.46.101 pode emitir *gets* usando somente o nome de comunidade *comname*. Não é possível utilizar nomes de host após a diretiva IP;; use endereços IP.

```
get-community-name      comname IP: 10.123.56.25 10.123.46.101  
set-community-name     private IP: 10.123.56.25
```

Você também pode configurar o agente de modo a restringir o acesso às subárvore da MIB com base em endereços IP. O exemplo a seguir permite que qualquer host obtenha qualquer objeto existente em *iso.org.dod.internet.mgmt.mib-2*, exceto os objetos da subárvore *interfaces*. O sinal de menos (-) à frente das interfaces instrui o agente a impedir o acesso a essa subárvore.

```
get-community-name      public VIEW: mib-2 -interfaces
```

O último exemplo configura vários nomes de comunidade para os *sets* e *gets*. Um administrador localizado no host 10.123.46.25 e ciente da string de comunidade *admin* tem acesso de leitura à árvore da MIB inteira; com a string de comunidade *adminset*, ele tem acesso de gravação à árvore inteira. Quem tiver a string de comunidade *operator* poderá estar em qualquer lugar e acessar todo o conteúdo da *mib-2* exceto a subárvore *interfaces*, mas deverá estar na própria mesa (10.123.56.101) para emitir *sets* e não poderá fazer qualquer definição na subárvore *mib-2*.

```
get-community-name      operator VIEW: mib-2 -interfaces  
get-community-name      admin    IP: 10.123.56.25  
set-community-name     operset  IP: 10.123.46.101 VIEW: -mib-2  
set-community-name     adminset IP: 10.123.56.25
```

## Net-SNMP (o antigo UCD-SNMP)

O Net-SNMP é um agente de fonte aberta, totalmente disponível em <http://netsnmp.sourceforge.net>. Focalizaremos o Net-SNMP Version 4.2, que é o mais re-

cente a partir desta publicação. Depois que você fizer o download e desempacotar a distribuição, mude (com o comando *cd*) para o diretório em que você desempacotou o Net-SNMP e leia os arquivos *README* e *INSTALL*. Esses arquivos fornecem informações gerais sobre a instalação do agente e não exigem muita explicação aqui.

O Net-SNMP usa um script *configure* para assegurar que o ambiente possui alguns utilitários e bibliotecas importantes instalados para permitir uma compilação bem sucedida. Algumas opções de configuração podem ser definidas ao executar esse script. Para ver uma lista dessas opções, execute o seguinte comando:

```
ucd-snmp-4.2/> ./configure --help
```

Uma opção comum é *--prefix=PATH*, que especifica um diretório de instalação alternativo. Por default, o Net-SNMP instalará no */usr/local/bin*, */usr/local/man*, etc.

Executaremos o *configure* sem opções, o que significa que nossa construção do Net-SNMP terá valores predefinidos atribuídos a várias opções. Por exemplo, o binário do agente será colocado em */usr/local/sbin*. Execute o seguinte comando para iniciar o processo de configuração:

```
ucd-snmp-4.2/> ./configure
```

Serão exibidas diversas mensagens sobre os recursos que o *configure* está procurando e se são encontrados ou não.

Após executar por alguns instantes, o *configure* solicitará algumas informações básicas do SNMP:

```
***** Configuration Section *****
```

You are about to be prompted by a series of questions. Answer them carefully, as they determine how the snmp agent and related applications are to function.

After the configure script finishes, you can browse the newly created config.h file for further - less important - parameters to modify. Be careful if you re-run configure though since config.h will be over written.

-Press return to continue-

Ao pressionar Return, você será solicitado a fornecer informações sobre contato do sistema:

```
disabling above prompt for future runs... yes  
checking System Contact Information...
```

```
*** System Contact Information:
```

Describes who should be contacted about the host the agent is running on. This information is available in the MIB-II tree. This can also be over-ridden using the "syscontact" syntax in the agent's configuration files.

```
System Contact Information (root@): snmpadmin@ora.com
setting System Contact Information to... snmpadmin@ora.com
checking System Location...
```

Decidimos definir nossas informações de contato com algo útil, mas poderíamos ter deixado em branco. O item seguinte a ser configurado é a localização do sistema. Escolhemos um valor informativo, mas também poderia ficar em branco:

\*\*\* System Location:

Describes the location of the system. This information is available in the MIB-II tree. This can also be over-ridden using the "syslocation" syntax in the agent's configuration files.

```
System Location (Unknown): FTP Server #1, O'Reilly Data Center
setting System Location to... FTP Server #1, O'Reilly Data Center
checking Location to write logfile...
```

A última opção a ser configurada é localização do arquivo de log *snmpd*:

\*\*\* Logfile location:

Enter the default location for the snmpd agent to dump information & errors to. If not defined (enter the keyword "none" at the prompt below) the agent will use stdout and stderr instead. (Note: This value can be over-ridden using command line options.)

```
Location to write logfile (/var/log/snmpd.log):
setting Location to write logfile to... /var/log/snmpd.log
```

\*\*\* snmpd persistent storage location:

Enter a directory for the snmp library to store persistent data in the form of a configuration file.

```
Location to write persistent information (/var/ucd-snmp):
setting Location to write persistent information to... /var/ucd-snmp
updating cache ./config.cache
creating ./config.status
creating Makefile
creating MakefileMakefile
creating snmplib/Makefile
creating agent/Makefile
```

```
creating apps/Makefile
creating apps/snmpnetstat/Makefile
creating agent/mibgroup/Makefile
creating agent/dlmods/Makefile
creating local/Makefile
creating testing/Makefile
creating man/Makefile
creating ov/Makefile
creating mibs/Makefile
creating config.h
```

O valor default é `/var/log/snmpd.log`, que deve funcionar na maioria dos sistemas Unix.

Quando script `configure` termina, gera um arquivo específico do sistema, denominado `config.h`. Antes de continuar, dê uma olha nesse arquivo, que armazena algumas variáveis de configuração locais que talvez você prefira modificar antes de iniciar a compilação. Eis alguns fragmentos de meu arquivo `config.h`:

```
/* default list of mibs to load */

#define DEFAULT_MIBS "IP-MIB:IF-MIB:TCP-MIB:UDP-MIB:SNMPv2-MIB: \
RFC1213-MIB:UCD-SNMP-MIB:SNMPv2-PARTY-MIB:SNMPv2-M2M-MIB: \
SNMP-VIEW-BASED-ACM-MIB"

/* default location to look for mibs to load using the above tokens
   and/or those in the MIBS environment variable */

#define DEFAULT_MIBDIRS "/usr/local/share/snmp/mibs"

/* LOGFILE: If defined it closes stdout/err/in and opens this in
out/err's
place. (stdin is closed so that sh scripts won't wait for it) */

#define LOGFILE "/var/log/snmpd.log"

/* default system contact */
#define SYS_CONTACT "snmpadmin@ora.com"

/* system location */
#define SYS_LOC "FTP Server #1, O'Reilly Data Center"
```

Agora, você pode compilar o novo pacote com o comando `make`. O processo de compilação exibe algumas mensagens mas você pode ignorar a maioria delas. Em resumo, se for concluído, você obteve êxito e poderá passar para a instalação; caso contrário, você verá os erros e deve investigar o que deu errado.

Se você alterar um pouco o arquivo `config.h` e sua construção falhou, tente recriar o `config.h`. Sem modificar esse novo `config.h`, tente outra construção. Isso deve eliminar os problemas que você criou dentro desse arquivo.

Instale o novo pacote com o comando *make install*. Por default, esse comando instala diversos executáveis em */usr/local/bin* e outras informações importantes em */usr/local/share/snmp*.

Nessa etapa, você também pode configurar o agente por meio de uma das duas abordagens

- Executar o programa */usr/local/bin/snmpconf*, que faz diversas perguntas e gera um arquivo de configuração. Entretanto, o script de configuração é surpreendentemente confuso, de modo que é difícil recomendar essa proposta.
- Elaborar uma configuração manualmente. Se você não estiver interessado no SNMPv3, essa opção é bem fácil.

### *Executar o script de configuração*

O script de configuração é longo e complexo. Eis alguns indicadores:

- Ele inicia perguntando se deve criar o *snmp.conf* ou o *snmpd.conf*. Para configurar o agente, selecione *snmpd.conf*. O *snmp.conf* configura alguns defaults para as ferramentas de linha de comando, como o *snmpget*. Na realidade, não é necessário criar o *snmp.conf*.
- A maioria das opções configuráveis está relacionada ao SNMPv3. Embora a Versão 3 seja um avanço muito importante, certamente você pode ignorá-la; raros fornecedores têm suporte para a v3, discutida no Apêndice F.
- Quando você terminar de configurar, o script deixará o arquivo de configuração em seu diretório atual. Você pode colocar os arquivos em *~/.snmp*, se forem para seu uso pessoal, ou em */usr/local/share/snmp*, para que essa configuração seja usada por todos no sistema.

### *Criar uma configuração manualmente*

Se você não deseja executar uma tarefa complexa, é fácil criar uma arquivo de configuração próprio. Examine a seguir um arquivo de configuração muito simples:

```
syslocation      "O'Reilly Data Center"
syscontact       snmpadmin@oreilly.com
rwcommunity      private
rocommunity      public
authtrapenable  1
trapcommunity   trapsRus
trapsink         nmshost.oreilly.com
trap2sink        nmshost.oreilly.com
```

Os itens da configuração já devem ser conhecidos: estamos configurando a localização do sistema, o contato do sistema, as strings de comunidade read-write, read-only e trap, e os destinos das traps. Também estamos ativando as traps de autenticação. Observe que configuramos destinos para as traps do SNMP Version 1 e Version 2. As linhas de destino das traps (*trapsink* e *trap2sink*)

`sink`) também podem ter uma string de comunidade trap, se a NMS em determinado host exigir um nome de comunidade diferente.

As linhas `rwcommunity` e `rocommunity` nos permitem um pouco mais de sofisticação do que no exemplo. É possível especificar a rede ou sub-rede (subnet) à qual as strings de comunidade são aplicáveis e um ID de objeto que restringe consultas aos objetos da MIB subordinados a esse OID. Por exemplo, para limitar o acesso de leitura-gravação às estações de gerenciamento da sub-rede (subnet) 10.0.15.0/24, você poderia usar a linha:

```
rwcommunity private 10.0.15.0
```

Se você seguir esse percurso, certamente visualizara o arquivo `EXAMPLE.conf` no diretório em que você construiu o Net-SNMP. É possível modificar esse arquivo e instalá-lo na localização adequada (`~/.snmp/snmpd.conf` ou `/usr/local/share/snmp/snmpd.conf`) ou aproveitar as idéias nele contidas e usá-las em sua própria configuração. Esse arquivo contém alguns truques particularmente inteligentes que discutiremos no Capítulo 11, mas que estão muito além da configurar simples aqui abordada.

## *Agente SystemEDGE da Concord para o Unix e NT*

O SystemEDGE da Concord é um produto comercial que pode ser utilizado como um subagente para o agente padrão do Windows NT. Nos sistemas Unix, é possível utilizá-lo como um agente independente ou lado a lado com um agente já existente. É executado no Linux, Solaris e em outros sistemas operacionais. O CD em que o produto é fornecido contém agentes para todas as plataformas suportadas pelo SystemEDGE. Sempre que possível, o SystemEDGE usa o gerenciador do pacote interno da plataforma para facilitar a instalação. Cada versão do agente, dependente de arquitetura, é fornecida com um arquivo `README` fácil de seguir, para a instalação. Leia o Capítulo 11 para obter uma discussão sobre os recursos desse agente.

### *Configuração simples*

O arquivo de configuração do SystemEDGE está localizado em `/etc/sysedge.cf`. Use seu editor preferido para fazer alterações nesse arquivo. É necessário interromper e reiniciar o SystemEDGE para que as modificações entrem em vigor. O formato do arquivo de configuração é o mesmo para todas as versões do SystemEDGE.

Para uma configuração comum do SNMP, o `sysedge.cf` ficaria assim:

```
community public read-only
community veryprivate read-write 127.0.0.1 10.123.56.25
community traps 127.0.0.1
```

As linhas de comentários iniciam com um caractere `#`. O primeiro parâmetro define a comunidade `read-only` com `public`. A comunidade `read-write` é defi-

nida como `veryprivate`. Os dois endereços IP que aparecem depois da string de comunidade `read-write` são uma lista de acesso que instrui o agente a autorizar operações de `set` no `localhost` (`127.0.0.1`) e `10.123.56.25` somente. Use sempre uma lista de acesso, se possível; sem esse recurso de segurança, qualquer host pode executar operações de `set`. Observe que existe um espaço entre os dois endereços, não um caractere de tabulação. A terceira opção informa ao agente o local para onde enviar as traps; nesse caso, para o `localhost` (`127.0.0.1`).

Por default, o agente enviar as traps de falha de autenticação e recomendamos enfaticamente o uso dessas traps. Para não usar as traps de falha de autenticação, inclua a seguinte linha no arquivo de configuração:

```
no_authen_traps
```

## Configuração avançada

O SystemEDGE oferece alguns recursos poderosos de automonitoração. Essas extensões (encontradas somente na MIB de empresa privada do Empire da Concord) são semelhantes à MIB Remote Network Monitoring (RMON), discutida no Capítulo 9. As extensões do Empire podem reduzir a carga da rede ao permitir que o agente, e não uma NMS, faça a monitoração (polling) de objetos importantes do sistema. Por exemplo, o agente pode ser instruído a verificar se o espaço disponível no sistema de arquivos da raiz permanece acima de um limiar predefinido. Quando esse limiar for ultrapassado, o agente enviará uma trap à NMS para que a condição seja tratada adequadamente.

A linha a seguir mostra como é possível monitorar e reiniciar o `sendmail` se ele for derrubado:

```
watch process procAlive 'sendmail' 1 0x100 60 'Watch Sendmail'  
'/etc/init.d/sendmail start'
```

Esse monitor envia uma trap para a NMS, definida anteriormente como `community traps 127.0.0.1`, quando o processo do `sendmail` é derrubado. O agente, então, executa o `/etc/init.d/sendmail start` para reiniciar o processo. A forma genérica desse comando é:

```
watch process procName 'procname' index flags interv 'description' 'action'
```

O parâmetro `procname` é uma expressão regular que o SystemEDGE usa para selecionar os processos que está monitorando; nesse caso, estamos observando os processos com o nome `sendmail`. Cada entrada na tabela de monitoração de processos deve ter um índice exclusivo; neste exemplo, usamos o valor 1. Podíamos ter escolhido qualquer inteiro, desde que esse inteiro não estivesse em uso na tabela. O parâmetro `flag` é uma flag hexadecimal\* que altera o comporta-

\* Em termos gerais, há várias maneiras de representar números hexadecimais. O SystemEDGE usa o conceito de um número prefixado com `0x`, que deve ser conhecido para os programadores das linguagens C e Perl.

mento do monitor. Especificamos uma flag de 0x100, que informa ao monitor que o processo sob observação gera processos filhos; essa flag assegura que o SystemEDGE só entre em ação quando o processo *sendmail* pai (e não qualquer um dos filhos) for derrubado.

O uso de flags de monitor de processo está além do âmbito deste capítulo; consulte o manual que acompanha o SystemEDGE, para obter mais informações. O parâmetro *interv* especifica a freqüência (em segundos) com que o agente verifica o status do processo. Definimos o intervalo com 60 segundos. O parâmetro *description* contém informações sobre o processo sendo monitorado e pode ter até 128 caracteres. Convém utilizar uma descrição que indique o que está sendo monitorado, porque o agente guarda esse valor na tabela de monitores para ser recuperado por uma NMS, e o inclui nas vinculações de variáveis quando uma trap é enviada. O último parâmetro é a ação do monitor quando o processo é derrubado; optamos por reiniciar o daemon.

O SystemEDGE pode ser estendido com plug-ins que gerenciam e monitoram aplicativos, como o Apache (servidor da web), Exchange (correio da Microsoft) e Oracle (banco de dados), para citar apenas alguns. Um plug-in dos “principais processos” denominado *topprocs* acompanha cada distribuição. A instrução a seguir orienta o SystemEDGE a carregar esse plug-in para o Solaris de 64 bits (essa instrução é semelhante para o NT e para outras plataformas do Unix):

```
sysedge_plugin /opt/EMPsysedge/plugins/topprocs/topprocs-sol64bit.so
```

O pessoal da Concord teve o cuidado de adicionar comentários úteis ao arquivo *sysedge.cf*. Geralmente, os comentários são tudo o que você precisa para configurar o agente.

## Dispositivos Cisco

A Cisco Systems fabrica uma grande variedade de roteadores, comutadores e outros equipamentos para operação em rede. O processo de configuração é praticamente o mesmo em todos os dispositivos Cisco, porque compartilham o sistema operacional IOS.\* Existem algumas diferenças pequenas nos parâmetros a serem configurados em cada dispositivo; essas diferenças geralmente estão relacionadas aos recursos do dispositivo em questão, e não com a implementação do SNMP.

Para configurar os parâmetros do SNMP, é necessário estar no modo *enable*. Você pode utilizar os comandos a seguir para conhecer as traps disponíveis:

```
router> enable  
Password: mypassword  
router# config terminal  
router(config)#snmp-server enable traps ?
```

\* Existem algumas exceções a essa regra, como os firewalls PIX, que geralmente significam que o produto é fabricado por uma empresa incorporada pela Cisco.

```
bgp          Enable BGP state change traps
envmon      Enable SNMP environmental monitor traps
frame-relay  Enable SNMP frame-relay traps
isdn         Enable SNMP isdn traps
<cr>
```

O ponto de interrogação instrui o roteador a responder com os complementos possíveis para o comando sendo digitado. Você pode utilizar esse recurso na interface de linha de comando inteira.

Se a parte do comando já digitada contiver um erro de sintaxe, o roteador emitirá a mensagem “Unrecognized command” (Comando não reconhecido) quando você digitar o ponto de interrogação. <cr> informa que você pode sair sem configurar o comando (nesse caso, snmp-server enable traps), digitando um *carriage return* (retorno de cursor).

## Configuração simples

Eis uma configuração simples que permite iniciar usando o agente do SNMP:

```
router(config)#snmp-server community private RW
router(config)#snmp-server community public RO
router(config)#snmp-server trap-authentication
router(config)#snmp-server location Delta Building - 1st Floor
router(config)#snmp-server contact J Jones
router(config)#snmp-server host 10.123.135.25 public
```

A maioria desses comandos define parâmetros que você já deve conhecer agora. Definimos duas comunidades, *public* e *private*, com permissões read-only (RO) e read-write (RW), respectivamente. snmp-server trap-authentication ativa as traps de falha de autenticação. O comando snmp-server host 10.123.135.25 public configura o destino das traps. O endereço IP é definido com o endereço de nossa NMS. A string de comunidade *public* será incluída nas traps.

## Configuração avançada

O item de configuração a seguir informa ao dispositivo a interface a ser usada ao enviar traps do SNMP:

```
router(config)#snmp-server trap-source VLAN1
```

Configurar a origem da trap é útil porque os roteadores, por definição, possuem várias interfaces. Esse comando permite enviar todas as traps por meio de uma interface específica

Em algumas ocasiões, você enviará apenas determinadas traps para sua NMS. O item a seguir envia somente traps de monitoração ambiental para o host especificado, 172.16.52.25 (a opção *envmon* não está disponível em todos os dispositivos Cisco):

```
router(config)#snmp-server host 172.16.52.25 public envmon
```

Um dos sets mais assustadores do SNMP é o shutdown da Cisco, que permite encerrar o roteador na NMS. A novidade é que você deve incluir um switch na configuração para que o roteador responda aos comandos de shutdown. Emitir os seguinte comando desativa o shutdown:

```
router(config)#no snmp-server system-shutdown
```

Para receber traps de falha de autenticação (algo tentando consultar o dispositivo com o nome de comunidade incorreto), inclua a seguinte linha:

```
router(config)#snmp-server trap-authentication
```

O último parâmetro de configuração avançada é uma lista de acesso. A primeira linha configura a lista de acesso 15 e declara que o endereço IP 10.123.56.25 pode acessar o agente. A segunda linha informa que quem passar pela lista de acesso 15 (por exemplo, um host com o endereço IP 10.123.56.25) e fornecer o nome de comunidade *notsopublic* tem acesso read-only (RO) ao agente. As listas de acesso são ferramentas muito poderosas para controlar o acesso à rede. Estão além do âmbito deste livro mas se você ainda não as conhece, procure conhecê-las.

```
router(config)#access-list 15 permit 10.123.56.25
```

```
router(config)#snmp-server community notsopublic RO 15
```

Terminou! Agora, você tem uma configuração operacional do SNMP para o roteador da Cisco.

## **APC Symetra**

Os UPSs da APC são típicos de uma grande classe de produtos, que geralmente não são considerados dispositivos de rede, mas incorporaram uma interface de rede para fins de gerenciamento.

Para configurar um UPS da APC, você pode usar a respectiva porta de gerenciamento (uma conhecida porta serial à qual é possível conectar um terminal de console) ou, presumindo que você já tenha executado a configuração básica da rede, telnet para o endereço IP do UPS. A configuração do SNMP é a mesma independentemente do método aplicado. Seja como for, você verá uma Text User Interface (TUI – interface de usuário de texto) que apresenta menus à moda antiga – você digita a opção do menu (geralmente, um número) seguida por Enter para navegar nos menus.

Pressupomos que você já tenha feito a configuração básica da rede, como atribuir um endereço IP para o UPS. Para configurar o SNMP, vá para o menu Network e selecione “5” para entrar no submenu SNMP. Você deve acessar um menu como este:

```

----- SNMP -----
1- Access Control 1
2- Access Control 2
3- Access Control 3
4- Access Control 4
5- Trap Receiver 1
6- Trap Receiver 2
7- Trap Receiver 3
8- Trap Receiver 4
9- System
10- Summary

?- Help
<ENTER> Redisplay Menu
<ESC> Return To Previous Menu
>

```

É necessário configurar três seções distintas: Access Control, Trap Receiver e System. Para ver um resumo das configurações atuais do SNMP, use o submenu Summary.

Esse dispositivo específico permite definir quatro endereços IP para controle do acesso e quatro endereços IP para receber traps. Os itens do controle de acesso permitem configurar os endereços IP e usar estações de gerenciamento – semelhantemente às listas de acesso que examinamos em outros dispositivos e, obviamente, é fundamental para a segurança. O UPS responderá somente às consultas dos endereços IP listados. A configuração é um pouco esquisita – você deve entrar em um menu separado para configurar cada endereço IP. Eis o que você verá ao configurar o submenu Access Control 1:

```

----- Access Control 1 -----
Access Control Summary
# Community Access      NMS IP
-----
1 public     Read        10.123.56.25
2 private    Write       10.123.56.25
3 public2   Disabled    0.0.0.0
4 private2  Disabled    0.0.0.0

1- Community      : public
2- Access Type    : Read
3- NMS IP Address : 10.123.56.25
4- Accept Changes :

?- Help
<ENTER> Redisplay Menu
<ESC> Return To Previous Menu
>

```

A primeira parte do menu resume o estado do controle de acesso. Nesse menu, só é possível modificar o primeiro item na lista. O endereço especial 0.0.0.0 é um curinga – significa que o UPS responderá às consultas de qualquer endereço IP. Embora os endereços 3 e 4 estejam definidos com 0.0.0.0, esses endereços encontram-se desativados no momento, e é assim que os manteremos. Queremos que o UPS responda somente às estações de gerenciamento listadas explicitamente

Nesse menu, configuramos os itens 1 (a string de comunidade), 2 (o tipo de acesso) e 3 (o endereço IP). Definimos a string de comunidade com `public` (não é uma opção que você aprovaria em uma configuração real), o tipo de acesso com `Read` (permitindo várias operações de `get` do SNMP, mas nenhuma operação de `set`) e o endereço IP da NMS com 10.123.56.25. O efeito na rede é que o agente de SNMP do UPS aceitará solicitações de `get` do endereço IP 10.123.56.25 como nome de comunidade `public`. Quando você aprovar a configuração, digite 4 para confirmar as alterações efetuadas.

Para configurar o segundo item de controle de acesso, pressione Esc para retornar ao menu anterior; em seguida, selecione 2. Dessa forma, permitimos que o endereço 10.123.56.25 execute operações de `set`. Como não temos outras estações de gerenciamento, deixamos os itens 3 e 4 desativados.

Após concluir a seção Access Control, você pode iniciar a configuração das traps. A seção Trap Receivers é apenas uma lista de NMSs que recebem traps. Assim como a seção Access Control, é possível configurar quatro receptores de traps. Para acessar o primeiro receptor de trap, volte ao menu SNMP e selecione o menu 5. Uma configuração comum de receptor de traps é parecida com esta:

```
----- Trap Receiver 1 -----  
  
Trap Receiver Summary  
# Community Generation Authentication Receiver NMS IP  
-----  
1 public Enabled Enabled 10.123.56.25  
2 public Enabled Enabled 0.0.0.0  
3 public Enabled Enabled 0.0.0.0  
4 public Enabled Enabled 0.0.0.0  
  
1- Trap Community Name : public  
2- Trap Generation : Enabled  
3- Authentication Traps: Enabled  
4- Receiver NMS IP : 10.123.56.25  
5- Accept Changes :  
  
?- Help  
<ENTER> Redisplay Menu  
<ESC> Return To Previous Menu  
  
>
```

Mais uma vez, a primeira parte do menu é um resumo da configuração dos receptores de traps. Já definimos o primeiro receptor de trap com o endereço de nossa NMS, ativamos a geração de traps e a geração de traps de autenticação – como sempre, é uma boa idéia. As traps que gerarmos incluirão a string de co-

munidade *public*. Observe que os receptores de traps 2, 3 e 4 estão definidos com 0.0.0.0. Nesse menu, 0.0.0.0 não é um curinga; é apenas um endereço inválido, que significa que você ainda não configurou o endereço IP do receptor de traps. É basicamente o mesmo que deixar a entrada desativada.

Os últimos itens de configuração que devem ser definidos são o submenu System, no menu principal do SNMP:

----- System -----

```
1- sysName      : ups1.ora.com
2- sysContact   : Douglas Mauro
3- sysLocation  : Apache Hilo Deck
4- Accept Changes :
```

?- Help

<ENTER> Redisplay Menu

<ESC> Return To Previous Menu

>

Quando você terminar de configurar todos os parâmetros do SNMP, use o submenu Summary para visualizar rapidamente o que você fez. Uma configuração normal ficaria assim:

-----  
SNMP Configuration Summary

```
sysName      : ups1.ora.com
sysLocation  : Apache Hilo Deck
sysContact   : Douglas Mauro
```

Access Control Summary

#	Community	Access	NMS IP
1	public	Read	10.123.56.25
2	private	Write	10.123.56.25
3	public2	Disabled	0.0.0.0
4	private2	Disabled	0.0.0.0

Trap Receiver Summary

#	Community	Generation	Authentication	Receiver	NMS IP
1	public	Enabled	Enabled		10.123.56.25
2	public	Enabled	Enabled		0.0.0.0
3	public	Enabled	Enabled		0.0.0.0
4	public	Enabled	Enabled		0.0.0.0

Press <ENTER> to continue...

Após concluir e verificar, use a tecla Esc para sair imediatamente do menu Logout.

# 8

## Polling e definição

Investimos esforços em todas as configurações para utilizar o SNMP com eficácia. Entretanto, após instalar um gerenciador de nós simulado e configurar agentes em todos os dispositivos, o que podemos fazer? Como interagir com os dispositivos existentes?

As três operações básicas do SNMP são *snmpget*, *snmpset* e *snmpwalk*. Essas operações são auto-explicativas: *snmpget* lê um valor de um dispositivo gerenciado, *snmpset* define um valor em um dispositivo e *snmpwalk* lê uma parte da árvore da MIB de um dispositivo. Por exemplo, você pode utilizar o *snmpget* para consultar um roteador e descobrir o contato administrativo (ou seja, a pessoa a ser chamada se o roteador apresentar problemas), o *snmpset* para modificar essa informação de contato e o *snmpwalk* para percorrer uma MIB de ponta a ponta, para ter uma idéia dos objetos implementados pelo roteador ou para recuperar informações de status sobre as interfaces do roteador.

- Este capítulo ensina a utilizar essas operações no gerenciamento cotidiano da rede. Primeiramente, usaremos a linguagem Perl para demonstrar como é possível definir (com *set*), obter (com *get*) e percorrer (com *walk*) objetos em um script (a vantagem de utilizar a linguagem Perl é a possibilidade de estender com facilidade os scripts simples, apresentados neste capítulo, para adequá-los às suas necessidades e ambiente). Em seguida, usaremos o OpenView da HP e o Net-SNMP para executar as mesmas operações, mas na linha de comando. Finalmente, como uma alternativa para a linha de comando, demonstraremos o Browser de MIB gráfico do OpenView, que tem uma bela interface para obter, definir e percorrer dados de MIB.

### Recuperando um valor de uma MIB individual

Inicialmente, consultaremos um roteador para obter o nome do contato administrativo. Essa operação, denominada *polling*, é executada por meio do comando *get* do SNMP. O script em Perl a seguir, *snmpget.pl*, usa um módulo de Perl do SNMP para recuperar as informações necessárias (no Capítulo 5, você encontrará o URL deste módulo):

```
#!/usr/local/bin/perl
#filename: /opt/local/perl_scripts/snmpget.pl
use BER;
use SNMP_util;
use SNMP_Session;
$MIB1 = ".1.3.6.1.2.1.1.4.0";
$HOST = "orarouter1";
($value) = &snmpget("public@$HOST", "$MIB1");
if ($value) { print "Results :$MIB1: :$value:\n"; }
else { warn "No response from host :$HOST:\n"; }
```

Obviamente, esse script é muito simples, mas é fácil de entender, mesmo que você não seja um usuário experiente em linguagem Perl. O mais importante não é o que o script faz, o que é muito pouco, mas serve como um modelo que você pode utilizar para inserir operações do SNMP em outros programas. (Se você não está acostumado a escrever programas rápidos em Perl ou se não conhece a linguagem, um bom ponto de partida é o site da Web da Perl, <http://www.perl.com>.) O script começa com três instruções `use`, semelhantes às instruções `#include` da linguagem C. As instruções `use` carregam módulo em Perl contendo funções e definições para trabalhar com o SNMP. Os três módulos utilizados são:

#### BER

Descreve como codificar os dados do gerenciamento em padrões de bits para transmissão. O *Basic Encoding Rules* (BER) é um padrão ISO.

#### SNMP\_util

Define um conjunto de funções que usa o módulo `SNMP_Session` para torná-lo amistoso ao programador. O próprio `SNMP_util` usa o padrão BER e o `SNMP_Session` mas, nesse primeiro script, preferimos fazer uma referência explícita a esses outros módulos. Em programas futuros, usaremos somente o `SNMP_util`.

#### SNMP\_Session

Fornece a linguagem Perl com a funcionalidade básicas do SNMP.

As duas linhas seguintes especificam os dados que desejamos obter. Codificamos por hardware a ID de objeto de um fragmento de dado específico, definido pela MIB, e o nome do host em que os dados dessa MIB serão recuperados. Em um programa mais flexível, convém obter esses valores na linha de comando ou criar uma interface do usuário para ajudar os próprios usuários a especificar com exatidão o que desejam recuperar. Entretanto, por enquanto, isso nos ajuda a iniciar. É bastante fácil substituir `orarouter1` pelo nome do host ou endereço IP do dispositivo a ser consultado. A OID que estamos solicitando está armazenada na variável `$MIB1`. O valor `.1.3.6.1.2.1.1.4.0` solicita o contato administrativo do dispositivo. Mais uma vez, enfatizamos que você pode substituir por qualquer OID de sua preferência. Usamos a forma numérica desse objeto, mas você também pode utilizar a forma em texto da OID, que é `.org.dod.internet.mgmt.mib-2.system.sysContact.0`.

É possível abreviar ainda mais para *sysContact* porque *SNMP\_util* define algumas partes da string da OID para nós (por exemplo, *SNMP\_util* define *sysContact* como *1.3.6.1.2.1.1.4.0*), mas geralmente é mais seguro ser explícito e usar a OID inteira. Não se esqueça de incluir o *.0*, que informa que desejamos a primeira (0) e única instância de *iso.org.dod.internetmgmt.mib-2.system.sysContact.0*, no final daOID.

A linha seguinte consulta o dispositivo. A função *snmpget* recupera os dados do dispositivo especificado pela variável \$HOST. Observe os dois argumentos existentes para a função. O primeiro é o dispositivo a ser consultado, precedido pelo nome de comunidade *public*. (Se você precisar utilizar outro nome de comunidade – você realmente mudou os nomes de comunidade quando configurou os dispositivos, não foi? – será necessário modificar essa linha e inserir o nome de comunidade em questão.) O segundo argumento da função *snmpget* é a OID em que estamos interessados. Se você mesmo digitar o código, não se esqueça dos parênteses ao redor de \$value. Se você omitir os parênteses, \$value será definido com o número de itens no array retornado por *snmpget*.

Após consultar o dispositivo, será impressa uma saída ou uma mensagem de erro. Inseri um caractere de dois-pontos antes e depois de qualquer saída impressa; isso facilita a visualização dos possíveis caracteres ocultos existentes na saída. O inteiro decimal “16” é muito diferente de “16\n”, que é o inteiro decimal 16 seguido por um caractere de nova linha.

Executemos, então, o programa:

```
$ /opt/local/perl_scripts/snmpget.pl  
Results :1.3.6.1.2.1.1.4.0: :ORA IT Group:
```

*snmpget.pl* imprime a OID solicitada, seguida pelo valor real desse objeto, que é ORA IT Group. Não se preocupe se o valor de retorno de *sysContact* for incorreto ou em branco. (O truque de inserir caracteres de dois-pontos antes e depois da saída evidenciará se *sysContact* estiver em branco ou vazia.) Provavelmente, isso significa que ninguém configurou um contato administrativo ou que esse contato foi configurado incorretamente. Ensinaremos a corrigir esse problema quando discutirmos a operação de *set*. Se você receber um erro, vá até o final deste capítulo para ver uma lista de alguns erros e os respectivos ajustes adequados.

Modificaremos agora o *snmpget.pl* para consultar qualquer host e qualquer OID almejados. Isso é feito passando o host e a OID como argumentos de linha de comando para o script em Perl:

```
#!/usr/local/bin/perl  
#filename: /opt/local/perl_scripts/snmpget.pl  
use SNMP_util;  
$MIB1 = shift;  
$HOST = shift;  
($MIB1) && ($HOST) || die "Usage: $0 MIB_OID HOSTNAME";  
($value) = &snmpget("$HOST","$MIB1");  
if ($value) { print "Results :$MIB1: :$value:\n"; }  
else { warn "No response from host :$HOST:\n"; }
```

Agora que esse programa está um pouco mais flexível, é possível pesquisar diversos tipos de informações sobre diferentes hosts. Inclusive, excluímos a string de comunidade, que nos permite consultar hosts com nomes de comunidade distintos. Veja a seguir como executar a nova versão do *snmpget.pl*:

```
$ /opt/local/perl_scripts/snmpget.pl .1.3.6.1.2.1.1.1.0 public@orarouter1
Results :.1.3.6.1.2.1.1.1.0: :Cisco Internetwork Operating System Software
IOS (tm) 3000 Software (IGS-I-L), Version 11.0(16), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1997 by cisco Systems, Inc.
Compiled Tue 24-Jun-97 12:20 by jaturner:
```

Neste exemplo, solicitamos ao roteador uma autodescrição, pesquisando a OID *.1.3.6.1.2.1.1.1.0* (*system.sysDesc.0*). O resultado informa que o orarouter1 é um roteador da Cisco executando a Versão 11.0(16) do sistema operacional IOS, além de algumas outras informações úteis.

### *Usando o OpenView da HP para recuperar valores*

Vamos iniciar consultando o contato administrativo de nosso roteador (*system.sysContact.0*), para constatar se obteremos o mesmo resultado do script anterior em Perl. Os argumentos do *snmpget*\* do OpenView são o nome de comunidade, o nome do host do dispositivo a ser consultado e a OID dos dados solicitados; fornecemos a OID no formato numérico mas, novamente, poderíamos tê-la informado como uma string de texto:

```
$ /opt/OV/bin/snmpget -c public orarouter1 .1.3.6.1.2.1.1.4.0
system.sysContact.0 : DISPLAY STRING- (ascii): ORA IT Group
```

Embora essa saída pareça um pouco diferente da do script em Perl, o resultado é o mesmo. *snmpget* imprime a OID solicitada na linha de comando, facilitando confirmar que consultamos o objeto correto. Mais uma vez, observe que o *.0* final é importante. A saída também nos informa o tipo de dado do objeto: *DISPLAY STRING- (ascii)*. No Capítulo 2, discutimos os tipos de dados utilizados pelo SNMP; alguns dos mais conhecidos são *INTEGER*, *OCTET STRING*, *Counter* e *IpAddress*. Finalmente, a saída nos fornece as informações solicitadas: o roteador é administrado pelo ORA IT Group, que é o valor retornado pela solicitação de *get* do SNMP.

Agora, façamos o mesmo usando a interface da GUI do OpenView. Quando o Network Node Manager for exibido, selecione "Misc SNMP MIB Browser".\*\* Se o NNM não estiver em execução, você poderá iniciar o MIB Browser na linha de comando: */opt/OV/bin/xnmbsrowser*. A Figura 8-1 mostra a GUI, cujos campos são semelhantes às variáveis que temos definido em nossos scripts em Perl: Name ou IP Address, Community Name, MIB Object ID, MIB Instance, SNMP Set Value e MIB Values.

\* A maioria dos arquivos executáveis do OpenView está localizada em */opt/OV/bin*.

\*\* Se você descobrir que o item de menu SNMP MIB Browser está esmaecido e não recebe cliques, clique em um objeto do SNMP em seu mapa do NNM. A partir de então, você poderá clicar no item de menu para iniciar sua GUI.

Usemos esse navegador para executar um *snmpget*. Comece inserindo um Nome (em Name) ou Endereço IP (em IP Address) e um nome de comunidade (em Community Name) nos campos de entrada fornecidos. Para inserir o objeto a ser recuperado, use o campo MIB Object ID e a caixa de texto abaixo desse campo. O campo MIB Object ID informa que estamos atualmente na subárvore *.iso.org.dod.internet*. A área de texto mostra os objetos do nível seguinte da árvore: *directory*, *mgmt*, etc. (Para ver as OIDs numéricas desses objetos, clique em seus nomes e, em seguida, no botão “Describe”.) Depois, percorra a MIB, clicando duas vezes em *mgmt*, depois em *mib-2*, *system* e, por último, em *sysContact*. Clique em ai e, em seguida, em “Start Query”. O resultado exibido no campo “MIB Values” (como mostra a Figura 8-2) deve ser parecido com o valor retornado no exemplo da linha de comando.

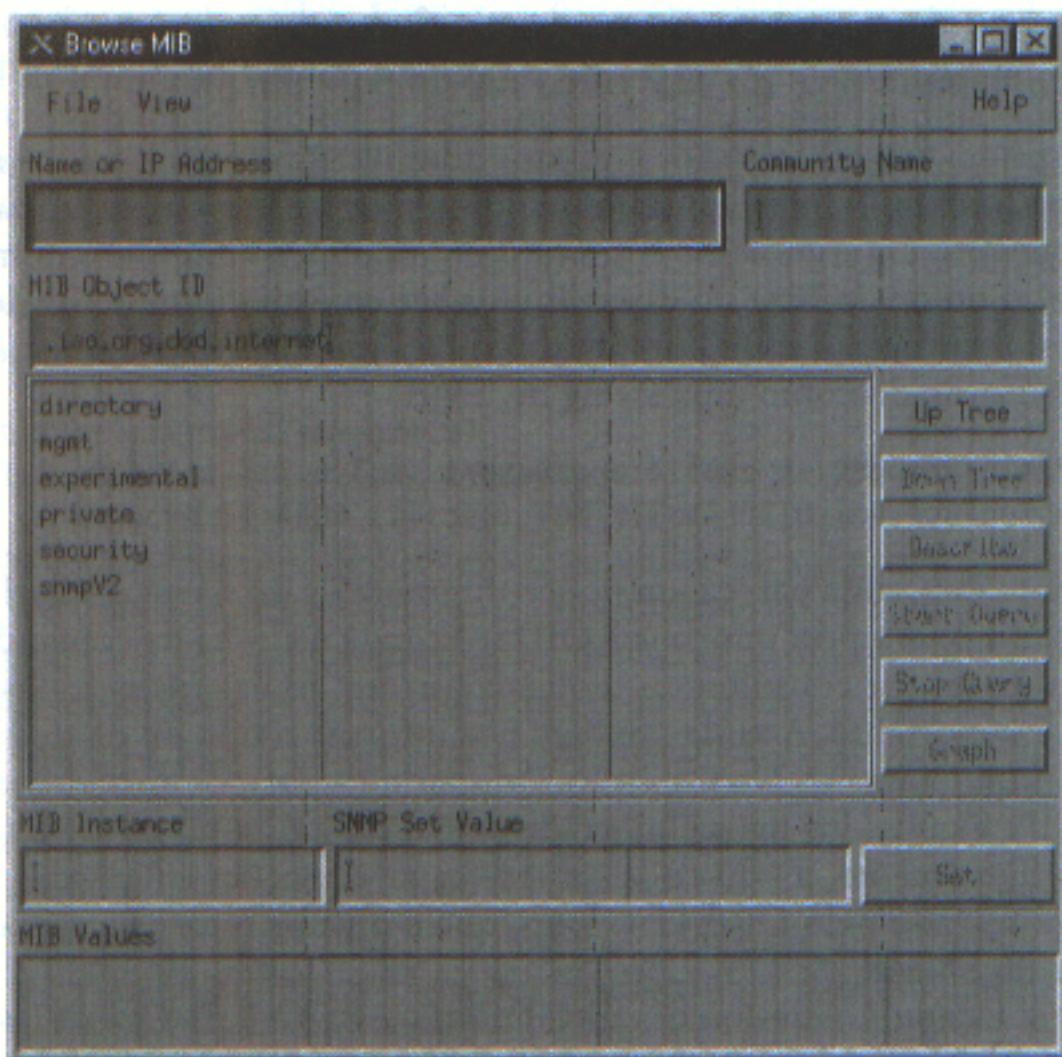


Figura 8-1 Default do xnmrowser do OpenView

Vamos voltar à linha de comando e pesquisar *sysDesc* novamente:

```
$ /opt/OV/bin/snmpget orarouter1 .1.3.6.1.2.1.1.1.0
system.sysDescr.0 : DISPLAY STRING- (ascii): Cisco Internetwork Operating
System Software IOS (tm) 3000 Software (IGS-I-L), Version 11.0(16), RELEASE
SOFTWARE (fc1)Copyright (c) 1986-1997 by cisco Systems, Inc. Compiled Tue
24-Jun-97 12:20 by jaturner
```

Parece o mesmo, concorda? Observe que ignoramos a string de comunidade. Podemos fazê-lo porque a string de comunidade padrão do *get* é *public*, que é a string de comunidade correta para o host de destino, *orarouter1*. Você pode alterar suas strings de comunidade padrão nas definições globais do OpenView. Vejamos se é possível obter um objeto com outro tipo de dado:

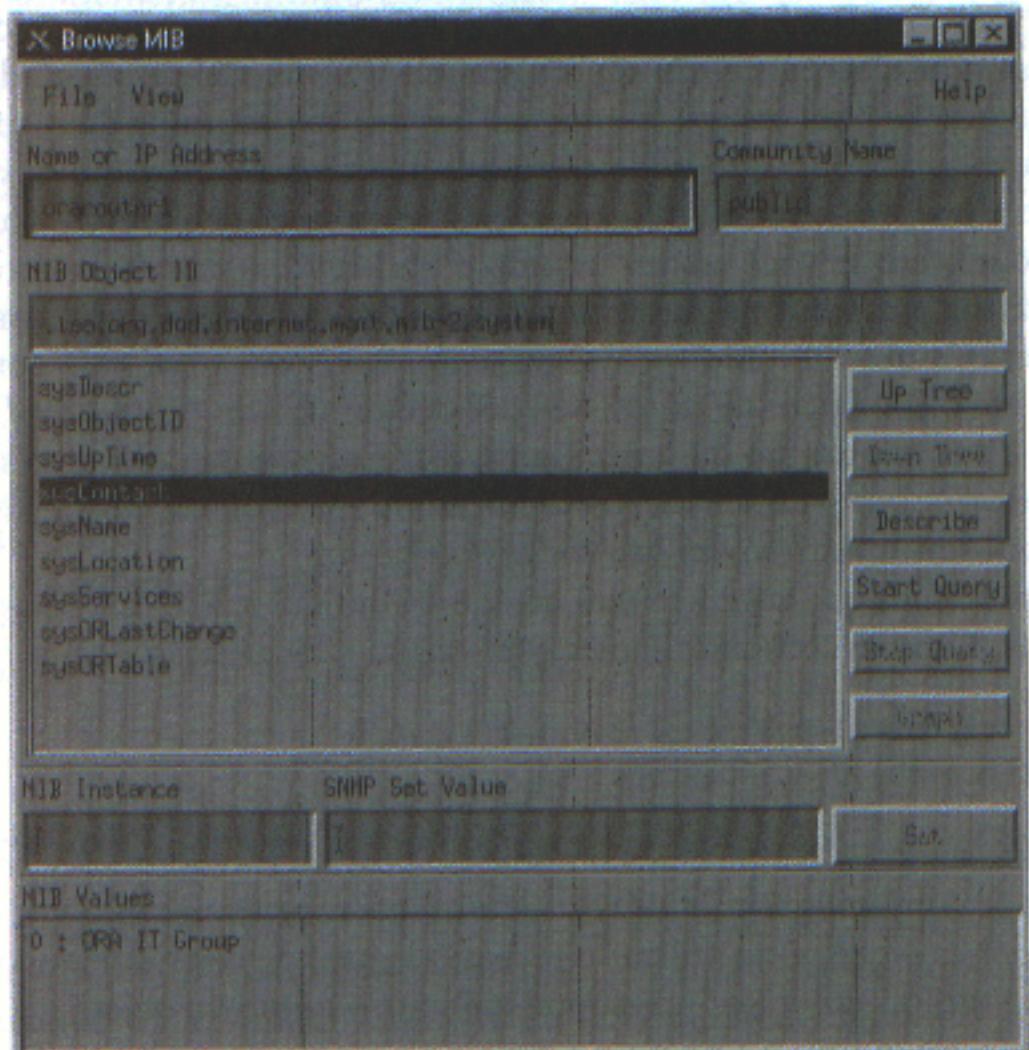


Figura 8-2 Resposta do xnmrowser do OpenView

```
$ /opt/OV/bin/snmpget orarouter1 .1.3.6.1.2.1.1.3.0
system.sysUpTime.0 : Timeticks: (159857288) 18 days, 12:02:52.88
```

Esse comando retorna o tempo de funcionamento do sistema, que é do tipo TimeTicks. TimeTicks (RFC 1155) representa um inteiro não negativo, que conta o tempo em centésimos de segundo a partir de um momento específico. Ignorando o número entre parênteses, sabemos que o roteador está funcionando há 18 dias, 12 horas, 2 minutos, etc. O número enorme entre parênteses é o período de tempo exato de funcionamento da máquina, em centésimos de segundo. Se você fizer os cálculos, descobrirá 18.501 dias ou 18 dias, 12 horas, e um pouco mais: exatamente o que esperamos.

## Usando o Net-SNMP

As ferramentas do Net-SNMP oferecem uma excelente interface de linha de comando para as operações do SNMP. Essas ferramentas também são comumente denominadas UCD-SNMP – você também encontrará esse nome mais antigo em algumas referências, e até mesmo no próprio código.

O Capítulo 7 abordou a compilação, instalação e configuração do agente do Net-SNMP. Se você fez isso, já compilou e instalou as ferramentas do SNMP, fornecidas no mesmo pacote do agente do SNMP, e não há necessidade de qualquer configuração para essas ferramentas. Existe um programa de configuração, denominado *snmpconf*, que pode ser usado para gerar um arquivo *snmp.conf* que fornece valores predefinidos para algumas opções para os comandos.\* Entretanto, exceto se você estiver usando o SNMPv3, não é realmente necessário. Talvez seja prático configurar uma string de comunidade default mas, na realidade, será de uso limitado: provavelmente, você tem strings de comunidade diferentes em dispositivos distintos. Se você decidir utilizar o *snmpconf* para criar o arquivo de configuração da ferramenta, certifique-se de colocar o *snmp.conf* no subdiretório *.snmp* de seu diretório inicial ou (para que as opções sejam aplicáveis a todos os usuários) em */usr/local/share/snmp*.

Presumiremos que você não fará qualquer configuração e usaremos apenas as ferramentas. Eis uma consulta simples que solicita a localização de um roteador:

```
$ snmpget orarouter1 public .1.3.6.1.2.1.1.6.0  
system.sysLocation.0 = Sebastopol CA
```

É muito simples: fornecemos o nome do host do roteador a ser consultado, uma string de comunidade e a OID do objeto a ser recuperado. Em vez de utilizar a OID numérica, você pode utilizar o formato legível pelo ser humano. Para economizar digitação, o *snmpget* pressupõe tudo até o nome do objeto e a ID da instância. Por conseguinte, o comando a seguir é exatamente idêntico ao anterior:

```
$ snmpget orarouter1 public sysLocation.0  
system.sysLocation.0 = Sebastopol CA
```

Examinaremos os comandos *snmpwalk* e *snmpset* fornecidos no pacote do Net-SNMP, mais adiante neste capítulo, mas o pacote possui várias ferramentas e compensa uma explicação com mais detalhes. Uma ferramenta particularmente útil é o *snmptranslate*, que converte nomes números e textuais de objetos MIB e pode executar atividades como procurar a definição de um objeto em um arquivo de MIB. Na distribuição do software constam algumas MIBs padrão; você pode inserir arquivos de MIB adicionais em */usr/local/share/snmp/mibs*. O Apêndice C apresenta uma visão geral do pacote do Net-SNMP.

\* Este é o mesmo comando utilizado para criar o *snmpd.conf*, que configura o agente do Net-SNMP. O arquivo de configuração do *snmp.conf* em o formato semelhante ao *snmpd.conf*.

## Recuperando diversos valores de MIB

A sintaxe do *snmpwalk* é semelhante à de seu parceiro, *snmpget*. Conforme discutido no Capítulo 2, o *snmpwalk* percorre uma MIB a partir de um objeto específico, retornando continuamente valores até alcançar o final da ramificação desse objeto. Por exemplo, o script em Perl a seguir comece a percorrer o objeto *.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr* e fornece uma descrição de cada interface de Ethernet existente no dispositivo consultado.

Esse novo script é uma pequena modificação do *snmpget.pl*. Transformamos o escalar \$value no array @values;\* necessitamos de um array porque esperamos receber vários valores. Além disso, chamamos a função *snmpwalk* em vez de *snmpget* (a sintaxe das duas funções é idêntica):

```
#!/usr/local/bin/perl
#filename: /opt/local/perl_scripts/snmpwalk.pl
use SNMP_util;
$MIB1 = shift;
$HOST = shift;
($MIB1) && ($HOST) || die "Usage: $0 MIB_OID HOSTNAME";
(@values) = &snmpwalk("$HOST","$MIB1");
if (@values) { print "Results :$MIB1: :@values:\n"; }
else { warn "No response from host :$HOST:\n"; }
```

Eis como executar o script:

```
$ /opt/local/perl_scripts/snmpwalk.pl .1.3.6.1.2.1.2.2.1.2 orarouter1
```

Esse comando percorre o objeto *.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr*, retornando informações sobre as interfaces existentes no roteador. Os resultados são parecidos com os seguintes:

```
Results :.1.3.6.1.2.1.2.2.1.2: :1:Ethernet0 2:Serial0 3:Serial1:
```

A saída depende das interfaces existentes no host ou roteador sendo consultado. Para apresentar alguns exemplos, executei esse script em algumas máquinas de minha rede. Examine os resultados.

Roteador Cisco 7000:

```
Results :.1.3.6.1.2.1.2.2.1.2: :1:Ethernet0/0 2:Ethernet0/1
3:TokenRing1/0
4:TokenRing1/1 5:TokenRing1/2 6:TokenRing1/3 7:Serial2/0 8:Serial2/1
9:Serial2/2 10:Serial2/3 11:Serial2/4 12:Serial2/5 13:Serial2/6
14:Serial2/7
15:FastEthernet3/0 16:FastEthernet3/1 17:TokenRing4/0 18:TokenRing4/1:
```

\* O programa em Perl que usamos anteriormente poderia ter utilizado o array em vez do escalar também. Isso é possível porque a versão Perl do *snmpget* aceita diversas OIDs, não somente uma. Para especificar várias OIDs, insira um caractere de vírgula (,) entre cada OID. Lembre-se de colocar cada OID entre aspas duplas.

Estação de trabalho da Sun:

Results : .1.3.6.1.2.1.2.2.1.2: :1:lo0 2:hme0:

PC do Windows NT:

Results : .1.3.6.1.2.1.2.2.1.2: :1:MS TCP Loopback interface  
2:PCI2 Token-Ring Network 16/4 Adapter :

No-break APC:

Results : .1.3.6.1.2.1.2.2.1.2: :1:peda:

Para cada dispositivo, vemos pelo menos uma interface. Como você esperava, o roteador possui várias interfaces. A primeira interface existente no roteador está listada como 1:Ethernet0/0, a segunda está listada como 2:Ethernet0/1, e assim por diante, até a interface 18. O SNMP rastreia as interfaces em uma tabela, que pode ter várias entradas. Até mesmo os dispositivos domésticos têm duas entradas na tabela: uma para a interface da rede e outra para a interface de loopback. O único dispositivo no exemplo acima, que realmente possui uma única interface, é o no-break APC – mas até nesse caso, o SNMP rastreia a interface por meio de uma tabela indexada por um número de instância.

Esse recurso permite anexar um número de instância a uma OID, para procurar um elemento específico da tabela. Por exemplo, usariamos a OID .1.3.6.1.2.1.2.2.1.2.1 para examinar a primeira interface do roteador da Cisco, .1.3.6.1.2.1.2.2.1.2.2 para examinar a segunda, e assim por diante. Em uma forma mais legível, *ifDescr.1* é o primeiro dispositivo da tabela de descrição de interfaces, *ifDescr.2* é o segundo dispositivo e assim, sucessivamente.

### *Percorrendo a árvore de MIB com o OpenView*

Alternando para o comando *snmpwalk* do OpenView, tentemos obter cada objeto contido na subárvore *.iso.org.dod.internet.mgmt.mib-2.system*:

```
$ /opt/OV/bin/snmpwalk oraswitch2 .1.3.6.1.2.1.1
system.sysDescr.0 : DISPLAY STRING- (ascii): Cisco Internetwork
Operating
System Software IOS (tm) C2900XL Software (C2900XL-H-M), Version 11.2(8)
SA1,RELEASE SOFTWARE (fc1)Copyright (c) 1986-1998 by cisco Systems, Inc.
Compiled Tue 03-Feb-98 14:59 by rheaton
system.sysObjectID.0: OBJECT IDENTIFIER:
.iso.org.dod.internet.private.enterprises.cisco.ciscoProducts.cisco2509
system.sysUpTime.0 : Timeticks: (168113316) 19 days, 10:58:53.16
system.sysContact.0 : DISPLAY STRING- (ascii): J.C.M. Pager 555-1212
system.sysName.0 : DISPLAY STRING- (ascii): oraswitch2.ora.com
system.sysLocation.0 : DISPLAY STRING- (ascii): Sebastopol CA
system.sysServices.0 : INTEGER: 6
```

Passemos para a GUI MIB Browser e experimentemos esse mesmo percurso. Repita as etapas aplicadas para o *snmpget*, usando a GUI. Dessa vez, insira a OID *.1.3.6.1.2.1.1* e pressione o botão “Start Query”. Verifique os resultados.



A GUI detecta se é necessário executar um *snmpwalk* ou *snmpget*. Se você fornecer um valor de instância (sendo específico), o navegador executará um *snmpget*. Caso contrário, emitirá um *snmpwalk*. Se você procura mais velocidade e menos despesas em sua rede, inclua o valor da instância.

O que acontecerá se você percorrer a subárvore *.iso* completa? Pode prejudicar ou até derrubar sua máquina porque, na maioria dos casos, o dispositivo pode retornar alguns milhares de valores. Cada interface em um roteador pode adicionar milhares de valores às tabelas da MIB do roteador. Se cada objeto demorar 0,0001 segundo para calcular e retornar, e existirem 60.000 valores a serem retornados, serão necessários ao dispositivo 6 segundos para retornar todos os valores – sem contar a carga sobre a rede ou sobre a estação de monitoração. Se possível, é sempre conveniente executar um *snmpwalk* a partir da subárvore da MIB que fornecerá as informações específicas que você procura, em vez de percorrer a MIB inteira.

Seria útil ter uma noção da quantidade de objetos de MIB implementados por um dispositivo específico. Uma maneira de fazer isso é contar o número de objetos retornados por cada *snmpwalk*. Isso pode ser feito com o comando *grep* do Unix. A opção *-c* do *grep* instrui a retornar o número de linhas encontradas. O caractere do ponto (.) instrui o *grep* a fazer uma correspondência com tudo. Começando no objeto *.system (.1.3.6.1.2.1.1)*, retornemos um e constatemos quantos objetos são implementados na subárvore *mib-2*. Retire o último *.1* da ID de objeto e execute o comando *snmpwalk* novamente, dessa vez, canalizando os resultados para *grep -c*:

```
$ /opt/0V/bin/snmpwalk oraswitch2 .1.3.6.1.2.1 | grep -c .
```

O número de objetos visualizado dependerá do tipo de dispositivo e do software em execução nesse dispositivo. Quando experimentei alguns dispositivos diferentes, obtive resultados que variavam de 164 a 5193.

Esse comando é excelente para percorrer uma MIB e verificar todos os tipos de valores que um dispositivo pode retornar. Ao experimentar um novo dispositivo ou uma MIB, geralmente percorro uma parte razoável da MIB e examino todos os valores retornados, procurando quaisquer informações interessantes. Quando algo me chama a atenção, entra na definição da MIB e leio sua descrição. Alguns MIB Browsers de MIB permitem verificar a descrição com o clique de um botão. Na GUI do OpenView, clique na OID e, em seguida, em “Describe”.

## Percorrendo a árvore com o Net-SNMP

O *snmpwalk* do Net-SNMP tem forma e função parecidas com o do OpenView. Eis como usá-lo:

```
$ snmpwalk orarouter1 public .1.3.6.1.2.1.1
system.sysDescr.0 = Cisco Internetwork Operating System Software
IOS (tm) C820 Software (C820-Y6-M), Version 12.1(3)XG3, EARLY DEPLOYMENT
RELEASE
SOFTWARE (fc1)
TAC:Home:SW:IOS:Specials for info
Copyright (c) 1986-2000 by cisco Systems, Inc.
Compiled Wed 20-Dec-00 16:21
system.sysObjectID.0 = OID: enterprises.9.1.284
system.sysUpTime.0 = Timeticks: (100946413) 11 days, 16:24:24.13
system.sysContact.0 = thenetworkadministrator@oreilly.com
system.sysName.0 = orarouter1@oreilly.com
system.sysLocation.0 = Sebastopol CA
system.sysServices.0 = 6
system.sysORLastChange.0 = Timeticks: (0) 0:00:00.00
```

Na realidade, nenhuma surpresa aqui. Mais uma vez, você pode utilizar um nome de objeto em vez de uma ID numérica; como você está percorrendo uma árvore, não é necessário especificar um número de instância.

## Definindo um valor de MIB

Com o *snmpget* e *snmpwalk*, temos recuperado informações de gerenciamento somente de dispositivos. A próxima etapa lógica é modificar o valor de um objeto da MIB via SNMP. Essa operação é conhecida como *snmpset* ou *set*. Nesta seção, leremos o valor de um objeto, usaremos o *snmpset* para alterar seu valor, depois leremos o valor novamente para confirmar se foi modificado.

Evidentemente, há um perigo aqui: o que acontece se você mudar uma variável crítica para o estado do sistema que você está monitorando? Neste capítulo, lidaremos apenas com alguns objetos simples, como o contato administrativo, que não prejudicará nada se for modificado incorretamente. Portanto, se você mantiver as OIDs corretas, não precisará se preocupar em provocar danos em quaisquer dos dispositivos. Todos os objetos definidos neste capítulo possuem ACCESS leitura-gravação. Convém ter uma idéia dos objetos graváveis, a consultar a MIB em que os objetos estão definidos – em uma das RFCs ou em um arquivo de MIB fornecido por seu fornecedor.

Mãos á obra! Execute o seguinte comando do OpenView (ou use um dos outros programas discutidos) para descobrir o *sysContact* do dispositivo selecionado:

```
$ /opt/OV/bin/snmpget -c public orarouter1 .1.3.6.1.2.1.1.4.0
system.sysContact.0 : DISPLAY STRING- (ascii): ORA IT Group
```

A opção `-c public` passa a string de comunidade `public` para o comando `snmpget`.



Lembre-se de que seus dispositivos não devem utilizar as mesmas strings de comunidade (default) usadas neste livro. Além disso, é uma péssima idéia utilizar a mesma string para as comunidades read-only (`snmpget`) e read-write (`snmpset`).

Executemos agora o comando `snmpset` do OpenView. Esse comando usa o valor especificado entre aspas na linha de comando para definir o objeto indicado pela OID especificada. Use a mesma OID (`system.sysContact.0`). Como o novo valor de `sysContact` contém palavras e possivelmente números, também devemos especificar o tipo de variável `octetstring`.\* Execute o comando `snmpset` do OpenView com os seguintes parâmetros:

```
$ /opt/OV/bin/snmpset -c private orarouter1 .1.3.6.1.2.1.1.4.0 \
octetstring "Meg A. Byte 555-1212"
system.sysContact.0 : DISPLAY STRING- (ascii): Meg A. Byte 555-1212
```

O resultado mostra que o `snmpset` modificou com êxito a pessoa de contato do roteador para Meg A. Byte 555-1212. Se você não receber esse resultado, o *set* não foi bem-sucedido. A Tabela 8-2 traz algumas mensagens de erro comuns que provavelmente você receberá, e as etapas para corrigir os problemas. Para confirmar o valor armazenado pelo dispositivo em `sysContact`, podemos repetir o comando `snmpget`.

Ao utilizar a GUI do OpenView, é mais fácil visualizar, definir e confirmar. Use a GUI para obter o valor de `sysContact`. Após confirmar a presença de um valor, digite uma descrição na caixa de texto SNMP Set Value. Como existe somente uma instância para o `sysContact`, insira um 0 (zero) para a MIB Instance (Instância da MIB). Após preencher todos os itens de entrada obrigatória, clique no botão “Set” posicionado à direita da caixa de texto “SNMP Set Value”. É exibida uma janela pop-up com a mensagem “Set has completed successfully.” (Set bem-sucedido.) Para verificar se o *set* realmente ocorreu, clique em “Start Query”. (Agora, você já deve ter certeza de que o uso de uma GUI como o programa MIB Browser do OpenView facilita muito mais a obtenção e definição de objetos da MIB.)

Para demonstrar como fazer isso por programação, escreveremos outro script pequeno em Perl, denominado `snmpset.pl`:

```
#!/usr/local/bin/perl
#filename: /opt/local/perl_scripts/snmpset.pl
use SNMP_util;
```

\* Se você consultar a RFC 1213 (MIV-II), observará que `sysLocation` tem uma SYNTAX de `DisplayString`. Na realidade, essa é uma convenção de texto do tipo OCTET STRING, com um tamanho de 0..255 octetos.

```

$MIB1 = ".1.3.6.1.2.1.1.6.0";
$HOST = "oraswitch2";
$LOC = "@ARGV";
($value) = &snmpset("private@$HOST", "$MIB1", 'string', "$LOC");
if ($value) { print "Results :$MIB1: :$value:\n"; }
else { warn "No response from host :$HOST:\n"; }

```

Vamos executar este script:

```

$ /opt/local/perl_scripts/snmpset.pl A bld JM-10119 floor 7
Results :.1.3.6.1.2.1.1.6.0: :A bld JM-10119 floor 7:

```

Usando o script *snmpget.pl*, podemos verificar se o *set* aconteceu:

```

$ /opt/local/perl_scripts/snmpget.pl .1.3.6.1.2.1.1.6.0 public@oraswitch2
Results :.1.3.6.1.2.1.1.1.0: :A bld JM-10119 floor 7:

```

Agora, usaremos o utilitário *snmpset* do Net-SNMP para modificar o contato do sistema:

```

$ snmpset oraswitch2 private sysContact.0 s myself
system.sysContact.0 = myself
$ snmpget oraswitch2 public sysContact.0
system.sysContact.0 = myself

```

Não há nada confuso aqui. Fornecemos uma string de comunidade, um nome de host e uma ID de objeto, seguidos por uma tipo de dado (s de String) e o novo valor de *sysContact*. Só para confirmar que o *set* realmente aconteceu, emitir, em seguida, um *snmpget*. A única informação adicional que você precisa saber é o mecanismo para especificar tipos de dados. O Net-SNMP usa as abreviações de um caractere apresentadas na Tabela 8-1.

Tabela 8-1 Abreviações de tipo de dado do Net-SNMP

Abreviação	Significado
a	Endereço IP
b*	Bits
d	Seqüência decimal
D	Duplo
F	Flutuante
i	Inteiro
I	int64 com sinal
n	Nulo
o	ID de objeto
s	String
t	Contagem de tempo

**Tabela 8-1 Continuação**

Abreviação	Significado
u	Inteiro não atribuído
U	int64 sem sinal
x	Seqüência hexadecimal

\* As páginas do manual apresentam como um tipo de dado válido, mas a Ajuda da saída do comando não.

## Respostas de erro

A Tabela 8-2 mostra as respostas de erro que um dispositivo pode retornar ao executar os comandos apresentados neste capítulo. Consulte sua documentação local se essas explicações não englobarem seu problema específico.

**Tabela 8-2 Tabela de respostas de erros**

Resposta do servidor	Explicação
Contained under subtree	O <code>snmpwalk</code> retorna esse erro se você tiver tentado percorrer uma MIB, mas já estiver no final, ou se a árvore não existir no cliente.
No response arrived before timeout	Possíveis causas são nome de comunidade inválido, o agente não está em execução ou o nó está inacessível.
Agent reported error with variable	Você está tentando definir com um objeto com um tipo de dado que não é o mesmo (nem próximo a) do tipo especificado da variável. Por exemplo, se a variável deseja um <code>DisplayString</code> , você receberá esse erro se enviar um <code>INTEGER</code> para ela. Pesquise a MIB para conhecer o tipo de <code>SYNTAX</code> necessário à variável.
Missing instance value for . . .	Ao definir um valor, forneça a OID completa e uma instância. Um objeto escalar terminará com zero (0) e um objeto tabular terminará com o número de instância do objeto em uma tabela. Verifique se o número de instância que você está usando com o <code>snmpget</code> está correto e repita o <code>set</code> .
Access is denied for variable	Pode acontecer se você tentar definir um valor em um objeto somente leitura. Examine a MIB qual é a definição de <code>ACCESS</code> do objeto.

## Polling e limiares

O SNMP permite verificar seus dispositivos regularmente, obtendo informações sobre o gerenciamento desses dispositivos. Além disso, é possível informar à NMS que existem determinados limiares que, quando ultrapassados, exigirão algum tipo de ação. Por exemplo, você poderia ser notificado se o tráfego em uma interface saltasse para um valor extremamente alto (ou baixo); esse evento poderia indicar um problema na interface ou capacidade insuficiente, ou até mesmo um ataque hostil à sua rede. Quando tal condição ocorrer, a NMS poderá direcionar um alarme para um mecanismo de correlação de eventos ou instruir um ícone a piscar em um mapa do OpenView. Para tornar tudo isso mais concreto, vamos supor que a NMS esteja verificando o status de uma interface em um roteador. Se a interface for derrubada, a NMS informará o que aconteceu para que o problema seja rapidamente solucionado.

O SNMP pode fazer a polling interna ou externa. Geralmente, a polling interna é usada em combinação com um aplicativo executado como um *daemon* ou uma facilidade como um *cron*, que executa periodicamente uma aplicação local. A polling externa é feita pela NMS. A NMS do OpenView oferece uma excelente implementação de polling externa, que pode grafar e salvar seus dados para recuperação posterior ou enviar uma notificação se ocorrer algo errado. Alguns pacotes de software fornecem NMSs eficientes e, se você sabe criar scripts, pode incorporar uma NMS adaptada às suas necessidades. Neste capítulo, examinaremos alguns pacotes disponíveis.

Fazer polling é como verificar o nível do óleo de um carro; essa analogia pode ajudá-lo a esquematizar estratégias de polling adequadas. Três aspectos diferentes são relevantes ao verificar o óleo: o processo físico (abrir o reservatório, retirar a vareta medidora de nível e reinseri-la); a marcação predefinida que indica se existe um problema (se o nível do óleo está muito alto, muito baixo ou no nível correto?) e a freqüência da verificação do óleo (uma vez por hora, semana, mês ou ano?).

Vamos supor que você peça ao mecânico para checar o nível do óleo de seu carro. Esse processo é parecido com o de uma NMS enviando um pacote para um roteador executar um *snmpget* por algum fragmento de informação.

Quando o mecânico terminar, você pagará a ele 30 reais e prosseguirá seu caminho. Como um nível baixo de óleo pode resultar danos no motor, convém verificar o nível do óleo regularmente. Sendo assim, quanto tempo você deve esperar para que o mecânico verifique o carro novamente? A verificação do nível do óleo tem um custo: nesse cenário, você pagou 30 reais. Nas redes, você paga pela largura de banda. Assim como o dinheiro, você não pode desperdiçar a largura de banda frivolamente. Então, a pergunta é: quanto tempo você pode esperar para verificar o nível do óleo novamente, para não estourar seu orçamento?

A resposta está no próprio carro. Um carro de corrida bem ajustado precisa ter os fluidos nos níveis ideais. Um VW Beetle,\* ao contrário de um carro de corrida, pode dispor de aproximadamente um-quarto de qualquer período de tempo, sem prejudicar o desempenho. Provavelmente, você não está dirigindo um Beetle, mas provavelmente não está dirigindo um carro de corridas também. Então, você decide que pode verificar o nível do óleo a cada 3 semanas. Entretanto, como saber o que é realmente um nível baixo, alto ou correto?

A vareta medidora do nível de óleo do carro pode informar. O mecânico não precisa conhecer o modelo do carro, o tipo de motor nem o volume de óleo contido no carro; ele só precisa saber ler o que a vareta indica. Em uma rede, a vareta de um dispositivo é denominada uma agente e a leitura da vareta é o pacote de resposta do SNMP. Todos os dispositivos compatíveis com o SNMP possuem agentes padronizados (varetas medidoras) que podem ser lidos por qualquer mecânico (NMS). É importante lembrar que os dados obtidos são tão eficientes quanto o agente ou o mecânico que os gerou.

Em ambos os casos, um limiar predefinido determina a ação adequada. No exemplo do óleo, o limiar é “nível baixo”, o que aciona uma resposta automática: colocar óleo. (Ultrapassar o limiar “nível alto” pode acionar um tipo de resposta diferente.) Se considerarmos uma interface de um roteador, os possíveis valores recebidos são “funcionando” e “paralisada”. Imagine que o gateway para a Internet em sua empresa, uma porta em um roteador, deve permanecer funcionando 24 horas por, 7 dias por semana. Se essa porta for derrubada, você pode perder 10.000 reais a cada segundo de paralisação. Você verificaria essa porta freqüentemente? A maioria das organizações não paga a alguém para verificar as interfaces do roteador a cada hora. Mesmo que você tivesse tempo, isso não teria graça alguma, concorda? É exatamente aí que a polling do SNMP entra em cena. Ela permite que os gerentes da rede garantam o funcionamento correto dos dispositivos de missão crítica, sem precisar pagar a alguém para monitorar constantemente os roteadores, servidores etc.

Assim que você avaliar suas necessidades de monitoração, poderá especificar o intervalo de verificação de um dispositivo ou grupo de dispositivos. Geralmente, esse intervalo é citado como intervalo de polling e pode ser ter a granularidade necessária (por exemplo, a cada segundo, por hora etc.). O valor limiar para tomar uma ação não precisa ser binário: você pode detectar que algo está realmente errado se o número de pacotes de saída de sua conexão com a Internet ficar abaixo de um nível específico.

\* Os modelos抗igos dos anos sessenta, não os modernos de imitação.



Sempre que você avaliar a freqüência de polling de um dispositivo, lembre-se de três aspectos: o agente/CPU do dispositivo, o consumo de largura de banda e os tipos de valores que você está solicitando. Alguns valores recebidos podem ser médias de 10 minutos. Se esse for o caso, é um desperdício verificar com um intervalo de poucos segundos. Examine as MIBs relacionadas aos dados que você está pesquisando. Minha preferência é começar a polling logo no início. Tão logo eu tenha os valores das tendências e de picos, tomo uma atitude, que pode aumentar o congestionamento na rede mas garante que nenhuma informação importante desapareça.

Seja qual for a freqüência dessa polling, lembre-se de outros aspectos que podem estar ocorrendo na rede. Certifique-se de escalar os períodos de polling para evitar outros eventos, se possível. Lembre-se dos backups, carregamentos de dados, atualizações de direcionamento e de outros eventos que podem sobrecarregar as redes ou CPUs.

## *Polling interna*

Talvez pareça um desperdício de largura de banda verificar um dispositivo só para confirmar que está tudo correto. Em um dia comum, você pode fazer uma polling em dezenas de dispositivos, centenas ou milhares de vezes, sem detectar quaisquer falhas nem interrupções. Evidentemente, é exatamente isso que você deseja descobrir – e provavelmente concluirá que o SNMP atendeu a seu propósito, na primeira vez em que você descobrir um dispositivo com defeito e conseguir retornar o dispositivo on-line antes que os usuários comecem a reclamar. Entretanto, na melhor das hipóteses, você teria os benefícios da polling sem o custo: isto é, sem dedicar muita largura de banda de sua rede para monitorar o funcionamento.

Nesse momento, a polling interna entra em ação. Como o próprio nome indica, a polling interna é executada por um agente interno ou incorporado ao dispositivo a ser gerenciado. Como a polling é interna ao dispositivo, não exige tráfego entre o agente e sua NMS. Além disso, o agente encarregado da polling não precisa ser um agente real do SNMP, o que pode permitir monitorar sistemas (máquinas ou softwares) que não têm suporte para o SNMP. Por exemplo, alguns fornecedores de condicionadores de ar de porte industrial fornecem informações do status operacional por meio de uma porta serial. Se a unidade condicionadora de ar estiver acoplada a um servidor de terminais ou dispositivo semelhante, será fácil utilizar linguagens de script para monitorar a unidade e gerar traps se a temperatura ultrapassar um limiar estabelecido. Esse programa interno pode ser escrito em sua linguagem de script favorita e pode verificar quaisquer informações de status que você possa acessar. Tudo o que você precisa é de um método de obtenção de dados do script para a estação de gerenciamento.

Uma estratégia para escrever um programa de polling é utilizar “ganchos” em um programa para extrair as informações que podem ser direcionadas posteriormente para uma trap do SNMP e enviadas para a NMS. As traps serão discutidas com mais detalhes no Capítulo 10. Outro método de fazer a polling interna é usar um programa (por exemplo, o *sh*, Perl ou C) executado em intervalos definidos.

(No Unix, você usaria um *cron* para executar um programa em intervalos fixos; existem serviços semelhantes em outros sistemas operacionais.) Os ganchos e os scripts orientados por *cron* permitem verificar variáveis internas e informar os erros encontrados. Examine a seguir um script em Perl que verifica a presença de um arquivo e envia uma trap se o arquivo não for encontrado:

```
#!/usr/local/bin/perl
# Filename: /opt/local/perl_scripts/check4file.pl

use SNMP_util "0.54"; # Carregará os módulos BER e SNMP_Session para nós

$FILENAME = "/etc/passwd";

#
# se o arquivo /etc/passwd não existir, envie uma trap!
#
if(!(-e $FILENAME)) {
    snmptrap("public:@nms:162", ".1.3.6.1.4.1.2789", "sunserver1", 6, 1547, \
".1.3.6.1.4.1.2789.1547.1", "string", "File \:$FILENAME\: Could\
    NOT Be Found");
}
```

Um crontab ao estilo Sun é parecido com este:

```
$ crontab -l

# Procure este arquivo a cada 15 minutos e informe uma trap se não encontrado
4,19,34,49 * * * * /opt/local/perl_scripts/check4file.pl
```

Observe que verificamos quatro minutos a cada quarto de hora, em vez de 15 em 15 minutos. A próxima polling que inseriremos no arquivo *crontab* pode executar cinco minutos após o quarto de hora (5,20,35,50). Esse procedimento evita a inicialização de um grande número de programas simultaneamente. É uma idéia particularmente boa para evitar a polling a cada hora cheia – esse é um horário consagrado para os programas em geral e jobs de *cron* inicializarem. Consulte a página do manual sobre *cron* se você desconhece sua operação.

## Remote Monitoring (RMON)

RMON é um suplemento para o grupo da MIB-II. Esse grupo, se aceito pelo agente de SNMP do dispositivo, permite a polling interna e externa. Podemos verificar dispositivos por meio de uma NMS remota (polling externa) ou ins-

truir o agente de RMON local a verificar-se periodicamente e informar quaisquer erros ocorridos (polling interna). O agente de RMON enviará traps quando forem detectadas condições de erro.

Alguns dispositivos aceitam o RMON, tornando-o um mecanismo eficaz para a polling interna. Por exemplo, a Cisco aceita as categorias de ROM, Events e Alarms. É possível configurar a categoria Alarms para verificar MIBs internamente e reagir de diferentes modos quando um limiar for ultrapassado ou reduzido. Cada limiar tem a opção de chamar um Evento interno. A Figura 9-1 mostra o fluxo dessas duas categorias de RMON.

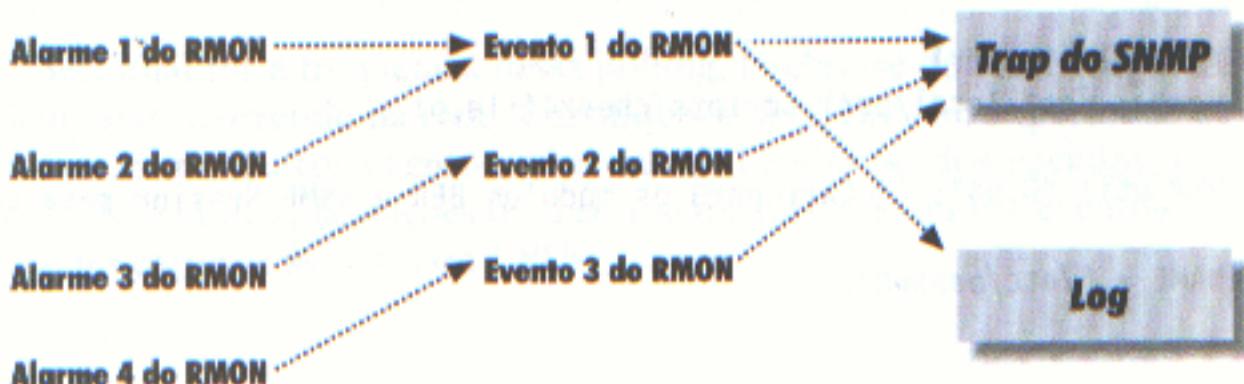
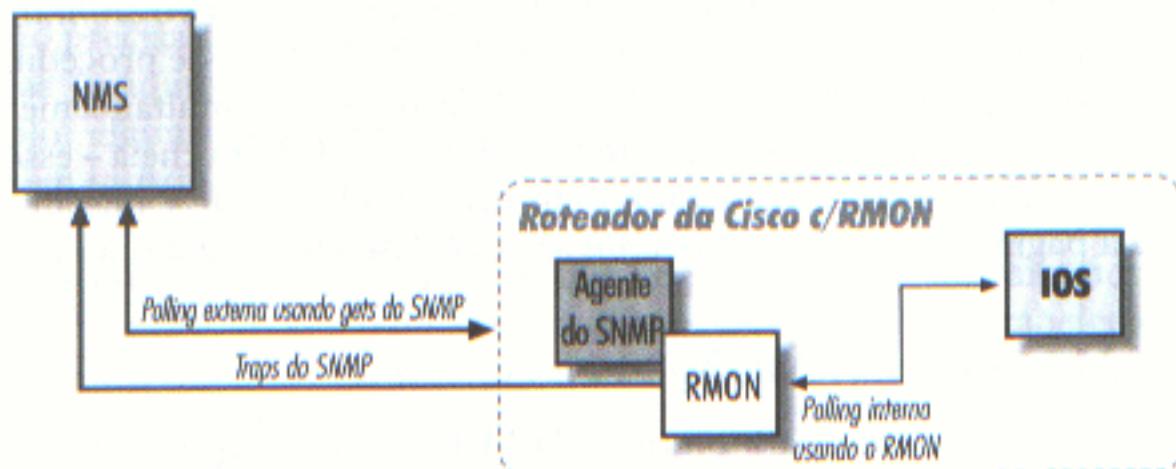


Figura 9-1 Fluxo de processos do RMON

A diferença entre alarmes e eventos é importante. Cada alarme está associado a um evento específico, que define a ação a ser executada quando o alarme disparar. Assim que um limiar for atingido, acionando um alarme, o alarme chamará o evento, que pode executar funções adicionais, incluindo o envio de traps para a NMS e escrever um registro em um log. As traps de SNMP padrão são pré-configuradas pelo fornecedor do agente, e os gerentes da rede não têm qualquer controle sobre a definição de quaisquer tipos de limiares; entretanto, o RMON permite que um gerente da rede defina limiares crescentes e decrescentes. A Figura 9-2 representa a interação entre o agente de RMON de um roteador e uma NMS.



128 | Figura 9-2 Interação entre o RMON e a NMS

Na Figura 9-2, o agente de SNMP do roteador da Cisco envia uma trap para a NMS. Observe a direção da comunicação: a transmissão da trap do RMON é unidirecional. A NMS recebe a trap do roteador da Cisco e determina a ação a ser acionada, se aplicável.

Além de enviar traps, também podemos protocolar eventos; se necessário, podemos até protocolar o agente sem gerar uma trap. Esse protocolo pode ser particularmente útil durante a configuração inicial de alarmes e eventos de RMON. Se você tornar as condições de alarmes muito sensitivas, poderá congestionar sua NMS com eventos de RMON acionados a todo momento. O protocolo também pode ajudar a ajustar os alarmes de RMON antes de liberá-los para produção.

## Configuração do RMON

Como exemplo prático de como configurar o RMON, usaremos a implementação do RMON da Cisco, começando pelos eventos. O comando de IOS a seguir define um evento de RMON:

```
rmon event número [log] [comunidade da trap] [string da description]  
[string do owner]
```

Se você conhece o IOS, deve estar esperando um comando *no* correspondente que descarta um evento de RMON:

```
no rmon event número
```

Os parâmetros para esses comandos do IOS são:

*número*

Especifica o número de identificação exclusivo do evento. Esse valor deve ser maior que 0; não é permitido o valor 0.

*log*

Instrui o agente a registrar a entrada quando acionado. Esse argumento é opcional.

*comunidade da trap*

Especifica a string de comunidade da trap; por exemplo, uma string de comunidade a ser incluída com a trap. Alguns programas de gerenciamento de rede podem ser configurados para responder somente às traps com uma string de comunidade específica.

*string de description*

Descreve o evento.

*string de owner*

Associa o evento ou item a uma pessoa específica.

Examine a seguir dois exemplos de como criar eventos de RMON da Cisco. A primeira linha cria um alarme ascendente, que facilita o envio de uma trap para a NMS. A segunda gera um alarme descendente que pode indicar que o tráfego voltou a um nível aceitável (esse alarme é registrado mas não gera uma trap):

```
(config)#rmon event 1 log trap public description "High ifInOctets" owner dmauro  
(config)#rmon event 2 log description "Low ifInOctets" owner dmauro
```

Você também pode utilizar o protocolo de registro para rastrear quando os eventos foram chamados. Embora seja possível configurar traps sem protocolo de registro, o que acontece se a linha para sua NMS cair? O protocolo de registro assegura que você não perca informações quando a NMS estiver desabilitada. Sugemos o uso do log e da trap em todos os seus eventos. Você pode visualizar os logs de seus eventos de RMON ao emitir o seguinte comando no roteador:

```
orarouter1# show rmon event
```

Event 1 is active, owned by dmauro

Description is High ifInOctets

Event firing causes log and trap to community public, last fired 00:05:04

Current log entries:

index	time	description
1	00:00:31	High ifInOctets
2	00:05:04	High ifInOctets

Event 2 is active, owned by dmauro

Description is Low ifInOctets

Event firing causes log, last fired 00:00:11

Current log entries:

index	time	description
1	00:00:11	Low ifInOctets

O comando a seguir percorre a tabela de eventos de *rmon*, que exibe os valores que acabamos de definir:

```
$ snmpwalk orarouter1 .iso.org.dod.internet.mgmt.mib-2.rmon.event.eventTable  
rmon.event.eventTable.eventEntry.eventIndex.1 : INTEGER: 1  
rmon.event.eventTable.eventEntry.eventIndex.2 : INTEGER: 2  
rmon.event.eventTable.eventEntry.eventDescription.1  
                                : DISPLAY STRING- (ascii): High ifInOctets  
rmon.event.eventTable.eventEntry.eventDescription.2  
                                : DISPLAY STRING- (ascii): Low ifInOctets  
rmon.event.eventTable.eventEntry.eventType.1 : INTEGER: log-and-trap  
rmon.event.eventTable.eventEntry.eventType.2 : INTEGER: log  
rmon.event.eventTable.eventEntry.eventCommunity.1 : OCTET STRING- (ascii): public  
rmon.event.eventTable.eventEntry.eventCommunity.2 : OCTET STRING- (ascii):  
rmon.event.eventTable.eventEntry.eventLastTimeSent.1 : Timeticks: (0) 0:00:00.00  
rmon.event.eventTable.eventEntry.eventLastTimeSent.2 : Timeticks: (0) 0:00:00.00  
rmon.event.eventTable.eventEntry.eventOwner.1 : DISPLAY STRING- (ascii): dmauro  
rmon.event.eventTable.eventEntry.eventOwner.2 : DISPLAY STRING- (ascii): dmauro  
rmon.event.eventTable.eventEntry.eventStatus.1 : INTEGER: valid  
rmon.event.eventTable.eventEntry.eventStatus.2 : INTEGER: valid
```

A maioria das informações definidas na linha de comando está disponível por meio do SNMP. Vemos dois eventos, com os índices 1 e 2. O primeiro evento tem a descrição High ifInOctets; ele é registrado e uma trap é gerada; a string de comunidade do evento é public; o criador do evento é dmauro; o evento é valid (válido), o que significa basicamente que está ativado; e também vemos que o evento ainda não ocorreu. Em vez de utilizar a linha de comando para definir esses eventos, poderíamos ter usado o *snmpset* para criar novos eventos ou para modificar eventos já existentes. Se você tomar esse caminho, lembre-se de definir *eventEntry.eventStatus* com 1, de “válido”, para que o evento funcione corretamente.



Você pode verificar os objetos *ifDescr* e *ifType* na subárvore *mgmt.interfaces.ifEntry* para ajudá-lo a identificar o número da instância que você deve utilizar para seus dispositivos. Se você estiver usando um dispositivo com várias portas, talvez precise procurar o *ifType*, *ifAdminStatus* e *ifOperStatus* para ajudá-lo a identificar o quê é o quê. Na próxima seção, veremos que não é necessário rastrear essas variáveis da MIB (o software de polling externa se encarrega disso).

Agora que nossos eventos estão configurados, iniciemos a configuração dos alarmes para fazer uma polling interna. Precisamos saber o que iremos verificar, que tipo de dado será retornado e a freqüência dessa polling. Suponha que o roteador seja nosso gateway default com a Internet. Desejamos verificar a segunda interface do roteador, que é a interface serial. Sendo assim, desejamos verificar a *mgmt.interfaces.ifEntry.ifInOctets.2* para obter o número de octetos de saída nessa interface, que é um tipo INTEGER.\* Para ser exato, o objeto de MIB *ifInOctets* está definido como “O número total de octetos recebidos na interface, incluindo os caracteres da moldura”. (O .2 no final da OID indica a segunda entrada na tabela *ifEntry*. Em nosso roteador, denota a segunda interface, que é a que queremos verificar.) Queremos ser avisados se o tráfego nessa interface ultrapassar 90.000 octetos/segundo; presumiremos que o funcionamento se normalizará quando o tráfego cair para menos de 85.000 octetos/segundo. Isso nos fornece os limiares ascendente e descendente para nosso alarme. Em seguida, precisamos saber o intervalo de polling para esse objeto. Iniciemos com uma polling a cada 60 segundos.

Agora, precisamos inserir todas essas informações em um comando *alarm* do RMON da Cisco. Eis o comando que cria um alarme:

```
rmon alarm number variable interval {delta | absolute}  
    rising-threshold value [event-number]  
    falling-threshold value [event-number]  
    [owner string]
```

\* Na RFC 1757, a *alarmVariable* (o objeto/MIB que verificaremos) deve ser resolvida para um tipo primitivo ASV.1 de INTEGER, Counter, Gauge ou TimeTicks.

O comando a seguir descarta o alarme:

```
no rmon alarm number
```

Os parâmetros para esses comandos são:

```
number
```

Especifica o número de identificação exclusivo atribuído ao alarme.

```
variable
```

Especifica o objeto de MIB a ser monitorado.

```
interval
```

Especifica a freqüência em que o alarme monitora a variável de MIB.

```
delta
```

Indica que os valores de limiar especificados no comando devem ser interpretados em termos da diferença entre as leituras sucessivas.

```
absolute
```

Indica que os valores de limiar especificados no comando devem ser interpretados como valores absolutos; ou seja, a diferença entre o valor atual e os valores anteriores é irrelevante.

```
rising-threshold value event-number
```

Especifica o valor em que o alarme deve ser acionado, chamando o evento, quando o valor estiver subindo. *event-number* é o evento que deve ser chamado quando o alarme ocorrer. O número do evento é opcional porque o limiar não precisa receber a atribuição de um evento. Se um dos limiares ficar em branco, o número do evento será definido com 0, e não executará qualquer ação.

```
falling-threshold value event-number
```

Especifica o valor em que o alarme deve ser acionado, chamando o evento, quando o valor estiver caindo. *event-number* é o evento que deve ser chamado quando o alarme ocorrer. O número do evento é opcional porque o limiar não precisa receber a atribuição de um evento. Se um dos limiares ficar em branco, o número do evento será definido com 0, e não executará qualquer ação.

```
owner string
```

Associa esse alarme a uma pessoa específica.

Para configurar as definições do alarme que acabamos de descrever, digite o seguinte comando, no modo *configuration*, em um console Cisco:

```
orarouter1(config)#rmon alarm 25 ifEntry.10.2 60 absolute \
rising-threshold 90000 1 falling-threshold 85000 2 owner dmauro
```

Esse comando configura o número de alarme 25, que monitora o objeto em *ifEntry.10.2* (instância 2 de *ifEntry.ifInOctets*, ou os octetos de entrada na interface 2) a cada 60 segundos. Tem um limiar ascendente de 90.000 octetos, que possui o evento nº 1 associado: o evento 1 é chamado quando o tráfego nessa interface ultrapassa 90.000 octetos/segundo. O limiar descendente está definido com 85.000 octetos e tem o evento nº 2 associado. Eis como fica o alarme nas tabelas internas do roteador:

```
orarouter1#show rmon alarm
```

```
Alarm 1 is active, owned by dmauro
Monitors ifEntry.10.2 every 60 second(s)
Taking absolute samples, last value was 87051
Rising threshold is 90000, assigned to event 1
Falling threshold is 85000, assigned to event 2
On startup enable rising or falling alarm
```

A última linha da saída informa que o roteador habilitará o alarme durante a reinicialização. Como você esperava, também é possível examinar as definições do alarme por meio da RMON MIB, começando na subárvore 1.3.6.1.2.1.16. Assim como acontece com os próprios eventos, podemos criar, modificar, editar e excluir entradas usando o *snmpset*.

Um problema na polling interna é a dificuldade de obter tendências e visualizar os dados em um gráfico ou tabela. Mesmo que você desenvolva o sistema de back-end para obter os objetos da MIB e os exiba graficamente, a recuperação dos dados é dolorosa. O MRTG (Multi Router Traffic Grapher) é um excelente programa que permite a polling interna e externa. Além disso, é elaborado para gerar gráficos de seus dados no formato HTML. O MRTG será discutido no Capítulo 13.

## Polling externa

Geralmente, é impossível fazer a polling interna de um dispositivo, por motivos técnicos, de segurança ou de diretrizes. Por exemplo, o grupo System Administration pode não ter o hábito de divulgar a senha da raiz (root), dificultando a instalação e manutenção de scripts de polling interna. Contudo, podem não ter problemas para instalar e manter um agente do SNMP como o SystemEDGE da Concord ou Net-SNMP. Também é possível que você se encontre em um ambiente em que você não sabe construir as ferramentas necessárias à polling interna. Apesar da situação, se existir um agente do SNMP em uma máquina que possui objetos que compensam ser verificados, você poderá utilizar um dispositivo externo para fazer uma polling na máquina e ler os valores dos objetos.\* Esse dispositivo externo pode ser uma ou mais NMSs ou outras máquinas ou dispositivos. Por

\* Alguns dispositivos informam uma compatibilidade com o SNMP, mas suportam somente algumas MIBs. Isso praticamente impossibilita uma polling. Se você não tem o(s) objeto(s) que será(ão)verificado(s), não pode fazer milagres, exceto se existirem ganhos para um agente extensível. Mesmo com agentes extensíveis, a menos que você saiba programar, o "S" (de simples) do SNMP passará bem longe.

exemplo, no caso de uma rede de tamanho razoável, às vezes é conveniente e, possivelmente, necessário, distribuir a polling entre várias NMSs.

Cada mecanismo de polling externa que examinaremos usa os mesmos métodos de polling, embora algumas NMSs implementem a polling externa de modo diferente. Começaremos pelo programa *xnmgraph* do OpenView, que pode ser usado para obter e exibir dados graficamente. Inclusive, você pode utilizar o OpenView para salvar os dados para recuperação e análise posteriores. Incluiremos alguns exemplos para demonstrar como é possível obter dados e armazená-los automaticamente e como recuperar esses dados para exibição. O SNMPc da Castle Rock também dispõe de um excelente recurso de obtenção de dados que usaremos para obter e grafar os dados.

## *Obtendo e exibindo dados com o OpenView*

Um dos métodos mais fáceis para conseguir alguns gráficos interessantes com o OpenView é usar o programa *xnmgraph*. Você pode executar o *xnmgraph* na linha de comando e alguns dos menus do NNM. Um método prático de grafar é por meio do *xnmbrowser*\* do OpenView, para obter alguns dados e, em seguida, clicar em “Graph”. É fácil assim. Se o nó sendo verificado tiver mais de uma instância (digamos, várias interfaces), o OpenView grafará todas as instâncias conhecidas. Quando uma NMS consulta um dispositivo como um roteador, determina a quantidade de instâncias existentes em *ifTable* e recupera dados de gerenciamento para cada entrada na tabela.

## *Representação gráfica no OpenView*

A Figura 9-3 mostra o tipo de gráfico que você pode criar com o NNM. Para criar esse gráfico, iniciamos o navegador (Figura 8-2) e clicamos para expandir a árvore da MIB até encontrarmos a lista *.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry*. Nessa lista, clicamos em *ifInOctets*; em seguida, mantendo pressionada a tecla Ctrl, clicamos em *ifOutOctets*. Com ambos selecionados e após confirmar que o campo “Name or IP Address” exibe o nó que desejamos verificar, clicamos no botão “Graph”.

Após iniciar o gráfico, é possível modificar o intervalo de polling e as cores usadas para exibir os diversos objetos. Você também pode desativar a exibição de algumas ou de todas as instâncias do objeto. O item de menu “View → Line Configuration” permite especificar os objetos a serem exibidos; também pode definir os multiplicadores de diferentes itens. Por exemplo, para exibir tudo o que existe em K, multiplique os dados por .001. Também existe uma opção (“View → Statistics”) que mostra um resumo estatístico de seu gráfico. A Figura 9-4 apresenta alguns dados estatísticos do gráfico da Figura 9-3. Enquanto o menu Statistics estiver aberto, você pode clicar com o botão esquerdo no gráfico; a janela de dados estatísticos exibirá os valores da data e hora específicas para as quais você está apontando com o mouse.

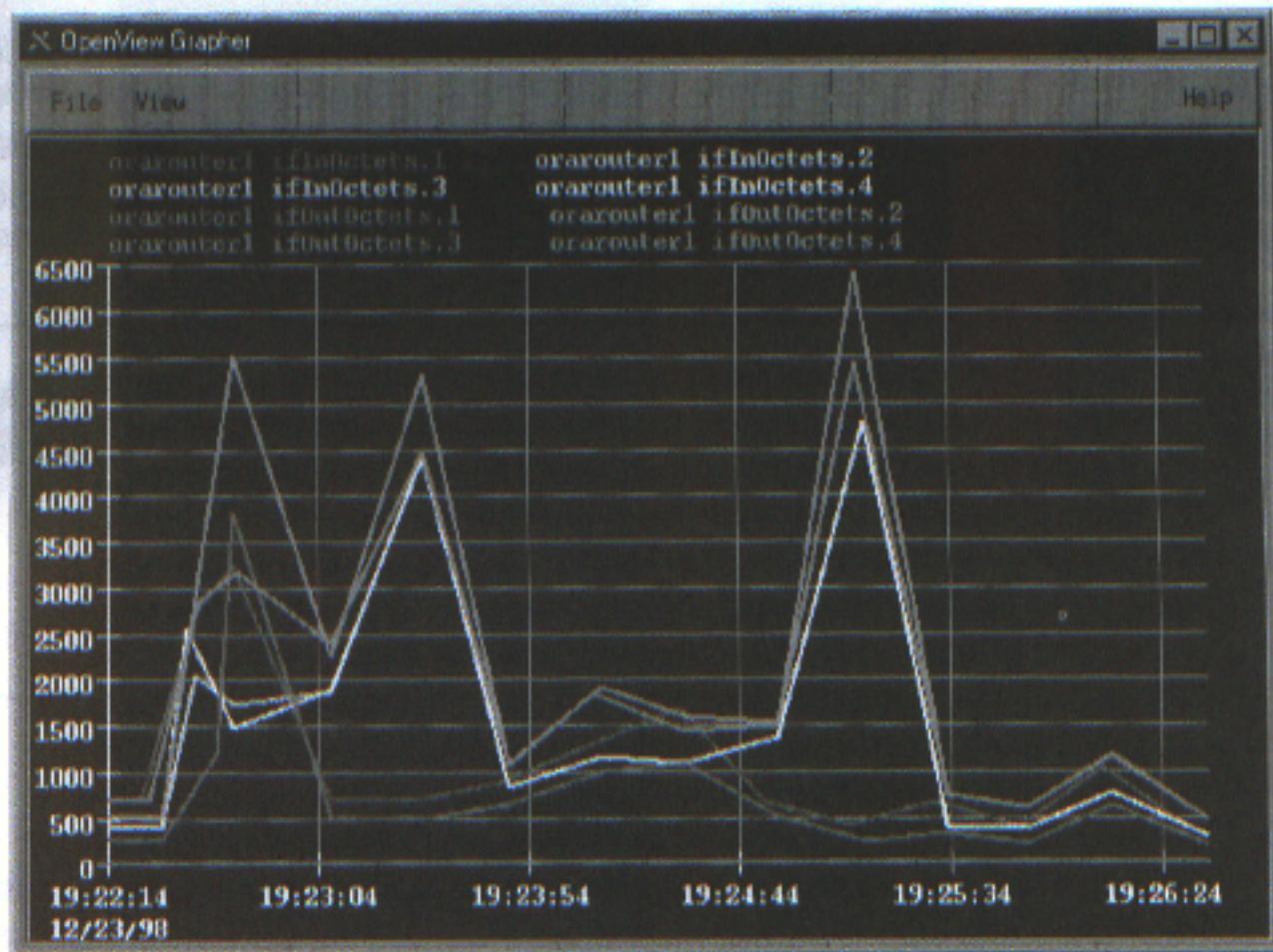


Figura 9-3 O xnmgraph de octetos de entrada/saída, do OpenView

Line	Minimum	Average	Maximum	Last Value
orarouter1_ifInOctets.1	875.16	793.95	2069.66	2069.66
orarouter1_ifInOctets.2	276.99	1106.54	4773.98	1544.64
orarouter1_ifInOctets.3	0.00	0.00	0.00	0.00
orarouter1_ifInOctets.4	275.97	1105.79	4777.44	3548.53
orarouter1_ifOutOctets.1	167.64	473.00	3443.99	3443.99
orarouter1_ifOutOctets.2	462.16	1609.71	6421.61	2688.47
orarouter1_ifOutOctets.3	0.00	0.00	0.00	0.00
orarouter1_ifOutOctets.4	460.98	1495.76	5441.12	2437.46

Figura 9-4 Dados estatísticos do xnmgraph

Alguns menus default do NNM permitem utilizar o grafador para verificar dispositivos, de acordo com o tipo. Por exemplo, Você pode selecionar o tipo de objeto “roteador” no NNM e gerar um gráfico que inclui todos os roteadores. Quer você inicie na linha de comando ou no menu, em algumas ocasiões, você receberá a mensagem “Requesting more lines than number of colors (25). Reducing number of lines.”. (*Solicitando mais linhas que o número de cores (25). Reduzindo número de linhas.*) Essa mensagem indica que não existem cores suficientes disponíveis para exibir os objetos que você está tentando grafar. Os únicos métodos eficazes para evitar esse problema são dividir os gráficos de modo a fazer polling de menos objetos ou eliminar as instâncias dos objetos des-

necessários. Por exemplo, provavelmente você não deseja grafar as interfaces paralisadas do roteador (seja qual for o motivo) e outros objetos “mortos”. Veremos mais adiante como é possível utilizar uma expressão regular como um dos argumentos para o comando *xnmgraph* grafar somente as interfaces em funcionamento e execução.



Iniciar o *xnmgraph* na linha de comando permite iniciar o grafador com um período de polling específico e oferece várias outras opções. Por default, o OpenView verifica em intervalos de 10 segundos. Na maioria dos casos, isso é perfeito, mas se você estiver verificando um roteador de várias portas para checar se algumas portas estão congestionadas, um intervalo de polling de 10 segundos pode ser muito rápido e talvez provoque problemas operacionais. Por exemplo, se a CPU estiver ocupada, respondendo a consultas do SNMP a cada 10 segundo, o roteador pode se tornar muito lento, principalmente se for responsável por OSPF (Open Shortest Path First – protocolo de roteamento para redes IP) ou por outras tarefas que consomem bastante a CPU. Talvez você receba mensagens do OpenView reclamando que outra polling foi iniciada enquanto era aguardado o retorno da polling anterior. Geralmente, aumentar o intervalo de polling elimina essas mensagens.

Embora a interface gráfica seja muito prática, a interface de linha de comando oferece mais flexibilidade. O script a seguir exibe o gráfico da Figura 9-3 (isto é, o gráfico que geramos por meio do navegador):

```
#!/bin/sh
# filename: /opt/0V/local/scripts/graphOctets
# syntax: graphOctets <hostname>
/opt/0V/bin/xnmgraph -c public -mib \
".iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifInOctets::::::::::," \
".iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifOutOctets::::::::::" \
$1
```

Você pode executar esse script com o comando:

```
$ /opt/0V/local/scripts/graphOctets orarouter1
```

A pior parte de escrever o script é conhecer as opções de linha de comando necessárias – principalmente as longas strings de nove opções separadas por caracteres de dois-pontos. Todas essas opções permitem ajustar o que você deseja grafar, a freqüência de polling dos objetos e o modo de exibição dos dados. (Discutiremos a sintaxe dessas opções durante a explicação, mas para conseguir a história completa, consulte a página do manual relacionada ao *xnmgraph(1)*.) Nesse script, estamos grafando os valores de dois objetos de MIB, *ifInOctets* e *ifOutOctets*. Cada OID que desejamos grafar é a primeira (e neste caso, a única) opção na seqüência de opções separadas por caracteres de dois-pontos. Em nossa rede, esse comando gera oito traços: octetos de entrada e saída para cada uma das qua-

tro interfaces. Você pode adicionar outras OIDs para grafar, incluindo conjuntos de opções mas, em algum ponto, o gráfico se tornará muito confuso para ser considerado útil. É necessário praticar para conseguir utilizar o comando *xnmgraph* com eficiência, mas quando você aprender a gerar gráficos úteis, ficará se perguntando como você conseguiu trabalhar até agora sem esses gráficos.



Manter os script inteligíveis é não somente uma boa prática, como também mais agradável em termos estéticos. O uso de uma “\” no final de uma linha indica que a linha seguinte é uma continuação da atual. Quebrar linhas com inteligência torna os scripts mais legíveis. Saiba que as shells do Unix não gostam de espaço em branco adicional depois do caractere da “\” (barra invertida). O único caractere aceito depois da “\” deve ser um retorno de cursor.

Agora, vamos modificar o script de modo a incluir rótulos mais razoáveis – em termos mais específicos, gostaríamos que o gráfico mostrasse cada interface, em vez de apresentar apenas o número de índice. Em nosso script modificado, usamos IDs de objetos numéricas, principalmente por questão de formatação, e incluímos uma sexta opção à terrível seqüência de opções separadas por caracteres de dois-pontos: .1.3.6.1.2.1.2.2.1.2 (esse é o objeto *ifDescr*, ou a descrição da interface, na tabela de interfaces). Essa opção instrui a fazer uma polling de cada instância e utilizar como rótulo o valor de retorno de *snmpget*, .1.3.6.1.2.1.2.2.1.2.INSTANCE, o que deve nos oferecer rótulos bem significativos. Eis o novo script:

```
#!/bin/sh
# filename: /opt/OV/local/scripts/graphOctets
# syntax: graphOctets <hostname>
/opt/OV/bin/xnmgraph -c public -title Bits_In_n_Out -mib \
".1.3.6.1.4.1.9.2.2.1.1.6:::::.1.3.6.1.2.1.2.2.1.2:::, \
.1.3.6.1.4.1.9.2.2.1.1.8:::::.1.3.6.1.2.1.2.2.1.2:::" $1
```

Para ver o que obteremos para os rótulos, eis o resultado de percorrer .1.3.6.1.2.1.2.2.1.2:

```
$ snmpwalk orarouter1 .1.3.6.1.2.1.2.2.1.2
interfaces.ifTable.ifEntry.ifDescr.1 : DISPLAY STRING- (ascii):
Ethernet0
interfaces.ifTable.ifEntry.ifDescr.2 : DISPLAY STRING- (ascii): Serial0
interfaces.ifTable.ifEntry.ifDescr.3 : DISPLAY STRING- (ascii): Serial1
```

A Figura 9-5 mostra o novo gráfico. Com a inclusão dessa sexta opção, fica mais fácil ler os nomes e rótulos.

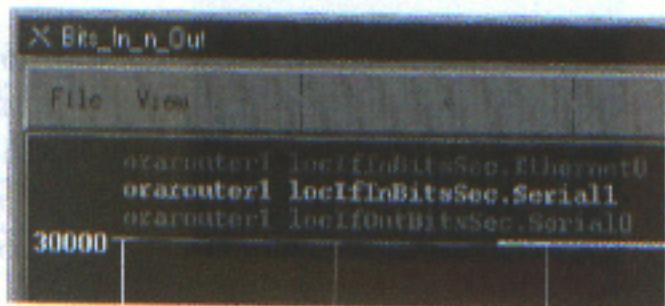


Figura 9-5 O *xnmgraph* do OpenView, com novos rótulos

Rótulos e títulos significativos são importantes, principalmente se a diretoria está interessada em ver os gráficos. Um rótulo que contém uma OID e não uma descrição em texto é inútil. Alguns objetos úteis ao criar rótulos são *ifType* (.1.3.6.1.2.1.2.2.1.3) e *ifOperStatus* (.1.3.6.1.2.1.2.2.1.8). Tenha cuidado ao utilizar *ifOperStatus*; se o status da interface mudar durante uma polling, o rótulo não será alterado. O rótulo é avaliado uma única vez.

Um dos maiores desperdícios é fazer uma polling de um objeto inútil. Geralmente, isso acontece quando uma interface está paralisada, em termos administrativos, ou não está configurada. Imagine que existem 20 interfaces seriais mas somente uma é realmente usada. Se você estiver procurando octetos de entrada e saída das interfaces seriais, estará fazendo 40 pollings e 38 delas sempre lerão o valor 0. O *xnmgraph* do OpenView permite especificar uma OID e uma expressão regular para selecionar o que deve ser grafado. Para utilizar esse recurso, vamos percorrer a MIB para conhecer as informações disponíveis:

```
$ snmpwalk orarouter1 .1.3.6.1.2.1.2.2.1.8
interfaces.ifTable.ifEntry.ifOperStatus.1 : INTEGER: up
interfaces.ifTable.ifEntry.ifOperStatus.2 : INTEGER: up
interfaces.ifTable.ifEntry.ifOperStatus.3 : INTEGER: down
```

Isso nos informa que existem apenas duas interfaces atualmente ativas. Ao examinar *ifDescr*, vemos que as interfaces em funcionamento são a *Ethernet0* e *Serial0*; a *Serial1* está paralisada. Observe que o tipo de *ifOperStatus* é INTEGER, mas o valor de retorno se assemelha a uma string. Como assim? A RFC 1213 define valores de string para cada valor de retorno possível:

```
ifOperStatus OBJECT-TYPE
  SYNTAX  INTEGER {
    up(1),      - ready to pass packets
    down(2),
    testing(3) - in some test mode
  }
  ACCESS  read-only
  STATUS   mandatory
  DESCRIPTION
    "The current operational state of the interface. The testing(3)
     state indicates that no operational packets can be passed."
 ::= { ifEntry 8 }
```

É muito fácil ler isso: o valor inteiro 1 é convertido para a string up. Por conseguinte, podemos utilizar o valor 1 em uma expressão regular que testa *ifOperStatus*. Para cada instância, verificaremos o valor de *ifOperStatus*; faremos uma polling nessa instância e grafaremos o resultado somente se o status retornar 1. No falso código, a operação ficaria assim:

```
if (ifOperStatus == 1) {  
    pollForMIBData;  
    graphOctets;  
}
```

Eis a próxima versão de nosso script de grafação. Para inserir essa lógica em um gráfico, usamos a OID para *ifOperStatus* como a quarta opção de dois-pontos, e a expressão regular (1) como a quinta opção:

```
#!/bin/sh  
# filename: /opt/OV/local/scripts/graphOctets  
# syntax: graphOctets <hostname>  
/opt/OV/bin/xnmgraph -c public \  
-title Octets_In_and_Out_For_All_Up_Interfaces \  
-mib ".1.3.6.1.2.1.2.2.1.10::::.1.3.6.1.2.1.2.2.1.8:1::::, \  
.1.3.6.1.2.1.2.2.1.16::::.1.3.6.1.2.1.2.2.1.8:1::::" $1
```

Esse comando grafa os *ifInOctets* e *ifOutOctets* de qualquer interface que possua um estado operacional igual a 1 ou up. Sendo assim, o comando faz uma polling e grafa apenas as portas importantes, economizando largura de banda da rede e simplificando o gráfico. Além disso, é menos provável que faltem cores ao criar o gráfico porque não as atribuiremos a objetos inúteis. Contudo, observe que essa seleção só acontece durante a primeira e permanece em vigor pela duração do processo gráfico. Se o status de qualquer interface mudar depois que o gráfico for iniciado, nada no gráfico será modificado. A única maneira de descobrir as mudanças ocorridas no status da interface é reiniciar o *xnmgraph*.

Por último, examinemos:

- Como adicionar um rótulo a cada uma das OIDs grafadas
- Como multiplicar cada valor por uma constante
- Como especificar o intervalo de polling

O gráfico recortado da Figura 9-6 mostra mostram a mudança dos rótulos quando executamos o seguinte script:

```
#!/bin/sh  
# filename: /opt/OV/local/scripts/graphOctets  
# syntax: graphOctets <hostname>  
/opt/OV/bin/xnmgraph -c public -title Internet_Traffic_In_K -poll 68 -mib \  
.1.3.6.1.4.1.9.2.2.1.1.6:Incoming_Traffic::::.1.3.6.1.2.1.2.2.1.2:::001:, \  
.1.3.6.1.4.1.9.2.2.1.1.8:Outgoing_Traffic::::.1.3.6.1.2.1.2.2.1.2:::001:" \  
$1
```

Os rótulos são fornecidos pelo segundo e sexto campos nas opções separadas por caracteres de dois-pontos (o segundo campo fornece um rótulo de texto que identifica os objetos que estamos grafando e o sexto usa o campo *ifDescr* para identificar a interface específica); o multiplicador de constante (.001) é dado pelo oitavo campo; e o intervalo de polling (em segundo) é especificado pela opção *-poll*.

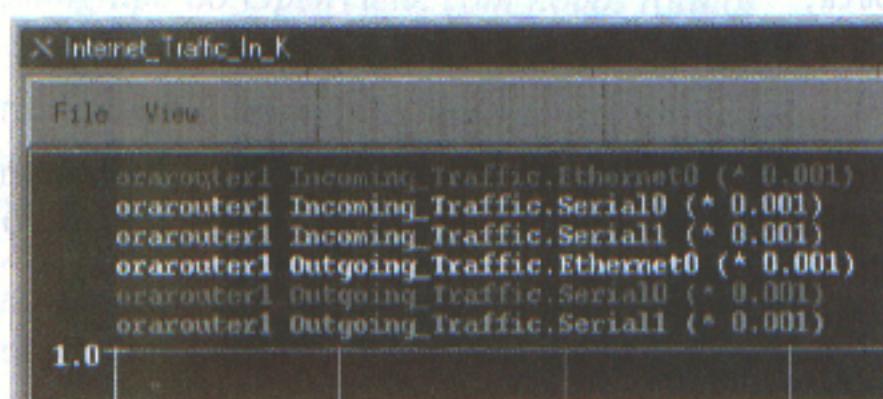


Figura 9-6 O *xnmgraph* com rótulos e multiplicadores

Por enquanto, é evidente a real flexibilidade do programa *xnmgraph* do OpenView. Esses gráficos podem ser ferramentas importantes para solucionar problemas na rede. Quando um gerente de rede receber reclamações dos clientes, relacionadas a conexões lentas, poderá examinar o gráfico de *ifInOctets* gerado pelo *xnmgraph* para detectar se alguma interface do roteador tem picos de tráfego alto.

Gráficos como esses também são úteis ao redefinir limiares para alarmes e outros tipos de traps. A última coisa que você deseja é um limiar que vive mudando ou que não mudará até que a construção inteira seja derrubada. Geralmente compensa examinar alguns gráficos para ter uma idéia do comportamento da rede antes de começar a definir quaisquer limiares. Esses gráficos oferecem uma base a partir da qual é possível trabalhar.

Por exemplo, vamos supor que você queira ser avisado quando a bateria do no-break estiver baixa (o que significa que está sendo usada) e quando voltar ao normal (totalmente carregada). O modo óbvio de implementar isso é gerar um alarme quando a bateria cair abaixo de uma porcentagem específica da carga total e outro alarme quando retornar à carga completa. Então, a pergunta é: que valor é possível definir para o limiar? Devemos usar 10% para indicar que a bateria está sendo usada e 100% para indicar que voltou ao normal? Podemos encontrar a linha de base, grafando as MIBs do dispositivo.\* Por exemplo, com um gráfico de 4 dias, podemos definir que a bateria do no-break permanece em 94-97% quando não está sendo usada. Em um período curto, a bateria caiu para 89%, quando estava executando um autoteste. A partir desses números, podemos definir o limiar “em uso” com 85% e o limiar “volta ao normal” com 94%.

\* Os diversos fornecedores possuem diferentes MIBs de no-break. Consulte a MIB do fornecedor específico para descobrir qual objeto representa baixa potência de bateria.

Esse par de limiares confirma quando a bateria está em uso mas não gerará alarmes inúteis quando o dispositivo estiver no modo autoteste. Os limiares adequados dependem do tipo de dispositivo sendo verificado, assim como dos dados da MIB obtidos. Fazer um teste inicial e uma polling para obter uma linha de base (números normais) ajudará a definir limiares significativos e úteis.

Antes de sair do *xnmgraph*, daremos uma última olhada no aspecto mais desagradável desse programa: a seqüência de nove opções separadas por caracteres de dois-pontos. Nos exemplos, demonstramos as combinações mais úteis de opções. Com mais detalhes, eis a sintaxe da especificação do gráfico:

*object:label:instances:match:expression:instance-label:truncator:multiplier:nodes*

Os parâmetros são:

#### *object*

A OID do objeto cujos valores devem ser grafados. Pode ser no formato numérico ou legível pelo ser humano, mas não deve ter um número de instância no final. Também pode ser o nome de uma expressão (as expressões estão discutidas no Apêndice A).

#### *label*

Uma string para cria o rótulo de todas as instâncias do objeto em questão. Pode ser uma string literal ou a OID de algum objeto com um valor de string. O rótulo usado no gráfico é criado por meio da combinação desse rótulo (para todas as instâncias do objeto) com *instance-label*, que identifica as instâncias individuais de um objeto em uma tabela. Por exemplo, na Figura 9-6, os rótulos são *Incoming\_Traffic* e *Outgoing\_Traffic*; *instance-label* é 1.3.6.1.2.1.2.2.1.2 ou o campo *ifDescr* para cada objeto sendo grafado.

#### *instances*

Uma expressão regular que especifica quais as instâncias do *object* serão grafadas. Se for omitida, serão grafadas todas as instâncias. Por exemplo, a expressão regular 1 limita o gráfico à instância 1 do *object*; a expressão regular [4-7] limita o gráfico às instâncias 4 a 7. Você pode utilizar os campos *match* e *expression* para especificar ainda mais os objetos a serem correspondidos.

#### *match*

A OID de um objeto (sem incluir a ID de instância) para ser comparada a uma expressão regular (a expressão de comparação), para determinar as instâncias do objeto que serão exibidas no gráfico.

#### *expression*

Uma expressão regular; para cada instância, o objeto fornecido por *match* é comparado a essa expressão regular. Se os dois corresponderem, a instância será grafada.

#### *instance-label*

Um rótulo a ser usado para identificar instâncias específicas do objeto sendo grafado; usado em combinação com os campos *label* e *truncator* para criar um rótulo para cada instância de cada objeto sendo grafado.

#### *truncator*

Uma string que será removida da parte inicial do rótulo da instância, para torná-lo mais curto.

#### *multiplier*

Um número usado para escalar os valores sendo grafados.

#### *nodes*

Os nós a serem verificados para criar o gráfico. Você pode listar qualquer número de nós, separados por espaços. O curinga “\*” verifica todos os nós contidos no banco de dados do OpenView. Se você omitir esse campo, o *xnmgraph* usará a lista de nós do último argumento na linha de comando.

O único campo obrigatório é *object*; entretanto, como vimos anteriormente, você deve ter os oitos caracteres de dois-pontos, mesmo que você deixe alguns (ou a maioria) dos campos vazios.

### *Obtenção de dados e limiares no OpenView*

Quando você fechar os gráficos do OpenView, os dados neles contidos desaparecerão definitivamente. O OpenView oferece uma maneira de corrigir esses problemas com a obtenção de dados. A coleta de dados permite que o usuário consulte e registre os dados continuamente. Além disso, também permite examinar esses resultados e disparar eventos. Uma vantagem da coleta de dados é a possibilidade de observar a rede para você, na sua ausência; você pode iniciar a coleta de dados na sexta-feira, e sair para o final de semana, sabendo que todos os eventos importantes serão registrados na sua ausência.

Você pode iniciar a função Data Collection and Thresholds do OpenView na linha de comando, usando o comando *\$OV\_BIN/xnmcollect*, ou no NNM, menu Options. É exibida a janela “Data Collection and Thresholds”, como mostra a Figura 9-7, que exibe uma lista das coletas configuradas e um resumo dos parâmetros de coleta.

As coletas configuradas, que estão no modo “Suspended”, são exibidas em uma fonte escura ou em negrito. Isso indica que o OpenView não está coletando quaisquer dados para esses objetos. Um status “Collecting” indica que o OpenView está fazendo uma polling nos nós selecionados para o objeto específico e salvando os dados. Para modificar o status de uma coleta, selecione o objeto, clique em “Actions” e, em seguida, clique em “Suspend Collection” ou “Resume Collection”. (Salve as alterações para que entrem em vigor.)

The screenshot shows the 'Data Collection' window in OpenView. At the top, there's a menu bar with 'File', 'Edit', 'Actions', 'Help'. Below it is a title 'MIB Objects Configured for Collection'. A table lists eight MIB objects with their status, label, and MIB object ID:

Status	Label	MIB Object ID
Suspended	ifInOctets	.1.3.6.1.2.1.2.2.1.10
Suspended	ifOutOctets	.1.3.6.1.2.1.2.2.1.16
Suspended	ifInErrors	.1.3.6.1.2.1.2.2.1.14
Suspended	ifOutErrors	.1.3.6.1.2.1.2.2.1.20
Suspended	15MinLoadAvg	.1.3.6.1.4.1.11.2.3.1.1.5
Suspended	snmpInPkts	.1.3.6.1.2.1.11.1
Suspended	IfZUtil	IfZUtil
Suspended	DiskZUtil	DiskZUtil

Below this is a section titled 'MIB Object Collection Summary' with a table:

Interval	Store	Threshold	Source	Instances

Figura 9-7 Janela Data Collection e Thresholds do OpenView

### Elaborando coletas

Para elaborar uma nova coleta, clique em “Edit → Add MIB Object”. Você acessará uma nova tela. Na parte superior dessa tela, clique em “MIB Object”\* e vá clicando na árvore até encontrar o objeto a ser verificado.

Para examinar o status da bandeja de papel de nossa impressora, por exemplo, precisamos navegar até `.iso.org.dod.internet.private.enterprises.hp.nm.system.net-peripheral.net-printer.generalDeviceStatus.gdStatusEntry.gdStatusPaperOut (.1.3.6.1.4.1.11.2.3.9.1.1.2.8)`. A descrição do objeto indica que se trata do item que procuramos: “This indicates that the peripheral is out of paper.” (*Isso indica que o periférico está sem papel.*) (Se você sabe o que procura, pode inserir o nome ou a OID diretamente.) Encontrado o local, você pode mudar o nome da coleta para outro mais fácil de ler. Clique em “OK” para avançar. É exibido o menu da Figura 9-8.

No campo “Source”, especifique os nós de origem dos dados. Insira os nomes dos host ou endereços IP a serem verificados. Você pode utilizar curingas como `198.27.6.**` nos endereços IP; também é possível clicar em “Add Map”

\* Você pode coletar o valor de uma expressão em vez de um objeto de MIB individual. O tema “expressões” está além do âmbito deste livro mas você encontrará uma explicação na página do manual de `mibExpr.conf(4)`.

\*\* Este objeto se encontra na MIB privada da HP, de modo que não estará disponível, a não ser que você possua impressoras HP e tenha instalado as MIBs pertinentes. Observe que existe uma MIB de impressoras padrão, a RFC 1759, mas a MIB da HP tem informações mais úteis.

para adicionar quaisquer nós atualmente selecionados. Sugerimos que você comece com um único nó, para fins de teste.

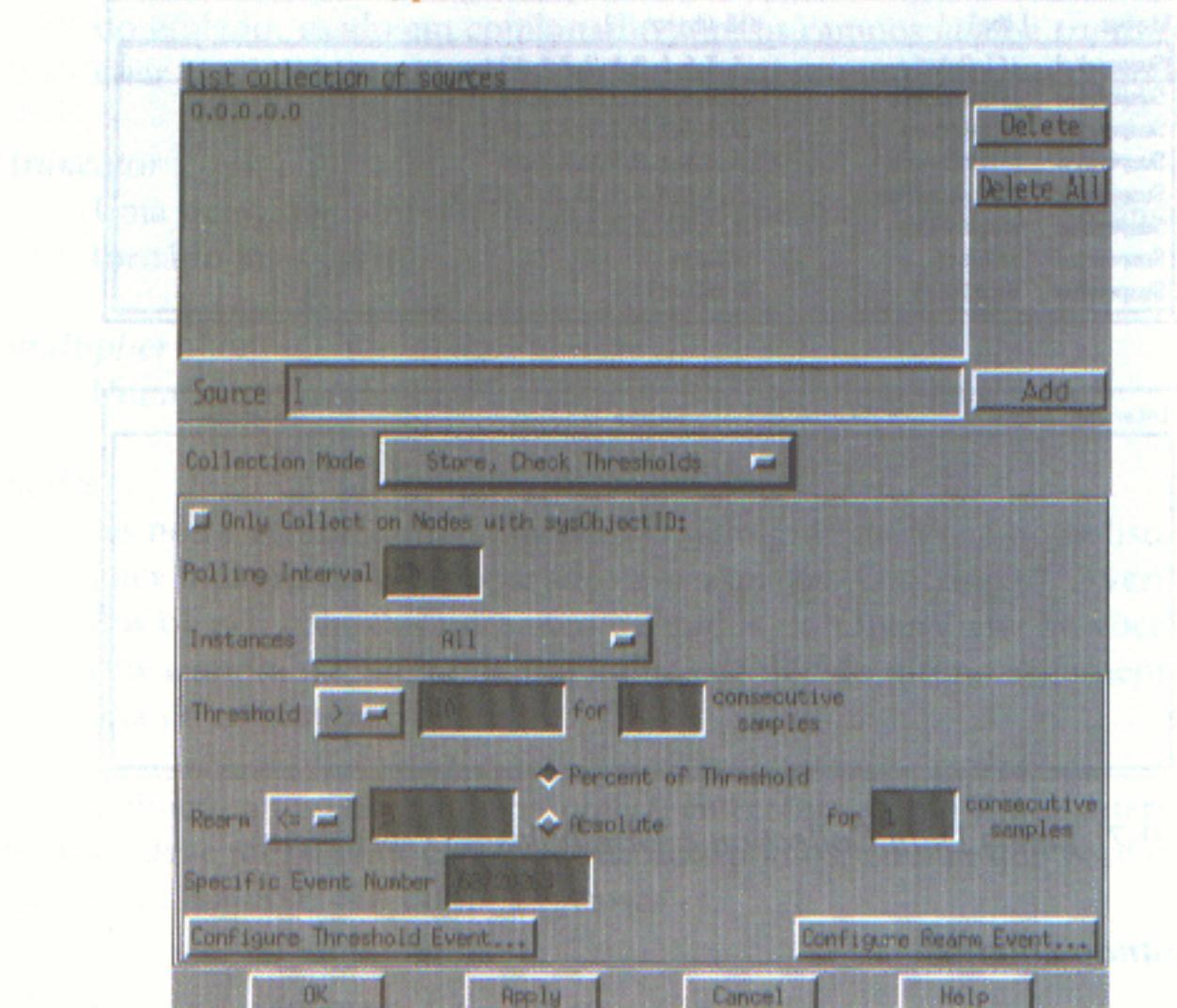


Figura 9-8 Menu de configuração de polling do OpenView

É fácil adicionar mais nós a uma coleção se tudo estiver configurado corretamente; basta retornar à janela da Figura 9-8 e incluir os nós na lista Source.

O “Collection Mode” permite especificar o que fazer com os dados coletados pelo NNM. Há quatro modos de coleta: “Exclude Collection”, “Store, Check Thresholds”, “Store, No Thresholds” e “Don’t Store, Check Thresholds”. Exceto o modo “Exclude Collection”, que permite desativar coletas individuais para cada dispositivo, os modos de coleta são basta auto-explicativos. (“Exclude Collection” pode parecer estranho mas é muito útil para excluir alguns dispositivos da coleta, sem interromper o processo inteiro; por exemplo, pode existir um roteador com um problema no hardware, que está bombardeando o sistema com dados sem sentido.) A coleta de dados sem um limiar é mais fácil do que aquela com um limiar, de modo que iniciaremos aí. Defina o Collection Mode com “Store, No Thresholds”. Esse modo desativa (esmaece) a parte inferior do menu, usada para os parâmetros de limiar. (Selecione “Store, Check Thresholds” se você deseja coleta de dados e monitoração de limiares.) Em seguida, clique em “OK” e salve a nova coleta. A partir de então, você poderá observar sua coleta crescer no diretório \$OV\_DB/snmpCollect. Cada coleta consiste em um arquivo de dados binários, mais um arquivo de mesmo nome mas pre-

cedido por um ponto de exclamação (!); esse arquivo armazena as informações da coleta. Os arquivos de coleta de dados crescem descomodidamente. Para eliminar esses arquivos sem prejudicar o coletor, exclua todos aqueles que não contêm um ponto de exclamação ("!").

Clicar em "Only Collect on Nodes with sysObjectID:" permite inserir um valor para *sysObjectID*. *sysObjectID* (*iso.org.dod.internet.mgmt.mib-2.system.sys-ObjectID*) permite limitar a polling aos dispositivos de um fabricante específico. Seu valor é o número da empresa registrado pelo fabricante do dispositivo junto ao IANA. Por exemplo, o número de empresa da Cisco é 9 e o da HP é 11 (existe uma lista completa disponível em <http://www.isi.edu/in-notes/iana/assignments/enterprise-numbers>); portanto, para restringir a polling aos dispositivos fabricados pela HP, defina a *sysObjectID* com 11. A RFC 1213 define oficialmente a *sysObjectID* (1.3.6.1.2.1.1.2) assim:

```
sysObjectID OBJECT-TYPE
  SYNTAX OBJECT IDENTIFIER
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "The vendor's authoritative identification of the network
     management subsystem contained in the entity. This value
     is allocated within the SMI enterprises subtree (1.3.6.1.4.1)
     and provides an easy and unambiguous means for determining
     what kind of box' is being managed. For example, if vendor
     'Flintstones, Inc.' was assigned the subtree 1.3.6.1.4.1.4242,
     it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its
     'Fred Router'."
 ::= { system 2 }
```

O intervalo de polling é o período em que a polling ocorre. Você pode utilizar abreviações de uma letra para especificar unidades: "s" de segundos, "m" de minutos, "h" de horas, "d" de dias. Por exemplo, 32s indica 32 segundos; 1.5d indica 1 dia e meio. Ao elaborar uma coleta de dados, geralmente começo com um intervalo de polling muito curto – em geral, 7s (7 segundos entre cada polling). Provavelmente, na prática, você não usaria um intervalo de polling assim tão curto (todos os dados que você coletar devem ser armazenados em algum lugar), mas quando você estiver configurando uma coleta, geralmente compensa utilizar um intervalo de polling curto. Você não deseja aguardar muito tempo para descobrir se está coletando os dados corretos.

A opção seguinte é um menu suspenso que especifica as instâncias que devem ser consultadas. As opções são "All", "From List" e "From Regular Expression". Nesse caso, estamos fazendo uma polling de um item escalar, de modo que não precisamos nos preocupar com as instâncias; podemos deixar a definição com "All" ou selecionar "From List" e especificar a instância "0" (o número de instância para todos os objetos escalares). Se você estiver fazendo uma polling de um objeto tabular, poderá especificar uma lista de instâncias separadas por vírgulas ou selecionar a opção "From Regular Expression" e escrever uma

expressão regular que selecione as instâncias necessárias. Salve as alterações (“File → Save”) e pronto!

## Criando um limiar

Depois disso tudo, você configurou o NNM para obter periodicamente o status da bandeja de papel de sua impressora. Agora, faremos algo mais interessante: usemos os limiares para gerar um tipo de notificação quando o tráfego procedente de uma das interfaces da rede ultrapassar um nível determinado. Para isso, examinaremos um objeto específico da Cisco, *locIfInBitsSec* (mais exatamente, *iso.org.dod.internet.private.enterprises.cisco.local.interfaces.lifTable.lifEntry.locIfInBitsSec*), cujo valor é a média de cinco minutos da velocidade em que os dados alcançam a interface, em bits por segundo. (Existe um objeto correspondente chamado *locIfOutBitsSec*, que avalia os dados saindo da interface.) A primeira parte do processo deve ser conhecida: inicia a função Data Collection and Thresholds, no menu Options do NNM; depois clique em “Edit → Add MIB Object”. Navegue pela árvore de objetos até alcançar *locIfInBitsSec*; clique em “OK” para retornar à tela mostrada na Figura 9-8. Especifique os endereços IP das interfaces a serem monitoradas e defina o modo de coleta com “Store, Check Thresholds”; esse procedimento permite recuperar e visualizar os dados posteriormente. (Geralmente, eu ativo a função “Store” para verificar se o coleto está realmente funcionando e visualizar os dados acumulados.) Escolha um intervalo de polling razoável – mais uma vez, ao testar, compensa utilizar um intervalo mais curto – em seguida, selecione as instâncias a serem verificadas, após o quê você poderá definir os limiares.

O campo “Threshold” permite especificar o ponto em que o valor sendo monitorado torna-se interessante. O significado da palavra “interessante” fica a seu critério. Neste caso, vamos supor que estejamos monitorando uma conexão T1, com capacidade de 1.544 Mbits/segundo.

Especificaremos um pouco arbitrariamente que começaremos a nos preocupar quando o tráfego de entrada exceder 75% da capacidade. Sendo assim, após multiplicar, definimos o limiar com “> 1158000”. Evidentemente, o tráfego da rede é basicamente explosivo, de modo que não nos preocuparemos com um pico isolado – mas se em duas ou três leituras consecutivas, o limiar for ultrapassado, precisaremos ser avisado. Portanto, vamos definir “consecutive samples” com 3: isso evita o recebimento de notificações indesejadas e ainda fornece notificação abrangente se acontecer algo errado.

Definir um valor adequado de “consecutive samples” (amostras consecutivas) tornará sua vida muito mais agradável, embora a escolha de um valor correto seja praticamente uma arte. Outro exemplo é monitorar a partição */tmp* de um sistema Unix. Nesse caso, convém definir o limiar com “>= 85”, o número de amostras consecutivas com 2 e o intervalo de polling com 5m. Isso gerará um evento quando a utilização em */tmp* exceder 85% em duas pollings consecutivas. Essa escolha nas definições significa que você não receberá falsos alarmes se um usuário copiar um arquivo grande para o */tmp* e excluir o arquivo alguns minutos depois. Se você definir as amostras consecutivas com 1, o NNM gerará um evento

Threshold assim que detectar que o */tmp* está totalmente cheio, mesmo se a condição for apenas temporária e não ocorra nada com que se preocupar. Em seguida, gerará um evento Rearm depois que o usuário eliminar o arquivo. Na realidade, como só estamos preocupados com o fato de o */tmp* ficar totalmente preenchido e continuar cheio, definir o limiar de consecutivo com 2 pode reduzir o número de alarmes falsos. Geralmente, esse é um valor inicial eficiente para as amostras consecutivas, a não ser que o intervalo de polling seja muito alto.

Os parâmetros de rearmação permitem especificar quando tudo voltar ao normal ou, no mínimo, começar a voltar ao normal. Esse estado deve ocorrer antes que outro limiar seja atingido. É possível especificar um valor absoluto ou uma porcentagem. Ao monitorar os pacotes chegando a uma interface, convém definir o limiar de rearmação com um valor como 926.400 bits por segundo (um valor absoluto que representa 60% da capacidade total) ou 80% do limiar (também 60% da capacidade). De modo semelhante, se você estiver gerando um alarme quando o */tmp* exceder 85% da capacidade, seria conveniente rearmar quando o espaço livre retornar a 80% de nosso limiar de 85% (68% da capacidade). Você também pode especificar o número de amostras consecutivas que devem estar abaixo do ponto de rearmação para que o NNM considere a condição de rearmação atendida.

A última opção, “Configure Threshold Event”, pergunta os eventos do OpenView que você gostaria de executar para cada estado. Você pode deixar o evento default ou pode consultar o Capítulo 10 para obter mais informações sobre configurar eventos. O estado de “Threshold” precisa de um número de evento específico que deve residir na empresa HP. O evento Threshold default é *OV\_DataCollectThresh - 58720263*. Observe que o evento Threshold é sempre um número ímpar. O evento Rearm é o número seguinte depois do evento Threshold: neste caso, *58720264*.

Para configurar eventos diferentes do default, clique em “Configure Threshold Event” e, quando o novo menu for exibido, adicione um evento (com um número ímpar) à seção HP e um segundo evento para o respectivo Rearm. Após as inclusões, salve e retorne às janelas Collection para inserir o novo número.

Quando você terminar de configurar a coleta de dados, clique em “OK.” Você retornará ao menu Data Collection and Thresholds. Clique em “File → Save” para ativar as inclusões efetuadas. Na metade inferior da janela “MIB Object Collection Summary”, clique no novo objeto e, em seguida, em “Actions → Test SNMP”. É exibida uma janela mostrando os resultados de um teste do SNMP feito nessa coleta. Após o teste, espere um tempo suficiente para que o intervalo de polling expire uma ou duas vezes. Em seguida, clique no objeto collection novamente, mas dessa vez clique em “Actions → Show Data”. Essa janela mostra os dados obtidos até então. Tente lançar os dados na interface para ver se consegue disparar um evento Threshold. Se os eventos Threshold não estiverem ocorrendo, verifique se o limiar e o intervalo de polling estão definidos corretamente. Depois que você confirmar a ocorrência de um evento Threshold, observe como o evento Rearm é executado. Quando você terminar de testar, volte e configure períodos de polling realísticos, inclua quaisquer nós adicionais que você

gostaria de consultar e desative o armazenamento se você não quiser coletar dados para análise de tendências. Consulte o arquivo `$OV_LOG/snmpCol.trace` se enfrentar problemas na rolagem de sua coleta de dados. O manual do OpenView da HP deve descrever o uso desse arquivo de rastreamento para solucionar a maioria dos problemas.

Após coletar alguns dados, você pode utilizar o `xnmgraph` para exibi-los. O comando `xnmgraph` a ser utilizado é semelhante aos que vimos anteriormente; é um comando desajeitado que você desejará salvar em um script. No script a seguir, a opção `-browse` indica o grafador nos dados armazenados:

```
#!/bin/sh
# filename: /opt/OV/local/scripts/graphSavedData
# syntax: graphSavedData <hostname>
/opt/OV/bin/xnmgraph -c public -title Bits_In_n_Out_For_All_Up_Interfaces \
-browse -mib \
".1.3.6.1.4.1.9.2.2.1.1.6::::1.3.6.1.2.1.2.2.1.8:1:.1.3.6.1.2.1.2.2.1.2::::, \
.1.3.6.1.4.1.9.2.2.1.1.8::::1.3.6.1.2.1.2.2.1.8:1:.1.3.6.1.2.1.2.2.1.2::::" \
$1
```

Quando o gráfico for iniciado, nenhum dado real será grafado; a exibição está limitada aos dados coletados. Você pode clicar em “File → Update Data” para procurar e inserir quaisquer dados obtidos desde o início do gráfico. Outra opção é ignorar `-browse`, o que permite que o gráfico continue coletando e exibindo dados reais juntamente com os dados coletados.

Finalmente, para grafar todos os dados coletados para um nó específico, vá para o NNM e selecione o nó a ser investigado. Em seguida, selecione “Performance → Graph SNMP Data → Select Nodes” nos menus. Você obterá um gráfico de todos os dados coletados para o nó selecionado. Como alternativa, selecione a opção “All” em “Performance → Graph SNMP Data”. Com o número de cores limitado a 25, em geral, você descobrirá que não pode encaixar tudo em um único gráfico.

## SNMPc da Castle Rock

A edição *workgroup* do programa SNMPc da Castle Rock tem recursos semelhantes aos do pacote do OpenView. Usa a expressão “trend reporting” (relatório de tendências) para os recursos de coleta de dados e atribuição de limiares. A edição *enterprise* do SNMPc permite até exportar dados para uma página da Web. Em todos os nossos exemplos, usamos a edição *workgroup* do SNMPc.

Para constatar o funcionamento do SNMPc, grafemos o objeto `snmpOutPkts`. A OID desse objeto é `1.3.6.1.2.1.11.2` (`iso.org.dod.internet.mgmt.mib-2.snmp.snmpOutPkts`) e ele está definido na RFC 1213 como segue:

```
snmpOutPkts OBJECT-TYPE
  SYNTAX Counter
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
```

"The total number of SNMP messages which were passed from the SNMP protocol entity to the transport service."

::= { snmp 2 }

Usaremos o dispositivo *orahub* neste exemplo. Comece clicando na guia MIB Database selection mostrada na Figura 9-9; essa guia se encontra na parte inferior da tela e é parecida com uma planilha eletrônica – é a segunda a partir da esquerda. Clique na árvore até alcançar *iso.org.dod.internet.mgmt.mib-2.snmp*. Clique no objeto a ser grafado (neste exemplo, o *snmpOutPkts*). É possível selecionar vários objetos com a tecla Ctrl.

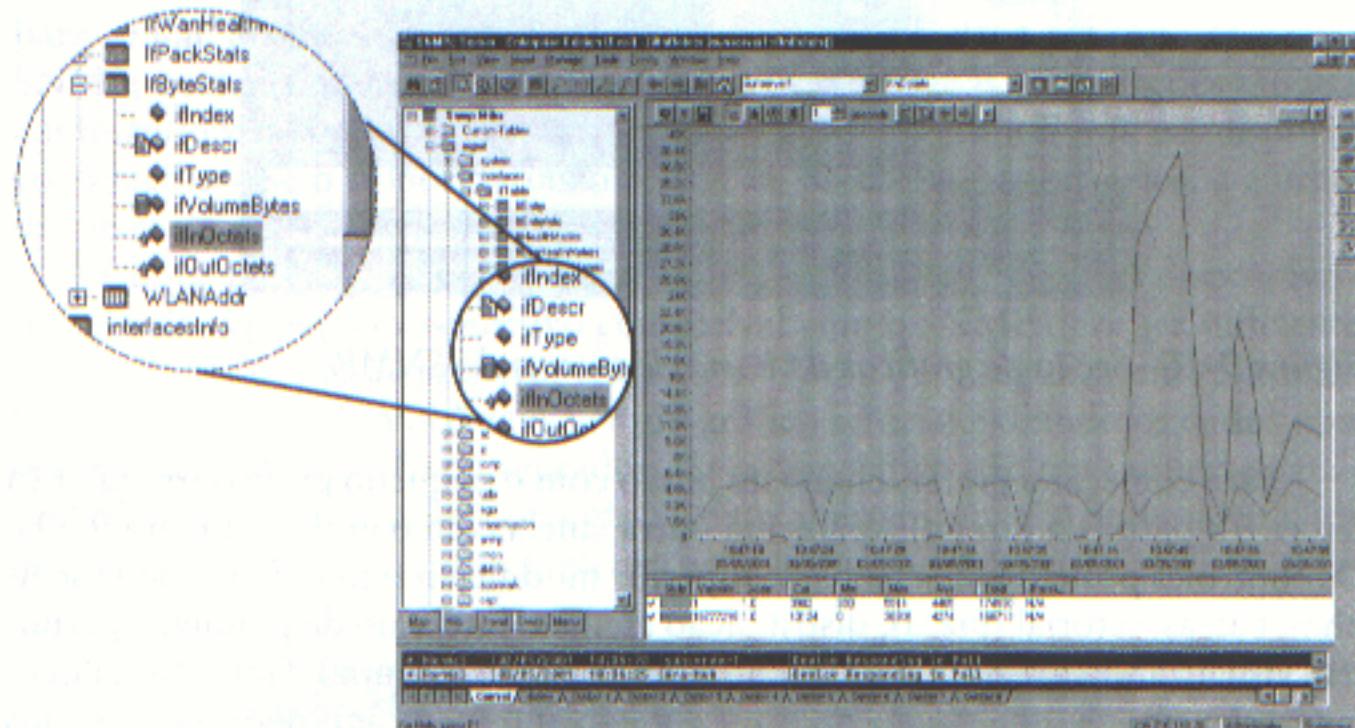
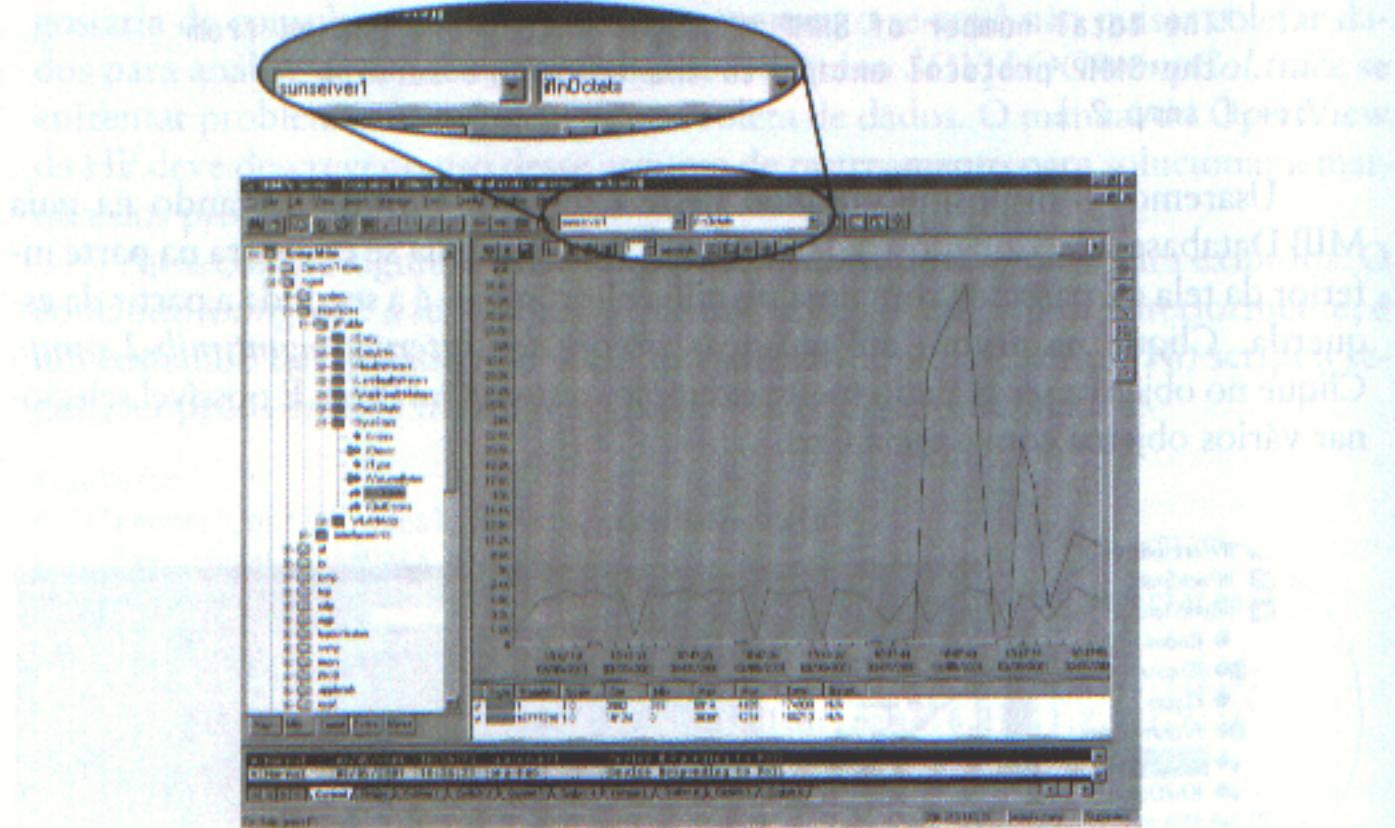


Figura 9-9 Visão MIB Database do SNMPc



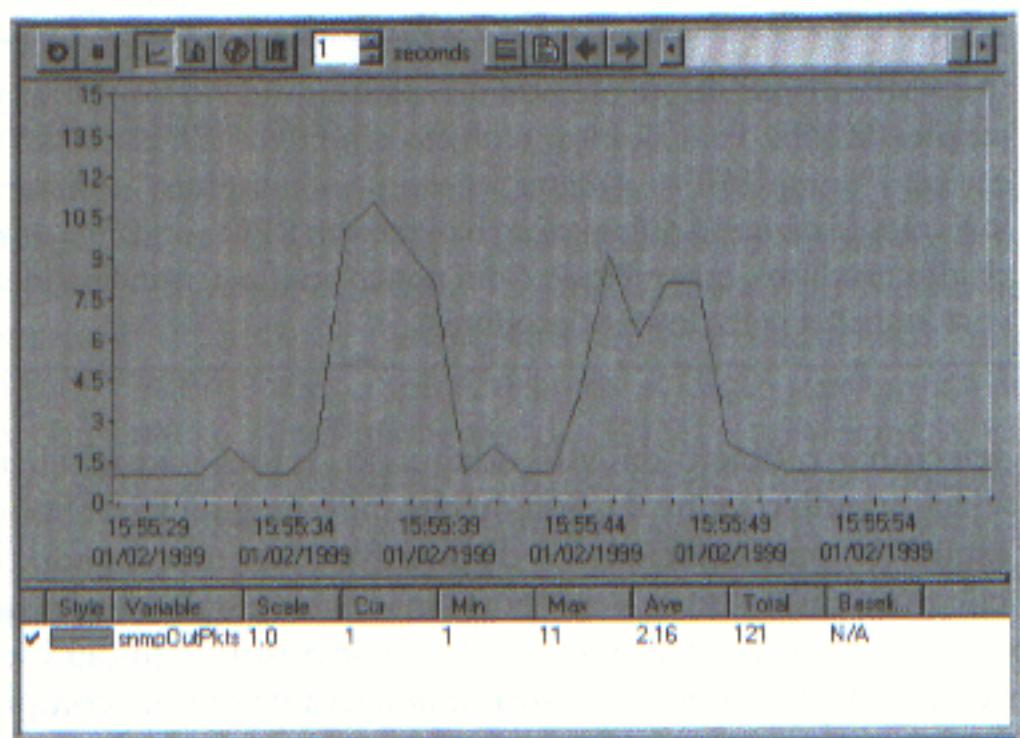
O SNMPc dispõe de um método não padronizado de organizar informações de MIB. Para alcançar o objeto *snmpOutPkts*, é necessário clicar em: "Snmp MIBs → mgmt → snmp → snmpInfo". Embora seja mais rápido do que a organização baseada em RFCs, utilizada na maioria dos produtos, esse método é um pouco confuso, principalmente se você trabalha com diversos produtos.

Após selecionar o objeto da MIB adequado, retorne ao nível superior de seu mapa, selecionando o ícone da casa ou clicando na guia Root Subnet (na extremidade esquerda) para selecionar o dispositivo a ser investigado. Em vez de localizar e clicar no dispositivo, você pode inserir manualmente o nome desse dispositivo. Se você investigou anteriormente o dispositivo, poderá selecioná-lo na caixa suspensa. A Figura 9-10 mostra uma barra de menus completa.



*Figura 9-10 Seção de gráficos da barra de menus do SNMPc*

Para começar a grafar, clique no botão com o pequeno gráfico irregular (o terceiro a partir da direita). Aparecerá outra janela com o gráfico (Figura 9-11). Os controles posicionados na parte superior modificam o tipo do gráfico (de linhas, barras, setorial (pizza), distribuição etc.) e o intervalo de polling, e permitem visualizar dados do histórico (barra deslizante horizontal). Consulte a documentação ou, melhor ainda, pratique diretamente para aprender esses menus mais rapidamente.



*Figura 9-11 Seção de gráfico do snmpOutPkts do SNMPc*

Quando você tiver uma coleção dos gráficos mais utilizados, poderá inseri-los nos menus personalizados. Vamos inserir um item de menu no menu Tools, que exiba todas as informações contidas na tabela *snmpInfo* como um gráfico setorial (de pizza). Clique na guia Custom Menus (a última), clique com o botão direito do mouse na pasta Tools e depois clique com o botão esquerdo no menu “Insert Menu”. Você acessará a janela “Add Custom Menu” (Figura 9-12). Digite um nome de menu e selecione “Pie” para o tipo de exibição. Use o botão browse (>>) para navegar na árvore de objetos da MIB até acessar a tabela *snmpInfo*; em seguida, clique em “OK”. Voltando a “Add Custom Menu”, use as caixas de seleção da seção “Use Selected Object” para especificar os tipos de nós que poderão responder a esse item de menu personalizado. Por exemplo, para grafar a *snmpInfo*, obviamente um dispositivo deve ter suporte para o SNMP; portanto, marcamos a caixa “Has SNMP”. Essas informações são usadas quando você (ou outro usuários) tentar gerar esse gráfico para um dispositivo específico. Se o dispositivo não aceitar os protocolos necessários, a entrada de menu do gráfico setorial será desativada.

Clique em “OK” e alterne para seu mapa para encontrar um dispositivo a ser testado. Qualquer dispositivo compatível com o SNMP deve ser suficiente. Após selecionar um dispositivo, clique em “Tools”, em seguida, em “Show Pie Chart of *snmpInfo*”. Você deve ver um gráfico setorial exibindo os dados obtidos dos objetos da MIB configurados. (Se o dispositivo não tiver suporte para o SNMP, essa opção estará desativada.) Como alternativa, você poderia ter clicado duas vezes no novo item de menu, na guia Custom Menu.

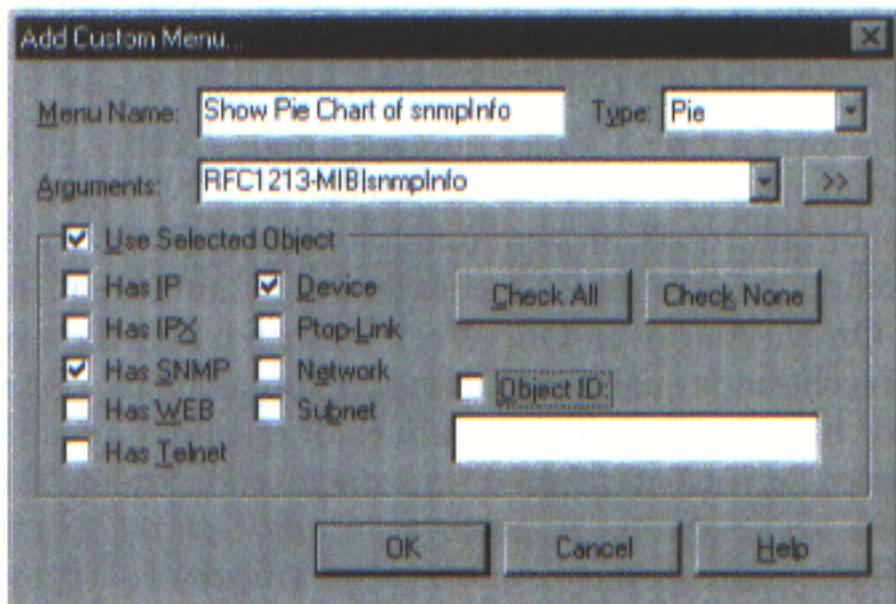


Figura 9-12 Janela Add Custom Menu do SNMPc

O SNMPc dispõe de um sistema de limiares denominado Automatic Alarms, que pode rastrear o valor de um objeto no decorrer do tempo, para determinar seus altos e baixos e conseguir uma linha de base. Após obter essa linha, o sistema alerta se algo ultrapassar os limiares. No menu principal, clicar em “Config → Trend Reports” abre o menu mostrado na Figura 9-13.

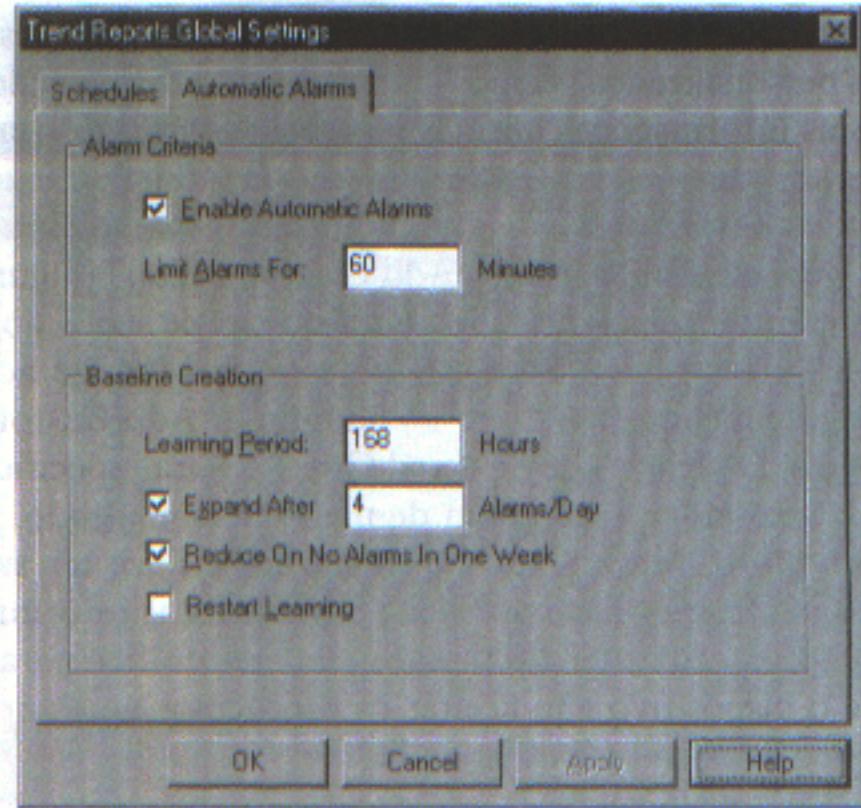


Figura 9-13 Menu Trend Reports Global Settings do SNMPc

Marque a caixa “Enable Automatic Alarms” para ativar esse recurso. A caixa “Limit Alarms For” permite especificar o tempo decorrido até que você receba outro alarme da mesma espécie.

Isso impede que o sistema seja inundado com a mesma mensagem várias vezes. A seção seguinte, “Baseline Creation”, permite configurar como a linha de base será assimilada. O período de aprendizagem é o tempo necessário ao SNMPc para calcular a linha de base. A opção “Expand After”, quando marcada, informa a quantidade de alarmes que você pode receber em um dia, antes que o SNMPc aumente os parâmetros da linha de base. Na Figura 9-13, se tivéssemos que receber quatro alarmes em um único dia, o SNMPc aumentaria o limiar para impedir a geração dessas mensagens com tanta freqüência. Marcar a caixa “Reduce On No Alarms In One Week” instrui o SNMPc a reduzir a linha de base se não recebermos quaisquer alarmes em uma semana. Essa opção impede que a linha de base seja definida com um valor tão alto que nunca recebemos quaisquer alarmes. Se você marcar a última opção e clicar em “OK”, o SNMPc reiniciará o processo de aprendizagem.

### Ferramentas de origem aberta para coleta e gravação de dados

Uma das ferramentas mais poderosas para a coleta e gravação de dados é o MRTG, conhecido de muitos usuários na comunidade de origens abertas. Essa ferramenta obtém dados estatísticos e gera relatórios gráficos na forma de páginas da web. Em alguns aspectos, é diferente das ferramentas discutidas nesse capítulo. Discutiremos o MRTG no Capítulo 13.

# 10

## Traps

As traps são um método para que um agente envie a uma estação de monitoração uma notificação assíncrona sobre as condições que o monitor deve conhecer. Aquelas que um agente pode gerar são definidas pelas MIBs por ele suportadas; o número de traps pode variar de zero a centenas. Para conhecer as traps definidas em um arquivo de MIB, procure a expressão “TRAP-TYPE” (SMIv1) ou “NOTIFICATION-TYPE” (SMIv2) no arquivo de MIB. Com essa pesquisa, você obterá rapidamente uma lista das possíveis traps.

Evidentemente, não é assim terrivelmente útil apenas direcionar traps assíncronas para sua NMS. Você pode configurar a resposta de sua NMS para diferentes traps; a resposta pode ser desde o descarte da trap até a execução de um script que envia uma mensagem para seu pager (ou até executa uma ação drástica, como desligar seu abastecimento de energia). Neste capítulo, ensinaremos a lidar com as traps recebidas, usando o OpenView e outras ferramentas como a linguagem Perl. Em seguida, discutiremos como ler e configurar os diversos aspectos dos eventos da trap. Finalmente, ensinaremos como definir as próprias traps que informam condições especiais de interesse particular para sua rede.

### Noções básicas sobre traps

Antes de abordar as ferramentas para receber e gerar traps, compensa revisar um que é uma trap. As traps foram apresentadas no Capítulo 2. Uma trap é basicamente uma notificação assíncrona enviada de um agente do SNMP para uma estação de gerenciamento de rede. Como tudo o mais no SNMP, as traps são enviadas por meio do UDP (*User Datagram Protocol* – porta 162) e, portanto, não são confiáveis. Isso quer dizer que o emissor não pode presumir que a trap realmente chegará, nem o destino pode pressupor que esteja recebendo todas as traps que lhe são enviadas. Naturalmente, em uma rede funcionando bem, a maioria das traps deve alcançar seus destinos. Mas se as redes estivessem sempre saudáveis, não precisaríamos do SNMP.

Em um contexto mais detalhado, uma trap é um pacote de dados definidos por uma MIB. As traps são classificadas em duas categorias: genéricas e específicas.

cas da empresa. Existem sete números de traps genéricas (0-6), definidas na Tabela 2-8, para as condições que variam desde reinicializações do sistema (*cold-Start*) e mudanças de estado de interfaces (*linkUp* e *linkDown*) até a trap genérica 6 (*enterpriseSpecific*). As traps específicas de empresa são as aberturas que tornam o mecanismo da trap tão poderoso. Quem tiver um número de empresa pode definir traps específicas de empresa para quaisquer condições cuja monitoração seja compensadora. Uma trap específica de empresa é identificada por dois fragmentos de informação: a ID de empresa da organização que definiu a trap e um número de trap específico atribuído por essa organização. O conceito de uma trap específica de empresa é extremamente flexível porque as próprias organizações podem subdividir suas empresas como desejarem. Por exemplo, se um número de empresa for 2789, a ID de empresa será .1.3.6.1.4.1.2789. Mas é possível subdividir ainda mais, ao definir traps com IDs de empresa como .1.3.6.1.4.1.2789.5000, .1.3.6.1.4.1.2789.5001 e assim por diante.

O fato de receber uma trap e, por conseguinte, conhecer seu número de trap genérica, ID e número de trap específica de empresa geralmente é tudo o que você precisa para diagnosticar um problema. Mas as traps também transmitem outras informações. No caso das traps genéricas 0-5, as informações específicas são predefinidas e incorporadas por hardware à NMS. Quando você recebe uma trap genérica, a NMS sabe interpretar as informações nela contidas e poderá exibi-las adequadamente, quer no momento da reinicialização ou da identificação da interface que acabou de mudar de estado. Ao contrário, as informações transmitidas por uma trap específica de empresa ficam totalmente a critério da pessoa que definiu a trap. Uma trap específica de empresa pode conter qualquer número de vinculação de variáveis ou pares de objeto-valor de MIB. Ao definir as próprias traps, você especificar as informações adequadas para serem transmitidas por essas traps. Os objetos contidos em uma trap podem ser objetos padrão de MIB, objetos específicos do fornecedor ou objetos criados por você. É comum definir objetos apenas com a finalidade de incluí-los em uma trap.

## *Traps do SNMPv2*

O SNMPv2 define traps de uma maneira um pouco diferente. Em uma MIB, as traps da Versão 1 são definidas como TRAP-TYPE, enquanto as da Versão 2 são definidas como NOTIFICATION-TYPE. O SNMPv2 também resiste ao conceito das traps genéricas – em vez disso, define várias traps específicas (isto é, notificações) em MIBs públicas. As traps do SNMPv3, discutidas resumidamente no Apêndice F, são simplesmente as traps da SNMPv2 com a inclusão dos recursos de autenticação e privacidade. A maioria das implementações do SNMP aceita apenas a Versão 1.

## *Recebendo traps*

Comecemos discutindo como lidar com as traps recebidas. O tratamento das traps recebidas fica por conta da NMS e algumas NMSs tão-somente exibem as

traps recebidas na saída padrão (*stdout*). Entretanto, geralmente um servidor da NMS pode reagir às traps do SNMP que receber.

Por exemplo, quando uma NMS recebe uma trap de *linkDown* de um roteador, poderia responder ao evento avisando à pessoa de contato, exibindo uma mensagem pop-up em um console de gerenciamento ou direcionando o evento para outra NMS. Esse procedimento é dinamizado em pacotes comerciais, mas também pode ser alcançado nos programas de origem aberta distribuídos gratuitamente.

## *OpenView da HP*

O OpenView usa três peças de software para receber e interpretar traps:

- ovtrapd (1M)
- xnmtrap
- xnmevents

O principal daemon de tratamentos de traps do OpenView é denominado *ovtrapd*. Esse programa detecta as traps geradas por dispositivos na rede e as direciona para o daemon Postmaster (*pmd*). Por sua vez, o *pmd* aciona o que o OpenView chama de evento. É possível configurar eventos para executar ações que variam desde enviar uma janela pop-up para usuários do NNM, direcionar o evento para outras NMSs ou simplesmente ignorar. O processo de configuração usa o *xnmtrap*, a GUI do Event Configurations. O programa *xnmevents* exibe os eventos que chegaram, classifica-os em categorias configuráveis pelo usuário.

O OpenView mantém um histórico de todas as traps recebidas; para recuperar esse histórico, use o comando `$OV_BIN/ovdumpevents`. As versões anteriores do OpenView mantinham um arquivo de log de eventos em `$OV_LOG/trapd.log`. Por default, esse arquivo é interrompido quando seu tamanho alcança 4 MB. A partir de então, é renomeado para *trapd.log.old* e um novo arquivo *trapd.log* é iniciado. Se você está enfrentando problemas com traps, ou porque você não sabe se elas estão alcançando a NMS ou porque sua NMS está sendo bombardeada com um número excessivo de eventos, poderá usar `tail -f` para observar o *trapd.log* e visualizar as traps à medida que chegam. (Você também pode utilizar o *ovdumpevents* para criar um novo arquivo.) Para conhecer outros detalhes do formato desse arquivo, consulte as páginas do manual do OpenView, relacionadas ao *trapd.conf* (4) e *ovdumpevents* (1M).

Talvez compense definir o que é exatamente um evento do OpenView. Imagine esse evento como um pequeno registro, semelhante a um registro de banco de dados. Esse registro define qual trap o OpenView deve procurar. Em seguida, ele define o tipo de ação (enviar um e-mail, uma mensagem de pager para alguém, etc.), se pertinente, a ser executada.

## Usando o Event Configurations do NNM

O OpenView usa um arquivo de definição interna para determinar como reagir a situações específicas. Esse arquivo de definição é mantido pelo programa *xnmtrap*. É possível iniciar o *xnmtrap* por meio do item de menu “Options → Event Configurations” (na GUI do NNM) ou emitindo o comando \$OV\_BIN/*xnmtrap*.

Na janela Enterprise Identification, role para baixo e clique no nome da empresa “OpenView .1.3.6.1.4.1.11.2.17.1”. Será exibida uma lista na janela Event Identification. Role novamente para baixo nessa lista até encontrar “OV\_Node\_Down”. Clique duas vezes nesse evento para carregar o Event Configurator (Figura 10-1).

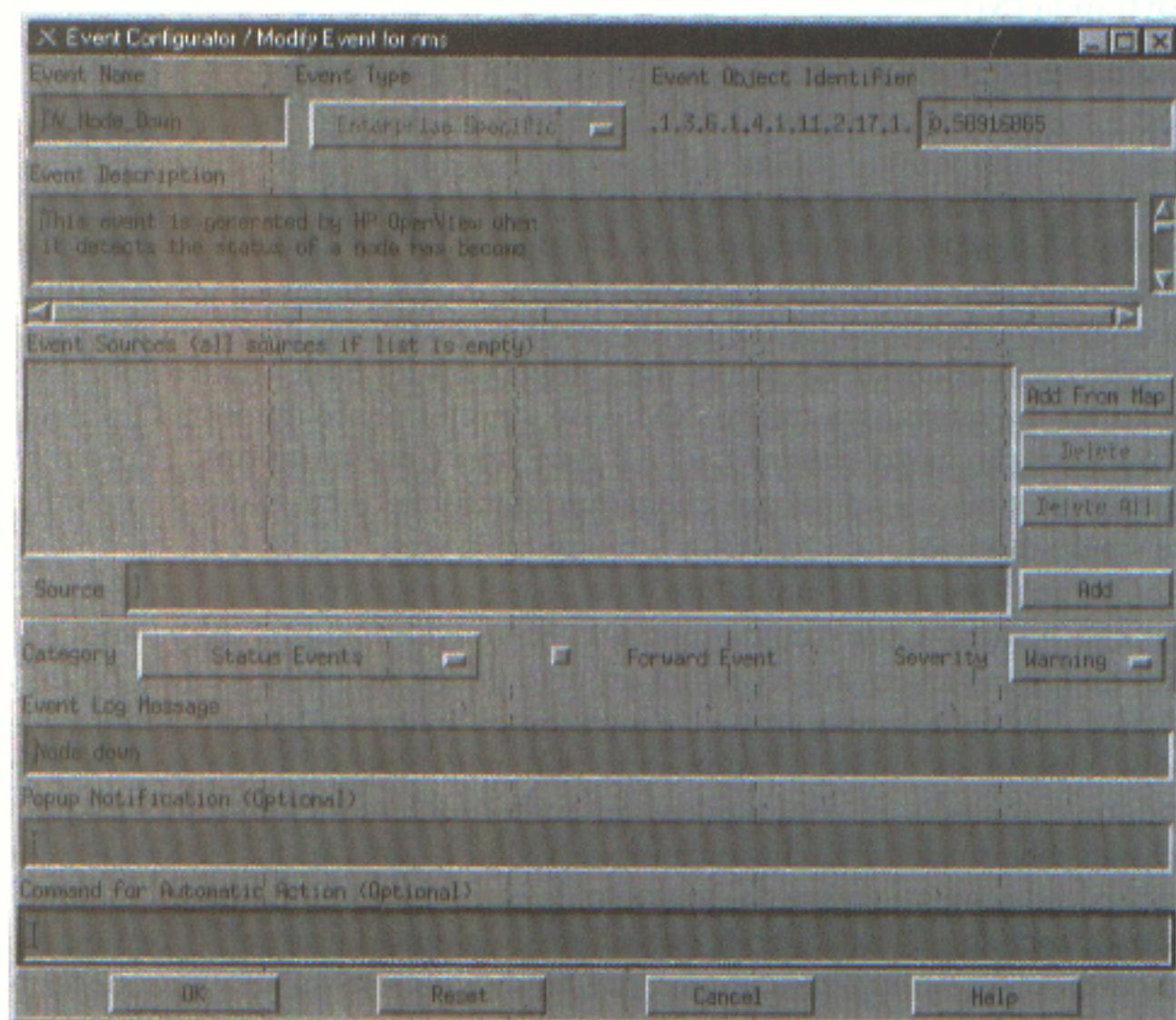


Figura 10-1 Event Configurator do OpenView–OV\_Node\_Down

A Figura 10-1 mostra o evento *OV\_Node\_Down* no Event Configurator. Quando esse evento é acionado, ele insere uma entrada contendo a mensagem “Node down” (Nó paralisado), com um nível de severidade “Warning”, na categoria Status Events. O OpenView gosta de ter um 0 (zero) final no Event Object Identifier (Identificador de objeto Evento), o que indica tratar-se de um evento ou de uma trap – e não há como modificar esse valor. O número anterior ao 0 é

a OID da empresa; o número posterior ao 0 é o número de trap específica, nesse caso 58916865.\* Mais adiante, usaremos esses números como parâmetros, ao gerar nossas próprias traps.

### *Selecionando origens de eventos*

A opção Source é útil para receber traps de determinados nós e ignorar as de outros. Por exemplo, se em seu ambiente existe um roteador de desenvolvimento, que as pessoas ligam e desligam o dia inteiro, é muito provável que você não receba todos os eventos gerados pela atividade do roteador. Nesse caso, você poderia usar o campo Source para listar todos os nós dos quais você gostaria de receber traps, e basta ignorar o roteador de desenvolvimento. Para isso, você pode digitar cada nome de host manualmente e clicar em “Add” após cada nome ou selecionar cada nó (com a seqüência de Ctrl + clique do mouse) em seu Network Node Map do OpenView e clicar em “Add From Map”. Infelizmente, a lista resultante não é fácil de gerenciar. Mesmo que você tenha tempo para adicionar todos os roteadores atuais às Event Sources (origens de eventos), ocasionalmente você adicionará um novo roteador (ou outro item de hardware que você deseja gerenciar). Então, você precisa voltar a todos os seus eventos e adicionar os novos dispositivos como origens. As versões mais recentes do OpenView permitem utilizar arquivos de correspondência de padrões e de origem, facilitando ainda mais a adaptação e manutenção da lista de origens.

### *Definindo categorias de evento*

Ao receber um evento, o NNM o classifica em uma categoria de evento. A caixa suspensa Categories permite atribuir o evento sendo configurado a uma categoria. Provavelmente, a lista de categorias disponíveis incluirá as seguintes categorias predefinidas (é possível personalizar essa lista, incluindo as categorias específicas para sua rede e eliminando outras, conforme examinaremos mais adiante nesta seção):

- Eventos Error
- Eventos Threshold
- Eventos Status
- Eventos Configuration
- Eventos Application alert
- Eventos Don't log or display
- Log only

Na realidade, as duas últimas categorias não são de eventos no verdadeiro sentido da palavra. Se você selecionar “Don't log or display”, o OpenView não salvará o evento no respectivo banco de dados nem exibirá a Event Log Message

\* Esse é o número default que o OpenView usa para essa trap OV\_Node\_Down.

(*mensagem de log de evento*) em quaisquer Event Categories (categorias de eventos). O OpenView exibirá a Popup Notification (*notificação pop-up*) em uma janela pop-up e executará o Command for Automatic Action (*comando para ação automática*). A opção “Log only” instrui o OpenView a não exibir o evento mas a manter um log do evento no banco de dados.\*



A opção “Log only” é útil para alguns eventos basicamente informativos; você não deseja vê-los ao chegarem mas, sim, registrá-los para referência futura. Um evento da Cisco, `frDLCIStatusChange - .1.3.6.1.2.1.10.32.0.1`, é um exemplo desse tipo de evento, que nos informa quando um Circuito Virtual mudou o respectivo estado operacional. Se exibido, veremos notificações sempre que um nó cair e um circuito mudar o estado operacional para paralisado. Essas informações são redundantes porque já recebemos um evento status de “node down” e uma mudança DLCI.\* Com esse evento definido como “Log only”, só devemos acessar o arquivo de log se desconfiarmos de algo suspeito.

### *Direcionando eventos e severidades de eventos*

Uma vez marcado, o botão de rádio “Forward Event” permite direcionar um evento para outras NMSs. Esse recurso é prático para diversas NMSs ou para uma arquitetura de gerenciamento de redes distribuídas. Vamos supor que você esteja em Atlanta, mas sua rede tenha uma estação de gerenciamento em Nova Iorque, adicionalmente àquela existente em sua mesa. Você não deseja receber todos os eventos de Nova York, mas, sim, a informação de `node_down`. Na NMS de Nova York, você poderia clicar em “Forward Event” e digitar o endereço IP de sua NMS de Atlanta. Quando a de Nova York receber um evento `node_down`, direcionará esse evento para Atlanta.

A lista suspensa Severity atribui um nível de severidade ao evento. O OpenView oferece suporte para seis níveis de severidade: Unknown, Normal, Warning, Minor, Major e Critical. Os níveis de severidade são codificados por cores para facilitar a identificação; A Tabela 10-1 indica a cor associada a cada nível de severidade. Os níveis estão listados em ordem de severidade crescente. Por exemplo, um evento com um nível de severidade “Minor” tem prioridade mais alta do que um outro com severidade “Warning”.

\* Mais uma vez, em versões anteriores do OpenView, esse log estava armazenado em `$OV_LOG/trapd.log`. As versões novas usam o Event Database (*banco de dados de eventos*) do OpenView, que é compatível com as versões anteriores, usando o comando `ovdumpevents` para gerar um arquivo `trapd.log`.

Tabela 10-1 Níveis de Severidade do OpenView

Severidade	Cor
Unknown	Azul
Normal	Verde
Warning	Cian
Minor	Amarelo
Major	Laranja
Critical	Vermelho

As cores são utilizadas tanto nos mapas do OpenView, quanto nas categorias de eventos (Event Categories). Os objetos pai, que representam o ponto inicial de uma rede, são exibidos na cor do nível de severidade mais alto associado a qualquer objeto abaixo deles.\* Por exemplo, se um objeto representar uma rede com 250 nós e um desses nós estiver paralisado (uma severidade Critical), o objeto será a cor vermelha, independentemente da quantidade de nós em operação e funcionando normalmente. A expressão que representa o modo como o OpenView exibe cores em relação aos objetos é *origem do status* e é explicada com mais detalhes no Capítulo 6.

### Mensagens de log, notificações e ações automáticas

Voltando à Figura 10-1, os campos Event Log Message e Popup Notification são semelhantes mas atendem a objetivos distintos. O campo Event Log Message é exibido quando você visualiza as Event Categories (categorias de eventos) e seleciona uma categoria na lista suspensa. O campo opcional Popup Notification exibe sua mensagem em uma janela que aparece em qualquer servidor executando o NNM do OpenView. A Figura 10-2 mostra uma mensagem pop-up habitual. Nesse caso, o nome do evento, *delme*, é exibido na barra de título. A hora e data da ocorrência do evento são seguidas pela mensagem do evento, “Popup Message Here” (*Mensagem pop-up aqui*). Para criar uma mensagem pop-up como essa, digite “Popup Message Here” na seção Popup Notification do Event Configurator. Sempre que o evento for chamado, será exibido um pop-up.

A última seção do Event Configurator é Command for Automatic Action. A ação automática permite especificar um comando do Unix ou script a ser executado quando o OpenView receber um evento. Você pode executar vários comandos, separando-os com um caractere de ponto-e-vírgula, praticamente como você faria em uma shell do Unix. Ao configurar uma ação automática, lembre-se de que o *rsh* pode ser muito útil. Gosto de utilizar o *rsh sunserver1 au-*

\* Versões mais recentes do OpenView possuem um recurso denominado Event Correlation (*correlação de eventos*) com informações redundantes. Você pode personalizar essas definições com um kit do desenvolvedor.

`dioplay -v50 /opt/local/sounds/siren.au`, que faz um arquivo de som de sirene tocar. A ação automática pode variar desde tocar um arquivo até abrir um *ticket* de problemas.



Figura 10-2 Mensagem pop-up do OpenView

Em cada Event Log Message, Popup Notification e Command for Automatic Action, as variáveis especiais podem ajudar a identificar os valores de traps ou eventos. Essas variáveis fornecem ao usuário informações adicionais sobre o evento. Examine a seguir algumas variáveis que você pode utilizar; a Ajuda on-line tem uma lista completa:

#### § 1

Imprima o primeiro atributo transmitido (por exemplo, o valor da primeira vinculação de variáveis) da trap.



Antes de começar a executar scripts para um evento, descubra o número médio de traps que você provavelmente receberá para o evento em questão. Isso vale principalmente para o OV\_Node\_Down. Se você escrevesse um script que abre um *ticket* de problemas sempre que um nó é paralisado, poderia terminar com centenas de *tickets* no final do dia. Monitorar sua rede o tornará terrivelmente consciente do quanto a rede “oscila” ou sobe e desce. Mesmo se a rede cair por um segundo, seja qual for o motivo, você receberá uma trap que, por sua vez, gera um evento que pode registrar um novo ticket, enviar uma página etc. A última ocorrência que você deseja é “A Rede que depreciei!” Você e os outros integrantes de sua equipe começarão a ignorar todos os avisos falsos e podem deixar passar problemas sérios ocorridos. Uma maneira de prever uma freqüência de recebimento de eventos é registrá-los em um arquivo (opção “Log only”). Depois de uma semana aproximadamente, verifique no arquivo de log quantos eventos se acumularam (isto é, o número de traps recebidas). Isso não é uma pesquisa científica, em hipótese alguma, mas você terá uma idéia do que esperar.

\$2

Imprima o segundo atributo transmitido.

\$n

Imprima o enésimo atributo como uma string de valores, que deve estar no intervalo de 1-99.

\$\*

Imprima todos os atributos como [seq] nome (tipo).

## Categorias de eventos personalizadas

O OpenView usa as categorias default para todos os respectivos eventos default. Examine no arquivo `$OV_CONF/C/trapd.conf` como os eventos default estão atribuídos às categorias. Você pode adicionar categorias, acessando “Event Configuration  $\Rightarrow$  Edit  $\Rightarrow$  Configure  $\Rightarrow$  Event Categories”. A Figura 10-3 mostra esse menu, com algumas categorias personalizadas incluídas.

Compensa investir algum tempo para analisar as categorias adequadas a seu ambiente. Se você juntar tudo nas categorias default, será incomodado pelo evento Crítico, “Printer Needs Paper” (Impressora sem papel), quando você realmente deseja ser informado sobre o evento Crítico “Production Server on Fire” (Incêndio no servidor da produção). Ambos os eventos “avermelharão” os Status Events.

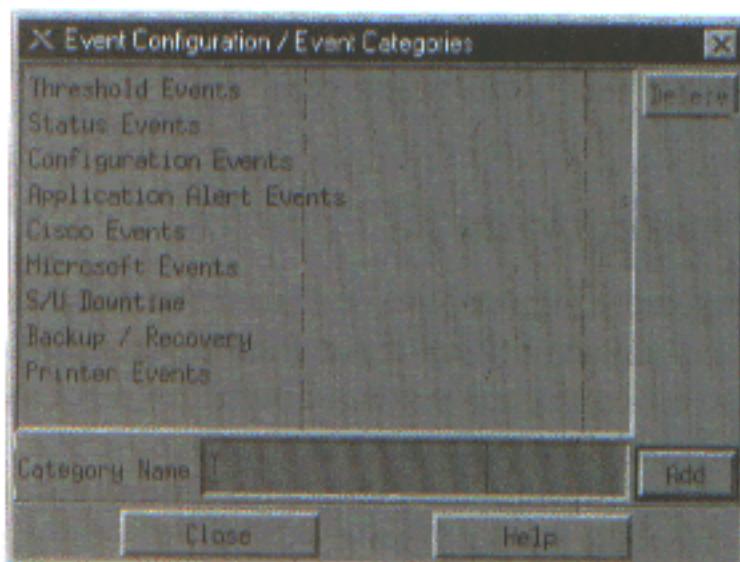


Figura 10-3 Adicionando categorias de eventos no OpenView

As categorias da Figura 10-3 são um começo eficiente mas refletia sobre os tipos de eventos e as atividades que serão úteis em sua rede. A categoria Scheduled and Unscheduled (S/U) Downtime é um bom exemplo de uma categoria que está mais para a intervenção do ser humano do que para informar erros da rede. Printer Events (Eventos da impressora) é um destino adequado para as mensagens “Printer Needs Paper” (Impressora sem papel) e “Printer Jammed” (Atolamento na impressora).

Embora nenhuma das categorias default seja obrigatória (exceto a categoria Error), recomendamos que você não as exclua, exatamente porque são utilizados para todos os eventos default. Eliminá-las sem reconfigurar primeiramente todos os eventos default ocasionará problemas. Todo evento que não possuir uma categoria de evento disponível será colocado na categoria default, Error. Para editar as categorias, copie o arquivo *trapd.conf* para */tmp* e modifique o */tmp/trapd.conf* com seu editor preferido. O arquivo tem alguns avisos enormes solicitando que você não o edite manualmente mas, às vezes, algumas edições simples são mais adequadas a atribuição de eventos. Uma entrada na parte do arquivo que define o comportamento de eventos é parecida com a seguinte:

```
EVENT RMON_Rise_Alarm .1.3.6.1.2.1.16.0.1 "Threshold Events" Warning
FORMAT RMON Rising Alarm: $2 exceeded threshold $5; value = $4.
(Sample type = \$3; alarm index = $1)
SDESC
This event is sent when an RMON device exceeds a preconfigured threshold.
EDESC
```

É óbvio o que essas linhas fazem: mapeiam um evento RMON específico na categoria Threshold Events, com um nível de severidade de Warning; além disso, especificam o que acontecerá quando o evento ocorrer. Para mapear esse evento em outra categoria, mude da categoria Threshold Events para outra adequada. Após editar o arquivo, use o seguinte comando para confirmar suas atualizações:

```
$ $OV_BIN/xnmevents -l load /tmp/trapd.conf
```

## *Exibição das categorias de eventos*

A janela Event Categories (Figura 10-4) é exibida na tela do usuário quando o NNM é iniciado. Essa janela apresenta um resumo sucinto do que está acontecendo na rede; se estiver configurada adequadamente, você saberá em um piscar de olhos se existem problemas preocupantes.

Se a janela se fechar durante uma sessão do OpenView, você poderá reiniiciá-la por meio do item de menu “Fault → Events” ou emitindo o comando *\$OV\_BIN/xnmevents*. O menu exibe todas as categorias de evento, inclusive as que você criou. Existem duas categorias especiais: a categoria Error é a default utilizada quando um evento é associado a uma categoria que não pode ser encontrada; a categoria All é um marcador para todos os eventos e não pode ser configurada pelo Event Configurator. A janela mostra o nível de severidade mais alto de todo evento em cada categoria de evento.

A caixa à esquerda de Status Events é cian (um azul claro), indicando que a severidade mais alta não confirmada na categoria Status Events é Warning. Clicar nessa caixa exibe um browser de alarme que lista todos os eventos recebidos na categoria. Um ótimo recurso da exibição das Event Categories é a possibili-

dade de restaurar o estado de um navegador ou recarregar eventos dos arquivos *trapd.log* e *trapd.log.old*. Recarregar eventos é útil se você precisar restaurar mensagens eliminadas anteriormente.

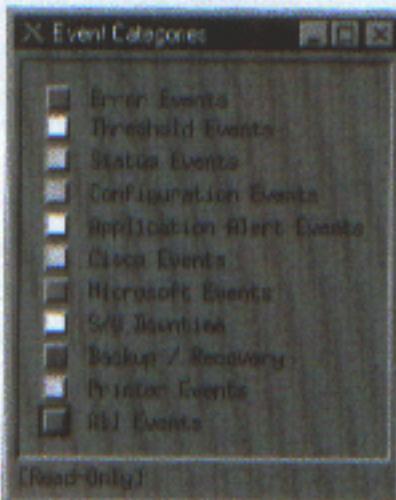


Figura 10-4 Event Categories do OpenView



As versões mais recentes do OpenView estendem as possibilidades das Event Categories, mantendo um banco de dados comum para eventos confirmados e não confirmados. Assim, quando um usuário confirmar um evento, todos os outros usuários verão esse evento atualizado.

Na parte inferior da Figura 10-4, a expressão “[Read-Only]” indica que você não tem acesso para gravação às Event Categories. Se essa expressão não aparecer, você terá acesso para gravação. O OpenView rastreia os eventos por usuário, usando um banco de dados especial, localizado em `$OV_LOG/xnmevents.username`.<sup>\*</sup> Com acesso para gravação, você pode atualizar esse arquivo sempre que sair do sistema. Por default, você tem acesso para gravação em seu próprio banco de dados de categorias de eventos, a não ser que alguém tenha iniciado uma sessão de trabalho no banco de dados, usando seu username. Deve existir somente um acesso para gravação em Event Categories por usuário, com o primeiro usuário obtendo esse acesso e todos os outros com privilégios somente leitura.

## Browser de alarme

A Figura 10-5 mostra o browser de alarme da categoria Status Events. Nessa categoria, vemos um único evento Warning, que ocasiona a exibição da categoria Status Events na cor cian.

\*Versões mais recentes do OpenView só têm um banco de dados comum a todos os usuários.

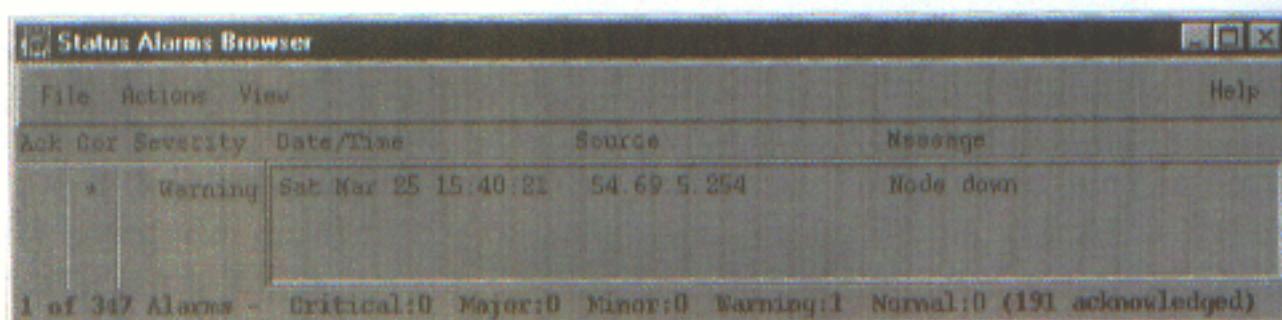


Figura 10-5 Browser de alarme do OpenView

A cor da caixa Status Events é determinada pelo evento de prioridade mais alta na categoria. Portanto, a cor não mudará até que você confirme o evento de prioridade mais alta ou chegue um evento com uma prioridade ainda mais alta. Clicar na coluna mais à esquerda (Ack) confirma a mensagem\* e define a severidade com 0.

O menu Actions no browser de alarme permite confirmar, desmarcar a confirmação ou eliminar alguns ou todos os eventos. Você pode até alterar a severidade de um evento. Lembre-se de que esse procedimento *não* muda a severidade do evento nas outras seções de Event Categories que estejam em execução. Por exemplo, Se um usuário alterar a severidade de um evento, de Critical para Normal, o evento continuará com a severidade Critical para os demais usuários. O menu View permite definir filtros, o que lhe permite incluir ou descartar as mensagens correspondentes ao filtro.

Ao configurar eventos, lembre-se de que você pode receber mais traps do que deseja. Quando isso acontece, você tem duas opções. Primeiramente, você pode acessar o agente e desativar a geração de traps, se o agente aceitar esse procedimento.

Em segundo lugar, você pode configurar a visão de traps de modo a ignorar essas traps. Já vimos como fazer isso: é possível definir o evento com "Log only" ou tentar excluir o dispositivo da lista Event Sources. Se a largura de banda for uma questão importante, investigue por que o agente está enviando tantas traps antes de tentar mascarar o problema.

## Criando eventos no OpenView

O OpenView permite criar outros eventos (privados). Os eventos privados são semelhantes aos eventos comuns, exceto pelo fato de que pertencem à sua subárvore de empresas privadas, e não a uma MIB pública. Para criar seus próprios eventos, abra a janela Event Configuration no menu Options do NNM. Você verá uma lista de todos os eventos atualmente carregados (Figura 10-6).

A janela está dividida em dois painéis. O painel superior exibe a Enterprise Identification (identificação da empresa), que é a parte posicionada mais à esquerda em uma OID. Clicar em uma ID de empresa exibe todos os eventos pertencentes a essa empresa, no painel inferior.

\* As versões mais atuais do OpenView têm suporte para o recurso Event Correlation, que também possui uma coluna nessa janela.

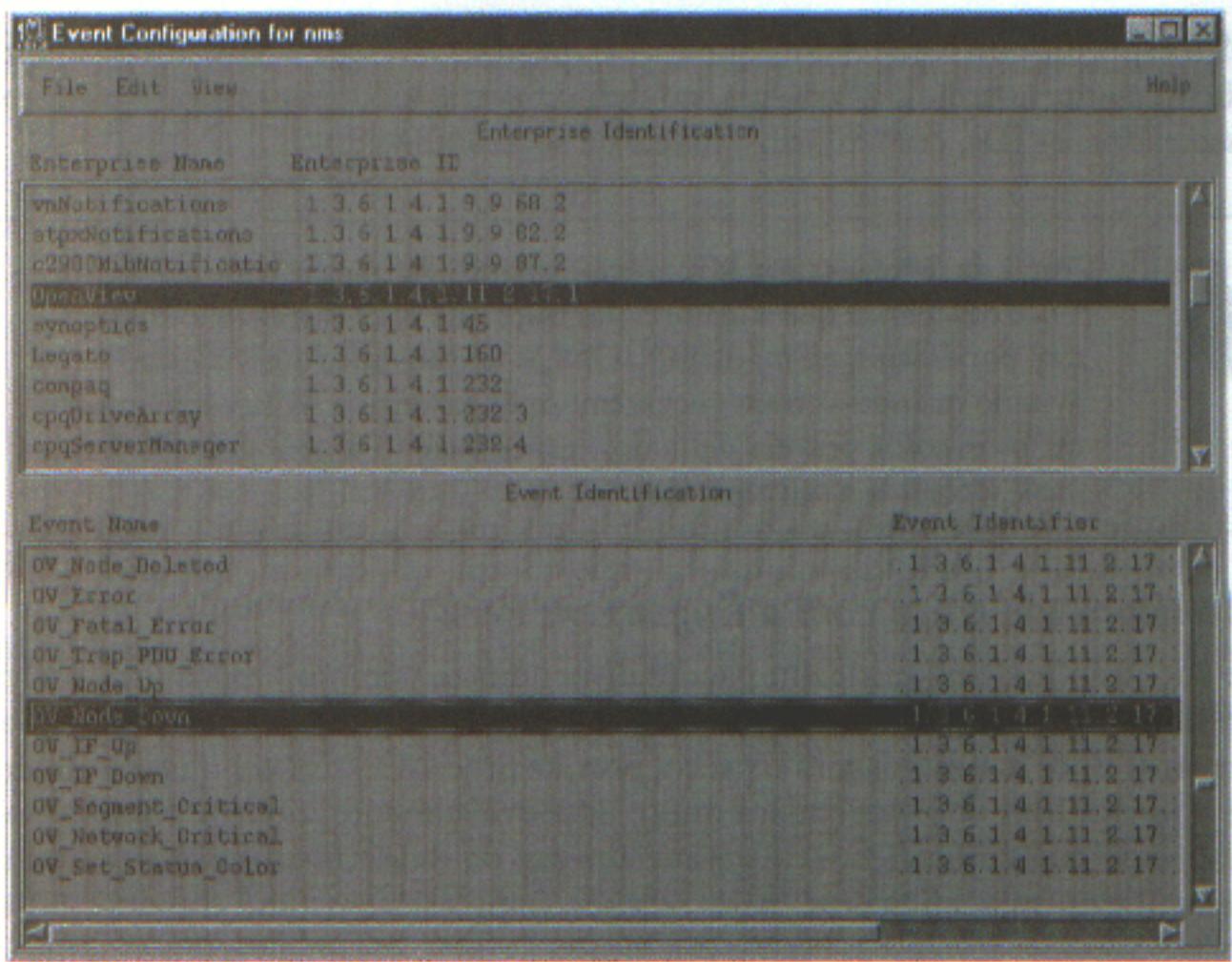


Figura 10-6 Event Configuration do OpenView

Para adicionar a ID de sua empresa, selecione “Edit → Add → Enterprise Identification” e digite o nome de sua empresa e uma ID de empresa registrada.\* A partir de agora, você pode criar eventos privados. Clique no nome da empresa que você acabou de criar; a ID de empresa que você associou a esse nome será usada para formar a OID para o novo evento. Clique em “Edit → Add → Event”; em seguida, digite o nome (Event Name) do novo evento, certificando-se de utilizar Enterprise Specific (o default) para o tipo de evento. Digite um Identificador de Objeto Evento (Event Object Identifier). Esse identificador pode ser qualquer número ainda não atribuído a um evento na empresa atualmente selecionada. Por último, clique em “OK” e salve a configuração do evento (usando “File → Save”).

Para copiar um evento existente, clique no evento a ser copiado e selecione “Edit → Copy Event”; será exibida uma nova janela com o evento selecionado. Desse ponto em diante, o processo é o mesmo.

As traps com “no format” são aquelas para as quais não existe qualquer definição na janela Event Configuration. Há duas maneiras de solucionar esse problema: você pode criar os eventos necessários por conta própria ou pode carregar uma MIB que contém as necessárias definições de traps, conforme discutido no Capítulo 6. Geralmente, as traps com “No format” são as definidas em uma

\* Para obter informações sobre como ter sua própria ID de empresa, leia o Capítulo 2.

MIB específica do fornecedor, que foi carregada. Carregar a MIB adequada frequentemente corrige o problema, ao definir as traps do fornecedor e os nomes associados, as IDs, comentários, níveis de severidade etc.



Antes de carregar uma MIB, revise os tipos de traps aceitos pela MIB. Você descobrirá que a maioria das traps que você carrega são fornecidas, por default, no modo LOGONLY. Isso significa que você não será avisado quando as traps chegarem. Após carregar a MIB, convém editar os eventos por ela definidos, especificando a configuração local mais adequada a seu ambiente.

## *Monitorando traps com a linguagem Perl*

Se você não pode adquirir um pacote dispendioso, como o OpenView, poderá utilizar a linguagem Perl para escrever seu próprio utilitário de monitoração e registro. Você mesmo pagará o preço, porque precisará escrever quase tudo desde o início. Mas você aprenderá muito e, provavelmente, poderá observar melhor os detalhes minuciosos do gerenciamento da rede. Um dos programas mais elementares, mas eficientes, para receber traps está em uma distribuição do SNMP Support for Perl 5, criado por Simon Leinen. Eis uma versão modificada do programa de Simon:

```
#!/usr/local/bin/perl

use SNMP_Session "0.60";
use BER;
use Socket;

$session = SNMPv1_Session->open_trap_session ( );
while (( $trap, $sender, $sender_port ) = $session->receive_trap ( ))
{
    chomp ($DATE=~/bin/date \'+%a %b %e %T\''');
    print STDERR "$DATE - " . inet_ntoa($sender) . " - port:
$sender_port\n";
    print_trap ($session, $trap);
}
1;

sub print_trap ( $$ ) {
    ($this, $trap) = @_;
    ($community, $ent, $agent, $gen, $spec, $dt, @bindings) = \
        $this->decode_trap_request ($trap);
    print "    Community:\t$community.\n";
    print "    Enterprise:\t$BER::pretty_oid ($ent).\n";
    print "    Agent addr:\tinet_ntoa ($agent).\n";
    print "    Generic ID:\t$gen\n";
```

```

print "  Specific ID:\t$spec\n";
print "  Uptime:\t".BER::pretty_uptime_value ($dt)."\n";
$prefix = "  bindings:\t";
foreach $encoded_pair (@bindings)
{
    ($oid, $value) = decode_by_template ($encoded_pair, "%(%0%0");
    #next unless defined $oid;
    print $prefix.BER::pretty_oid ($oid)." => ".pretty_print
    ($value)."\n";
    $prefix = "  ";
}
}

```

Esse programa exibe traps à medida que são recebidas dos diversos dispositivos existentes na rede. Examine a seguir uma parte da saída que mostra duas traps:

```

Mon Apr 28 22:07:44 - 10.123.46.26 - port: 63968
community: public
enterprise: 1.3.6.1.4.1.2789.2500
agent addr: 10.123.46.26
generic ID: 6
specific ID: 5247
uptime: 0:00:00
bindings: 1.3.6.1.4.1.2789.2500.1234 => 14264026886

Mon Apr 28 22:09:46 - 172.16.51.25 - port: 63970
community: public
enterprise: 1.3.6.1.4.1.2789.2500
agent addr: 172.16.253.2
generic ID: 6
specific ID: 5247
uptime: 0:00:00
bindings: 1.3.6.1.4.1.2789.2500.2468 => Hot Swap Now In Sync

```

O formato da saída é o mesmo para ambas as traps. A primeira linha mostra a data e hora em que a trap ocorreu, juntamente com o endereço IP do dispositivo emissor da trap.

Você deve conhecer a maioria dos itens da saída restantes. O item da saída, bindings, lista as vinculações de variáveis enviadas na PDU da trap. No exemplo acima, cada trap continha uma vinculação de variáveis. A ID de objeto está no formato numérico, que não é particularmente amistoso. Se uma trap tiver mais de uma vinculação de variáveis, esse programa exibirá cada vinculação, uma após a outra.

Um sistema de monitoração instantânea pode ser adaptado, usando esse script em Perl para coletar traps e algum outro programa para inspecionar as traps quando recebidas. Após analisar as traps, as possibilidades são infinitas. Você pode escrever regras definidas pelo usuário que investigam as traps significativas e, quando acionadas, enviam um alerta de e-mail, atualizam um banco de

dados de eventos, enviam uma mensagem para um pager etc. Essas modalidades de soluções funcionam bem para uma empresa com pouco ou nenhum orçamento para os softwares de NMS distribuídos no mercado ou se você está em uma rede pequena e não precisa de uma ferramenta de gerenciamento de rede muito complexa.

## *Usando o Trap Receiver da Network Computing Technologies*

O Trap Receiver da Network Computing Technologies é um programa distribuído gratuitamente que compensa experimentar.\* Esse programa, executado atualmente apenas nos sistemas baseados no Windows, exibe informações sobre as traps recebidas. Tem uma interface padrão mas pode ser configurado para executar algumas ações sobre as traps, como a função Command for Automatic Action do OpenView. A Figura 10-7 mostra a interface do usuário do Trap Receiver.

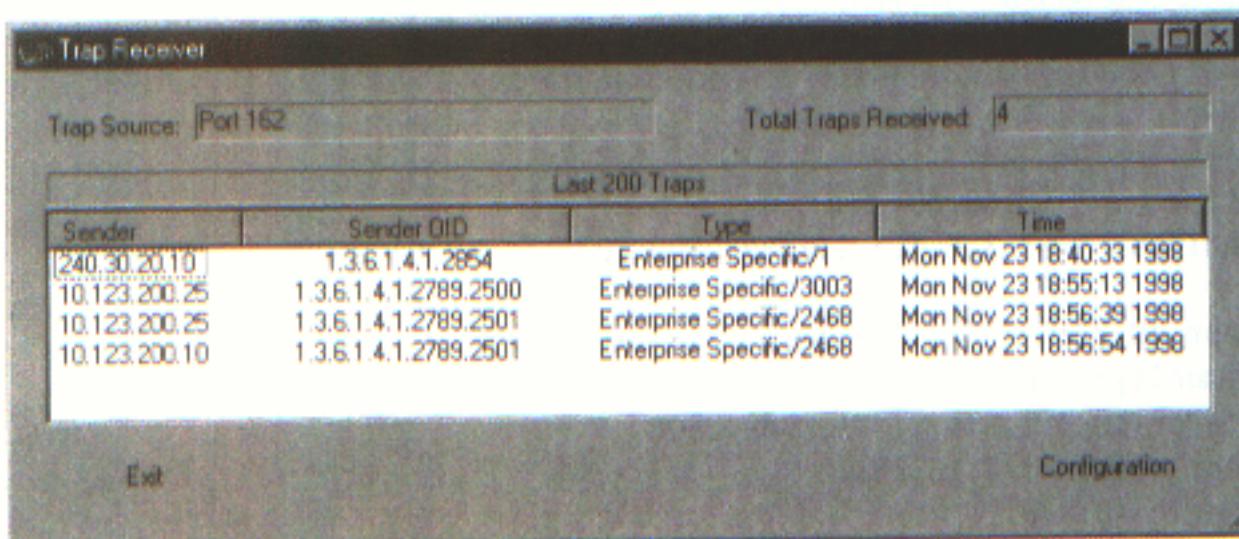


Figura 10-7 Trap Receiver

Existem métodos para registrar e enviar mensagens e traps, enviar e-mail ou mensagem de pager em resposta a uma trap, assim como executar comandos. Ao escrever um código em C ou C++, você pode acessar um fluxo de traps internas.

Esse programa pode ser um bom ponto inicial para os administradores do Windows que desejam utilizar o SNMP mas não dispõem de recursos para implementar algo como o OpenView. É fácil de usar, extensível e gratuito.

## *Recebendo traps com o Net-SNMP*

O último recebedor de traps que discutiremos faz parte do pacote do Net-SNMP, também distribuído gratuitamente. O *snmptrapd* permite enviar mensa-

\* Esse software pode ser encontrado na página da Web da empresa, em <http://www.ncomtech.com>.

gens de traps do SNMP para instalações como o *syslog* ou *stdout* do Unix. Para a maioria das aplicações, o programa funciona em segundo plano, direcionando mensagens para o *syslog(8)*. Existem alguns parâmetros de configuração para o lado *syslog* do *snmptrapd*; esses parâmetros informam ao *snmptrapd* o nível de facilidade que ele deve usar para as mensagens do *syslog*. O comando a seguir direciona traps para a saída padrão (-P) e não para o *syslog*, à medida que recebidas:

```
$ ./snmptrapd -P  
2000-12-13 19:10:55 UCD-SNMP Version 4.1.2 Started.  
2000-12-13 19:11:14 sunserver2.ora.com [12.1.45.26]  
enterprises.2789.2500:  
    Enterprise Specific Trap (1224) Uptime: 5 days, 10:01:20.42  
    enterprises.2789.2500.1224 = 123123  
  
2000-12-13 19:11:53 sunserver2.ora.com [12.1.45.26]  
enterprises.2789.2500:  
    Enterprise Specific Trap (1445) Uptime: 5 days, 10:01:21.20  
    enterprises.2789.2500.1445 = "Fail Over Complete"
```

Agora, a saída deve ser conhecida; é semelhante aos relatórios gerados pelos outros programas que discutimos no capítulo. O daemon de trap do Net-SNMP é outra ferramenta excelente para quem escreve scripts. Um script simples em linguagem Perl pode verificar o arquivo em que o *snmptrapd* registra as respectivas traps, procurando os eventos importantes e reagindo adequadamente. É fácil construir um sistema de monitoração poderoso e flexível, com pouco ou nenhum custo.

Conhecemos diversos pacotes que podem receber traps e agir a partir delas, de acordo como conteúdo das traps. Lembre-se de que todos esses programas, sejam gratuitos ou custando dezenas de milhares de dólares, estão fazendo basicamente o mesmo: ouvindo em alguma porta (geralmente, a porta 162 do UDP) e aguardando a chegada de mensagens do SNMP. O que distingue os diversos pacotes é a possibilidade de fazer algo construtivo com as traps. Alguns permitem programar ganchos que executam outro programa quando uma trap específica é recebida. Os mais simples monitores de traps enviam apenas uma mensagem registrando a trap para um ou mais arquivos ou facilidades. Geralmente, esses pacotes são menos dispendiosos do que os monitores de traps comerciais, mas podem ser elaborados para funcionarem como sistemas completos com algum esforço de programação adicional. Programas como o Perl permitem estender esses pacotes mais simples.

## Enviando traps

Agora, você já deve ter um mecanismo para receber traps. Nesta seção, examinaremos alguns utilitários diferentes, que enviam e permitem desenvolver traps adequadas a seu ambiente.

Você observará que quase todos os utilitários de traps são baseados na linha de comando. Isso permite executar o comando a partir do próprio script, o que geralmente é o que se deseja. Por exemplo, você pode escrever um script de shell que verifica o espaço de disco a cada cinco minutos e envia uma trap à NMS, se o sistema estiver executando com pouco espaço de disco. Você também pode utilizar esses geradores de traps dentro de programas e scripts já existentes. Se você tiver um script em Perl que acessa um banco de dados, poderá utilizar o módulo de SNMP da linguagem Perl para enviar uma trap a partir do script se uma inclusão em um banco de dados falhar. As possibilidades são quase infinitas.

Embora existam vários programas *snmptrap* diferentes, todos eles são semelhantes. Em particular, embora a sintaxe da linha de comando possa variar, todos eles esperam basicamente os mesmos argumentos:

#### *Porta*

A porta do UDP para a qual a trap deve ser enviada. A porta default é a 162.

#### *Versão do SNMP*

A versão do SNMP adequada à trap que será enviada. Algumas traps são definidas somente para a Versão 2. Observe que algumas ferramentas do SNMP aceitam somente a Versão 1.

#### *Nome do host ou endereço IP da NMS*

O nome do host ou o endereço IP de sua NMS – ou seja, o destino da trap. É melhor utilizar um endereço IP do que um nome de host, caso esteja enviando traps durante uma interrupção do Domain Name System (DNS). Lembre-se de que o SNMP é mais valioso quando sua rede está falhando; portanto, evite pressupor que você possui uma rede totalmente funcional, ao elaborar traps.

#### *Nome de comunidade*

O nome de comunidade a ser enviado com a trap. É possível configurar a maioria das estações de gerenciamento para ignorar as traps sem a string de comunidade adequada.

#### *OID de empresa*

A OID completa da empresa para a trap que você deseja enviar: tudo na OID da trap, desde os primeiros caracteres .1 até o número da empresa, incluindo as subárvore dentro da empresa mas não o número específico da trap. Por exemplo, se o número de sua empresa for 2789, você subdividiu ainda mais a empresa de modo a incluir um grupo de traps com a numeração 5000, e você desejar enviar a trap específica 1234, a OID da empresa será .1.3.6.1.4.1.2789.5000.

Se existir algum motivo para enviar uma trap genérica, você poderá definir a ID da empresa com algo desejado – mas, provavelmente, seria convenien-

te definir essa ID com o número de sua própria empresa, se existir. Afora, o caso mais confuso. Existem algumas traps específicas, definidas em várias MIBs públicas. Como enviá-las? Basicamente, construa algo parecido com uma OID de empresa. Convém examinar um exemplo. Uma trap similar é a *rdbmsOutOfSpace*, definida na RDBMS MIB. Sua OID completa é .1.3.6.1.2.1.39.2.2 (.iso.org.dod.internet.mgmt.mib-2.rdbmsMIB.rdbmsTraps.rdbmsOutOfSpace). Para enviar essa trap, que é realmente uma notificação do SNMPv2, você usaria tudo até o *rdbmsTraps* como OID de empresa, e a ID de objeto inteira como o número de trap específica.

#### *Nome do host ou endereço IP do emissor*

O endereço IP do agente que está enviando a trap. Embora isso possa parecer supérfluo, pode ser importante se existir um servidor proxy entre o agente e a NMS. Esse parâmetro permite registrar o endereço real do agente dentro do pacote do SNMP; por sua vez, a NMS lerá o endereço do agente da trap e ignorará o endereço do emissor do pacote. Se você não especificar esse parâmetro, será utilizado quase sempre o endereço da máquina emissora da trap.

#### *Número de trap genérica*

Um número no intervalo de 0-6. As verdadeiras traps genéricas têm números de 0-5; ao enviar uma trap específica de empresa, defina esse número com 6. A Tabela 2-8 lista as traps genéricas.

#### *Número de trap específica*

Um número indicando a trap específica a ser enviada. Ao enviar uma trap genérica, esse parâmetro é ignorado – seria mais conveniente defini-lo com zero. Se estiver enviando uma trap específica, o número da trap fica a seu critério. Por exemplo, se você enviar uma trap com a OID .1.3.6.1.4.1.2500.3003.0, a parte 3003 é o número da trap específica.

#### *Marcação do tempo*

O tempo decorrido entre a última inicialização da entidade da rede e a geração da trap.

#### *OID\_1, tipo\_1, valor\_1*

Vinculações de dados a serem incluídas na trap. Cada vinculação de dados consiste em uma OID acompanhada de um banco de dados, seguido do valor a ser enviado. A maioria dos programas permite incluir qualquer número de vinculações de dados em uma trap. Observe que as OIDs para essas vinculações de variáveis são geralmente específicas para a trap e, portanto, “subordinada” à OID específica da trap. Mas isso não é uma exigência, e freqüentemente é útil enviar vinculações não definidas como parte da trap.

Antes de prosseguirmos no assunto desta seção, vamos revisar o que aprendemos no Capítulo 2 sobre os diversos tipos de dados:

- Toda variável enviada possui um tipo de dado específico.
- Diferentes tipos de dados são aceitos pelas diferentes versões do SNMP.
- Alguns tipos de dados comuns são INTEGER, OctetString, Null, Counter, Gauge e TimeTicks.

Saiba que nem todos os programas aceitam todos os tipos de dados. Por exemplo, o módulo do SNMP da Perl aceita somente os tipos OctetString, INTEGER e OID, enquanto os comandos *snmptrap* do OpenView e Net\_SNMP têm suporte para esses três tipos e muito mais. Para cada um dos pacotes utilizados listaremos, se aplicável, cada tipo de dado aceito pelo programa.

Nas próximas seções, discutiremos os programas *snmptrap* do OpenView, da Network Computing Technologies e do Net-SNMP. Além disso, incluiremos um script que usa um módulo de Perl para enviar traps. Se você não utiliza esses programas específicos em seu ambiente, não se preocupe. Você também poderá relacionar esses exemplos a seus programas internos.

## *Enviando traps com o OpenView*

O OpenView dispõe de um programa de linha de comando para gerar traps aleatórias denominadas *snmptrap*. O programa *snmptrap* aceita os tipos de dados counter, counter32, counter64\*, gauge, gauge32, integer, integer32, ipaddress, null, objectidentifier, octetstring, octetstringascii, octetstringhex, octetstringoctal, opaque, opaqueascii, opaquehex, opaqueoctal, timeticks e unsigned32. Sua estrutura de linha de comando é parecida com a seguinte:

```
snmptrap -c community [-p port] node_addr enterprise_id agent-addr  
generic \  
specific timestamp [OID type value] ...
```

Eis um comando *snmptrap* característico, que envia uma trap, com três vinculações de variáveis de strings ASCII para os valores:

```
$ /opt/DV/bin/snmptrap -c public nms \  
.1.3.6.1.4.1.2789.2500 "" 6 3003 "" \  
.1.3.6.1.4.1.2789.2500.3003.1 octetstringascii "Oracle" \  
.1.3.6.1.4.1.2789.2500.3003.2 octetstringascii "Backup Not Running" \  
.1.3.6.1.4.1.2789.2500.3003.3 octetstringascii "Call the DBA Now for  
Help"
```

É um comando completo é difícil imaginar que você o digite na linha de comando. Separemos em duas partes. A primeira linha especifica a string de comunidade (public) e o endereço para o qual a trap deve ser enviada (nms, embo-

ra, na prática, seria melhor utilizar um endereço IP em vez de um nome de nó). Em alguns aspectos, a linha seguinte é a mais complexa. Especifica a ID de empresa para a trap a ser enviada (.1.3.5.1.6.1.2789.2500, que é uma subárvore da árvore específica da empresa pertinente às traps); o endereço do agente emissor da trap (nesse caso, a string nula "", que utiliza (por default) o endereço do agente; se estiver usando um servidor proxy, convém especificar explicitamente o endereço do agente); o número da trap genérica (6, que é utilizado para todas as traps específicas da empresa); o número da trap específica (3003, que atribuímos) e uma marcação de hora ("", que usa, por default, a hora atual).

As três linhas restantes especificam três vinculações de variáveis a serem incluídas com a trap. Para cada vinculação, temos a ID de objeto da variável, os respectivos tipos de dado e valor. As variáveis que estamos enviando estão definidas em nossa MIB privada (específica de empresa), de modo que todas as suas OIDs começam com .1.3.6.1.4.1.2789.2500. Todas as variáveis são strings, e o respectivo tipo de dado é octetstringascii. A PDU da trap será empacotada com essas três strings, entre outros itens. O programa que receber a trap decodificará a PDU da trap e detectará que existem três vinculações de variáveis na trap. Essas vinculações de variáveis, assim como a que indica "Call the DBA Now for Help" (Chame o DBA agora para obter ajuda), podem ser utilizadas para avisar ao operador que algo errado aconteceu.

## Enviando traps com a Perl

No Capítulo 8, aprendemos a utilizar o *get* e *set* do módulo Perl do SNMP. Nesta seção, veremos como usar a rotina *snmptrap()* para gerar traps. Atualmente, o *SNMP\_util* aceita somente três tipos para as traps: *string*, *int* e *oid*. Isso pode parecer limitante mas atende à maioria das necessidades. Examine a seguir como o *snmptrap* é chamado:

```
snmptrap(communityname@host:port_number, enterpriseOID, host_name_from, \
generic_ID, specific_ID, OID, type, value, [OID, type, value ...])
```

Uma chamada ao *snmptrap* pode incluir qualquer número de valores; para cada valor, você deve especificar a ID de objeto, o tipo de dado e o valor sendo informado. O script a seguir gera uma trap com único valor:

```
#!/usr/local/bin/perl
# Filename: /opt/local/perl_scripts/snmptrap.pl

use SNMP_util "0.54"; # This will load the BER and SNMP_Session for us

snmptrap("public\@nms:162", ".1.3.6.1.4.1.2789", "sunserver1", 6, 1247, \
".1.3.6.1.4.1.2789.1247.1", "int", "2448816");
```

A chamada à rotina *snmptrap()* envia uma trap à porta 162 na nms do host. A trap é enviada do sunserver1 do host; contém uma única vinculação de variáveis, para o objeto .1.3.6.1.4.1.2789.1247.1. O tipo da OID é int e seu valor é 2448816.

Agora, tentemos enviar uma trap com diversos valores (várias vinculações de variáveis). O primeiro objeto que informaremos é um inteiro, ao qual atribuiremos um valor arbitrário, 4278475. O segundo objeto possui um valor de string e é um aviso de que nosso banco de dados está paralisado. Como estamos utilizando a OIDs pertencente a nossa própria empresa, podemos definir esses objetos com o que desejarmos:

```
snmptrap("public\@nms:162", ".1.3.6.1.4.1.2789", "sunserver2", 6, 3301, \
          ".1.3.6.1.4.1.2789.3301.1", "int",      "4278475", \
          ".1.3.6.1.4.1.2789.3301.2", "string", "Sybase DB Stopped");
```

Podemos usar o programa *snmptrapd* do Net-SNMP para monitorar as traps recebidas. Executamos o código em Perl anterior, rodando o *snmptrapd* no modo *stdout*, e recebemos:

```
$ ./snmptrapd -P
1999-10-12 09:45:08 [12.1.45.26] enterprises.2789.3000:
    Enterprise Specific Trap (3301) Uptime: 0:00:00
    enterprises.2789.3301.1 = 4278475
    enterprises.2789.3301.2 = "Sybase DB Stopped"
```

O *snmptrapd* informou ambos os valores que enviamos na trap: vemos o valor inteiro, 4278475, e a notificação de que o Sybase paralisou. Embora esse exemplo seja muito artificial, não é tão diferente daquilo que você faria ao escrever seu software de monitoração. Você escreveria o código necessário para monitorar sistemas fundamentais, como seu banco de dados, e usar o módulo do SNMP da Perl para enviar traps quando ocorrerem eventos significativos. Assim, você pode utilizar qualquer programa capaz de receber traps para informar a você a chegada das traps. Se necessário, você pode adicionar uma lógica que analise os valores enviados na trap ou execute outras ações, como notificar um operador via mensagem de pager.

## *Enviando traps com o Trap Generator da Network Computing Technologies*

Esse utilitário de linha de comando baseado no Windows oferece os mesmos recursos que seus parceiros do Unix. Reconhece os tipos de dados String, Counter, Gauge, Integer, Address, OID e TimeTicks. A linha de comando do *nttrapgen* parece com a seguinte:

```
nttrapgen.exe -d DestinationIpAddress:port -c CommunityName
               -o senderOID -i senderIP -g GenericTrapType
               -s SpecificTrapType -t timestamp -v OID TYPE VALUE
```

Eis como utilizar o *nttrapgen* para enviar uma trap que informa que a bateria do no-break está baixa. Usamos o tipo de dado String para enviar uma mensagem informativa e usamos a trap 4025.1 da ID de nossa empresa privada, 2789:

```
C:\tools> nttrapgen.exe -d nms:162 -c public -o ^\snmptrap\nttrapgen\nttrapgen.vbs  
1.3.6.1.4.1.2789.4025 -i 10.123.456.4 -g 6 -s 4025 -t 124501 ^\snmptrap\nttrapgen.vbs  
-v 1.3.6.1.4.1.2789.4025.1 STRING 5 Minutes Left On UPS Battery
```

Essa trap será enviada para nossa estação de gerenciamento de rede (que possui o nome de host nms) na porta 162, que é a porta padrão para as traps do SNMP. Toda estação de gerenciamento deve ser capaz de receber a trap e reagir adequadamente. Você pode utilizar esse comando em scripts em batch, que são basicamente idênticos aos scripts de shell do Unix. Portanto, você pode utilizar o *nttrapgen* para gerar traps conforme a necessidade: é possível escrever scripts que monitorem os principais processos e gerem traps quando eventos interessantes ocorrerem. Assim como no exemplo anterior em Perl, você pode utilizar esse gerador de traps simples em seu ambiente, se não precisar de um sistema de gerenciamento mais complexo.

## Enviando traps com o Net-SNMP

Esse programa *snmptrap* é muito semelhante ao programa de mesmo nome do OpenView. Utiliza uma única letra para indicar os tipos de dados, conforme apresentados na Tabela 10-2.

Tabela 10-2 Tipos de dados do *snmptrap* do Net-SNMP

Abreviação	Tipo de dado
a	Endereço IP
c	Counter
d	String Decimal
i	Inteiro
n	Nulo
o	ID de objeto
s	String
t	Marcação de tempo
u	Inteiro sem sinal
x	String hexadecimal

Examine a seguir como o programa *snmptrap* do Net-SNMP é chamado:

```
snmptrap hostname community enterprise-oid agent \
generic-trap specific-trap uptime [OID type value]...
```

Se você usar apóstrofos (") no lugar da hora, o *snmptrap* inserirá a hora atual na trap. O comando a seguir gera uma trap com um único valor. A ID de objeto é 2005.1, em nossa empresa privada; o valor é uma string que informa que o servidor da Web foi reiniciado:

```
$ snmptrap nms public .1.3.6.1.4.1.2789.2005 ntserver1 6 2476317 '' \  
.1.3.6.1.4.1.2789.2005.1 s "WWW Server Has Been Restarted"
```

Eis como enviar uma notificação da Versão 2 com o Net-SNMP:<sup>\*</sup>

```
$ snmptrap -v2c nms public '' .1.3.6.1.6.3.1.1.5.3 \  
ifIndex i 2 ifAdminStatus i 1 ifOperStatus i 1
```

Na realidade, o comando é mais simples do que seu equivalente da Versão. Não existem números genéricos, números específicos nem IDs de fornecedor. O argumento "" usa como default o tempo de funcionamento do sistema atual. A OID especifica a notificação de *linkDown*, com três vinculações de dados especificando o status do link. A definição de *linkDown* na IF-MIB declara que a notificação de *linkDown* deve incluir os objetos *ifIndex*, *ifAdminStatus* e *ifOperStatus*, que informam o índice da interface paralisada, o respectivo status administrativo e operacional, respectivamente.

Para os objetos *ifAdminStatus* e *ifOperStatus*, um valor de 1 indica que o link está funcionando. Sendo assim, essa notificação informa que a interface 2 mudou do estado “paralisado” para “em funcionamento”.

Mais uma vez, a ferramenta de linha de comando *snmptrap* permite integrar a monitoração do SNMP a scripts de shell e outros programas.

### Instruindo o hardware a gerar traps

Ao instalar um novo equipamento, você deve verificar se esse item gera traps corretamente. Testar a possibilidade de gerar traps em seu equipamento ajuda a testar o comportamento de sua NMS; você pode certificar-se de que o equipamento lida com as traps conforme almejado. A melhor maneira de testar o novo hardware é procurar na MIB de seu fornecedor todos os TRAP-TYPEs definidos. Com essa informação, você conhecerá o tipo de traps implementado pelo fornecedor. Por exemplo, consultei nossa MIB do APC e observei que a unidade enviará uma trap ao alternar para a bateria quando a força de AC (corrente alternada) acabar. Para testar esse recurso, protegi a área de nosso centro de dados e desliguei o interruptor do circuito para simular uma falta de energia. A trap foi gerada, mas apareceu na categoria de eventos Error porque eu não tinha a MIB correta carregada no OpenView. Usei a OID dos eventos Error e procurei uma correspondência nas MIBs do APC. Quando encontrei, carreguei o arquivo da MIB no OpenView e repeti o teste. Dessa vez, quando a trap foi recebida, o OpenView inseriu uma mensagem informativa nas Event Categories.

A maioria dos roteadores, comutadores e dispositivos de rede compatíveis com o SNMP, pode gerar traps de *linkDown*. Na RFC 1157, uma trap de *linkDown* é uma “falha em um dos links de comunicação representados na configuração do agente”. Isso significa que se você começar a desconectar portas em seu roteador, deverá receber traps, certo? Sim mas, primeiramente, certifique-se de

\* Para obter informações sobre como enviar notificações da Versão 3 com o Net-SNMP, consulte o Apêndice F.

não começar a desconectar os servidores do banco de dados da produção. Além disso, não desconecte a porta pela qual o dispositivo enviaria a trap de volta para a NMS. Lembre-se de que o SNMP é elaborado a partir da premissa de que a rede não é confiável – se algo enviar uma trap mas não for possível para a trap alcançar seu destino, ninguém descobrirá. Por default, uma trap de *linkDown* na aparecerá nas Event Categories do OpenView porque a definição default para *linkDown* é “Log only”; observe no arquivo de log \$OV\_LOG/trapd.log a chegada dessas traps. Quando você tiver um mecanismo para receber traps, ativar e desativar o link em seu dispositivo deverá enviar algumas traps para seu sistema.

## *Usando ganchos em seus programas*

Um gancho é uma interface conveniente, que permite integrar seu próprio código a outro produto. O editor de texto *Emacs* é um bom exemplo de programa que usa ganchos, quase o tempo todo, para permitir que seus usuários estendam seu modo de operação. Examinemos o programa simples a seguir para explicar esse conceito um pouco mais:

```
# Logical Sample Program NH1
# PROGRAM COMMENTS
# PROGRAM BEGINS

PROGRAM ADDS      $VAR1 + $VAR2 = $VAR3
PROGRAM SUBTRACTS $VAR5 - $VAR6 = $VAR7
PROGRAM PRINTS RESULTS $VAR3 $VAR7

# PROGRAM ENDS
```

Esse programa simplesmente ADICIONA, SUBTRAI e IMPRIME RESULTADOS; não possui quaisquer ganchos. Para adicionar um recurso, é necessário modificar o código. Para um programa pequeníssimo como esse, trata-se de um exercício trivial, mas seria difícil em um programa de outro tamanho. O programa a seguir tem alguns ganchos que permitem adicionar extensões:

```
# Logical Sample Program H1
# PROGRAM COMMENTS
# PROGRAM BEGINS
PROGRAM RUNS $PATH/start.sh

PROGRAM ADDS      $VAR1 + $VAR2 = $VAR3
PROGRAM SUBTRACTS $VAR5 - $VAR6 = $VAR7
PROGRAM PRINTS RESULTS $VAR3 $VAR7

PROGRAM RUNS $PATH/end.sh
# PROGRAM ENDS
```

Observe as duas instruções RUNS adicionais. Esses ganchos permitem executar tudo o que você desejar, no início ou final do programa. O primeiro progra-

ma, *start.sh*, poderia ser tão simples quanto o comando *echo "I am starting"* (*Estou iniciando*), que envia uma mensagem simples para o console do sistema ou de gerenciamento. Além disso, esse script poderia chamar um dos programas de geração de traps para enviar uma trap para a NMS informando que algum programa está iniciando. Seria ainda mais útil enviar uma mensagem quando o programa terminar, incluindo possivelmente informações sobre o status do programa. Eis um programa um pouco mais complexo, que executa um script, e fornece alguns argumentos para que o script possa retornar informações úteis para a NMS ao gerar uma trap:

```
# Logical Sample Program H2
# PROGRAM COMMENTS
# PROGRAM BEGINS
PROGRAM RUNS $PATH/start.sh $PROGRAM_NAME

PROGRAM ADDS      $VAR1 + $VAR2 = $VAR3
PROGRAM SUBTRACTS $VAR5 - $VAR6 = $VAR7
PROGRAM PRINTS RESULTS $VAR3 $VAR7

PROGRAM RUNS $PATH/end.sh $PROGRAM_NAME $VAR1 $VAR2 $VAR3 $VAR5 $VAR6
$VAR7
# PROGRAM ENDS
```

Com os argumentos adicionais disponíveis para os programas de gancho, é possível gerar mensagens como “As Garalhufas do Programa terminaram com vendas a 4 dólares e YTD em 7 dólares”. Se seus programas de gancho forem scripts de shell, basta adicionar comandos *snmptrap* por meio de um editor de texto. Assim que você terminar de adicionar o código do *snmptrap*, poderá testar o programa de gancho, executando-o na linha de comando.

Na maioria dos sistemas, muitos scripts podem se beneficiar dos ganchos de *snmptrap*. Por exemplo, nas máquinas do Solaris ou Linux, alguns de seus scripts de */etc/init.d* podem ser readaptados de modo a fazer uso de comandos *snmptrap*. Seria conveniente ter algum tipo de notificação quando processos importantes, como seus servidores da Web ou de DNS iniciarem e pararem. Ter conhecimento dessas informações facilita muito mais a vida de sua helpdesk. (O agente de SNMP do SystemEDGE da Concord oferece recursos mais rigorosos de monitoração de processos. Leia o Capítulo 11, para obter informações sobre esse produto.)

É mais difícil adicionar ganchos aos programas escritos em linguagens como C, porque você precisa acessar o código-fonte e ainda adivinhar onde devem ser inseridos os ganchos. Após identificar onde os ganchos devem entrar e adicioná-los, você deve recompilar o código-fonte. Alguns programas dispõem de ganchos internos, o que lhe permite executar programas externos ou RPCs. Consulte a documentação de seu programa para obter as localizações desses ganchos. Isso é muito mais prático do que tentar construir seus próprios ganchos em outro programa. Após definir quais programas externos serão chamados, você poderá começar a escrever suas próprias traps ou adicionar às já existentes.

## *Agentes extensíveis de SNMP*

Em algumas situações, você precisará estender a funcionalidade de um agente. Geralmente, estender um agente significa adicionar ou modificar as MIBs que o agente suporta. Alguns agentes, que supostamente oferecem suporte para o SNMP, cobrem somente um número mínimo de MIBs um tanto inúteis – obviamente uma situação frustrante para quem planeja muitas atividades de gerenciamento de redes automatizado. O upgrade de seu software para uma versão mais recente do SNMP, como a Versão 2 ou 3, não será de grande ajuda; você não obterá mais informações de um dispositivo do que se estivesse utilizando a SNMPv1. As versões mais recentes do SNMP adicionam recursos ao protocolo (como segurança adicional ou opções mais sofisticadas para recuperar e definir valores) mas as informações disponíveis em qualquer dispositivo são definidas nas MIBs do agente, que independem do protocolo em si.

Ao enfrentar as limitações de um agente, você pode utilizar os agentes extensíveis.\* Esses programas ou extensões para programas existentes permitem estender uma MIB de um agente específico e recuperar valores em uma origem externa (um script, programa ou arquivo). Em alguns casos, é possível retornar os dados como se procedessem do próprio agente. Na maioria das vezes, você não perceberá uma diferença entre as MIBs internas do agente e seus agentes extensíveis. Vários agentes extensíveis permitem ler arquivos, executar programas e retornar resultados; podem até retornar tabelas de informações. Alguns agentes têm opções configuráveis que permitem executar programas externos e ter funções predefinidas, como verificadores de espaço de disco, incorporadas.

Os agentes do OpenView, Net-SNMP e SystemEDGE são exemplos de agentes extensíveis. O OpenView oferece um agente extensível separado, que permite estender o agente principal (*snmpdm*); as solicitações para o agente extensível não funcionarão, exceto se o agente principal estiver em execução. É possível iniciar e interromper o agente extensível sem prejudicar o agente principal. Para personalizar o agente extensível, defina os novos objetos usando o formato

\* Não diferenciamos entre os agentes existentes, que podem ser estendidos, e os agentes que existem unicamente para dar suporte às extensões. Denominamos os dois agentes “agentes extensíveis”.

ASN.1, conforme especificado pelo SMI. O agente da Net-SNMP usa uma abordagem alternativa e não distingue entre o agente principal e o agente extensível; existe um único agente com o qual lidar. Você pode utilizar o ASN.1 para definir novos objetos (como os agentes extensíveis do OpenView) mas também é possível adicionar extensões sem criar qualquer ASN.1, tornando esse agente muito mais acessível para o administrador iniciante. O SystemEDGE é semelhante ao Net-SNMP, no sentido que existe apenas um agente. Dos três agentes discutidos neste capítulo, é o mais fácil de estender. A Figura 11-1 compara as estratégias de design dos agentes do OpenView, Net-SNMP e SystemEDGE.

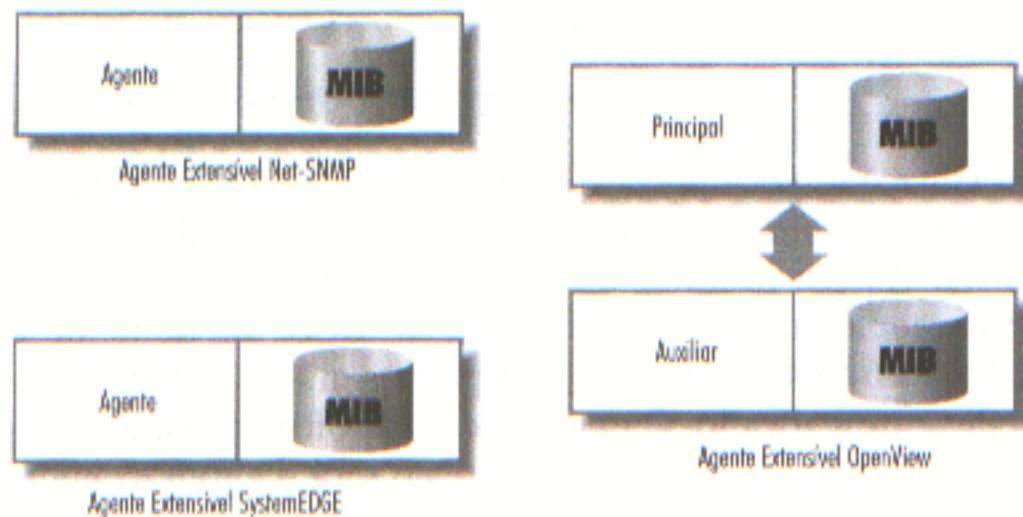


Figura 11-1 Arquitetura de agentes extensíveis

Os três agentes possuem opções de configuração bastante abrangentes e permitem estender o agente local sem utilizar uma programação pesada. Talvez você precise escrever alguns scripts ou alguns programas curtos do C, mas com os exemplos de programas oferecidos aqui e os milhares existentes na Internet,\* os usuários não programadores também podem conseguir grandes façanhas.

Começaremos com o agente do Net-SNMP por ser o mais simples, depois passaremos para o SystemEDGE. Encerramos a discussão como agente extensível do OpenView. Para obter informações sobre onde obter esses agentes, consulte o Capítulo 5.

## Net-SNMP

Quando você instala o pacote do Net-SNMP, ele gera um exemplo de arquivo de configuração `snmpd.conf` denominado `EXAMPLE.conf` no diretório de origem. Esse arquivo contém alguns exemplos que demonstram como estender seu agente. Examine esse arquivo para conhecer as atividades permitidas e não permitidas. Discutiremos apenas alguns recursos do Net-SNMP: procurar qualquer

\* Consulte o Capítulo 1 para obter uma lista de alguns sites da Web que têm links para softwares de SNMP comerciais e gratuitos.

número de processos em execução (proc), executar um comando que retorne uma única linha de saída (exec), executar um comando que retorne várias linhas de saída (exec) e verificar a utilização do espaço de disco (disk).

É possível encontrar o principal arquivo de configuração do Net-SNMP em `$NET_SNMP_HOME/share/snmp/snmpd.conf`, onde `$NET_SNMP_HOME` é o diretório em que você instalou o Net-SNMP. Examine a seguir o arquivo de configuração que usaremos ao longo do restante desta seção:

```
# FileName: $NET_SNMP_HOME/share/snmp/snmpd.conf
# Pesquisa processos em execução
# Os itens daqui aparecerão na ucdavis.procTable
proc sendmail 10 1
proc httpd

# Retornar o valor do programa executado com um parâmetro transmitido.
# Os itens daqui aparecerão na ucdavis.extTable
exec FileCheck /opt/local/shell_scripts/filecheck.sh /tmp/vxprint.error

# Retorno de várias linhas do comando
# Precisa de uma OID exclusiva
# Usei um subconjunto de minha ID de empresa registrada ID (2789) dentro da OID
exec .1.3.6.1.4.1.2021.2789.51 FancyCheck /opt/local/shell_scripts/fancycheck.sh \
/core

# Procura valores mínimos em discos
disk / 100000
```

Sempre que você efetuar alterações no arquivo de configuração do agente do Net-SNMP, poderá instruí-lo a reler a configuração ao enviar ao processo um sinal de HUP:

```
$ ps -ef | grep snmpd
root 12345 1 0 Nov 16 ? 2:35 /usr/local/bin/snmpd
$ kill -HUP 12345
```

Examinemos agora o arquivo em si. O primeiro comando proc instrui a procurar o processo `sendmail`. Os números 10 e 1 definem quantos processos `sendmail` desejamos executar em determinado momento (no máximo 10 e no mínimo 1). O segundo comando proc informa que queremos pelo menos um processo `httpd` em execução. Para constatar os efeitos desses comandos sobre nosso agente, examinemos um `snmpwalk` da `ucdavis.procTable` (.1.3.6.1.4.1.2021.2):

```
$ snmpwalk sunserver2 public .1.3.6.1.4.1.2021.2
enterprises.ucdavis.procTable.prEntry.prIndex.1 = 1
enterprises.ucdavis.procTable.prEntry.prIndex.2 = 2
enterprises.ucdavis.procTable.prEntry.prNames.1 = " sendmail"
enterprises.ucdavis.procTable.prEntry.prNames.2 = "httpd"
enterprises.ucdavis.procTable.prEntry.prMin.1 = 1
enterprises.ucdavis.procTable.prEntry.prMin.2 = 0
enterprises.ucdavis.procTable.prEntry.prMax.1 = 10
```

```
enterprises.ucdavis.procTable.prEntry.prMax.2 = 0
enterprises.ucdavis.procTable.prEntry.prCount.1 = 1
enterprises.ucdavis.procTable.prEntry.prCount.2 = 6
enterprises.ucdavis.procTable.prEntry.prErrorFlag.1 = 0
enterprises.ucdavis.procTable.prEntry.prErrorFlag.2 = 0
enterprises.ucdavis.procTable.prEntry.prErrMessage.1 = ""
enterprises.ucdavis.procTable.prEntry.prErrMessage.2 = ""
enterprises.ucdavis.procTable.prEntry.prErrFix.1 = 0
enterprises.ucdavis.procTable.prEntry.prErrFix.2 = 0
```

O agente retorna o conteúdo da *procTable*. Nessa tabela, as entradas dos processos *sendmail* e *httpd* ocupam as instâncias 1 e 2. *prMin* e *prMax* são os números mínimo e máximo que definimos para os processos *sendmail* e *httpd*.\* O valor de *prCount* indica o número de processos em execução no momento: parece que temos um processo *sendmail* e seis processos *httpd*. Para saber o que acontece quando o número de processos extrapola a faixa especificada, eliminemos os seis processos *httpd* e examinemos novamente a *procTable* (em vez de listar a tabela inteira, caminharemos somente pela instância 2, que descreve o processo *httpd*):

```
$ snmpwalk sunserver2 public .1.3.6.1.4.1.2021.2
enterprises.ucdavis.procTable.prEntry.prIndex.1 = 1
enterprises.ucdavis.procTable.prEntry.prNames.1 = "httpd"
enterprises.ucdavis.procTable.prEntry.prMin.1 = 0
enterprises.ucdavis.procTable.prEntry.prMax.1 = 0
enterprises.ucdavis.procTable.prEntry.prCount.1 = 0
enterprises.ucdavis.procTable.prEntry.prErrorFlag.1 = 1
enterprises.ucdavis.procTable.prEntry.prErrMessage.1 = "Nenhum processo
httpd
em execução."
enterprises.ucdavis.procTable.prEntry.prErrFix.1 = 0
```

Tínhamos seis processos *httpd* em execução e agora, de acordo com *prCount*, temos um. *prErrMessage* informa o problema e *prErrorFlag* mudou de 0 to 1, indicando que algo está errado. Esse flag facilita o polling no agente, usando as técnicas discutidas no Capítulo 9, e permite verificar se os processos *httpd* foram interrompidos. Experimentemos uma variação desse tema. Se definirmos *prMin* de modo a indicar que desejamos mais de seis processos *httpd* em execução, e depois a reiniciar o *httpd*, a *prErrMessage* será:

```
enterprises.ucdavis.procTable.prEntry.prErrMessage.1 = "Número
insuficiente de
httpd em execução (# = 0)"
```

O comando seguinte no arquivo de configuração é o *exec*, que permite executar qualquer programa, retornar os resultados o programa e emitir o valor para o agente. Esse recurso é útil quando já existe um programa que você deseja

\* Quando *prMin* e *prMax* são 0, isso indica que desejamos, no mínimo, um processo e, no máximo, um número infinito de processos em execução.

utilizar em conjunto com o agente. Criamos um script de shell simples, denominado *filecheck.sh*, que verifica se existe o arquivo passado para ele na linha comando. Se o arquivo existir, retorna 0 (zero); caso contrário, retorna 1 (um):

```
#!/bin/sh
# FileName: /opt/local/shell_scripts/filecheck.sh
if [ -f $1 ]; then
    exit 0
fi
exit 1
```

Nosso arquivo de configuração usa o *filecheck.sh* para verificar a existência do arquivo */tmp/vxprint.error*. Quando o script *filecheck.sh* estiver no lugar, você poderá ver os resultados por ele retornados, ao percorrer a *ucdavis.extTable* (.1.3.6.1.4.1.2021.8):

```
$ snmpwalk sunserver2 public .1.3.6.1.4.1.2021.8
enterprises.ucdavis.extTable.extEntry.extIndex.1 = 1
enterprises.ucdavis.extTable.extEntry.extNames.1 = "FileCheck"
enterprises.ucdavis.extTable.extEntry.extCommand.1 =
"/opt/local/shell_scripts/filecheck.sh /tmp/vxprint.error"
enterprises.ucdavis.extTable.extEntry.extResult.1 = 0
enterprises.ucdavis.extTable.extEntry.extOutput.1 = ""
enterprises.ucdavis.extTable.extEntry.extErrFix.1 = 0
```

O primeiro argumento do comando\* exec no arquivo de configuração é um rótulo que identifica o comando para que possamos reconhecê-lo facilmente na *extTable*. Em nosso caso, usamos *FileCheck* – que não é um nome muito adequado, porque talvez fosse necessário verificar a presença de vários arquivos, mas poderíamos tê-lo nomeado com algo mais descritivo. Seja qual for o nome selecionado, será retornado como o valor do objeto *extTable.extEntry.extNames.1*. Como o arquivo */tmp/vxprint.error* existe, *filecheck.sh* retorna 0, que aparece na tabela como o valor de *extTable.extEntry.extResult.1*. Você também pode instruir o agente a retornar uma linha de saída do programa. Modifique o *filecheck.sh* de modo a executar um *ls -la* no arquivo, se existir:

```
#!/bin/sh
# FileName: /opt/local/shell_scripts/filecheck.sh

if [ -f $1 ]; then
    ls -la $1
    exit 0
fi

exit 1
```

\* Consulte o arquivo de configuração *EXAMPLE.conf* apresentado no início deste capítulo.

Ao pesquisar o agente, veremos a saída do script no valor de *extOutput* que o agente retorna:

```
enterprises.ucdavis.extTable.extEntry.extOutput.1 = \
" 16 -rw-r-r- 1 root      other    2476 Feb 3 17:13 \
/tmp/vxprint.error."
```

Esse truque simples só funcionará se o script retornar uma única linha de saída. Se o script retornar mais de uma linha, insira uma OID à frente do nome da string no comando exec.

Eis o próximo comando de nosso arquivo *snmpd.conf*:

```
exec .1.3.6.1.4.1.2021.2789.51 FancyCheck
/opt/local/shell_scripts/fancycheck.sh \
/core
```

Este comando executa o programa *fancycheck.sh*, com a string identificadora *FancyCheck*. Não é importante listar o *fancycheck.sh*; é semelhante ao *filecheck.sh*, exceto pelo fato de adicionar uma verificação para determinar o tipo de arquivo. A OID identifica se o agente colocará o resultado da execução do comando na árvore de MIBs, que deve estar na empresa *ucdavis* (.1.3.6.1.4.1.2021). É recomendável colocar depois da ID da empresa *ucdavis* o número de sua própria empresa, para evitar colisões com os objetos definidos por outras origens e impedir a sobregravação de uma das subárvores da *ucdavis*. Depois do número de sua empresa, coloque outro número que identifique esse comando específico. Nesse caso, a ID de nossa empresa é 2789 e atribuímos a esse comando o número 51 aleatoriamente. Assim, a OID completa é .1.3.6.1.4.1.2021.2789.51.

Eis os resultados do percurso na subárvore .1.3.6.1.4.1.2021.2789.51:

```
$ snmpwalk sunserver2 public .1.3.6.1.4.1.2021.2789.51
enterprises.ucdavis.2789.51.1.1 = 1
enterprises.ucdavis.2789.51.2.1 = "FancyCheck"
enterprises.ucdavis.2789.51.3.1 =
"/opt/local/shell_scripts/fancycheck.sh /core"
ucdavis.2789.51.100.1 = 0
ucdavis.2789.51.101.1 = "-rw-r-r- 1 root      other
346708 Feb 14 16:30 /core."
ucdavis.2789.51.101.2 = "/core:..ELF 32-bit MSB core file SPARC
Version 1, from 'httpd'.""
ucdavis.2789.51.102.1 = 0
```

Observe que existem algumas linhas adicionais na saída. 2789.51.100.1 é o número da saída, 2789.51.101.1 e 2789.51.101.2 são a saída do comando, e 2789.51.102.1 é o valor de *errorFix*. Esses valores podem ser úteis quando você tentar depurar a nova extensão. (Infelizmente, *snmpwalk* só pode oferecer a OID numérica, não um nome significativo porque o *snmpwalk* não sabe o que é 2789.51.x.)

A última tarefa do agente extensível do Net-SNMP é efetuar um monitoramento de espaço de disco. Essa é uma opção excelente, que permite verificar a disponibilidade do espaço de disco e retornar diversos valores úteis. A opção disk usa um ponto de montagem do sistema de arquivos seguido de um número. Eis a nossa entrada no *snmpd.conf*:

```
# Check disks for their mins  
disk / 100000
```

A definição da opção disk no *UCD-SNMP-MIB.txt* é “Espaço mínimo necessário em disco (em KB) antes do acionamento dos erros.” Examinemos primeiramente o *sunserver2* para descobrir o que o programa *df* comum retorna:

```
$ df -k /  
Filesystem      kbytes   used   avail capacity  Mounted on  
/dev/dsk/c0t0d0s0    432839  93449  296110    24%      /
```

Para saber o que o SNMP tem a dizer sobre o espaço de disco existente em nosso servidor, execute o *snmpwalk* sobre o objeto *ucdavis.diskTable* (.1.3.6.1.4.1.2021.9). Esse procedimento retornará praticamente as mesmas informações que o comando *df*:

```
$ snmpwalk sunserver2 public .1.3.6.1.4.1.2021.9  
enterprises.ucdavis.diskTable.dskEntry.dskIndex.1 = 1  
enterprises.ucdavis.diskTable.dskEntry.dskPath.1 = "/" Hex: 2F  
enterprises.ucdavis.diskTable.dskEntry.dskDevice.1 =  
"/dev/dsk/c0t0d0s0"  
enterprises.ucdavis.diskTable.dskEntry.dskMinimum.1 = 100000  
enterprises.ucdavis.diskTable.dskEntry.dskMinPercent.1 = -1  
enterprises.ucdavis.diskTable.dskEntry.dskTotal.1 = 432839  
enterprises.ucdavis.diskTable.dskEntry.dskAvail.1 = 296110  
enterprises.ucdavis.diskTable.dskEntry.dskUsed.1 = 93449  
enterprises.ucdavis.diskTable.dskEntry.dskPercent.1 = 24  
enterprises.ucdavis.diskTable.dskEntry.dskErrorFlag.1 = 0  
enterprises.ucdavis.diskTable.dskEntry.dskErrorMsg.1 = ""
```

Como você pode constatar, o agente do Net-SNMP tem alguns recursos personalizáveis que permitem adaptar o monitoramento sem precisar criar suas próprias definições de objeto. Revise no *\$NET\_SNMP\_HOME/share/snmp/mibs/UCD-SNMP-MIB.txt* as definições completas de todas as variáveis do Net-SNMP. Embora tenhamos abordado apenas algumas opções personalizáveis aqui, você encontrará várias outras opções úteis no arquivo *EXAMPLE.conf* que acompanha o pacote do Net-SNMP.

## SystemEDGE

O agente do SystemEDGE também é extensível. Não é necessário executar quaisquer outros processos do sistema para estender esse agente, fornecido com três

objetos estendidos predefinidos: Domain Name System (DNS) para o Unix, Network Information System (NIS) para o Unix e o Remote Pinger para o Unix e Windows NT. O primeiro objeto retorna o nome do domínio do sistema operacional básico, o segundo retorna o nome de domínio NIS do sistema operacional básico e o terceiro envia solicitações do ICMP para um host remoto no sistema em que o agente estiver em execução. Embora seja interessante ter esses scripts, desejamos destacar o modo de inclusão de suas próprias OIDs no agente.

## *Extensibilidade para o Unix e Windows NT*

O agente do SystemEDGE tem uma MIB privada que define uma tabela denominada *extensionGroup*. A OID completa é 1.3.6.1.4.1.546.14 (*iso.org.dod.internet.private.enterprises.empire.extensionGroup*). Defina seus objetos aqui. O primeiro objeto definido tem a OID *extensionGroup.1.0* (1.3.6.1.4.1.546.14.1.0), onde .0 indica que o objeto é escalar; o objeto seguinte tem a OID *extensionGroup.2.0*, e assim por diante. Observe que todos os objetos definidos dessa maneira devem ser escalares. Para os usuários avançados, a Concord desenvolveu uma estrutura de plug-ins para o SystemEDGE, que permite desenvolver objetos estendidos complexos (incluindo tabelas) e MIBs completas.

Para estender o agente, comece editando o arquivo *sysedge.cf*, que informa ao agente as OIDs estendidas a que ele deve responder. O formato de um comando nesse arquivo é:

```
extension LeafNumber Type Access 'Command'
```

A palavra-chave *extension* informa ao agente que essa entrada de configuração é uma extensão pertencente à *extensionGroup*. *LeafNumber* é o número do objeto extensão – ou seja, o número atribuído ao objeto na tabela *extensionGroup*. *Type* é o tipo de SNMP para a OID. Os tipos válidos são *Integer*, *Counter*, *Gauge*, *Octetstring*, *TimeTicks*, *Objectid* e *IPAddress*. *Access* é definido com *Read-Only* ou *Read-Write*. E por último, *Command* é o script ou o programa que o agente executará quando essa OID específica for consultada por um NMS. Discutiremos esse assunto com outros detalhes, mais adiante. Eis alguns exemplos de objetos extensão:

```
extension 1 Integer Read-Only '/usr/local/bin/Script.sh'  
extension 2 Gauge Read-Only '/usr/local/bin/Script.pl'  
extension 33 Counter Read-Write '/usr/local/bin/Program'
```

O primeiro item define uma OID somente-leitura do tipo *Integer*. A OID é 1.3.6.1.4.1.546.14.1.0. O agente executará o comando */usr/local/bin/exampleScript.sh* quando essa OID for consultada. A segunda entrada é semelhante, mas o tipo é *Gauge* e a OID numérica é 1.3.6.1.4.1.546.14.2.0. O terceiro exemplo indica apenas que *LeafNumber* não precisa ser seqüencial; você pode utilizar qualquer número, desde que exclusivo.

Estender o agente permite criar scripts exclusivos que fazer o que você ne-

porte para SNMP, desde que você escreva um script que consulte o status desses dispositivos ou programas. No exemplo acima, */usr/local/bin/Script.sh*, */usr/local/bin/Script.pl* e */usr/local/bin/Program* são exemplos de scripts que o agente executará quando a OID atribuída a cada script for consultada. O script ou programa deve atender a duas exigências:

- Todas as solicitações de *set*, *get* e *getnext* devem gerar saída. Para *get* e *getnext*, a saída do script deve ser o valor real do objeto solicitado. Isso significa que o script ou programa que busca as informações necessárias deve retornar um único valor. Para uma solicitação de *set*, o script deve retornar o novo valor do objeto. A solicitação falhará se não existir uma saída. (Observe que para uma solicitação de *set*, um script pode obter êxito ao mudar o estado do dispositivo mesmo que não gere uma saída e o agente considere que o script falhou.)
- O script ou programa deve imprimir as informações a serem retornadas (com base no tipo de solicitação), seguidas por um caractere de nova linha. O agente analisará somente até esse caractere. Se o caractere de nova linha for o primeiro encontrado, o agente gera um erro e retorna esse erro para a aplicação do NMS ou SNMP.

O agente envia três argumentos para o script ou programa que ele executa: o *LeafNumber*, o tipo de solicitação (GET, GETNEXT ou SET, com letras maiúsculas) e uma string que representa um valor a ser definido (o terceiro argumento é usado somente para as solicitações de SET). A seguinte estrutura de script de Perl, denominado *skel.pl*, mostra como utilizar os três argumentos:

```
#!/usr/local/bin/perl

if ($ARGV[0] == 1) {
    # a OID consultada é 1.3.6.1.4.1.546.14.1.0
    if ($ARGV[1] eq "SET") {
        # use $ARGV[2] para definir o valor de algo e retornar o valor definido,
        # seguido por um caractere de nova linha, para o agente
    } elsif (($ARGV[1] eq "GET") || ($ARGV[1] eq "GETNEXT")) {
        # obtenha a informação à qual essa OID pertence, depois retorne-a,
        # seguida por um caractere de nova linha, para o agente
    }
} else {
    return 0;
    # retorne 0, porque não sei o que fazer com essa OID
}
```

Tudo o que você precisa fazer é adicionar a lógica que emite alguma ação para recuperar (ou definir) o valor adequado e retornar o valor correto para o agente. A entrada correspondente no *sysedge.cf* ficaria parecida com esta:

```
extension 1 Integer Read-Write '/usr/local/bin/skel.pl'
```

O que fizemos até agora permite que o agente responda às solicitações de um novo tipo de dado. Ainda precisamos resolver a outra parte do que-

bra-cabeça: informar à estação de gerenciamento que existe um novo tipo de dado disponível para recuperação. Isso exige a criação de uma entrada em um arquivo de MIB.\* Após adicionar essa entrada ao arquivo, recompile a MIB no sistema NMS para que o NMS conheça o acesso e o tipo de cada objeto estendido na MIB para o qual deve efetuar consultas. Eis uma entrada de MIB correspondente à extensão do agente anterior:

```
skeletonVariable OBJECT-TYPE  
    SYNTAX Integer  
    ACCESS Read-Write  
    DESCRIPTION  
        "Este é um exemplo de objeto."  
    ::= { extensionGroup 1 }
```

Após compilar essa entrada no NMS, você pode consultar o objeto, especificando o nome completo, (*iso.org.dod.internet.private.enterprises.empire.extensionGroup.skeletonVariable.0*). Como alternativa, é possível utilizar a OID numérica; por exemplo:

```
$ snmpget server.ora.com public .1.3.6.1.4.1.546.14.1.0
```

A segurança pode ser um problema ao criar seus scripts de extensão. Nos sistemas Unix, convém criar um usuário e grupo separados para executar as extensões, em vez de permitir que o usuário raiz execute seus scripts.

### *Mais extensibilidade para o Windows NT*

Embora o *extensionGroup* seja aceito em todas as plataformas, a versão Windows NT do SystemEDGE permite estender o próprio SystemEDGE com objetos obtidos no registro e registro de desempenho. Você pode acessar os dados de configuração e de desempenho, normalmente visualizados no *regedit* e *perfmon*. O grupo de extensão do Windows NT é definido como *iso.org.dod.internet.private.enterprises.empire.nt.ntRegPerf* (1.3.6.1.4.1.546.5.7). Assim como acontece com as extensões do Unix, as extensões do NT são definidas no arquivo *sysedge.cf*.

Para configurar uma extensão do registro, adicione uma linha com a seguinte sintaxe ao *sysedge.cf*:

```
ntregperf LeafNumber Type Registry 'Key' 'Value'
```

A palavra-chave *ntregperf* define essa linha como um objeto extensão de desempenho do registro do NT. *LeafNumber* e *Type* são idênticos para as extensões do Unix. A palavra-chave *Registry* identifica essa entrada como uma extensão do registro. As extensões do Registro são somente leitura. *Key* é uma string

\* A Concord recomenda que você mantenha todos os objetos da MIB estendidos em arquivo separado, afastado do arquivo SystemEDGE.MIB. Esse procedimento facilita a recompilação no NMS.

entre aspas simples que especifica a chave do registro a ser acessada. *Value* é o valor que você deseja ler a partir da chave. Eis um exemplo:

```
ntregperf 1 OctetString Registry  
'SYSTEM\CurrentControlSet\Control\CrashControl' 'DumpFile'
```

Isso cria um objeto extensão de registro, que retorna o caminho para o arquivo dump de controle de quedas do sistema. A OID é 1.3.6.1.4.1.546.5.7.1.0 (*iso.org.dod.internet.private.enterprises.empire.nt.ntRegPerf.1.0*).

Para configurar uma extensão de desempenho, use a seguinte sintaxe:

```
ntregperf LeafNumber Type Performance 'Object' 'Counter' 'PerfInstance'
```

Aqui, mais uma vez, *ntregperf* é a palavra-chave que indica que este é um objeto extensão de registro/desempenho do NT. Você já deve conhecer *LeafNumber* e *Type*. A palavra-chave *Performance* indica que estamos lendo um valor do registro de desempenho; as extensões de desempenho são somente leitura. *Object* especifica o objeto desempenho a ser acessado. *Counter* representa o valor do contador de desempenho do objeto, a ser acessado. Finalmente, *PerfInstance* define a instância do contador de desempenho a ser acessada. Essa instância deve idêntica ao que é listado com *perfmon*. Eis uma extensão de desempenho característica:

```
ntregperf 2 Counter Performance 'TCP' 'Segments Sent/sec' '1'
```

Você pode utilizar essa extensão para observar o número total de segmentos do TCP transmitidos pelo sistema. Sua OID é 1.3.6.1.4.1.546.5.7.2.0 (*iso.org.dod.internet.private.enterprises.empire.nt.ntRegPerf.2.0*). Lembre-se de que você deve criar uma entrada de MIB (em um arquivo de MIB) para quaisquer extensões do NT criadas, semelhante à entrada que definimos acima para *skeletonVariable*.

Os exemplos desta seção devem ser suficientes para que você conheça um agente estendido do SystemEDGE. Leia o manual do SystemEDGE para obter uma discussão completa sobre esse tópico.

## Agente extensível do OpenView

Antes de começar a lidar com o agente extensível do OpenView, verifique se o respectivo agente principal (*snmpdm*) está configurado e em execução correta. Você também deve obter um número de empresa porque estender o agente do OpenView requer a criação de suas próprias definições de MIB e os objetos definidos por você devem fazer parte da subárvore de *empresas*.\* O Capítulo 2 descreve como obter um número de empresa.

\* Não use o número da minha empresa. A obtenção de um número de empresa é fácil e gratuita. A utilização de meu número só confundirá você e outras pessoas.

As MIBs são escritas por meio da SMI, que possui duas versões: SMIv1, definida nos RFCs 1155 e 1212; e SMIv2, definida nos RFCs 2578, 2579 e 2580. O RFC 1155 observa que “as construções do ASN.1 são usadas para definir a estrutura, embora todo o ASN.1 não seja permitido”. Mesmo que o arquivo *snmpd.extend* do agente extensível do OpenView use o ASN.1 para definir objetos, são necessárias algumas entradas adicionais para criar um objeto utilizável. O *snmpd.extend* não tem suporte para algumas construções da SMI do SNMPv2. Neste capítulo, discutiremos somente as construções aceitas.

Por default, o arquivo de configuração do agente extensível na versão Unix do OpenView é */etc/SnmpAgent.d/snmp.extend*. Para acessá-lo imediatamente, copie o exemplo de arquivo para essa localização e, em seguida, reinicie o agente:

```
$ cp /opt/0V/prg_samples/eagent/snmpd.extend /etc/SnmpAgent.d/  
$ /etc/rc2.d/S98SnmpExtAgt stop  
$ /etc/rc2.d/S98SnmpExtAgt start
```

Você não deve encontrar erros e deve obter um código de saída de 0 (zero). Se ocorrerem erros, verifique o arquivo *snmpd.log*.<sup>\*</sup> Se agente iniciar com êxito, tente percorrer um dos objetos monitorizados pelo agente extensível. O comando a seguir verifica o status da fila do correio:

```
$ snmpwalk sunserver1 .1.3.6.1.4.1.4242.2.2.0  
4242.2.2.0 : OCTET STRING- (ascii): Mail queue is empty
```

Grande início! Iniciamos e consultamos com êxito o agente extensível.

A chave para o arquivo *snmpd.extend* do OpenView é DESCRIPTION. Se você acha estranho, é mesmo! A execução de comandos dentro da seção DESCRIPTION é específica para este agente, não faz parte do design do SNMP. Esta seção informa ao agente onde deverá examinar para ler, gravar e executar arquivos. Você pode colocar diversos parâmetros dentro da seção DESCRIPTION mas usaremos apenas alguns dos mais conhecidos. Eis a sintaxe do arquivo *snmpd.extend*:

```
seu-rótulo-entra-aqui DEFINITIONS ::= BEGIN
```

— insira seus comentários aqui

```
enterprise-name OBJECT IDENTIFIER ::= { OID-label(1) OID-label(2) 3 }  
subtree-name1 OBJECT IDENTIFIER ::= { OID-label(3) 4 }  
subtree-name2 OBJECT IDENTIFIER ::= { OID-label(123) 56 }  
data-Identifier** OBJECT-TYPE  
    SYNTAX Integer | Counter | Gauge | DisplayString***  
    ACCESS read-only | read-write  
    STATUS mandatory | optional | obsolete | deprecated****
```

\* Nas máquinas do Solaris e HP-UX, este arquivo está localizado em */var/adm/snmpd.log*.

\*\* Denominados ocasionalmente como nó de folha, nó, objeto ou MIB.

\*\*\* Apenas para citar alguns tipos de dados.

\*\*\*\* Por enquanto, usaremos sempre mandatory como STATUS.

```

DESCRIPTION
"
  Digite sua descrição aqui
READ-COMMAND: /your/command/here passed1 passed2
  READ-COMMAND-TIMEOUT: timeout_in_seconds (defaults to 3)
  FILE-COMMAND: /your/file-command/here passed1 passed2
  FILE-COMMAND-FREQUENCY: frequency_in_seconds (defaults to 10)
  FILE-NAME: /your/filename/here
"
 ::= { parent-subtree-name subidentifier }
END

```

Podemos reunir algumas diretrizes de estilo do RFC 2578. Embora existam diversas diretrizes, algumas mais úteis do que outras, um aspecto é importante: o tamanho da letra é realmente relevante. Grande parte do ASN.1 distingue maiúsculas de minúsculas. Todas as palavras-chave e macros do ASN.1 devem ter letras maiúsculas: OBJECT-TYPE, SYNTAX, DESCRIPTION, etc. Os *data-Identifiers* (por exemplo, nomes de objetos) devem começar com letras minúsculas e não podem conter espaços. Se você leu algumas das MIBs do RFC ou fez alguma polling, deve ter observado que todos os nomes de objetos seguem esta convenção. Procure utilizar nomes descritivos e mantenha os nomes dentro do limite de 64 caracteres; o RFC 2578 afirma que tudo o que tiver mais de 32 caracteres não é recomendável. Se você definir um objeto em uma subárvore existente, deve utilizar o nome desta subárvore ou o nome da árvore-pai, antes do nome de cada objeto novo criado. A subárvore *ip* na *mib-2* (RFC 1213) fornece um exemplo prático:

```

ip          OBJECT IDENTIFIER ::= { mib-2 4 }

ipForwarding OBJECT-TYPE
...
 ::= { ip 1 }

ipDefaultTTL OBJECT-TYPE
...
 ::= { ip 2 }

```

Este arquivo começa definindo a subárvore *ip*. Os nomes de objetos dentro dessa subárvore começam com *ip* e usam *ip* como nome da subárvore-pai. Embora essa prática recomendada seja útil, é inadequada em algumas situações. Por exemplo, essa prática dificulta a movimentação de objetos para outros pais ao construir um arquivo MIB.

Eis um arquivo de trabalho *snmpd.extend* que contém três definições: *psZombieNum*, *prtDiagExitC* e *whosOnCall*. Coloque todos esses objetos dentro de minha empresa privada (2789, que denominei de *mauro*). A Figura 11-2 mostra essa parte de minha subárvore privada.

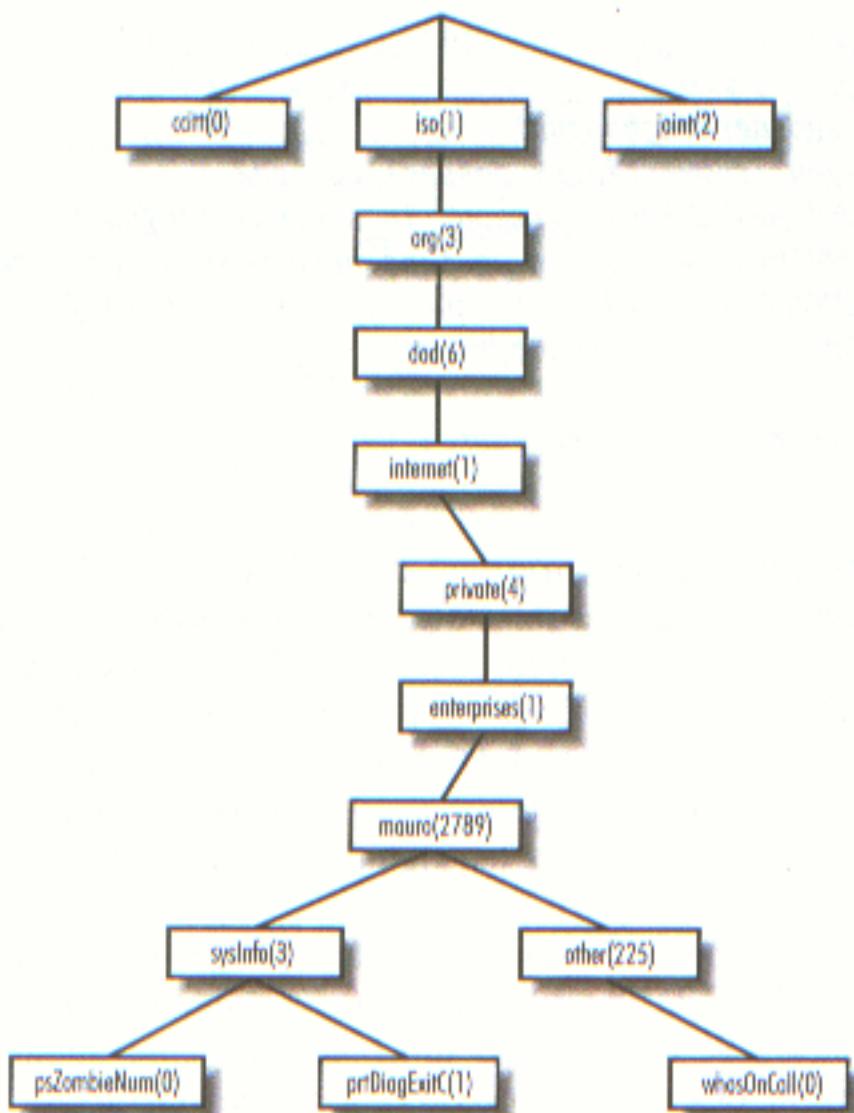


Figura 11-2 Subárvore *mauro*

Agora, é possível percorrer a árvore e constatar a aparência de meus objetos; minha árvore inicia na OID `.1.3.6.1.4.1.2789`, o que equivale a `.iso.org.dod.internet.private.enterprises.mauro`. Posso organizar minha subárvore como eu quiser, de modo que eu a dividi em duas ramificações abaixo de *mauro*: *mauro.sysInfo* (2789.3) armazenará informações sobre o status do sistema em si (*psZombieNum* e *prtDiagExitC*) e *mauro.other* (2789.255) guardará outras informações (*whosOnCall*). Ao examinar mais abaixo, você encontrará os três nós de folha que definimos neste arquivo:

SampleExt DEFINITIONS ::= BEGIN

– Os comentários entram aqui, depois dos traços

```

internet      OBJECT IDENTIFIER ::= { iso{1} org{3} dod{6} 1 }
enterprises   OBJECT IDENTIFIER ::= { internet{1} private{4} 1 }
mauro         OBJECT IDENTIFIER ::= { enterprises{1} 2789 }
  
```

– Após definir *mauro*, definiremos alguns objetos

```

sysInfo       OBJECT IDENTIFIER ::= { mauro 3 }
other         OBJECT IDENTIFIER ::= { mauro 255 }
  
```

```

psZombieNum OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS   mandatory
    DESCRIPTION
        "Percorra ps e retorne o número de cadáveres."
        READ-COMMAND: VALUE=`ps -ef | grep -v grep | grep -c \<defunct\>`; echo $VALUE
    ::= { sysInfo 0 }

prtDiagExitC OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS   mandatory
    DESCRIPTION
        "No Solaris, prtdiag exibe informações de diagnóstico do sistema. A manpage informa que se a saída desse comando for um valor diferente de zero, teremos um problema. Este é um excelente mecanismo de polling para alguns sistemas."
        READ-COMMAND: /usr/platform/`uname -m`/sbin/prtdiag > /dev/null; echo $??
    ::= { sysInfo 1 }

whosOnCall OBJECT-TYPE
    SYNTAX  OctetString
    ACCESS  read-write
    STATUS   mandatory
    DESCRIPTION
        "Este arquivo contém o nome da pessoa que estará na chamada de hoje. A helpdesk usa este arquivo. Somente a helpdesk e os gerentes devem atualizá-lo. Se você adoecer ou não puder comparecer à chamada, entre em contato com seu gerente e/ou com a helpdesk.

        FILE-NAME: /opt/local/oncall/today.txt"
    ::= { other 0 }

END

```

Os dois primeiros objetos, *psZombieNum* e *prtDiagExitC*, usam READ-COMMAND na DESCRIPTION. Isso instrui o agente a executar o comando indicado e a enviar a saída gerada pelo comando para o NMS. Por default, o programa deve encerrar em três segundos e deve ter o valor da saída igual a 0 (zero). É possível aumentar o timeout adicionando um READ-COMMAND-TIMEOUT:

```

READ-COMMAND: /some/fs/somecommand.pl
READ-COMMAND-TIMEOUT: 10

```

Isso instrui o agente a aguardar a resposta durante 10 segundos em vez de 3, antes de eliminar o processo e retornar um erro.

O último objeto, *whosOnCall*, usa um FILE-NAME na DESCRIPTION. Isso instrui o agente a retornar a primeira linha do arquivo, programa, script, etc. especificada após FILE-NAME. Mais adiante, aprenderemos a lidar com esse arquivo.

Após criarmos um arquivo MIB com as novas definições, é necessário carregar a nova MIB no OpenView. Essa etapa não é estritamente necessária mas é muito mais prático trabalhar com nomes textuais do que lidar com IDs alfanuméricas. Para isso, use *xnmloadmib*, discutido no Capítulo 6. Após carregar o arquivo da MIB que contém nossos três objetos novos, deveremos ver os nomes desses objetos no browser da MIB e poderemos consultá-los por nome.

Depois que você copiar o arquivo da MIB para o diretório adequado e instruiu o agente extensível, *extsubagt*, a reler a respectiva configuração (usando *kill -HUP*), tente percorrer os novos objetos com o programa *snmpwalk* do OpenView:

```
$ snmpwalk sunserver2 -c public .1.3.6.1.4.1.2789  
auro.sysInfo.psZombieNum.0 : INTEGER: 0  
mauro.sysInfo.prtDiagExitC.0 : INTEGER: 2
```

Você observou algo estranho em relação aos valores de retorno? Não obtivemos nada para *whosOnCall*. Nada foi retornado para esse objeto porque não criamos o arquivo *oncall.txt* cujo conteúdo estamos tentando ler. Primeiramente, devemos criá-lo e inserir alguns dados nesse arquivo. Existem dois métodos para fazer isso. Evidentemente, é possível criar esse arquivo com o editor de texto favorito. Mas o método inteligente é utilizar o *snmpset*:

```
$ snmpset -c private sunserver2 \  
.1.3.6.1.4.1.2789.255.0.0 octetstring "david jones"  
mauro.Other.whosOnCall.0 : OCTET STRING- (ascii): david jones
```

Esse comando instrui o agente do SNMP a colocar david jones no arquivo */opt/local/oncall/today.txt*. O nome do arquivo é definido pelo comando FILE-NAME: */opt/local/oncall/today.txt* que escrevemos na MIB estendida.

O .0 adicional, no final da OID, informa ao agente que desejamos a primeira (e única) instância de *whosOnCall*. (Poderíamos ter utilizado *.iso.org.dod.internet.private.enterprises.mauro.other.whosOnCall.0* em vez da OID numérica.) Além disso, o comando *snmpset* especifica o tipo de dado *octetstring*, o que corresponde à sintaxe de *OctetString* que definimos na MIB. Esse tipo de dado permite inserir valores de string no arquivo. Finalmente, podemos definir o valor desse objeto com *snmpset* porque temos acesso *read-write* (leitura-gravação) ao objeto, conforme especificado na MIB.

Se você preferir utilizar um editor para criar o arquivo, lembre-se de que tudo o que vier depois da primeira linha desse arquivo será ignorado. Para ler várias linhas, é necessário utilizar uma tabela; as tabelas serão discutidas na próxima seção.

Adicionemos agora outro objeto à MIB para o agente estendido. Usaremos uma alteração do exemplo que o OpenView nos oferece. Criaremos um objeto denominado *fmailListMsgs* (2) que resume as mensagens da fila do correio. Esse objeto residirá em uma nova subárvore, denominada *fmail* (4), sob a subárvore privada *mauro*. Portanto, o nome de nosso objeto será *mauro.fmail.fmailListMsgs* ou, na forma numérica, *.1.3.6.1.4.1.2789.4.2*. Primeiro, precisamos

definir a ramificação *fmail* na subárvore *mauro*. Para isso, adicione a seguinte linha ao *snmpd.extend*:

```
fmail      OBJECT IDENTIFIER ::= { mauro 4 }
```

Selecionamos 4 para o número da ramificação, mas podíamos ter escolhido qualquer número que não entrasse em conflito com outras ramificações (3 e 255). Após definirmos *fmail*, poderemos inserir a definição para *fmailListMsgs* no *snmpd.extend*, posicionando-a antes da instrução END:

```
fmailListMsgs OBJECT-TYPE
    SYNTAX DisplayString
    ACCESS read-only
    STATUS mandatory
    DESCRIPTION
        "Lista de mensagens da fila de correio.
        READ-COMMAND: /usr/lib/sendmail -bp
        READ-COMMAND-TIMEOUT: 10"
    ::= { fmail 2 }
```

Quando consultada, *fmailListMsgs* executa o comando *sendmail -bp*, que imprime um resumo da fila do correio. Depois que tudo isso for feito, você pode utilizar sua estação de gerenciamento ou uma ferramenta como *snmpget* para ler o valor de *mauro.fmail.fmailListMsgs* e ver o status da fila do correio de saída.

## Tabelas

As tabelas permitem que o agente retorne várias linhas de saída (ou outros conjuntos de valores) do comando executado. Em sua elaboração máxima, uma tabela permite que o agente retorne algo como uma planilha. É possível recuperar essa planilha por meio do *snmpwalk* – um processo muito mais fácil do que emitir operações de get separadas para recuperar um valor dos dados de cada vez. Já vimos a tabela *.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable*, definida na MIB-II, que contém informações sobre todas as interfaces de um dispositivo.

Toda tabela contém um *índice de inteiros*, que é uma chave exclusiva que distingue as linhas da tabela. O índice começa com 1, para a primeira linha e aumenta uma (1) unidade a cada linha seguinte. O índice é usado como um identificador de instância para as colunas da tabela; especificada a coluna, o índice permite selecionar os dados (ou seja, a linha) necessários. Examinemos uma tabela pequena, representada pelo arquivo de texto, *animal.db*:

1	Tweety	Bird	Chirp	2
2	Madison	Dog	Bark	4
3	"Big Ben"	Bear	Grrr	5

Nosso objetivo é tornar essa tabela legível por meio do SNMP, usando o agente extensível do OpenView. Este arquivo já está no formato exigido pelo agente. Cada coluna é delimitada por espaço em branco; uma nova linha marca

o final de cada linha. Os dados que incluem um espaço interno estão entre aspas. O OpenView não permite títulos de coluna na tabela, mas gostaríamos de pensar nos nomes dos objetos em cada linha. Logicamente, os títulos de coluna não são nada mais do que os nomes dos objetos que recuperaremos na tabela. Em outras palavras, cada linha de nossa tabela consiste em cinco objetos:

#### *animalIndex*

Um índice que especifica a linha na tabela. A primeira linha é 1, conforme previsto para as tabelas do SNMP. Portanto, a SYNTAX deste objeto é INTEGER.

#### *animalName*

O nome do animal. É uma string de texto, de modo que a SYNTAX deste objeto será *DisplayString*.

#### *animalSpecies*

A espécie do animal (outra string de texto representada como um *DisplayString*).

#### *animalNoise*

O som emitido pelo animal (outro *DisplayString*).

#### *animalDanger*

Uma indicação do grau de periculosidade do animal. É outro INTEGER, cujo valor pode ser de 1 a 6, denominado um “íntero numerado”; é possível atribuir siglas textuais aos valores inteiros.

Nessa etapa, temos quase tudo o que precisamos saber para criar a MIB que nos permite ler a tabela. Por exemplo, sabemos que desejamos um objeto denominado *animalNoise.2* para acessar o objeto *animalNoise* na segunda linha da tabela; esse objeto tem o valor Bark. É fácil perceber como essa notação pode ser utilizada para localizar qualquer objeto existente na tabela. Agora, vamos escrever a definição da MIB para a tabela.

```
TableExtExample DEFINITIONS ::= BEGIN
```

```
internet      OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) 1 }
enterprises   OBJECT IDENTIFIER ::= { internet(1) private(4) 1 }
mauro         OBJECT IDENTIFIER ::= { enterprises(1) 2789 }
other          OBJECT IDENTIFIER ::= { mauro 255 }
```

```
AnimalEntry ::=
```

```
SEQUENCE {
  animalIndex INTEGER,
  animalName DisplayString,
  animalSpecies DisplayString,
  animalNoise DisplayString,
```

```
animalDanger INTEGER
}

animalTable OBJECT-TYPE
SYNTAX SEQUENCE OF AnimalEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
"Esta é uma tabela de animais que apresenta:
Nome
Espécie
Som
Grau de periculosidade
FILE-NOME: /opt/local/animal.db"
::= { other 247 }

animalEntry OBJECT-TYPE
SYNTAX AnimalEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
"Lista de animalNum"
INDEX { animalIndex }
::= { animalTable 1 }

animalIndex OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION
"Número de índice único que usaremos para cada linha"
::= { animalEntry 1 }

animalName OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
"O apelido de cada animalzinho"
::= { animalEntry 2 }

animalSpecies OBJECT-TYPE
SYNTAX DisplayString
ACCESS read-only
STATUS mandatory
DESCRIPTION
"A espécie do animal"
::= { animalEntry 3 }
```

```

animalNoise OBJECT-TYPE
  SYNTAX DisplayString
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "O ruído ou som emitido pelo animal"
  ::= { animalEntry 4 }

animalDanger OBJECT-TYPE
  SYNTAX INTEGER {
    no-Danger(1),
    can-Harm(2),
    some-Damage(3),
    will-Wound(4),
    severe-Pain(5),
    will-Kill(6)
  }
  ACCESS read-write
  STATUS mandatory
  DESCRIPTION
    "O grau de periculosidade de cada animal específico"
  ::= { animalEntry 5 }

END

```

A tabela começa com uma definição do objeto *animalTable*, que dá a DESCRIPTION e informa ao agente onde o arquivo *animal.db* está localizado. A SYNTAX é SEQUENCE OF *AnimalEntry*. *AnimalEntry* (fique atento ao tamanho da letra) oferece uma visão sucinta de todas as colunas. Você pode deixar de fora *AnimalEntry* mas é recomendável incluí-la porque documenta a estrutura da tabela.

Na realidade, a tabela é formada com base nos elementos de *animalEntry* – como os nomes dos objetos distinguem maiúsculas de minúsculas, este objeto é diferente de *AnimalEntry*. *animalEntry* informa qual objeto devemos usar no índice ou na chave; o objeto utilizado como chave está entre chaves depois da palavra-chave INDEX.

As definições dos demais objetos são parecidas com aquelas que já vimos. A subárvore-pai de todos esses objetos é *animalEntry*, que efetivamente gera uma linha de tabela a partir de cada um desses objetos. O único objeto particularmente interessante é o *animalDanger*, que usa uma extensão do tipo de dado INTEGER. Como observado anteriormente, este objeto é um inteiro numerado, que permite associar rótulos de texto a valores inteiros.

Os valores que você pode utilizar em um tipo numerado devem ser uma sequência de inteiros consecutivos, começando com 1.\* Por exemplo, o objeto *animalDanger* define seis valores, variando de 1 a 6, com strings como no-danger associadas aos valores.

---

\* Alguns compiladores de MIB compatíveis com a SMI do SNMPv1 não permitirão um tipo numerado com 0 (zero).

Você pode salvar essa definição de tabela em um arquivo e usar o comando `xnmloadmib` para carregá-la no OpenView. Após carregá-la e criar o arquivo `animal.db` em um editor de texto, você pode percorrer a tabela:

```
$ snmpwalk sunserver1 .1.3.6.1.4.1.mauro.other.animalTable
animalEntry.animalIndex.1 : INTEGER: 1
animalEntry.animalIndex.2 : INTEGER: 2
animalEntry.animalIndex.3 : INTEGER: 3
animalEntry.animalName.1 : DISPLAY STRING-(ascii): Tweety
animalEntry.animalName.2 : DISPLAY STRING-(ascii): Madison
animalEntry.animalName.3 : DISPLAY STRING-(ascii): Big Ben
animalEntry.animalSpecies.1 : DISPLAY STRING-(ascii): Bird
animalEntry.animalSpecies.2 : DISPLAY STRING-(ascii): Dog
animalEntry.animalSpecies.3 : DISPLAY STRING-(ascii): Bear
animalEntry.animalNoise.1 : DISPLAY STRING-(ascii): Chirp
animalEntry.animalNoise.2 : DISPLAY STRING-(ascii): Bark
animalEntry.animalNoise.3 : DISPLAY STRING-(ascii): Grrr
animalEntry.animalDanger.1 : INTEGER: can-Harm
animalEntry.animalDanger.2 : INTEGER: will-Wound
animalEntry.animalDanger.3 : INTEGER: severe-Pain
```

`snmpwalk` percorre uma coluna da tabela por vez, informando todos os dados contidos em uma coluna antes de passar para a seguinte. Esse procedimento é confuso – seria mais fácil se o `snmpwalk` lesse uma linha da tabela de cada vez. Na forma atual, é necessário saltar de linha em linha ao tentar ler uma linha; por exemplo, para descobrir tudo o que existe sobre o animal de nome Tweety, você deve examinar toda terceira linha (todos os itens .1) na saída.

Compensa observar outros dois aspectos na saída do `snmpwalk`. O primeiro conjunto de valores que o `snmpwalk` informa são os valores de índice (`animalIndex`). Em seguida, ele anexa cada valor de índice a cada OID para fazer o restante do percurso. Em segundo lugar, a saída de `animalDanger` apresenta strings, como `can-Harm`, em vez de inteiros. A conversão de inteiros em strings ocorre porque definimos o objeto `animalDanger` como um inteiro numerado, o que associa um conjunto de valores possíveis a strings.

Evidentemente, a simples leitura de uma tabela não resolver tudo. Suponhamos que precisássemos atualizar periodicamente esse arquivo de modo a refletir as mudanças ocorridas no comportamento dos animais. O objeto `animalDanger` tem ACCESS (acesso) read-write (de leitura-gravação), o que permite definir o respectivo valor e atualizar o arquivo do banco de dados usando as ferramentas do SNMP. Imagine que o cachorro na linha 2 se torne muito sórdido. Precisaremos mudar o grau de periculosidade para 5 (severe-Pain). Poderíamos editar o arquivo manualmente mas é mais fácil emitir um `snmpset`:

```
$ snmpset -c private sunserver2 \
mauro.other.animalTable.animalEntry.animalDanger.2 integer "5"
mauro.other.animalTable.animalEntry.animalDanger.2 : INTEGER: severe-Pain
```

Agora, voltemos e verifiquemos se a variável foi atualizada:<sup>\*</sup>

```
$ snmpget sunserver2 \
mauro.other.animalTable.animalEntry.animalDanger.2
mauro.other.animalTable.animalEntry.animalDanger.2 : INTEGER; severe-Pain
```

Quando o *snmpset* estiver concluído, verifique o que mudou no arquivo. Além de modificar o grau de periculosidade do cachorro, ele colocou todas as strings entre aspas:

```
1 "Tweety" "Bird" "Chirp" 2
2 "Madison" "Dog" "Bark" 5
3 "Big Ben" "Bear" "Grrr" 5
```

Existem outras possibilidades de manter o arquivo atualizado. Por exemplo, você poderia utilizar um programa ou aplicativo do sistema para editar esse arquivo. Um job *cron* poderia ser acionado a cada hora e atualizar o arquivo. Essa estratégia permitiria gerar o arquivo usando uma consulta SQL a um banco de dados como o Oracle. Depois, você poderia colocar os resultados da consulta em um arquivo e pesquisar o arquivo com o SNMP para ler os resultados. Um problema nessa estratégia é a necessidade de assegurar a sincronia entre o aplicativo e os períodos de polling do SNMP. Certifique-se de pesquisar o arquivo depois que o Oracle o atualizar; caso contrário, você estará visualizando dados antigos.

Um método eficiente de assegurar que o arquivo esteja atualizado ao lê-lo é utilizar FILE-COMMAND dentro da definição da tabela. Isso instrui o agente a executar um programa que atualize a tabela antes de retornar quaisquer valores. Vamos supor que escrevemos um script denominado *get\_animal\_status.pl* que determina o status dos animais e atualiza o banco de dados adequadamente. Examine a seguir como integraríamos esse script na definição da tabela:

```
animalTable OBJECT-TYPE
    SYNTAX   SEQUENCE OF AnimalEntry
    ACCESS   not-accessible
    STATUS   mandatory
    DESCRIPTION
        "Esta é uma tabela de animais que apresenta:
         Nome
         Espécie
         Som
         Grau de periculosidade
         FILE-COMMAND: /opt/local/get_animal_status.pl
         FILE-NAME: /opt/local/animal.db"
        ::= { other 247 }
```

\* Já poderíamos ter concluído que o *set* obteve êxito quando o *snmpset* não retornou um erro. Entretanto, esse exemplo mostra, na realidade, como é possível obter uma única instância em uma tabela com um *snmpget*.

O comando deve encerrar em 10 segundos ou o agente eliminará o processo e retornará os valores antigos da tabela. Por default, o agente só executa o programa especificado por FILE-COMMAND se não tiver recebido uma solicitação nos últimos 10 segundos. Por exemplo, vamos supor que você emita dois comandos *snmpget*, dois segundos depois. Para o primeiro *snmpget*, o agente executa o programa e retorna os dados da tabela com as modificações efetuadas. Na segunda vez, o agente não executará o programa para atualizar os dados – ele retornará os dados antigos, pressupondo que nada foi alterado. Esse procedimento é efetivamente uma operação de cache. Você pode aumentar o período de tempo em que o agente mantém o respectivo cache, especificando um valor em segundos, após FILE-COMMAND-FREQUENCY. Por exemplo, para atualizar o arquivo apenas a cada 20 minutos (no máximo), inclua os seguintes comandos na definição de sua tabela:

```
FILE-COMMAND: /opt/local/get_animal_status.pl  
FILE-COMMAND-FREQUENCY: 1200  
FILE-NAME: /opt/local/animal.db"
```

Este capítulo apresentou uma introdução resumida aos três agentes extensíveis do SNMP mais conhecidos no mercado. Embora um exame mais abrangente de cada opção de configuração de cada agente esteja além do âmbito deste capítulo, ajuda saber utilizar os agentes extensíveis. Com o agente extensível, as possibilidades são praticamente infinitas.

# 12

## *Adaptando o SNMP a seu ambiente*

O SNMP pode facilitar muito sua vida de administrador de sistema, ao executar diversas tarefas que você teria que fazer manualmente ou de forma automática, escrevendo um script inteligente. É relativamente fácil cuidar das tarefas cotidianas de monitoramento do sistema: o SNMP pode verificar a utilização do espaço de disco, avisar quando os espelhos estiverem sincronizando ou registrar quem está fazendo login ou logout no sistema. Os scripts do SNMP deste capítulo representam apenas alguns dos aspectos que o SNMP permite fazer; use-os como um trampolim para suas próprias idéias.

### *Programa de criação de traps gerais*

O Capítulo 10 continha alguns scripts para reunir informações do SNMP usando o Perl, o programa *snmptrap* do OpenView e algumas outras ferramentas. Examine a seguir como utilizamos o *snmptrap* para gerar uma trap fornecendo informações sobre alguns problemas ocorrido com o banco de dados:

```
$ /opt/0V/bin/snmptrap -c public nms .1.3.6.1.4.1.2789.2500 "" 6 3003 "" \  
.1.3.6.1.4.1.2500.3003.1 octetstringascii "Oracle" \  
.1.3.6.1.4.1.2500.3003.2 octetstringascii "Backup Not Running" \  
.1.3.6.1.4.1.2500.3003.3 octetstringascii "Call the DBA Now for Help"
```

O modo de envio de uma trap no Perl é um pouco mais complexo mas ainda é fácil de utilizar:

```
#!/usr/local/bin/perl  
# Filename: /opt/local/perl_scripts/snmptrap.pl  
  
use SNMP_util "0.54"; # Isso carregará BER e SNMP_Session  
  
snmptrap("public@nms:162", ".1.3.6.1.4.1.2789", "sunserver1",  
         6, 1247, ".1.3.6.1.4.1.2789.1247.1", "int", "2448816");
```

Neste capítulo, não examinaremos minuciosamente o modo como escrever comandos como esses, mas como utilizá-los de forma inteligente. Poderíamos incluir comandos como semelhantes em arquivos de inicialização ou chama-los por meio de ganchos para outros programas. Começaremos escrevendo um código que registra logins com êxito.

## Quem está entrando em minha máquina? (I-Am-in)

Quando os usuários do Unix efetuam login, o sistema executa automaticamente um perfil; para os usuários das shells Bourne, Korn ou *bash*, o perfil no nível do sistema é denominado */etc/profile*. Existe um arquivo semelhante para os usuários do *csh* e *tcsh* (*/etc/login*). Podemos utilizar o SNMP para registrar logins, adicionando uma trap a esses perfis. Isoladamente, isso não é tão interessante porque o Unix já mantém um registro de logins de usuários. Entretanto, vamos supor que você esteja monitorizando algumas dezenas de máquinas e não queira verificar o log de cada máquina. A inclusão de uma trap no perfil no nível do sistema permite monitorizar os logins em todos os seus sistemas, em um único local. Também torna seu login mais seguro. Não é tão difícil para um usuário inteligente eliminar o arquivo *utmp* que armazena os registros de login do Unix. Se você utilizar o SNMP para fazer o login, armazenará as informações em outro host, sobre o qual você deve ter mais controle.\*

Para gerar a trap, chame o programa externo */opt/local/mib\_programs/os/iamin* em */etc/profile* (você pode chamar o mesmo programa dentro de */etc/login*). Eis o código do *iamin*:

```
#!/usr/local/bin/perl
#
# Filename: /opt/local/mib_programs/os/iamin
#
chomp ($WHO = `~/bin/who am i \| awk \{\'print \$1\'\}`);
exit 123 unless ($WHO ne '');

chomp ($WHOAMI = `/usr/ucb/whoami`);
chomp ($TTY = `/bin/tty`);
chomp ($FROM = `/bin/last -1 $WHO \| /bin/awk \{\'print \$3\'\}`);
if ($FROM =~ /Sun|Mon|Tue|Wed|Thu|Fri|Sat/) { $FROM = "N/A"; }

# DEBUG BELOW
# print "WHO :$WHO:\n"; print "WHOAMI :$WHOAMI:\n"; print "FROM :$FROM:\n";
if ("$WHOAMI" ne "$WHO") { $WHO = "$WHO\-\>$WHOAMI"; }

# Enviando uma trap usando o Net-SNMP
#
```

\* Sim, um usuário inteligente poderia interceptar e modificar pacotes de SNMP, ou reescrever o perfil da shell, ou executar inúmeras ações para destruir o registro; queremos apenas dificultar as coisas.

```

system "/usr/local/bin/snmptrap nms public .1.3.6.1.4.1.2789.2500 \" 6 1502 \""
    .1.3.6.1.4.1.2789.2500.1502.1 s \"$WHO\""
    .1.3.6.1.4.1.2789.2500.1502.2 s \"$FROM\""
    .1.3.6.1.4.1.2789.2500.1502.3 s \"$TTY\"";
#
# Enviando uma trap com a Perl
#
#use SNMP_util "0.54"; # Isso carregará BER e SNMP_Session
#snmptrap("public@nms:162", ".1.3.6.1.4.1.2789.2500", mylocalhostname, 6, 1502,
#"1.3.6.1.4.1.2789.2500.1502.1", "string", "$WHO",
#"1.3.6.1.4.1.2789.2500.1502.2", "string", "$FROM",
#"1.3.6.1.4.1.2789.2500.1502.3", "string", "$TTY");
#
# Enviando uma trap com o snmptrap do OpenView
#
#system "/opt/OV/bin/snmptrap -c public nms .1.3.6.1.4.1.2789.2500 \"\" 6 1502 \"\""
#"1.3.6.1.4.1.2789.2500.1502.1 octetstringascii \"$WHO\""
#"1.3.6.1.4.1.2789.2500.1502.2 octetstringascii \"$FROM\""
#"1.3.6.1.4.1.2789.2500.1502.3 octetstringascii \"$TTY\"";
#
#
#
print "\n#####\n";
print "# NOTICE \n# - You have been logged: :$WHO: :$FROM: :$TTY: \n"; #
print "#####\n\n";

```

Este script é um pouco mais volumoso do que esperávamos porque precisamos eliminar algumas entradas falsas. Por exemplo, alguns programas são executados dentro de uma shell, de modo que chamam os mesmos perfis de shell. Por conseguinte, é necessário adivinhar se o perfil está sendo chamado por um usuário humano; em caso negativo, encerraremos.<sup>4</sup> A etapa seguinte é conhecer mais detalhes sobre a identidade do usuário; por exemplo, o local em que o usuário está fazendo login e qual a real identidade desse usuário – não desejamos ficar confusos com alguém que está usando *su* para mudar para outra identidade. A terceira parte do programa envia a trap com todas as informações recém-encontradas (quem é o usuário, o host de login e em que TTY o usuário se encontra). Incluímos um código de geração de traps por meio dos utilitários do Net-SNMP, o módulo Perl interno e com os utilitários do OpenView. Faça sua escolha e use a versão que você conhece mais. A última parte desse programa informa ao usuário que ele entrou no sistema.

Esse script não está totalmente sem problemas. O usuário sempre pode interromper o script antes de seu término, ignorando o login. Você pode contar essa tentativa usando a trap(1), que responde a sinais diferentes. Isso instrui o usuário a concluir o programa, em vez de parar no meio do fluxo. Essa estratégia cria problemas próprios, porque o usuário raiz não possui um método para ignorar a verificação. Por um lado, isso é bom: queremos ser particularmente cuidadosos quanto aos logins da raiz.

---

<sup>4</sup> Isso também falhará se o usuário estiver usando o *su* para outro usuário. Em um ambiente bem elaborado, os usuários realmente não devem precisar usar o *su* tão freqüentemente – usar o *sudo* ou elaborar grupos adequados deve reduzir consideravelmente a necessidade do *su*.

Mas o que acontece se você estiver tentando investigar um problema de falha de rede ou de DNS? Nesse caso, o script ficará travado enquanto o DNS tenta examinar o host em que você está fazendo login. Isso pode ser decepcionante. Antes de implementar um script como esse, examine o ambiente e especifique os perfis que você deve bloquear.

É possível utilizar qualquer um dos pacotes de receber traps para detectar as traps geradas por este programa.

## Eliminando arquivos do núcleo

Geralmente, os programas deixam os dumps de memória para trás. Um arquivo básico (do núcleo) contém todas as informações de processos, pertinentes à depuração. Geralmente, esse arquivo é escrito quando um programa é encerrado de modo anormal. Embora existam métodos para limitar o tamanho de um dump ou para impedir totalmente dumps de memória, em algumas situações esses dumps são necessários temporariamente. Por conseguinte, a maioria dos sistemas Unix tem algum tipo de script *cron* que procura automaticamente os arquivos básicos e os elimina. Adicionemos alguma inteligência a esses scripts para rastrear os arquivos encontrados, os respectivos tamanhos e os nomes dos processos que os criaram.

O programa Perl a seguir está dividido em quatro partes: procura um arquivo com um nome específico (definido como o nome *core*), obtém os dados estatísticos do arquivo, elimina esse arquivo\* e, em seguida, envia uma trap. A maior parte do processamento é executada internamente pela Perl, mas utilizamos o comando *ls -l \$FILENAME* para incluir a informação do arquivo básico pertinente dentro da trap do SNMP. Esse comando permite que nossos operadores vejam informações sobre o arquivo em um formato fácil de reconhecer. Também usamos o comando *file*, que determina o tipo de um arquivo e seu autor. A menos que você saiba quem criou o arquivo, não poderá corrigir o problema.

```
#!/usr/local/bin/perl

# Procura e exclui arquivos do núcleo. Ao terminar, envia traps e
# os erros. Os argumentos são:
# -path directory : pesquisa diretório (e subdiretórios); default /
# -lookfor filename : nome do arquivo a ser pesquisado; default core
# -debug value     : nível de depuração

while ($ARGV[0] =~ /^-/)
{
    if ($ARGV[0] eq "-path") { shift; $PATH      = $ARGV[0]; }
    elsif ($ARGV[0] eq "-lookfor") { shift; $LOOKFOR   = $ARGV[0]; }
    elsif ($ARGV[0] eq "-debug") { shift; $DEBUG     = $ARGV[0]; }
```

\* Antes de começar a excluir arquivos básicos (do núcleo), você deve descobrir quem e o que os está criando, e verificar se o autor precisa desses arquivos. Em alguns casos, um arquivo básico pode ser o único meio de depuração.

```

    shift;
}
#####
##### Begin Main #####
#####
require "find.pl";      # Isso nos concede a função de busca.

$LOOKFOR = "core" unless ($LOOKFOR); # Se não tivermos alguma coisa
# em $LOOKFOR, default para core

$PATH    = "/"    unless ($PATH);      # Usarmos / se não obtivermos
# um na linha de comando

(-d $PATH) || die "$PATH is NOT a valid dir!"; # Poderemos pesquisar
# somente os
# diretórios válidos

&find("$PATH");

#####
##### Begin SubRoutines #####
#####

sub wanted
{
    if (/^$LOOKFOR$/)
    {
        if (!(-d $name)) # Saltar os diretórios denominados core
        {
            &get_stats;
            &can_file;
            &send_trap;
        }
    }
}

sub can_file
{
    print "Deleting :$_: :$name:\n" unless (!$DEBUG);
    $RES = unlink "$name";
    if ($RES != 1) { $ERROR = 1; }
}

sub get_stats
{
    chop ($STATS = `ls -l $name`);           `
    chop ($FILE_STATS = `/bin/file $name`);

    $STATS =~ s/\s+/ /g;
    $FILE_STATS =~ s/\s+/ /g;
}

sub send_trap
{
}

```

```

{
    if ($ERROR == 0) { $SPEC = 1535; }
    else             { $SPEC = 1536; }
    print "STATS: $STATS\n" unless (!$DEBUG);
    print "FILE_STATS: $FILE_STATS\n" unless (!$DEBUG);

# Enviando uma trap usando o Net-SNMP
#
#system "/usr/local/bin/snmptrap nms public .1.3.6.1.4.1.2789.2500 ' 6 $SPEC "
#.1.3.6.1.4.1.2789.2500.1535.1 s \"\$name\"
#.1.3.6.1.4.1.2789.2500.1535.2 s \"\$STATS\"
#.1.3.6.1.4.1.2789.2500.1535.3 s \"\$FILE_STATS\";

# Enviando uma trap usando o Perl
#
use SNMP_util "0.54"; # Isso carregará BER e SNMP_Session
snmptrap("public@nms:162", ".1.3.6.1.4.1.2789.2500", mylocalhostname, 6, $SPEC,
".1.3.6.1.4.1.2789.2500.1535.1", "string", "$name",
".1.3.6.1.4.1.2789.2500.1535.2", "string", "$STATS",
".1.3.6.1.4.1.2789.2500.1535.3", "string", "$FILE_STATS");

# Enviando uma trap usando o snmptrap do OpenView
#
#system "/opt/OV/bin/snmptrap -c public nms
#.1.3.6.1.4.1.2789.2500 \"\" 6 $SPEC \"\""
#.1.3.6.1.4.1.2789.2500.1535.1 octetstringascii \"\$name\"
#.1.3.6.1.4.1.2789.2500.1535.2 octetstringascii \"\$STATS\"
#.1.3.6.1.4.1.2789.2500.1535.3 octetstringascii \"\$FILE_STATS\";
}

```

A lógica é simples, embora seja um pouco difícil perceber porque a maior parte dessa lógica ocorre implicitamente. O segredo é a chamada ao `find( )`, que configura muitos itens, entra em cada diretório abaixo do diretório especificado por `$PATH` e define automaticamente `$_` (para que a instrução `if` no início da subrotina `wanted( )` funcione). Além disso, define o nome da variável como o nome de caminho completo para o arquivo atual; isso permite testar se esse arquivo atual é ou não um diretório que não desejariamos eliminar.

Portanto, percorremos todos os arquivos, procuramos os arquivos com o nome especificado na linha de comando (ou denominados `core`, se nenhuma opção `-lookfor` estiver especificada). Quando encontrarmos um arquivo, armazenaremos seus dados estatísticos, eliminaremos esse arquivo e enviaremos uma trap para o NMS reportando o nome do arquivo e outras informações. Usamos a variável `SPEC` para armazenar a ID da trap específica. Usamos duas IDs específicas: 1535, se o arquivo foi excluído com êxito e 1536, se tentamos eliminar o arquivo mas não conseguimos. Mais uma vez, escrevemos o código da trap para usar a Perl interna, o Net-SNMP ou o OpenView. Remova o comentário da versão que desejar. Empacotamos a trap com três associações de variáveis, que contêm o nome do arquivo, os resultados de `ls -l` sobre o arquivo e os resultados da execução de `/bin/file`. Juntando tudo, temos muitas informações sobre o arquivo eliminado. Observe que foi necessário definir as IDs de objeto para as três variáveis; além disso, embora tenhamos colocado essas IDs | 207

de objeto em 1535, nada nos impede de usar os mesmos objetos ao enviarmos a trap específica 1536.

Agora que temos um programa para eliminar arquivos básicos (do núcleo) e enviar traps informando o que foi eliminado, a próxima etapa é informar ao receptor da trap o que fazer com essas traps recebidas. Suponhamos que estamos usando o OpenView. Para informá-lo sobre essas traps, é necessário adicionar duas entradas ao *trapd.conf*, mapeando essas traps para eventos. Eis as duas entradas:

```
EVENT foundNDelCore .1.3.6.1.4.1.2789.2500.0.1535 "Status Alarms" Warning
FORMAT Core File Found :$1: File Has Been Deleted - LS :$2: FILE :$3:
SDESC
```

Este evento é chamado quando um servidor usando cronjob procura arquivos básicos e os elimina.

```
$1 - octetstringascii      - Name of file
$2 - octetstringascii      - ls -l listing on the file
$3 - octetstringascii      - file $name
EDESC
```

```
#
#
```

```
EVENT foundNNotDelCore .1.3.6.1.4.1.2789.2500.0.1536 "Status Alarms" Minor
FORMAT Core File Found :$1:
```

O arquivo não foi eliminado por algum motivo - LS :\$2: FILE :\$3:

```
SDESC
```

Este evento é chamado quando um servidor usando cronjob procura arquivos básicos mas NÃO CONSEGUE eliminá-los por algum motivo.

```
$1 - octetstringascii      - Name of file
$2 - octetstringascii      - ls -l listing on the file
$3 - octetstringascii      - file $name
EDESC
```

```
#
#
```

Para cada trap, temos uma instrução EVENT especificando o nome do evento, a ID específica da trap, a categoria em que o evento será classificado e a severidade. A instrução FORMAT define uma mensagem a ser utilizada quando recebermos a trap; pode ocupar várias linhas e pode usar os parâmetros \$1, \$2 etc. para fazer referência às associações de variáveis incluídas na trap.

Embora fosse uma boa idéia, não precisamos adicionar nossas associações de variáveis a nosso arquivo de MIB privado; o *trapd.conf* contém informações suficientes para o OpenView interpretar o conteúdo da trap.

Eis alguns exemplos de traps\* geradas pelo script *throwcore*:

```
Core File Found :/usr/sap/HQD/DVEBMGS00/work/core: File Has Been \
Deleted - LS :--rw--rw---- 1 hqdadm sapsys 355042304 Apr 27 17:04 \
/usr/sap/HQD/DVEBMGS00/work/core: \
```

```

FILE :/usr/sap/HQD/DVEBMGS00/work/core: ELF 32-bit MSB core file \
SPARC Version 1, from 'disp+work':SPR D3100 S0-Tovidas 10-10x1b ba
Core File Found :/usr/sap/HQI/DVEBMGS10/work/core: File Has Been \
Deleted - LS :-rw-r--r-- 1 hqiadm sapsys 421499988 Apr 28 14:29 \
/usr/sap/HQI/DVEBMGS10/work/core: \
FILE :/usr/sap/HQI/DVEBMGS10/work/core: ELF 32-bit MSB core file\b \STAT2X
SPARC Version 1, from 'disp+work':

```

Eis o *crontab* da raiz, que executa o script *throwcore* em intervalos específicos. Observe que usamos o switch *-path*, que permite verificar a área de desenvolvimento a cada hora:

```

# Procurar arquivos do núcleo todas as noites e a cada hora, nos dirs
especiais
27 * * * * /opt/local/mib_programs/scripts/throwcore.pl -path /usr/sap
23 2 * * * /opt/local/mib_programs/scripts/throwcore.pl

```

## Verificação de disco Veritas

O Veritas Volume Manager é um pacote que permite manipular discos e suas partições, adicionar e remover espelhos, trabalhar com arrays de RAID e redimensionar partições, apenas para citar algumas possibilidades. Embora o Veritas seja um pacote especializado e caro, geralmente encontrado nas grandes centrais de dados, nem pense em ignorar esta seção. O objetivo não é ensinar a monitorizar o Veritas, mas mostrar como é possível fornecer traps significativas usando um programa de status comum. Você deve extrair as idéias do script apresentado aqui e usá-las em seu próprio contexto.

O Veritas Volume Manager (*vxvm*) é fornecido com um utilitário denominado *vxprint*. Esse programa exibe os registros da configuração do Volume Manager e apresenta o status de cada disco local. Se ocorrer um erro, como disco com defeito ou espelho partido, esse comando informará. Um *vxprint* sadio no rootvol (/) é parecido com este:

```
$ vxprint -h rootvol
Disk group: rootdg
```

TY	NAME	ASSOC	KSTATE	LENGTH	Ploffs	STATE	TUTILO
PUTILO							
v	rootvol	root	ENABLED	922320	-	ACTIVE	-
-							
p1	rootvol-01	rootvol	ENABLED	922320	-	ACTIVE	-
-							
sd	roottdisk-B0	rootvol-01	ENABLED	1	0	-	-
Block0							
sd	roottdisk-02	rootvol-01	ENABLED	922319	1	-	-
-							
p1	rootvol-02	rootvol	ENABLED	922320	-	ACTIVE	-

```
- sd disk01-01      rootvol-02    ENABLED  922320 0
```

As colunas KSTATE (estado do kernel) e STATE apresentam uma perspectiva interna de nossos discos, espelhos etc. Sem explicar a saída minuciosamente, um KSTATE definido com ENABLED é um bom sinal; um STATE ACTIVE ou - indica ausência de problemas. Podemos usar essa saída e canalizá-la para um script que envia traps do SNMP quando são detectados erros. É possível enviar outras traps com uma severidade adequada, com base no tipo de erro que o *vxprint* reportou. Eis um script que executa o *vxprint* e analisa os resultados:

```
#!/usr/local/bin/perl -wC

$VXPRINT_LOC      = "/usr/sbin/vxprint";
$HOSTNAME         = `bin/uname -n`; chop $HOSTNAME;

while ($ARGV[0] =~ /^-/)
{
    if    ($ARGV[0] eq "-debug")        { shift; $DEBUG = $ARGV[0]; }
    elsif ($ARGV[0] eq "-state_active") { $SHOW_STATE_ACTIVE = 1; }
    shift;
}

#####
##### Begin Main #####
#####

&get_vxprint; # Get it, process it, and send traps if errors found!

#####
##### Begin SubRoutines #####
#####

sub get_vxprint
{
    open(VXPRINT,$VXPRINT_LOC) || die "Can't Open $VXPRINT_LOC";
    while($VXLINE=<VXPRINT>)
    {
        print $VXLINE unless ($DEBUG < 2);
        if ($VXLINE ne "\n")
        {
            &is_a_disk_group_name;
            &split_vxprint_output;

            if (($TY ne "TY")   &&
                ($TY ne "Disk") &&
                ($TY ne "dg")   &&
```

```

        ($TY ne "dm"))
    {
        if (($SHOW_STATE_ACTIVE) && ($STATE eq "ACTIVE"))
        {
            print "ACTIVE: $VXLINE";
        }
        if (($STATE ne "ACTIVE") &&
            ($STATE ne "DISABLED") &&
            ($STATE ne "SYNC") &&
            ($STATE ne "CLEAN") &&
            ($STATE ne "SPARE") &&
            ($STATE ne "-") &&
            ($STATE ne ""))
        {
            &send_error_msgs;
        }
        elsif (($KSTATE ne "ENABLED") &&
               ($KSTATE ne "DISABLED") &&
               ($KSTATE ne "-") &&
               ($KSTATE ne ""))
        {
            &send_error_msgs;
        }
    } # end if (($TY
    } # end if ($VXLINE
} # end while($VXLINE)
} # end sub get_vxprint

sub is_a_disk_group_name
{
    if ($VXLINE =~ /Disk\sgroup\:\s(\w+)\n/)
    {
        $DISK_GROUP = $1;
        print "Found Disk Group :$1:\n" unless (!$DEBUG);
        return 1;
    }
}

sub split_vxprint_output
{
    ($TY, $NAME, $ASSOC, $KSTATE,
     $LENGTH, $PLOFFS, $STATE, $TUTIL0,
     $PUTIL0) = split(/\s+/, $VXLINE);

    if ($DEBUG) {
        print "SPLIT: $TY $NAME $ASSOC $KSTATE ";
        print "$LENGTH $PLOFFS $STATE $TUTIL0 $PUTIL0:\n";
    }
}

```

```

}

sub send_snmp_trap
{
    $SNMP_TRAP_LOC          = "/opt/OV/bin/snmptrap";
    $SNMP_COMM_NAME         = "public";
    $SNMP_TRAP_HOST          = "nms";

    $SNMP_ENTERPRISE_ID      = ".1.3.6.1.4.1.2789.2500";
    $SNMP_GEN_TRAP           = "6";
    $SNMP_SPECIFIC_TRAP      = "1000";

    chop($SNMP_TIME_STAMP        = "1" . `date +%H%S`);
    $SNMP_EVENT_IDENT_ONE     = ".1.3.6.1.4.1.2789.2500.1000.1";
    $SNMP_EVENT_VTYPE_ONE      = "octetstringascii";
    $SNMP_EVENT_VAR_ONE        = "$HOSTNAME";

    $SNMP_EVENT_IDENT_TWO     = ".1.3.6.1.4.1.2789.2500.1000.2";
    $SNMP_EVENT_VTYPE_TWO      = "octetstringascii";
    $SNMP_EVENT_VAR_TWO        = "$NAME";

    $SNMP_EVENT_IDENT_THREE   = ".1.3.6.1.4.1.2789.2500.1000.3";
    $SNMP_EVENT_VTYPE_THREE    = "octetstringascii";
    $SNMP_EVENT_VAR_THREE      = "$STATE";

    $SNMP_EVENT_IDENT_FOUR    = ".1.3.6.1.4.1.2789.2500.1000.4";
    $SNMP_EVENT_VTYPE_FOUR     = "octetstringascii";
    $SNMP_EVENT_VAR_FOUR       = "$DISK_GROUP";

    $SNMP_TRAP = "$SNMP_TRAP_LOC \c $SNMP_COMM_NAME $SNMP_TRAP_HOST
$SNMP_ENTERPRISE_ID \"\$\" $SNMP_GEN_TRAP $SNMP_SPECIFIC_TRAP"
$SNMP_TIME_STAMP
    $SNMP_EVENT_IDENT_ONE    $SNMP_EVENT_VTYPE_ONE
\"$SNMP_EVENT_VAR_ONE\"
    $SNMP_EVENT_IDENT_TWO    $SNMP_EVENT_VTYPE_TWO
\"$SNMP_EVENT_VAR_TWO\"
    $SNMP_EVENT_IDENT_THREE   $SNMP_EVENT_VTYPE_THREE
\"$SNMP_EVENT_VAR_THREE\"
    $SNMP_EVENT_IDENT_FOUR    $SNMP_EVENT_VTYPE_FOUR
\"$SNMP_EVENT_VAR_FOUR\"";

    # Enviando uma trap com o Net-SNMP
    #
    #system "/usr/local/bin/snmptrap $SNMP_TRAP_HOST $SNMP_COMM_NAME
#$SNMP_ENTERPRISE_ID '' $SNMP_GEN_TRAP $SNMP_SPECIFIC_TRAP ''
#$SNMP_EVENT_IDENT_ONE s \"\$SNMP_EVENT_VAR_ONE\"
#$SNMP_EVENT_IDENT_TWO s \"\$SNMP_EVENT_VAR_TWO\"
#$SNMP_EVENT_IDENT_THREE s \"\$SNMP_EVENT_VAR_THREE\"
#$SNMP_EVENT_IDENT_FOUR s \"\$SNMP_EVENT_VAR_FOUR\"";
}

```

```

# Enviando uma trap com a Perl
# Use o comando berdump -t no terminal para analisar o resultado
#use SNMP_util "0.54"; # Isso carregará BER e SNMP_Session
$snmptrap("$SNMP_COMM_NAME@$SNMP_TRAP_HOST:162",
"$SNMP_ENTERPRISE_ID",
#my$localhostname, $SNMP_GEN_TRAP, $SNMP_SPECIFIC_TRAP,
#"$SNMP_EVENT_IDENT_ONE", "string", "$SNMP_EVENT_VAR_ONE",
#"$SNMP_EVENT_IDENT_TWO", "string", "$SNMP_EVENT_VAR_TWO",
#"$SNMP_EVENT_IDENT_THREE", "string", "$SNMP_EVENT_VAR_THREE",
#"$SNMP_EVENT_IDENT_FOUR", "string", "$SNMP_EVENT_VAR_FOUR");

# Enviando uma trap com o snmptrap do OpenView (usando VARs citadas
acima)
#
if($SEND_SNMP_TRAP) {
    print "Problem Running SnmpTrap with Result ";
    print ":$SEND_SNMP_TRAP: :$SNMP_TRAP:\n";
}

sub send_error_msgs
{
    $TY =~ s/^v/Volume/;
    $TY =~ s/^pl/Plex/;
    $TY =~ s/^sd/SubDisk/;

    print "VXfs Problem: Host:[$HOSTNAME] State:[$STATE]
DiskGroup:[$DISK_GROUP]
Type:[$TY] FileSystem:[$NAME] Assoc:[$ASSOC] Kstate:[$KSTATE]\n"
unless (!$DEBUG);

    &send_snmp_trap;
}

```

Conhecendo a saída do *vxprint*, podemos formular instruções do Perl para determinar quando é necessário gerar uma trap. Essa tarefa ocupa a maior parte da subrotina *get\_vxprint*. Também sabemos que tipos de mensagens de erro serão geradas. Nossa script tenta ignorar todas as informações dos discos sadios e classificar as mensagens de erro. Por exemplo, se o campo STATE contiver NEEDSYNC, provavelmente os espelhos de disco não estão sincronizados e o volume precisa de atenção. O script não manipula esse caso específico explicitamente, mas ele é detectado com a entrada default.

O mecanismo real para enviar a trap está amarrado a um grande número de variáveis. Entretanto, usamos basicamente qualquer um dos utilitários de trap discutidos; a ID da empresa é .1.3.6.1.4.1.2789.2500; a ID da trap específica é 1000; e incluímos quatro vinculações de variáveis, que informam o nome do host, o nome do volume, o estado do volume e o grupo de discos.

Como no script anterior, basta executar este script periodicamente e observar os resultados no software de gerenciamento de rede que você estiver

usando, seja ele qual for. Também é fácil perceber como você poderia desenvolver scripts semelhantes, que gerem relatórios de outros programas de status.

## Verificador de espaço de disco

O agente do OpenView possui um objeto *fileSystemTable* que contém dados estatísticos sobre a utilização do disco e outros parâmetros do sistema de arquivos. A princípio, parece muito útil: você pode utilizá-lo para descobrir nomes do sistema de arquivos, blocos disponíveis etc. Mas tem algumas artimanhas e precisaremos de alguns truques para utilizar essa tabela com eficiência. Ao percorrer *fileSystemTable.fileSystemEntry.FileSystemDir (.1.3.6.1.4.1.11.2.3.1.2.2.1.10)*, serão listados os sistemas de arquivos atualmente montados:<sup>\*</sup>

```
[root][nms] /opt/OV/local/bin/disk_space> snmpwalk spruce \
.1.3.6.1.4.1.11.2.3.1.2.2.1.10
fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.14680064.1
: DISPLAY STRING- (ascii): /
fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.14680067.1
: DISPLAY STRING- (ascii): /var
fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.14680068.1
: DISPLAY STRING- (ascii): /export
fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.14680069.1
: DISPLAY STRING- (ascii): /opt
fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.14680070.1
: DISPLAY STRING- (ascii): /usr
fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.41156608.1
: DISPLAY STRING- (ascii): /proc

fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.41680896.1
: DISPLAY STRING- (ascii): /dev/fd
fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.42991617.1
: DISPLAY STRING- (ascii): /net
fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.42991618.1
: DISPLAY STRING- (ascii): /home
fileSystem.fileSystemTable.fileSystemEntry.FileSystemDir.42991619.1
: DISPLAY STRING- (ascii): /xfn
```

Examinemos como seria possível escrever um programa para detectar o espaço de disco disponível. À primeira vista, parece fácil. Mas essa tabela contém alguns objetos que não são sistemas de arquivos no contexto normal; */proc*, por exemplo, dá acesso aos processos em execução no sistema e não representa o armazenamento. Isso gera problemas se começarmos ao pesquisar os blocos disponíveis: */proc* não terá quaisquer blocos livres e */dev/fd*, que representa um disquete, só terá blocos livres se existir um disco inserido na unidade. Esperaria-

\* Truncamos a parte *.iso.org.dod.internet.private.enterprises.hp.nm.system.general* para o resultado do percurso, por questão de espaço.

mos que */home* se comportasse como um sistema de arquivos normal mas, no servidor, tem montagem automática, o que significa que seu comportamento é imprevisível; se não estiver em uso, não deve ser montado. Portanto, se pesquisarmos os blocos livres usando o objeto *fileSystem.fileSystemTable.FileSystemEntry.FileSystemBavail*, as cinco últimas instâncias devem retornar 0 sob condições normais. Sendo assim, os resultados obtidos da pesquisa de todas as entradas na tabela do sistema de arquivos não são significativos sem uma interpretação mais abrangente. No mínimo, precisamos descobrir quais sistemas de arquivos são importantes ou não para nós. Provavelmente, isso exigirá perspicácia quanto ao número de instâncias.

Quando descobri esse problema, observei que todos os sistemas de arquivos que eu desejava verificar tinham números de instâncias com os mesmos dígitos iniciais; por exemplo, *fileSystemDir.14680064.1*, *fileSystemDir.14680067.1*, *fileSystemDir.14680068.1* etc. Essa observação provou-se menos útil do que parecia – com o tempo, aprendi que não somente os outros servidores têm terminações de números de instâncias diferentes, como também que, em qualquer servidor, os números de instâncias podem mudar. Entretanto, mesmo que o número da instância mude, os dígitos finais da instância permanecem inalterados para todos os discos ou sistemas de arquivos do mesmo tipo. Por exemplo, os arrays de disco podem ter números de instância como *fileSystemDir.12312310.1*, *fileSystemDir.12312311.1*, *fileSystemDir.12312312.1*, e assim por diante. Os discos internos de seu sistema podem ter números de instância como *fileSystemDir.12388817.1*, *fileSystemDir.12388818.1*, *fileSystemDir.12388819.1*, e outros.

Sendo assim, é possível trabalhar com números de instância, sem problemas – além disso, não há nada estático que possa ser pesquisado. Não há como dizer “Forneça-me os dados estatísticos de todos os sistemas de arquivos locais” ou “Informe os dados estatísticos de */usr*”. Fui obrigado a escrever um programa que se encarregasse de grande parte do processamento de números de instância, partindo de premissas baseadas no comportamento que observei. Precisei utilizar o *snmpwalk* para descobrir os números de instância dos sistemas de arquivos em questão, antes de fazer algo mais interessante. Ao comparar os dígitos iniciais dos números de instância, consegui descobrir quais sistemas de arquivos eram locais, quais estavam em rede e quais eram de “propósito especial” (como o */proc*). Eis o resultado:

```
#!/usr/local/bin/perl
# filename: polling.pl
# options:
#   -min n : envie trap se o número de blocos livres for inferior a n
#   -table f : tabela de servidores a serem observados (definida com ./default)
#   -server s : especifica um único servidor a ser pesquisado
#   -inst n : número de dígitos iniciais de números de instâncias a serem comparados
#   -debug n : nível de depuração

$|++;

$SNMPWALK_LOC = "/opt/OV/bin/snmpwalk -r 5";
$SNMPGET_LOC = "/opt/OV/bin/snmpget";
```

```

$HOME_LOC      = "/opt/OV/local/bin/disk_space";          # Localização do script
$LOCK_FILE_LOC = "$HOME_LOC/lock_files";                # Localização para o lock file
$GREP_LOC      = "/bin/grep";
$TOUCH_LOC     = "/bin/touch";
$PING_LOC      = "/usr/sbin/ping";                      # Localização do Ping
$PING_TIMEOUT   = 7;                                     # Segundos de espera por um ping

$MIB_C = ".1.3.6.1.4.1.11.2.3.1.2.2.1.6";           # fileSystemBavail
$MIB_BSIZE = ".1.3.6.1.4.1.11.2.3.1.2.2.1.7";        # fileSystemBsize
$MIB_DIR = ".1.3.6.1.4.1.11.2.3.1.2.2.1.10";         # fileSystemDir

while ($ARGV[0] =~ /^-/)
{
  if    ($ARGV[0] eq "-min") { shift; $MIN = $ARGV[0]; }  # Em blocos de 1024
  elsif ($ARGV[0] eq "-table") { shift; $TABLE = $ARGV[0]; }
  elsif ($ARGV[0] eq "-server") { shift; $SERVER = $ARGV[0]; }
  elsif ($ARGV[0] eq "-inst") { shift; $INST_LENGTH = $ARGV[0]; }
  elsif ($ARGV[0] eq "-debug") { shift; $DEBUG = $ARGV[0]; }
  shift;
}

#####
##### Begin Main #####
#####

$ALLSOURCES = 1 unless ($SERVER);
$INST_LENGTH = 5 unless ($INST_LENGTH);

$TABLE = "default" unless ($TABLE);

open(TABLE,"$HOME_LOC/$TABLE") || die "Não é possível abrir o arquivo $TABLE";
while($LINE=<TABLE>)
{
  if ($LINE ne "\n")
  {
    chop $LINE;
    ($HOST,$IGNORE1,$IGNORE2,$IGNORE3) = split(/\:/,$LINE);

    if (&ping_server_bad("SHOST")) { warn "Can't Ping Server
      :$HOST:" unless (!$DEBUG); }
    else
    {
      &find_inst;

      if ($DEBUG > 99)
      {
        print "HOST:$HOST: IGNORE1 :$IGNORE1: IGNORE2 :$IGNORE2:
          IGNORE3 :$IGNORE3:\n";
        print "Running :$SNMPWALK_LOC $HOST $MIB_C \| $GREP_LOC
          \.$INST:\n";
      }

      $IGNORE1 = "C1ANT5MAT9CHT4HIS"
      unless ($IGNORE1); # If we don't have anything then let's set
    }
  }
}

```

```

$IGNORE2 = "CA2N4T6M8A1T3C5H7THIS"
    unless ($IGNORE2); # to something that we can never match.
$IGNORE3 = "CAN3TMA7TCH2THI6S" unless ($IGNORE3);

if (($SERVER eq "$HOST") || ($ALLSERVERS))
{
    open(WALKER,"$SNMPWALK_LOC $HOST $MIB_C \> $GREP_LOC
        \.$INST |") || die "Can't Walk $HOST $MIB_C\n";
    while($WLINE=<$WALKER>)
    {
        chop $WLINE;
        ($MIB,$TYPE,$VALUE) = split(/\:/,$WLINE);
        $MIB =~ s/\s+//g;
        $MIB =~ /(\d+\.\d+)/;
        $INST = $1;

        open(SNMPGET,"$SNMPGET_LOC $HOST $MIB_DIR*$INST |");
        while($DLINE=<$SNMPGET>)
        {
            ($NULL,$NULL,$DNAME) = split(/\:/,$DLINE);
        }
        $DNAME =~ s/\s+//g;
        close SNMPGET;

        open(SNMPGET,"$SNMPGET_LOC $HOST $MIB_BSIZE*$INST |");
        while($BLINE=<$SNMPGET>)
        {
            ($NULL,$NULL,$BSIZE) = split(/\:/,$BLINE);
        }

        close SNMPGET;
        $BSIZE =~ s/\s+//g;

        $LOCK_RES = &inst_found; $LOCK_RES = "\[ $LOCK_RES \]";

        print "LOCK_RES :$LOCK_RES:\n" unless ($DEBUG < 99);

        $VALUE = $VALUE * $BSIZE / 1024; # Coloque-o em blocos de 1024

        if (($DNAME =~ /.*$IGNORE1.*/) ||
            ($DNAME =~ /.*$IGNORE2.*/) ||
            ($DNAME =~ /.*$IGNORE3.*/))
        {
            $DNAME = "$DNAME <ignored>";
        }

        else
        {
            if (($VALUE <= $MIN) && ($LOCK_RES eq "\[ 0 \]"))
            {
                $LOCK_RES = "\[ 1 \]";
            }
        }
    }
}

```

```

        &write_lock;
        &send_snmp_trap(0);
    }

    elsif (($VALUE > $MIN) && ($LOCK_RES eq "\[ 1 \]"))
    {
        &remove_lock;
        &send_snmp_trap(1);
    }
}

$VALUE = $VALUE / $BSIZE * 1024; # Exiba-o como o
                                # tamanho do bloco original
                                # para facilitar a leitura

write unless (!$DEBUG);

} # end while($WLINE=<WALKER>)
} # end if (($SERVER eq "$HOST") || ($ALLSERVERS))
} # end else from if (&ping_server_bad("$HOST"))

} # end if ($LINE ne "\n")
} # end while($LINE=<TABLE>)

#####
##### Begin SubRoutines #####
#####

format STDOUT_TOP =
Server      MountPoint      BlocksLeft      BlockSize      MIB      LockFile
-----  -----  -----  -----
.

format STDOUT =
@<<<<<<  @<<<<<<<<  @<<<<<<<  @<<<<<<<  @<<<<<<  @<<<<<<
$HOST,      $DNAME,          $VALUE,          $BSIZE,          $INST,      $LOCK_RES
.

sub inst_found
{
    if (-e "$LOCK_FILE_LOC/$HOST\.$INST") { return 1; }
    else { return 0; }
}

sub remove_lock
{
    if ($DEBUG > 99) { print "Removing Lockfile $LOCK_FILE_LOC/$HOST\.$INST\n"; }
    unlink "$LOCK_FILE_LOC/$HOST\.$INST";
}

sub write_lock
{
    if ($DEBUG > 99) { print "Writing Lockfile
                                $TOUCH_LOC $LOCK_FILE_LOC/$HOST\.$INST\n"; }
    system "$TOUCH_LOC $LOCK_FILE_LOC/$HOST\.$INST";
}

```

```

}

## send_snmp_trap ##

## Esta subrotina permite enviar traps diferentes, de acordo com
## o parâmetro passado, e permite enviar as traps boas e
## ruins.
#
## $1 - inteiro - Isso será adicionado à ID do evento específico.
#
## Se criássemos duas traps:
## 2789.2500.0.1000 = Principal
## 2789.2500.0.1001 = Boa
#
## Se declarássemos:
## $SNMP_SPECIFIC_TRAP      = "1000";
#
## Poderíamos enviar a 1ª com:
## send_snmp_trap(0); # Eis o cálculo (1000 + 0 = 1000)
## E enviar a segunda:
## send_snmp_trap(1); # Eis o cálculo (1000 + 1 = 1001)
#
## Assim, poderíamos configurar várias traps com erros diferentes, usando
## a mesma função para todas.
#
## sub send_snmp_trap
{
    $TOTAL_TRAPS_CREATED      = 2; # Façamos uma verificação/confirmação
                                    # aqui. Este deve ser o número
                                    # total de traps que você criou
                                    # no nms.

    $SNMP_ENTERPRISE_ID       = ".1.3.6.1.4.1.2789.2500";
    $SNMP_SPECIFIC_TRAP        = "1500";

    $PASSED_PARM              = $_[0];
    $SNMP_SPECIFIC_TRAP += $PASSED_PARM;

    $SNMP_TRAP_LOC             = "/opt/OV/bin/snmptrap";
    $SNMP_COMM_NAME            = "public";
    $SNMP_TRAP_HOST             = "nms";

    $SNMP_GEN_TRAP              = "6";

    chop($SNMP_TIME_STAMP)      = "1" . `date +%H%S`;

    $SNMP_EVENT_IDENT_ONE      = ".1.3.6.1.4.1.2789.2500.$SNMP_SPECIFIC_TRAP.1";
    $SNMP_EVENT_VTYPE_ONE       = "octetstringascii";
    $SNMP_EVENT_VAR_ONE         = "$DNAME";
}

```

```

$SNMP_EVENT_IDENT_TWO = ".1.3.6.1.4.1.2789.2500.$SNMP_SPECIFIC_TRAP.2";
$SNMP_EVENT_VTYPE_TWO = "integer";
$SNMP_EVENT_VAR_TWO = "$VALUE";

$SNMP_EVENT_IDENT_THREE = ".1.3.6.1.4.1.2789.2500.$SNMP_SPECIFIC_TRAP.3";
$SNMP_EVENT_VTYPE_THREE = "integer";
$SNMP_EVENT_VAR_THREE = "$BSIZE";

$SNMP_EVENT_IDENT_FOUR = ".1.3.6.1.4.1.2789.2500.$SNMP_SPECIFIC_TRAP.4";
$SNMP_EVENT_VTYPE_FOUR = "octetstringascii";
$SNMP_EVENT_VAR_FOUR = "$INST";

$SNMP_EVENT_IDENT_FIVE = ".1.3.6.1.4.1.2789.2500.$SNMP_SPECIFIC_TRAP.5";
$SNMP_EVENT_VTYPE_FIVE = "integer";
$SNMP_EVENT_VAR_FIVE = "$MIN";

$SNMP_TRAP = "$SNMP_TRAP_LOC \c $SNMP_COMM_NAME $SNMP_TRAP_HOST
$SNMP_ENTERPRISE_ID \"\$HOST\" $SNMP_GEN_TRAP $SNMP_SPECIFIC_TRAP
$SNMP_TIME_STAMP
$SNMP_EVENT_IDENT_ONE $SNMP_EVENT_VTYPE_ONE \"\$SNMP_EVENT_VAR_ONE\"
$SNMP_EVENT_IDENT_TWO $SNMP_EVENT_VTYPE_TWO \"\$SNMP_EVENT_VAR_TWO\"
$SNMP_EVENT_IDENT_THREE $SNMP_EVENT_VTYPE_THREE \"\$SNMP_EVENT_VAR_THREE\"
$SNMP_EVENT_IDENT_FOUR $SNMP_EVENT_VTYPE_FOUR \"\$SNMP_EVENT_VAR_FOUR\"
$SNMP_EVENT_IDENT_FIVE $SNMP_EVENT_VTYPE_FIVE \"\$SNMP_EVENT_VAR_FIVE\"";
```

if (!(\$PASSED\_PARM < STOTAL\_TRAPS\_CREATED))

{

die "ERROR SNMPTrap with a Specific Number \>
 \$TOTAL\_TRAPS\_CREATED\n\$SNMP\_TRAP:\$SNMP\_TRAP:\n";
}

# Enviando uma trap com o Net-SNMP

#

#system "/usr/local/bin/snmptrap \$SNMP\_TRAP\_HOST \$SNMP\_COMM\_NAME
#\$SNMP\_ENTERPRISE\_ID '' \$SNMP\_GEN\_TRAP \$SNMP\_SPECIFIC\_TRAP ''
#\$SNMP\_EVENT\_IDENT\_ONE s \"\\$SNMP\_EVENT\_VAR\_ONE\"
#\$SNMP\_EVENT\_IDENT\_TWO i \"\\$SNMP\_EVENT\_VAR\_TWO\"
#\$SNMP\_EVENT\_IDENT\_THREE i \"\\$SNMP\_EVENT\_VAR\_THREE\"
#\$SNMP\_EVENT\_IDENT\_FOUR s \"\\$SNMP\_EVENT\_VAR\_FOUR\";
#\$SNMP\_EVENT\_IDENT\_FIVE i \"\\$SNMP\_EVENT\_VAR\_FIVE\"";

# Enviando uma trap com Perl

#

#use SNMP\_util "0.54"; # Isso carregará BER e SNMP\_Session
\$snmptrap("\$SNMP\_COMM\_NAME@\$SNMP\_TRAP\_HOST:162", "\$SNMP\_ENTERPRISE\_ID",
mylocalhostname, \$SNMP\_GEN\_TRAP, \$SNMP\_SPECIFIC\_TRAP,
"\$SNMP\_EVENT\_IDENT\_ONE", "string", "SNMP\_EVENT\_VAR\_ONE",
"\$SNMP\_EVENT\_IDENT\_TWO", "int", "SNMP\_EVENT\_VAR\_TWO",
"\$SNMP\_EVENT\_IDENT\_THREE", "int", "SNMP\_EVENT\_VAR\_THREE",
"\$SNMP\_EVENT\_IDENT\_FOUR", "string", "SNMP\_EVENT\_VAR\_FOUR",
"\$SNMP\_EVENT\_IDENT\_FIVE", "int", "SNMP\_EVENT\_VAR\_FIVE");

# Enviando uma trap com o snmptrap do OpenView (usando VARs citadas acima)

#

```

if($SEND_SNMP_TRAP) {
    print "ERROR Running SnmpTrap Result ";
    print ":$SEND_SNMP_TRAP: :$SNMP_TRAP:\n"
}

sub find_inst
{
    open(SNMPWALK2,"$SNMPWALK_LOC $HOST $MIB_DIR |") || die "Can't Find Inst for $HOST\n";
    while($DLINE=<SNMPWALK2>)
    {
        chomp $DLINE;
        ($DIRTY_INST,$NULL,$DIRTY_NAME) = split(/\:/,$DLINE);
        $DIRTY_NAME =~ s/\s+//g; # Lose the whitespace, folks!
        print "$DIRTY_INST :$DIRTY_INST:\n$DIRTY_NAME :$DIRTY_NAME:\n"
            unless (!$DEBUG > 99);
        if ($DIRTY_NAME eq "/")
        {
            $DIRTY_INST =~ /fileSystemDir\.(\\d*)\\.1/;
            $GINST = $1;
            $LENGTH = (length($GINST) - $INST_LENGTH);
            while ($LENGTH--) { chop $GINST; }
            close SNMPWALK;
            print "Found Inst DIRTY_INST :$DIRTY_INST: DIRTY_NAME\
                :$DIRTY_NAME: GINST :$GINST:$GINST:\n"
                unless (!$DEBUG > 99);
            return 0;
        }
    }
    close SNMPWALK2;
    die "Não foi possível encontrar a Inst do HOST :$HOST:";

}

sub ping_server_bad
{
    local $SERVER = $_[0];
    $RES = system "$PING_LOC $SERVER $PING_TIMEOUT > /dev/null";
    print "Res from Ping :$RES: \-\-$PING_LOC $SERVER:\n"
        unless (!$DEBUG));
    return $RES;
}

```

O script contém diversos recursos úteis:

- Usamos um arquivo ASCII externo para uma lista de servidores a serem pesquisados. Especificamos o arquivo por meio do switch *-table FILENAME*. Se nenhum switch *-table* for especificado, será usado o arquivo denominado *default* no diretório atual.
- É possível especificar um único nome de servidor (que deve aparecer no arquivo acima) para ser pesquisado, por meio do switch *-server SERVER\_NAME*.

- Podemos ignorar até três sistemas de arquivos por servidor. Por exemplo, poderíamos ignorar os sistemas de arquivos sendo utilizados para desenvolvimento de software.
- O script pesquisa somente os servidores que respondem a um *ping*. Não desejamos receber traps de sistema de arquivos de um servidor parado ou não pertencente a uma rede.
- Podemos definir o limiar mínimo para cada lista de servidores em blocos de 1024 bytes, por meio da opção *-min blocks*.
- O script envia uma trap quando o limiar de um servidor foi atingido e envia outra trap quando o estado retorna ao normal.
- Usamos lockfiles para impedir que o servidor emita um número excessivo de traps redundantes.\* Quando um limiar for atingido, será criado um arquivo denominado *hostname.instance*. Enviamos uma trap somente se não existir um lockfile. Quando o espaço for liberado, eliminaremos o lockfile, o que permitirá gerar uma trap na próxima vez em que o armazenamento livre ficar abaixo do limiar.
- Podemos definir o número de dígitos iniciais de instâncias usados para marcar o sistema de arquivos adequado, com o switch *-inst*. Infelizmente, o número de dígitos de instância que você pode utilizar com segurança para isolar um sistema de arquivos local varia de uma instalação para outra. O default são cinco mas um valor mais baixo pode ser adequado.
- O script exibe uma tabela útil quando o chamamos com o flag *-debug*.

O script começa lendo a tabela de servidores em que estamos interessados, emite um *ping* para os servidores e ignora os que não respondem. Em seguida, chama a subrotina *find\_inst*, que incorpora a maior parte da lógica de números de instância. Essa subrotina percorre a tabela do sistema de arquivos para localizar uma lista de todos os sistemas de arquivos e os respectivos números de instância; extrai a entrada do sistema de arquivos da raiz (/), que sabemos que existe, e que presumimos tratar-se de um disco local. (Não podemos pressupor que o sistema de arquivos raiz será listado em primeiro lugar; na realidade, presumimos que você não usará um script como este para monitorizar as estações de trabalho sem discos.) Então, armazenamos os dígitos do número de instância do primeiro *INST\_LENGTH* na variável *GINST* e retornamos.

De volta ao programa principal, solicitamos o número de blocos disponíveis para cada sistema de arquivos; comparamos o número da instância com *GINST*, que seleciona os sistemas de arquivos locais (por exemplo, os sistemas de arquivos com um número de instâncias cujos dígitos iniciais correspondem ao número de instâncias de /). Em seguida, solicitamos o número total de blocos, o

---

\* Provavelmente, em algumas situações, ignoramos o fato de que o sistema estivesse preenchido porque uma trap desapareceu durante a transmissão. Usando o *cron*, eliminamos freqüentemente todo o conteúdo do diretório *lock*. Esse procedimento reenvia as entradas, se existirem, nessa ocasião.

que permite comparar o espaço disponível com os limiares definidos. Se o valor for inferior ao mínimo, enviaremos uma das duas traps específicas de empresa, que definimos para este programa, 1500, que indica que o espaço livre do sistema de arquivos está abaixo do limiar. Se o espaço livre retornou a um nível seguro, enviaremos a trap 1501, que é uma notificação de “fora de perigo”. Uma lógica adicional usa um lockfile para impedir que o script bombardeie o NMS com notificações repetidas; enviamos, no máximo, um aviso por dia e uma mensagem “fora de perigo” somente se enviamos anteriormente um aviso. Em ambos os casos, incluímos na trap algumas informações úteis: algumas vinculações de variáveis especificando o sistema de arquivos, o espaço disponível, a capacidade total, o número de instância e o limiar definidos. Mais adiante, aprendere-mos a mapear essas traps em categorias do OpenView.

Coloquemos o programa para funcionar, criando uma tabela denominada *default* que lista os servidores que queremos observar:

```
db_serv0  
db_serv1  
db_serv2
```

Agora, podemos executar o script com a opção *-debug* para exibir uma tabela de resultados. O comando seguinte solicita todos os sistemas de arquivos existentes no servidor *db\_serv0* que possuem menos de 50.000 blocos (50 MB) disponíveis:

```
$ /opt/0V/local/bin/disk_space/polling.pl -min 50000 -server db_serv0 -debug 1  
Res from Ping :0: - :/usr/sbin/ping db_serv0:  
Server      MountPoint      BlocksLeft   BlockSize   MIB      LockFile  
-----  
db_serv0    /           207766       1024     38010880.1   [ 0 ]  
db_serv0    /usr         334091       1024     38010886.1   [ 0 ]  
db_serv0    /opt         937538       1024     38010887.1   [ 0 ]  
db_serv0    /var         414964       1024     38010888.1   [ 0 ]  
db_serv0    /db1         324954       1024     38010889.1   [ 0 ]
```

Observe que não precisamos especificar uma tabela explicitamente; como omitimos a opção *-table*, o script *polling.pl* usou o arquivo *default* que inserimos no diretório atual. O switch *-server* permite limitar o teste ao servidor denominado *db\_serv0*; se tivéssemos omitido essa opção, o script teria verificado todos os servidores existentes na tabela *default*. Se o espaço disponível em qualquer um dos sistemas de arquivos estiver abaixo de 50.000 blocos de 1024 bytes, o programa enviará uma trap e criará um lockfile com o número da instância.

Como as traps do SNMP usam UDP, não são confiáveis. Isso significa que é provável que algumas traps nunca alcancem seu destino. Isso poderia indicar um desastre – nessa situação, estamos enviando traps para avisar a um gerente que existe um sistema de arquivos cheio. Não desejamos que essas traps desapareçam, principalmente porque elaboramos nosso programa de modo a não enviar notificações repetidas. Uma solução seria instruir o *cron* a eliminar alguns ou todos os arquivos do diretório *lock*.

Preferimos eliminar todo o conteúdo do diretório *lock* a cada hora; isso significa que receberemos uma notificação por hora, até um armazenamento disponível aparecer no sistema de arquivos. Outro procedimento plausível seria eliminar somente os lockfiles do servidor da produção. Com esse procedimento, receberemos a notificação horária sobre os problemas de capacidade dos sistemas de arquivos, ocorridos no servidor em que estamos mais interessados; nas outras máquinas (por exemplo, máquinas de desenvolvimento, máquinas de teste), receberemos somente uma única notificação.

Vamos supor que o sistema de arquivos */db1* é um sistema de teste e não tem importância se esse sistema estiver cheio. Podemos ignorá-lo, especificando-o em nossa tabela. Podemos listar até três sistemas de arquivos que gostaríamos de ignorar, depois do nome do servidor (que devem ser seguidos por um caractere ":"):

```
db_serv0:/db1
```

A reexecução do script *polling.pl* dá os seguintes resultados:

```
$ /opt/OV/local/bin/disk_space/polling.pl -min 50000 -server db_serv0 -debug 1
Res from Ping :0: - :/usr/sbin/ping db_serv0:
Server      MountPoint      BlocksLeft   BlockSize   MIB      LockFile
-----
db_serv0      /            207766       1024        38010880.1    [ 0 ]
db_serv0      /usr          334091       1024        38010886.1    [ 0 ]
db_serv0      /opt          937538       1024        38010887.1    [ 0 ]
db_serv0      /var          414964       1024        38010888.1    [ 0 ]
db_serv0      /db1 (ignored) 324954       1024        38010889.1    [ 0 ]
```

Quando o sistema de arquivos */db1* estiver abaixo do espaço de disco mínimo, o script não enviará quaisquer traps nem criará quaisquer lockfiles.

Agora, avancemos um pouco mais. As entradas do *crontab* a seguir executam nosso programa duas vezes por hora:

```
4,34 * * * * /opt/OV/bin/polling.pl -min 50000
5,35 * * * * /opt/OV/bin/polling.pl -min 17000 -table stocks_table
7,37 * * * * /opt/OV/bin/polling.pl -min 25000 -table bonds_table -inst 3
```

Em seguida, precisamos definir como as traps geradas pelo *polling.pl* devem ser tratadas quando alcançarem o NMS. Eis a entrada no arquivo *trapd.conf* do OpenView, que mostra como tratar essas traps:

```
EVENT DiskSpaceLow .1.3.6.1.4.1.2789.2500.0.1500 "Threshold Alarms" Major
FORMAT Disk Space For FileSystem :$1: Is Low With :$2:
1024 Blocks Left - Current FS Block Size :$3: - Min Threshold
:$5: - Inst :$4:
SDESC
$1 - octetstringascii  - FileSystem
$2 - integer             - Current Size
$3 - integer             - Block Size
$4 - octetstringascii  - INST
```

Essas entradas definem dois eventos do OpenView: um evento *DiskSpaceLow*, usado quando a capacidade do sistema de arquivos está abaixo do limiar, e um evento *DiskSpaceNormal*. Inserimos esses dois eventos na categoria Threshold Alarms; o evento de baixo espaço de disco tem um nível de severidade *Major*, enquanto o evento “normal” tem um nível de severidade *Normal*. Se você estiver usando outro pacote para detectar traps, configure-o adequadamente

*Monitor de Portas*

A maioria dos serviços de TCP/IP usa portas estáticas para detectar as solicitações recebidas. O monitoramento dessas portas permite saber se servidores ou serviços específicos estão respondendo ou não. Por exemplo, é possível saber se o servidor de correio está funcionando, verificando periodicamente a porta 25, porta em que o servidor de SMTP detecta solicitações. Algumas outras portas a serem monitorizadas são FTP (23), HTTP (80) e POP3 (110).<sup>\*</sup> Um programa gratuito denominado *netcat* pode conectar-se a e interagir com uma porta específica em qualquer dispositivo. Podemos escrever um wrapper (envoltório) para esse programa, para observar determinada porta ou serviço; se acontecer algo fora da operação normal, poderemos enviar uma trap. Nesta seção, desenvolveremos um wrapper que verifica a porta de SMTP (25) no servidor de correio. O programa é muito simples mas os resultados são surpreendentes!

Antes de começar a escrever o programa, vamos definir o que podemos fazer. Conecte-se à porta 25 do servidor de SMTP, via telnet. Uma vez conectado,

\* Procure em seu arquivo *services* uma listagem de números de portas e os respectivos serviços. Nos sistemas Unix, geralmente este arquivo se encontra no diretório */etc*; no Windows, em geral, reside em um diretório como *C:\WINNT\System32\drivers\etc*, embora a localização desse diretório varie em função da versão do Windows utilizada.

você pode emitir o comando *HELO mydomain.com*, que deve responder com 250. Após receber uma resposta do servidor de correio, emita um comando *QUIT*, que informa ao servidor que você terminou. Sua sessão deve ficar parecida com esta:

```
$ telnet mail.ora.com 25
220 smtp.oreilly.com ESMTP O'Reilly & Associates Sendmail 8.11.2 ready
HELO mydomain.com
250 OK
QUIT
221 closing connection
```

O programa *netcat* precisar conhecer os comandos que você deseja enviar para a porta sendo monitorizada. Enviaremos somente dois comandos para o servidor de correio; portanto, criaremos um arquivo denominado *input.txt*, parecido com este:

```
HELO mydomain.com
QUIT
```

Em seguida, devemos testar esse arquivo e observar a saída recebida do servidor. O verdadeiro *netcat* executável é denominado *nc*; para testar o arquivo, execute-o assim:

```
$ /opt/0V/local/bin/netcat/nc -i 1 mailserver 25 < input.txt
```

Esse comando gera os mesmos resultados que a sessão do *telnet*. Você não verá os comandos repetidos no arquivo *input.txt*, mas deverá ver as respostas do servidor. Após verificar se o *netcat* funciona e emite a mesma resposta sempre, salve uma cópia da saída no arquivo *mail\_good*, que será usado para determinar como deve ser uma resposta normal do servidor de correio. Você pode salvar a saída em um arquivo, com o seguinte comando:

```
$ /opt/0V/local/bin/netcat/nc -i 1 mailserver 25 < input.txt > mail_good
```

Uma alternativa seria procurar a linha nº 250 na saída do servidor de correio. Esse código indica que o servidor está normal e em execução, embora não necessariamente processando o correio corretamente. Seja qual for o caso, procurar a linha 250 o resguardará contra as variações na resposta do servidor à sua conexão.

Examine a seguir um script denominado *mail\_poller.pl* que automatiza o processo. Edite as linhas adequadas nesse script, de modo a refletir seu ambiente local. Após personalizar o script, você poderá seguir em frente. Não existem argumentos de linha de comando. O script gera um arquivo de saída denominado *mail\_status* que conterá um 0 (zero) se o servidor está funcionando corretamente (por exemplo, se a saída do *netcat* corresponder a *\$GOOD\_FILE*); qualquer número diferente de 0 indica que ocorreu um erro:

```

#!/usr/local/bin/perl
# filename: mail_poller.pl

$HOME_LOC      = "/opt/OV/local/bin/netcat";
$NC_LOC        = "/opt/netcat/nc";
$DIFF_LOC      = "/bin/diff";
$ECHO_LOC      = "/bin/echo";

$MAIL_SERVER   = "mail.exampledomain.com";
$MAIL_PORT     = 25;
$INPUT_FILE    = "$HOME_LOC\input.txt";
$GOOD_FILE     = "$HOME_LOC\mail_good";
$CURRENT_FILE  = "$HOME_LOC\mail_current";
$EXIT_FILE     = "$HOME_LOC\mail_status";

$DEBUG = 0;

print "$NC_LOC -i 1 -w 3 $MAIL_SERVER $MAIL_PORT
\< $INPUT_FILE \> $CURRENT_FILE\n" unless (!$DEBUG);

$NETCAT_RES = system "$NC_LOC -i 1 -w 3 $MAIL_SERVER $MAIL_PORT
\< $INPUT_FILE \> $CURRENT_FILE";
$NETCAT_RES = $NETCAT_RES / 256;

if ($NETCAT_RES)
{
    # Problema no netcat... talvez um timeout?
    system "$ECHO_LOC $NETCAT_RES > $EXIT_FILE";
    &cleanup;
}

$DIFF_RES = system "$DIFF_LOC $GOOD_FILE $CURRENT_FILE";
$DIFF_RES = $DIFF_RES / 256;

if ($DIFF_RES)
{
    # parece que as coisas estão diferentes!
    system "$ECHO_LOC $DIFF_RES > $EXIT_FILE";
    &cleanup;
}
else
{
    # Todos os sistemas normais!
    system "$ECHO_LOC 0 > $EXIT_FILE";
    &cleanup;
}

sub cleanup

```

```
{  
    unlink "$CURRENT_FILE";  
    exit 0;  
}
```

Após executar o programa, examine os resultados em *mail\_status*. Se for possível, tente encerrar o servidor de correio e executar o script novamente. Agora, seu arquivo deve conter um status de erro diferente de zero.

Depois que você confirmar que o script funciona em seu ambiente, você pode inserir uma entrada *crontab* para executar esse programa em intervalos especificados. Em nosso ambiente, aplicamos um intervalo de 10 minutos:

```
# Verifique o servidor de correio e crie um arquivo que pode pesquisar  
via OpenView  
1,11,21,31,41,51 * * * * /opt/0V/local/bin/netcat/mail_poller.pl
```

Observe que seccionamos a polling para evitar uma verificação a cada hora, meia-hora ou quinze minutos. Assim que o *cron* começar a atualizar regularmente o *mail\_status*, você poderá utilizar ferramentas, como o agente extensível do OpenView, para verificar o conteúdo do arquivo. É possível configurar o agente para pesquisar regularmente o arquivo e enviar os resultados para seu console de gerenciamento. A entrada em meu */etc/SnmpAgent.d/snmpd.extend* é parecida com esta:

```
serviceInfo      OBJECT IDENTIFIER ::= { mauro 5 }  
  
- BEGIN - serviceInfo  
-  
  
serMailPort  OBJECT-TYPE  
    SYNTAX  INTEGER  
    ACCESS  read-only  
    STATUS  mandatory  
    DESCRIPTION  
        "Este arquivo é atualizado via crontab. Ele usa o netcat para  
testar a  
porta e inserir um valor neste arquivo.  
FILE-NAMES: /opt/0V/local/bin/netcat/mail_status"  
::= { serviceInfo 0 }
```

Discutimos a sintaxe deste arquivo no Capítulo 11. Basicamente, essa entrada só define um objeto MIB na árvore *serviceInfo*, que é o nó 5 na árvore de minha empresa privada. Em outras palavras, a OID desse objeto é *mauro.serviceInfo.serMailPort (2789.5.0)*. O objeto pode ser lido por qualquer programa que possa emitir uma operação *get* do SNMP. A *DESCRIPTION*, como vimos no Capítulo 11, especifica um nome de arquivo em que o agente lerá um valor inteiro a ser utilizado como valor desse objeto. Esse programa pode ser facilmente modificado para monitorizar qualquer porta em qualquer número de máquinas.

Se você é ambicioso, convém pensar em transformar o objeto *serMailPort* em um array que informe o status de todos os seus servidores de correio.

Nosso objetivo neste capítulo não foi fornecer scripts para inserir imediatamente em seu ambiente. Em termos mais específicos, quisemos mostrar a você as possibilidades e fazer você pensar sobre como deve escrever scripts que ofereçam recursos elaborados de monitoramento personalizado. Se você estiver pensando criativamente sobre o que pode fazer com o SNMP, alcançamos nosso objetivo.

## MRTG

O *Multi Router Traffic Grapher* (MRTG) é uma ferramenta de análise de tendências com distribuição grátis e totalmente configurável, fácil de configurar e usar. Trata-se de um pacote surpreendentemente pequeno e leve porque não implementa uma interface do usuário muito pesada. Em vez disso, gera gráficos na forma de imagens GIF ou PNG; esses gráficos são incorporados a páginas HTML padrão. Por conseguinte, é possível exibir a saída do MRTG em qualquer navegador da Web gráfico e até mesmo tornar visíveis os respectivos relatórios na rede, usando um servidor da Web.

Embora o MRTG seja mais adequado à exibição de gráficos de utilização para interfaces de roteador, é possível configurá-lo para grafar aspectos como utilização da memória, média de carregamento e utilização do disco em equipamentos do servidor. O MRTG é útil principalmente para determinar quando um item “ultrapassar o limite” por um longo período de tempo, o que indica um problema de capacidade e necessidade de atualização. Por exemplo, você poderia detectar que uma interface T1 estoura o limite máximo no horário de pico e você precisa de um upgrade para um circuito maior, ou poderia descobrir que precisa adicionar mais memória a um servidor. De modo semelhante, o MRTG pode informar que as conexões da rede estão funcionando a uma fração da largura de banda disponível e que, por conseguinte, você pode eliminar alguns circuitos T1 e reduzir os custos de telecomunicações.

Alguns sites que usam o MRTG aplicam seus recursos de criação de gráficos defaults para o planejamento de capacidades e para o abastecimento. O MRTG não dispõe de ferramentas de dados estatísticos de alta precisão, necessárias para calcular informações de linha de base ou para prever a necessidade de um upgrade da rede. Entretanto, pode ser uma ferramenta muito útil para as empresas que não dispõem dos recursos necessários para adquirir um pacote completo de análise de tendências. As linhas de base e as projeções são imprescindíveis, mas os gráficos do MRTG podem oferecer um comportamento semelhante, de modo rápido; seus olhos são perfeitos para localizar comportamento e tendências característicos, mesmo que não fornecam a análise de dados estatísticos que sua diretoria aprovaria.

O MRTG tem algumas opções que permitem personalizar o modo de operação do produto. Está além do âmbito deste capítulo discutir cada opção; em vez disso, abordaremos como instalar o MRTG e utilizar seus recursos de criação de gráficos padrão. Também descreveremos a configuração do MRTG para obter informações do sistema em um servidor.

É importante entender que o MRTG não é uma solução para NMS. Embora seus recursos de criação de gráficos o tornem parecido com uma NMS, na realidade, é apenas um mecanismo de polling muito criterioso quanto à saída gerada. Executa as mesmas funções de *get* que uma NMS, mas sua missão não é detectar nem solucionar problemas. Não dispõe de capacidade de gerar alarmes ou processar traps, nem a opção de definir objetos. É elaborado apenas para oferecer uma visão gráfica do desempenho de sua rede. Se você está interessado em um pacote de NMS de fonte aberta, pesquise na Bluebird (<http://www.opennms.org>).

## Usando o MRTG

Para utilizar o MRTG, é necessário fazer o download do software e instalá-lo. O site da Web básico do MRTG é o <http://www.mrtg.org>. O link de download fornece acesso a um diretório mantido pelo inventor e principal desenvolvedor do MRTG, Tobias Oetiker (<http://ee-staff.ethz.ch/~oetiker/webtools/mrtg/pub/>). Esse diretório contém algumas versões mais antigas do MRTG, assim como a versão atual. Fizemos o download do arquivo *mrtg-2.9.10.tar.gz* (a versão para Unix) na lista. Abordaremos essa versão neste capítulo.

O MRTG requer quatro pacotes externos para executar: o Perl Version 5.004\_5 (no mínimo) e as bibliotecas *gd*, *libpng* e *zlib*. O MRTG é fornecido com uma implementação do SNMP baseada na linguagem Perl, para que você não se preocupe em obter e instalar quaisquer bibliotecas do SNMP. Você pode saber a versão do Perl existente em seu sistema (e se está instalada), digitando o comando *perl -v*. Esse comando pode exibir ou não diversas informações. Se exibir, a primeira linha será a versão da linguagem Perl instalada. Se você obter algum tipo de erro “command not found” (comando não encontrado), é possível que a Perl não esteja instalada. Seja como for, entre em <http://www.perl.com> para obter a versão mais recente da Perl.

A biblioteca *gd* é usada para gerar as imagens GIF exibidas pelo MRTG. Você pode fazer o download dessa biblioteca em <http://www.boutell.com/gd/>. Os outros dois pacotes, *libpng* e *zlib*, também são usados para controlar vários aspectos da criação de imagens gráficas. Encontram-se disponíveis em <http://www.libpng.org/pub/png/> e <http://www.info-zip.org/pub/infozip/zlib/>.

Após assegurar que a Perl, *gd*, *libpng* e *zlib* estão instaladas em sua máquina, faça o download e desempacote a versão Unix do MRTG, com os seguintes comandos:

```
[root][linuxserver] > cd /usr/local  
[root][linuxserver] > tar -zxvf mrtg-2.9.10.tar.gz
```

Uma vez desempacotada, alterne (com o comando `cd`) para o diretório criado (que deve ser o `mrtg-2.9.10`) e leia as dicas de instalação do arquivo `README`. Para construir o MRTG, execute três comandos:

```
[root][linuxserver] ~/mrtg-2.9.10> ./configure  
[root][linuxserver] ~/mrtg-2.9.10> make  
[root][linuxserver] ~/mrtg-2.9.10> make install
```

Esses três comandos geram muita saída, que omitimos. O comando `configure` procura no sistema as ferramentas necessárias para construir o MRTG. Ele informará os itens ausentes e onde consegui-los. Executar o `make` cria o MRTG mas não se preocupe em rodar esse comando se o comando `configure` falhar; o MRTG não será construído, a não ser que tudo tenha sido instalado e configurado corretamente. Por último, o `make install` instala o MRTG e os arquivos associados nas localizações adequadas. Mais uma vez, não se preocupe me executar o `make install` se o comando `make` anterior finalizou com erros. A localização default dos executáveis do MRTG é `/usr/local/mrtg-2/bin`. Convém adicionar esse diretório ao caminho de pesquisa.

Após construir o MRTG, especifique onde serão colocados os gráficos gerados. Como os gráficos do MRTG são elaborados para exibição em um navegador da Web, geralmente são armazenados em um diretório visível para um servidor da Web. Entretanto, não tem importância o local de armazenamento. O mais importante é quem deseja visualizar os gráficos. Provavelmente, você não quer que o mundo veja os dados estatísticos de sua rede. Em uma rede pequena, você pode guardar os gráficos em um diretório fora da visibilidade do servidor da Web e depois usar um navegador da Web para exibir os relatórios HTML no sistema de arquivos local. Em uma rede maior, outras pessoas (como outra equipe ou a administração da rede) podem precisar acessar os relatórios; para autorizar o acesso sem publicar os dados estatísticos de sua rede para o restante do mundo, convém configurar algum tipo de servidor de Web protegido. Seja qual for a velocidade, o próximo conjunto de comandos a ser executado é algo parecido com este:

```
[root][linuxserver] ~/mrtg-2.9.10> mkdir /mrtg/images  
[root][linuxserver] ~/mrtg-2.9.10> cp ./images/mrtg*.gif /mrtg/images/
```

O primeiro comando cria um diretório para armazenar os gráficos criados pelo MRTG. O segundo comando copia algumas imagens do MRTG no diretório recém-criado, para uso posterior em arquivos HTML. No restante deste capítulo, presumiremos que os gráficos se encontram armazenados em `/mrtg/images`.

Agora, você está preparado para configurar seu primeiro dispositivo a ser pesquisado, denominado *target* no MRTG. O MRTG usa um arquivo de configuração para informar os dispositivos a serem pesquisados, as opções a serem aplicadas à criação dos gráficos por ele gerados, etc. A sintaxe do arquivo de configuração é complexa mas o MRTG dispõe de uma ferramenta denominada `cfgmaker` para ajudar a construí-lo. Provavelmente, você precisará editar o ar-

quivo manualmente, mas é muito mais fácil iniciar com um modelo operacional. Examine a seguir como executar o *cfgmaker*:

```
[root] [linuxserver] ~/mrtg-2.9.10> setenv PATH
```

```
/usr/local/mrtg-2/bin:$PATH
```

```
[root] [linuxserver] ~/mrtg-2.9.10> cfgmaker --global 'WorkDir:
```

```
/mrtg/images' \
```

```
--output /mrtg/run/mrtg.cfg public@router
```

O primeiro argumento do *cfgmaker* define a variável *WorkDir* no arquivo de configuração. Essa variável informa ao MRTG onde armazenar os dados obtidos nos dispositivos que pesquisará. O segundo argumento especifica para onde a saída do *cfgmaker* será enviada; nesse caso, em */mrtg/run/mrtg.cfg*.

O último argumento especifica o dispositivo a ser pesquisado e a string de comunidade que será usada ao pesquisar esse dispositivo; seu formato é *community\_string@device*.

A saída do *cfgmaker* é uma combinação de comandos e HTML. Ele executa comandos *get-next* sobre o dispositivo especificado na linha de comando, para detectar a quantidade de interfaces do dispositivo, quais delas estão funcionando, quais encontram-se paralisadas etc. Ele percorre a árvore *iso.org.dod.internet.mgmt.mib-2.interfaces* (1.3.6.1.2.1.2) para descobrir o número total de interfaces nessa tabela. Em seguida, cria entradas lógicas que representam uma lista de dispositivos a serem pesquisados mas, na realidade, essa lista de dispositivos é um único dispositivo com cada número de interface especificado como um alvo. Por exemplo, *Ethernet0* encontra-se na quarta linha da tabela *interfaces* no roteador da Cisco, de modo que o *cfgmaker* criou uma entrada Target denominada *cisco.4*. Se essa interface ocupasse a segunda linha na tabela *interfaces*, a entrada Target seria denominada *cisco.2*.

Eis uma versão reduzida do arquivo *mrtg.cfg*:

```
WorkDir: /mrtg/images/
```

```
Target[cisco.4]: 4:public@cisco
MaxBytes[cisco.4]: 1250000
Title[cisco.4]: cisco (cisco): Ethernet0
PageTop[cisco.4]: <H1>Traffic Analysis for Ethernet0
</H1>
<TABLE>
<TR><TD>System:</TD><TD>cisco in Atlanta, Ga</TD></TR>
<TR><TD>Maintainer:</TD><TD></TD></TR>
<TR><TD>Interface:</TD><TD>Ethernet0 (4)</TD></TR>
<TR><TD>IP:</TD><TD>cisco ( )</TD></TR>
<TR><TD>Max Speed:</TD>
    <TD>1250.0 kBytes/s (ethernetCsmacd)</TD></TR>
</TABLE>
```

Compensa conhecer o formato do arquivo de configuração. As linhas de comentários começam com #; em um arquivo de configuração real, você en-

contrará algumas delas. A maioria das linhas contidas no arquivo são comandos ou fragmentos de HTML que serão usados nos arquivos de saída do MRTG. Os comandos do MRTG assumem o formato de *comando[chave]: opções*. Por exemplo, o comando para a terceira linha é Target, a chave é cisco.4 e as opções são 4:public@cisco. A chave é uma string de identificação que agrupa entradas no arquivo de configuração e fornece um nome de arquivo básico para o MRTG usar ao gerar gráficos e arquivos HTML. Em um site complexo, o MRTG poderia ser usado para monitorar dezenas de aparelhos, com centenas de interfaces; a chave mantém o arquivo de configuração em um tipo de ordem. As opções fornecem os parâmetros reais para o comando.

Com essas explicações, você deve entender o arquivo de configuração. A primeira linha especifica o diretório de trabalho em que o MRTG armazenará os gráficos e os arquivos HTML. Esse é um comando global, de modo que não é necessária uma chave. Geralmente, o diretório de trabalho fica armazenado em alguma posição na árvore de servidores da Web, de modo que os relatórios do MRTG sejam visualizados em um navegador da Web.

Definimos nosso diretório com /mrtg/images/. A terceira linha (Target) informa ao MRTG o dispositivo a ser pesquisado. O formato dessa opção é *interface:community\_string@device* ou, em nosso caso, 4:public@cisco. O dispositivo é especificado pelo respectivo nome do host ou endereço IP; já conhecemos as strings de comunidade. Como o MRTG é apenas uma ferramenta de obtenção de dados, basta a string de comunidade read-only. Interface especifica a interface no dispositivo a ser pesquisado, de acordo com a *ifTable* do dispositivo. Nesse caso, estamos pesquisando a interface 4 na *ifTable*.

A linha MaxBytes configura o valor máximo para os parâmetros que o MRTG lerá nessa interface. Por default, o MRTG lê o *ifInOctets* e *ifOutOctets*. Ele tenta atribuir um valor máximo justo, dependendo do tipo de interface, que ele deverá ler no próprio dispositivo. Como se trata de uma interface Ethernet, o MRTG define MaxBytes com 1250000. Title especifica o título da página HTML gerada para o gráfico. Por último, PageTop e as linhas seguintes informam ao MRTG o tipo de informação que deve ser posicionada no início da página HTML contendo os gráficos de utilização. O comando contém um código HTML real, gerado por *cfgmaker*.

Adicionalmente, essa entrada instrui o MRTG a procurar os objetos defaults (*ifInOctets* e *ifOutOctets*) na entrada 4 na tabela de interfaces do dispositivo *cisco*. Por conseguinte, o MRTG emitirá comandos *get* para as OIDs .1.3.6.1.2.1.2.2.1.10.4 (*iso.org.dod.internet.mgmt.mib-2.interfaces.ifEntry.ifInOctets.4*) e .1.3.6.1.2.1.2.2.1.16.4 (*iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifOutOctets.4*). Por default, o MRTG gerará os seguintes gráficos:

- Gráfico diário com médias de 5 minutos
- Gráfico semanal com médias de 30 minutos
- Gráfico mensal com médias de 2 horas
- Gráfico anual com médias de 1 dia

Quando você terminar, tente executar o MRTG manualmente, para verificar se existem problemas no script de configuração:

```
[root][linuxserver] ~/mrtg-2.9.10> mrtg /mrtg/run/mrtg.cfg
```

Se o MRTG não tiver problemas com seu arquivo de configuração, será executado sem erros de arquivo de configuração. Se realmente ocorrerem problemas, ele apresenta uma descrição bastante explicativa do problema. Na primeira execução do MRTG, ele reclamará por não conseguir encontrar os arquivos de log. Se você executar o MRTG três vezes, receberá mensagens parecidas com estas:

```
[root][linuxserver] ~/mrtg-2.9.10> mrtg /mrtg/run/mrtg.cfg
```

```
Rateup WARNING: /mrtg/run//rateup could not read the primary log file for cisco.4  
Rateup WARNING: /mrtg/run//rateup The backup log file for cisco.4 was invalid as well
```

```
Rateup WARNING: /mrtg/run//rateup Can't remove cisco.4.old updating log file  
Rateup WARNING: /mrtg/run//rateup Can't rename cisco.4.log to cisco.4.old  
updating log file
```

```
[root][linuxserver] ~/mrtg-2.9.10> mrtg /mrtg/run/mrtg.cfg
```

```
Rateup WARNING: /mrtg/run//rateup Can't remove cisco.4.old updating log file
```

```
[root][linuxserver] ~/mrtg-2.9.10> mrtg /mrtg/run/mrtg.cfg
```

```
[root][linuxserver] ~/mrtg-2.9.10>
```

Como você pode constatar, na primeira execução, o programa emite alguns erros. A segunda gerou somente um erro e a última execução ocorreu sem erros. Ao executar o MRTG pela primeira vez, esses erros são normais; não se preocupe com isso.

A próxima etapa é assegurar a execução do MRTG a cada cinco minutos. Não é necessário que o MRTG seja executado pelo usuário root (raiz); qualquer usuário pode fazê-lo. Adicione uma linha como a seguinte à entrada *crontab* do usuário adequado:

```
*/5 * * * * /usr/local/mrtg-2/bin/mrtg /mrtg/run/mrtg.cfg
```

Esse procedimento executará o MRTG a cada cinco minutos, todos os dias. Observe que a notação \*/5 é específica do Linux; em outros sistemas Unix, é necessário especificar a quantidade de vezes explicitamente (0,5,10,15,20,25,30,35,40,45,50,55). Se sua rede for muito grande, talvez você enfrente problemas se o MRTG não finalizar todas as suas tarefas de pesquisa antes de iniciar o próximo ciclo de polling. Se esse for o caso, talvez não seja eficiente definir um intervalo de polling a cada cinco minutos. Provavelmente, você precisará experimentar para definir um intervalo eficaz para seu ambiente.

## Exibindo gráficos

Após gerar alguns gráficos, você precisará examiná-los para verificar os resultados. Para facilitar a visualização dos gráficos, o MRTG dispõe de um script *indexmaker* que gera páginas de índice HTML. Eis como executar o *indexmaker* para um group de gráficos comuns:

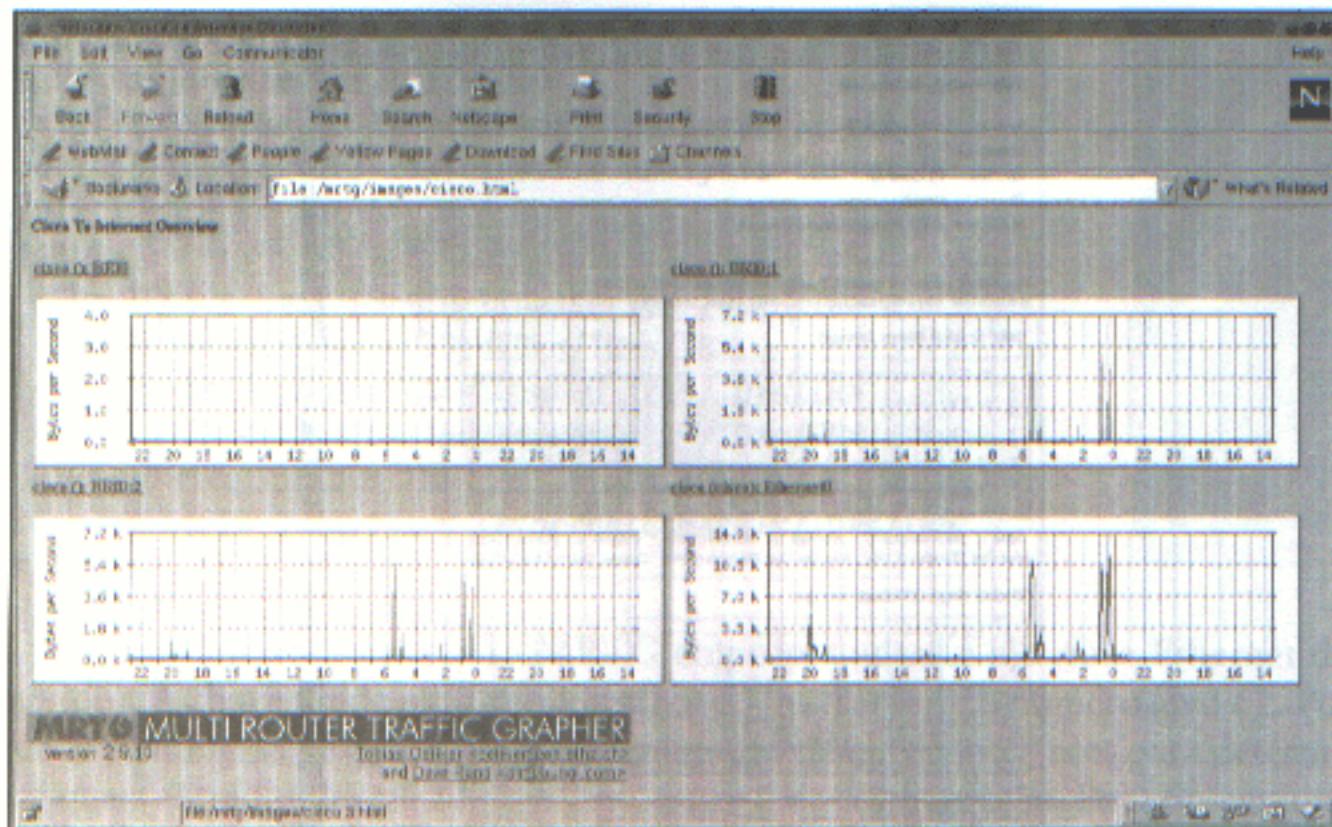
```
[root][linuxserver] ~/mrtg-2.9.10> indexmaker --title "Cisco to Internet" \
--filter name=~'cisco' --output /mrtg/images/cisco.html /mrtg/run/mrtg.cfg
```

Esse comando cria uma página de índice com o gráfico de médias de cinco minutos para cada alvo especificado em seu arquivo *mrtg.cfg*. Lembre-se de que o alvo é a interface em que você está obtendo dados. Se existirem quatro alvos para o roteador, existirão quatro gráficos no arquivo de índice, todos apontando para os gráficos de resumo diário, semanal, mensal e anual do alvo em questão. A opção *--title* informa ao *indexmaker* o título a ser usado no arquivo de índice. *--filter name=~cisco* permite selecionar alguns dos alvos contidos no arquivo *mrtg.cfg*, usando uma expressão regular: instruímos o *indexmaker* a encontrar todos os alvos que contêm a string *cisco*. A opção *--output* é o nome do arquivo de índice. O último argumento da linha de comando é o caminho completo para o arquivo de configuração. A Tabela 13-1 apresenta um resumo dessas opções assim como outras opções úteis ao *indexmaker*.

Tabela 13-1 Opções de linha de comando para o *indexmaker*

Opção	Descrição
<i>--title</i>	Especifique um título para a página HTML.
<i>--filter</i>	Especifique a expressão regular que será usada para procurar um alvo específico do arquivo <i>mrtg.cfg</i> . Esses alvos encontrados são usados para criar os arquivos de relatórios HTML.
<i>--output</i>	Indique o nome de caminho completo para o arquivo HTML a ser gerado. O default é a saída padrão.
<i>--sort</i>	Classifique o modo de exibição dos gráficos na página de índice.
<i>--columns</i>	Organize os gráficos na página de índice por x colunas. O default são 2.
<i>--width</i>	Defina a largura dos gráficos. Essa opção não é definida por default.
<i>--height</i>	Defina a altura dos gráficos. Essa opção não é definida por default.
<i>--show</i>	Escolha o gráfico que aparecerá na página de índice. O default é <i>day</i> . Outras opções são <i>week</i> , <i>month</i> , <i>year</i> e <i>none</i> .

Para exibir a lista inteira de opções para o *indexmaker*, execute o comando sem opções. A Figura 13-1 mostra o arquivo *cisco.html* gerado pelo *indexmaker* quando carregado em um navegador da Web.



*Figura 13-1 Visão geral de gráfico da Cisco*

Existem quatro gráficos na página, um para cada interface operacional (interfaces em funcionamento quando executamos o *cfgmaker*) em nosso roteador. Essa página possui links para outras páginas com informações mais detalhadas sobre as interfaces individuais; a Figura 13-2 mostra os gráficos de tráfego diário, semanal, mensal e anual para a interface *Ethernet0*.

O gráfico diário (que, na realidade, representa um período de 32 horas) é aquele que a maioria das pessoas têm interesse em visualizar. Mostra a média de cinco minutos do tráfego nessa interface específica. O tráfego de entrada (*ifInOctets*) é representado por uma linha verde; o tráfego de saída (*IfOutOctets*) é representado por uma linha azul. Se tivéssemos clicado em uma das outras interfaces na página de índice da Cisco (Figura 13-1), teríamos visto um gráfico semelhante.

Isso é tudo o há para examinar com relação à visualização de gráficos. O MRTG armazena os dados não processados obtidos, no formato de arquivo de texto simples mas, devido aos recursos inteligentes de rolagem de log, os arquivos de log não crescem desordenadamente; seus tamanhos continuam gerenciáveis mesmo que você use muito o MRTG.

## *Representação gráfica de outros objetos*

Por default, o MRTG pesquisa e representa graficamente as variáveis de MIB, *ifInOctets* e *ifOutOctets*, mas é possível pesquisar e grafar os valores de outros objetos, além de pesquisar diferentes tipos de dispositivos. Primeiramente, vamos instruir o MRTG a coletar octetos de entrada e saída em um servidor. Para isso, execute o seguinte comando:

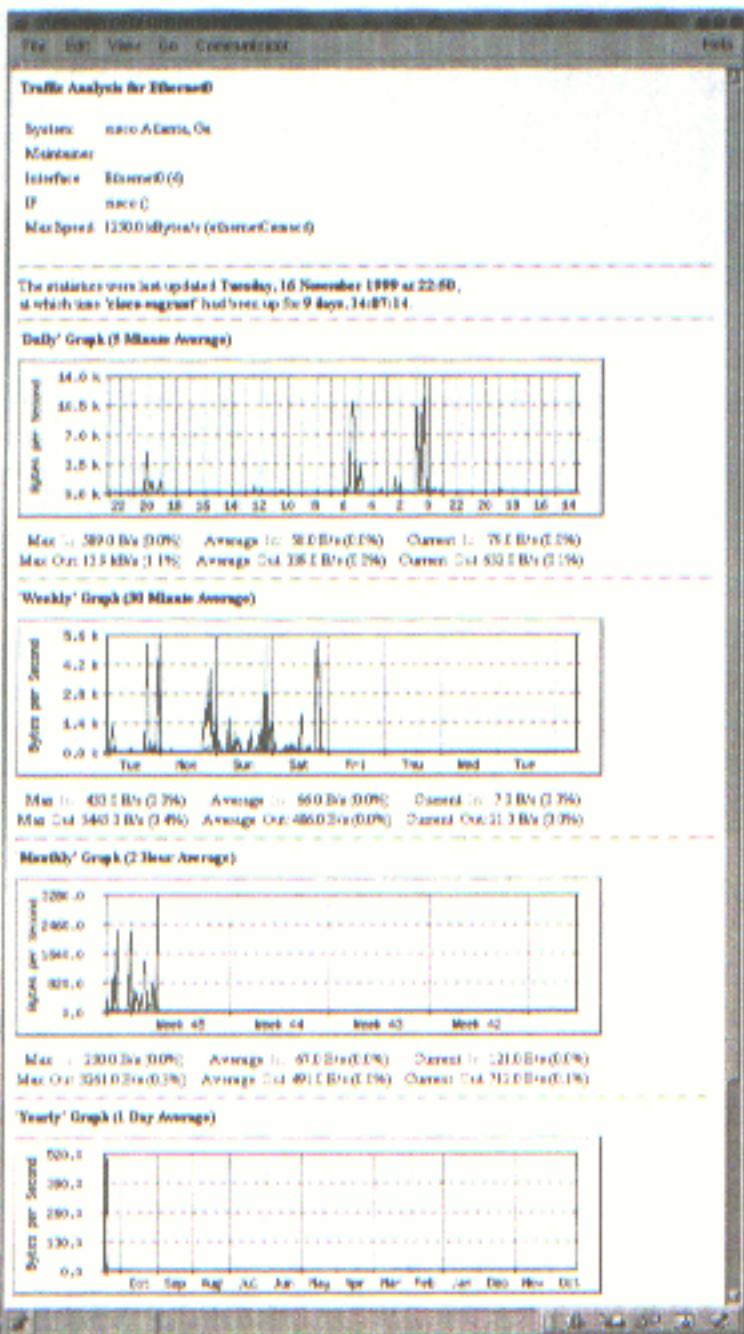


Figura 13-2 Gráficos diário, semanal, mensal e anual do Etherenet0

```
[root] [linuxserver] ~/mrtg-2.9.10> cfgmaker public@linuxserver >> \
/mrtg2/run/mrtg.cfg
```

Esse comando é quase idêntico ao executado anteriormente neste capítulo, exceto quanto à string de comunidade e o alvo\* (*public@linuxserver*). Anexamos a saída ao arquivo *mrtg.cfg*, em vez de especificar um arquivo de saída com a opção *--output*; esse procedimento permite incluir um novo host no arquivo de configuração existente, e não iniciar um novo arquivo. Como o arquivo existente já especifica um diretório de trabalho, também omitimos a opção pertinente (*--global 'WorkDir: .. '*). Esse comando *cfgmaker* adiciona algumas linhas ao arquivo de configuração, como as seguintes:

---

\* Verifique se o alvo está executando um agente de SNMP. Leia o Capítulo 7 para obter uma discussão sobre como configurar vários agentes de SNMP para o Unix e o Windows NT.

```

Target[linuxserver]: 2:public@localhost
MaxBytes[linuxserver]: 1250000
Title[linuxserver]: linuxserver(linuxserver): eth0
PageTop[linuxserver]: <H1>Traffic Analysis for eth0
</H1>
<TABLE>
<TR><TD>System:</TD><TD>linuxserver</TD></TR>
<TR><TD>Maintainer:</TD><TD></TD></TR>
<TR><TD>Interface:</TD><TD>eth0 (2)</TD></TR>
<TR><TD>IP:</TD><TD>linuxserver( )</TD></TR>
<TR><TD>Max Speed:</TD>
    <TD>1250.0 kBytes/s (ethernetCsmacd)</TD></TR>
</TABLE>

```

Essas linhas informam ao MRTG como pesquisar a interface Ethernet do servidor. A chave usada para essa interface é `linuxserver` e o número alvo é 2. Por que 2? Lembre-se de que o *cfgmaker* examina a tabela de interfaces para determinar quais entradas deverá adicionar ao arquivo de configuração. Por conseguinte, você verá um grupo de linhas como essas para cada interface no dispositivo, incluindo a interface de retorno. Na realidade, os números alvo são índices para a tabela de interfaces; nesse servidor, a interface de retorno tem o índice 1.

Criemos agora uma entrada para grafar o número de usuários conectados ao servidor e o número total de processos em execução. O MRTG pode grafar esses parâmetros mas é necessário especificar explicitamente as variáveis de MIB a serem grafadas. Além disso, especifique duas variáveis – o MRTG não grafará apenas uma. (Essa é uma limitação muito estranha mas, pelo menos, é consistente: os gráficos predefinidos mostram os octetos de entrada e saída.)

Primeiro, examinemos as variáveis de MIB que pretendemos grafar. As duas variáveis, `hrSystemNumUsers` e `hrSystemProcesses`, estão definidas como as OIDs `1.3.6.1.2.1.25.1.5.6.0` e `1.3.6.1.2.1.25.1.6.0`, respectivamente. O `.0` no final de cada OID indica que esses dois objetos são variáveis escalares, e não fazem parte de uma tabela. Ambos procedem da Host Resources MIB (RFC 2790), que define um grupo de objetos gerenciados para administração do sistema. (Alguns agentes executados em sistemas de servidor implementam essa MIB mas, infelizmente, os agentes da Microsoft e Solaris não o fazem.) As definições desses objetos são:

```

hrSystemNumUsers OBJECT-TYPE
  SYNTAX Gauge
  ACCESS read-only
  STATUS mandatory
  DESCRIPTION
    "The number of user sessions for which this host is storing state
     information. A session is a collection of processes requiring a
     single act of user authentication and possibly subject to collective
     job control."
 ::= { hrSystem 5 }

```

`hrSystemProcesses` OBJECT-TYPE

```

SYNTAX Gauge
ACCESS read-only
STATUS mandatory
DESCRIPTION
    "The number of process contexts currently loaded or running on
     this system."
 ::= ( hrSystem 6 )

```

A entrada que adicionamos ao arquivo de configuração é parecida com a seguinte:

```

Target[linuxserver.users]:1.3.6.1.2.1.25.1.5.0&1.3.6.1.2.1.25.1.6.0:
public@linuxserver
MaxBytes[linuxserver.users]: 512
Options[linuxserver.users]: gauge
Title[linuxserver.users]: linuxserver (linuxserver): Number of users and
processes
YLegend[linuxserver.users]: Users/Processes
LegendI[linuxserver.users]: &nbsp;Users:
LegendO[linuxserver.users]: &nbsp;Processes:
PageTop[linuxserver.users]: <H1>Number of users and processes</H1>
<TABLE>
    <TR><TD>System:</TD><TD>linuxserver<TD></TR>
    <TR><TD>Maintainer:</TD><TD></TD></TR>
    <TR><TD>IP:</TD><TD>linuxserver( )</TD></TR>
</TABLE>

```

Destacamos em negrito as modificações e inclusões efetuadas no arquivo de configuração. A primeira linha especifica o dispositivo que o MRTG deve pesquisar, juntamente com as duas OIDs (*hrSystemNumUsers* e *hrSystemProcesses*) que serão grafadas. Evidentemente, essa instrução é mais complexa do que a instrução Target examinada anteriormente; a sintaxe é *OID1&OID2:community\_string@device*. As OIDs devem ser separadas por um caractere de E-comercial (&). Por meio dessa sintaxe, você instrui o MRTG a grafar quaisquer duas variáveis de MIB de valor escalar.

Na linha seguinte, definimos MaxBytes com 512. Esse é o valor máximo para o gráfico; os valores acima de 512 são definidos com 512. (Esqueça os bytes; MaxBytes define apenas um valor máximo.) Para o número de usuários conectados, esse é um número alto; não deverão existir tantas pessoas conectadas a nosso sistema, simultaneamente. O mesmo se aplica ao número total de processos em execução no sistema. Você pode escolher valores relevantes para seu ambiente específico. Se você precisar de valores máximos distintos para cada objeto, substitua MaxBytes por duas linhas definindo MaxBytes1 e MaxBytes2.

O comando Options é uma novidade: permite modificar o modo como o MRTG trata os dados obtidos. A única opção que especificamos foi gauge, que instrui o MRTG a tratar os dados obtidos como dados Gauge, não como dados Counter. Lembre-se de que os dados Counter aumentam sempre, enquanto os dados Gauge não. Como as definições de MIB para os dois objetos especificam o tipo de dado Gauge, essa opção é pertinente.

As opções `YLegend`, `LegendI` e `LegendO` também são novas. `YLegend` modifica apenas o rótulo afixado ao eixo Y do gráfico. Como estamos grafando o número de usuários e processos, definimos a legenda com `Users/Processes`. É importante que a legenda seja curta; se for muito longa, o MRTG a ignorará silenciosamente e não imprimirá nada para o rótulo. `LegendI` muda a legenda usada abaixo do gráfico para a conhecida “variável de entrada” (neste caso, o número de usuários conectados ao sistema – lembre-se de que o MRTG pressupõe estar grafando octetos de entrada e saída). `LegendO` altera a legenda da “variável de saída” (o número total de processos em execução no sistema). A terminologia é terrível; basta lembrar que o MRTG sempre grava um par de objetos e a legenda de entrada está sempre relacionada ao primeiro objeto, enquanto a legenda de saída se relaciona ao segundo.

Após adicionar essa entrada ao arquivo de configuração e salvá-lo, o MRTG começará a obter os dados no dispositivo, sempre que for executado. Se você incluiu a entrada adequada no arquivo `crontab`, já definiu tudo.

Agora, usaremos o `indexmaker` para criar arquivos de índice intuitivos para os gráficos do servidor, assim como fizemos para os gráficos do roteador. O comando para criar um novo arquivo de índice é semelhante ao utilizado para criar um arquivo de índice da Cisco:

```
[root] [linuxserver] ~/mrtg-2.9.10> indexmaker --title "Linux Server" \
--filter name=='linuxserver' --output /mrtg/images/linux.html
/mrtg/run/mrtg.cfg
```

A Figura 13-3 mostra a página de índice dos gráficos do servidor, que contém somente dois gráficos: um para o tráfego na interface Ethernet e outro para o número de processos em execução em comparação com o número de usuários conectados ao sistema.

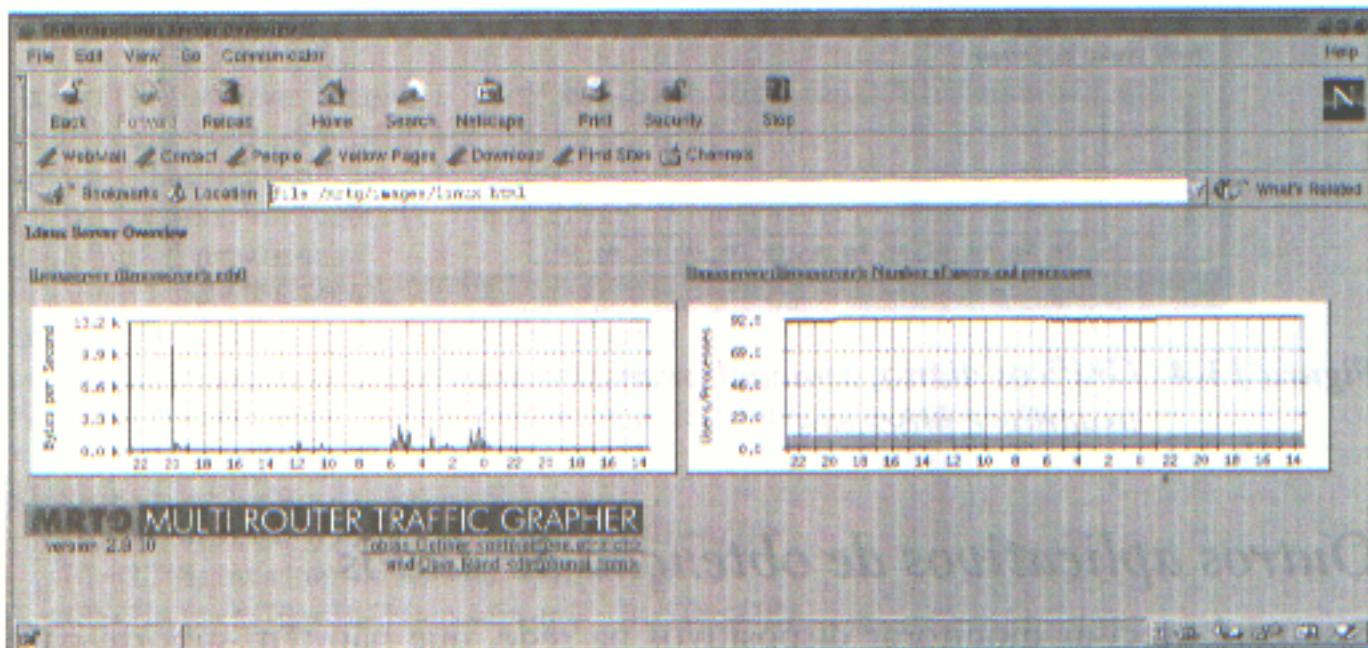


Figura 13-3 Gráficos de visão geral do Linux Server

A Figura 13-4 mostra os gráficos diário, semanal, mensal e anual para o número de usuários e processos conectados ao sistema.

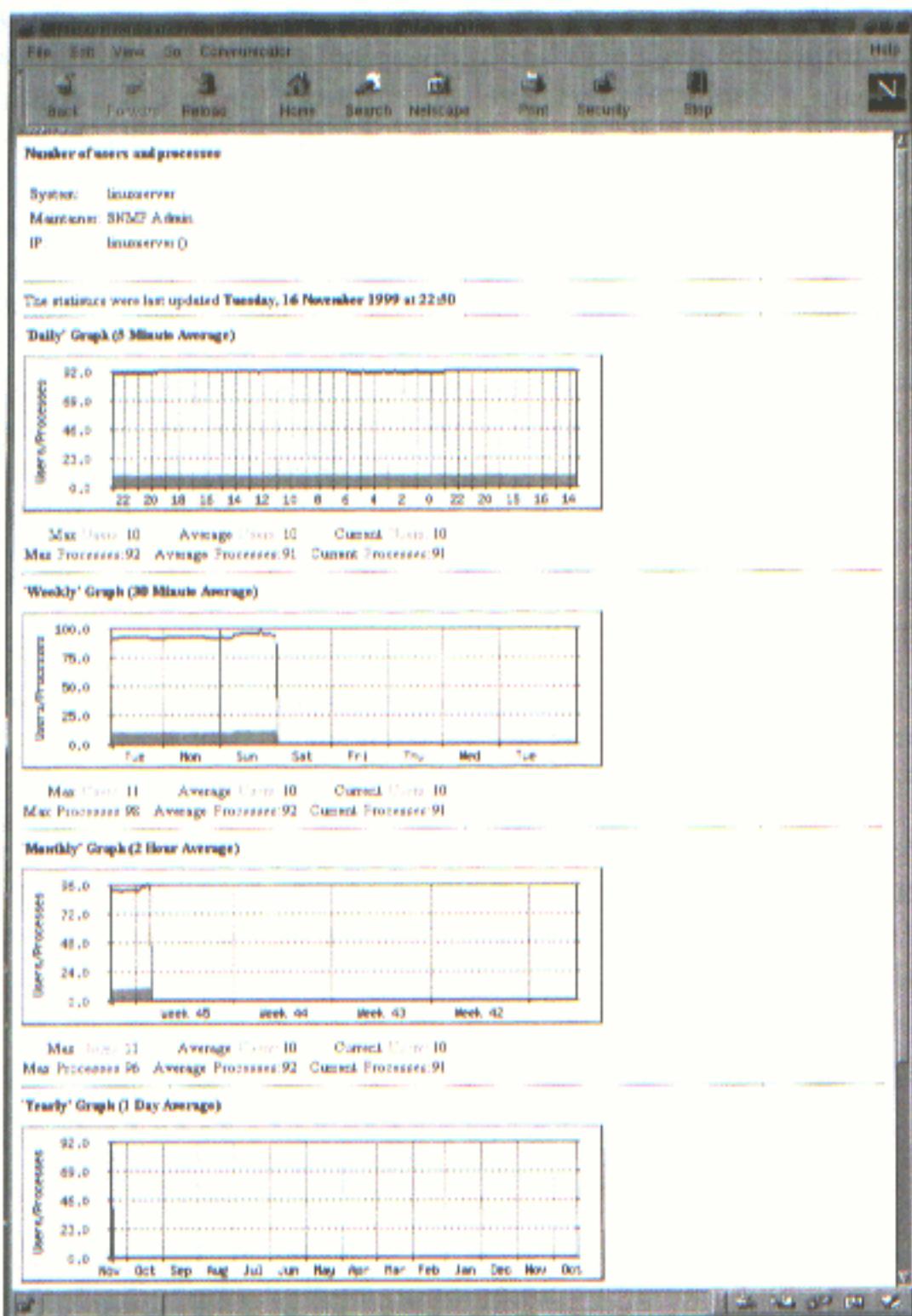


Figura 13-4 Gráficos diário, semanal, mensal e anual para o número de usuários e processos

## Outros aplicativos de obtenção de dados

E se você precisar monitorar dispositivos na rede, que não têm suporte para SNMP? O MRTG pode se encarregar da tarefa. Por exemplo, você pode ter um script em Perl que obtém dados estatísticos sobre utilização de algum dispositivo sem suporte para SNMP. Como é possível obter e grafar esses dados? Tornemos

essa perspectiva mais concreta. Suponha que você possui o seguinte script, `/usr/local/scripts/hostinfo.pl`, que informa o número de usuários e processos existentes no sistema:

```
#!/usr/bin/perl

$who = "/usr/bin/who | wc -l";
$ps = "/bin/ps -ef | wc -l";

chomp($numUsers = int(`$who`));
# Subtraímos dois porque ps gera um cabeçalho e o processo ps
# é considerado como em execução.

chomp($numProcesses = int(`$ps`) - 2);
print "$numUsers\n";
print "$numProcesses\n";

#
# O código a seguir imprime o tempo de funcionamento do sistema e nome do host.
# Esses dois itens devem ser incluídos em todo script criado e devem ser
# impressos por último.
#

chomp($uptime = `/usr/bin/uptime`);
print "$uptime\n";

chomp($hostname = `/bin/hostname`);
print "$hostname\n";
```

Esse script imprime quatro variáveis: o número de usuário e o número de processos (os dados que o MRTG deve obter) e o tempo de funcionamento do sistema e nome do host (exigidos pelo MRTG). Para que o MRTG execute esse script, é necessário editar manualmente o `mrtg.cfg`. Na realidade, a alteração é mais simples do que ocorreu em nosso exemplo anterior. Eis a nova entrada efetuada no `mrtg.cfg`, com as modificações destacadas em negrito:

```
Target[linuxserver.users]: `/usr/bin/perl /usr/local/bin/hostinfo.pl`
MaxBytes[linuxserver.users]: 512
Options[linuxserver.users]: gauge
Title[linuxserver.users]: linuxserver (linuxserver): Número de
usuários e processos
YLegend[linuxserver.users]: Users/Processes
LegendI[linuxserver.users]: &nbsp;Users:
LegendO[linuxserver.users]: &nbsp;Processes:
PageTop[linuxserver.users]: <H1>Número de usuários e processos
</H1>
<TABLE>
<TR><TD>System:</TD><TD>linuxserver</TD></TR>
<TR><TD>Maintainer:</TD><TD></TD></TR>
<TR><TD>IP:</TD><TD>linuxserver( )</TD></TR>
</TABLE>
```

Observe a inclusão de ``/usr/bin/perl /usr/local/bin/hostinfo.pl`` no comando Target. Você deve conhecer o restante. O MRTG interpreta o primeiro

valor que o script imprime (o número de usuários) como dados de entrada; o segundo valor (o número de processos) são os dados de saída. Ao gerar os gráficos, ele aplica as legendas de entrada e saída pertinentes (*LegendI* e *LegendO*).

## Atenção

Alguns dispositivos com recurso de SNMP mudam a ordem das interfaces na tabela *interfaces* sempre que uma nova placa de interface é inserida ou uma placa antiga é removida. Se você administra um ambiente de roteadores muito estáticos (por exemplo, dificilmente você adiciona ou remove placas de seus roteadores), os exemplos de configuração aqui apresentados devem funcionar bem para você. Mas nos ambientes atuais de redes em mudanças constantes, a estabilidade é rara. O comando *cfgmaker* do MRTG dispõe de uma opção de linha de comando, *--ifref*, para ajudar nesse problema. A opção não soluciona o problema mas permite gerar gráficos em que as interfaces são rotuladas com os respectivos endereços, descrições ou nomes; com essas informações, você não precisa memorizar se a interface 1 é a interface de rede local ou a conexão T1. A Tabela 13-2 resume a utilização da opção *--ifref*.

Tabela 13-2 Resumo de opções *--ifref*

Opção	Descrição
<i>--ifref=ip</i>	Identifique cada interface pelo respectivo endereço IP.
<i>--ifref=eth</i>	Use o endereço Ethernet para identificar a interface.
<i>--ifref=descr</i>	Use a descrição da interface para identificá-la.
<i>--ifref=name</i>	Use o nome da interface para identificá-la.

Portanto, para rotular interfaces com os respectivos endereços IP, execute o *cfgmaker* conforme demonstrado a seguir:

```
[root][linuxserver] ~/mrtg-2.9.10> cfgmaker --global 'WorkDir: /mrtg/images' \
--output /mrtg/run/mrtg.cfg --ifref=ip public@router
```

Leia o manual do *cfgmaker* que acompanha a documentação do MRTG.

## Como obter ajuda

O site da Web do MRTG, <http://www.mrtg.org>, oferece muitas informações e ajuda. Nessa página, você pode se inscrever na lista de endereçamento do MRTG. O MRTG também é discutido freqüentemente no Usenet newsgroup *comp.dcom.net-management*. Por último, não ignore a documentação do MRTG, localizada no subdiretório *doc* da distribuição do MRTG. Você encontrará uma documentação completa e abrangente, no formato texto e em HTML.



## Usando octetos de entrada e saída

Para ser compatível com o SNMP, um dispositivo IP deve ter suporte para os objetos da MIB-II (*iso.org.dod.internet.mgmt.mib-2*). A MIB-II contém a tabela *interfaces* (*mib-2.interfaces.ifTable.ifEntry*), que é um dos objetos mais úteis para a monitoração de redes. Essa tabela tem informações sobre as interfaces de rede do sistema. Alguns de seus objetos são:

### *ifDescr*

Uma descrição fornecida pelo usuário para a interface

### *ifType*

O tipo de interface (token ring, Ethernet etc.)

### *ifOperStatus*

Se a interface está funcionando, paralisada ou em alguma espécie de modo de teste

### *ifMtu*

Tamanho do maior pacote que pode ser enviado por meio da interface

### *ifSpeed*

Largura de banda máxima da interface

### *ifPhysAddress*

Endereço de baixo nível (hardware) da interface

### *ifInOctets*

Número de octetos recebidos pela interface

### *ifOutOctets*

Número de octetos enviados pela interface

Examinamos algumas partes dessa tabela em outros capítulos, mas evita-

mos discutir com mais detalhes os *ifInOctets* e *ifOutOctets*. A RFC 1213 informa que *ifOutOctets* e *ifInOctets* são o número total de octetos enviados e recebidos em uma interface, incluindo os caracteres de moldura.

Em alguns ambientes, essas informações são fundamentais. Empresas como os Provedores de Serviços de Internet (*Internet service providers – ISPs*) sobrevivem fornecendo a seus clientes largura de banda utilizável e, com isso, gastam muito tempo e dinheiro monitorando e avaliando suas interfaces, circuitos etc. Quando essas canalizações ficam cheias ou congestionadas, os clientes se irritam. Portanto, a principal pergunta é: como monitorar a largura de banda com eficiência? A resposta a essa pergunta é geralmente uma questão de vida ou morte.

As informações necessárias para responder a essa pergunta são fragmentadas. Primeiramente, você precisa conhecer o tipo de linha que está tentando monitorar. Sem essa informação, os números não fazem muito sentido. Em seguida, é necessário descobrir a velocidade máxima da linha e se ela é usada no modo full-duplex ou half-duplex. Na maioria dos casos, você conseguirá esses dois fragmentos de informação por meio do SNMP. O objeto *ifSpeed* definido na tabela *interfaces* da MIB-II fornece “uma estimativa da largura de banda atual da interface, em bits por segundo”. Você pode pesquisar esse objeto para descobrir a velocidade máxima da linha ou, pelo menos, o que o agente reconhece como velocidade máxima dessa linha. Entretanto, saiba que você precisa levar em consideração alguns fatores. Por exemplo, os roteadores da Cisco possuem larguras de banda máximas padrão para vários tipos de links, mas essas predefinições talvez estejam fora da realidade: por exemplo, a largura de banda predefinida para uma linha serial é de 1.544 Mbps, independentemente da velocidade real da linha. Para obter dados significativos, configure o roteador de modo a informar corretamente a largura de banda máxima. Ocasionalmente, os administradores de rede definem intencionalmente a largura de banda da interface com um número incorreto para testar caminhos de direcionamento de várias maneiras. Se esse for o caso, você terá problemas para obter dados significativos por meio do SNMP.)

É mais fácil obter informações confiáveis sobre o modo duplex da linha. As linhas seriais operam no modo full-duplex. Isso significa que podem enviar e receber informações simultaneamente (por exemplo, uma linha serial de 56 Kbps pode fazer upload e download a 56 Kbps, simultaneamente, para um total de 112 Kbps). Outros tipos de linhas, como a 10BaseT Ethernet, podem lidar apenas com o modo half-duplex. Em um ambiente comum da 10BaseT, a diferença entre fazer upload e download de dados é insignificante; a largura de banda total na linha está limitada a 10 Mbps de entrada e saída combinadas. Alguns dispositivos possuem placas de 10/100, o que torna a identificação ainda mais difícil.

Alguns fornecedores têm MIBs privadas que retornam o estado duplex. Por exemplo, o objeto Cisco a seguir retorna o estado duplex para uma interface no comutador modelo 2900: *iso.org.dod.internet.private.enterprises.cisco.ciscoMgmt.ciscoC2900MIB.c2900MIBObjects.c2900Port.c2900PortTable.c2900PortEntry.c2900PortDuplexStatus*.

A tabela à qual esse objeto pertence também contém um objeto que pode ser usado para alternar o estado duplex de uma interface. Esse objeto é útil se você tiver um dispositivo que está negociando incorretamente o half-duplex em vez de full-duplex; você pode usá-lo para impor a porta no estado duplex correto.

Assim que você descobrir a velocidade máxima da linha e o modo duplex, poderá calcular a porcentagem de utilização. Alguns produtos de NMS permitem criar expressões, denominadas fórmulas, que usam objetos da MIB como variáveis. O OpenView permite definir expressões no arquivo `$OV_CONF/mibExpr.conf`. A sintaxe desse arquivo é complexa. As expressões são escritas em notação posfixada.\* Por default, o arquivo contém algumas entradas; essas expressões são freqüentemente úteis e talvez nem precisem de ajustes\*\* para funcionar em seu ambiente. Eis a definição padrão da expressão `If%util`:

```
If%util \
"Percent of available bandwidth utilized on an interface\n\
Computed by:\n\
(Received byte rate + transmitted byte rate) * 8\n\
----- \n\
                interface link speed\n\
then converted to a percentage."\n\
.1.3.6.1.2.1.2.2.1.10. \
.1.3.6.1.2.1.2.2.1.16. \
+ \
8 \
* \
.1.3.6.1.2.1.2.2.1.5. \
/ \
100 \
*
```

Essa expressão está dividida em três partes: um nome de expressão, comentários e a expressão em si. Usaremos o nome da expressão dentro de `xnmgraph` para nossas definições de coleta de dados. Os comentários nos ajudarão a entender o que essa expressão realmente faz. A sintaxe da expressão é definida na manpage `mibExpr.conf (4)`. Em resumo, ela soma os valores de dois objetos da MIB (`ifInOctets` e `ifOutOctets`), multiplica por 8 para obter o número de bits percorrendo a interface, divide pela velocidade da interface (`ifSpeed`) e converte o resultado em uma porcentagem. Como você pode observar aqui, é possível dividir expressões em várias linhas, usando o conhecido caractere de escape (barra invertida) do Unix, no final de cada linha.

Após definirmos `If%util`, podemos usá-lo para plotar a utilização com o `xnmgraph`:

```
S /opt/OV/bin/xnmgraph -monochrome -c public -poll 5 -title Ifutil_Formula -mib \
If%util:CiscoRouter1a::::.1.3.6.1.2.1.2.2.1.2:::" CiscoRouter14a
```

\* Também citada como “notação polonesa invertida”. Em vez de escrever “`! + 2`”, você escreveria “`1 2 +`”.

\*\* O método recomendado de modificar `$OV_CONF/mibExpr.conf` é usar o `xnmxcollect` com o switch `-delExpr` ou `-loadExpr`.

É exibido um gráfico da utilização percentual de cada interface no dispositivo CiscoRouter14a. Observe que é possível utilizar um nome de expressão como o primeiro dos argumentos separados por caracteres de dois-pontos, no comando *xnmgraph*.

Antes de começar a utilizar o *If%util* para medir a organização inteira, observe que essa expressão avalia somente as linhas half-duplex – isto é, ela compara a soma dos octetos de entrada e saída à capacidade da linha. Qualquer linha full-duplex grafada com esse cálculo parecerá incorreta. Para tirar uma prova, considere uma linha serial full-duplex com uma velocidade máxima de 500 Kbps em cada direção enviando atualmente 125 Kbps e recebendo 125 Kbps. A fórmula de *If%util* indica a utilização de 50%, que é incorreta: a linha se encontra realmente em 25% da capacidade. Para uma linha full-duplex, faz mais sentido efetuar cálculos separados para os dados de entrada e saída. Assim, você terá uma representação mais eficiente do que a rede está fazendo, porque no modo full-duplex a taxa dos dados de entrada não é afetada pelos dados de saída. Examine a seguir expressões revisadas para a utilização do envio (*WANIf%SendUtil*) e a utilização do recebimento (*WANIf%RecvUtil*):

```
WANIf%SendUtil \
"% utilização da interface em (ifOutOctets * 8 * 100) / ifSpeed"\ \
.1.3.6.1.2.1.2.2.1.16. \
8 \
* \
100 \
* \
.1.3.6.1.2.1.2.2.1.5. \
/ \
WANIf%RecvUtil \
"% de utilização da interface em (ifInOctets * 8 * 100) / ifSpeed"\ \
.1.3.6.1.2.1.2.2.1.10. \
8 \
* \
100 \
* \
.1.3.6.1.2.1.2.2.1.5. \
/
```

Examinemos agora alguns gráficos reais. Grafamos diferentes expressões e objetos da MIB ao mesmo tempo, para uma interface Ethernet 10BaseT (half-duplex). Criamos um tráfego na interface e obtivemos os resultados. Eis o script que gera os gráficos:

```
/opt/OV/bin/xnmgraph -monochrome -c public -poll 5 -title \
Cisco_Private_Local_Mib -mib \
".1.3.6.1.4.1.9.2.2.1.1.6:CiscoRouter1a:4::::1.3.6.1.2.1.2.2.1.2:::, \
.1.3.6.1.4.1.9.2.2.1.1.8:CiscoRouter1a:4::::1.3.6.1.2.1.2.2.1.2:::" \
CiscoRouter1a &
/opt/OV/bin/xnmgraph -monochrome -c public -poll 5 -title Ifutil_Formula \
-mib "If%util:CiscoRouter1a:4::::1.3.6.1.2.1.2.2.1.2:::" CiscoRouter1a &
```

```

./opt/OV/bin/xnmgraph -monochrome -c public -poll 5 -title \
WANIfRecvUtil_Formula -mib \
"WANI%RecvUtil:CiscoRouterla:4::::1.3.6.1.2.1.2.2.1.2:::" CiscoRouterla &

./opt/OV/bin/xnmgraph -monochrome -c public -poll 5 -title
WANIfSendUtil_Formula -mib \
"WANI%SendUtil:CiscoRouterla:4::::1.3.6.1.2.1.2.2.1.2:::" CiscoRouterla &

./opt/OV/bin/xnmgraph -monochrome -c public -poll 5 -title ifInOctets -mib \
".1.3.6.1.2.1.2.2.1.10:CiscoRouterla:4::::1.3.6.1.2.1.2.2.1.2:::" \
CiscoRouterla &

./opt/OV/bin/xnmgraph -monochrome -c public -poll 5 -title ifOutOctets -mib \
".1.3.6.1.2.1.2.2.1.16:CiscoRouterla:4::::1.3.6.1.2.1.2.2.1.2:::" \
CiscoRouterla &

```

A Figura A-1 mostra os objetos da MIB, *.iso.org.dod.internet.private.enterprises.cisco.local.interfaces.lifTable.lifEntry.locIfInBitsSec* e *.iso.org.dod.internet.private.enterprises.cisco.local.interfaces.lifTable.lifEntry.locIfOutBitsSec*. São objetos da MIB privada da Cisco que informam a taxa de dados de entrada e saída de uma interface, em bits por segundo.

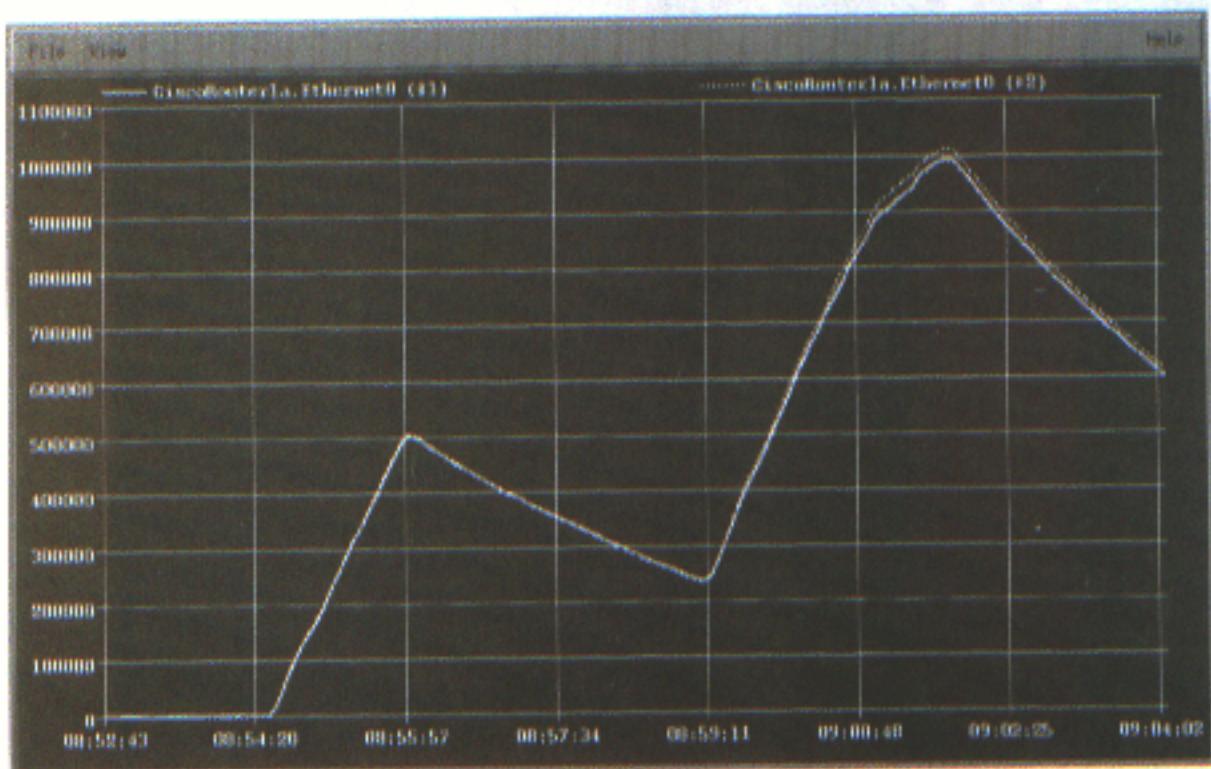


Figura A-1 Gráfico de objetos da MIB privada da Cisco

O gráfico seguinte, visualizado na Figura A-2, mostra a expressão *If%util*. É muito diferente. A diferença reside no fato de que a Cisco usa média de decaimento de 5 minutos para esses dois objetos. Isso pode ser bom e mau. A média de decaimento pode evitar a visualização dos picos e vales locais na utilização. Nesse exemplo, vemos dois picos de utilização, que a média de decaimento marca por um período mais longo de tempo. Ao utilizar MIBs privadas de fornecedor, procure saber como o fornecedor calcula esses números.

As Figuras A-3 e A-4 mostram a expressões WANIf%RecvUtil e WANIf%SendUtil. Como se trata de uma interface half-duplex, não é necessário examinar cada direção (entrada e saída) separadamente, mas talvez ajude verificar se o caminho de recebimento ou de envio ultrapassou o limite máximo. A comparação entre as Figuras A-3 e A-4 indica que existe mais tráfego de saída do que de entrada.

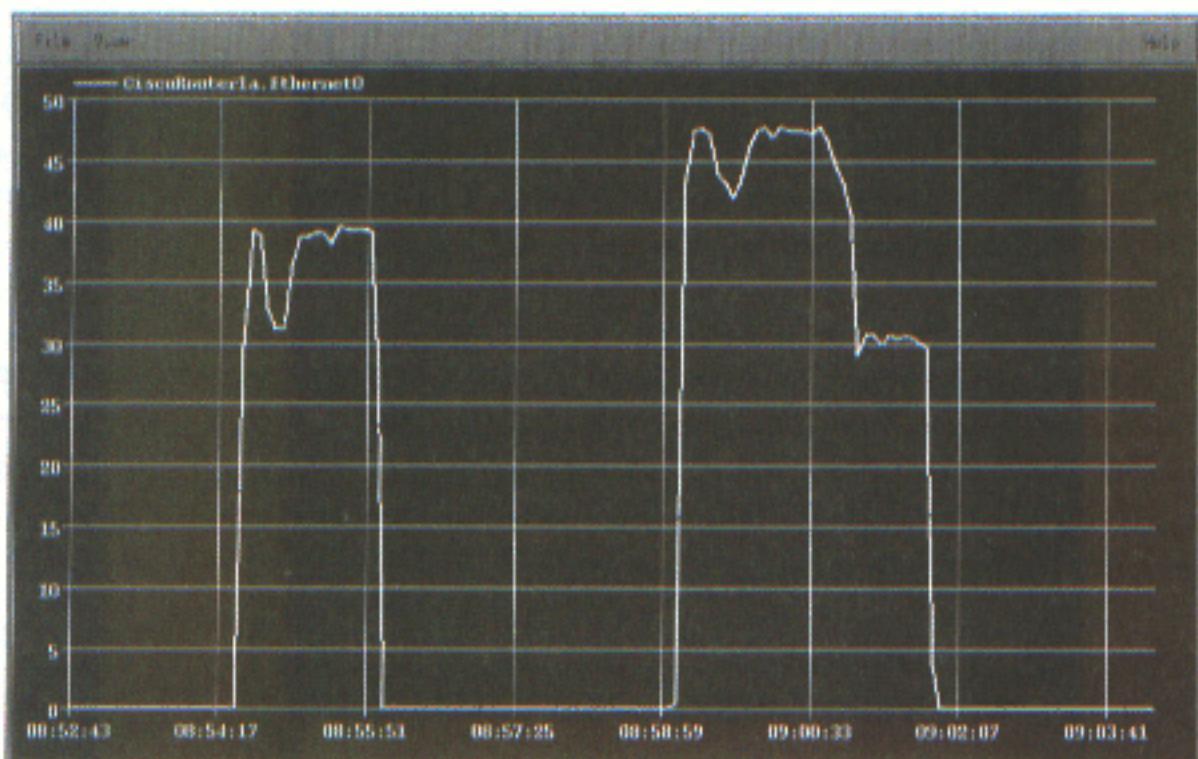


Figura A-2 Gráfico de If%util



Figura A-3 Gráfico de WANIf%RecvUtil



Figura A-4 Gráfico de WANIf%SendUtil

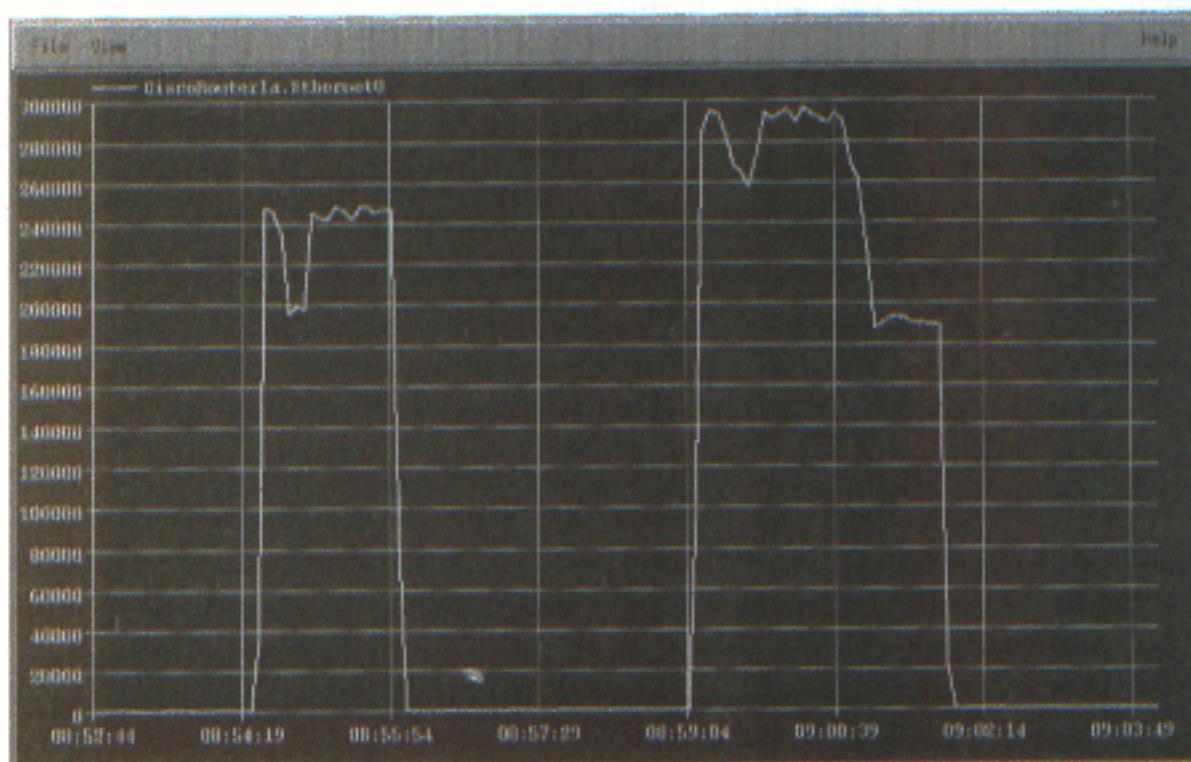


Figura A-5 Gráfico de ifInOctets

Os objetos da MIB-II padrão, *ifInOctets* e *ifOutOctets*, estão grafados nas Figuras A-5 e A-6. Lembre-se de que esses objetos não exibem bits por segundo. Mais uma vez, esses objetos indicam que há mais tráfego de saída do que de entrada. Os gráficos de octetos nas Figuras A-5 e A-6 mostram uma imagem em tempo real, como as expressões de WAN mas diferentes de objetos da MIB privada da Cisco.

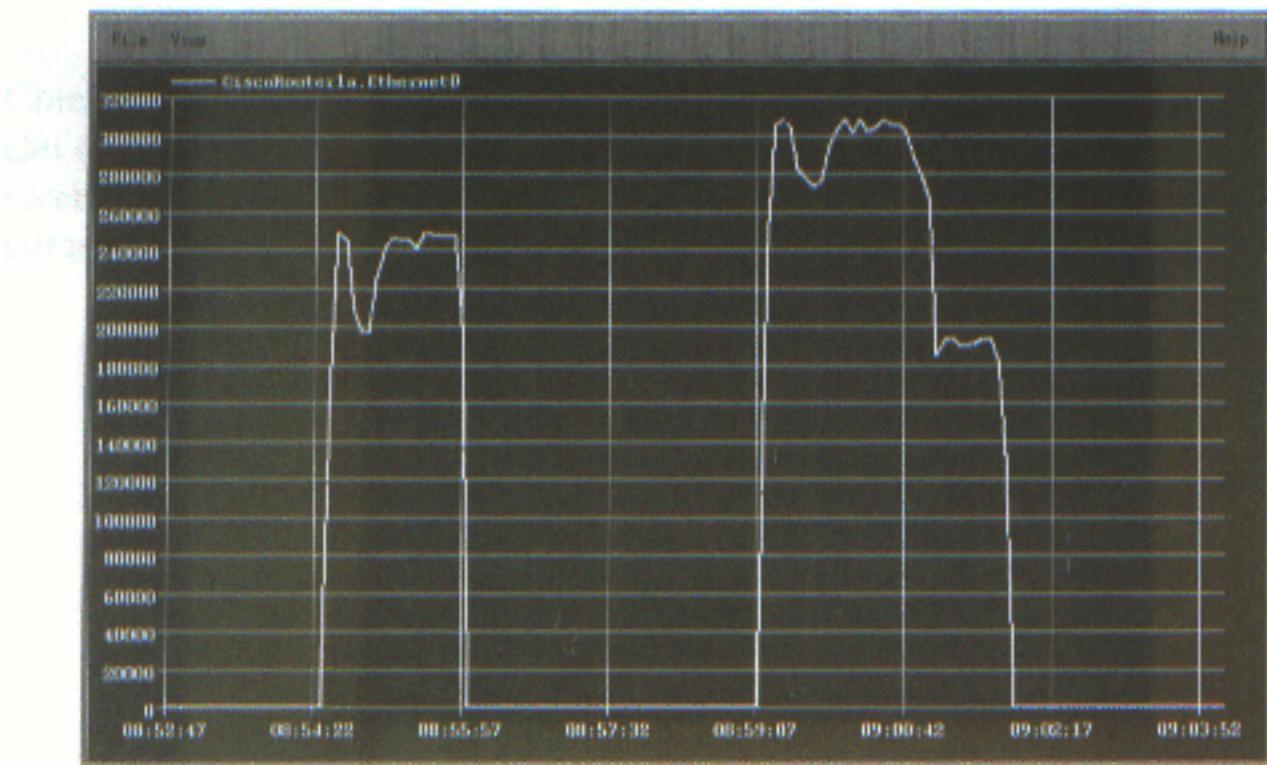


Figura A-6 Gráfico de *ifOutOctets*

Tente fazer uma idéia do que você está procurando, antes de começar a escrever as expressões. Você está tentando descobrir quem está inundando a rede ou está apenas pesquisando uma média semanal? Independentemente do que você esteja grafando, certifique-se de pesquisar os objetos da MIB do dispositivo antes de começar a gerar gráficos que possam parecer corretos mas têm dados sem sentido. Verifique novamente as variáveis sempre que você criar novos gráficos.

Lembre-se de que alguns dispositivos podem alternar automaticamente do modo full-duplex para half-duplex. Esteja atento ao ponto de saturação de sua interface, que é aquele ponto em que não é mais possível enviar nem receber qualquer tráfego. Esse ponto de saturação é indicado nos gráficos por uma linha horizontal contínua e só pode realmente ser visto nos intervalos prolongados de tempo. Assim, enquanto existirem algumas linhas horizontais nos gráficos deste apêndice, obviamente não estaremos próximos à capacidade da interface.

Para utilizar gráficos como esses, certifique-se de planejar em função de uma média e não com base nas exceções (picos). Todas as redes têm ocasionalmente picos de tráfego; a menos que você prefira altos investimentos em telecomunicações, muito acima do que é realmente necessário, planeje sua rede para que ela atenda às atividades médias cotidianas, não a picos eventuais.

## *Considerações adicionais sobre o NNM da OpenView*

Você já deve conhecer o NNM do OpenView e seus utilitários de apoio. Embora alguns administradores de rede possam trabalhar com as informações básicas do OpenView fornecidas neste livro, há muito mais para conhecer. Configurar o NNM com suas ferramentas personalizadas torna muito mais eficaz a utilização do produto.

Mesmo que não possamos discutir todos os recursos do NNM neste apêndice, abordaremos os seguintes temas:

- Utilização de dados externos com o *xnmgraph*
- Inclusão de itens de menu adicionais no menu do NNM
- Criação de perfis do NNM para usuários diferentes
- Uso do NNM como um dispositivo de comunicação centralizada

### *Utilização de dados externos*

O Capítulo 9 apresentou o comando *xnmgraph*, mas citou rapidamente seus recursos. Um recurso particularmente útil é a possibilidade de grafar dados de fontes externas. Para saber como é possível grafar dados externos, gire primeiramente um gráfico de qualquer tipo – por exemplo, um dos gráficos criados no Capítulo 9 – e salve os dados em um arquivo. Em seguida, examine o conteúdo do arquivo. Cada arquivo de saída contém um tutorial resumido que ensina a reexibir o gráfico. Consulte \$APP\_DEFS/Xnmgraph, que contém as definições padrão do *xnmgraph*.

Examine a seguir uma tabela criada manualmente, copiando o formato de um arquivo de dados padrão do *xnmgraph*. Os pontos de dados estão organizados em fluxos (*streams*). Um *fluxo* é um grupo de dados plotados como uma única curva no gráfico. Todos os fluxos existentes no arquivo serão combinados em um gráfico individual com várias curvas. O StartTime é ignorado. O StopTime | 253

fornecer o valor do eixo X (horizontal) e o Value determina o valor do eixo Y (vertical):

```
# /tmp/data1
#
# Stream Number StartTime      StopTime          Value
# -----
#
# Start of Stream 1
#
1      0      04.28.2001-12:32:16    7
1      0      04.28.2001-12:32:20    3
1      0      04.28.2001-12:32:24   23
1      0      04.28.2001-12:32:28    4
1      0      04.28.2001-12:32:31    7
1      0      04.28.2001-12:32:35   12
1      0      04.28.2001-12:32:39    1
#
# Start of Stream 2
#
2      0      04.28.2001-12:32:16   17
2      0      04.28.2001-12:32:20   21
2      0      04.28.2001-12:32:24   8
2      0      04.28.2001-12:32:28  28
2      0      04.28.2001-12:32:31   2
2      0      04.28.2001-12:32:35  22
2      0      04.28.2001-12:32:39   9
```

O comando *xnmgraph* a seguir exibe nosso arquivo de dados. Observe que usamos números de fluxo, precedidos por sinais de menos, em vez de IDs de objetos. O sinal de menos indica que o fluxo pode assumir valores negativos. Se o número do fluxo estiver precedido pelos símbolos de + ou =, o *xnmgraph* usará o valor absoluto de todos os números negativos contidos no arquivo.

```
cat /tmp/data1 | xnmgraph -mib "-1:Stream One::::::,-2:Stream Two::::::"
```

A Figura B-1 mostra o resultado desse comando. Se seu gráfico parece cortado, clique com o botão direito do mouse sobre ele e, em seguida, com o botão esquerdo em "Show All." Uma opção no menu View permite gerar um gráfico em preto-e-branco, que geralmente funciona melhor com um pequeno número de fluxos.

Uma vez que podemos converter os dados em um formato que o *xnmgraph* pode exibir, vejamos se podemos gerar alguns gráficos a partir da saída do utilitário *vmstat* do Unix, que todos os administradores do Unix devem conhecer; esse utilitário fornece diversas informações sobre a memória do sistema, em um formato estranho. Eis o tipo de saída gerada pelo *vmstat*:

procs	memory	page	disk	faults	cpu
r b w	swap free mf pi po fr de sr s6 s2 sd	in	sy cs us sy id		
0 4 0	5431056 33672 1 2371 0 8 8 0 0 0 18 18 2 2161 5583 4490 17 14 69				
0 2 0	5430912 33576 1 2499 0 20 20 0 0 0 1 1 0 2997 8374 7030 25 18 58				

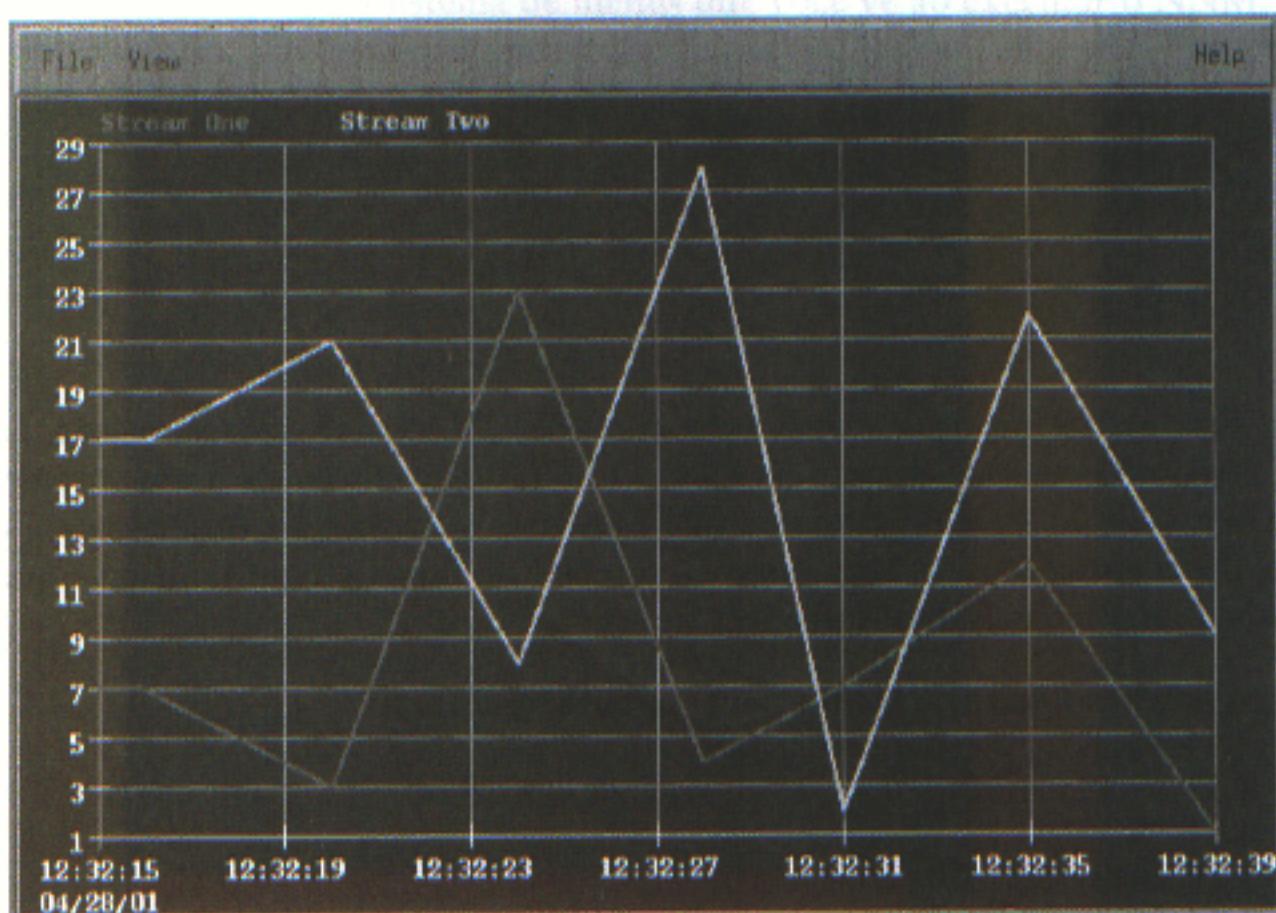


Figura B-1 Exemplo de gráfico do OpenView

0 2 0	5431296 33824 0 179 4 0 0 0 0 0 0 0 1 2587 3990 6379 18 8 74
0 0 0	5431240 33792 1 2460 4 8 8 0 0 0 1 1 0 2909 7768 7080 25 18 57
0 3 0	5431216 33768 1 2359 0 12 12 0 0 0 2 2 0 1934 5057 3818 18 13 70
0 0 0	5431288 33824 0 136 0 0 0 0 0 0 0 1 1842 2190 3803 13 5 82
0 2 0	5431216 32920 2 1189 0 3196 3176 0 0 0 0 0 4 2734 9980 5642 24 11 65
0 4 0	5431032 32352 8 1571 0 3100 3044 0 0 0 2 2 5 2763 7767 5817 22 15 63

Imagine analisar 10.000 linhas dessa saída e tentar descobrir as tendências (min/avg/max) em qualquer parâmetro específico. Não é fácil. Mas com o auxílio de um script em Perl, podemos transformar esses dados em um arquivo de entrada do *xnmgraph*. Veja como ficaria o script em Perl:

```

#!/usr/local/bin/perl
# Filename: /usr/local/bin/perl_scripts/cputimes

$|++; # Unbuffer the output!

open(VMSTAT,"/bin/vmstat 2 |") || die "Can't Open VMStat";
while($CLINE=<VMSTAT>)
{
    ($null,$r,$b,$w,$swap,$free,$re,$mf,$pi,$po,$fr,$de,$sr,$aa,$dd1,\
```

```

$dd2,$f0,$in,$sy,$cs,$us,$sycpu,$id) = split(/\s+/, $CLINE);
if (($id) && ($id ne "idle"))
{
    $DATE = `date +%m.%d.%y-%H:%M:%S`;
    chomp $DATE;
    print "1 0 $DATE $us \n";
    print "2 0 $DATE $sycpu \n";
    print "3 0 $DATE $id \n";
}
sleep 2;
}

```

Esse script imprime a utilização atual da CPU, como uma porcentagem, nos estados User (\$us), System (\$sycpu) e Idle (\$ide); o fluxo 1 é a porcentagem de User, o fluxo 2 é a porcentagem do System e o fluxo 3 é a porcentagem de Idle. O primeiro item em cada linha é o número do fluxo (stream); observe que é possível intercalar os dados dos três fluxos:

```

[root][nms] /> /usr/local/bin/perl_scripts/cputimes
1 0 8.14.99-21:00:22 6
2 0 8.14.99-21:00:22 3
3 0 8.14.99-21:00:22 92
1 0 8.14.99-21:00:24 0
2 0 8.14.99-21:00:24 0
3 0 8.14.99-21:00:24 100
1 0 8.14.99-21:00:26 1
2 0 8.14.99-21:00:26 0
3 0 8.14.99-21:00:26 98
1 0 8.14.99-21:00:28 1
2 0 8.14.99-21:00:28 0
3 0 8.14.99-21:00:28 99

```

O comando a seguir gera um gráfico com base na saída do script:

```
/usr/local/bin/perl_scripts/cputimes | xnmgraph -title "CPU Time" -mib \
"+1:User:::::,+2:System:::::,+3:Idle:::::"
```

Embora esse gráfico seja baseado em dados reais, é comum salvar os dados em um formato adequado e escrever um script que obtém dados de histórico de seus logs e os imprime com o *xnmgraph*.

## *Inclusão de um menu no NNM*

Com uma caixa de ferramentas de scripts, adicioná-los a um menu do NNM facilita ainda mais o acesso e a execução. Esse truque pode ser útil principalmente para utilizar a interface gráfica do NNM.

O segredo para incluir menus personalizados é o diretório \$OV\_REGISTRATION/C. (\$OV\_REGISTRATION contém diretórios para todas as linguagens disponíveis em seu sistema; C é o diretório para a linguagem default e, provavelmente, é o local onde você deve iniciar.) O diretório C possui todos os arquivos que integram o sistema de menus que você vê ao executar o NNM. Por exemplo, o arquivo ovw contém as conhecidas opções da janela principal (New, Open, Refresh etc.).

Examinemos o arquivo \$OV\_REGISTRATION/C/ovsnmp/xnmloadmib. É muito mais visualizar como incluir um comando externo em um menu. Vamos acessar imediatamente e criar um menu de dois níveis, com duas opções de menu:

```
Application "Graph Menu"
{
    Menubar <100> "Local_Graphs" _p
    {
        <100> "Network"      _N f.menu "network_menu";
    }

    Menu "network_menu"
    {
        <90> "5 Minute CPU"   _M f.action "5mincpu";
        <90> "Bits In and Out For All Up Interfaces" \
              _B f.action "bit_for_all_up";
    }

    Action "5mincpu" {
        Command "/opt/OV/local/scripts/Cisco_5min_cpu \
                  \${OVwSelections}\\"";
        MinSelected    1;
        MaxSelected    7;
        SelectionRule  (isSNMPSupported || isSNMPProxied) ;
    }

    Action "bit_for_all_up" {
        Command "/opt/OV/local/scripts/Cisco_Line_Up_Bits \
                  \${OVwSelections}\\"";
        MinSelected    1;
        MaxSelected    3;
        SelectionRule  (isSNMPSupported || isSNMPProxied) ;
    }
}
```

Crie um arquivo dentro do \$OV\_REGISTRATION/C e insira a listagem do código anterior. Depois disso, execute o ovw com o switch -verify, que procura erros.\* Você verá os erros ou avisos sobre o novo item de menu mas, se você obter sucesso, verá um item parecido com o menu da Figura B-2.

\* Não deixe arquivos de backup armazenados em quaisquer diretórios porque o NNM considera cada arquivo com seriedade. Os arquivos de backup ou redundantes gerarão avisos quando você executar o ovw.



O NNM pode ser seletivo em relação aos arquivos de registro. Se você não vir o menu, tente o truque *ovw -verify*. Se não forem revelados os erros, retire algumas entradas e reinicie o *ovw*. Continue fazendo isso até que os itens apareçam. Você também deve dividir os itens de seus menus em vários arquivos. Não coloque todos os menus e ações em um único arquivo. Quanto maior o número de arquivos, mais fácil será diagnosticar e solucionar problemas relacionados aos novos itens de menu.

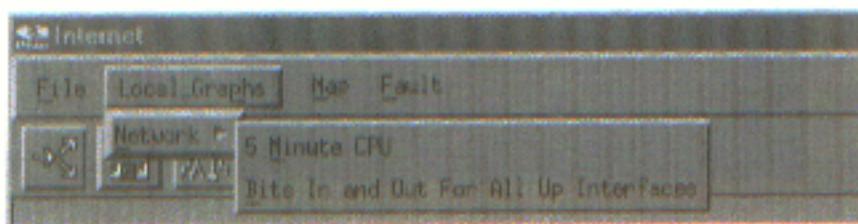


Figura B-2 Um novo menu

Consideremos alguns aspectos comuns existentes em nosso arquivo de registro:

- Cada menu e item de menu estão associados a um atalho de teclado que permite que o usuário o acesse. O caractere acionador é precedido por um caractere de sublinhado. Por exemplo, no menu “Local\_Graphs Network”, você pode pressionar “M” para ir para o item “5 Minute CPU”.
- Cada item de menu tem um número de prioridade entre símbolos de maior e menor-que. Isso permite controlar a ordem de exibição dos itens. Os itens com prioridade mais alta aparecem em primeiro lugar no menu; os itens de mesma prioridade estão listados na ordem em que aparecem no arquivo. Por exemplo, se reduzirmos a prioridade do item “5 Minute CPU”, de <Z> para <80>, ele aparecerá depois do item “Bits In and Out”, porque o item de prioridade mais alta vem primeiro.

A entrada Menubar contém os menus que aparecerão na barra de menus inicial do NNM. Usamos a função f.menu para chamar um submenu. O código a seguir mostra como poderíamos ter usado f.action para chamar uma ação imediatamente:

```
Menubar <precedence> "menubar Label" _MnemonicChar
{
    <precedence> "SubMenu Label" _MnemonicChar f.menu "menu-name"
    <precedence> "Action Name" _MnemonicChar f.action "action-name"
}
```

Um Menu é parecido e se comporta como uma barra de menus (ou menu) que o contém, com algumas diferenças. Os Menus não declaram caracteres mneômicos nem prioridade; esses são definidos pelo menu contínuo ou pela barra de menus. O *menu-name* é o nome de ligação que aparece depois de f.menu.

```
Menu "menu-name"
{
    <precedence> "SubMenu Label" _MnemonicChar f.menu "menu-name"
    <precedence> "Action Name" _MnemonicChar f.action "action-name"
}
```

As Actions são chamadas exatamente como os Menus. O *action-name* é o nome de ligação de uma ação chamada quando esse nome é selecionado em um item anterior (um Menu ou Menubar):

```
Action "action-name"
{
    Command "/opt/0V/local/scripts/Cisco_5min_cpu
\"${OVwSelections}\";
    MinSelected    1;
    MaxSelected    7;
    SelectionRule  (isSNMPSupported || isSNMPProxied);
}
```

Existem alguns parâmetros adicionais na declaração de Action:

- Command especifica qual programa ou script deve ser executado. O `\"${OVwSelections}\\"` no final da string de comando transmite ao programa todos os objetos atualmente selecionados, como argumentos.
- MinSelected declara quantos nós devem estar selecionados antes da disponibilização desse item. Se nada estiver selecionado, a opção de menu correspondente ficará desativada e não aceitará cliques
- MaxSelected funciona de modo idêntico mas declara o número máximo de objetos que podem ser selecionados.
- SelectionRule usa os campos de capacidade\* dentro de uma instrução lógica. Essas normas declararam o que é necessário para que uma opção seja considerada uma “boa opção”.

As declarações de Action podem conter vários parâmetros adicionais, assim como arquivos de registro. Os exemplos apresentados devem ser suficientes para que você acerte seus ponteiros. *OVwRegIntro (5)* define com detalhes a sintaxe dos arquivos de registro; leia essa página com atenção se estiver interessado em aprender a adicionar itens de menu personalizados.

## Perfis para usuários diferentes

Alguns usuários talvez tenham métodos específicos para utilizar o NNM. Por exemplo, um operador observando problemas na rede pode precisar de um grupo limitado de menus e ferramentas; um engenheiro sênior de operação em rede

\* Consulte o `$OV_FIELDS` para obter mais definições de campos de capacidade.

pode preferir um grupo de opções bem maior. Você pode utilizar o diretório `$OV_REGISTRATION` e a variável de ambiente `$OVwRegDir` para personalizar o NNM para cada usuário.

A seção anterior ensina a adicionar menus, modificando os arquivos no diretório `$OV_REGISTRATION/C`. Por default, esse é o diretório que o NNM usa ao ser iniciado. Entretanto, você pode criar a quantidade de perfis necessária, no diretório `$OV_REGISTRATION`. Após criar outro diretório de perfis, você pode modificar a variável de ambiente `$OVwRegDir` de modo a indicar esse novo diretório. A partir de então, ao ser iniciado, o NNM usará o novo perfil.

Uma maneira de configurar os perfis específicos do usuário é criar uma conta que qualquer um pode usar para iniciar uma sessão do NNM. Com essa conta, o mapa da rede é aberto somente para leitura\* e apresenta apenas os menus mínimos (“File Exit”, “Map Refresh”, “Fault Alarms” etc.). Crie um novo perfil para essa conta no diretório `$OV_REGISTRATION/skel` copiando todos os arquivos do perfil predefinido, `$OV_REGISTRATION/C`, para o novo diretório `skel`. Em seguida, modifique esse perfil, removendo a maioria das opções de menu, impedindo dessa forma que o operador execute quaisquer comandos externos.\*\* Para iniciar o NNM usando esse perfil, você deve apontar a variável de ambiente `$OVwRegDir` para o diretório do novo perfil. Para testar o novo perfil, emita os seguintes comandos do Bourne shell:

```
[root] [nms] /> OVwRegDir=/etc/opt/OV/share/registration/skel  
[root] [nms] /> export OVwRegDir  
[root] [nms] /> $OV_BIN/ovw<
```

Quando você tiver certeza de que esse novo perfil funciona, crie uma conta para executar o NNM com permissões mínimas e, no script de inicialização dessa conta, defina adequadamente a variável `$OVwRegDir` (ou seja, para apontar para sua configuração básica). Em seguida, certifique-se de que os usuários não podem executar o NNM em suas contas habituais – talvez limitando o acesso de execução do NNM a um grupo específico, o que obrigará os usuários não pertencentes a esse grupo a utilizar a conta especial para executar o NNM. Certifique-se também de que os usuários não confiáveis não possam modificar o diretório `$OV_REGISTRATION` nem os respectivos subdiretórios.

## Usando o NNM para comunicações

Um dos métodos mais exóticos para utilizar o SNMP é como uma ferramenta de transmissão de mensagens. Por exemplo, certamente é útil saber que o banco de

\* Ao iniciar o NNM na linha de comando, use `$OV_BIN/ovw -ro` Para abrir o mapa predefinido no modo somente leitura. Isso impedirá que os usuários efetuem quaisquer modificações no mapa (movimentações, inclusões, exclusões etc.).

\*\* Só porque um mapa está aberto no modo somente leitura, isso não significa que os usuários não podem efetuar modificações no back-end do NNM. O usuário que tiver a possibilidade de carregar os itens de menu pode fazer mudanças exatamente como um superuser. A melhor maneira de impedir essas alterações é remover todas/algumas opções de menu da configuração.

dados do Oracle está desabilitado, mas é ainda mais útil enviar mensagens para os principais usuários, informando que o banco de dados está com problemas e que entrará em manutenção no final do dia. Em um ambiente pequeno, é fácil fornecer os diversos tipos de notificação. Mas em uma empresa grande com vários escritórios, é importante ter um método padrão para a comunicação com outros departamentos. A ferramenta Event Categories do NNM é perfeita para ser utilizada como um dispositivo de comunicação centralizada.

Imagine uma interface da Web que permite enviar traps para Event Categories. O preenchimento de um formulário simples em um navegador gera automaticamente uma trap, que é anunciada nas categorias pertinentes. A Figura B-3 mostra tal interface.

Select Your Name	Other:	
Select Desc Of Action	Other:	
Select Server(s) Affected	Other:	
Select Services Affected	Other:	
Select Time Of Event	Seconds	From Now
Select Est Time Of Completion	Seconds	
Select Severity	* See Below For Severity Descriptions	
Additional Comments		
<input type="button" value="Submit Event"/> - <input type="button" value="Clear Form"/>		

Figura B-3 Interface de Web do SNMP

Que perguntas uma pessoa (você, gerentes, usuários etc.) costuma fazer quando ocorre um problema? As mais comuns são:

Quem está encarregado? Nome, telefone, pager

O que está acontecendo? Reinicialização, upgrade, falha

Quais os servidores afetados? Produção, teste, desenvolvimento

Quais os serviços afetados? Correio, notícias, banco de dados, servidor da Web

Quando aconteceu? Por exemplo, há 10 minutos, 4 dias

Quando será corrigido? Por exemplo, imediatamente, amanhã

Qual é o grau de severidade? Normal, Aviso, Relevância Mínima, Relevância Máxima, Crítico

É possível responder a todas essas perguntas por meio do formulário HTML apresentado na Figura B-3. O script CGI ou servlet Java que processa o

formulário pode se recusar a aceitá-lo até que o usuário tenha preenchido todos os campos, garantindo o recebimento de informações completas e consistentes.

Não é difícil configurar um sistema informativo como esse. Você pode utilizar qualquer servidor da Web padrão,\* um pouco de HTML e sua linguagem preferida para processar o formulário. Assim que você analisar a saída do formulário, poderá usar qualquer um dos programas de geração de traps discutidos, para enviar a trap. Mais adiante, essa trap será exibida em uma das Event Categories (Categorias de Evento) do NNM. (Se você não estiver usando o NNM, discutimos sobre outros daemons de traps, que podem ser usados para receber a trap e avisar aos usuários. Entretanto, o NNM é prático porque faz tudo para você.)

O objetivo de toda essa configuração é fazer com que as pessoas usem e observem o NNM. Se ele não for usado por todos, esse mecanismo realmente não fará diferença. Talvez não seja fácil treinar usuários de departamentos não-técnicos a observar no NNM notificações importantes mas se você conseguir essa façanha, terá criado um mecanismo sofisticado para transmitir informações importantes para os usuários.

---

\* Visite o endereço <http://www.apache.org> para obter mais informações sobre um servidor da Web gratuito do Unix ou NT.

# C

## Ferramentas do Net-SNMP

Este apêndice fornece resumos sucintos das ferramentas de linha de comando incluídas na Versão 4.2 do pacote Net-SNMP (disponível em <http://net-snmp.sourceforge.net>).

Em vez de tentar descrever todas as opções para todos os comandos, destacamos as mais importantes e úteis. Também citamos alguns casos em que o comportamento dos comandos é diferente do descrito nas páginas do manual. Infelizmente, há várias discrepâncias. Evidentemente, a situação atual está longe da ideal, mas a documentação ou os comandos serão corrigidos em alguma versão futura.

### Net-SNMP e arquivos de MIBs

Por default, o Net-SNMP lê os arquivos de MIB no diretório `/usr/local/share/snmp/mibs`. Ao ser instalado, o Net-SNMP preenche esse diretório com algumas dezenas de arquivos de MIB, incluindo a UCD MIB (o Net-SNMP era chamado de UCD-SNMP) e a RFC 1213 MIB (MIB-II). O Net-SNMP usa os arquivos de MIB para conversão entre IDs numéricas de objetos e as respectivas representações em texto. Os arquivos de MIB também fornecem às ferramentas acesso a informações sobre cada objeto (sintaxe, o tipo de acesso permitido, descrição etc.). Adicionar um arquivo de MIB específico do fornecedor ao Net-SNMP é tão simples quanto inseri-lo no diretório `mibs` e definir a variável de ambiente `$MIBS` com `ALL`, conforme discutido na próxima seção.

### Comandos comuns de linha de comando

Em sua grande maioria, os comandos do Net-SNMP seguem uma estrutura de comandos semelhante; compartilham algumas opções e usam praticamente a mesma sintaxe. Por exemplo, teoricamente, um comando `snmpget` é assim:

```
snmpget opções nome_do_host comunidade ID_objeto...
```

Em outras palavras, o nome do comando é seguido por uma seqüência de opções: o nome do host do sistema a ser pesquisado, a string de comunidade e uma ou mais IDs de objeto. (Observe que você pode utilizar a opções de comunidade *-c* em vez de colocar a string de comunidade depois do nome do host. Você também pode fornecer um nome de host default em seu arquivo *snmp.conf*.) A sintaxe do *snmpset* é apenas um pouco diferente; como o *snmpset* altera valores de objetos, requer a especificação do tipo de dado do objeto e o novo valor:

*snmpset opções nome\_do\_host comunidade ID\_objeto tipo valor...*

A Tabela C-1 resume algumas das opções mais úteis, comuns a todos os comandos do Net-SNMP. Para obter uma lista completa, consulte a página do manual do *snmpcmd(1)*.

Tabela C-1 Resumo de opções de linha de comando

Opção	Descrição
<i>-m</i>	Especifica quais módulos de MIB o comando deve carregar. Para que o comando analise o arquivo de MIB de um fornecedor específico, copie esse arquivo para <i>/usr/local/share/snmp/mibs</i> e chame o comando com a opção <i>-m ALL</i> . O argumento <i>ALL</i> instrui o comando a ler todos os arquivos de MIB contidos no diretório. A definição da variável de ambiente \$MIBS com <i>ALL</i> alcança o mesmo resultado. Para que o comando não leia todos os arquivos de MIB, você pode colocar depois da opção <i>-m</i> uma lista (separada por caracteres de dois-pontos) dos arquivos de MIB a serem analisados.
<i>-M</i>	Permite especificar uma lista (separada por caracteres de dois-pontos) de diretórios onde pesquisar arquivos de MIB. Essa opção é útil para evitar copiar arquivos de MIB para a localização de MIB predefinida. A definição da variável de shell SMIBDIRS surte o mesmo efeito.
<i>-IR</i>	Faz uma busca de acesso aleatório, no banco de dados da MIB, de um rótulo de OID. Por default, os comandos pressupõem que você especificou uma ID de objeto relativa ao <i>.iso.org.dod.internet.mgmt.mib-2</i> . Na prática, essa opção evita a digitação de OIDs longas para os objetos não armazenados na subárvore <i>mib-2</i> . Por exemplo, existe um grupo de objetos na MIB da Cisco denominado <i>lcpu</i> . Ao usar a opção <i>-IR</i> , você pode recuperar objetos contidos nesse grupo sem digitar a OID inteira; o comando a seguir é suficiente:  <i>snmpget -IR nome_do_host comunidade lcpu.2</i>

Se existir mais de um objeto com o nome especificado, as ferramentas do Net-SNMP acessarão o primeiro objeto encontrado. Como esse recurso é considerado uma busca de acesso aleatório, não é possível prever qual objeto as ferramentas encontrarão primeiro. Nas MIBs padrão, raramente (ou nunca) os objetos têm o mesmo nome, mas não há garantias de que um nome seja único, principalmente ao usar MIBs específicas de fornecedores.

**Tabela C-1 Continuação**

Opção	Descrição
-On	Imprime as OIDs numericamente (por exemplo, .1.3.6.1.2.1.1.3.0). Observe que as opções -O podem ser combinadas, desde que a combinação faça sentido.
-Of	Imprime a OID inteira (ou seja, começando em .1).
-Os	Exibe somente a parte final da OID, em forma simbólica (por exemplo, sysUpTime.0).
-OS	Idêntica à -Os, mas prefixa o nome do objeto com o nome do arquivo de MIB em que o objeto foi obtido (por exemplo, SNMPv2-MIB::sysUpTime.0).
-T	Especifica se o comando deve utilizar o TCP ou UDP como o protocolo de camada de transporte. UDP é o default; -T tcp usa o TCP.
-v	Especifica a versão do SNMP a ser usada. Por default, os comandos usam a Versão 1. As opções válidas são -v 1, -v 2c e -v 3. Observe que alguns comandos, como o snmpbulkget, estão disponíveis somente para as Versões 2c e 3.
-h	Exibe informações de ajuda sobre o comando.
-c	Especifica a string de comunidade para o comando. Como alternativa, você pode colocar a string de comunidade depois do nome do host e omitir a opção -c.

## Ferramentas de linha de comando do Net-SNMP

Esta seção descreve resumidamente cada uma das ferramentas do Net-SNMP. Por default, a instalação do Net-SNMP coloca todos esses comandos em `/usr/local/bin`. Todos os exemplos desta seção pressupõem que o `/usr/local/bin` existe no caminho de seu sistema.

### *snmpwalk*

O `snmpwalk` executa a operação de get-next. Nós o utilizamos no livro inteiro, de modo que você já deve conhecer; nesta seção, usaremos para demonstrar algumas das opções apresentadas na Tabela C-1.

Vamos supor que você queira executar um `snmpwalk` em um roteador da Cisco. Se você não tiver quaisquer MIBs da Cisco instaladas, verá o seguinte:

```
$ snmpwalk cisco.ora.com public .1.3.6.1.4.1.9
enterprises.9.2.1.1.0 = "..System Bootstrap, Version 11.2(17)GS2, [htseng 180]
EARLY DEPLOYMENT RELEASE SOFTWARE (fc1)..Copyright (c) 1999 by Cisco Systems,
Inc..."
enterprises.9.2.1.2.0 = "reload"
enterprises.9.2.1.3.0 = "cisco"
enterprises.9.2.1.4.0 = "ora.com"
```

```
enterprises.9.2.1.5.0 = IpAddress: 127.45.23.1
enterprises.9.2.1.6.0 = IpAddress: 0.0.0.0
enterprises.9.2.1.8.0 = 131890952
enterprises.9.2.1.9.0 = 456
enterprises.9.2.1.10.0 = 500
enterprises.9.2.1.11.0 = 17767568
enterprises.9.2.1.12.0 = 0
enterprises.9.2.1.13.0 = 0
enterprises.9.2.1.14.0 = 104
enterprises.9.2.1.15.0 = 600
...
...
```

Lembre-se de que o `.1.3.6.1.4.1` é `.iso.org.dod.internet.private.enterprises`, e 9 é o número de empresa privada da Cisco. Portanto, o comando anterior está percorrendo toda a subárvore da Cisco, que é muito grande; eliminados a maior parte da saída. A saída que você vê não está muito legível porque ainda não instalamos as MIBs da Cisco, de modo que o comando `snmpwalk` não pode fornecer nomes de objetos legíveis. Simplesmente, precisamos adivinhar quais são esses objetos.

E fácil solucionar esse problema. Copie as MIBs\* da Cisco para o repositório principal do Net-SNMP (`/usr/local/share/snmp/mibs`) e use a opção de linha de comando `-m ALL`. Com essa opção, o `snmpwalk` analisa todos os arquivos contidos no repositório da MIB. Dessa forma, obtemos as IDs de objeto na forma de strings (legíveis) e podemos percorrer a subárvore `cisco` por nome, em vez de especificar a ID numérica completa do objeto (`.1.3.6.1.4.1.9`):

```
$ snmpwalk -m ALL cisco.ora.com public cisco
enterprises.cisco.local.lcpu.1.0 = "..System Bootstrap, Version
11.2(17)GS2,
```

```
Systems, Inc..."
```

```
enterprises.cisco.local.lcpu.2.0 = "reload"
enterprises.cisco.local.lcpu.3.0 = "cisco"
enterprises.cisco.local.lcpu.4.0 = "ora.com"
enterprises.cisco.local.lcpu.5.0 = IpAddress: 127.45.23.1
enterprises.cisco.local.lcpu.6.0 = IpAddress: 0.0.0.0
enterprises.cisco.local.lcpu.8.0 = 131888844
enterprises.cisco.local.lcpu.9.0 = 456
enterprises.cisco.local.lcpu.10.0 = 500
enterprises.cisco.local.lcpu.11.0 = 17767568
enterprises.cisco.local.lcpu.12.0 = 0
enterprises.cisco.local.lcpu.13.0 = 0
enterprises.cisco.local.lcpu.14.0 = 104
enterprises.cisco.local.lcpu.15.0 = 600
...
...
```

Agora, vamos cortar a saída adicionando a opção `-Os`, que omite a parte inicial de cada OID:

266 \* Você encontrará algumas MIBs da Cisco em <ftp://ftp.cisco.com/pub/mibs/>

```
$ snmpwalk -m ALL -Os cisco.ora.com public cisco
1cpu.1.0 = "..System Bootstrap, Version 11.2(17)GS2, [htseng 180] EARLY
1cpu.2.0 = "reload"
1cpu.3.0 = "cisco"
1cpu.4.0 = "ora.com"
1cpu.5.0 = IpAddress: 127.45.23.1
1cpu.6.0 = IpAddress: 0.0.0.0
1cpu.8.0 = 131888844
1cpu.9.0 = 456
1cpu.10.0 = 500
1cpu.11.0 = 17767568
1cpu.12.0 = 0
1cpu.13.0 = 0
1cpu.14.0 = 104
1cpu.15.0 = 600
...

```

Essa saída é um pouco mais fácil de ler, porque elimina a parte redundante de cada OID. Avancemos esse comando um passo à frente:

```
snmpwalk -OsS cisco.ora.com public system
RFC1213-MIB::sysDescr.0 = "Cisco Internetwork Operating System Software ..IOS (tm)
GS Software (GSR-K4P-M), Version 12.0(15)S, EARLY DEPLOYMENT RELEASE SOFTWARE
(fcl)..TAC Support: http://www.cisco.com/cgi-bin/ibld/view.pl?i=support..
Copyright (c) 1986-2001 by Cisco Systems, Inc..."
RFC1213-MIB::sysObjectID.0 = OID: DTRConcentratorMIB::catProd.182
EXPRESSION-MIB::sysUpTimeInstance = Timeticks: (344626986) 39 days, 21:17:49.86
RFC1213-MIB::sysContact.0 = "O'Reilly Data Center"
RFC1213-MIB::sysName.0 = "cisco.ora.com"
RFC1213-MIB::sysLocation.0 = "Atlanta, GA"
RFC1213-MIB::sysServices.0 = 6
RFC1213-MIB::system.8.0 = Timeticks: (0) 0:00:00.00
```

Esse comando percorre a subárvore *systems* subordinado à *mib-2* não há necessidade de usar *-m ALL*. É uma das MIBs carregadas automaticamente pelas ferramentas do Net-SNMP. A inclusão do Símbolo -Oída com o nome do arquivo de MIB; é possível perceber que cada linha começa com *RFC1213-MIB* é o nome do arquivo que define a *mib-2*.

## *snmpget*

O comando *snmpget* emite uma única operação *get*. A sintaxe é:

```
snmpget opções nome_do_host comunidade ID_objeto...
```

## *snmpbulkget*

O SNMPv2 oferece uma operação chamada *get-bulk*, implementada pelo comando *snmpbulkget*. A operação *get-bulk* permite recuperar um grupo de infor-

mações em uma única operação, ao contrário de um único *get* ou seqüência de operações *get-next*. A sintaxe de *snmpbulkget* é:

```
snmpbulkget -v 2c opções nome_do_host comunidade ID_objeto
```

A opção *-v 2c* é necessária porque *get-bulk* é definido pelo SNMP Version 2.

Existe uma única opção específica do comando, *-B nonrep rep*. *nonrep* é o número de objetos escalares que esse comando retornará; *rep* é o número de instâncias de cada objeto não-escalar que o comando retornará. Se você emitir essa opção, serão usados os valores predefinidos de *nonrep* e *rep*, 1 e 100, respectivamente.

### *snmpbulkwalk*

O comando *snmpbulkwalk* usa a seqüência de comandos *get-bulk* para recuperar partes de uma MIB. A diferença em relação ao *snmpbulkget* é que este comando não precisa da opção *-B* definida; ele percorre a árvore inteira até alcançar o final ou recupera todos os objetos solicitados. A sintaxe é:

```
snmpbulkwalk -v 2c opções nome_do_host comunidade ID_objeto
```

### *snmpset*

O comando *snmpset* é usado para modificar ou definir o valor de um objeto da MIB. O comando fica assim:

```
snmpset opções nome_do_host comunidade ID_objeto tipo valor...
```

Você pode fornecer qualquer número do trio *ID\_objeto/tipo/valor*; o comando executará operações de *set* para todos os objetos informados. *tipo* é uma abreviação de um único caractere que indica o tipo de dado do objeto sendo definido. A Tabela C-2 lista os tipos válidos.

Tabela C-2 Tipos de objetos do *snmpset*

Abreviação	Tipo
a	Endereço IP
b*	Bits
d	String decimal
D	Duplo
F	Flutuante
I	Inteiro
I	int64 com sinal
n	Nulo
o	ID de objeto

Tabela C-2 Continuação

Abreviação	Tipo
s	String
t	Marcações de tempo
u	Inteiro sem sinal
U	int64 sem sinal
x	String hexadecimal

<sup>1</sup> As páginas do manual consideram um tipo de dado válido, mas não a saída da Ajuda do comando.

## snmptrap

Para enviar uma trap, use o comando *snmptrap*. A sintaxe desse comando é:

```
snmptrap opções nome_do_host comunidade trap parâmetros...
```

Para a Versão 1, são necessários os seguintes parâmetros de trap:

```
oid_empresa agente tipo_trap tipo_específico tmp_funcionamento ID_objeto  
tipo valor...
```

Esse comando é discutido com detalhes no Capítulo 10. Cada trio de *ID\_objeto/tipo/valor* especifica uma vinculação de variáveis a ser incluída com a trap; é possível incluir qualquer número de vinculações de variáveis. Observe que o agente e o *tmp\_funcionamento* não são opcionais; contudo, se você fornecer uma string vazia ("") como marcador, esses parâmetros serão definidos com o endereço IP do sistema emissor da trap e com o tempo de funcionamento atual do sistema.

Os parâmetros são mais simples para as traps da Versão 2, principalmente porque as traps (atualmente denominadas como *notificações*) são objetos completos de MIB. São necessários os seguintes parâmetros:

```
snmptrap -v 2c opções nome_do_host comunidade tmp_funcionamento oid_trap  
ID_objeto tipo valor...
```

## snmpdelta

O comando *snmpdelta* monitora OIDs e rastreia as alterações efetuadas nos valores da OID com o passar do tempo. A sintaxe é:

```
snmpdelta opções nome_host comunidade ID_objeto...
```

O *snmpdelta* requer a especificação de uma OID de um objeto escalar, com valor em inteiro – esse comando não pode monitorar tabelas. Por exemplo, para observar os octetos alcançando uma interface, você não pode especificar somente os *ifInOctets*; é necessário informar o número da interface adicionamente ao

nome do objeto (por exemplo, *ifInOctets.3*). Por default, o *snmpdelta* pesquisa o objeto especificado a cada segundo.

A Tabela C-3 lista algumas das opções específicas do *snmpdelta*. Existem vários problemas na documentação deste comando, mas se você adotar as opções listadas a seguir, alcançará seu objetivo.

Tabela C-3 Opções do *snmpdelta*

Opção	Descrição
<i>-t</i>	A documentação informa: “Determine o intervalo de tempo da entidade monitorada”. O significado dessa sentença não está claro, mas temos a impressão de que essa entrada é necessária para obter leituras diferentes de zero.
<i>-s</i>	Exibe uma marcação de hora em cada grupo de resultados.
<i>-m</i>	Imprima o valor máximo obtido.
<i>-l</i>	Escreva a saída em um arquivo. O nome do arquivo está na forma <i>nome_do_host-OID</i> . Por exemplo, para monitorar as variáveis <i>ifInOctets.3</i> e <i>ifOutOctets.3</i> no roteador do host, a opção <i>-l</i> criará dois arquivos, <i>nome_do_host-ifInOctets.3</i> e <i>nome_do_host-ifOutOctets.3</i> , onde a saída do <i>snmpdelta</i> será escrita. (Observe que essa saída não tem ligação aparente com a configuração, como informa a documentação.)
<i>-p</i>	Especifica o intervalo de polling (o default é 1 segundo).
<i>-T</i>	Imprima a saída em formato de tabela.

### *snmpdf*

O *snmpdf* funciona exatamente como o comando *df* do Unix, exceto pelo fato de usar o SNMP para consultar hosts em uma rede. A sintaxe é:

```
snmpdf -Cu opções... nome_do_host comunidade
```

A opção *-Cu* instrui o comando a consultar a MIB privada do Net-SNMP. Por default, é usada a Host Resources MIB.

### *snmpgetnext*

O comando *snmpgetnext* usa a operação *get-next* para recuperar o objeto seguinte em um host. Por exemplo, se você solicitar ao comando a execução de um *get-next* de *ifOutOctets.4*, ele recuperará o objeto seguinte na árvore da MIB, que provavelmente será o *ifOutOctets.5*. (Se a máquina sendo pesquisada possuir somente quatro interfaces, você obterá o próximo objeto na MIB, seja ele qual for. Saiba também que existem situações mais obscuras que geram um “bu-

raco” na tabela de interfaces, de modo que a interface depois da .4 pode ser .6 ou .7.) Você pode utilizar esse comando para implementar uma versão exclusiva do *snmpwalk*. A sintaxe é:

```
snmpgetnext opções... nome_do_host comunidade ID_objeto...
```

Não há opções específicas para o *snmpgetnext*.

### ***snmpstatus***

O comando *snmpstatus* recupera informações de status de um host e imprime as seguintes informações:

- Endereço IP da entidade
- Uma descrição em texto da entidade (*sysDescr.0*)
- Tempo de funcionamento da entidade (*sysUpTime.0*)
- Soma dos pacotes recebidos em todas as interfaces (*ifInUcastPkts.\* + ifInNUcastPkts.\**)
- Soma dos pacotes transmitidos em todas as interfaces (*ifOutUcastPkts.\* + ifOutNUcastPkts.\**)
- Número de pacotes de entrada de IP (*ipInReceives.0*)
- Número de pacotes de saída de IP (*ipOutRequests.0*)

A sintaxe do *snmpstatus* é simples e não há opções específicas desse comando:

```
snmpstatus opções... nome_do_host comunidade
```

### ***snmptable***

O comando *snmptable* usa os comandos *get-next* para imprimir o conteúdo de uma tabela no formato tabular. A sintaxe é:

```
snmptable opções... nome_do_host comunidade ID_objeto
```

O parâmetro *ID\_objeto* deve ser a ID de uma tabela (como *ifTable*), não de um objeto dentro de uma tabela. A Tabela C-4 lista algumas opções específicas do comando *snmptable*.

*Tabela C-4 Opções do snmptable*

Opção	Descrição
-Cf F	Separe colunas de tabela do a string F. Por exemplo, -Cf : separa colunas com um caractere de dois-pontos, o que pode facilitar ainda mais a importação da saída do <i>snmptable</i> em outro programa.

**Tabela C-4 Continuação**

Opção	Descrição
-Cw W	Defina a largura máxima de uma tabela com W. Se o comprimento das linhas for maior que W, a tabela será dividida em seções. Como as tabelas podem ter várias colunas, é bem provável que você use essa opção.
-Ci	Prefixe todas as linhas impressas com o índice da entrada.
-Cb	Exiba um título resumido.
-Ch	Imprima somente os títulos de coluna.
-CH	Omita os títulos de coluna.

### *snmpusm*

O comando *snmpusm* fornece acesso fácil à tabela *User-based Security Model* (USM) do agente, usada basicamente para configurar os recursos do SNMPv3 do agente (gerenciando usuários, definindo e modificando frases-senha etc.). Este comando é discutido no Apêndice F.

### *snmpconf*

Este comando é um script em Perl interativo usado para criar e manter os arquivos de configuração do Net-SNMP, *snmp.conf* e *snmpd.conf*. A sintaxe é:

*snmpconf nome\_do\_arquivo*

O parâmetro *nome\_do\_arquivo* deve ser o *snmp.conf* ou o *snmpd.conf*.

### *snmpinform*

Este comando pode ser usado para enviar uma trap do SNMPv2. Se você enviar uma trap com o *snmpinform*, o comando aguardará uma resposta do destinatário. Observe que é possível enviar um *inform* usando o comando *snmptrap*, se você especificar a opção -Ci. As opções do comando *snmpinform* são idênticas às do *snmptrap*.

### *snmptranslate*

O pacote Net-SNMP é fornecido com uma ferramenta prática, denominada *snmptranslate*, que faz conversões entre nomes de objeto numéricos e em texto. Em termos mais genéricos, pode ser usado para pesquisar informações em arquivos de MIB. A sintaxe é:

*snmptranslate opções ID\_objeto*

O *snmptranslate* não faz consultas em quaisquer dispositivos, de modo que não são necessários os parâmetros *nome\_do\_host* nem *comunidade*. O único objeto deste comando é ler arquivos de MIB e gerar saída sobre objetos específicos.

cos. Antes de examinar os exemplos, convém observar que o *snmptranslate* interpreta as opções *-O* como “seja gentil, interessante”. Francamente, é óbvio que essas interpretações estão incorretas. Os exemplos a seguir mostram o que realmente acontece quando você usa essas opções – deixaremos a conclusão a seu critério. Esperamos a correção desses problemas em uma versão posterior do Net-SNMP.

Vamos supor que você precise saber a OID empresarial da Cisco Systems. O comando a seguir faz o truque:

```
$ snmptranslate -m ALL -IR -Of cisco  
.1.3.6.1.4.1.9
```

O retorno nos informa que a OID empresarial da Cisco é *.1.3.6.1.4.9*. O uso da opção *-IR*, que instrui o *snmptranslate* a pesquisar aleatoriamente um objeto denominado *cisco*. Se você não especificar essa opção, o *snmptranslate* falhará porque tentará localizar o objeto *cisco* na árvore *mib-2*.

Suponhamos que você queira converter *.1.3.6.1.4.1.9* no nome simbólico completo. É fácil:

```
$ snmptranslate -m ALL -Ofn .1.3.6.1.4.1.9  
.iso.org.dod.internet.private.enterprises.cisco
```

Nesse caso, a opção *-IR* não é necessária porque não estamos fazendo uma pesquisa aleatória. A opção *-Ofn* assegura a impressão da ID do objeto completa, na forma simbólica (texto). Veja o que acontece se usarmos a opção *-Of* isoladamente:

```
$ snmptranslate -m ALL -Of .1.3.6.1.4.1.9  
enterprises.cisco
```

Como mencionamos anteriormente, não era esse o comportamento esperado das opções *-Ofn* e *-Of*. Ao escrever scripts, não espere que esse comportamento seja idêntico nas próximas versões.

Agora, vamos supor que você precise de outras informações sobre um objeto específico. A opção *-Td* exibe a definição do objeto conforme consta no arquivo de MIB:

```
$ snmptranslate -Td system.sysLocation  
.1.3.6.1.2.1.1.6  
sysLocation OBJECT-TYPE  
-- FROM SNMPv2-MIB, RFC1213-MIB  
  
-- TEXTUAL CONVENTION DisplayString  
SYNTAX OCTET STRING (0..255)  
DISPLAY-HINT "255a"  
MAX-ACCESS read-write  
STATUS current  
DESCRIPTION "The physical location of this node (e.g., 'telephone closet, 3rd floor'). If the location is unknown, the value is the zero-length string."
```

```
::= { iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) system(1) 6 }
```

A opção *-Td* pode economizar muito trabalho de pesquisa em arquivos de MIB, para localizar a definição adequada, principalmente quando combinada com a opção *-IR*. Além disso, a última linha apresenta a ID do objeto completa nas formas numéricas e de string, não somente o pai do objeto. Observe que os outros comandos do Net-SNMP possuem uma opção *-T* não relacionada; não confunda. *-T* não faz sentido para este comando, porque o *snmptranslate* só pesquisa um arquivo local e não precisa acessar a rede.

A opção *-Tp* imprime uma árvore de OIDs inteira. A melhor maneira de entender tudo isso é visualizando:

```
$ snmptranslate -Tp system
+--system(1)
|
+-- -R-- String    sysDescr(1)
|      Textual Convention: DisplayString
|      Size: 0..255
+-- -R-- ObjID     sysObjectID(2)
+-- -R-- TimeTicks sysUpTime(3)
+-- -RW-- String   sysContact(4)
|      Textual Convention: DisplayString
|      Size: 0..255
+-- -RW-- String   sysName(5)
|      Textual Convention: DisplayString
|      Size: 0..255
+-- -RW-- String   sysLocation(6)
|      Textual Convention: DisplayString
|      Size: 0..255
+-- -R-- Integer   sysServices(7)
+-- -R-- TimeTicks sysORLastChange(8)
|      Textual Convention:TimeStamp
|
+--sysORTable(9)
|
+--sysOREntry(1)
|
+-- - — Integer   sysORIndex(1)
+-- -R- ObjID     sysORID(2)
+-- -R- String   sysORDescr(3)
|      Textual Convention: DisplayString
|      Size: 0..255
+-- -R- TimeTicks sysORUpTime(4)
|      Textual Convention:TimeStamp
```

Exibimos a subárvore *system* porque é bem curta. Com base nessa saída, é relativamente fácil ver todos os objetos subordinados à *system*, juntamente com os respectivos tipos e convenções de texto. Essa é uma maneira excelente de visualizar os objetos definidos em uma MIB, assim como suas relações com outros objetos. A saída pode ser volumosa mas é uma forma prática de obter um mapa e determinar os objetos que provavelmente serão úteis.

# D

## RFCs do SNMP

Este apêndice fornece uma lista resumida de todas as RFCs do SNMP, juntamente com o status de cada RFC. Essa lista (geralmente citada como Resumo Padrão) foi extraída do *The Simple Times*, uma publicação on-line que todos os que trabalham com o SNMP devem conhecer, tem autorização para ser utilizada e pode ser encontrada em cada edição trimestral da revista. Para obter informações sobre como se inscrever para receber essa publicação gratuita, visite <http://www.simple-times.org>.

### *SMIv1 Linguagem de Definição de Dados*

Padrões completos:

- [RFC 1155 -Structure of Management Information](#)
- [RFC 1212 -Concise MIB Definitions](#)

Informativos:

- [RFC 1215 -A Convention for Defining Traps](#)

### *SMIv2 Linguagem de Definição de Dados*

Padrões completos:

- [RFC 2578 -Structure of Management Information](#)
- [RFC 2579 -Textual Conventions](#)
- [RFC 2580 -Conformance Statements](#)

### *SNMPv1 Protocolo*

Padrões completos:

- [RFC 1157 -Simple Network Management Protocol](#)

Padrões sugeridos:

- [RFC 1418 -SNMP over OSI](#)
- [RFC 1419 -SNMP over AppleTalk](#)
- [RFC 1420 -SNMP over IPX](#)

## ***SNMPv2 Protocolo***

**Padrões de Draft:**

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

**Experimental:**

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

## ***SNMPv3 Protocolo***

**Padrões de Draft:**

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

**Padrões sugeridos:**

- RFC 2576 -Coexistence between SNMP Versions*

**Informativos:**

- RFC 2570 -Introduction to SNMPv3*

**Experimental:**

- RFC 2786 -Diffie-Hellman USM Key Management*

## ***SNMP – Possibilidade de Extensão de Agente***

**Padrões sugeridos:**

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

## ***SMIv1 – Módulos de MIB***

**Padrões completos:**

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

**Padrões de Draft:**

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

**Padrões sugeridos:**

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

- RFC 1414 -Identification MIB
- RFC 1461 -X.25 Multiprotocol Interconnect MIB
- RFC 1471 -PPP Link Control Protocol MIB
- RFC 1472 -PPP Security Protocols MIB
- RFC 1473 -PPP IP NCP MIB
- RFC 1474 -PPP Bridge NCP MIB
- RFC 1512 -FDDI Interface Type (SMT 7.3) MIB
- RFC 1513 -RMON Token Ring Extensions MIB
- RFC 1515 -IEEE 802.3 MAU MIB
- RFC 1525 -Source Routing Bridge MIB
- RFC 1742 -AppleTalk MIB

## *SMIv2 – Módulos de MIB*

Padrões completos:

- RFC 2819 -Remote Network Monitoring MIB

Padrões de Draft:

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

Padrões sugeridos:

- RFC 1611 -DNS Server MIB
- RFC 1612 -DNS Resolver MIB
- RFC 1666 -SNA NAU MIB
- RFC 1696 -Modem MIB
- RFC 1697 -RDBMS MIB
- RFC 1747 -SNA Data Link Control MIB
- RFC 1749 -802.5 Station Source Routing MIB
- RFC 1759 -Printer MIB
- RFC 2006 -Internet Protocol Mobility MIB
- RFC 2011 -Internet Protocol MIB
- RFC 2012 -Transmission Control Protocol MIB

- RFC 2013 -User Datagram Protocol MIB  
RFC 2020 -IEEE 802.12 Interfaces MIB  
RFC 2021 -RMON Version 2 MIB  
RFC 2024 -Data Link Switching MIB  
RFC 2051 -APPN MIB  
RFC 2096 -IP Forwarding Table MIB  
RFC 2108 -IEEE 802.3 Repeater MIB  
RFC 2127 -ISDN MIB  
RFC 2128 -Dial Control MIB  
RFC 2206 -Resource Reservation Protocol MIB  
RFC 2213 -Integrated Services MIB  
RFC 2214 -Guaranteed Service MIB  
RFC 2232 -Dependent LU Requester MIB  
RFC 2238 -High Performance Routing MIB  
RFC 2266 -IEEE 802.12 Repeater MIB  
RFC 2287 -System-Level Application Mgmt MIB  
RFC 2320 -Classical IP and ARP over ATM MIB  
RFC 2417 -Multicast over UNI 3.0/3.1 / ATM MIB  
RFC 2452 -IPv6 UDP MIB  
RFC 2454 -IPv6 TCP MIB  
RFC 2455 -APPN MIB  
RFC 2456 -APPN Trap MIB  
RFC 2457 -APPN Extended Border Node MIB  
RFC 2465 -IPv6 Textual Conventions and General Group MIB  
RFC 2466 -ICMPv6 MIB  
RFC 2493 -15 Minute Performance History TCs  
RFC 2494 -DS0, DS0 Bundle Interface Type MIB  
RFC 2495 -DS1, E1, DS2, E2 Interface Type MIB  
RFC 2496 -DS3/E3 Interface Type MIB  
RFC 2512 -Accounting MIB for ATM Networks  
RFC 2513 -Accounting Control MIB  
RFC 2514 -ATM Textual Conventions and OIDs  
RFC 2515 -ATM MIB  
RFC 2558 -SONET/SDH Interface Type MIB  
RFC 2561 -TN3270E MIB  
RFC 2562 -TN3270E Response Time MIB  
RFC 2564 -Application Management MIB  
RFC 2576 -SNMP Community MIB  
RFC 2584 -APPN/HPR in IP Networks  
RFC 2591 -Scheduling MIB  
RFC 2592 -Scripting MIB  
RFC 2594 -WWW Services MIB  
RFC 2605 -Directory Server MIB  
RFC 2613 -RMON for Switched Networks MIB  
RFC 2618 -RADIUS Authentication Client MIB

**RFC 2619 -RADIUS Authentication Server MIB**  
**RFC 2667 -IP Tunnel MIB**  
**RFC 2662 -ADSL Line MIB**  
**RFC 2665 -Ethernet-Like Interface Types MIB**  
**RFC 2668 -IEEE 802.3 MAU MIB**  
**RFC 2669 -DOCSIS Cable Device MIB**  
**RFC 2670 -DOCSIS RF Interface MIB**  
**RFC 2677 -Next Hop Resolution Protocol MIB**  
**RFC 2720 -Traffic Flow Measurement Meter MIB**  
**RFC 2737 -Entity MIB**  
**RFC 2742 -AgentX MIB**  
**RFC 2787 -Virtual Router Redundancy Protocol MIB**  
**RFC 2788 -Network Services Monitoring MIB**  
**RFC 2789 -Mail Monitoring MIB**  
**RFC 2837 -Fibre Channel Fabric Element MIB**  
**RFC 2851 -Internet Network Address TCs**  
**RFC 2856 -High Capacity Data Type TCs**  
**RFC 2864 -Interfaces Group Inverted Stack MIB**  
**RFC 2895 -RMON Protocol Identifier Reference**  
**RFC 2925 -Ping, Traceroute, Lookup MIBs**  
**RFC 2932 -IPv4 Multicast Routing MIB**  
**RFC 2933 -IGMP MIB**  
**RFC 2940 -COPS Client MIB**  
**RFC 2954 -Frame Relay Service MIB**  
**RFC 2955 -Frame Relay/ATM PVC MIB**  
**RFC 2959 -Real-Time Transport Protocol MIB**

#### Informativos:

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

#### Experimental:

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

## **Módulos de MIB Mantidos pela IANA**

### Interface Type Textual Convention

<ftp://ftp.iana.org/mib/ianaiftype.mib>

### Address Family Numbers Textual Convention

<ftp://ftp.iana.org/mib/ianaaddressfamilynumbers.mib>

## [TN3270E Textual Conventions](#)

<ftp://ftp.iana.org/mib/ianatn3270etc.mib>

## [Language Identifiers](#)

<ftp://ftp.iana.org/mib/ianalanguage.mib>

## [IP Routing Protocol Textual Conventions](#)

<ftp://ftp.iana.org/mib/ianairouteprotocol.mib>

## ***Documentos Relacionados***

### **Informativos:**

[RFC 1270 -SNMP Communication Services](#)

[RFC 1321 -MD5 Message-Digest Algorithm](#)

[RFC 1470 -Network Management Tool Catalog](#)

[RFC 2039 -Applicability of Standard MIBs to WWW Server Management](#)

[RFC 2962 -SNMP Application Level Gateway for Payload Address Translation](#)

### **Experimental:**

[RFC 1187 -Bulk Table Retrieval with the SNMP](#)

[RFC 1224 -Techniques for Managing Asynchronously Generated Alerts](#)

[RFC 1238 -CLNS MIB](#)

[RFC 1592 -SNMP Distributed Program Interface](#)

[RFC 1792 -TCP/IPX Connection MIB Specification](#)

[RFC 2593 -Script MIB Extensibility Protocol](#)

# E

## *Suporte para a linguagem Perl no SNMP*

Este apêndice resume o módulo `SNMP_util` de autoria de Mike Mitchell, que usamos em nossos scripts em Perl neste livro inteiro. Esse módulo é distribuído com o módulo SNMP Perl de Simon Leinen; O módulo de Mike combinado ao módulo de Simon pode facilitar ainda mais a programação do SNMP. É possível obter esses módulos em <http://www.switch.ch/misc/leinen/snmp/perl/> ou em <http://www.cpan.org>.

Os scripts em Perl necessitam de duas instruções `use` para se beneficiar do módulo SNMP Perl:

```
use BER;  
use SNMP_Session;
```

Os módulos `BER` e `SNMP_Session` são a base do pacote de Simon. O módulo `SNMP_util`, discutido neste apêndice, facilita um pouco o uso deste pacote, e requer somente uma instrução `use`:

```
use SNMP_util;
```

O pacote de Mike utiliza os outros dois módulos, de modo que não é necessário incluir os três em seus scripts.

### *Rotinas de gerenciamento de MIBs*

As seções a seguir descrevem um grupo de rotinas para trabalhar com MIBs.

#### *`snmpmapOID()`*

Os objetos de MIB contidos na RFC 1213 (MIB-II) e RFC 2955 (Frame Relay) são pré-carregados pelas rotinas neste pacote. Isso significa que você pode fazer referência a um nome simbólico, como `sysLocation.0`, em vez de uma OID nu-

mérica (.1.3.6.1.2.1.1.6). A rotina `snmpmapOID()` permite adicionar pares de nome-OID a esse mapa. A rotina é usada assim:

```
snmpmapOID(text, OID, [text, OID...])
```

Todos os parâmetros são strings. *text* é o nome em texto (ou simbólico) que você deseja usar e *OID* é a ID numérica do objeto ao qual o nome se refere. Uma única chamada a essa rotina pode especificar qualquer número de pares de nome-OID.

Se falhar, `snmpmapOID()` retornará `undef`, de modo que é possível testar a presença de erros assim:

```
@return = snmpmapOID(...);
if(!@return) {
    # error
}
```

### *snmpMIB\_to\_OID()*

Esta rotina usa o nome-de-arquivo de uma MIB como um argumento, lê, analisa o arquivo de MIB e associa as IDs de objetos definidas pela MIB aos respectivos nomes em texto. Retorna o número de mapeamentos criados. Um valor de retorno zero significa que nenhum mapeamento foi criado; -1 significa que ocorreu um erro (por exemplo, não foi possível abrir o arquivo). A rotina é usada como segue:

```
snmpMIB_to_OID(nome_de_arquivo)
```

### *snmpLoad\_OID\_Cache()*

Esta rotina permite mapear nomes em texto para IDs de objeto, usando um arquivo. O arquivo deve consistir em algumas linhas no formato:

```
nome_texto OID
```

O uso dessa rotina é mais veloz do que chamar `snmpMIB_to_OID()` porque não requer a análise de um arquivo de MIB. O único argumento dessa rotina é o nome do arquivo que contém os dados pré-analisados:

```
snmpLoad_OID_Cache(nome_de_arquivo)
```

`snmpLoad_OID_Cache()` retorna -1 se não for possível abrir o arquivo; um valor de retorno 0 indica êxito em sua aplicação.

### *snmpQueue\_MIB\_File()*

Esta rotina especifica uma lista de arquivos de MIB que serão usados para mapear nomes em texto para IDs de objeto. Se um nome ou OID não for encontrado

no mapa interno, cada arquivo de MIB será analisado, até ser encontrada uma correspondência. A rotina é usada assim:

```
snmpQueue_MIB_File(nome_de_arquivo, [nome_de_arquivo])
```

## Operações de SNMP

As rotinas para executar operações de SNMP correspondem às operações\* padrão do SNMP Version 1 e têm os seguintes parâmetros em comum:

### *community (opcional)*

A string de comunidade. Se nenhuma string de comunidade for especificada, será usada a string *public*.

### *host (obrigatório)*

O nome do host ou endereço IP do dispositivo a ser consultado.

### *port (opcional)*

O número da porta para a qual deve ser enviada a consulta ou a trap. O default para todas as rotinas, exceto para a *snmptrap()* é 161. O default para *snmptrap()* é 162.

### *timeout (opcional)*

O tempo limite em segundos; se nenhuma resposta for recebida nesse intervalo, a operação é considerada como tendo falhado e é repetido. O default são 2 segundos.

### *retries (opcional)*

O número de novas tentativas antes de a rotina retornar uma falha. O default são 5.

### *backoff (opcional)*

O valor de reversão; para cada nova tentativa sucessiva, o novo tempo limite é obtido, multiplicando-se o tempo limite atual pela reversão. O default é 1.

### *OID (obrigatório)*

A ID de objeto ou o nome em texto do objeto sendo consultado.

## *snmpget()*

A sintaxe da rotina *snmpget()* é:

```
snmpget([community@]host[:port:]timeout:retries:backoff, OID, [OID...])
```

\* O pacote de Simon Leinen tem suporte para o SNMP v1 e v2; o módulo *SNMP\_util* de Mike Mitchell aceita apenas o v1.

Se falhar, `snmpget()` retornará `undef`.

Lembre-se de que todos os objetos da MIB-II são pré-carregados neste módulo de Perl, de modo que o código a seguir é válido:

```
@sysDescr = snmpget("public@cisco.ora.com", "sysDescr");
```

Não especificamos qualquer parâmetro opcional (tempo limite, reversão etc.); serão usados os valores defaults. Esta rotina permite solicitar "sysDescr" como abreviação de `sysDescr.0`. Quando o módulo de Perl construir seu mapeamento de nomes para IDs de objeto, anexará automaticamente o `.0` final em quaisquer objetos escalares encontrados.

Como `sysDescr` é um objeto escalar definido pela MIB-2 e os objetos da MIB-2 são pré-carregados, `sysDescr` é mapeado para `.1.3.6.1.2.1.1.1.0`. Se você solicitar um objeto escalar de uma MIB privada, deverá incluir `.0` na OID.

Como uma única chamada à `snmpget()` pode recuperar vários objetos, os valores de retorno serão armazenados em um array. Por exemplo:

```
@oids = snmpget("public@cisco.ora.com", "sysDescr", "sysName");
```

Quando essa chamada de função for executada, o valor de `sysDescr` será armazenado em `$oids[0]`; o valor de `sysName` será guardado em `$oids[1]`. Todas as rotinas desse pacote compartilham esse comportamento.

### ***snmpgetnext()***

A rotina `snmpgetnext()` executa uma operação *get-next* para recuperar o valor do objeto da MIB posterior ao objeto passado para a rotina. A sintaxe é:

```
snmpgetnext(community@host:port:timeout:retries:backoff, OID, [OID...])
```

Se falhar, `snmpgetnext()` retornará `undef`.

Assim como em `snmpget()`, é possível solicitar várias OIDs; o valor de retorno de `snmpgetnext()` é um array, com o resultado de cada operação *get-next* em cada posição sucessiva no array. O array retornado de `snmpgetnext()` é diferente daquele retornado por `snmpget()`, pois o valor de cada objeto é precedido pela ID do objeto, no formato:

*OID:valor*

Esta rotina retorna a OID e o valor porque, com a operação *get-next*, você não sabe necessariamente qual será o próximo objeto na árvore da MIB.

### ***snmpwalk()***

A rotina `snmpwalk()` poderia ser facilmente implementada com chamadas repetidas à `snmpgetnext()`; ela percorre a árvore de objetos inteira, começando pelo objeto passado para ela. A sintaxe é:

```
snmpwalk(community@host:port:timeout:retries:backoff, OID)
```

Se falhar, `snmpwalk()` retornará `undef`.

Diferentemente de algumas rotinas deste módulo, `snmpwalk()` permite somente uma OID como argumento. Assim como as outras rotinas, ela retorna um array de valores; cada elemento do array consiste em uma ID de objeto seguida pelo respectivo valor, separada por um caractere de dois-pontos. Por exemplo, após executar este código:

```
@system = snmpwalk("public@cisco.ora.com", "system");
```

O conteúdo do array `@system` seria algo como:

```
1.0:cisco.ora.com Cisco
2.0:1.3.6.1.4.1.0
3.0:23 days, 11:01:57
4.0:Ora Network Admin Staff
5.0:cisco.ora.com
6.0:Atlanta, GA
7.0:4
```

Observe que o array não inclui a ID de objeto completa. Instruímos a rotina `snmpwalk()` a percorrer a árvore começando no objeto `system`, que possui a OID `.1.3.6.1.2.1.1`. O primeiro objeto filho e primeiro item no array é `sysName`, que é `.1.3.6.1.2.1.1.1.0`. `snmpwalk()` retorna `1.0:cisco.ora.com` porque omite a parte genérica da OID (nesse caso, `system`) e imprime somente a parte específica da instância (1.0). De modo semelhante, o item seguinte no array é `system.2.0` ou `system.sysObjectID.0`; seu valor é a ID empresarial da Cisco.

## ***snmpset()***

A rotina `snmpset()` permite definir o valor de um objeto em um dispositivo gerenciado por SNMP. Além dos argumentos padrão (`nome_do_host`, comunidade, etc.), essa rotina espera três argumentos para cada objeto a ser definido: A ID do objeto, o tipo de dado e o valor. A sintaxe dessa rotina é:

```
snmpset([community@]host[:port]:timeout:retries:backoff,
         OID, type, value, [OID, type, value...])
```

O argumento `type` (`tipo`) deve ser uma das seguintes strings:

**string**

Representa o tipo de string

**int**

Representa o tipo inteiro de 32 bits

**ipaddr**

Representa o tipo de endereço IP

**oid**  
Representa o tipo de identificador de objeto (OID)  
Se falhar, `snmpset()` retornará `undef`.

É fácil executar um *set* em um script. O código a seguir define o valor de *sysContact* com "Joe@Ora". Se a operação obtiver êxito, `snmpset()` retornará o novo valor de *sysContact*. Se a operação falhar, a variável *fs* não será definida e `snmpset()` imprimirá uma mensagem de erro:

```
$setResponse =  
    snmpset("private@cisco.ora.com", sysContact,"string","Joe@Ora");  
  
if ($setResponse) {  
    print "SET: sysContact: $setResponse\n";  
} else {  
    print "No response from cisco.ora.com\n";  
}
```

Os motivos mais comuns para a falha de uma rotina `snmpset()` são: host paralisado, host não executando um agente do SNMP ou string de comunidade incorreta.

## *snmptrap()*

A rotina `snmptrap()` gera uma trap do SNMPv1. A maioria dos argumentos é conhecida:

```
snmptrap(community@host:port:timeout:retries:backoff,  
          enterpriseOID, agent, generalID, specificID,  
          OID, type, value, [OID, type, value...])
```

Os argumentos *enterpriseOID*, *agent*, *generalID* e *specificID* estão discutidos no Capítulo 10. Cada trio de OID/tipo/valor define uma vinculação de dados a ser incluída na trap. *OID* é a ID de objeto da variável a ser enviada, *value* é o valor a ser enviado para este objeto e *type* é o tipo de dado do objeto. O argumento *type* deve ser uma das três seguintes strings:

**string**  
Representa o tipo de string

**int**  
Represente o tipo inteiro de 32 bits

**oid**  
Representa o tipo de identificador de objeto (OID)

Se falhar, `snmptrap()` retornará `undef`. Leia o Capítulo 10 para obter uma abordagem com mais detalhes das traps do SNMP.

# F

## SNMPv3

A segurança tem sido o ponto fraco do SNMP desde o início. A autenticação nas versões 1 e 2 do SNMP não significa mais do que uma senha (string de comunidade) enviada em texto explícito entre um gerenciador e um agente. Qualquer administrador de rede ou de sistema consciente da questão da segurança sabe as senhas em texto explícito não oferecem qualquer segurança. É comum a interceptação de string de comunidade por qualquer pessoa e, uma vez ocorrida, a senha pode ser utilizada para recuperar informações de dispositivos na rede, modificar as respectivas configurações e até derrubá-los.

O *Simple Network Management Protocol Version 3* (SNMPv3) lida com os problemas de segurança que infestaram o SNMPv1 e SNMPv2. Para todos os objetivos práticos, a segurança é a única questão que o SNMPv3 endereça; não ocorreram outras alterações no protocolo nem existem operações novas; o SNMPv3 tem suporte para todas as operações definidas nas Versões 1 e 2. Há várias convenções de texto novas que são apenas métodos mais precisos de interpretar os tipos de dados definidos em versões anteriores.

Este apêndice é uma introdução da SNMPv3 que discute a configuração do SNMPv3 para um roteador da Cisco e sobre o agente do Net-SNMP. Embora o SNMPv3 não seja um padrão completo, alguns fornecedores vendem produtos com suporte para o SNMPv3. Optamos por discutir duas implementações populares do SNMPv3 em nossos exemplos de configuração.

### Modificações no SNMPv3

Embora o SNMPv3 não modifique o protocolo, exceto ela inclusão de segurança codificada, seus desenvolvedores tentaram torná-lo diferente ao introduzir novas convenções de texto, conceitos e terminologia. As alterações efetuadas na terminologia são tão radicais, que é difícil acreditar que os novos termos descrevam fundamentalmente o mesmo software, como os antigos termos o faziam. Mas descrevem, sim. Entretanto, a diferença está no modo como se relacionam entre si e no fato de especificarem com muito mais precisão os componentes necessários a uma implementação do SNMP.

A alteração mais importante reside no fato de que a Versão 3 abandona a idéia de gerenciadores e agentes que, a partir dessa versão, passam a ser denominados *entidades* do SNMP. Cada entidade consiste em um mecanismo do SNMP e em uma ou mais aplicações do SNMP, o que serão discutido nas seções a seguir. Esses novos conceitos são importantes porque definem uma arquitetura, em vez de definir tão-somente um conjunto de mensagens; a arquitetura ajuda a isolar os diferentes componentes do sistema do SNMP para permitir uma implementação com segurança. Examinemos o significado desses conceitos, começando com as RFCs que os definem (Tabela F-1).

*Tabela F-1 RFCs do SNMPv3*

Nome	Número	Status	Data da Última Atividade
Architecture for SNMP Frameworks	RFC 2571	Draft	Abril de 1999
Message Processing and Dispatching	RFC 2572	Draft	Abril de 1999
SNMP Applications	RFC 2573	Draft	Abril de 1999
User-based Security Model	RFC 2574	Draft	Abril de 1999
View-based Access Control Model	RFC 2575	Draft	Abril de 1999
Protocol Operations for SNMPv2	RFC 1905	Draft	Janeiro de 1996
Transport Mappings for SNMPv2	RFC 1906	Draft	Janeiro de 1996
MIB for SNMPv2	RFC 1907	Draft	Janeiro de 1996
Coexistence Between SNMP Versões	RFC 2576	Sugerida	Março de 2000
Introduction to SNMPv3	RFC 2570	Informativa	Abril de 1999
Diffie-Hellman USM Key Management	RFC 2786	Experimental	Março de 2000

### *Mecanismo do SNMPv3*

O mecanismo é formado por quatro componentes: o Dispatcher (Escalonador), o Message Processing Subsystem (Subsistema de processamento de mensagens), o Security Subsystem (Subsistema de segurança) e o Access Control Subsystem (Subsistema de controle de acesso). A missão do Dispatcher é enviar e receber mensagens. Ele tenta detectar a versão de cada mensagem recebida (por exemplo, v1, v2 ou v3) e, se a versão for aceita, direciona a mensagem para o Message Processing Subsystem. O Dispatcher também envia mensagens do SNMP para outras entidades.

O Message Processing Subsystem prepara mensagens para serem enviadas e extrai dados das mensagens recebidas. Um sistema de processamento de mensagens pode conter diversos módulos de processamento de mensagens. Por exemplo, um subsistema pode conter módulos para processar solicitações do SNMPv1, SNMPv2 e SNMPv3, assim como um módulo para outros modelos de processamento que ainda serão definidos.

O Security Subsystem oferece recursos de autenticação e serviços de privacidade. A autenticação usa strings de comunidade (SNMP Versões 1 e 2) ou autenticação baseada em usuário do SNMPv3. A autenticação baseada em usuário utiliza os algoritmos MD5 ou SHA para autenticar usuários sem enviar uma senha explicitamente. Os serviços de privacidade usam o algoritmo DES para codificar e decodificar mensagens do SNMP. Atualmente, o DES é o único algoritmo utilizado, embora exista a possibilidade de incluir outros mais adiante.

O Access Control Subsystem responde pelo controle de acesso aos objetos da MIB. É possível controlar os objetos que o usuário pode acessar e as operações que executará nesses objetos. Por exemplo, convém limitar o acesso de leitura-gravação de um usuário a algumas partes da árvore *mib-2*, e permitir o acesso somente leitura à árvore inteira.

## *Aplicações do SNMPv3*

A Versão 3 divide em algumas aplicações a maior parte daquilo que supomos como SNMP:

### *Command generator (Gerador de comandos)*

Gera solicitações de *get*, *get-next*, *get-bulk* e *set* e processa as respostas. Essa aplicação é implementada por uma Network Management Station (NMS), que pode emitir consultar e solicitações de *set* para entidades em roteadores, comutadores, hosts do Unix etc.

### *Command responder (Replicador de comandos)*

Responde às solicitações dos comandos *get*, *get-next*, *get-bulk* e *set*. Essa aplicação é implementada por uma entidade em roteador da Cisco ou em um host do Unix. (Para as Versões 1 e 2, o replicador de comandos é implementado pelo agente do SNMP.)

### *Notification originator (Gerador de notificações)*

Gera traps e notificações do SNMP. Essa aplicação é implementada por uma entidade em um roteador ou um host do Unix. (Nas Versões 1 e 2, o gerador de notificações integra um agente do SNMP. Também existem utilitários gratuitos para gerar traps.)

### *Notification receiver (Receptor de notificações)*

Recebe traps e mensagens informativas. Essa aplicação é implementada por uma NMS.

## *Proxy forwarder (Direcionador proxy)*

Facilita a transmissão de mensagens entre entidades.

A RFC 2571 permite a definição de aplicações adicionais no decorrer do tempo. Essa possibilidade de estender a estrutura do SNMPv3 é uma vantagem significativa em relação às versões anteriores do SNMP.

## *Como é uma entidade?*

Até o momento, discutimos sobre a entidade SNMPv3 em termos de definições abstratas. A Figura F-1 (obtida na RFC 2571) mostra o conjunto de componentes que formam uma entidade.

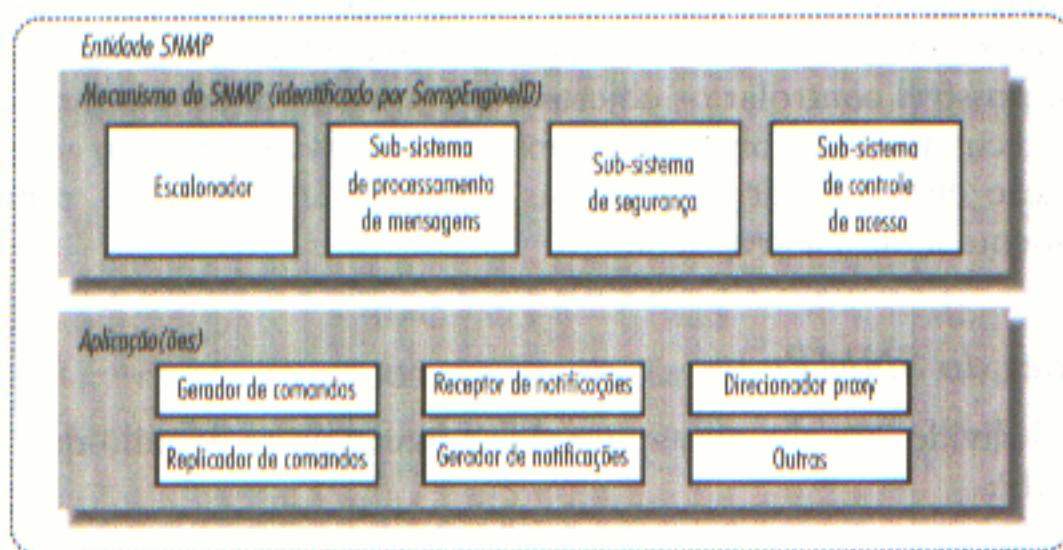


Figura F-1 Entidade SNMPv3

## *Convenções de texto do SNMPv3*

O SNMPv3 define diversas convenções de texto adicionais, descritas na Tabela F-2.

Tabela F-2 Convenção de texto do SNMPv3

Convenção de texto	Descrição
SnmpEngineID	Um identificador exclusivo, em termos administrativos, de um mecanismo do SNMP. Os objetos desse tipo servem para identificação, não para endereçamento, embora seja possível utilizar um endereço na geração de um valor específico. A RFC 2571 oferece uma discussão com detalhes do modo de criação das SnmpEngineIDs.
SnmpSecurityModel	Um securityModel (SNMPv1, SNMPv2 ou USM) do SNMP. USM significa <i>User-based Security Model</i> (modelo de segurança baseado no usuário), que é o método de segurança usado no SNMPv3.

Tabela F-2 Convenção de texto do SNMPv3

Convenção de texto	Descrição
SnmpMessageProcessing Model	Um modelo de processamento de mensagens usado pelo Subsistema de processamento de mensagens.
SnmpSecurityLevel	O nível de segurança em que as mensagens do SNMP podem ser enviadas ou o nível de segurança em que as operações estão sendo processadas. Os possíveis valores são noAuthNoPriv (sem autenticação e sem privacidade), authNoPriv (com autenticação mas sem privacidade) e authPriv (com autenticação e com privacidade). Esses três valores são solicitados de modo que noAuthNoPriv seja inferior a authNoPriv e authNoPriv seja inferior a authPriv.
SnmpAdminString	Uma string de octetos contendo informações administrativas, preferivelmente em formato legível pelo homem. A string pode ter até 255 bytes.
SnmpTagValue	Uma string de octetos contendo um valor de tag. Os valores de tag são preferivelmente no formato legível pelo homem. De acordo com a RFC 2573, as tags válidas são acme, router e host.
SnmpTagList	Uma string de octetos contendo uma lista de valores de tags. Os valores de tag são preferivelmente no formato legível pelo homem. De acordo com a RFC 2573, exemplos válidos de lista de tags são a string vazia, acme router e host managerStation.
KeyChange	Um objeto usado para modificar chaves de autenticação e privacidade.

## Configurando o SNMPv3

Agora, aplicaremos os conceitos do SNMPv3. Examinaremos dois exemplos: configuração de um roteador da Cisco e configuração de ferramentas do Net-SNMP em um sistema executando o Unix. Os conceitos são os mesmos para as duas entidades; a única diferença é o modo de configuração do SNMPv3.

A maioria das tarefas administrativas do SNMPv3 está relacionada ao gerenciamento de usuários e das respectivas senhas. Não será surpresa alguma que a tabela de usuários, senhas e outras informações de autenticação sejam apenas outra tabela do SNMP, denominada *usmUser*. A ID de objeto completa da tabela é *.iso.org.dod.internet.snmpV2 snmpModules snmpUsmMIB usmMIBObjects usmUser*; a forma numérica é *.1.3.6.1.6.3.15.1.2*.

## Configurando o SNMPv3 para um roteador da Cisco

O Capítulo 7 descreve como configurar o SNMP em um roteador da Cisco. Esta seção pressupõe que você já conhece o IOS e que não precisaremos explicar os conceitos básicos, como conectar-se ao roteador e acessar o modo privilegiado. A seção também presume que você já leu o Capítulo 7 e configurou o SNMP básico em seu roteador.

A primeira tarefa ao configurar o SNMPv3 é definir uma visão. Para simplificar, criaremos uma visão que permite acesso à subárvore *internet* completa:

```
router(config)#snmp-server view readview internet included
```

Esse comando cria uma visão chamada *readview*. Para limitar a visão à árvore *system*, por exemplo, substitua *internet* por *system*. A palavra-chave *included* declara que a árvore especificada deve ser incluída na visão; use *excluded* para excluir uma subárvore específica.

Em seguida, crie um grupo que usará a nova visão. O comando a seguir cria um grupo denominado *readonly*; *v3* significa que deve ser usado o SNMPv3. A palavra-chave *auth* especifica que a entidade deve autenticar pacotes sem codificá-los; *read readview* indica que a visão *readview* deve ser usada sempre que os integrantes do grupo *readonly* acessarem o roteador.

```
router(config)#snmp-server group readonly v3 auth read readview
```

Criemos agora um usuário. O comando a seguir gera um usuário denominado *kschmidt*, pertencente ao grupo *readonly*. *auth md5* especifica que o roteador deve utilizar o MD5 para autenticar o usuário (a opção possibilidade é *sha*). O último item na linha de comando é a senha ou frase-senha do usuário, que não pode ultrapassar 64 caracteres.

```
router(config)#snmp-server user kschmidt readonly v3 auth md5 mysecretpass
```

Essa configuração usa a codificação somente para impedir a transferência explícita das senhas. Os próprios pacotes do SNMP, que podem conter informações que não devem ser disponibilizadas ao público, são enviados sem codificação e podem, com isso, ser lidas por quem possuir um *sniffer* de pacotes e acesso à sua rede. Para avançar um pouco mais e codificar os pacotes, use um comando como este:

```
router(config)#snmp-server user kschmidt readonly v3 auth md5  
mysecretpass \  
priv des56 passphrase
```

As palavras-chave adicionais nesse comando especificam a privacidade (por exemplo, codificação para todos os pacotes do SNMP), uso da codificação de 56 bits do algoritmo DES 56, e uma frase-senha para ser usado ao codificar os pacotes.

As senhas e frases-senhas codificadas dependem da ID do mecanismo, de modo que se a ID do mecanismo mudar, você precisará excluir os usuários definidos (como conhecido comando *no* do IOS) e recriá-los (com os comandos *snmp-server user*). Por que o ID do mecanismo mudaria? É possível definir um ID de mecanismo na linha de comando do IOS. Você não deveria sequer definir o ID do mecanismo explicitamente mas se o fizer, será necessário excluir e recriar os usuários.

Esta foi a introdução mais resumida à configuração do SNMPv3 em um roteador da Cisco. Para obter mais informações, consulte a documentação da Cisco, disponível em <http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t3/snmp3.htm>.

## Configurando o SNMPv3 para o Net-SNMP

O Capítulo 7 descreve a configuração básica para o Net-SNMP. Nesta seção, discutiremos a configuração dos recursos da versão 3 do Net-SNMP. Primeiramente, abordaremos como configurar o SNMPv3, editando os arquivos *snmpd.conf*\*. Você deve instalar o OpenSSL antes de editar os arquivos, para usar o DES ou SHA. O OpenSSL está disponível em <http://www.openssl.org>.

Para criar um usuário denominado *kschmidt* que possui acesso leitura-gravação à subárvore *system*, adicione a seguinte linha ao arquivo *snmpd.conf*:

```
rwuser kschmidt auth system
```

Para criar um usuário com acesso somente leitura, use o comando *rouser* em vez de *rwuser*. A palavra-chave *auth* solicita autenticação protegida, mas não privacidade: os próprios pacotes do SNMP não são codificados. As outras opções são *noauth* (sem autenticação/privacidade) e *priv* (com autenticação e privacidade). Adicionemos a seguinte linha ao arquivo */var/ucd-snmp/snmpd.conf*:

```
createUser kschmidt MD5 mysecretpass
```

Essa linha cria uma senha do MD5 para o usuário *kschmidt*. A senha atribuída ao *kschmidt* é *mysecretpass*. Para criar um usuário com uma frase-senha do algoritmo DES adicionalmente a uma senha do MD5, inclua esta linha ao arquivo */var/ucd-snmp/snmpd.conf*:

```
createUser kschmidt MD5 mysecretpass DES mypassphrase
```

Se você omitir *mypassphrase*, o Net-SNMP definirá a frase-senha do DES idêntica à senha do MD5. A RFCs para o SNMPv3 recomenda que as senhas ou frases-senhas tenham no mínimo oito caracteres; o Net-SNMP impõe essa recomendação e não aceita senhas mais curtas.

\* Existem dois arquivos *snmp.conf* em cena, neste caso: o arquivo normal, */usr/share/snmp/snmp.conf*, e o arquivo persistente, */var/ucd-snmp/snmp.conf*. O arquivo persistente será discutido no momento oportuno.

Após efetuar essas modificações, interrompa e reinicie o agente. Ao ser reiniado, o agente lê o arquivo de configuração, determina as chaves secretas dos usuários adicionados e exclui os comandos createUser do arquivo. Em seguida, insere a chave secreta no arquivo de configuração. Esse comportamento tem algumas consequências. A chave secreta é baseada na ID do mecanismo que, para o Net-SNMP, baseia-se no endereço IP. Portanto, você não pode copiar os arquivos de configuração de uma máquina para outra. Além disso, se você modificar o endereço IP de uma máquina, será necessário reconfigurar o Net-SNMP: interrompa o agente, edite o arquivo `/var/ucd-snmp/snmpd.conf`, exclua as entradas que o Net-SNMP adicionou para seus usuários, adicione comandos createUser para recriar os usuários e inicie o agente novamente.

Agora, podemos executar um `snmpwalk` usando a autenticação da Versão 3. O comando a seguir especifica a Versão 3, com o nome de usuário `kschmidt`, solicitando autenticação sem privacidade por meio do algoritmo MD5. A senha é `mysecretpass`:

```
$ snmpwalk -v 3 -u kschmidt -l authNoPriv -a MD5 -A mysecretpass \
server.ora.com
system.sysDescr.0 = Linux server 2.2.14-VA.2.1 #1 Mon Jul 31 21:58:22 PDT 2000 i686
system.sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.linux
system.sysUpTime.0 = Timeticks: (1360) 0:00:13.60
system.sysContact.0 = "Ora Network Admin"
system.sysName.0 = server
system.sysLocation.0 = "Atlanta, Ga"
system.sysServices.0 = 0
system.sysORLastChange.0 = Timeticks: (0) 0:00:00.00
system.sysORTable.sysOREntry.sysORID.1 = OID: ifMIB
...
system.sysORTable.sysOREntry.sysORUpTime.9 = No more variables left in this MIB View
```

Observe que são exibidos apenas os objetos da subárvore `system`, embora o comando tente percorrer a árvore inteira. Essa limitação ocorre porque concedemos ao usuário `kschmidt` acesso restrito à subárvore `system`. Se o usuário `kschmidt` tentar consultar uma subárvore para a qual não possui acesso autorizado, receberá o seguinte resultado:

```
$ snmpwalk -v 3 -u kschmidt -l authNoPriv -a MD5 -A mysecretpass \
server.ora.com interfaces
interfaces = No more variables left in this MIB View
```

Se você deseja privacidade e autenticação, use um comando como este:

```
$ snmpwalk -v 3 -u kschmidt -l authPriv -a MD5 -A mysecretpass -x DES -X \
mypassphrase server.ora.com
```

Lembre-se de que para utilizar a privacidade do algoritmo DES, é necessário instalar a biblioteca OpenSSL.

## Usando o *snmpusm* para gerenciar usuários

O utilitário *snmpusm* do Net-SNMP é usado para a manutenção dos usuários do SNMPv3. O comando a seguir cria um usuário *kjs* clonando o usuário *kschmidt*:

```
$ snmpusm -v 3 -u kschmidt -l authNoPriv -a MD5 -A mysecretpass localhost  
create \ kjs kschmidt
```

Como *kjs* foi clonado de *kschmidt*, os dois usuários possuem a mesma autorização, senha e frase-senha. Evidentemente, é fundamental modificar a senha de *kjs*. Para isso, use o *snmpusm* com a opção *-Ca*. De modo semelhante, para modificar a frase-senha de privacidade, use *-Cx*. Os dois comandos a seguir modificam a senha e a frase-senha do novo usuário *kjs*:

```
$ snmpusm -v3 -l authNoPriv -u kjs -a MD5 -A mysecretpass localhost passwd \  
-Co -Ca mysecretpass mynewpass  
$ snmpusm -v3 -l authPriv -u kjs -a MD5 -A mysecretpass localhost passwd \  
-Co -Cx mypassphrase mynewphrase
```

Existem alguns aspectos a serem observados em relação a essa operação simples:

- Você deve conhecer a senha e frase-senha de *kschmidt* para configurar uma nova senha e frase-senha para o *kjs*.
- Segundo a documentação, o Net-SNMP permite clonar qualquer usuário específico somente uma única vez. Não está bem claro se isso significa que é possível criar somente um clone de um usuário ou após criar um clone, não é possível criar um clone desse clone. Seja qual for o caso, essa restrição não parece ser obrigatória.
- O *snmpusm* só pode clonar usuários; não pode criá-los desde o início. Portanto, você deve criar o usuário inicial manualmente, usando o processo descrito anteriormente. (Na realidade, isso não é uma verdade absoluta. O *snmpusm* pode criar um usuário mas assim que você fizer isso, deverá atribuir uma senha a esse usuário, alterando sua senha anterior. Então, você fica “num mato sem cachorro”: o novo usuário não possui uma senha; portanto, você não pode modificar a senha desse usuário.)

Para que o usuário conste no arquivo persistente *snmpd.conf*, você deve interromper e reiniciar o agente ou enviar um sinal HUP ao processo do *snmpd*. Essa ação instruirá o agente a escrever no disco o estado atual da tabela de usuários, para que o agente releia essa tabela ao reiniciar. Observe que o *kill -9* não gera o resultado almejado.

O comando *snmpusm* existe basicamente para permitir que os usuários finais gerenciem as próprias senhas ou frases-senhas. Como administrador, convém alterar periodicamente as senhas e frases-senhas de seus usuários. Isso só será possível se você mantiver uma lista principal de usuários e das respectivas senhas e frases senhas.

Se a ID do mecanismo for alterada, você deverá recriar os nomes de usuários, senhas e frases-senhas. (Lembre-se de que a ID do mecanismo depende, em parte, do endereço IP do host e, por conseguinte, mudará se você precisar modificar o endereço.) Para isso, interrompa o agente e edite o arquivo `/var/ucd-snmp/snmpd.conf`. Remova todas as entradas do `usmUser` persistente e adicione novos comandos `createUser` (conforme descrito anteriormente) para seus usuários. Uma entrada do `usmUser` é parecida com a seguinte:

```
usmUser 1 3 0x800007e580e134af77b9d8023b 0x6b6a7300 0x6b6a7300 NULL  
.1.3.6.1.6.3.10.1.1.2 0xb84cc525635a155b6eb5fbe0e3597873  
.1.3.6.1.6.3.10.1.2.2 0x1cf8d3cadd95abce8efff7962002e24 ""
```

### *Simplificando comandos por meio de definições de defaults (valores predefinidos)*

Nessa etapa, talvez você esteja questionando por que alguém usaria o SNMPv3, visto que os comandos são tão longos e complexos que é praticamente impossível digitá-los corretamente. Felizmente, há uma maneira de contornar esse problema. O Net-SNMP permite definir variáveis de configuração que os comandos utilizam ao serem executados. Crie um diretório em seu diretório inicial, denominado `.snmp`, e edite o arquivo `snmp.conf`. Adicione entradas como as seguintes:

```
defSecurityName kschmidt  
defAuthType MD5  
defSecurityLevel authPriv  
defAuthPassphrase mysecretpass  
defPrivType DES  
defPrivPassphrase mypassphrase  
defVersion 3
```

Os campos desse arquivo são:

`defSecurityName`

O nome do usuário do SNMPv3.

`defAuthType`

O método de autenticação (MD5 ou SHA).

`defSecurityLevel`

O nível de segurança do usuário. Os níveis válidos são `noAuthNoPriv`, `authNoPriv` e `authPriv`.

`defAuthPassphrase`

Sua senha, que deve ter pelo menos oito caracteres.

#### **defPrivType**

O protocolo de privacidade a ser utilizado. No momento, somente o DES é aceito.

#### **defPrivPassphrase**

Sua frase-senha de privacidade; desnecessária, se o nível de segurança for noAuthNoPriv ou authNoPriv. Deve ter no mínimo oito caracteres.

#### **defVersion**

A versão do SNMP a ser usada (neste caso, oSNMPv3).

Você também pode utilizar o comando *snmpconf* para configurar esse arquivo. O *snmpconf* solicita as diversas senhas e palavras-chave que devem constar no arquivo. Em nossa opinião, seria mais fácil você mesmo escrever o arquivo.

Após criar o *snmp.conf*, você pode utilizar os defaults para simplificar os comandos. Por exemplo, o comando:

```
$ snmpwalk -v3 -u kschmidt -l authPriv -a MD5 -A mysecretpass -x DES -X \
mypassphrase localhost
```

torna-se:

```
$ snmpwalk localhost
```

Esses defaults são aplicáveis a todos os comandos do Net-SNMP, incluindo o *snmpusm*.

### *Enviando traps do SNMPv3 com o Net-SNMP*

É fácil enviar uma trap do SNMPv3 com o Net-SNMP.\* Basta executar o *snmptrap* com as opções normais de traps do SNMPv2 combinadas com as opções do SNMPv3. Por exemplo:

```
$ snmptrap -v3 -l authPriv -u kjs -a MD5 -A mysecretpass -x DES -X
mypassphrase \ localhost '' .1.3.6.1.6.3.1.1.5.3 ifIndex i 2
ifAdminStatus i 1 ifOperStatus i 1
```

Definir as opções de configuração adequadas no *~/.snmp/snmp.conf* reduz bastante a complexidade do comando:

```
$ snmptrap localhost '' .1.3.6.1.6.3.1.1.5.3 ifIndex i 2 ifAdminStatus i
1 \ ifOperStatus i 1
```

---

\* As traps do SNMPv3 são apenas traps do SNMPv2 com recursos adicionados de autenticação e privacidade.

## *Considerações finais sobre o SNMPv3*

Embora os fornecedores tenham começado a fornecer suporte para o SNMPv3 em seus produtos, lembre-se de que ainda se trata de um padrão em draft, não um padrão completo (full). Para acompanhar os acontecimentos relacionados ao SNMPv3, você pode visitar o site do grupo de trabalho do SNMPv3 da Internet Engineering Task Force's (IETF), em <http://www.ietf.org/html.charters/snmpv3-charter.html>.

# Índice Remissivo

## Símbolos e Números

- (sinais de menos), em fluxos, 254
- ifref, opções, 243
- ! (pontos de exclamação)
  - em nomes de arquivos, 144-145
  - símbolos NOT lógicos, 83
- " (aspas duplas), para strings vazias, 269-270
- # (indicadores de linhas de comentários), 103, 233
- #include, instruções no C, 111
- & (E-comercial)
  - em instruções alvo do MRTG, 240
  - símbolos AND lógicos, 83
- (Consulte também OpenView Network Node Manager)
- \* (asteriscos), como caracteres curingas, 86, 142, 143
- . (pontos), separando nomes com, 15
- // (barras diagonais), linhas de comentários em filtros, 82
- /tmp, partições, monitorando, 146
- : (dois-pontos), em saída, 112
- ::= (operadores de definição), 15
- ? (pontos de interrogação), em comandos de roteador, 104
- \ (barras invertidas), em scripts, 136-137
- || (símbolos OR lógicos), 83
- 3Com
  - 3ComTotal Control, 66
  - RMON, provas e switches, 8
  - SuperStack, switches, 70
  - suporte para RS-232 MIB, 54

## A

- A3Com, 70
- Abreviações
  - em intervalos de polling, 146
  - snmpset, tipos de objeto, 268
  - tipos de dados Net-SNMP, 122, 175
- Abstract Syntax Notation One (consulte ASN.1)
- Access Control Subsystem (Mecanismo do SNMPv3), 289
- Acesso leitura-gravação, 14, 34-35
- Acesso somente gravação, 34-35
- Acesso somente leitura, 25
  - Event Categories, janela, 163-164
  - usuários, 293-294
- Acesso
  - bloqueando pacotes de SNMP, 91
  - desautorizando, 97
- listas de acesso, 103, 106
- agente do Windows NT/2000, 95
- UPSs, 107-108
- Ações automáticas
  - Automatic Alarms (SNMPc), 151-152
  - configurando no OpenView, 159
  - configurando no Trap Receiver, 168
- Agente de SNMP Extensível HP, 60
- Agentes bilingües, 54
- Agentes extensíveis, 179
  - Net-SNMP, 180-185
  - OpenView, 189-201
  - segurança e, 189
  - SystemEDGE, 185-189
- Agentes principais
  - e subagentes, 57
  - interrompendo e iniciando, 95
  - OpenView, snmpdpm, 179, 189-190
- Agentes, 5
  - agentes bilingües, 54
  - como monitores, 103
  - definições de parâmetros, 88-90
  - em pilha de protocolos, 11
  - enviando retenções (traps), 5, 153, 169-172
    - ganchos em programas, 176
    - instruindo o hardware a enviar retenções (traps), 176
  - Net-SNMP, 175
  - OpenView, 172
  - operações com traps, 37-41
  - Scripts em Perl, 173
  - SNMPv3, 282
  - Trap Generator, 174
- estendendo (consulte agentes extensíveis)
- limitando solicitações, 91, 102
- listando objetos gerenciados, 6
- listando traps, 153
- listas de acesso, 103, 106
- mestre e subagentes, 57, 102, 189-190
- MIBs, 6
- Net-SNMP, 97-102, 180-185
- nomes de comunidades, 13
- OpenView, 95-97, 189-201
- pacotes de software, 60
- polling externa e, 133-134
- polling interna e, 126
- questões de segurança, 90
- recursos de dimensionamento de memória, 33
- RMON, 127
- SystemEDGE, 102-104, 185-189
- tipos de memória em, 27

- UPSs, 106-109  
 Windows 9x, 92  
 Windows NT/2000, 93  
 (Consulte também Entidades)  
**AgentX**, 57  
 AIX, sistema operacional, 62, 66  
**Alertas**  
     aumentando alertas de limiar, 128, 129, 132-133  
     Automatic Alarms (SNMPc), 151  
     browser de alertas (OpenView), 163  
     configurando, 131  
     definindo limiares, 140  
     falhas secundárias, 77  
     queda de alertas de limiares, 128, 129, 132-133  
 Algoritmo de DES, 289, 292, 293  
 APC Symetra, 106-109  
 Aplicando zoom em mapas, 74, 85  
 Aplicativos específicos da função, 65  
 Aplicativos NMS de linha de comando, 31  
**Aplicativos**  
     Agentes SNMP, 60  
     específico do fornecedor, 65  
     gerenciadores de elementos, 65  
     NMS, coleções, 61  
     OpenView NNM (consulte OpenView Network Node Manager)  
         retenções (traps) e, 60  
     SNMPc Enterprise Edition (consulte SNMPc)  
         software de análise de tendências, 67  
         software de suporte, 68  
 Aprisma, 66  
 Arquivo config.h (Net-SNMP), 100  
 Arquivo EXAMPLE.conf, 102, 180  
 Arquivos .adm, 92  
 Arquivos de áudio, reproduzindo, 159  
 Arquivos de registro em NNM, 258  
 Arquivos iniciais, 78  
**Arquivos**  
     arquivos de configuração  
         copiando, 293  
         recarregando, 295-296  
     arquivos de filtro, 82  
     arquivos de inscrição/registro (NNM), 258  
     arquivos de log, 98-99, 234, 238-239  
     arquivos de menu (NNM), 257  
     arquivos de MIB, 19-25, 282  
     arquivos de som, 159  
     arquivos HTML, 240-241  
     arquivos iniciais, 78  
     em scripts, 282  
     escrevendo saída em, 269  
     listas de polling, 222  
     mantendo atualizados, 200  
**Arrays**  
     tabelas e agentes extensíveis, 194-201  
**Árvores de objetos** (consulte MIBs, subárvores)  
**ASN.1**, 15  
     agentes extensíveis e, 179, 189-190  
     distinguindo maiúsculas de minúsculas, 191  
**Aspas duplas ("")**, para strings vazias, 269  
 Atualizações, evitando polling durante, 126  
**Atualizando**  
     arquivos, 200  
     dados de tabela, 199  
     gráficos, 147-148  
 Autenticação baseada no usuário no SNMPv3, 289  
**Autenticação, SNMPv3**  
     codificação, 292  
     definições padrão, 296, 297  
     métodos e algoritmos, 289, 292, 293  
     níveis, 290
- B**
- Backups**  
     evitando polling durante, 126  
     NNM e arquivos de backup, 257, 258  
**Basic Encoding Rules (BER)**, 15, 111  
**Blocos livres**  
     pesquisando, 215  
     verificando, 222  
**Bluebird**, 231  
**BMC NMS**, coleção, 63  
**BOOTP**, filtros, 81
- C**
- C**, ganchos para programar em, 126, 178  
**C++**, 71  
**Cabletron**, software de gerenciamento, 66  
**Campos de recursos**, 259  
**Caracteres curingas**  
     filtros, 86  
     NNM, definições, 78  
     OpenView, polling, 141-142, 143  
     UPS, configuração, 107  
**Caracteres de nova linha**, 186, 195-196  
**Cartões 10/100**, 246  
**Castle Rock SNMPc** (consulte SNMPc)  
**Categorias de eventos em NNM**, 157  
     All, 162-163  
     Error, 162, 163  
     padrão, 160  
     personalizadas, 160  
**CCITT** (International Telegraph and Telephone Consultative Committee), 15  
**-Cf**, opção (Net-SNMP), 271  
**cfgmaker**, ferramenta (MRTG), 232  
**CGI** (Common Gateway Interface), 51  
**-CH**, opção (Net-SNMP), 272  
**-Ch**, opção (Net-SNMP), 272  
**-Ci**, opção (Net-SNMP), 272  
**Cisco**  
     CiscoWorks 2000, 66  
     configurando dispositivos, 104  
         ativar modo, 104  
         desativando encerramentos, 105  
         envmon, opção, 105

modo de configuração, 132-133  
SNMPv3 para roteadores, 291  
implementação de ROM, 129-133  
número de empresa privada, 16  
procurando MIBs, 266  
Clonando usuários, 294  
Codificação  
algoritmo de DES, 289, 292, 293  
definições padrão, 297  
falta de no SNMP, 90  
IDs de mecanismos e, 292  
SNMPv3, 289, 292  
Virtual Private Networks, 14  
Codificando  
BER, 15  
mensagens de erro, 36  
coldStart, retenções (traps), 39-40, 153  
Colunas em tabelas  
adicionando, 26  
identificadores, 195-196  
Common Gateway Interface (CGI), 51  
Compatibilidade com o SNMP, 53-56  
Compiladores da linguagem C do ANSI, 65  
Compilando MIBs, 193-194  
OpenView, 193-194  
SNMPc, 86  
Components Wizard (Windows), 93  
Computer Associates Unicenter TNG Framework, 64  
Comunicação gerente-a-gerente, 40-41  
Comunidades, 13  
configurando  
para agentes do Windows, 92, 95  
para dispositivos Cisco, 105  
para OpenView, 96  
para SNMPc, 86  
mais recomendável para strings, 13  
nomes de comunidades de agentes, 13  
SNMPv1, 4  
(Consulte também Strings de comunidades)  
Concise MIB Definition, 22  
Concord Communications, 54  
eHealth, 67  
Empire MIB, 103  
SystemEDGE (consulte SystemEDGE)  
Configuração  
agentes Cisco em dispositivos, 104-106  
agentes do Windows, 92, 93  
definições de parâmetros, 88-90  
MRTG, 231-235  
Net-SNMP (consulte Net-SNMP)  
OpenView (consulte OpenView; OpenView Network Node Manager)  
RMON, 129-133  
SNMPc, 84-87  
SNMPv3, 291-297  
SystemEDGE (consulte SystemEDGE)  
UPSs, 106-109  
Confirmar  
eventos (OpenView), 163-164  
recepção de retenções (traps), 41, 271-272

Conjunto de dados  
atualizando gráficos, 147-148  
elaborando coleções, 143-146  
excluindo arquivos, 144-145  
exibindo dados, 147-148  
ferramentas de origem aberta, 152-153  
(consulte também MRTG)  
intervalos de polling, 145  
limiares, 144-148  
requisitos de hardware, 46  
restringindo, 145  
SNMPc, 148-149  
testando eventos e limiares, 147-148  
xnmgraph e, 133-142, 147-148  
Considerações sobre o hardware  
fazendo upgrade, 56  
NMSs, 45-47  
Contatos do sistema (consulte sysContact, parâmetro)  
Contatos para dispositivos  
definindo  
Net-SNMP, 99  
OpenView, 120  
recuperando com OpenView, 113  
sysContact, 89  
Convenções para texto  
SMIv2, 26  
SNMPv2, 26  
SNMPv3, 290  
Cores  
categorias de eventos em OpenView, 159  
mapear cores em NNM, 80  
níveis de severidade de eventos e, 158, 162-163  
CPUs  
grafando utilização, 256  
requisitos do NMS, 46  
-Cu, opção (Net-SNMP), 270  
-Cw, opção (Net-SNMP), 272

## D

Dados estatísticos em polling do OpenView, 134-135  
Dados estatísticos sobre desempenho (grupos da MIB-II), 29  
Dados externos em gráficos, 253-256  
Datagramas perdidos, 10  
Datagramas, 10  
Datas  
datas do sistema, 42  
DateAndTime, convenção, 27  
Declarações Action (NNM), 259  
Demos como agentes, 5  
Descartando  
alarmes, 131  
eventos, 129-130  
DeskTalk Systems, Inc., 67  
DHCP  
filtros, 81  
opções de polling, 75

Digital Unix, sistema operacional, 63  
Diretórios de trabalho (MRTG), 232-233  
Diretórios, estrutura no OpenView, 72  
Dispatcher (Mecanismo do SNMPv3), 288  
Dispositivos de IP  
    agentes, 5  
    gerenciando com o SNMP, 1  
Dispositivos do Internet Protocol (consulte Dispositivos de IP)  
Dispositivos gerenciáveis, 57  
Dispositivos iniciais, 85, 86  
Dispositivos  
    aplicativos da suite NMS, 61  
    aplicativos específico do fornecedor, 65  
    compatibilidade com o SNMP, 53-56  
    contando objetos da MIB em, 119  
    definições de parâmetros, 88-90  
    descobrindo com o SNMPC, 85  
    dispositivos não-SNMP, 242  
    enviando retenções (traps), 169-178  
        ganchos em programas, 176  
        instruindo o hardware a enviar retenções (traps), 176  
    Net-SNMP, 175  
    OpenView, 172  
    Scripts em Perl, 173  
    Trap Generator, 174  
    fazendo upgrade, 56  
    gerenciadores de elementos, 65  
    gerenciamento do SNMP, 1  
    localizações, 88  
    MRTG e, 232, 233  
    nomes de domínio totalmente qualificados, 89  
    polling  
        externa, 133-152  
        interna, 126-133  
        por tipo, 135-136  
    recursos para manutenção, 3  
    segurança, 13, 90, 91  
    software de agente em, 60  
    software de análise de tendências, 67-68  
    software de suporte, 69  
    tipos gerenciáveis, 57  
    versões do SNMP em, 54  
Distributed Management Task Force, 52  
Diversos valores, recuperando, 116  
DNS (Domain Name System)  
    objetos SystemEDGE, 186

## E

Editor do Arquivo de Registro, 93  
egpNeighborLoss, retenção (trap), 40  
Emacs, editor de texto, 176  
Emissor proxy, 289  
Encerrando roteadores, 105  
Endereços (consulte Endereços IP, arquivos iniciais)  
Endereços de IPv4, de 32 bits, 18  
Endereços de IPv6, de 128 bits, 18

Endereços IP  
    configurando propriedades no NNM, 78  
    conjunto de dados, 143  
    em arquivos iniciais, 78  
    em listas de acesso, 106  
    enviando retenções (traps) para, 170, 172  
    filtrando, 83  
    nomes de host e, 89  
    para os dispositivos iniciais, 86  
    questões relacionadas à SMI, 18  
    recuperando com o Net-SNMP (snmpstatus), 271  
        UPSS, 106  
Endereços no nível físico, 26  
Entidades (SNMPv3), 288, 290  
Enviando retenções (traps), 169-178  
    dispositivos Cisco, 105  
    ganchos em programas, 176  
    instruindo, 176  
    Net-SNMP, 175, 297  
    OpenView, 172  
    Perl, 173  
    SNMPv3, 289, 297  
    testando, 176  
    Trap Generator, 174  
envmon, opção (Cisco), 105  
Equipamento (consulte Dispositivos; NMSs)  
Equipamento de ar-condicionado, 57, 126  
Espaço de disco  
    grafando utilização, 230  
    informações sobre recursos do host, 42  
    requisitos do NMS, 46  
    verificando com o Net-SNMP, 184  
    verificando com o OpenView, 214  
Espaço em branco  
    em scripts, 136  
    em tabelas, 195-196  
espelhos, 209  
Estações de gerenciamento (consulte NMSs)  
Ethernet  
    consultando interfaces com o MRTG, 237-238  
    endereços, 26  
    estatística de ROM, 43  
    recuperando informações da interface com o Perl, 116  
Event Categories (OpenView NNM), 162-163  
    mapear cores e, 80  
    níveis de severidade, 158  
        modificando, 163-164  
    usando para mensagens, 252  
Event Configurations GUI (xnmtrap), 155  
Event Configurator (OpenView), 155-160  
Eventos específicos de empresa, 164  
Eventos privados (OpenView), 164  
Eventos  
    ações e, 159  
    browser de alarmes, 163-164  
    encaminhando, 49, 158  
    número médio de retenções (traps) para, 159  
    OpenView  
        acionando por retenções (traps), 155  
        atribuindo a categories, 157

categorias predefinidas, 157  
configuração, 155-160  
configurando para limiares, 146-147  
confirmando e anulando confirmação, 163-164  
criando eventos privados, 164-165  
encaminhando, 158  
excluindo, 163-164  
exibindo, 155, 162-163  
limitando origens, 157  
níveis de severidade, 158  
registrando, 155, 158, 159  
testando, 147-1489  
variáveis, 160

pilha de protocolos, 11

RMON

alarmes *versus* eventos, 127  
configuração do Cisco, 129  
registrando, 128, 129

Exceções, definindo no NNM, 78

Excluindo

arquivos de conjunto de dados antigo, 144-145  
eventos, 163-164  
nós e mapas, 84

Exemplo animal.db, 195-196

Exibindo

dados em coleções, 147-148  
eventos, 155, 162-163  
gráficos MRTG, 236-237  
retenções (traps), 167

Exigências para armazenamento de dados, 46

Expressões (consulte Expressões regulares)

Expressões regulares

correspondência, 141-142  
limitando polling com, 137-138, 145  
para instâncias, 141-142

extensionGroup, tabela (SystemEDGE), 186

Extensões do Registro, 188-189

extsubagt, utilitário (OpenView), 194

## F

FAQs para SNMP, 9

Fila de mensagens

resumindo mensagens em, 194  
verificando status, 189-190

FILE-COMMAND, opção (OpenView), 200

Filiais, 16

FilterExpressions (OpenView), 82, 83

Filtros

atributos, 83  
combinando, 83  
configurando no NNM, 81-84  
DHCP, opções de polling, 75  
reduzindo tráfego com, 81  
SNMPc, filtros, 85, 86

Firewalls, 13, 91

Firmware, fazendo upgrade, 56

Flags (Net-SNMP), 182

Flags de erro (Net-SNMP), 182

Fluxos em gráficos (NNM), 253

Fórmulas

Fornecedores (consulte fornecedores de MIBs)

aplicativos específico do fornecedor, 65

atribuições de números de empresa, 16

MIBs específicas de fornecedor, 7, 18

carregando no OpenView, 84

strings de comunidades padrão, 13, 55

FQDNs (nomes de domínio totalmente qualificados), 89

## G

Ganchos em programas, 126, 176

Gateways

como dispositivos iniciais, 85

em arquivos iniciais, 78

gd, biblioteca, 231

Gerador de comandos (SNMPv3), 289

Gerando retenções (traps) (consulte Enviando retenções (traps))

Gerenciadores de elementos, 65

Gerenciamento de rede baseado na Web, 51, 91

Gerenciando redes

cenários antes/depois, 2

monitorando, 2

SNMP, função em, 1

GIF, imagens, 231

GPL (GNU Public License), 65

Gráficos em preto-e-branco, 254

Gráficos setoriais (SNMPc), 151

Gráficos

limiares de alarme e, 139-140

MRTG, 230, 234, 236-241

OpenView xnmgraph, 134-142, 253-256

atualizando, 147-148

páginas de índice para, 236

polling externa e, 133

polling interna e, 133

preto-e-branco, 254

SNMPc, 148-152

Grupos de gerenciamento (MIB-II), 29

GTK/GDK, kit de ferramentas, 64

GxSNMP, coleção, 64

## H

-h, opção (Net-SNMP), 265

Hardware, instruindo o envio de retenções (traps), 176

Horas

DateAndTime, convenção, 27

incluindo em retenções (traps), 175

TimeInterval, convenção, 27

TimeStamp, convenção, 27

unidades em intervalos de polling, 145

Hosts e gerenciamento de hosts, 7, 42

consultando, 270-271

em pilha de protocolos, 11

- enviando retenções (traps), 170  
OIDs do host, 42  
polling  
    Net-SNMP, 115  
    OpenView, 113, 144  
    Perl, 111, 116  
    recuperando nomes de host no Perl, 111  
HP OpenView (consulte OpenView)  
HP OpenView NNM (consulte OpenView Network Node Manager)  
HP-UX, sistema operacional,  
    coleções do NMS, 62, 63  
    OpenView e, 96-98  
software agente, 80  
software de análise de tendências, 67  
software de suporte, 70  
software específico do fornecedor, 66  
HTML  
    caminhos de arquivos, 236  
    em saída de MRTG, 233, 235  
    formulário para enviar mensagens, 261  
    títulos em páginas, 236  
HTTP  
    descobrindo suporte para (SNMPc), 85  
    NMS's baseados na Web e, 51  
HUP, sinais, 181, 194, 295  
HyperText Transport Protocol (consulte HTTP)
- I**
- iadmin, script, 203  
IANA (Internet Assigned Numbers Authority)  
    atribuições de números de empresa, 16  
    listas de fornecedores de MIBs, 9  
IDs de mecanismos, 290  
    codificação e, 292  
    dados do usuário e, 295-296  
IDs de objetos (consulte OIDs)  
IETF (Internet Engineering Task Force)  
    extensões de agente, 57  
    grupo do SNMPv3, 298  
    RFCs, 3  
ifAdminStatus, objeto (MIB), 131, 175  
ifDescr, objeto (MIB), 131, 245  
ifEntry, objeto (MIB), 245  
IfEntry, sequência (MIB), 23, 24  
ifIndex, objeto (MIB), 175  
ifInOctets, objeto (MIB), 245-252  
    grafando, 136, 237, 252  
    medidas de largura de banda, 247  
    MRTG e, 234  
    polling no OpenView, 134-135  
ifOperStatus, objeto (MIB), 131, 138, 175, 245  
ifOutOctets, objeto (MIB), 245-252  
    grafando, 136, 237, 252  
    medidas de largura de banda, 247  
    MRTG e, 234  
    polling no OpenView, 134  
ifSpeed, objeto (MIB), 245, 246, 247  
ifTable, objeto (MIB), 23, 24  
ifType, objeto (MIB), 130-131, 138, 245  
Ignorando  
    dispositivos (consulte filtros)  
    retenções (traps), 163-164  
Importando  
    dados externos, 253-256  
    tipos de dados, 21  
Imprimindo (Net-SNMP)  
    árvore de OIDs, 274  
    informações de status, 270-271  
    OIDs, 265  
Índices  
    em tabelas de monitoramento de processos, 103  
    índices de inteiros em tabelas, 196  
    RMON, tabelas de eventos, 130-131  
Informações de gerenciamento (consulte Objetos gerenciados)  
Informações de status (Net-SNMP), 270-271  
InfoVista, 69  
Instalação  
    MRTG, 231  
    Net-SNMP, 97-98  
    SNMPc, 85  
Instâncias  
    conjunto de dados e, 145  
    identificadores, 31, 196  
    números, 118, 130-131, 214-215  
    rotulando, 142  
Instruções use em Perl, 111, 281  
Inteiros, convertendo em strings, 199  
Interfaces da Web para NNM, 260  
Interfaces de loopback, 118  
Interfaces gráficas  
    NMS, aplicativos, 31  
    OpenView, 136-137  
Interfaces  
    calculando necessidades de armazenamento de dados, 47  
    dados estatísticos, MIB-II, 6  
    enviando retenções (traps) com (Cisco), 105  
    informações sobre retenções, 39-40  
    largura de banda de, 246  
    listando, 118  
    MRTG e, 233, 243-244  
    mudanças de status, 138-139  
    Pontos de saturação, 252  
    retenções de mudança de estado, 153  
    tabelas de objetos gerenciados, 22-23  
International Organization for Standardization (consulte ISO)  
International Telegraph and Telephone Consultative Committee (CCITT), 15  
Internet Assigned Numbers Authority (consulte IANA)  
Internet Engineering Task Force (consulte IETF)  
Internet  
    pesquisando na, 47  
    questões de segurança e, 50, 91  
Intervalos de polling, 125  
    fatores em, 126  
Net-SNMP (snmpdelta), 269

OpenView  
    NNM, 78-80, 146  
    xnmgraph, 134-136, 140

RMON, 131

SNMPc, 85, 144

SystemEDGE, 103

Intervalos  
    calculando necessidades de armazenamento de dados, 46  
    descoberta, 74, 85  
    polling, 125  
        fatores em, 126  
        Net-SNMP (snmpdelta), 269  
        OpenView, 78-80, 134, 135-136, 140  
        RMON, 131  
        SNMPc, 85, 144  
        SystemEDGE, 103

TimeInterval, convenção, 27

Intrusos, impedindo, 13

IOS, sistema operacional,  
    dispositivos Cisco e, 104  
    RMON, eventos e, 129

IP, filtros, 13

-IR, opção (Net-SNMP), 264, 274

ISO (International Organization for Standardization)  
    administração de subárvore, 15  
    camadas em sysServices, 95

## L

-l, opção (Net-SNMP), 270

Largura de banda  
    como fator em intervalos de polling, 126  
    medindo, 246-252

Leinen, Simon, 166, 281

libpng, biblioteca, 231

Licenciando  
    OpenView NNM, 72  
    SNMPc, 85

Limitando  
    acesso  
        a agentes, 106  
        à árvore da MIB, 97-98  
        a NMSs, 102  
        (Consulte também Acesso)

alarmes no SNMPc, 151-152

tipos de retenções enviadas (Cisco), 106

Limiares  
    definindo mínimos para servidores, 222  
    expandindo ou reduzindo em SNMPc, 151

OpenView  
    conjunto de dados, 142-148  
    criando (NNM), 145  
    definindo com entrada de gráfico, 140  
    eventos, 146-147, 157  
    monitorando em conjunto de dados, 144-15  
    rearm, parâmetro, 146-147  
    testando, 147-148

polling e, 124

RMON, 127  
    alarmes, 128  
    aumentando limiares, 129, 132-133  
    eventos, 128  
    octetos de saída, 131-132  
    queda de limiares, 129, 133  
    valores absolutos, 131-132  
    valores delta, 131-132

SNMPc  
    Automatic Alarms, 151  
    períodos de aprendizagem, 151

Linhas de base, definindo

SNMPc  
    Automatic Alarms, 151  
    períodos de aprendizagem, 151

xnmgraph, 140

Linhas de comentários  
    em arquivos de configuração, 103  
    em filtros, 82  
    em MRTG, 233

Linhas em tabelas, 24, 27, 195-196

Linhas Ethernet 10BaseT, 246

Linhas seriais, velocidade, 246

Linhas, velocidade máxima de, 246

linkDown, notificação ou retenção (SNMPv2), 40, 41, 153

Categorias de Eventos e, 176  
desconectando portas e, 176  
objetos em, 175  
respostas a, 154

links privativos em arquitetura NMS, 50

linkUp, notificação ou retenção (SNMPv2), 40

Linux, sistema operacional,  
    snmptrap, ganchos em programas, 178  
    SystemEDGE, 81, 102

Listando  
    diretórios de pesquisa de arquivos de MIB, 264-265  
    interfaces, 118  
    números de porta, 225  
    objetos gerenciados, 6  
    sistemas de arquivos, 222  
    tipos de retenção (trap), 104, 153, 176

loadhosts, utilitário (NNM), 78

Local Registration Files (LRF), 78

Localizações no sistema (consulte sysLocation, parâmetro)

Logins, registrando, 203

LOGONLY, modo, padrão para retenções (traps), 166

LRF (Local Registration Files), 78

## M

-M, opção (Net-SNMP), 264

-m, opção (Net-SNMP), 264, 266

-m, opção (snmpdelta), 270

Mac OS, 69

MAC, camada em pilha de protocolos, 11

Management Information Bases (consulte MIBs)

- Mapa Root  
    NNM, 73  
    SNMPc, 85
- Mapas  
    OpenView  
        aplicando zoom, 85  
        cores, 80  
        de objetos não gerenciados, 77  
        removendo nós, 84
- SNMPc  
        aplicando zoom, 74  
        mapa Root, 85  
        submapas, 85
- MD5, algoritmo, 289, 292, 293
- Mecanismos de correlação de eventos  
    limiares de polling e, 124  
    Netcool, 71
- Médias de carregamento, grafando, 230
- Medium Access Control (MAC), camada em pilha de protocolos, 11
- Memória  
    gerenciamento do SNMP, 7  
    grafando utilização, 230  
    requisitos de NMS, 45, 46  
    tipos para agentes, 27  
    totais do sistema, 41-42
- Mensagens de erro  
    consultando e definindo respostas, 123  
    mensagens do SNMPv1, 36  
    mensagens do SNMPv2, 36  
    Net-SNMP, 182
- Mensagens pop-ups, 159
- Mensagens  
    dimensionar recursos de agentes, 33  
    enviando com NNM, 260
- Menus (NNM), 258
- Menus  
    adicionando ao OpenView, 256-259  
    caracteres acionadores em, 258  
    nímeros de precedência em, 258  
    personalizando no SNMPc, 151
- Message Processing Model, 290
- Message Processing Subsystem (Mecanismo do SNMPv3), 288
- MIB Browser (OpenView), 113, 118
- MIB de recursos do host, 7, 42
- mib-2, subárvore, 28-29
- MIBs (Management Information Bases), 6  
    agentes extensíveis e, 179, 185, 188  
    analizando, 282  
    arquivos como argumentos em scripts, 282  
    carregando, 84, 86, 263, 264  
    componentes de, 19-25  
    contando objetos implementados em dispositivos, 119  
    copiando, 266  
    definições de dados de tabelas, 195-196  
    draft e padrões sugeridos, 6  
    específico do fornecedor, 19  
    listando retenções (traps), 153  
    listas de fornecedores, 9  
    MIB de recursos do host, 7, 42
- MIB-II, 6  
        componentes de arquivos, 19-25  
        objetos, 245  
        subárvore em, 28-29
- MIBs do Cisco, 266  
        objetos gerenciados e tipos de dados em, 19  
        OIDs e identificadores de instâncias, 31  
        pares de objeto-valor (consulte Vinculações de variáveis)
- Perl, rotinas de gerenciamento, 281  
    privadas, 17, 54, 185  
    procurando em diretórios, 264  
    proprietárias, 7  
    recompilando, 187-188  
    RMON, MIB, 8, 43  
    seqüências em, 24
- MIBs privadas, 17, 54, 185
- MIBs proprietárias, 7
- Micromuse, 71
- Microsoft Windows (consulte Windows 9x/NT/2000, sistemas operacionais)
- Mitchell, Mike, 281
- Modo full-duplex  
        identificando velocidade da linha, 246  
        medidas, 247
- Modo half-duplex  
        identificando velocidade da linha, 246  
        medidas, 247
- Monitorando redes  
        RMON (consulte RMON)  
        (Consulte também NMSs; polling)
- MRTG (Multi Router Traffic Grapher), 68, 230-244  
        arquivos de log, 234, 238  
        gráficos, 230  
            armazenando, 232  
            especificando variáveis, 239  
            exibindo, 235-238, 240  
            opções de filtro, 240  
            opções de legenda, 240  
            padrões, 230, 234, 238  
            parâmetros, 234, 240  
            roteadores, 235-237  
            servidores, 237-241  
        instalando e executando, 231-235  
        Perl e, 243-244  
        pesquisando com, 231  
        pesquisando dispositivos não-SNMP, 242-243  
        problemas ao mudar interfaces, 243-244  
        recursos e Ajuda, 244-245  
        recursos, 230
- Multi Router Traffic Grapher (consulte MRTG)
- Multiplicadores(xnmgraph), 134, 140, 142

## N

- Navegador veloz (NNM), 74
- Necessidades de armazenamento de dados, 47
- NerveCenter, 64
- netcat, 225-229

- Netcool, 71  
netmon (NNM, processo demo), 74-78  
arquivos iniciais, 78  
filtros, 81-84  
problemas de polling, 79  
switches em, 80  
testando roteadores, 78  
Net-SNMP, 61  
arquivos de MIB, 263  
configurando, 97-102  
arquivos de configuração, 101, 116, 180, 271-272, 293, 295-296  
SNMPv3, 293-297  
enviando retenções (traps), 175, 268  
estendendo, 179-185  
ferramentas e utilitários de linha de comando, 265-274  
instalação, 97-98  
mensagens de erro, 182  
opções de linha de comando, 264  
operações de get, 115, 267  
operações de set, 121, 268  
polling com, 115  
recebendo retenções (traps), 169, 174  
sintaxe de comandos, 263  
snmpsum, 294  
tipos de dados, 122-123, 175  
tipos de objetos (snmpset), 268-269  
Net-SNMP, Biblioteca do C, 70  
Net-SNMP, Módulo de Perl 70  
Netview (Tivoli), 63  
Network Computing Technologies Trap Generator, 174  
Network Computing Technologies Trap Receiver, 71, 168  
Network Information System (NIS), 186  
Network Management Server, Universidade em Buffalo, 8, 59  
Network Node Manager (consulte OpenView Network Node Manager)  
Network Operations Centers (NOCs), 3  
NetworkAddress, tipo de dado (SMIV1), 18  
Newsgroups, Usenet, 8, 244-245  
NIS (Network Information System), 186  
Níveis de segurança, 290, 296-297  
Níveis de severidade de eventos (OpenView), 158  
NMS, aplicativos, 59  
(Consulte também NMSs)  
Agentes SNMP, 60  
específico do fornecedor, 65  
gerenciadores de elementos, 65  
OpenView Network Node Manager, 72-84  
SNMPc Enterprise Edition, 84-87  
software de análise de tendências, 67  
software de suporte, 69-71  
suites, 61  
NMS, requisitos de RAM, 45  
NMS's de arquitetura híbrida, 49  
NMSs (Network Monitoring Stations)  
agentes extensíveis e, 187  
aplicativos (consulte NMS, aplicativos)  
aplicativos gráficos versus aplicativos de linha de comando, 31  
arquitetura, 45, 47-51  
baseadas na Web, 51  
comunicação gerente-a-gerente, 41  
considerações sobre o hardware, 45-47  
encerrando roteadores, 105  
função em pilha de protocolos, 11  
impedindo mudanças efetuadas por duas estações, 27  
links privados e, 50  
OpenView Network Node Manager, 72-84  
operações de SNMP, 29-41  
pacotes de origem aberta, 231  
polling externa, 124, 133-152  
polling interna, 126-133  
portas, 11  
recebendo e manipulando retenções (traps), 153, 154  
Net-SNMP, 169  
OpenView, 155-166  
polling orientada por retenções, 51  
scripts em Perl, 166-168  
seqüência de geração de retenções (traps) e, 37-38  
Trap Receiver, 166  
(Consulte também recebendo retenções (traps))  
RMON, 8, 127-133  
SNMPc Enterprise Edition, 84-87  
SNMPv3, gerador de comandos e, 289  
NNM (consulte OpenView Network Node Manager)  
No-breaks (uninterruptible power supplies - UPSs), 57, 106-109  
NOCs (Network Operations Centers), 3  
Nomes de domínio totalmente qualificados (FQDNs), 89  
Nomes de domínio  
FQDNs, 89  
SystemEDGE, objetos estendidos, 185-186  
Nós de folha, 15  
Nós inatingíveis, 75, 77, 79, 123  
Nós  
adicionando a mapas do NNM, 78  
descobrindo com o netmon, 74  
disparando retenções (traps) quando desativadas, 156  
em filtros, 82, 83  
em hierarquias, 15  
grafando dados (consulte gráficos)  
limitando retenções (traps) em, 157  
necessidades de armazenamento de coleta de dados, 46  
nós inatingíveis, 75, 77, 79, 123  
OpenView, polling, 142, 143  
configurando intervalos, 78-80  
netmon, 74-78, 81  
removendo de mapas do NNM, 84  
SNMPc, polling, 85  
Notificações  
comparação com retenções (traps), 154, 269-270

compatibilidade com o SNMP e, 54  
OpenView, pop-ups, 159  
variáveis em, 160  
operações, 41  
SNMPv3, 289  
emissor de notificações, 289  
receptor de notificações, 289  
NT (consulte Microsoft Windows 9x/NT/2000, sistemas operacionais)  
nttrapgen, utilitário (Network Computing Utilities), 174  
Números de empresa privada, 16, 17  
Números de empresas, privadas, 17  
em retenções (traps), 156  
gerenciando, 16  
para estender OpenView, 189-190  
restringindo polling para, 144  
retenções (traps) específicas de empresa e, 153  
(Consulte também IANA)  
Números de retenções (traps) genéricas, 38, 39  
Números, convertendo em strings, 199

## O

-O, opção, 273  
Objetos de tabelas  
conjunto de dados e, 145  
OIDs, 31, 123  
Objetos escalares  
conjunto de dados e, 145  
grafando em MRTG, 240  
OIDs, 31, 185  
Objetos gerenciados, 6, 14-25  
(consulte também OIDs)  
adicionando a conjunto de dados, 142  
conjuntos de variáveis, 58  
hierarquia, 15  
MIB, extensão em SMIv2, 25-27  
objetos escalares e tabulares, 31  
OIDs  
definindo, 17-25  
nomeando, 15-17  
polling por tipo, 135-136  
tipos de dados  
SMIv1, 17  
SMIv2, 24-25  
vinculações de variáveis, 30, 38  
Octetos de saída (consulte ifOutOctets, objeto)  
Octetos  
informações sobre a MIB-II, 6  
Oetiker, Tobias, 231  
-Of, opção (Net-SNMP), 264  
-Ofn, opção (Net-SNMP), 273  
OIDs (IDs de objetos), 14  
anexando números de instância a, 118  
definindo, 17-25, 185  
em xnmgraph, 141  
erros em pesquisas, 36  
imprimindo com Net-SNMP, 264  
nomeando, 15-17

OIDs da base, mib-2, 28  
procurando em MIBs, 264  
rastreando modificações efetuadas em, 269  
recuperando na Perl, 111, 116  
rotulando em saída, 139-140  
SMIv1, tipo de dado, 18  
SMIv2, extensões, 25-27  
vinculações de variáveis, 30  
OIDs numéricas, 14, 15, 264  
convertendo em texto, 116, 136-137, 273  
(Consulte também OIDs)  
-On, opção (Net-SNMP), 264  
Opção -B (Net-SNMP), 267  
Opção -c (Net-SNMP), 265  
Opção -Cb (Net-SNMP), 271-272  
OpenNMS, coleção, 65  
OpenRiver (RiverSoft), 64  
OpenSSL, 292-293  
OpenView  
GUI baseada na Web no, 51  
xnmgraph, 133-142, 253-256  
OpenView ITO, 63  
OpenView Network Node Manager, 62, 72-84  
ações, 258  
adicionando menus personalizados, 256-259  
carregando MIBs, 84  
dados externos, 253-256  
enviando mensagens com, 260  
estrutura de diretórios, 72  
filtros, 81-84  
iniciando e executando, 73  
intervalos de polling, 78-80  
licença Instant-On, 72  
loadhosts, utilitário, 78  
mapas, 73, 80  
Menus, 258  
netmon, processo demo, 74-78  
perfis do usuário, 259  
scripts, 72  
OpenView  
agente de SNMP Extensível HP, 60  
atualizando gráficos, 147-148  
browser de alarmes, 163-164  
carregando MIBs, 193-194  
configurando, 96-97  
conjunto de dados, 142-148  
demo de manipulação de retenções (traps)  
(ovtrapd), 155  
elaborando coleções, 143-146  
enviando retenções (traps), 172  
estendendo, 179, 189-201  
eventos (consulte Eventos)  
expressões, 246  
interface gráfica versus interface de linha de comando, 136-137  
interfaces de linha de comando, 136-137  
intervalos de polling, 78-80  
limiares, 146-148  
operações de get, 113-115  
operações de set, 120  
recebendo e manipulando retenções (traps), 155-166

script de verificação de espaço de disco, 214-225  
tabelas, 195-201  
Operação em SNMP, 29-41  
(Consulte também os nomes de operações específicas)  
Operações de get, 30, 110  
compatibilidade com o SNMP e, 53, 54  
mensagens de erro, 36, 123  
Net-SNMP, 115, 267  
OpenView, 113-115  
Perl, 110-113, 283  
SNMPv3, 289  
SystemEDGE, requisitos de saída, 186-187  
usando com o comando set, 35  
Operações de get-next, 31  
compatibilidade com o SNMP e, 53  
mensagens de erro, 36  
MRTG, 233  
Net-SNMP, 270-272  
Perl, 284  
SNMPv3, 289  
SystemEDGE, requisitos de saída, 186-187  
Operações de get-response  
compatibilidade com o SNMP e, 53  
get e, 30  
get-next e, 31  
Operações de report, 42, 54  
Operações de set, 34, 110  
atualizando tabelas, 199  
compatibilidade com o SNMP e, 53  
criando arquivos de texto com, 193-194  
definindo e confirmando valores, 120-121  
mensagens de erro, 36, 123  
Net-SNMP, 268-269  
Perl, 285  
SNMPv3, 289  
SystemEDGE, requisitos de saída, 186  
Origem do status (OpenView), 80, 154  
-OS, opção (Net-SNMP), 264  
-Os, opção (Net-SNMP), 264, 266  
OS/390, sistema operacional, 63  
OSPF, polling e, 135-136  
OV\_Node\_Down, evento, 156, 160  
ovdump, arquivo de eventos, 155  
ovfiltercheck, utilitário (OpenView), 83  
ovtrapd, utilitário (OpenView), 155

## P

-p, opção (snmpdelta), 270  
Pacotes  
bloqueando, 91  
codificando, 292  
datagramas em UDP, 10  
informações sobre status do host, 270-271  
Parâmetros  
MRTG, 234, 239, 240  
OpenView xnmgraph, 142  
Perl, 283  
RMON, 129, 131

SNMP, definições, 88-90  
SystemEDGE, 103, 186, 188  
Pares de nome-OID, 281  
Partições, manipulando, 209  
PDUs (Protocol Data Units)  
diferenças nos formatos das retenções (traps), 41  
formatos para operações de SNMP, 29  
Perfis de shell, 204  
Perfis do usuário (NNM), 259  
Perfis  
no nível do sistema, 203  
perfis de shell, 204  
usuários do NMM, 259  
perfmon, utilitário (Windows), 188-189  
Períodos de aprendizagem para polling, 151-152  
Perl, 68  
consultando agente extensível SystemEDGE, 187  
enviando retenções (traps), 173, 202  
identificando versão de, 231  
instruções use, 281  
MIB, rotinas de gerenciamento, 281  
monitorando retenções (traps) com, 166-168  
monitorando usuários e processos, 242-243  
MRTG e, 231  
operações de get, 110-113  
operações de set, 121  
operações de SNMP, 283-286  
parâmetros, 283  
scripts de polling interna, 126  
scripts de status de disco, 213-214  
site da Web, 111  
SNMP, módulo de Perl, 111, 281  
SNMP, suporte para Perl, 69, 165-166  
SNMP\_util, módulo, 281-286  
snmpconf, script, 272  
vmstat, script, 255  
Personalizando  
categorias de eventos, 160  
menus no OpenView NNM, 256-259  
menus no SNMPc, 151  
Pesquisa de acesso aleatório em Net-SNMP, 264  
Pilha de protocolos, 11  
ping, comando  
pesquisando somente dispositivos que respondem, 222  
polling com, 80  
Remote Pinger, 186  
SNMPc, utilização, 85  
PIX, firewalls, 104  
Planejamento de capacidades (MRTG), 230  
Plug-ins, SystemEDGE, 104  
pmd (demo de Postmaster), 155  
Polling externa, 124, 133-152  
OpenView  
conjunto de dados, 142-148  
limiares, 139-140, 146-148  
xnmgraph, 133-142  
RMON, 127  
SNMPc, 148-152  
Polling interna, 124, 126-134  
agentes e, 126

ganchos de programa e, 126  
identificando números de instância, 130-131  
intervalos de polling, 131  
problemas com dados de tendências, 132-133  
RMON, 127-133

**Polling**  
agentes e, 103, 126  
através da Internet, 47  
calculando necessidades de armazenamento de dados, 46  
congestionamento da rede e, 126  
DHCP, opções, 75  
dispositivos não-SNMP, 242-243  
distribuindo, 133-134  
filtros em NNM, 81-84  
ganchos de programa e, 126  
intervalos (consulte Intervalos de polling)  
limiares e, 124, 146-149, 151-152  
MRTG, 231, 232  
Net-SNMP, 115  
números de instância e, 130-131  
objetos sem uso, 138  
OpenView, 113-115  
conjunto de dados, 142-149  
elaborando coleções, 143-146  
grafando, 134-142  
intervalos, 78-80, 139-140  
limitando, 144-145  
netmon, 74-78  
polling.pl, script, 215-216  
operações de get, 30, 110  
orientada por retenções (traps), 51  
perfodos de aprendizagem, 151  
Perl, 110-113  
polling externa, 124, 133-152  
polling interna, 124, 126-133  
por tipo de objeto, 135-136  
portas, 11  
RMON, 127-133  
servidores, 222  
SNMPc, 85, 148-152  
testando eventos, 147  
tráfego e, 81  
Pontos de saturação, 252  
Portas estáticas, monitorando, 225  
**Portas**  
listando números, 225  
monitorando scripts, 225-229  
NNM, definições de porta, 78  
números de instância, 130-131  
portas de gerenciamento em UPSs, 106  
UDP, 11  
Practical Extraction and Report Language (consulte Perl)  
**Processo de determinação**  
OpenView (netmon), 73, 74-78  
arquivos iniciais, 78  
filtros, 81-84  
intervalos de polling, 78-81  
SNMPc, 85  
dispositivos iniciais, 85-86

**Processos**  
gerenciando, 7, 42  
grafando, 239  
monitorando (SystemEDGE), 103  
verificando com Net-SNMP, 180  
**Programas (consulte Aplicativos)**  
Protegendo redes, 90  
**Protocol Data Units (consulte PDUs)**  
**Protocolos sem conexão, 10**

**Q**

**Quadros (consulte Gráficos)**

**R**

RAID, arrays, 209  
Raízes em árvores de objetos, 15  
Ramificações (consulte subárvores)  
**Recarregando**  
arquivos de configuração, 296  
MIBs, 193-194  
**Recebendo retenções (traps), 154**  
Net-SNMP, 169  
OpenView, 155-166  
Perl, 166-168  
SNMPv3, 282  
testando recebimento, 176  
Trap Receiver, 168  
**recebendo, 154**  
Net-SNMP, 169  
OpenView, 155-166  
Perl, 166-168  
SNMPv3, 289  
testando recebimento, 176  
Trap Receiver, 168  
redundante, 222  
retenções (traps) não recebidas, 10, 37-38, 153, 223  
retenções de monitoramento do ambiente (Cisco), 105  
**RMON**  
alarmes e eventos, 128  
aumentando alarmes de limiares, 128, 129, 132-133  
queda de alarmes de limiares, 128, 129, 132-133  
registrando, 129  
retenções unidirecionais, 128  
seqüência de eventos em, 37-38  
SMIv2, extensões para, 26  
snmptrap, argumentos, 169  
SNMPv1 e v2, 154, 175  
SNMPv3, 154, 289, 296  
UDP (consulte UDP)  
vinculações de variáveis, 38, 171, 173  
(Consulte também notificações)  
**Redes, gerenciando (consulte Gerenciando redes)**

- regedit, utilitário (Windows), 92, 188-189  
 Registrando dados de polling (consulte Conjunto de dados)  
 Registrando  
     MRTG, arquivos de log, 234, 238-239  
     Net-SNMP, arquivos de log, 99  
 OpenView  
     eventos, 155, 157, 160  
     retenções (traps), 155, 159  
     variáveis, 160  
 RMON  
     alarmes, 128  
     eventos, 128, 129  
 Registro de desempenho, 188-189  
 Registro do Windows (consulte Registro)  
 Registro, 92, 95, 188-189  
 Reinicializando, informações sobre retenções, 39, 153  
 Remote Network Monitoring (consulte RMON)  
 Remote Pinger, 185-186  
 Requests for Comments (consulte RFCs)  
 Requisitos de RAM para NMSs, 45, 46  
 retenções "no format" (OpenView), 165-166  
 Retenções (traps) assíncronas (consulte Traps (retenções))  
 Retenções (traps) específicas de empresa, 38, 153  
 Retenções (traps), 5, 37-41, 154  
     categorias de, 153  
     compatibilidade com o SNMP e, 53  
     confirmado recebimento, 41, 271-272  
     destinos, configuração de, 37-38
         Cisco, 105  
         Net-SNMP, 102  
         OpenView, 96  
         SystemEDGE, 103  
         UPS, 108-109  
         agentes do Windows, 92, 95  
     enviando mensagens com, 260  
     enviando, 169-178
         dispositivos Cisco, 104  
         ganchos em programas, 176  
         instruindo, 176  
         Net-SNMP, 175, 268, 297  
         OpenView, 172, 202  
         Perl, 173, 202, 286  
         SNMPv3, 289, 297  
         testando geração, 176  
         Trap Generator, 174  
     específicas da empresa, 38-39, 154  
     exibindo com Perl, 167  
     falha de autenticação (consulte Retenções por falha de autenticação)  
     gerando com logins, 203  
     incluindo horas em (Net-SNMP), 175  
     limiares do servidor, 222  
     marcações de hora, 171  
     mecanismo informativo, 40-41, 271-272  
     monitorando aplicativos com, 60  
     números específicos, 171  
     números genéricos, 38, 154  
     números médios de, 160  
     OpenView
         ignorando, 163-166  
         limitando fontes, 157  
         "Log only", opção, 158  
         registrando, 155  
         retenções "no format", 165-166  
     polling orientada por retenções, 51  
     portas, 11  
 Retenções de entrada (consulte Recebendo retenções)  
 Retenções por falha de autenticação, 90
     agente do Windows NT/2000, 95  
     dispositivos Cisco, 105  
     Net-SNMP, 102  
     retenções por falhas de autenticação genérica, 39  
     string incorreta de comunidade, 13  
     SystemEDGE, 103  
 Retenções redundantes, 222  
 Retransmissões, sobrecarga e, 11  
 Retry, definições (NNM), 78  
 RFCs (Requests for Comments)
     experimental, 3, 275-280  
     lista, 275-280  
     padrões históricos, 3  
     processo de, 3  
     SNMPv3, 288  
 RiverSoft OpenRiver, 64  
 RMON (Remote Network Monitoring), 2, 8, 43, 127-133
     alarmes e eventos, 127-133  
     comando de alarme, 131-132  
     configuração, 129-133  
     grupos, 43  
     parâmetros, 129, 131  
     polling externa, 127  
     polling interna, 127  
     problemas com dados de tendências, 131-133  
     registrando eventos, 128  
     RMON, MIB, 8, 43  
     software de análise de tendências e, 67  
     transmissão de retenção unidirecional, 128  
     versões, 8, 44  
     (Consulte também Cisco)  
 Roteadores
     encerrando, 105  
     listando comandos, 104  
     listando interfaces, 118  
     listas de acesso, 91  
     polling, 131-132  
     SNMP, suporte para, 57  
     SNMPv3, configuração de, 291-292  
     testando, 78  
 Rótulos, em gráficos, 138, 140, 141, 240  
 RRDTool, 68

## S

- s, opção (snmpdelta), 270
- Safda
  - gravando em arquivos, 270

- MRTG, 232  
recuperando e usando rótulos de texto em, 137-138  
retornando no Net-SNMP, 182  
rotulando OIDs em, 140  
SystemEDGE, requisitos, 186  
Tabelas, 195-201  
Scripts de shell  
ganchos para, 177  
OpenView, 72  
procurando arquivos, 182  
scripts de polling interna, 126  
SystemEDGE, 186  
verificando tipos de arquivos, 184  
Scripts  
adicionando a menus do OpenView, 256-259  
geração de retenções (traps), 202  
monitoramento de portas, 225-229  
registrando logins, 203  
verificador de espaço de disco, 214-225  
Veritas, script de verificação de disco, 209-213  
Security Subsystem (SNMPv3), 289  
Segurança  
firewalls, 91  
limitando solicitações a agentes, 91, 102  
polling através da Internet, 50  
retenções por falha de autenticação, 90  
scripts de extensões e, 188  
SNMP, desvantagens e, 90, 287  
SNMPv3 e, 91, 287  
codificação, 289, 292  
Net-SNMP, 292-289  
níveis, 290  
roteadores Cisco, 292  
USM, 290  
(Consulte também Acesso)  
strings de comunidades, 13, 90  
sendmail, processo  
monitorando, 103  
verificando e executando processos, 181  
Senhas  
codificando, 292  
configurações padrão do SNMPv3, 296  
criando no SNMPv3, 282, 283  
IDs de mecanismos e, 296  
tabelas em SNMPv3, 290  
(Consulte também Strings de comunidades)  
Serviços de transporte, 27  
Serviços privativos  
chaves, 292  
configuração padrão, 297  
configurando, 293, 294  
SNMPv3, 289, 290, 293  
Servidores de mensagens, verificando status, 225-229  
Servidores proxy, 171  
Servidores Apache da Web, 104, 261  
Servidores  
parâmetros de gravação, 239  
polling, 222  
script de verificação de espaço de disco, 214-222  
servidores da Web, 104, 261  
servidores de mensagens, monitorando, 225-229  
servidores proxy, 171  
Sets em filtros, 82, 83  
SGMP (Simple Gateway Management Protocol), 1  
SHA, algoritmo, 289, 293  
Símbolos lógicos  
AND (&&), 83  
NOT (!), 83  
OR (||), 83  
Simple Gateway Management Protocol (SGMP), 1  
Simple Network Management Protocol (consulte SNMP)  
SimpleWeb, 8  
sinais de menos (-), em fluxos, 254  
Sistemas de arquivos  
identificando objetos úteis, 215  
ignorando em pollings, 222  
listando, 222  
Sistemas distribuídos  
arquitetura NMS, 48  
enviando eventos, 158  
polling, 133-134  
Sistemas operacionais  
agentes em, 5  
NMS, suporte para, 51, 59-71  
(Consulte também os nomes de pacotes de SO específicos)  
SMI (Structure of Management Information), 6  
SMI Next Generation (SMING), 18  
SMIV1, 14-25  
definindo OIDs, 17-25  
nomeando OIDs, 15-17  
RFC, padrões, 275  
tipos de dados, 17  
SMIV2, 14, 25-27  
aprimoramentos de definição de objetos, 27  
convenções para texto, 27  
RFC, padrões, 275  
tipos de dados, 25  
SMTP, suporte em dispositivos, 85  
SMUX (SNMP Multiplexing Protocol), 57  
SNMP (Simple Network Management Protocol), ix, 1  
agentes bilingües, 54  
AgentX, 57  
cenários antes/depois, 2  
comunidades, 13-14  
Deficiências da segurança, 90, 287  
dispositivos compatíveis, 53  
em pilha de protocolos, 11  
FAQs, 9  
gerenciamento de hosts, 7, 42  
gerenciamento de rede, 1  
grupos de gerenciamento (MIB-II), 29  
história, 1  
informações e recursos, 8  
interfaces baseadas na Web, 51  
mensagens de erro, 36  
MIBs (consulte MIBs)  
monitoramento remoto (consulte RMON)

Perl, operações, 283-286  
RFC, padrões, 4, 275-280  
SMI, 6  
    SMIV1, 14-25  
    SMIV2, 25-27  
UDP e, 10-12  
visão geral das operações, 29-41  
(Consulte também SNMPv1; SNMPv2;  
SNMPv3)

SNMP Link, 8

SNMP MIB Browser (OpenView), 113, 118

SNMP Multiplexing Protocol (SMUX), 57

SNMP\_Session, módulo, 111

SNMP\_util, módulo, 111, 281-286

SNMP++, 70

snmpbulkget, comando (consulte snmpgetbulk, comando)

snmpbulkwalk, comando (Net-SNMP), 267

SNMPc, 63  
    configuração, 85-87  
        apresentação dos nós, 85  
        carregando MIBs, 86  
        dispositivos iniciais, 85, 86  
        instalação, 85  
edições de grupo de trabalho (workgroup) e de  
empresa (enterprise), 148

polling externa, 148-152  
    Automatic Alarms, 151  
    grafando, 149-150  
    limiares, 151  
    períodos de aprendizagem, 151  
    personalizando menus, 151

snmpconf, utilitário (Net-SNMP), 101, 116,  
272

snmpd, arquivos de log, 98-99, 189

snmpdelta, comando, 269

snmpdf, comando (Net-SNMP), 270

snmpdpm (OpenView, agente principal), 179,  
189-190

snmpget(), rotina, 283

snmpget, comando, 31, 110  
    confirmando comandos set, 35  
    diagnosticando versão do SNMP com, 54  
    OpenView, operações, 113-115  
    respostas para erros, 122-123  
    Scripts em Perl, 110-113  
        várias OIDs em, 117  
(Consulte também Operações de get)

snmpgetbulk, comando, 34  
    diagnosticando versão do SNMP com, 56  
    Net-SNMP, 267

snmpgetnext(), rotina, 284

snmpgetnext, comando, 270-271  
(consulte também Operações de get-next)

snmpInfo, tabela (SNMPc), 149-150

snmpinform, comando, 272

snmpLoad\_OID\_Cache(), rotina, 282

snmpmapOID(), rotina, 281

snmpMIB\_to\_OID(), rotina, 282

snmpOutPkts, objeto (MIB), 148-149

snmpQueue\_MIB\_File(), rotina, 282

snmpset(), rotina, 285

snmpset, comando, 35, 110  
    atualizando tabelas, 199  
    criando arquivos de texto, 194  
    definindo e confirmado valores, 120-123  
    Net-SNMP, 268  
    respostas para erros, 122-123  
(Consulte também Operações de set)

snmpstatus, comando, 271

snmptable, comando, 272

snmptranslate, utilitário (Net-SNMP), 116,  
272-273

snmptrap, programas, 169  
    ganchos em, 178  
    Net-SNMP, 175, 268  
    OpenView, 172  
        snmptrap(), rotina, 173, 286

snmptrapd, utilitário, 169, 174

snmpusm, utilitário, 272, 294

SNMPv1, 4  
    compatibilidade de dispositivos, 54  
    definição de retenções (traps), 154  
    determinando a compatibilidade, 53  
    geração de retenções (traps) (Net-SNMP), 175  
    mensagens de erro, 36

SNMPv2, 4  
    compatibilidade de dispositivos, 54  
    convenções para texto, 27  
    definição de retenções (traps), 154  
    determinando a compatibilidade, 54  
    geração de retenções (traps) (Net-SNMP), 175  
    mensagens de erro, 36  
    OpenView support, 189-190  
    snmpV2, ramificação, 25

snmpV2, ramificação, 25

SNMPv3, 4  
    aplicativos, 289  
    autenticação (consulte Autenticação, SNMPv3)  
    campos, 296-297  
    clonando usuários, 294  
    configuração de roteador Cisco, 291  
    configuração padrão, 295  
    convenções para texto, 290  
    copiando arquivos de configuração, 293-294  
    criando usuários, 292-293  
    definição de retenções (traps), 154  
    determinando a compatibilidade, 54  
    entidades, 288, 290  
    geração de retenções (traps), 296-297  
    grupos, 292  
    IETF, site do grupo de trabalho, 297-298  
    mechanismo, 288  
    modificações em, 287-291  
    Net-SNMP, configuração, 291-297  
    OpenSSL, 292  
    privacidade, 289, 290, 293  
    RFCs, 288  
    segurança (consulte Segurança)  
    suporte de dispositivo, 56  
    visões, 291

snmpwalk(), rotina, 284

snmpwalk, comando, 31, 110  
    Net-SNMP, 119, 265-268

OpenView, 118, 193-194  
recuperando espaço de disco, 185  
respostas para erros, 122-123  
riscos de sobreregar sistemas, 119  
RMON, tabelas de eventos, 131  
scripts em Perl, 116  
(Consulte também Operações de get-next)  
Software de análise de tendências, 67  
Software instalado, rastreando, 7, 42  
Solaris, sistema operacional,  
ganchos em programas, 178  
NMS, suítes, 62, 63  
OpenView e, 95-97  
software agente, 60, 61, 102  
software de análise de tendências, 67  
software de suporte, 71  
software específico do fornecedor, 66  
software gerenciador de componentes, 66  
Arquivos de som, reproduzindo para  
eventos, 159  
Solicitações do ICMP, 185-186  
Spectrum for Cabletron, 66  
SSL, 91  
Status de disco, verificando com o Veritas Volume  
Manager, 209-214  
Strings de comunidades leitura-gravação, 13, 88,  
89  
Strings de comunidades somente leitura, 13, 88,  
89  
Strings de comunidades, 13  
como definições de parâmetros, 88  
definições de agentes do Windows, 92, 95  
mensagens de erro, 37-38, 122  
modificando, 13, 96  
MRTG, 223, 224  
Net-SNMP, 102, 257  
NNM, 78  
padrões, 13, 55  
personalizações do fornecedor, 90  
questões de segurança, 90  
retenções por falha de autenticação, 39, 90  
RMON, 129  
selecionando, 13, 90  
SNMPc, 86  
SNMPv2 e, 4  
UPSs, 107  
(Consulte também Comunidades)  
Strings vazias (""), 269  
Strings, convertendo inteiros em, 199  
Structure of Management Information (consulte  
SMI)  
Subagentes, 57, 102  
Subárvore de Internet, 15, 25  
Subárvore egp (MIB-II), 29  
Subárvores, 15  
criando, 190  
definindo objeto na, 191  
Internet(1), 15  
MIB-II, 28-29  
ramificações de iso(1).org(3).dod(6).  
Suítes de NMS baseadas em Java, 66

Sun Microsystems, software agente, 51  
Suporte para pacotes de software de NMSs, 69-71  
Suporte para protocolo em dispositivos, 85  
Switches do CoreBuilder, 70  
Symetra, 106-109  
sysContact, parâmetro, 89  
definindo valores, 120  
recuperando com OpenView, 113  
recuperando no Perl, 111  
sysLocation, parâmetro, 88  
configurando no Net-SNMP, 98-99  
definindo valores, 34  
recuperando, 31, 116  
syslog, registros, 47, 169  
sysName, parâmetro, 89  
sysObjectID, parâmetro, 144-145  
sysServices, parâmetro, 94  
SystemEDGE, 61  
configurando, 102-104  
arquivo de configuração, 102, 186-187  
plug-ins, 104  
estendendo, 179, 185-189  
parâmetros, 103, 186, 188  
recursos de monitoramento automático, 103

## T

-t, opção (snmpdelta), 270  
-T, opção  
Net-SNMP, 264  
snmpdelta, 270  
Tabela de interfaces, 275  
Tabelas  
adicionando colunas, 26  
agente do OpenView e, 195-201  
escrevendo definições de MIBs, 196  
índices de inteiros, 196  
atualizando, 199  
espaços internos em, 199  
ifTable, 23, 24  
índices, 24  
linhas em, 24, 27  
snmpitable, 271-272  
Tcl, 65  
TCP  
comparação com UDP, 10  
especificando para comandos (Net-SNMP), 264  
monitorando segmentos transmitidos, 188  
TCP/IP  
informações sobre a MIB-II, 6  
monitorando portas, 225  
suite de protocolos, 11  
-Td, opção (Net-SNMP), 273  
Telnet, comandos, 225  
Testando  
geração de retenções (traps), 176  
limiares e eventos, 147-148  
perfis, 259  
roteadores, 78  
status de diretório, 207

Text User Interface (TUI), 106  
The Simple Times, 8, 275  
Timeout, definições (NNM), 78  
Timeouts, agente, 193-194  
Timestamps (marcações de hora)  
  em retenções (traps), 171, 172  
  Net-SNMP, opções, 269  
  TimeStamp, convenção para texto, 27  
Tipos de dados  
  em vinculações de dados de retenção, 171  
  importando de outras MIBs, 22  
  objetos gerenciados e, 15  
  tipos aceitos  
    Net-SNMP, 122, 175  
    OpenView, 172  
    Perl, 172  
    SMIv1, 17  
    SMIv2, 25  
    SNMP\_util, 173  
    SystemEDGE, 186  
    Trap Generator, 174  
  valores decimais em, 25  
Tipos enumerados  
  mensagens de erro, 36  
  SMIv1, 17  
  SMIv2, 27  
  valores, 199  
Tivoli Netview, 63  
Tkined, coleção, 65  
TNG Framework, 64  
-Tp, opção (Net-SNMP), 274  
Tráfego de entrada (consulte ifInOctets, objeto)  
Tráfego  
  definindo limiares para, 124  
  em linhas, medindo (If%util), 246-252  
  filtros e, 81  
  grafando com MRTG, 237  
  polling interna e, 126  
  Pontos de saturação, 252  
Transmission Control Protocol (consulte TCP)  
Trap Generator (Network Computing Technologies), 174  
Trap Receiver (Network Computing Technologies), 168  
Trinagy TREND, 68  
TUI (Text User Interface), 106

## U

UCD-SNMP, projeto (consulte Net-SNMP)  
UDP (continuação)  
  especificando para comandos, 264  
  retenções (traps) e falta de confiabilidade em, 10, 37, 153, 223  
UDP (User Datagram Protocol)  
  baixa sobrecarga, 11  
  em pilha de protocolos, 11  
  limitando tráfego, 13  
  portas 161 e 162, 11  
  SNMP e, 10-12

Unicenter TNG Framework, 63  
Unix, 102  
  agentes, 61  
  MRTG, 235  
  NMS, suítes, 62, 65  
  registrando logins, 203  
  software de análise de tendências, 69  
  software de suporte, 69, 70  
  software específico do fornecedor, 66  
  SystemEDGE, possibilidade de extensão para, 183-186  
  vmstat, utilitário, 254  
UPSs (uninterruptible power supplies), 57, 106-109  
Usenet, 8, 244  
User Datagram Protocol (consulte UDP)  
User-based Security Model (USM), 272, 290  
USM (User-based Security Model), 272, 290  
Usuários  
  grafando número registrado em (MRTG), 239  
  gerenciando, 42  
  NNM, perfis, 259  
  SNMPv3  
    clonando, 295  
    configuração padrão, 295-296  
    criando, 292, 293  
    IDs de mecanismos e dados, 295-296  
    mantendo com snmpusm, 295  
    usmUser, tabela, 291

## V

-v, opção (Net-SNMP), 265  
Valores delta  
  calculando, 18  
  em limiares, 131-132  
varbinds (consulte Vinculações de variáveis)  
Variáveis de ambiente no OpenView, 72, 259  
Variáveis  
  ambiente, no OpenView, 72, 259  
  conjuntos de variáveis gerenciados, 58  
  definições do fornecedor em MIBs, 7  
  mensagens de erro, 37  
Vários arquivos de menus, 257  
Velocidade de linhas, 246  
Veritas NerveCenter, 64  
Veritas Volume Manager, 209-213  
Vinculação de dados em retenções (traps), 171  
Vinculações de variáveis, 30  
  em retenções (traps) específicas de empresa, 154  
  em retenções (traps), 38, 171, 173  
  exibindo em saída de retenções, 167  
  fórmulas para solicitações de get em massa, 33  
Virtual Private Networks (VPNs), 14, 91  
Visões no SNMPv3, 291  
vmstat, utilitário, 254  
VPNs (Virtual Private Networks), 14, 91  
vxprint, utilitário (Veritas), 209-214  
vxvm, utilitário (Veritas), 209-214

## **W**

WILMA, 70  
Windows 95/98 Resource Kit, 92  
Windows 9x/NT/2000, sistemas operacionais  
agentes, 61  
NMS, suítes, 62, 63  
software de análise de tendências, 67, 68  
software de suporte, 69, 70  
software específico do fornecedor, 66  
SystemEDGE e, 185-189  
Windows System Policy Editor, 92

## **X**

xnmrowser (OpenView), 79, 113-115, 133-134  
xnmevents, utilitário (OpenView), 155, 162-163  
xnmgraph, utilitário (OpenView), 133-142  
atualizando gráficos, 147-148  
grafando conjuntos de dados, 147-148  
grafando dados externos, 253-256  
parâmetros, 140-141  
plotando medições de largura de banda, 247  
xnmloadmib, utilitário (OpenView), 84, 194, 199,  
257  
xnmtrap, utilitário (OpenView), 155

## **Z**

Zlib, biblioteca, 231