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

Sistema de Gerência de Energia para Redes Locais

por

LUIS FERNANDO POLLO

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Ciência da Computação

Profª Drª Ingrid Eleonora Schreiber Jansch-Pôrto
Orientadora

Porto Alegre, setembro de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Pollo, Luis Fernando

Sistema de Gerência de Energia para Redes Locais / por Luis Fernando Pollo. Porto Alegre: PPGC da UFRGS, 2002.
144 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2002. Orientadora: Jansch-Pôrto, Ingrid Eleonora Schreiber.

1. Gerência de energia. 2. Gerência de redes. 3. Eficiência energética. 4. SNMP. 5. Java. 6. Tolerância a falhas. I. Jansch-Pôrto, Ingrid Eleonora Schreiber. II. Título.

Em memória de meu avô, Orlando Polo, cuja lembrança me traz a certeza de que não é preciso ter tudo ou saber tudo para ser feliz.

"Ti manco, nonno."

Agradecimentos

Os últimos dois anos e meio foram um período tumultuado da minha vida. Descobrir que se está com câncer aos 23 anos de idade não é o que a maioria dos jovens alunos recém-formados e cheios de expectativas planeja nessa fase. Sob muitos aspectos, esse foi, sem dúvida, o período mais difícil que já vivi. Sob outros, foi também o mais especial. As pessoas que estiveram à minha volta durante esse tempo sabem do que estou falando. Foram muitos sacrifícios, muito sofrimento. Não só meus, entretanto. Pelo contrário, tenho certeza de que foi o sofrimento calado dessas mesmas pessoas, transformado em mais e mais força (uma força aparentemente infinita!) e convertido em muito amor e carinho, que me fizeram vencer.

Por uma dessas grandes coincidências (ou não) da vida, esse período coincidiu com a empreitada em busca do tão idealizado título de Mestre. Embora minha perspectiva diante das coisas seja diferente hoje em dia, e minhas prioridades já não sejam exatamente as mesmas de antes, esse foi, sim, mais um grande desafio. Não se deixem enganar pelo drama da pequena história que acabo de contar. Um mestrado dá muito trabalho! E, sendo assim, considero também uma grande conquista pessoal chegar ao fim de mais esta etapa. Gostaria de aproveitar este espaço para agradecer de maneira singela a algumas pessoas muito especiais.

Aos meus pais, Dari e Ivete, um demorado abraço do filho que não saberia descrever com palavras o sentimento de gratidão, respeito e amor que tem por vocês. Vocês sabem que os meus próprios filhos são ainda um sonho distante, mas eu seria uma pessoa realizada se conseguisse dar a eles um lar tão cheio de amor e carinho quanto o que vocês conseguiram nos dar. Obrigado pela dedicação incondicional, pela preocupação em nos dar todo o conforto possível, e por todas as oportunidades que, mesmo tão cedo, pudemos ter – tudo por causa da infinita generosidade de vocês. Obrigado pelas noites insones ao meu lado no hospital, pelas palavras de consolo, pela paciência interminável com os meus destemperos. Obrigado, obrigado, obrigado. Amo muito vocês!

Às minhas duas irmãs queridas, Vanessa e Alessandra, um beijo enorme! Nem preciso dizer que vocês são, e serão sempre, os meus "chuchus". Mas são também duas jovens mulheres lindas e talentosas. Não se preocupem demais com o sucesso, ele está escrito nas estrelas para cada uma de vocês. Sejam felizes, isso sim! Amem muito! E aproveitem cada segundo dessa jornada maravilhosa e inexplicável que é a vida. Obrigado pelo colo, pelos cafunés, pelos beijos e abraços. Amo vocês!!!

Ao restante de minha família, quase toda distante, um muito obrigado do fundo do coração. Perdi a conta de quantas foram as viagens para me ver. E de quantas tias se converteram em mães para dar um descanso à original, que não queria arredar pé de perto de mim. E de todos os telefonemas e e-mails dos tios, primos e avós, que não me deixaram esmorecer nas horas mais difíceis. Obrigado por terem dado um sentido tão especial à palavra *família*. Amo todos e cada um de vocês de maneira muito especial.

A Ingrid, por ter sido muito mais do que uma orientadora. É claro que teu acompanhamento técnico durante o mestrado foi decisivo para o meu sucesso. Mas quero que saibas que tua presença foi também marcante como amiga e conselheira. Obrigado pelos longos e-mails enquanto eu estava no hospital, pela tua preocupação em ajudar em tudo o que era possível. Enfim, pela tua dedicação incondicional. Acho que tivemos a sorte de sermos um par de perfeccionistas! O teu empenho em propiciar todas as condições para a realização do meu trabalho foi imprescindível para chegarmos até aqui. Muito obrigado por tudo!

Aos amigos, a maioria dos quais fisicamente distantes, mas sempre presentes em espírito, deixo um abraço carinhoso e o meu muito obrigado. Aos que conheci em Atlanta, durante meu estágio pela AIESEC, e que provavelmente nunca lerão essa dedicatória, obrigado

por muitos momentos inesquecíveis, e por me ajudarem a provar que um guri de Giruá pode, sim, ganhar o mundo. Aleš, Caroline, Tariq, Hai-Mien, Montse, Gursel, Cida, Igor, Boris, Melissa, Juan, Vidula e todos que eu possa ter esquecido: um abraço do tamanho do mundo! Aos que conheci aqui no Brasil, muitos dos quais há muitos e muitos anos, obrigado pela amizade duradoura, que resiste ao tempo e às diferenças. Cesar e Alisson, vocês continuam sendo os outros dois mosqueteiros. Espero que a vida nos permita voltar ao convívio freqüente. Silvia Bertagnolli, Jeferson, Lica, Marcia Pasin e todo o pessoal de Santa Maria: um abração! Roberta Ritter, Dani Kaercher e Tiago, os santo-angelenses mais próximos nos últimos anos: obrigado pela companhia, pelas conversas, pela amizade. Vocês são especiais. Para Tatiana Schabbach e Iandra Santana, duas novas amigas, um grande beijo. A Luci e toda sua família, companhias constantes no primeiro ano de mestrado, um beijo.

Aos médicos, enfermeiras e funcionários do Hospital Santa Rita, responsáveis por eu estar aqui, hoje (e cada vez mais cheio de saúde), a minha eterna gratidão. Em especial ao Dr. Sérgio Roithmann e sua equipe, e às enfermeiras Eucilene e Juliana, muito obrigado, do fundo do coração! Não haveria palavras suficientes para agradecer-lhes, portanto não tentarei. Deixo apenas um abraço carinhoso, e a promessa de que estou fazendo o melhor possível para ter merecido esta segunda chance. Ao Dr. Hugo Schunemann e à equipe do Instituto Kaplan, que aceitaram o desafio original, muito obrigado. A todos que atenderam ao chamado para doações de sangue durante a minha última internação, muito obrigado! Vocês já pensaram que carrego comigo um pouquinho de vida "emprestada" de cada um de vocês? Isso não se paga! Muito, muito obrigado!

Aos colegas, professores, funcionários e direção do Instituto de Informática da UFRGS, obrigado por tudo! Professores João Netto, Weber, Juergen e Lisandro: obrigado pela ajuda, pelas dúvidas esclarecidas e pelas novas dúvidas introduzidas (que, afinal, tornam-se apenas novos desafios). Luis Otavio, obrigado pelo teu empenho e dedicação fora do comum, tua preocupação constante em nos dar as melhores condições possíveis de trabalho nos laboratórios do Instituto. Eliane, teu sorriso dando boas-vindas, à entrada do prédio dos laboratórios, melhora o humor de qualquer pessoa! Beatriz, Ida e demais funcionários da biblioteca: nossa biblioteca é um tesouro à parte; obrigado por fazerem ela assim. Aos colegas, antigos e novos, obrigado pelo sorriso amigo, pelo incentivo, pelos cafés e almoços, pelos bate-papos. Foi um prazer! Um agradecimento em particular ao Tórgan, pela companhia no laboratório 122 e por toda a ajuda durante a minha interminável série de testes e experimentos.

À CP Eletrônica S.A., pelo projeto de convênio que mantém com o Instituto de Informática e que vem possibilitando as pesquisas na área de gerência de energia pelo Grupo de Tolerância a Falhas. Um agradecimento especial a Gerson Gabiatti e Everton Polina, que se dispuseram a ajudar com a realização de testes dentro da empresa.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, pela bolsa de estudos concedida durante os dois primeiros anos do curso, e à Fundação de Apoio da Universidade Federal do Rio Grande do Sul, FAURGS, pela bolsa concedida nos últimos meses.

A Renata, pelas lembranças que mantêm vivos dentro de mim o desejo e a esperança de encontrar um (outro) grande amor. Paulo Coelho diz que "quem conhece a felicidade não consegue mais aceitar humildemente a tristeza." Obrigado por ter me mostrado o que é ser feliz.

Sumário

Lista de Siglas e Abreviaturas

ACPI	Advanced Configuration and Power Interface
Ah	Ampère-hora
AML	ACPI Machine Language
APM	Advanced Power Management
ARP	Address Resolution Protocol
BIOS	Basic Input/Output System
CPU	Central Processing Unit
CSMA/CD	Carrier-Sense Multiple Access with Collision Detection
DOE	Department of Energy
DHCP	Dynamic Host Configuration Protocol
DPMS	Display Power Management Signaling
EPA	Environmental Protection Agency
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
kVA	Kilo Volt-Ampère
KWh	Kilo Watt-hora
LAN	Local Area Network
LBNL	Lawrence Berkeley National Laboratory
OSPM	Operating System-directed Power Management
PC	Personal Computer
PM	Power Management
PROCEL	Programa Nacional de Conservação de Energia Elétrica
RFC	Request for Comments
RMI	Remote Method Invocation
RWU	Remote Wake-up
SMTP	Simple Mail Transport Protocol
SNMPv1	SNMP version 1
SNMPv3	SNMP version 3
SO	Sistema Operacional
SSH	Secure Shell

SSL	Secure Sockets Layer
TCP	Transport Control Protocol
TWh	Tera Watt-hora
UDP	User Datagram Protocol
UPS	Uninterruptible Power Supply
WOL	Wake-on-LAN
XML	Extended Markup Language

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho apresenta a proposta e a implementação de um sistema de gerência de energia para redes locais de computadores (*Local Area Networks* ou LANs). Desde sua introdução, no início dos anos 90, os mecanismos de gerência de energia para computadores têm contribuído de maneira significativa para a redução do consumo nos períodos de inatividade, mas podem ter seu efeito minimizado por uma série de fatores, dentre os quais destaca-se a conexão do equipamento a uma rede. Em linhas gerais, o objetivo do sistema proposto é, justamente, facilitar a gerência de energia em ambientes de rede.

O funcionamento do sistema é baseado na aplicação de políticas de consumo definidas pelo administrador da rede. As políticas podem ser aplicadas em duas situações distintas: em horários pré-determinados (p. ex. depois do horário comercial), quando podem ajudar a reduzir o desperdício de energia, ou em resposta a alterações no fornecimento de energia, caso a rede seja protegida por *no-breaks*, quando a redução no consumo resulta em maior tempo de autonomia da fonte reserva (banco de baterias). As políticas são configuradas através de um mecanismo flexível, que permite não apenas desligar os equipamentos, mas colocá-los em estados intermediários de consumo e executar outros tipos de ações.

A arquitetura do sistema é baseada no modelo SNMP (*Simple Network Management Protocol*) de gerência de redes. É composta, basicamente, de *agentes*, elementos de *software* que residem nos equipamentos da rede e detêm o conhecimento específico sobre suas características de consumo de eletricidade, e de um *gerente*, elemento central que contém a configuração das políticas de consumo e que é responsável pelo monitoramento e controle dos agentes. Gerente e agentes comunicam-se através do protocolo SNMP, trocando informações segundo uma *base de informações* (MIB) projetada especificamente para a gerência de energia.

A ênfase da parte prática do trabalho está no gerente, que foi inteiramente implementado através da linguagem Java, utilizando bibliotecas disponíveis gratuitamente. Adicionalmente, foi implementado um agente-protótipo para a plataforma Windows, o que permitiu observar o sistema completo em execução. Os testes permitiram validar a praticabilidade da arquitetura e estimar o ganho potencial proporcionado pela utilização do sistema. São apresentadas medições que demonstram um aumento de até 6 vezes na autonomia do banco de baterias do *no-break* para uma configuração de rede pequena, utilizando o sistema para desligar automaticamente 90% dos computadores durante um corte no fornecimento externo. A economia decorrente da redução de consumo em horários de inatividade foi estimada em até R$0,63 por computador no período de um ano (tomando por base a tarifa média praticada no Brasil entre janeiro e maio de 2002).

O trabalho contribui, ainda, com uma breve análise qualitativa do suporte à gerência de energia nos sistemas operacionais Linux e Microsoft Windows.

Palavras-chave: gerência de energia, eficiência energética, SNMP, Java, ACPI, APM.

TITLE: “POWER MANAGEMENT SYSTEM FOR LOCAL AREA NETWORKS”

Abstract

This work presents the proposal and implementation of a power management system for local area networks (LANs). Since their introduction, in the early 1990s, power management mechanisms available in computer hardware have contributed to a significant decrease in consumption during periods of inactivity. However, that positive effect is often minimized as a result of several factors, particularly the devices being connected to data networks. The general goal of the system we propose is exactly to facilitate power management in network environments.

The proposed system operates by executing power policies defined by the network administrator, in two different situations: at predetermined times (e.g. after business-hours), which can help reduce energy waste; or in response to changes in the energy supply, in a network protected by UPS (Uninterruptible Power Supply) devices, in which case reduced consumption results in greater autonomy of the backup power source (batteries). The power policies are configured in a flexible manner, which allows not only equipment shutdown, but also the use of intermediate low-power states and the execution of other types of actions.

The underlying architecture is based on the SNMP (Simple Network Management Protocol) model, with *agents* residing on every manageable electronic device of the network and a *manager* that oversees their operation and determines when to apply changes in their power consumption, according to the predefined policies mentioned above. The agents hold the specific knowledge about the power consumption characteristics of the devices under their control, and communicate with the manager through SNMP, exchanging information as defined in a *management information base* (MIB) designed specifically for power management.

The main practical contribution of this work is the implementation of the manager, which was done entirely in Java, using freely available libraries. A prototype agent for the Microsoft Windows platform was also implemented, which allowed us to observe the entire system in operation. A series of tests allowed us to validate the feasibility of the architecture and to estimate the potential economy that may result from effectively using the system. We have measured UPS autonomy during utility power outages and found that it can increase by a factor of 7 in a small network setup, by using the management system to automatically power off 90% of the computers. Potential economy resulting from the decrease in consumption during non-business hours alone is estimated to be as high as R$0,63 (US$26.85 at the time of this writing) per computer during the period of a year.

This work also contributes with a brief analysis of the power management support available in the Linux and Microsoft Windows operating systems.

Keywords: power management, energy efficiency, SNMP, Java, ACPI, APM.

1 Introdução

1.1 Breve panorama do consumo de energia elétrica

A evolução tecnológica dos últimos anos tem superado todas as projeções feitas há algumas décadas. As mudanças foram tão significativas que, hoje, há o que se chama de "Nova Economia", centrada nas indústrias de alta tecnologia. O grande veículo para essas transformações foi a Internet, que alterou profunda e definitivamente a maneira como as informações são distribuídas e acessadas, e também o modo como os negócios são conduzidos.

Essa evolução, entretanto, introduziu uma série de novos problemas. Uma das conseqüências da explosão no uso de equipamentos de informática foi o aumento do consumo de eletricidade. Segundo o Departamento de Energia dos Estados Unidos, os equipamentos de escritório[1] são a fonte de consumo cuja demanda mais cresce atualmente [DOE2000].

Para atender à demanda crescente de energia elétrica, há duas soluções. A mais óbvia é aumentar a produção, o que agravaria o impacto ambiental que já é difícil de controlar nas condições atuais. A segunda alternativa é promover um uso mais eficiente da energia, de maneira que a demanda possa ser atendida sem ampliar a oferta na mesma proporção. Isso significa, principalmente, otimizar o desempenho energético de aparelhos elétricos e instalações, e conscientizar os consumidores quanto ao uso racional da eletricidade. No Brasil, o PROCEL (Programa Nacional de Conservação de Energia Elétrica), criado em 1985 pelos Ministérios de Minas e Energia e da Indústria e Comércio, supervisiona os esforços na área de conservação de energia.

Além do impacto ambiental, existe uma outra razão (talvez menos nobre mas bastante persuasiva) para a preocupação crescente com a quantidade de energia consumida por equipamentos de informática: o custo. Em pesquisa realizada pela Universidade da Califórnia, o consumo de energia elétrica por equipamentos de escritório e de rede nos Estados Unidos, no ano de 1999, foi estimado em 74 TWh, o que corresponde a cerca de 2% do consumo total[2] [KAW2001]. No Brasil, não se tem notícia de um estudo semelhante. No entanto, usando como parâmetro de comparação a proporção entre o número de computadores pessoais nos dois países (cerca de 13 computadores nos EUA para cada computador no Brasil[3]), pode-se fazer uma estimativa que, apesar de bastante grosseira, é suficiente para se ter uma idéia do consumo relativo, aqui. Os dados do último Balanço Energético Nacional, coletados pelo Ministério das Minas e Energia, indicam que o total de eletricidade consumido no país em 1999 foi de 314,7 TWh [MME2000]. Usando a proporção mencionada acima (de 1/13), pode-se estimar que a faixa correspondente ao consumo de eletricidade por equipamentos de escritório no Brasil seja de cerca de 5,6 TWh, ou 1,8% do total – um custo de mais de 500 milhões de reais[4].

Apesar das conseqüências visíveis do aumento da demanda por eletricidade, a maior parte da população não toma conhecimento dos problemas gerados pelo uso indiscriminado de aparelhos elétricos. No ano de 2001, contudo, a importância do controle do consumo de energia foi posta em evidência no Brasil em função da crise de energia e do racionamento. Apesar de ser

---

[1] Computadores, impressoras, copiadoras, aparelhos de fax, etc.

[2] Para uma projeção alternativa, ver Huber & Mills [HUB99]. Para uma análise comparativa dos dois trabalhos, ver Hayes [HAY2001].

[3] Estimados em 110 milhões de unidades nos EUA [KAW2001] e 8,3 milhões no Brasil [CAM99].

[4] Usando a tarifa média praticada no Brasil em 1999, de R$95,95 por MWh [ANE2002]. Se fosse considerada a média apenas entre as tarifas comercial e residencial (R$130,40), a estimativa ultrapassaria os 700 milhões de reais.

uma medida drástica e paliativa (quando o ideal seria que medidas preventivas houvessem sido tomadas com antecedência), o racionamento serviu para uma reeducação dos consumidores, que foram obrigados a rever seus hábitos para atingirem as metas de consumo estabelecidas pelo governo. Isso é importante porque o comportamento dos consumidores é um fator determinante para evitar o desperdício de energia, principalmente no que diz respeito ao uso de equipamentos eletrônicos.

É o comportamento do usuário que determina quanto tempo por dia um equipamento é utilizado, quanto tempo permanece ligado mas ocioso, e também quanto tempo permanece desligado. No caso dos computadores, esse comportamento é influenciado por uma série de fatores, tais como o tipo de *hardware* e sistema operacional do equipamento, tipos de *software* que executa, se ele está ou não conectado à rede, entre outros. Um outro estudo da Universidade da Califórnia, baseado em pesquisas de campo realizadas em duas grandes cidades americanas, demonstrou que apenas 44% dos computadores, 32% dos monitores e 25% das impressoras são desligados à noite [WEB2001]. Esta estatística torna-se particularmente alarmante quando se considera que este tipo de equipamento é efetivamente utilizado durante cerca de 25% da semana (tomando por base horário comercial das 9 às 17 horas).

Outro aspecto importante do comportamento do usuário é o padrão de utilização dos mecanismos de gerência de energia[1] disponíveis nos equipamentos. Praticamente todos os computadores produzidos atualmente possuem algum tipo de recurso de gerência de energia que permite colocá-los em estados de consumo reduzido. Outros equipamentos de escritório, particularmente impressoras e copiadoras, também possuem modos de baixo consumo. O gráfico da Figura 1.1, a seguir, mostra o uso estimado de eletricidade nos Estados Unidos por diversos tipos de equipamentos de escritório em função de diferentes níveis de gerência de energia.

Como se pode ver no gráfico, diferentes suposições sobre níveis de gerência de energia e taxas de desligamento do equipamento têm um impacto bastante significativo nas estimativas de consumo. Supondo que todos os equipamentos tivessem mecanismos de gerência de energia habilitados e operacionais, por exemplo, uma economia adicional de 23% (17 TWh/ano), em relação ao consumo total estimado, poderia ser obtida. A coluna A do gráfico ilustra o pior caso estimado, com a gerência de energia completamente desabilitada. A coluna B indica que a taxa estimada de gerência de energia para o caso real[2] (considerando os dados de 1999) contribuiu para uma economia de 23 TWh/ano. As colunas C e D ilustram, respectivamente, estimativas para os casos de completa saturação de gerência de energia habilitada e operacional, e de completa saturação de gerência de energia e de desligamento noturno dos equipamentos.

[1] O termo "gerência de energia", ou "gerenciamento de energia", é usado em diversos contextos. Os mais comuns são em referência ao controle de consumo de energia em computadores, ao monitoramento de *no-breaks* e ao controle de centrais elétricas. Neste trabalho, o termo é usado no sentido de controle de consumo de energia. Outros usos foram substituídos por termos alternativos.

[2] O estudo não apresenta o percentual estimado médio de gerência de energia habilitada e operacional para o caso real, tampouco o percentual de desligamento noturno. Algumas estimativas individuais presentes no trabalho são: 25% de habilitação de gerência de energia em computadores *desktop* e servidores, 60% em monitores e terminais, 80% em impressoras *laser* e 34% em copiadoras.

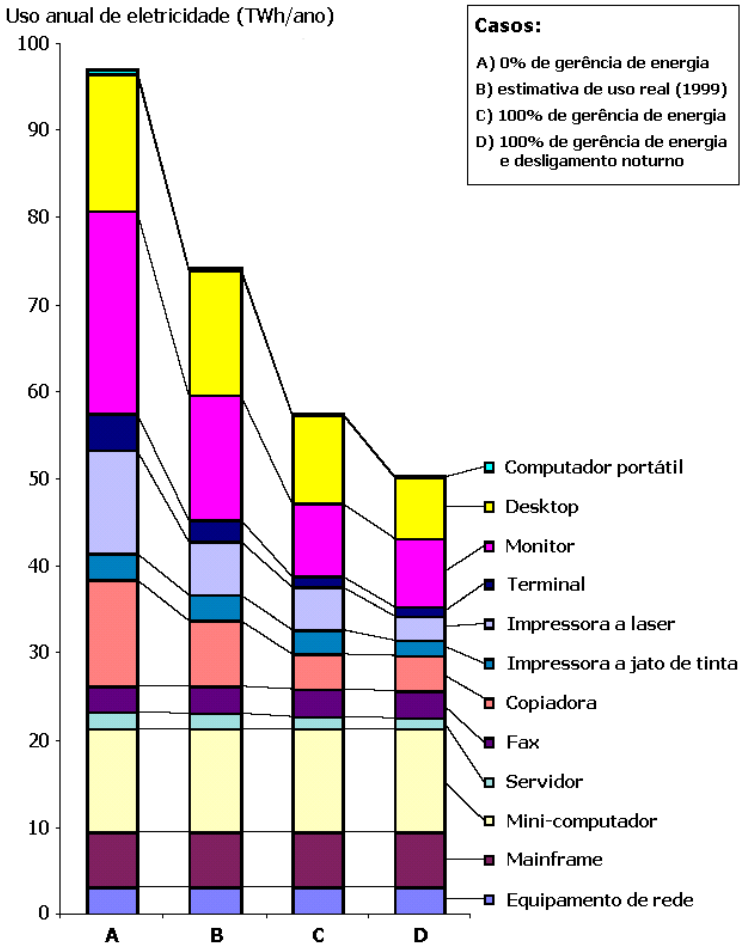


Figura 1.1 - Uso de eletricidade em função de níveis de gerência de energia nos EUA [KAW2001]

1.2 Gerência de energia ("power management")

A primeira iniciativa no sentido de estimular o uso de mecanismos de controle do consumo de energia elétrica foi tomada pela Agência de Proteção Ambiental americana (EPA), com o lançamento do programa Energy Star, em 1992 [EPA2002]. Este programa especifica pelo menos um modo de consumo reduzido de energia (*sleep mode*) para computadores, monitores, impressoras e diversos outros equipamentos, estabelecendo um limite máximo de consumo para cada modo e um limite de ociosidade que determina a troca automática para o modo imediatamente inferior. Os equipamentos que atendem às especificações do programa podem exibir um selo (Figura 1.2), que permite ao consumidor identificar os produtos que fazem uso mais eficiente da eletricidade.

Figura 1.2 - Logotipo do programa Energy Star

A Tabela 1.1 mostra estimativas[1] para os gastos com eletricidade de alguns equipamentos e a economia alcançada com a redução de consumo especificada pelo Energy Star (valores em dólares americanos).

Tabela 1.1 - Economia proporcionada por equipamentos Energy Star

Equipamento	Custo Anual de Eletricidade sem Energy Star	Custo Anual de Eletricidade com Energy Star	Valor Economizado
Computador/Monitor	36,94	16,40	20,54
Fax	22,35	10,11	12,24
Impressora	23,19	8,75	14,44
Scanner	27,11	8,53	18,58
Total	**109,59**	**43,79**	**65,80**

Fonte: Agência de Proteção Ambiental dos Estados Unidos (EPA)

Basicamente, o acordo que os fabricantes assinam com a EPA para exibirem o logotipo em seus produtos especifica *quanta* eletricidade os equipamentos devem economizar nos modos de consumo reduzido, mas não *como* essa economia pode ser alcançada. Foram criadas então especificações com esta finalidade em particular. Duas delas são consideradas padrões na indústria de PCs: APM (*Advanced Power Management*) [INT96] e ACPI (*Advanced Configuration and Power Interface*) [COM2000]. Estes padrões especificam uma série de modos de operação nos quais um computador pode ser colocado para reduzir o consumo de eletricidade. A estratégia usada para decidir quando o computador deve ser colocado num modo de baixo consumo é baseada na detecção da inatividade do sistema, embora o usuário também possa disparar as trocas manualmente. De qualquer maneira, depende do usuário configurar o que se chama de *políticas de consumo*, diretrizes que definem os limites de tempo utilizados para assumir quando o equipamento está ocioso e qual o estado de consumo em que ele deve ser colocado.

1.3 Cenários de uso de gerência centralizada de energia

Como se pode deduzir, a necessidade de intervenção do usuário para a configuração e habilitação desses mecanismos de controle pode tornar-se um empecilho à gerência de energia em ambientes corporativos onde o número de máquinas é consideravelmente grande. Neste tipo de ambiente, pode ser impraticável configurar cada equipamento localmente. Além disso, não raro a própria corporação estabelece uma política de não desligamento dos equipamentos à noite, ou mesmo de não habilitação da gerência de energia, procurando evitar complicações durante procedimentos em que os computadores precisam estar acessíveis através da rede (*backups*, por exemplo).

Uma maneira de contornar esses problemas é centralizar as operações que normalmente seriam executadas em cada máquina através de um mecanismo remoto de gerência. Deste modo, não só o administrador da rede teria condições de configurar políticas de consumo para todos os

[1] Os cálculos assumem um dia de trabalho com 9,5 horas de duração, com um período de inatividade de 5,5 horas por dia (cerca de 58% do tempo total), durante um ano (250 dias úteis), e um custo de 8 centavos de dólar por KWh. Nos EUA, é comum o uso do termo *"9-to-5"* para se referir ao horário comercial típico, embora, para efeito de cálculo, normalmente se utilize um número maior do que 8 horas (que representariam o intervalo exato), como acontece nestas estimativas da EPA.

equipamentos a partir de um ponto único, como poderia fazê-lo de maneira a não interferir com o andamento de outros processos da rede.

Um outro cenário no qual a economia de energia pode desempenhar um papel importante é o de redes cujos equipamentos são abastecidos através de um *no-break*[1]. A tarefa básica do *no-break* é, usando o banco de baterias associado, proporcionar a continuidade no abastecimento de energia para os equipamentos, de modo que eles não sejam abruptamente desligados em situações de interrupção ou falha no fornecimento externo. Neste caso, pode ser extremamente útil, senão essencial, estender o tempo de "sobrevida" (tempo de abastecimento proporcionado pelas baterias). Um dos motivos é permitir o desligamento correto e ordenado de todas as máquinas abastecidas pelo banco de baterias do *no-break* caso a falha no fornecimento externo se prolongue. Entretanto, para que esse processo seja automatizado, também é preciso um elemento central, capaz de monitorar as condições do fornecimento de eletricidade e disparar, remotamente, o desligamento dos equipamentos ou sua entrada em modos de baixo consumo.

1.4 Objetivos e visão geral do trabalho

O objetivo geral do sistema proposto neste trabalho é proporcionar a redução do consumo de energia em redes de computadores. Especificamente, espera-se reduzir o consumo de eletricidade nas duas situações descritas na seção anterior: durante a operação da rede em bateria, quando abastecida através de um *no-break*, e durante os períodos de inatividade dos equipamentos. No primeiro caso, as medidas de economia têm caráter emergencial, com o propósito de estender o tempo de autonomia das baterias do *no-break* e permitir o desligamento correto dos equipamentos em caso de falha prolongada. No segundo, a intenção é racionalizar o uso da energia e reduzir custos.

Para alcançar este objetivo, propõe-se centralizar o controle do consumo de energia dos equipamentos da rede numa *estação de gerência*. Em cada um dos equipamentos gerenciados, um pequeno programa, chamado de *agente*, é encarregado de realizar a comunicação com a estação de gerência para atender às requisições de troca de estado de consumo e também para notificá-la de trocas disparadas no próprio equipamento. Essa arquitetura deriva, basicamente, do modelo mais comumente usado para a gerência de redes TCP/IP, baseado no protocolo SNMP (*Simple Network Management Protocol*). A Figura 1.3, a seguir, mostra uma visão geral da arquitetura do sistema.

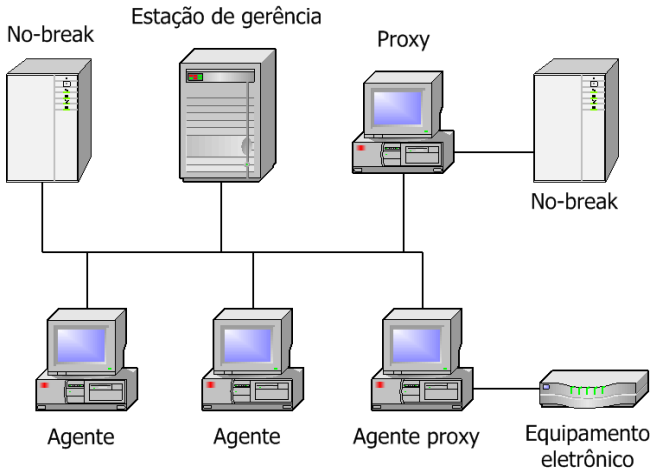


Figura 1.3 - Visão geral do sistema de gerência de energia

[1] *No-break* é como são chamados coloquialmente aparelhos de UPS (*Uninterruptible Power Supply*)

Como se pode ver na figura, tanto os *no-breaks* quanto os equipamentos gerenciados podem se comunicar com a estação de gerência através de um elemento intermediário (*proxy*). Isso permite que equipamentos que não disponham da infra-estrutura de *hardware* ou *software* necessária para a comunicação direta com o gerente[1] também sejam monitorados e controlados.

Na estação de gerência, é possível configurar políticas de consumo para os equipamentos da rede. Por exemplo, pode-se estabelecer que todos os monitores deverão ser automaticamente desligados depois de um certo horário, à noite. Além disso, pode-se configurar o desligamento ou a redução do consumo dos equipamentos em função da ocorrência de eventos relativos ao fornecimento de energia (por exemplo, desligar todos os equipamentos não essenciais em caso de interrupção no fornecimento, e também os servidores caso a carga das baterias esteja prestes a acabar).

As palavras de ordem durante a concepção do sistema foram simplicidade e flexibilidade. Manter o projeto simples (mas não incompleto) e definir as funções do sistema de maneira objetiva permitem uma implementação relativamente fácil. A flexibilidade se traduz na capacidade de gerenciar diferentes tipos de equipamentos, independente de *hardware*, *software* ou sistema específico de gerência de energia com o qual são compatíveis. Também o monitoramento de *no-breaks* é estruturado de maneira a permitir a operação independente de particularidades de cada modelo ou fabricante.

Finalmente, é preciso dizer que a implementação apresentada neste trabalho concentra-se na aplicação de gerência, parte mais complexa do sistema. Os agentes, contudo, são especificados detalhadamente e exemplificados através da implementação de um protótipo.

1.5 Trabalhos anteriores

Não se tem conhecimento de outros sistemas que tenham o objetivo de centralizar a gerência de energia de equipamentos ligados a uma rede. Entretanto, existe uma série de *softwares* para monitoramento de *no-breaks* que possibilitam o desligamento de um ou mais computadores na ocorrência de alterações no fornecimento de eletricidade. Em outras palavras, essas aplicações possuem características que o sistema proposto nesse trabalho pretende incorporar. Algumas dessas aplicações foram analisadas e serão mencionadas oportunamente.

Além disso, este não é o primeiro trabalho na área de gerência de energia realizado pelo Grupo de Tolerância a Falhas do PPGC da UFRGS. Outras atividades incluem:

- o projeto de uma MIB[2] para UPS, com a definição de informações referentes à rede elétrica e ao *no-break* que poderiam ser repassadas a um sistema de gerência, na dissertação de mestrado de André Peres [PER2000];
- os trabalhos de Iniciação Científica de Tórgan F. de Siqueira [SIQ98] e Clairton Buligon [BUL98], que implementaram mecanismos de controle remoto de *no-breaks*;
- o Trabalho Individual [KRO99a] e a Dissertação de Mestrado [KRO2001] de Roger Krolow, que abordaram, respectivamente, os sistemas de gerência de energia para computadores pessoais e a gerência de energia em redes de computadores através do protocolo SNMP.

---

[1] Os termos *gerente* e *aplicação de gerência* serão usados como sinônimos ao longo do texto.

[2] *Management Information Base*, descrição formal de um conjunto de objetos que podem ser gerenciados através do protocolo SNMP.

A dissertação de Roger Krolow é particularmente importante porque tem objetivos muito próximos dos estabelecidos neste trabalho. Em suma, Krolow propôs uma solução específica para o controle de consumo de energia em redes de computadores abastecidas através de UPS, "objetivando aumentar o tempo de operação da rede em caso de operação com recursos limitados e redução de consumo de energia em situações de fornecimento normal".

Embora os objetivos tenham sido revistos e, de certa maneira, limitados, pode-se dizer que o trabalho apresentado aqui é fundamentado naquela dissertação. Certamente, não seria possível chegar aos resultados obtidos sem as idéias introduzidas naquele trabalho.

1.6 Estrutura do texto

O restante do texto desta dissertação está organizado da seguinte maneira:

- o capítulo 2 contém uma síntese sobre os padrões e mecanismos de gerência de energia para computadores, com ênfase naqueles destinados a computadores pessoais (PCs);
- o capítulo 3 descreve sistemas de monitoramento de *no-breaks*;
- a partir da revisão de sistemas correlatos, apresentada nos capítulos 2 e 3, o capítulo 4 apresenta uma visão geral do sistema proposto. Este capítulo pode ser considerado o núcleo da dissertação, onde a proposta do sistema é descrita em detalhe;
- o capítulo 5 contém a especificação dos agentes do sistema;
- o capítulo 6 descreve a implementação da aplicação de gerência. Este capítulo traz uma série de detalhes técnicos que são complementares (embora não absolutamente essenciais) à compreensão da proposta do sistema;
- o capítulo 7 apresenta os resultados dos testes realizados em laboratório;
- o capítulo 8, finalmente, apresenta as conclusões, incluindo estimativas da economia proporcionada pelo sistema.

2 Sistemas de gerência de energia para computadores

Conforme apresentado no capítulo anterior, a capacidade de limitar o consumo de eletricidade vem se tornando um aspecto cada vez mais importante da computação. Os fabricantes de *hardware* têm investido em pesquisas que visam minimizar o consumo dos componentes eletrônicos (através da diminuição da tensão de operação dos circuitos, por exemplo), além de trabalharem em conjunto com a indústria de *software*, particularmente os fabricantes de sistemas operacionais, com o objetivo de desenvolver mecanismos mais eficazes de gerência de energia.

Fundamentalmente, todos os sistemas de gerência de energia adotam o mesmo princípio básico, que consiste em reduzir o consumo quando o equipamento está inativo – o que acontece durante a maior parte do tempo em que um PC típico fica ligado. A redução de consumo é alcançada através do que se chama, em inglês, de "*sleep modes*" (*modos de dormência*) ou "*low-power modes*" (*modos de baixo consumo* ou *modos de consumo reduzido*). Segundo estimativas da EPA, cerca de 85% dos PCs e praticamente 100% dos monitores fabricados atualmente possuem recursos de gerência de energia [EPA2000].

A capacidade de alternar entre diferentes níveis de consumo é a chave para a gerência de energia em computadores [NOR97]. A idéia central é que o nível de consumo de energia de um equipamento em qualquer instante deve ser proporcional ao nível de atividade em que ele se encontra. A Figura 2.1 ilustra o processo de alternância entre modos de consumo.

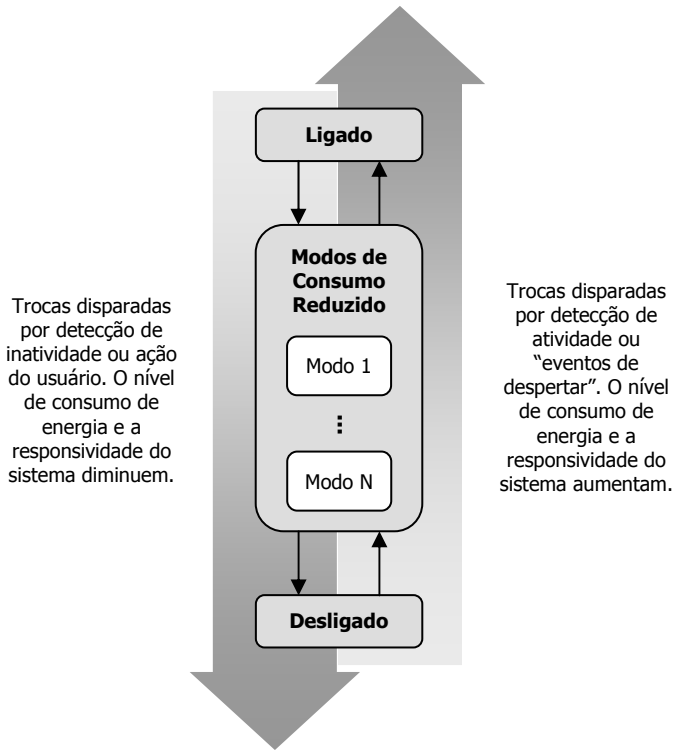


Figura 2.1 - Funcionamento básico de sistemas de gerência de energia

Conforme ilustra a figura, à medida que o nível de consumo decresce (metade esquerda do diagrama), decresce também a responsividade[1] do sistema. Em outras palavras, quanto menor é o nível de consumo, maior é o tempo que o sistema leva para retornar a um estado em que esteja disponível para utilização, e vice-versa.

A não ser quando disparados pelo próprio usuário, os modos de consumo reduzido são iniciados automaticamente, em decorrência da inatividade do sistema. Para que isso seja possível, temporizadores são utilizados para determinar quando o computador deve entrar no modo de consumo imediatamente inferior. Os intervalos entre cada modo precisam ser configurados pelo usuário. A situação inversa, quando o equipamento entra num modo de consumo mais alto, ocorre em resposta a interrupções de *hardware* que indicam atividade do usuário (como o movimento do *mouse*, por exemplo) ou de alguns dispositivos especiais (como uma interface de rede ou um fax-modem), aos quais é permitido "despertar" o computador.

O restante deste capítulo faz um apanhado geral sobre os sistemas de gerência de energia existentes, sem ter, no entanto, a pretensão de fazer uma revisão técnica detalhada sobre o assunto[2].

2.1 Estratégias de gerência de energia

A gerência de energia pode ser realizada em vários níveis num computador: no nível de *hardware*, no nível do sistema operacional, no nível de aplicação e no nível de usuário. Teoricamente, quanto mais alto o nível de abstração, mais adequado para gerência de energia, pois os níveis inferiores têm menos informações sobre a carga de trabalho geral do sistema. Em contrapartida, o usuário não tem conhecimento de detalhes sobre o consumo de energia da máquina nem disposição para desperdiçar seu tempo[3], o que torna este nível inadequado para as funções de gerência. Geralmente as aplicações também não têm informações suficientes sobre o estado da máquina por causa da abstração do sistema operacional[4]. Assim, restam os níveis de *hardware* e do sistema operacional, que têm sido explorados nas implementações existentes [LOR98].

Os primeiros mecanismos de controle de consumo de energia eram inteiramente implementados no nível de *hardware* e controlados pelo BIOS[5]. À medida que os padrões para gerência de energia vêm evoluindo, a tendência é de mover o controle para o nível do sistema operacional, permitindo inclusive alguma colaboração das aplicações. De qualquer maneira, a gerência de energia *sempre* requer a participação dos níveis inferiores (BIOS, processador e placas controladoras), pois a redução de consumo acontece, em última instância, no nível físico.

A Figura 2.2, a seguir, ilustra o envolvimento desses diferentes níveis no processo de gerência de energia, com os possíveis caminhos de comunicação entre os componentes. Conforme será apresentado adiante, os padrões de gerência mais conhecidos possuem arquiteturas que podem ser encaradas como especializações deste diagrama mais genérico.

---

[1] O termo é uma tradução livre do inglês *"responsiveness"*, que significa "capacidade de responder", mas possui um sentido implícito de "responder com rapidez, agilidade". A palavra "responsivo" consta do dicionário Aurélio [FER99]. O dicionário *online* do tradutor profissional Ivo Korytowski [KOR2002] contém também a palavra "responsividade".

[2] Para uma revisão em detalhe, ver Krolow [KRO99a].

[3] Para o usuário final, o computador é apenas uma ferramenta, e cada vez mais é encarado como um eletrodoméstico. Sob esse ponto de vista, ele não está disposto a desperdiçar "tempo produtivo" com decisões que não estejam diretamente relacionadas às tarefas para as quais ele usa o computador.

[4] Lu et al. [LU99] apresentam uma visão alternativa, argumentando sobre as vantagens de se implementar o gerenciamento de energia no nível de aplicação.

[5] *Basic Input/Output System*

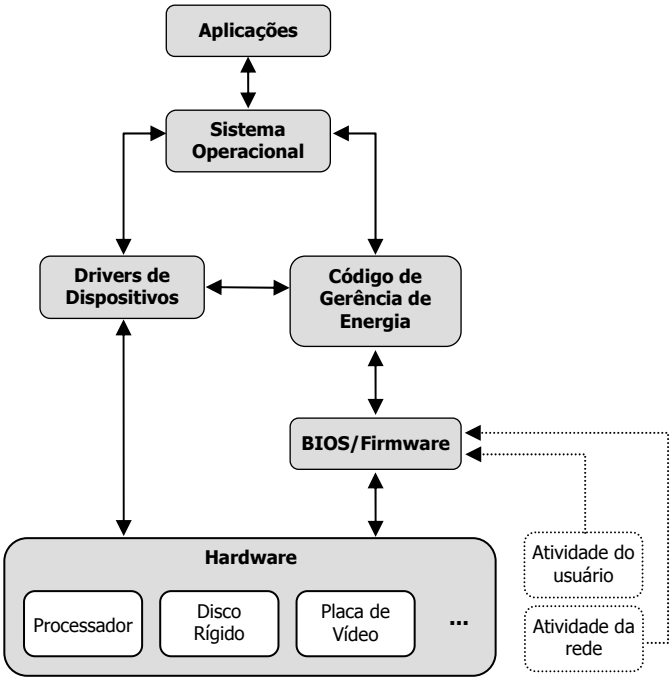


Figura 2.2 - Componentes envolvidos na gerência de energia

2.1.1 Gerência no nível de hardware

No nível de *hardware*, a gerência de energia é controlada inteiramente pelo BIOS, que se encarrega de enviar os sinais apropriados para os dispositivos quando os temporizadores internos sinalizam a inatividade do sistema. A interface de usuário para a configuração da gerência de energia varia bastante de fabricante para fabricante e também entre diferentes versões do *firmware*[1] de um mesmo fabricante. Na Figura 2.3, pode-se ver um exemplo de tela de configuração para gerência de energia no BIOS.

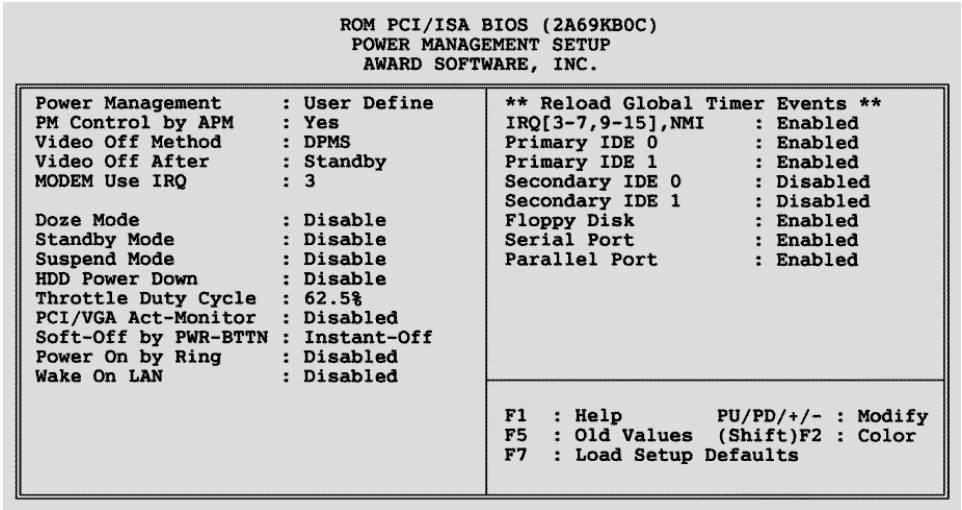


Figura 2.3 - Tela de configuração da gerência de energia no BIOS

[1] *Software* de baixo nível embutido em um dispositivo de *hardware,* que pode ser lido e executado, mas não modificado.

Como se pode ver na figura, há opções para o desligamento do monitor de vídeo e dos discos rígidos, bem como para estipular a freqüência em que o relógio da CPU deve operar quando o computador estiver em modo de baixo consumo. Também é possível configurar eventos de despertar em resposta à atividade da placa de rede ou do fax-modem.

2.1.2 Gerência no nível do sistema operacional

A transição da gerência de energia do nível de *hardware* para o nível do sistema operacional pode ser encarada como uma mudança natural, já que o sistema operacional tem melhores condições de determinar a carga geral do sistema (estado das aplicações, atividade do usuário, etc.) e, conseqüentemente, de detectar quando ele está ocioso – condição básica para disparar trocas nos estados de consumo de energia.

O conceito do sistema operacional funcionando como peça central da arquitetura de gerência de energia foi batizado pela Microsoft de "*Operating System-directed Power Management*" (OSPM). A partir deste conceito, e em conjunto com outras grandes empresas da indústria, a Microsoft criou uma nova especificação para gerência de energia (ver seção 2.3.2).

Apesar de estar no nível mais adequado para a gerência de energia, o sistema operacional não pode trabalhar sozinho. Observando-se novamente a Figura 2.2, pode-se constatar que o SO precisa da colaboração dos outros níveis para efetuar a gerência de energia adequadamente. Como será visto a seguir, os padrões de gerência de energia mais conhecidos possuem uma estrutura que é, basicamente, uma especialização da estrutura básica apresentada naquele diagrama.

Um outro aspecto muito importante do envolvimento do sistema operacional na gerência de energia é a possibilidade de configuração de políticas (ou esquemas) de consumo de energia. Essas políticas são conjuntos de decisões que determinam como o sistema economiza energia – quais dispositivos ele deve desligar ou colocar em estados de baixo consumo, e quando – e são baseadas nas preferências do usuário, nas necessidades das aplicações e nos recursos de *hardware* disponíveis no computador. As Figuras 2.4 e 2.5 mostram as interfaces de usuário para a configuração dessas políticas no Windows 2000 e no MacOS 8.5, respectivamente.

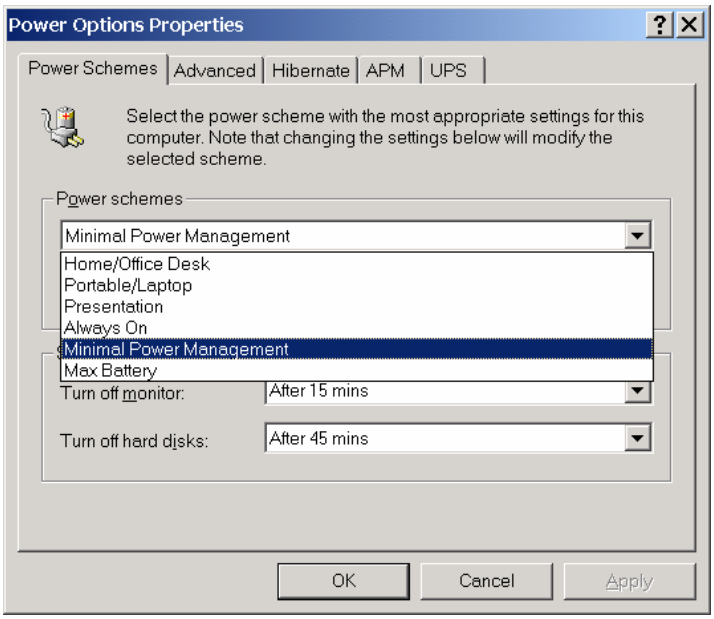


Figura 2.4 - Configuração de gerência de energia no Windows 2000

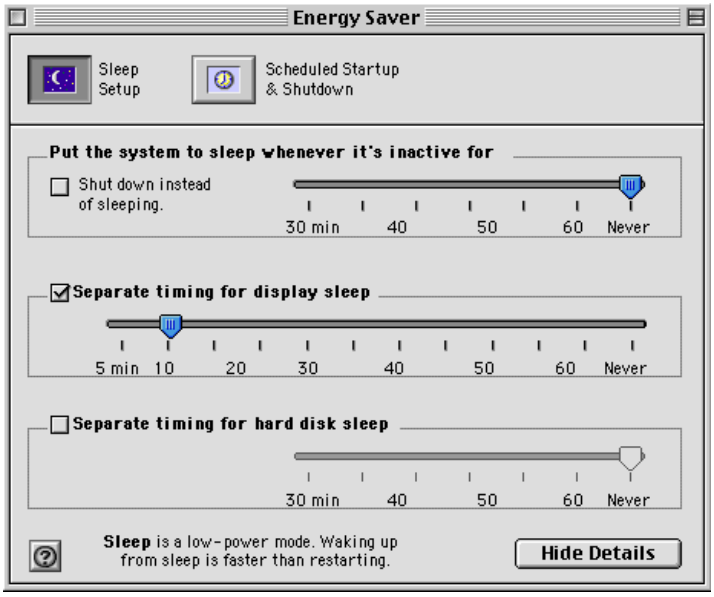


Figura 2.5 - Configuração de gerência de energia no MacOS 8.5

2.2 Implicações de redes de dados na gerência de energia

O fato de um computador fazer parte de uma rede de dados é um dos principais motivos para que a gerência de energia não seja realizada de maneira eficiente. Os equipamentos não apenas são deixados ligados em períodos de ociosidade (durante a noite e finais de semana, por exemplo), mas têm seus mecanismos de gerência de energia desabilitados para permitir o acesso remoto dos administradores do sistema e dos usuários.

A manutenção das conexões de rede nos modos de baixo consumo de energia era um problema comum em computadores mais antigos. Embora este problema persista em alguns dos modelos mais recentes, a maioria dos problemas que afetam a execução correta da gerência de energia é resultado da maneira como os sistemas de rede operam, e não de características do *hardware*. Em muitos desses sistemas, um servidor envia mensagens periódicas para os computadores da rede e, na ausência ou atraso da resposta, encerra as conexões. Isso é um dos motivos que levam à desabilitação da gerência de energia. O contrário também pode acontecer: em muitos computadores, essas mensagens causam atividade suficiente para manter o PC "acordado", impedindo a entrada em modos de consumo reduzido [NOR97].

Em versões de BIOS mais recentes, a atividade de rede recebe um tratamento diferente do que é dado a outros tipos de atividade do sistema (como do teclado e do *mouse*, por exemplo), fazendo com que só as partes do computador essenciais ao processamento das mensagens da rede sejam ativadas. Deste modo, o computador não atinge o modo de consumo mais alto, mas fica num nível intermediário até que a requisição seja processada. Existem também placas de rede capazes de responder a certos tipos de mensagens sem a intervenção da CPU e do sistema operacional, facilitando ainda mais a realização da gerência de energia (ver seção 7.5.3).

2.3 Padrões de gerência para PCs

Os computadores pessoais, especialmente os baseados na arquitetura Intel e similares, têm sido a plataforma-alvo das iniciativas mais relevantes da indústria no sentido de padronizar a gerência de energia. A partir de pesquisas que tiveram a participação das maiores empresas

mundiais de *hardware* e *software*, emergiram os dois padrões utilizados atualmente, APM e ACPI, brevemente descritos a seguir.

2.3.1 O padrão APM (Advanced Power Management)

A primeira versão da especificação APM foi apresentada pela Intel e pela Microsoft em 1992. Tendo sido revista pela última vez em 1996 [INT96], essa especificação define uma interface entre a parte do *firmware* encarregada da gerência de energia no nível físico e um *driver* de políticas de consumo do sistema operacional. A interface em si é independente do *hardware*, de maneira que os detalhes do nível físico são mascarados e o *software* de mais alto nível pode fazer uso do sistema de gerência de energia sem conhecer detalhes do *hardware*.

Em outras palavras, o APM divide a funcionalidade da gerência de energia numa hierarquia de camadas que cooperam entre si, e padroniza o fluxo de informações entre essas camadas (Figura 2.6).

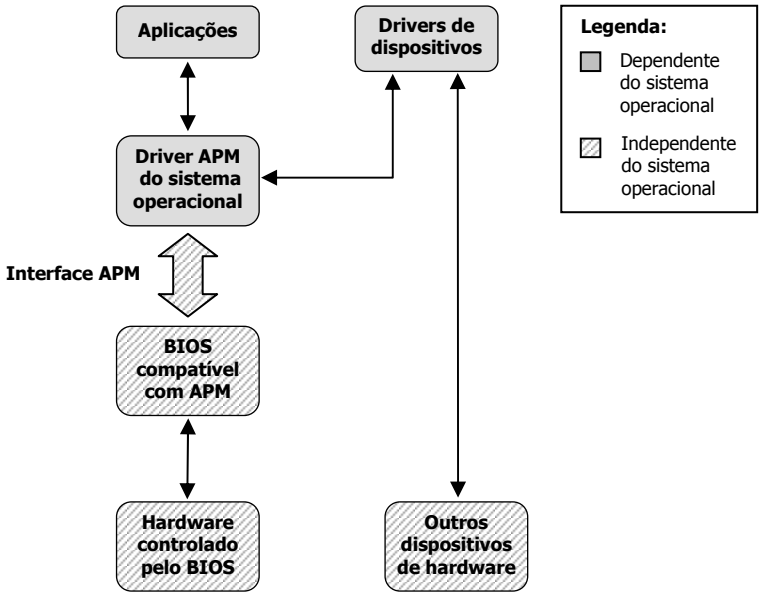


Figura 2.6 - Estrutura do sistema APM

Os principais componentes dessa hierarquia são os seguintes:

- BIOS compatível com o padrão APM: é a interface de *software* com a placa-mãe do computador e seus dispositivos;
- interface APM: define a interação entre o BIOS e o *driver* APM do sistema operacional;
- *driver* APM: comunica-se com o BIOS e controla as políticas de consumo;
- aplicações: comunicam-se com o *driver* APM para monitorar e/ou controlar a gerência de energia;
- *drivers* de dispositivos: permitem realizar a gerência de energia diretamente nos dispositivos, sem a intervenção do BIOS.

O APM define modos de consumo globais (que afetam o sistema inteiro) e também modos para dispositivos individuais e para o processador. Os modos globais vão desde "totalmente ligado" (*full on*), quando a gerência de energia não está em efeito, até "desligado" (*off*), passando por três modos graduais de consumo reduzido (*APM enabled*, *APM standby* e *APM suspend*). Os modos de dispositivos e do processador seguem o mesmo princípio.

2.3.2 O padrão ACPI (Advanced Configuration and Power Interface)

A especificação do padrão ACPI é resultado de uma iniciativa conjunta de cinco grandes empresas da indústria de computação: Compaq, Intel, Microsoft, Phoenix Technologies e Toshiba. A versão mais recente da especificação data de 2000 [COM2000].

O padrão ACPI foi concebido para substituir o sistema APM e outros sistemas de gerência de energia de mais baixo nível, e pretende ser aplicável a todos os tipos de computadores, de pequenos portáteis a grandes servidores. Ainda assim, como no caso do APM, a especificação do padrão ACPI descreve, basicamente, interfaces entre elementos de *hardware* e *software* que permitem a execução da gerência de energia. A grande diferença está no nível de detalhe das duas especificações. Enquanto a especificação do APM tem cerca de 80 páginas, a do ACPI tem quase 500. A especificação ACPI inclui uma série de recomendações para a construção de *hardware* compatível, o que anteriormente ficava completamente a cargo de cada fabricante.

A Figura 2.7 ilustra os elementos envolvidos na gerência de energia segundo o padrão ACPI.

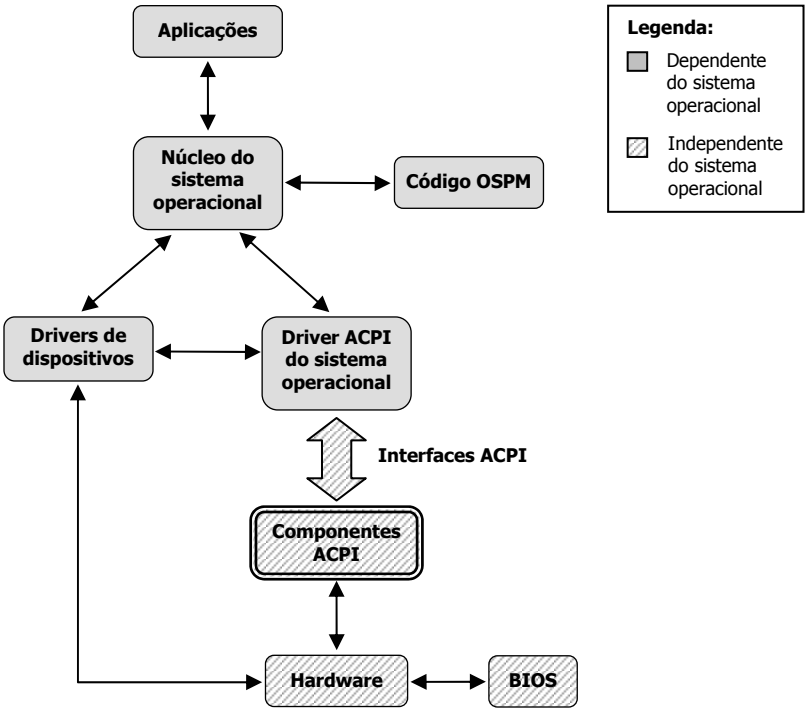


Figura 2.7 - Estrutura do sistema ACPI

Os *componentes ACPI* (indicados na figura pelo retângulo de borda dupla) são três:

- tabelas de descrição do sistema: descrevem o *hardware* do equipamento e podem conter seqüências de operações necessárias para fazer o *hardware* funcionar. Essas operações podem ser definidas através de uma espécie de pseudo-código (*ACPI Machine Language*, ou AML), que é interpretado pelo *driver* ACPI do sistema operacional. Estas tabelas podem ser consideradas o coração da estrutura de gerência de energia no padrão ACPI;
- registradores: realizam o controle do *hardware* descrito nas tabelas;

- BIOS compatível: parte do *firmware* que é compatível com a especificação ACPI. É o elemento responsável por prover as tabelas de descrição do sistema e implementar as interfaces para operações de gerência de energia (operações de *sleep, wake-up*, etc.).

Para cada um dos componentes, existe uma interface específica que padroniza a comunicação com o *driver* ACPI do sistema operacional. Apesar da separação aparente no diagrama, esses componentes são implementados no nível de *hardware*.

A figura também mostra que, de maneira alternativa, o *hardware* pode ser controlado diretamente por *drivers* de dispositivos que não são compatíveis com ACPI, mas implementam seus próprios mecanismos de gerência de energia.

Assim como no padrão APM, são definidos modos de consumo globais para o sistema e modos individuais para os dispositivos e o processador. A lógica dos modos de consumo também é semelhante, iniciando por um estado de energização total e passando por um ou mais modos de consumo reduzido, até um estado em que o dispositivo (ou o computador inteiro) está completamente desligado.

2.3.3 Uma nota sobre gerência de energia em monitores

A história da gerência de energia em monitores de vídeo é mais bem-sucedida do que a da gerência de energia em PCs. O padrão DPMS (*Display Power Management Signaling*), lançado em 1993, define um método para o envio de sinais especiais, pela placa de vídeo, que fazem o monitor entrar em modos de baixo consumo [VES2002]. Este padrão é largamente adotado por fabricantes de monitores e controladoras de vídeo.

Entre as razões que contribuem para o sucesso da gerência de energia em monitores e que a tornam ainda mais importante estão o grande potencial de economia de energia[1] e o fato de que os modos de consumo reduzido do monitor não interferem no funcionamento do restante do sistema.

2.4 Sutilezas de sistemas de gerência de energia

Como se pôde perceber até aqui, o uso de mecanismos de controle do consumo de energia traz uma série de benefícios. Alguns são bastante óbvios, como a redução de custos e a diminuição do impacto ambiental. Outros efeitos mais sutis incluem gastos menores com ar-condicionado[2], menor desgaste de partes móveis dos equipamentos (p. ex. discos rígidos), menos defeitos causados por superaquecimento, menor emissão de radiação por monitores de vídeo e redução do ruído provocado por ventiladores internos (*coolers*) e discos rígidos.

No entanto, há uma conseqüência adversa do uso de modos de consumo reduzido de energia: eles também podem acarretar desperdício. Com freqüência, o ato de desligar um aparelho eletrônico não necessariamente interrompe o consumo de eletricidade. Por uma série de razões (como manter ativo um sensor de controle remoto, por exemplo), vários aparelhos domésticos são colocados num estado de espera (*standby*), que reduz drasticamente o consumo, mas não é equivalente a desligar o aparelho da fonte. O efeito é um "vazamento" invisível de eletricidade que tem um sério impacto econômico e ambiental [RAL97].

---

[1] Enquanto o consumo médio de um computador *desktop* é de 50W em estado ativo e 25W em modo de baixo consumo (economia de 50%), o de um monitor é de 75W em estado ativo e 5W em baixo consumo (economia de mais de 90%) [KAW2001].

[2] Segundo Cramer, para cada 4 KWh economizados com um computador, 1 KWh adicional é economizado em refrigeração [CRA95].

Essa tendência de manter equipamentos eletrônicos responsivos 100% do tempo está sendo claramente incorporada pela indústria de PCs. A Intel e a Microsoft vêm desenvolvendo projetos cujo objetivo é tornar os PCs instantaneamente disponíveis, do mesmo modo que televisores e outros eletrodomésticos. A Intel usa o nome "*Instantly Available Technology*" [INT2001] e a Microsoft, "*OnNow Architecture*" [MIC2002]. As duas iniciativas especificam um conjunto de recomendações para a fabricação de PCs com características de *hardware* que permitam esse tipo de disponibilidade, e usam o padrão ACPI como elemento central para a gerência de energia.

Mesmo que se desconsidere a tendência do "PC-eletrodoméstico", há estudos que comprovam que, mesmo hoje, a confusão quanto à finalidade dos modos de baixo consumo gera desperdício. Para citar um exemplo prático, uma pesquisa do Lawrence Berkeley National Laboratory (Universidade da Califórnia) apontou que 68% dos monitores não são desligados depois do horário normal de expediente nas empresas, provavelmente porque os usuários assumem que os monitores irão desligar-se automaticamente, quando na verdade eles apenas entram num modo de consumo reduzido [WEB2001].

Esses dados comprovam o potencial para desperdício causado pelo uso incorreto dos mecanismos de gerência de energia, cujo propósito fundamental é justamente proporcionar uma redução no consumo. Por outro lado, eles alertam para a importância da configuração e utilização corretas desses mecanismos, e também de seu entendimento pelo usuário. Essas medidas podem não só evitar o gasto desnecessário de eletricidade, mas permitir uma economia efetiva.

Finalmente, é importante salientar que dificilmente haverá uma situação ideal de consumo, principalmente quando se trata de computadores ligados em rede. Nesse tipo de ambiente, outros custos relacionados com a manutenção da rede podem ser mais relevantes do que o eventual custo extra causado por equipamentos que não são completamente desligados, mas permanecem em estados de baixo consumo[1]. Por exemplo, manter os PCs de uma rede em estado de consumo mínimo ao longo de um final de semana – o que permitiria a um administrador despertá-los sob demanda e efetuar tarefas de manutenção – provavelmente teria um custo menor do que interferir com a operação da rede durante a semana para efetuar as mesmas tarefas. A melhor abordagem parece ser encontrar uma linha de equilíbrio entre a economia de energia e a disponibilidade dos equipamentos, adequada a cada ambiente específico.

[1] Freqüentemente, encontra-se na literatura o conceito de "custo total de propriedade" (*"total cost of ownership"*, ou TCO, em inglês) que engloba todos os custos envolvidos na posse de um computador, tanto os de aquisição como também os de manutenção e de gerenciamento.

3 Sistemas de monitoramento de no-breaks

3.1 No-breaks e proteção de dados

A instalação de um *no-break* numa rede cria um ambiente protegido de distúrbios no fornecimento de energia. Mas isso é feito com um propósito maior: preservar os dados nos computadores mesmo quando as falhas no abastecimento perduram. Para que isso seja possível, é preciso que o *no-break* seja constantemente monitorado e que, em caso de alterações no abastecimento, os equipamentos implicados sejam automaticamente desligados antes que o corte de energia efetivamente aconteça.

Esse tipo de monitoramento preventivo é normalmente realizado através de uma interface serial ou de rede disponível no equipamento. Em modelos de *no-break* mais recentes, controlados por microprocessadores, a comunicação é feita através de protocolos padrão, como SNMP ou HTTP, ou de um protocolo patenteado[1]. Os aparelhos mais simples, que não suportam um protocolo de comunicação propriamente dito, dispõem de um sistema de sinalização mais rudimentar, que permite detectar um número reduzido de eventos.

A maioria dos fabricantes fornece, junto com o *no-break*, um *software* cuja função é monitorar o funcionamento do aparelho e as condições do abastecimento de energia. Embora não exista uma nomenclatura padronizada para se referir a esse tipo de sistema, os termos *sistemas de gerência de energia*[2] e *sistemas de proteção de energia*[3] são utilizados com bastante freqüência, apesar do sentido ambíguo das expressões. Alguns desses aplicativos possuem também outras funções, como configuração e controle dos *no-breaks*. Neste trabalho, entretanto, o interesse é restrito à função de *monitoramento* – daí o título deste capítulo.

3.2 Características comuns

Geralmente, os aplicativos de monitoramento apresentam algum tipo de representação gráfica do *no-break*, como mostra o exemplo da Figura 3.1. Adicionalmente, permitem definir ações que devem ser executadas quando o *no-break* passa a alimentar a rede a partir de um banco de baterias, de modo a prevenir a perda de dados nos equipamentos. Essas ações tipicamente incluem avisar os usuários da condição de anormalidade e desligar automaticamente os computadores da rede[4]. Cabe salientar que o objetivo geral desses sistemas *não é* economizar energia, mas sim proteger os computadores da rede em caso de um corte iminente.

Na verdade, a operação em bateria é apenas um dos eventos (também chamados de "alertas") aos quais se pode associar a execução de ações. Normalmente, o conjunto de alertas inclui avisos sobre o nível baixo ou crítico da bateria e também sobre o restabelecimento da força externa. Alguns sistemas trabalham com um número bastante grande de alertas, a maioria dos quais se refere a parâmetros de operação do próprio *no-break* (como níveis de corrente ou tensão fora das margens de segurança, superaquecimento, etc.).

---

[1] Coloquialmente, usa-se o termo "proprietário" (uma tradução incorreta do inglês *"proprietary"*) para se referir a esse tipo de protocolo ou tecnologia fechada, de propriedade particular da empresa que detém os respectivos direitos autorais, e que normalmente mantém a especificação em segredo. [MIC98]

[2] *Power management systems*, em inglês.

[3] *Power protection systems*, em inglês.

[4] Esse procedimento é chamado, em inglês, de "*unattended shutdown*" (desligamento desassistido), e inclui o encerramento de todas as aplicações e do sistema operacional do computador antes do seu desligamento propriamente dito.

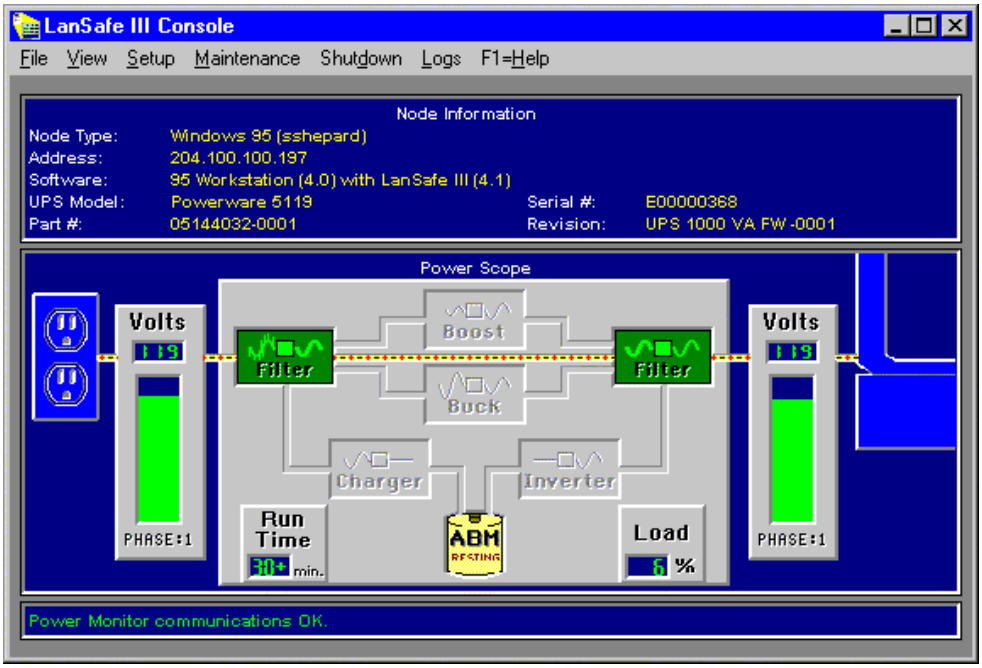


Figura 3.1 - Exemplo de sistema de monitoramento de *no-break* [POW2002]

Outras ações que podem ser comumente associadas à ocorrência de alertas são o envio de notificações via e-mail ou *pager* para o administrador da rede e a execução de comandos ou *scripts* nos computadores gerenciados. A Figura 3.2, a seguir, mostra outra tela do programa do exemplo anterior, na qual é possível associar alertas à execução de ações.

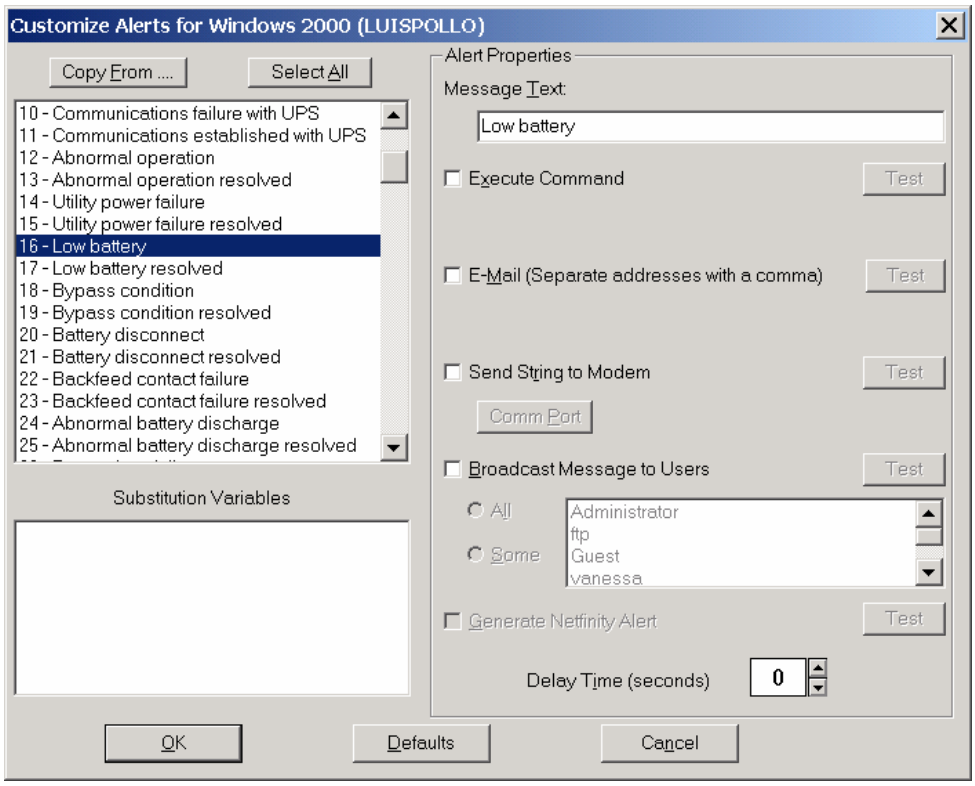


Figura 3.2 - Associação entre alertas e ações no programa LanSafe

Um recurso menos freqüente é a programação, com antecedência, do desligamento automático dos computadores. Entretanto, esse recurso é particularmente interessante porque, ao contrário das funções mais comuns, não está relacionado com a proteção dos computadores e dos dados, e sim com a racionalização do consumo. Essa programação consiste em definir o horário e a regularidade com que as máquinas devem ser desligadas (numa única data, diariamente, semanalmente, etc.).

Outros recursos incluem a geração de históricos de eventos (*log files*) e o acesso padronizado a informações e alertas da UPS através do protocolo SNMP. Este último permite que uma plataforma de gerência de rede (como o HP OpenView, por exemplo) possa interagir com o *no-break* mesmo quando ele não tem suporte para o protocolo SNMP diretamente. Em outras palavras, o *software* de monitoramento serve como um elemento intermediário (*proxy*) entre o *no-break* e a aplicação de gerência. A padronização do acesso deve-se ao uso da MIB genérica para a gerência de UPS, definida na RFC 1628 [CAS94].

3.3 Estrutura básica

Todos os sistemas de monitoramento de *no-break* capazes de coordenar o desligamento de vários equipamentos da rede na iminência de um corte de energia são implementados como aplicações distribuídas, baseadas no modelo cliente-servidor. Neste caso específico, o modelo consiste de um *nodo monitor*, encarregado da comunicação direta com a UPS, e de um conjunto de *nodos clientes*, para os quais o monitor repassa informações e alertas. Entre o monitor e os clientes, a comunicação é geralmente feita através de um protocolo de aplicação privado, que roda sobre TCP/IP. Como o monitor é o ponto central do sistema, responsável pelo disparo das ações de desligamento nos demais nodos, em alguns sistemas ele é chamado de *servidor de desligamento* (*shutdown server*).

A Figura 3.3 ilustra esse modelo básico. As linhas entre os elementos do diagrama representam as conexões da rede de dados, não da rede elétrica. Logicamente, só faz sentido que os computadores que são efetivamente alimentados por um determinado *no-break* sejam notificados dos eventos reportados por este mesmo *no-break*.

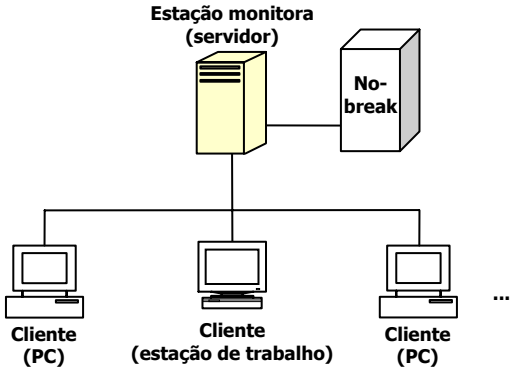


Figura 3.3 - Estrutura de sistemas de monitoramento de *no-breaks*

A Tabela 3.1 contém uma lista dos *softwares* de monitoramento analisados durante o trabalho[*] e dos respectivos fabricantes e endereços na Web, em adição àqueles previamente analisados na dissertação de Krolow[**]. Foi a partir da análise desses sistemas que se determinaram a estrutura e as características comuns descritas anteriormente.

Tabela 3.1 - Sistemas comerciais de monitoramento de no-break

	Software	Empresa	Endereço na Web
Krolow	PowerChute	American Power Conversion	http://www.apc.com/
	SiteNet Multilink	Liebert Corporation	http://www.liebert.com/
	SUPSNet	Engetron S.A.	http://www.engetron.com.br/
Pollo	CP-Monitor e CP-Ctrl	CP Eletrônica S.A.	http://www.cp.com.br/
	LanSafe, OnliNet e NetWatch	Powerware Corp.	http://www.powerware.com/
	PowerFlag	GE Digital Energy / IMV	http://www.gedigitalenergy.com/
	PowerMon e SmartMon	Systems Enhancement Corp.	http://www.seqhq.com/
	PowerNet (Manager/Agent)	American Power Conversion	http://www.apc.com/

A metodologia para a análise consistiu em revisar a documentação disponível para cada *software* e, caso possível, instalar o programa e fazer uma inspeção visual de seus recursos. Durante a revisão da documentação, guias de usuário e folhetos informativos (*datasheets*) foram particularmente úteis, assim como arquivos de ajuda durante a inspeção dos programas.

3.4 Finalizando a revisão de sistemas correlatos

Tendo discutido o funcionamento básico dos sistemas de gerência de energia para computadores no capítulo anterior, e o funcionamento dos sistemas de monitoramento de *no-breaks* neste capítulo, tem-se a base sobre a qual o restante do trabalho será apresentado, particularmente o próximo capítulo, que traz uma descrição do sistema que se está propondo e aproveita os conceitos introduzidos até aqui.

4 Descrição do sistema proposto

O sistema proposto neste trabalho pode ser encarado como uma combinação dos dois tipos de *software* apresentados nos capítulos anteriores: o de monitoramento de *no-breaks* e o de gerência de energia para computadores.

Conforme mencionado anteriormente, os sistemas de monitoramento de *no-break* provêem mecanismos para desligar os computadores da rede em caso de alterações no fornecimento de energia, sendo que alguns também permitem programar o desligamento periódico das máquinas. Por outro lado, os sistemas de gerência de energia trazem a noção de políticas de consumo, através das quais é possível configurar o modo como o computador economiza energia. A possibilidade de escolher quais dispositivos do computador devem ser desativados para reduzir o consumo torna-se particularmente interessante quando ele está conectado a uma rede e desligá-lo completamente não é uma alternativa adequada.

Em suma, a idéia central da proposta é aproveitar características desses dois tipos de sistemas com o objetivo de facilitar a economia de energia no ambiente de rede. Mais precisamente, a intenção é tomar como base um tipo de arquitetura de *software* cuja eficácia é comprovada pela ampla utilização na indústria de *no-breaks*, e adicionar a ela recursos que permitam não apenas desligar os equipamentos da rede, mas regular o seu consumo de energia através da configuração de políticas e da aplicação de estados intermediários de energização.

Unindo a infra-estrutura de comunicação de um sistema de monitoramento de *no-break* com a capacidade de programar e executar alterações no modo de consumo de energia de cada computador, tem-se o modelo de funcionamento básico do sistema proposto neste trabalho, como será apresentado no restante deste capítulo. O sistema foi batizado de *NetPower* (uma abreviação de *"Network-oriented Power Management System"*).

4.1 Esclarecimentos sobre os objetivos e capacidades do sistema

O sistema proposto nesta dissertação não tem o objetivo de substituir os *softwares* apresentados nos capítulos anteriores, cujas finalidades específicas são a gerência de energia (local) em PCs e a proteção de dados em ambientes abastecidos através de *no-break*. Como já foi enfatizado, a idéia é aproveitar alguns conceitos dos dois tipos de sistemas, mas não se tem a pretensão de criar um produto que substitua completamente qualquer um deles.

Como será apresentado ao longo deste capítulo, o sistema *NetPower* pode, de fato, atuar como uma aplicação de monitoramento de *no-breaks* genérica e garantir que os equipamentos da rede sejam desligados na iminência de um corte no fornecimento de energia. Entretanto, como foi visto no capítulo anterior, alguns programas dessa categoria são capazes de tratar uma série de diferentes eventos reportados pelo *no-break*, o que está além do escopo da aplicação proposta aqui.

Ainda em relação aos *softwares* de monitoramento, é importante enfatizar que há uma certa mudança de foco no sistema aqui proposto. Enquanto o centro da atenção naqueles sistemas é a proteção de dados, neste trabalho o ponto de maior interesse é a economia de energia – em como exercer, remotamente, algum controle sobre o consumo dos equipamentos da rede. Como a infra-estrutura criada para permitir a gerência de consumo também é adequada para a proteção de dados, o sistema pode atuar nas duas áreas. Vale lembrar que a maioria dos sistemas fornecidos pelos fabricantes de UPS não permite apenas monitorar o aparelho, mas também intervir, alterar parâmetros, controlar sua operação. Esses tipos de funções não serão

abordados na proposta contida neste trabalho. Aqui, o *no-break* desempenha um papel coadjuvante: funciona como um elemento informador, através do qual o sistema pode detectar condições anormais no fornecimento de energia.

Quanto à infra-estrutura local de gerência de energia disponível nos computadores, o objetivo é apenas fazer uso dos recursos que ela provê. Na verdade, como se viu anteriormente, os sistemas de gerência de energia existentes são implementações de padrões da indústria. A proposta apresentada neste trabalho, por sua vez, apóia-se no conjunto de funções especificado nesses padrões. Sendo assim, ela não tem a intenção de substituir quaisquer especificações ou implementações de sistemas de gerência local de energia[1]. Outra maneira de compreender o papel do sistema *NetPower*, neste sentido, é voltar ao conceito de gerência centralizada apresentado no capítulo 1. O que o sistema permite centralizar é a configuração de políticas de consumo, que são aplicadas a cada computador através de mecanismos já existentes, como os providos pelos sistemas APM e ACPI, por exemplo.

Além disso, não é objetivo do sistema substituir a configuração local de políticas de consumo em cada equipamento gerenciado. Como será apresentado adiante, o sistema prevê a habilitação e execução da gerência de energia dos computadores independentemente da intervenção da aplicação de gerência centralizada. É recomendável que cada computador da rede seja individualmente configurado para economizar o máximo de energia possível. O papel da aplicação de gerência centralizada é facilitar o trabalho de configuração e atuar nos equipamentos que, por ventura, não tenham sido configurados adequadamente.

4.2 Arquitetura

A arquitetura do sistema deriva, basicamente, do modelo padrão para a gerência de redes TCP/IP, baseado no protocolo SNMP. Consiste de um *gerente*[2], que concentra a maior parte da complexidade lógica, e de *agentes* instalados em cada uma das máquinas que se pretende gerenciar, conforme ilustra a Figura 4.1.

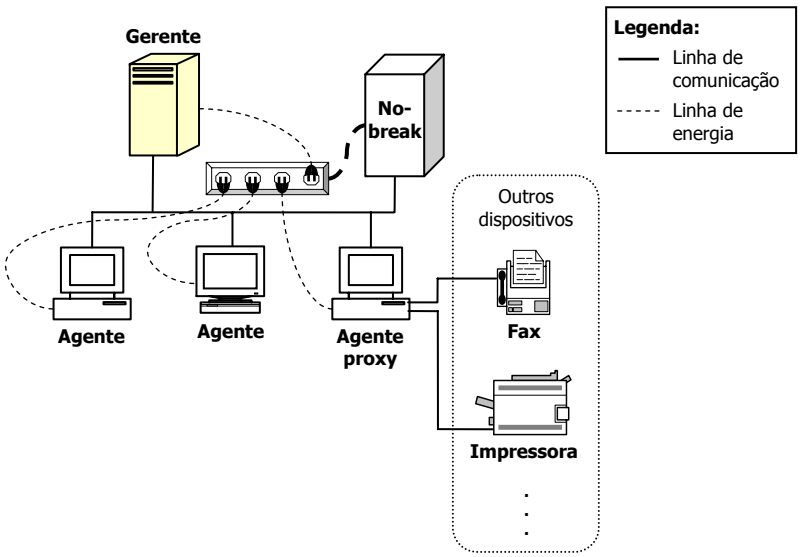


Figura 4.1 - Arquitetura do sistema *NetPower*

[1] O termo "gerência local de energia" será usado daqui por diante para diferenciar os sistemas de gerência de energia executados individualmente nos próprios PCs do sistema de "gerência centralizada" proposto no trabalho.

[2] Também chamado de *estação SNMP* ou *estação de gerência* (em inglês, usa-se a sigla NMS, de *"Network Management Station"*).

Teoricamente, qualquer dispositivo eletrônico da rede pode ser gerenciado pelo sistema, desde que atenda dois requisitos básicos: seja acessível através da rede TCP/IP (diretamente ou através de um computador intermediário) e possua algum mecanismo interno de gerência de energia que possa ser controlado pelo agente. De modo semelhante, *no-breaks* que não podem ser conectados à rede TCP/IP diretamente podem ser monitorados através de um *proxy*.

Embora não se possa dizer que este se trata de um modelo cliente-servidor sob a acepção tradicional do termo[1], pode-se facilmente fazer uma analogia com o modelo empregado nos sistemas de proteção de dados fornecidos pelos fabricantes de *no-break* (ver Figura 3.3, pág. 38). Por outro lado, deve ficar clara a responsabilidade do gerente de realizar a maior parte do processamento de dados dentro do esforço coordenado do sistema para proporcionar a economia de energia, o que caracteriza genericamente as aplicações distribuídas do tipo cliente-servidor.

Tanto o gerente quanto os agentes são aplicações de *software*. Os exemplos de *hardware* do diagrama são meramente ilustrativos. Como em qualquer aplicação SNMP, gerente e agentes precisam estar de acordo sobre as informações e operações disponíveis, o que é definido através de uma base de informações (MIB). Uma MIB específica para a gerência de energia foi desenvolvida neste trabalho (*Power Management MIB*, ver seção 5.2).

No diagrama, também a título de exemplo, equipamentos como fax e impressora não estão ligados às saídas de energia do *no-break*. Neste cenário, sob o ponto de vista da proteção da rede, não faria sentido associar o desligamento desses aparelhos a situações de falha no fornecimento. Normalmente, o corte de energia para equipamentos de escritório, com exceção dos próprios computadores, não tem maiores implicações (oscilações, por outro lado, podem danificar qualquer equipamento eletrônico). É claro que mesmo os equipamentos não abastecidos pelo *no-break* podem ser automaticamente desligados pelo sistema em horários pré-determinados, para economizar energia.

4.2.1 Comunicação gerente-agente

Normalmente, no modelo de gerência SNMP, a iniciativa de obter informações sobre um agente fica a cargo da estação de gerência. Para efeito de monitoramento, a estação de gerência é tipicamente configurada de maneira a consultar periodicamente os agentes da rede e determinar seu estado, incluindo condições de erro. Esse tipo de procedimento é chamado, em inglês, de *polling*.

Num sistema de gerência de energia, entretanto, a estratégia de *polling* não é adequada, pois o agente freqüentemente não está em condições de responder às requisições devido à entrada do computador em um modo de baixo consumo. A seção 2.2 explica as principais implicações de redes na gerência de energia e torna ainda mais claras as razões pelas quais a estratégia de *polling* não é uma solução viável para a implementação do sistema proposto neste trabalho.

Para contornar este problema, o sistema utiliza a estratégia alternativa: deixa a iniciativa de notificação do gerente a respeito de eventos relevantes num determinado equipamento por conta do agente correspondente. Isso evita a interferência da estação de controle na gerência de energia efetuada localmente no equipamento, bem como diminui consideravelmente o tráfego de mensagens SNMP. As mensagens de notificação também são conhecidas como "*traps*".

[1] No que se refere a quais elementos iniciam as conexões, já que, neste sentido, os papéis de servidor e cliente se misturam no protocolo SNMP.

Quanto se trata das operações de controle do consumo de energia dos equipamentos, entretanto, o gerente precisa ter, necessariamente, um comportamento ativo. Assim, para comandar a entrada de um equipamento em diferentes estados de consumo, por exemplo, o gerente envia mensagens SNMP do tipo `Set-Request` para alterar o valor de um objeto específico da MIB, que o agente interpreta como uma requisição para executar a ação correspondente. O mesmo procedimento é usado para a execução dos diversos tipos de ações admitidos pelo sistema (seção 4.3.2).

4.3 Modelo de funcionamento

A base do mecanismo de funcionamento do sistema está na capacidade de associar a ocorrência de eventos à execução de ações, de maneira muito semelhante ao que é feito em sistemas de monitoramento de *no-break* (ver seção 3.3). São definidos dois tipos básicos de eventos aos quais o sistema pode reagir: *eventos de energia* e *eventos programados*.

Os eventos de energia, como o nome indica, são referentes ao fornecimento de energia. Este tipo de evento é detectado pela aplicação de gerência através do monitoramento passivo ou ativo dos *no-breaks*. Quando o *no-break* é capaz de enviar notificações informando sobre alterações nas condições do abastecimento de energia (através de *traps* SNMP, por exemplo), a aplicação de gerência pode adotar um comportamento passivo, aguardando o recebimento desses avisos. Caso contrário, a aplicação de gerência precisa inquirir o equipamento periodicamente. Eventos de energia permitem evitar a perda de dados nos computadores da rede em caso de um corte iminente no fornecimento.

Os eventos programados, por sua vez, funcionam como "despertadores": são configurados para disparar a execução de ações numa determinada data e horário ou para repetirem-se diária, semanal ou mensalmente no mesmo horário. Estes eventos são os elementos centrais do sistema no que diz respeito à configuração de políticas de consumo para os equipamentos com a intenção de racionalizar o uso da energia. Por exemplo, eles permitem ao administrador do sistema especificar que computadores não essenciais à manutenção dos serviços de rede sejam desligados todas as noites.

Em resposta à ocorrência destes dois tipos de eventos, o sistema é capaz de executar diferentes tipos de ações, que permitem desde mostrar uma mensagem de aviso a um usuário até desligar completamente o computador. O conjunto de tipos de ações é especificado em detalhes na seção 4.3.2. A associação entre eventos e ações é feita através de um arquivo de configuração no gerente.

O diagrama da Figura 4.2 representa o modelo básico de execução de ações em resposta à ocorrência de eventos.

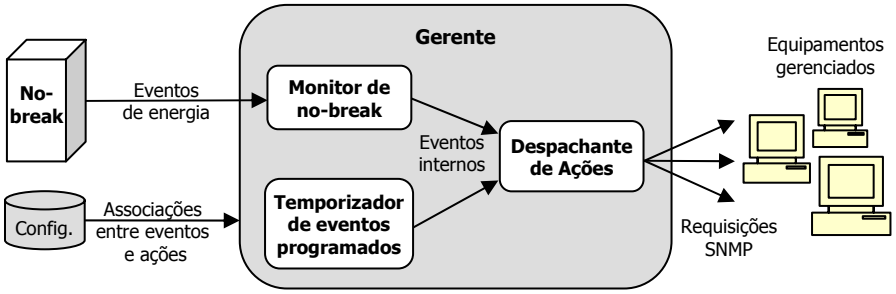


Figura 4.2 - Eventos e ações

4.3.1 Eventos de energia suportados

Através da investigação dos *softwares* de monitoramento de *no-break* listados na Tabela 3.1 (seção 3.3) e da MIB padrão para gerência de UPS (UPS-MIB, da RFC 1628), foi possível constatar que os eventos efetivamente relacionados ao estado do fornecimento de energia são poucos. Os eventos suportados pelo sistema proposto, listados na Tabela 4.1, são resultantes desta investigação.

Tabela 4.1 - Eventos de energia monitorados pelo sistema

Nome	Descrição do evento
`UPS_ON_BATTERY`	UPS operando exclusivamente em bateria (sem força externa)
`UPS_LOW_BATTERY`	Bateria da UPS em nível baixo
`UPS_DEPLETED_BATTERY`	Bateria esgotada (a UPS será incapaz de manter a carga atual se o fornecimento externo for interrompido)
`EXTERNAL_POWER_RETURNED`	Força externa restabelecida

Destes eventos, apenas um pode ser considerado absolutamente essencial. Em condições de funcionamento normais, um *no-break* sempre passa a operar em bateria antes de interromper definitivamente o fornecimento. Assim, basta ter conhecimento sobre o evento de início da operação em bateria (`UPS_ON_BATTERY`) para disparar o desligamento automático dos computadores abastecidos pelo *no-break* e protegê-los do corte.

Cabe salientar que o significado do evento `UPS_ON_BATTERY`, neste trabalho, é: "o *no-break* está operando **exclusivamente** em bateria". Ou seja, a força de entrada, proveniente da rede elétrica externa, não está sendo utilizada para abastecer a carga[1] ou carregar as baterias. Diferentes tipos de *no-break* fornecem energia aos equipamentos de maneiras distintas. Enquanto alguns modelos só alternam para a operação em bateria caso ocorra falha ou interrupção no fornecimento externo, outros operam suprindo a energia permanentemente através do circuito de bateria, utilizando a energia da concessionária apenas para manter a bateria carregada[2]. Isto significa que o monitoramento de cada *no-break* em particular deve ser adequado ao seu modelo de funcionamento, de maneira que o evento `UPS_ON_BATTERY` só seja enviado quando a única fonte de energia for a bateria. Afinal, é a partir do momento em que o *no-break* passa a descarregar as baterias (e enquanto a força externa não é restabelecida) que o abastecimento dos equipamentos pode ser comprometido. Este é um detalhe muito importante, porque o esquema de configuração criado para o sistema associa uma ou mais ações a *cada* ocorrência de um evento. Se o evento for gerado repetidamente, as ações também serão repetidas, o que certamente não é desejável.

A situação inversa ao início de operação em bateria equivale ao restabelecimento da força externa, representado pelo evento `EXTERNAL_POWER_RETURNED`. Neste caso, independente do tipo de *no-break* utilizado, o retorno da força externa cria condições para que os equipamentos da rede voltem a ser abastecidos. Sendo assim, equipamentos que tenham sido desligados quando o *no-break* passou a operar exclusivamente em bateria podem ser automaticamente religados pelo sistema. Isso pode ser particularmente útil para equipamentos que provêem serviços essenciais, como servidores de rede, por exemplo.

[1] O termo "carga", no jargão específico, significa o conjunto de equipamentos abastecidos pelo *no-break*.

[2] Para uma descrição mais aprofundada dos tipos de *no-break* e suas características, ver Peres [PER2000].

Além dos dois eventos básicos descritos acima, são definidos outros dois eventos, que permitem refinar a configuração do sistema: `UPS_LOW_BATTERY` e `UPS_DEPLETED_BATTERY`. O evento `UPS_LOW_BATTERY` indica que a bateria do *no-break* atingiu um nível baixo, e pode ser encarado como uma versão mais crítica do evento `UPS_ON_BATTERY`. Ele permite evitar a decisão precipitada de executar alguma ação em caso de falhas de curta duração no fornecimento. Podem-se associar ações menos drásticas (ou mesmo omiti-las) à entrada do *no-break* em bateria, e instruir o sistema a desligar os equipamentos apenas quando o *no-break* sinalizar que o nível da bateria está baixo. Se a força externa for restabelecida antes que o nível da bateria atinja este ponto, os equipamentos não terão sido desligados desnecessariamente.

O evento `UPS_DEPLETED_BATTERY` é usado para indicar que a bateria do *no-break* está esgotada, e tem, basicamente, uma função preventiva. Ele pode ser usado, por exemplo, para avisar o administrador do sistema de que o *no-break* não terá condições de sustentar a carga caso o fornecimento externo seja interrompido. Em termos práticos, um *no-break* tem, tipicamente, apenas alguns segundos de autonomia quando a bateria atinge este nível, de modo que não é seguro esperar a ocorrência deste evento para só então acionar o desligamento das máquinas abastecidas, quando o *no-break* já estiver operando em bateria. Entretanto, pode ser que a bateria se esgote mesmo quando a força externa esteja presente (pelo tempo de uso, por exemplo). Neste caso, pode ser útil notificar o administrador do sistema, conforme descrito acima.

4.3.2 Ações suportadas

O sistema admite cinco tipos diferentes de ações. A Tabela 4.2 lista esses tipos de ações e os parâmetros que devem ser especificados para cada um deles no arquivo de configuração. Cada ação é sempre associada a um "alvo" – o equipamento ou grupo de equipamentos no qual ela deve ser executada. Também é possível configurar um atraso para a execução da ação, isto é, um tempo entre a ocorrência do evento e o despacho da ação, que pode servir para evitar a execução de ações na ocorrência de "alarmes falsos" (uma queda momentânea do fornecimento de energia, por exemplo).

Tabela 4.2 - Ações suportadas pelo sistema

Tipo de Ação	Parâmetros		Significado
	Nome	Tipo	
`SHUTDOWN`	–	–	Desligar o equipamento.
`WAKEUP`	–	–	Ligar o equipamento.
`SET_POWER_STATE`	`component`	`String`	Alterar o estado de consumo do componente `component` para `state`. O agente deve suportar, no mínimo, um componente (`"global"`) que representa o sistema inteiro.
	`state`	`String`	
`RUN_COMMAND`	`command`	`String`	Executar o comando `command`.
`SHOW_MESSAGE`	`message`	`String`	Mostrar a mensagem `message` para o(s) usuário(s).

Com exceção da ação `WAKEUP`, todas as outras ações correspondem, na prática, ao envio de uma mensagem SNMP a partir do gerente para alterar o valor de uma variável específica na MIB do agente. Em outras palavras, todas elas disparam uma "ação remota" correspondente, controlada pelo programa agente, nos seus respectivos alvos. No caso da ação `WAKEUP`, usada para ligar um equipamento remotamente (ou despertá-lo de um estado de baixo consumo em

que o sistema deixa de processar), o agente não estará executando e, portanto, a correspondência não existe.

Somente equipamentos que dispõem de uma placa de rede compatível com a tecnologia *"Wake On LAN"* podem ser ligados remotamente. Esse tipo de interface de rede é capaz de permanecer num estado de baixíssimo consumo de energia enquanto o restante do computador fica desligado, e de "acordar" o sistema a partir da recepção de uma mensagem especial (chamada originalmente de *"Magic Packet"* pela AMD) [ADV95].

As ações do tipo `SET_POWER_STATE` merecem destaque especial, pois são elas que garantem a adaptabilidade do mecanismo de execução de ações aos diversos padrões de gerência de energia existentes e a futuras especificações. A idéia é definir um único tipo flexível de ação para mudanças de estado de consumo, ao invés de um grande conjunto fixo de ações para alterar o estado de cada dispositivo (por exemplo, uma ação para desligar o monitor, outra para desligar o disco rígido, e assim por diante). Para que o processo funcione, a interpretação dos campos `component` e `state` desse tipo de ação fica inteiramente a cargo do agente. Isso significa que a concordância entre a implementação de um agente e a configuração da aplicação de gerência, no que diz respeito da semântica desses valores, é essencial para que a execução das ações de troca de estado seja bem-sucedida. Por exemplo, nada impede que se configure a aplicação de gerência para requisitar a um agente que altere o estado do componente `"display"` para `"abracadabra"`. Desde que o agente esteja preparado para interpretar este valor corretamente[1], a integridade lógica da operação é mantida. Solicitações de operações não suportadas serão tratadas pelo agente através do envio de uma notificação de operação inválida. Um conjunto de convenções para nomes de estados de consumo é apresentado na próxima seção.

A especificação completa da MIB para gerência de energia é apresentada como parte da especificação dos agentes, no capítulo 4.

4.3.3 Convenções sobre estados de consumo

Para que a aplicação de gerência seja capaz de usar de alguma "inteligência" ao decidir quais ações podem ser executadas sobre um equipamento, dado o seu estado momentâneo e o estado requisitado, é preciso haver um nível mínimo de concordância sobre a semântica dos campos da MIB que descrevem estados de consumo. Como fica a cargo de cada implementação interpretar o significado dos valores desses campos (definidos simplesmente como *strings* na MIB), é preciso definir algumas convenções.

A abordagem utilizada para definir o conjunto de estados que a aplicação de gerência é capaz de reconhecer consistiu em procurar pontos comuns entre os padrões APM e ACPI. A partir dessa análise, os estados previstos nas especificações desses padrões foram classificados em três grupos que correspondem aos principais níveis de consumo de energia:

- equipamento completamente ativo;
- equipamento em modo de baixo consumo;
- equipamento desligado.

Os estados pertencentes a cada grupo são listados na Tabela 4.3. Também foram adicionados alguns sinônimos que fazem parte da nomenclatura comum de gerência de energia.

[1] Executando um protetor de tela bem colorido, talvez.

Tabela 4.3 - Grupos de estados de consumo

Grupo	Padrão	Estados	Estado do agente
Equipamento completamente ativo	APM	"full on"	Responsivo
	ACPI	"working"	
	-	"on", "active", "idle"	
Equipamento em modo de baixo consumo	APM	"enabled", "standby", "hibernate", "suspend"	Dependendo do estado específico e da implementação, pode estar apto a responder ou não
	ACPI	"sleeping"	
	-	"sleep", "low power"	
Equipamento desligado	APM	"off"	Não responsivo
	ACPI	"soft off", "mechanical off"	

Basicamente, este critério de classificação de estados é utilizado para determinar se, dado o estado atual de um equipamento, o gerente deve ou não enviar a mensagem requisitando a troca para o novo estado. Por exemplo, não faz sentido requisitar uma mudança para um estado de consumo reduzido quando o equipamento estiver desligado.

A última coluna da tabela indica se o agente tem condições de responder quando o equipamento está em um determinado tipo de estado. Convencionou-se que, quando um equipamento estiver num estado de consumo reduzido, o gerente não lhe enviará requisições, por dois motivos: não se pode garantir que o agente estará operacional nessa situação, e o envio da mensagem poderia atrapalhar a gerência de energia local da máquina (ver seção 2.2).

Para tornar o sistema o mais adaptável possível, estes conjuntos de estados também podem ser customizados através do arquivo de configuração da aplicação de gerência. Sempre que possível, o gerente usa essa informação para determinar as operações possíveis ou necessárias sobre um equipamento.

4.3.4 Atualização do mapa da rede

Como qualquer aplicação de gerência de rede, o sistema de gerência centralizada de energia precisa manter um mapa atualizado dos equipamentos monitorados/gerenciados. Esse mapa instrumenta a tomada de decisão do sistema e também pode ser representado graficamente para permitir a visualização do estado corrente da rede pelo administrador.

A atualização do mapa da rede é feita a partir da recepção de notificações enviadas pelos agentes. Quando uma mudança de estado (entrada em modo de baixo consumo, desligamento, etc.) está prestes a ocorrer num determinado equipamento, independente dela ter sido iniciada remotamente (pelo gerente) ou localmente (por ação do usuário ou do sistema operacional da máquina), é responsabilidade do agente enviar uma notificação. Da mesma forma, caso ocorra um erro durante a execução de uma ação remota, o agente informa o gerente através de uma notificação.

Esse modelo básico não cobre os casos em que um equipamento é abruptamente desligado ou um agente deixa de ser executado (por falha do *software*, intervenção do usuário, etc.). Nesses casos, nenhum aviso é enviado ao gerente. Por este motivo, os agentes dos

equipamentos ativos devem enviar mensagens periódicas do tipo *"I'm alive"* [1] ao gerente para mantê-lo atualizado.

No agente, a recepção de uma requisição e o subseqüente envio de uma notificação indicando que a requisição foi executada são duas operações completamente assíncronas. Isto acontece porque o agente não envia uma notificação até que ele próprio tenha sido notificado pelo sistema operacional de uma alteração no estado do equipamento (Figura 4.3).

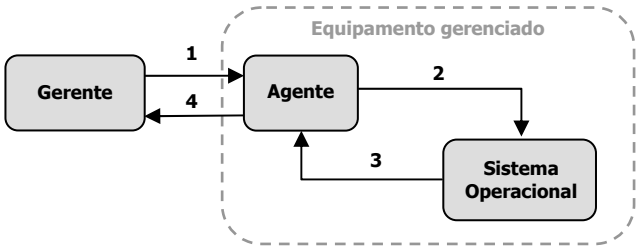


Figura 4.3 - Seqüência de passos na comunicação agente-gerente

Não se pode especular sobre o tempo que uma ação leva para ser executada por um equipamento (tempo entre os passos **2** e **3**, na figura), portanto o gerente não espera receber, dentro de um certo limite de tempo, notificações correspondentes às ações que disparou. Em vez disso, ele utiliza as mensagens de indicação de erro e a ausência de mensagens *"I'm alive"* para determinar que o estado do equipamento é desconhecido, caso as mensagens indicativas de mudança de estado não sejam recebidas.

Maiores detalhes do mecanismo de notificação são apresentados na especificação dos agentes (seções 5.3.2 e 5.3.3) e na descrição da implementação do gerente (seção 6.6).

4.3.5 Organização dos equipamentos em grupos lógicos

Para simplificar a configuração do sistema, os equipamentos da rede podem ser agrupados logicamente, segundo um critério qualquer. Deste modo, ao especificar as ações que devem ser executadas em resposta a um evento, pode-se definir como alvo não apenas um equipamento individual, mas um grupo de equipamentos com características comuns. Pode ser particularmente útil definir grupos de equipamentos que suportam o mesmo padrão de gerência de energia, por exemplo.

Entretanto, o uso de grupos na configuração do sistema deve ser feito com cuidado, pois depende exclusivamente do administrador garantir que todos os equipamentos reunidos em um grupo sejam capazes de executar o mesmo conjunto de ações. Em outras palavras, todos os equipamentos de um determinado grupo devem executar o mesmo tipo de agente, ou diferentes tipos de agentes que suportem o mesmo conjunto de ações.

Os membros de um grupo podem ser equipamentos ou outros grupos, como mostra o exemplo da Figura 4.4. Neste exemplo, o critério de agrupamento é o padrão de gerência de energia suportado pelos equipamentos. Assumindo que todo computador compatível com o padrão ACPI suporta um estado global chamado `"sleeping S1"`, por exemplo, uma única ação poderia ser configurada para todos os membros do grupo com o objetivo de colocá-los nesse estado.

[1] Esse tipo de mensagem também é chamado de *"heartbeat"* na literatura.

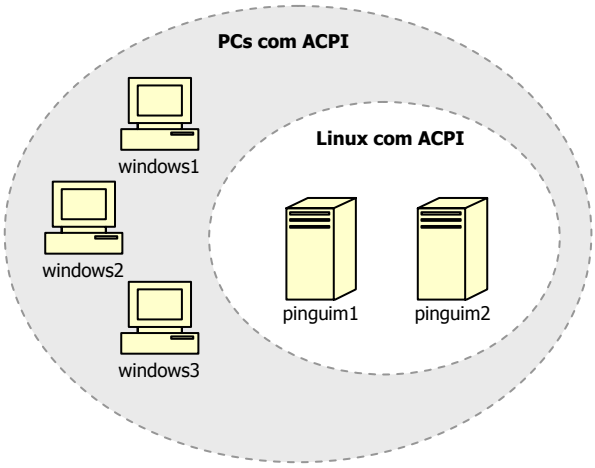


Figura 4.4 - Exemplo de organização de equipamentos em grupos

4.3.6 Interferência na operação dos equipamentos

Há uma diferença básica entre a gerência de energia realizada localmente em cada computador e o processo remoto de gerência de energia. Localmente, as requisições para que o computador entre num estado de baixo consumo são sempre avalizadas pelo usuário, quer explicitamente, através do pressionamento de um botão, quer implicitamente, através de sua inatividade prolongada.

Quando requisições para troca de estado são enviadas a partir do gerente do sistema, entretanto, essa permissão não existe. Por isso, caso a máquina esteja em atividade, é importante que o agente alerte o usuário da troca de estado iminente, o que pode ser feito através de uma janela de diálogo, por exemplo (ver exemplos na seção 5.3.4). Em situações deste tipo, a ação só seria executada se o usuário permitisse. Nos casos em que a máquina está ociosa, tem-se a confirmação implícita do usuário. Em suma, a operação de um computador sob o ponto de vista do usuário não é afetada pelo sistema de gerência.

Cabe notar que tanto o padrão APM quanto o padrão ACPI especificam mecanismos para que o sistema operacional notifique as aplicações em execução de uma transição iminente e, a partir do consentimento ou rejeição de cada uma delas, decida se deve prosseguir. Partindo desse pressuposto, e considerando o fato de que todas as requisições para trocas de estado são executadas, em última instância, pelo sistema operacional, o sistema de gerência centralizada não impõe nenhum risco de perda de dados maior do que se o equipamento fosse controlado apenas localmente.

4.4 Modelo de falhas

São previstas falhas nos seguintes elementos do sistema[1]:

- **nodos da rede:** os equipamentos podem apresentar falhas de *crash* (colapso) e recuperar-se assincronamente[2]. Em caso de falha do *host* do gerente, a rede deixa de ser gerenciada (sob o aspecto do consumo de energia) até que ele se recupere. Falhas de *crash* nos *hosts* dos agentes não têm nenhuma implicação sobre o processo de gerência, apenas fazem com que o gerente assuma que o estado do nodo em questão é

[1] Os tipos de falhas mencionados estão de acordo com a nomenclatura proposta por Cristian [CRI91]

[2] Um *crash* seguido de recuperação pode corresponder simplesmente a um reinício autônomo da máquina (*reboot*).

desconhecido. Não se assume sincronização de relógio entre os nodos. Assume-se a não ocorrência de falhas irrecuperáveis do armazenamento não-volátil dos nodos. Assume-se que, mesmo na ocorrência de *crash* de um nodo, o conteúdo do armazenamento não-volátil pode ser recuperado;

- **processos:** tanto a aplicação de gerência quanto os agentes do sistema funcionam como *daemons*[1] nos seus respectivos nodos. Assume-se que um *daemon* não pode falhar independentemente de seu *host*, ou seja, assume-se que, se um *host* está funcionando, o(s) processo(s) que residem nele devem estar funcionando também. São previstas falhas de omissão e de atraso no envio de sinais de vida pelos agentes, detectadas através de *timeouts* (ver seção 4.3.4);
- ***links* de comunicação:** assume-se que os *links* de comunicação da rede podem falhar e recuperar-se arbitrariamente. Em caso de particionamento da rede em decorrência de falha, a partição que contiver o nodo gerente continuará sendo monitorada e controlada; os nodos das partições inacessíveis ao *host* do gerente terão seu estado declarado desconhecido após o limite de tempo estabelecido. Estes nodos tampouco receberão os comandos de desligamento (ou quaisquer outros comandos) e poderão ser abruptamente desligados em caso de queda no fornecimento de energia.

Cabe uma nota a respeito da capacidade do sistema de garantir a proteção de dados nos equipamentos na iminência de um corte de energia, em função da carga da rede. Pode-se estabelecer uma relação intuitiva entre o desempenho geral de uma rede local e a probabilidade de falha na entrega das mensagens enviadas do gerente para os agentes comandando o seu desligamento. A Figura 4.5 mostra curvas de desempenho de diferentes tecnologias de redes locais[2], obtidas através de estudo analítico [MOU86].

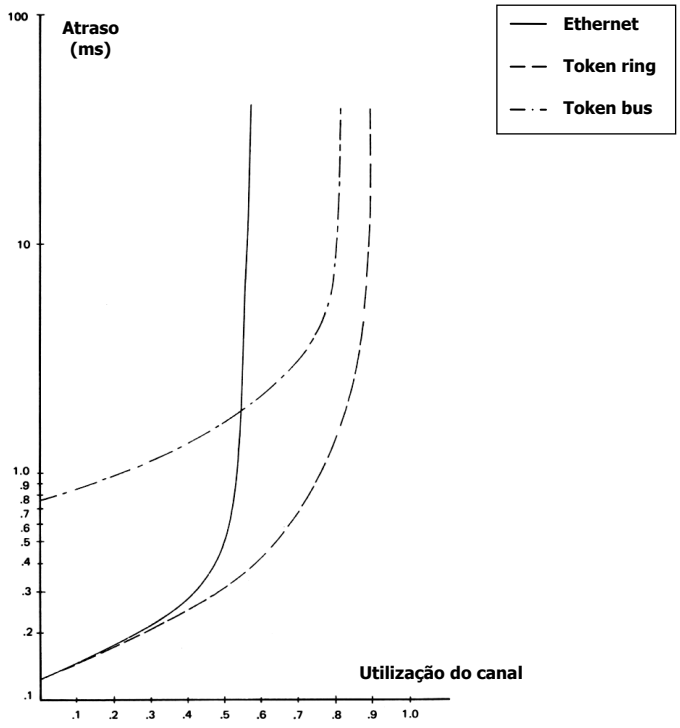


Figura 4.5 - Desempenho de diferentes tecnologias de redes locais

[1] Um *daemon* é um programa que não é invocado explicitamente, mas é carregado automaticamente pelo sistema operacional e permanece "dormente" (residente em memória), aguardando a ocorrência de condições a fim de executar tarefas.

[2] As curvas supõem tráfego homogêneo, capacidade máxima de 10 Mbit/s e um total de 100 estações compartilhando o meio.

Neste gráfico, o desempenho é medido através do atraso médio na transmissão dos pacotes em função da utilização do canal (vazão normalizada pela capacidade de transmissão do meio). Embora este gráfico apresente resultados específicos para as características da rede utilizadas na solução do modelo analítico, pode-se observar um comportamento semelhante das curvas variando os parâmetros ou até utilizando técnicas alternativas de avaliação do desempenho, como simulação, por exemplo. As curvas apresentam, invariavelmente, o mesmo formato, observando-se apenas uma mudança no posicionamento da assíntota que delimita o desempenho máximo da rede naquelas condições. Em outras palavras, observa-se um aumento ou diminuição da faixa de melhor desempenho dos protocolos de acesso em função da carga da rede. Para o gráfico apresentado, por exemplo, observa-se que o desempenho da rede Ethernet deteriora-se acentuadamente a partir de um fator de utilização de 0,4 (para aquelas características da rede em particular)[1].

Como a perda de pacotes num protocolo de transporte não-confiável (p. ex. UDP, que é o protocolo normalmente utilizado para transmitir mensagens SNMP) está diretamente relacionada com o atraso no nível de enlace, pode-se deduzir, a partir das curvas do gráfico, que a probabilidade de falha na transmissão de uma mensagem SNMP aumenta conforme a rede local atinge um nível de saturação. Ou seja, quanto maior o fator de utilização do canal, maior é a probabilidade de que o sistema de gerência não consiga requisitar o desligamento dos computadores antes do corte de energia. Intuitivamente, pode-se inferir que, dentro da faixa de utilização aceitável para a operação da rede (entre 0 e 30 a 40% de utilização, no exemplo do gráfico), o gerente terá condições de notificar todas as estações com sucesso. É importante frisar que, em condições normais de funcionamento, uma rede local dificilmente ultrapassa esses níveis de utilização, de modo que a probabilidade do gerente não conseguir enviar as mensagens é baixíssima. Vale lembrar, ainda, que as implementações do protocolo SNMP possuem alguns parâmetros de configuração que permitem ajustar o seu funcionamento ao desempenho da rede, entre os quais o tempo máximo para entrega de mensagens e o número de retransmissões de cada mensagem.

Finalmente, é importante salientar que, normalmente, os *no-breaks* também podem ser configurados de maneira a ajustar a antecedência com que os alarmes sobre o nível de carga das baterias devem ser enviados. Assim, é possível reduzir a probabilidade de que os agentes não sejam notificados a um valor extremamente baixo.

4.5 Modelo de segurança

O modelo de segurança considera aspectos da comunicação entre os componentes do sistema e possíveis tentativas de operação não-autorizada, como é explicado em detalhe a seguir.

4.5.1 Comunicação entre a aplicação de gerência e os no-breaks

Conforme enfatizado anteriormente, o objetivo do sistema no que se refere aos *no-breaks* consiste exclusivamente no seu monitoramento para efeito de detecção de alterações no

[1] A maioria das avaliações de desempenho do protocolo de acesso ao meio físico usados em redes Ethernet (CSMA/CD) é baseada em modelos analíticos ou simulações. Praticamente todas elas concluem que a utilização máxima do canal nesse tipo de protocolo fica muito abaixo da capacidade nominal da rede. Shoch e Hupp [SHO84], entretanto, contrariam firmemente os limites usualmente apontados. Com base em medições realizadas numa instalação real, eles afirmam que a utilização máxima do canal chega a mais de 95%, mesmo em situações de carga extrema. Esses resultados são corroborados por Boggs et al. [BOG88]. De qualquer modo, o barateamento do *hardware* para redes locais de alta velocidade (p. ex. placas Fast Ethernet, que operam a 100 Mbps) faz com que virtualmente todas as LANs sejam fartamente superdimensionadas, oferecendo capacidade mais do que suficiente para qualquer tipo de aplicação.

fornecimento de energia. Em outras palavras, a aplicação de gerência não executa sobre o *no-break* nenhum tipo de ação com a finalidade de controle. Sendo assim, não é necessário tomar nenhuma medida adicional de segurança para realizar a comunicação com os aparelhos. Além disso, os *no-breaks* capazes de se comunicar através do protocolo SNMP suportam, tipicamente, a versão 1 do protocolo (SNMPv1), que não oferece recursos de autenticação ou criptografia, mesmo nos casos em que a alteração de parâmetros do *no-break* é permitida.

4.5.2 Comunicação entre a aplicação de gerência e os agentes

Idealmente, o acesso ao controle de qualquer equipamento de uma rede por uma entidade externa (seja ela um usuário ou uma aplicação) deve ser restringido através de um conjunto de permissões, que são tipicamente associadas a uma "conta" dependente de autenticação. No caso específico da aplicação de gerência de energia, a execução de operações por uma entidade não autorizada pode implicar na indisponibilidade de serviços para os usuários e demais equipamentos da rede. Como o sistema permite o desligamento completo das máquinas gerenciadas, uma operação não autorizada poderia, por exemplo, interromper o funcionamento de um servidor e afetar indiretamente todos os equipamentos que dependem dos seus serviços.

Assim, prevê-se a utilização da versão 3 do protocolo SNMP [CAS99], que disponibiliza serviços de autenticação e sigilo (*confidentiality*), para o envio de mensagens de controle do gerente para os agentes do sistema. No sentido inverso (dos agentes para o gerente), a comunicação pode ser feita através de versões anteriores do protocolo, pois não há impacto significativo no recebimento de notificações falsas (a única conseqüência será a percepção incorreta do estado da rede). A comunicação entre o gerente e os agentes é ilustrada na Figura 4.6.

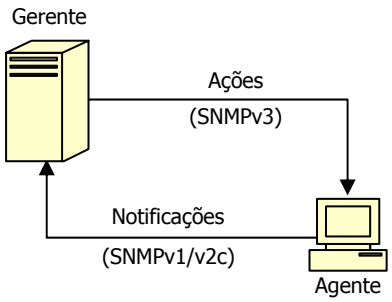


Figura 4.6 - Comunicação entre gerente e agentes

4.5.3 Acesso externo não-autorizado

Assume-se a execução do sistema em uma rede local (LAN), protegida de acessos externos não autorizados por *firewall*. Assume-se que, dentro da rede local, não haverá tentativas de *spoofing*[1].

O único ponto de acesso previsto para requisições geradas fora da rede local é a interface gráfica da aplicação de gerência, que pode ser implementada como um componente independente da aplicação em si. Uma solução para a interface gráfica, construída como uma aplicação *Web*, foi apresentada anteriormente [POL2000]. Naquela implementação, a segurança foi garantida através da combinação de um servidor HTTP seguro (HTTP sobre SSL) com um

[1] Interceptação, alteração e retransmissão de mensagens da rede de maneira a enganar o receptor quanto à veracidade dos dados e à identidade do transmissor.

mecanismo de autenticação interno, numa configuração semelhante à encontrada nos *sites* de comércio eletrônico da Internet.

A Figura 4.7 ilustra esse modelo de acesso. A comunicação entre a interface gráfica e a aplicação de gerência é feita, em última instância, através do protocolo de invocação remota de métodos da plataforma Java (*Remote Method Invocation* ou RMI). Isso permitiria ao cliente que executa a interface gráfica estar localizado tanto dentro da rede local quanto fora dela; por isso, prevê-se a utilização de *firewall* de maneira a bloquear tentativas de acesso externo via RMI, garantindo que o uso deste protocolo seja confinado aos limites da LAN.

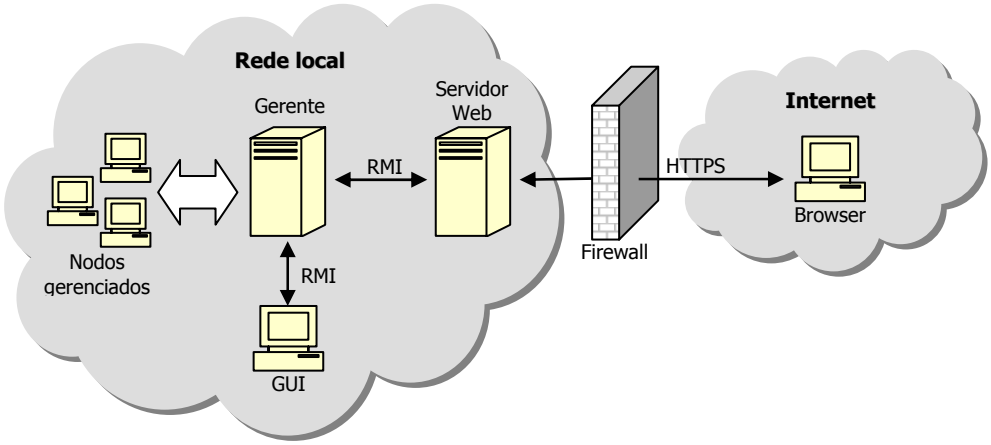


Figura 4.7 - Acesso externo ao sistema através de *firewall*

Este modelo também evita possíveis ataques de DoS[1] originados na rede externa, pois as requisições passam por um elemento intermediário (na Figura 4.7, o servidor *Web*) que está na posição adequada para filtrar as mensagens.

4.5.4 Configuração local de gerência de energia

Finalmente, recomenda-se que o programa de configuração do BIOS seja protegido por senha nos computadores gerenciados para evitar que os usuários desabilitem o controle de energia, especialmente em equipamentos que suportam apenas o padrão APM. Em *hardware* que suporta o padrão ACPI, a desabilitação dos recursos no BIOS teoricamente não impede a gerência de energia efetuada pelo sistema operacional (o sistema operacional tem controle direto também sobre o *firmware* compatível com o padrão).

[1] *Denial of Service*, ataque a um servidor através de sobrecarga de tráfego ou requisições.

5 Especificação dos agentes

O agente é a entidade que detém o conhecimento específico sobre as características de consumo de eletricidade de um equipamento que se deseja gerenciar. Para ter acesso a essas características e tomar conhecimento de alterações no consumo de energia que ocorram no equipamento, um agente contém, necessariamente, uma parte dependente de plataforma, escrita em linguagem nativa (como C ou C++), que é responsável por realizar a comunicação com o sistema operacional ou níveis de *software* ainda mais baixos.

Basicamente, o agente possibilita à aplicação de gerência monitorar remotamente os eventos de consumo de energia de um equipamento, bem como atuar sobre o equipamento para alterar seu estado. No primeiro caso, o agente envia notificações que indicam alguma mudança no estado do equipamento, como, por exemplo: "o computador entrará em modo de espera", "o monitor será desligado", etc. No segundo, permite ao gerente alterar, através de uma requisição, o estado global do equipamento (p. ex. "encerre o sistema operacional") ou ainda o estado de cada um de seus componentes, individualmente (p. ex. "desligue o monitor", "desligue o disco rígido", etc.).

O restante deste capítulo apresenta um conjunto de diretrizes para a implementação de agentes para o sistema de gerência de energia proposto neste trabalho.

5.1 Funções do agente

Um agente deve ser capaz de desempenhar as seguintes funções:

- ser carregado automaticamente pelo sistema operacional a cada *"boot"* da máquina;
- prover informações básicas sobre os recursos de gerência de energia do equipamento;
- alterar o estado de consumo do equipamento e/ou de seus dispositivos internos e periféricos;
- notificar a aplicação de gerência sobre alterações no estado de consumo de energia do equipamento (ligação, desligamento, entrada em modo de baixo consumo e mudança no estado de consumo de um dispositivo);
- disparar a execução de comandos no sistema operacional;
- mostrar algum tipo de notificação visual para o(s) usuário(s) do equipamento.

A partir da definição destas funções básicas, a base de informações (MIB) usada para controlar o comportamento do agente foi construída. Ela é descrita em detalhe a seguir.

5.2 Base de informações (MIB)

Para que a comunicação entre uma aplicação de gerência e um agente SNMP seja efetuada com sucesso, ambos devem estar de acordo sobre as informações e operações disponíveis. Isso é definido em uma base de informações de gerência, ou MIB. Uma MIB específica para a gerência de energia, chamada de *Power Management MIB*, foi definida como parte do trabalho e é explicada ao longo desta seção. A estrutura da MIB pode ser vista na Figura 5.1, a seguir.

Ao definir a base de informações, procurou-se levar em consideração as características dos padrões de gerência de energia em uso e os mecanismos utilizados para sua implementação. De modo resumido, pode-se dizer que estes padrões especificam uma série de estados de consumo nos quais um equipamento e, possivelmente, seus componentes internos, podem ser colocados para reduzir o consumo de eletricidade. Basicamente, tudo que é necessário para alterar o estado de consumo de um dispositivo é identificá-lo (através de um nome, por exemplo) e indicar qual o estado de consumo desejado. Isso normalmente é feito através de uma chamada do sistema operacional.

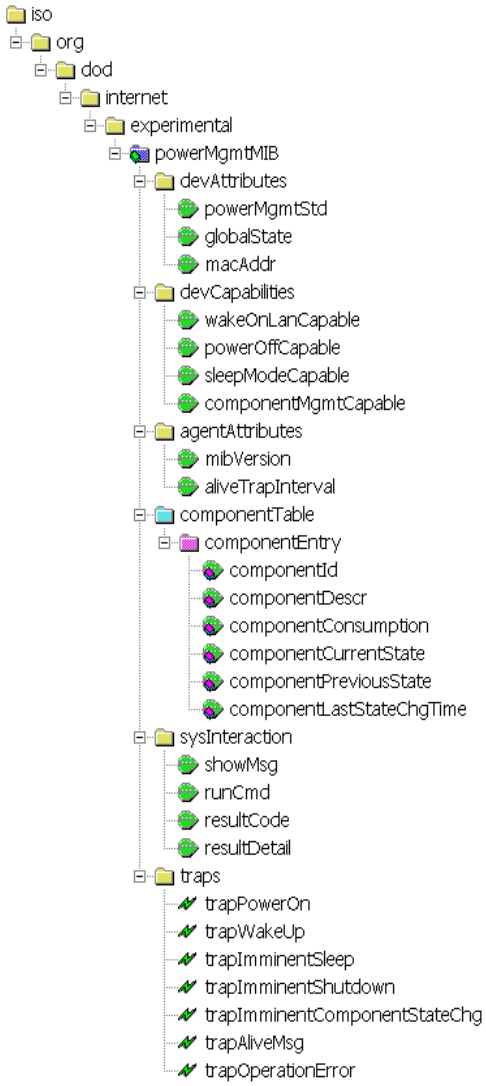


Figura 5.1 - Power Management MIB

Esta MIB é uma variação da MIB proposta na dissertação de Krolow [KRO2001]. Foram feitas algumas alterações na estrutura original, com a intenção de torná-la mais legível e funcional, priorizando o objetivo fundamental da MIB: capacitar o agente com funcionalidade suficiente para permitir a gerência de energia do equipamento, independente do padrão de gerência de energia que ele efetivamente suporta.

Como se pode observar na figura, a estrutura foi divida em cinco grupos de objetos e uma tabela, que serão explicados a seguir. O código-fonte completo da MIB pode ser encontrado no Apêndice C.

5.2.1 Convenções de nomenclatura

O termo *"device"* é utilizado, em inglês, com diferentes significados. Na área de informática, ele é geralmente encontrado com os significados de "equipamento" (um computador, por exemplo) e "dispositivo" (um disco rígido, um monitor de vídeo, etc.). Para evitar qualquer ambigüidade na terminologia usada ao longo deste trabalho, foram adotadas as seguintes convenções:

- a palavra *"device"* é usada no sentido de *"networked device"*, ou seja, um equipamento ligado à rede;
- a palavra *"component"* é usada no sentido de *"hardware component"* para descrever dispositivos de *hardware*, partes integrantes de um equipamento ou dispositivos periféricos.

5.2.2 Grupo "devAttributes"

Contém objetos que correspondem a atributos do equipamento gerenciado. Os atributos são listados na Tabela 5.1.

Tabela 5.1 - Objetos do grupo "devAttributes"

Objeto	Descrição	Acesso*
`powerMgmtStd`	Padrão de gerência de energia suportado pelo equipamento, incluindo número da versão (p. ex. `"APM 1.2"`, `"ACPI 1.0"`)	L
`globalState`	Estado de consumo global do equipamento (ver seção 5.2.5, a seguir)	L/E
`macAddr`	Endereço MAC[1] da interface de rede do equipamento	L

** L = Leitura; E = Escrita*

5.2.3 Grupo "devCapabilities"

Contém variáveis booleanas que descrevem algumas propriedades do equipamento quanto à gerência de energia. As variáveis são listadas na Tabela 5.2.

Tabela 5.2 - Objetos do grupo "devCapabilities"

Objeto	Descrição	Acesso
`wakeOnLanCapable`	Indica se o equipamento pode ser despertado através de um *"magic packet"* enviado para sua interface de rede	L
`powerOffCapable`	Indica se o equipamento é capaz de desligar sua própria fonte de eletricidade através de um comando de *software*	L
`sleepModeCapable`	Indica se o equipamento possui algum estado de "dormência" (por definição, um estado de baixo consumo entre "completamente ligado" e "completamente desligado")	L
`compMgmtCapable`	Indica se o equipamento é capaz de gerenciar o consumo de energia de seus componentes internos (disco rígido, processador, etc.) e/ou periféricos (p. ex. impressoras) individualmente	L

[1] Endereço de 48 bits usado para identificar a interface de rede no nível de enlace de dados. A camada MAC (*Medium Access Control*) é definida como parte do modelo de referência IEEE 802.

5.2.4 Grupo "agentAttributes"

Contém os atributos do agente SNMP, conforme a Tabela 5.3.

Tabela 5.3 - Objetos do grupo "agentAttributes"

Objeto	Descrição	Acesso
`mibVersion`	Versão da MIB que o agente implementa	L
`aliveTrapInterval`	Periodicidade, em minutos, com que o agente envia notificações do tipo *"I'm alive"* para o gerente. Recomenda-se que esse valor seja igual a 1	L/E

Pode-se notar que o endereço do gerente não faz parte dos atributos deste grupo, ou de qualquer outro grupo da MIB. O agente precisa conhecer o endereço do gerente para poder enviar as *traps* correspondentes aos diversos eventos. Não há um mecanismo padrão para configurar este endereço; a solução fica a cargo de cada implementação. Uma das alternativas possíveis é ler o endereço de um arquivo, outra é passá-lo para a aplicação através da linha de comando.

5.2.5 Tabela "componentTable"

Esta tabela contém entradas que correspondem aos dispositivos internos ou periféricos do equipamento que podem ser gerenciados quanto ao consumo de eletricidade. Possui no mínimo uma entrada que representa o sistema como um todo, cujo identificador deve ser, obrigatoriamente, `"global"`. Os elementos de cada entrada são listados na Tabela 5.4, a seguir.

Tabela 5.4 - Objetos da tabela "componentTable"

Objeto	Descrição	Acesso
`componentId`	Identificador do dispositivo, que deve ser único dentro do contexto de um equipamento	L
`componentDescr`	Descrição do componente	L
`componentConsumption`	Consumo de energia do componente no instante da requisição	L
`componentCurrentState`	Estado atual do componente. Quando anexado a notificações do tipo `trapImminentDevStateChg`, este objeto contém o estado em que o dispositivo está prestes a ser colocado. Quando o identificador do componente corresponde à palavra `"global"`, o mesmo valor presente neste objeto deve aparecer em `globalState`, do grupo `devAttributes` (seção 5.2.2). Quando alterado, o novo valor é interpretado pelo agente para determinar o estado em que o dispositivo deve ser colocado. Caso o valor não faça sentido, uma notificação do tipo `trapOperationError` é enviada	L/E
`componentPreviousState`	Estado do componente anterior à última mudança	L
`componentLastStateChgTime`	Data e hora da última mudança de estado do componente	L

Vale fazer dois breves comentários. Primeiro, o objeto `componentConsumption` não é fundamental para as funções de gerência de energia (sob esse ponto de vista, o agente não precisa necessariamente responder com um valor real). Segundo, quando se diz que o objeto

`componentCurrentState` "contém o estado em que o dispositivo está *prestes* a ser colocado", está-se referindo ao fato de que, logicamente, as notificações são enviadas imediatamente *antes* da troca de estado ocorrer. Deste modo, não é possível garantir com 100% de segurança que a troca de fato ocorreu, pois existe a possibilidade (ainda que pequena) de que, entre o envio da mensagem e a troca propriamente dita, um evento externo (como um movimento do *mouse*, por exemplo) tenha cancelado a operação original. Idealmente, este segundo evento deveria disparar o envio de uma nova notificação, que corrigiria a percepção incorreta do estado do dispositivo pelo gerente.

Obviamente, quando uma transição de estado é interrompida pela intervenção do usuário, o equipamento passa ao estado ativo. Assim, mesmo que essa segunda notificação não seja transmitida, o próximo sinal de vida enviado pelo agente (ver seção 5.3.3) corrigirá o problema.

5.2.6 Grupo "sysInteraction"

Os objetos deste grupo são utilizados para interação com o sistema operacional do equipamento e para armazenar códigos de erro. São listados na Tabela 5.5.

Tabela 5.5 - Objetos do grupo "sysInteraction"

Objeto	Descrição	Acesso
`showMsg`	Quando esta variável é alterada, seu conteúdo deve ser mostrado para o(s) usuário(s) do equipamento, através do recurso gráfico disponível (terminal de caracteres, sistema de janelas, etc.)	E
`runCmd`	Quando esta variável é alterada, o agente deve tentar executar o comando ou *script* especificado, através de uma requisição ao sistema operacional	E
`resultCode`	Código de resultado da última operação realizada sobre o sistema operacional. Por convenção, zero é usado para indicar sucesso e qualquer valor diferente de zero é usado para indicar erro	L
`resultDetail`	Descrição textual do resultado da última operação executada sobre o sistema operacional.	L

5.2.7 Grupo "traps"

Contém as notificações utilizadas pelo agente para avisar ao gerente da ocorrência de eventos significativos. Estas notificações são listadas na Tabela 5.6.

Tabela 5.6 - Objetos do grupo “traps”

Objeto	Descrição	Variáveis anexas
`trapPowerOn`	Enviada quando o equipamento é ligado ou reiniciado, ou quando o agente é recarregado	-
`trapWakeUp`	Enviada quando o equipamento “desperta” de um estado de baixo consumo.	-
`trapImminentSleep`	Enviada quando o equipamento está prestes a entrar num modo de baixo consumo (*sleep mode*)	`globalState`, para indicar o nome do estado de baixo consumo
`trapImminentShutdown`	Enviada quando o sistema operacional está prestes a ser encerrado	-
`trapImminentDevStateChg`	Enviada quando o estado de consumo de um dos componentes do equipamento está prestes a ser alterado	`componentId`, `componentCurrentState` e `componentPreviousState`, para indicar o novo estado e o estado anterior do componente
`trapAliveMsg`	Enviada periodicamente (a cada `aliveTrapInterval` minutos) para indicar ao gerente que o equipamento está ativo	`globalState`, para indicar o estado atual do equipamento (`“on”`, `“idle”`, `“active”`, etc.)
`trapOperationError`	Enviada para notificar o gerente da ocorrência de um erro durante a execução de uma operação	`resultCode` e `resultDetail`, para indicar o tipo de erro ocorrido

5.2.8 Nota sobre identificadores de estados de consumo

Não há restrição sintática ou semântica quanto aos nomes de estados que um agente pode suportar. Os objetos da MIB que podem conter nomes de estados são associados ao tipo `DisplayString` (uma cadeia de caracteres, com até 256 posições). Este tipo de dados foi escolhido para que a MIB fosse compatível com diversos padrões de gerência. Os padrões existentes identificam os diferentes estados de consumo através de um nome (p. ex. `“on”`, `“sleeping”`, `“off”`), portanto este tipo de dados é adequado, bastando que as implementações sejam programadas para interpretar os valores corretamente.

É imprescindível que, ao implementar um agente, sejam especificados quais dispositivos compõem o equipamento (i.e. quais identificadores estarão presentes na tabela `componentTable`) e quais os estados que cada dispositivo suporta, para que o administrador possa configurar a aplicação de gerência de acordo (ver seção 6.9).

Adicionalmente, para cada componente da tabela `componentTable`, o agente deve admitir, pelo menos, os valores `“on”` e `“off”`. Se o componente suportar um ou mais estados de baixo consumo, sugere-se que um estado `“sleep”` também seja admitido. O agente deve ser capaz de interpretar estes valores em adição a quaisquer outros valores efetivamente associados ao padrão de gerência de energia do equipamento.

Na aplicação de gerência, a interpretação dos identificadores de estado é feita segundo um conjunto flexível de convenções, conforme especificado na seção 4.3.3.

5.2.9 Nota sobre possíveis usos da MIB

É interessante apontar que, além de definir operações para gerência de energia, a MIB pode servir naturalmente como um repositório de informações sobre o comportamento do equipamento no que se refere ao consumo de eletricidade. Um sistema interessado, por exemplo, em coletar dados estatísticos sobre o consumo de energia, poderia fazê-lo através de consultas periódicas ao agente. Este tipo de coleta, no entanto, foge ao escopo da aplicação desenvolvida neste trabalho. Aqui, o interesse se restringe a utilizar o agente como um veículo para a execução das tarefas essenciais à gerência de energia.

5.3 Funcionamento do agente

Basicamente, a operação de um agente se resume às seguintes tarefas:

- tratar mensagens do sistema operacional que indicam alterações no estado de consumo de energia do equipamento (p. ex. "entrada no modo *sleep*") ou de seus componentes (p. ex. "disco rígido desligado"), avisando o gerente através da notificação (*trap* SNMP) correspondente;
- fazer a operação inversa, ou seja, receber uma mensagem via SNMP do gerente requisitando uma alteração no estado de consumo do equipamento e efetuá-la através de uma chamada de sistema.

Os fluxos básicos de eventos entre o gerente e o agente, e entre o agente e o sistema operacional, podem ser observados na Figura 5.2.

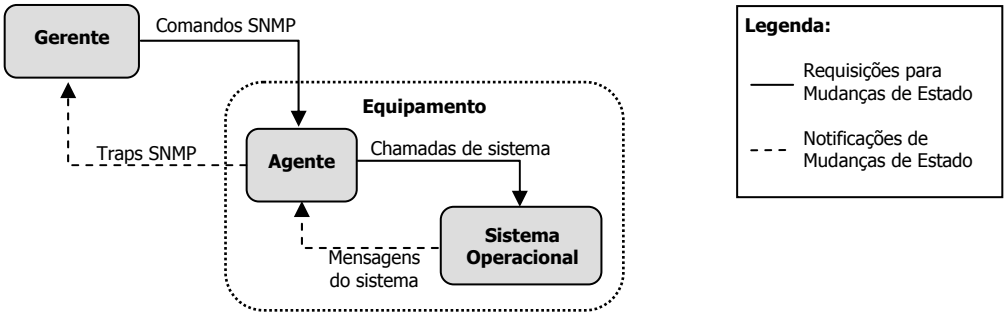


Figura 5.2 - Funcionamento básico do agente

É importante notar que nem todos os sistemas operacionais dispõem de um mecanismo de notificação das aplicações através de mensagens. Entre os sistemas para PCs, os da família Windows, da Microsoft, são capazes de enviar mensagens às aplicações; os sistemas Linux não têm um mecanismo semelhante, mas as aplicações podem utilizar soluções alternativas para descobrir alterações de estado no sistema (tipicamente, essas ações incluem o monitoramento de um arquivo especial no diretório `/proc`). O que realmente importa é que o agente consiga detectar as alterações no consumo de energia do computador, independente dos detalhes de implementação envolvidos nessa operação.

5.3.1 Relação com o sistema operacional

Inicialmente implementadas como parte do *firmware* dos computadores, as funções de gerência de energia vêm sendo gradativamente incorporadas ao nível do sistema operacional. Como foi explicado anteriormente, essa é uma transição natural, pois o sistema operacional tem melhores condições de determinar a carga do sistema e detectar quando ele está ocioso.

No padrão ACPI, por exemplo, o sistema operacional funciona como peça central da arquitetura de gerência. No entanto, como foi mencionado anteriormente, o sistema operacional precisa da colaboração dos outros níveis para tomar decisões. Particularmente no que se refere ao estado das aplicações, elas próprias são a fonte definitiva de informações. Para interagir com as aplicações durante a gerência de energia, dois recursos são fundamentais em um sistema operacional: primeiro, um mecanismo através do qual as aplicações possam ser notificadas de alterações no estado de energização do sistema (como discutido na seção anterior); segundo, as aplicações devem poder indicar ao sistema operacional, explicitamente, quais recursos do computador precisam utilizar.

Adicionalmente, o sistema operacional pode dispor de um conjunto de funções que permitam inquirir sobre o estado de consumo dos componentes do equipamento, e até mesmo modificá-lo. Neste caso, a gerência de energia pode não só ser efetuada com a participação das aplicações, como ser efetuada *pelas* aplicações. As versões mais recentes dos sistemas Windows possuem todos estes recursos. Já no sistema Linux, os mecanismos de gerência de energia estão em estágio de desenvolvimento [PEN2002, ACP2002].

Desde que o sistema disponha desses três tipos de recursos, um agente tem plenas condições de monitorar e comandar a gerência de energia de um equipamento.

5.3.2 Envio de notificações

As notificações que o agente pode enviar ao gerente foram definidas na seção 5.2.7. Com exceção de `trapOperationError` e `trapAliveMsg`, todas as outras servem para avisar o gerente de uma mudança no estado de consumo do equipamento (ou de seus componentes).

O envio dessas notificações é disparado sempre em decorrência da detecção, pelo agente, de uma alteração no estado do computador, sinalizada pelo sistema operacional. Isso pode acontecer em duas situações: em resposta à ação do próprio agente, quando ele efetua uma chamada ao sistema operacional para efetuar a troca de estado, ou quando o sistema operacional (ou uma aplicação) realiza a troca, independentemente do controle do agente. De qualquer maneira, o gerente sempre é notificado das mudanças no estado do equipamento, não importando como elas tenham sido iniciadas.

Como se pode constatar observando a Tabela 5.6, cada notificação tem um significado bastante específico. É importante que um agente seja fiel à semântica definida nessa tabela para que o gerente interprete as mensagens corretamente. Por exemplo, as notificações `trapPowerOn`, `trapWakeUp` e `trapAliveMsg`, que talvez pareçam ter o mesmo efeito (a detecção de que o equipamento está ativo), são interpretadas de maneiras distintas pelo gerente (conforme descrito na seção 6.6.2).

Preferencialmente, o agente não deve enviar notificações do tipo `trapComponentStateChg` para o componente `"global"`. Neste caso, uma das notificações que indicam alterações no estado do equipamento inteiro deve ser utilizada (`trapPowerOn`, `trapWakeUp`, `trapImminentSleep` ou `trapImminentShutdown`).

5.3.3 Mensagens "I'm alive" (sinais de vida)

Uma notificação que tem uma função ligeiramente diferente das demais é `trapAliveMsg`. Conforme explicado acima, o gerente utiliza as notificações de troca de estado para atualizar o mapa da rede. Entretanto, se um equipamento for abruptamente desligado (por uma queda no fornecimento de energia, por exemplo) ou seu agente deixar de ser executado por qualquer motivo (falha do *software*, intervenção do usuário, etc.), nenhum aviso será enviado.

Por isso, os agentes dos equipamentos ativos devem enviar mensagens periódicas ao gerente para mantê-lo atualizado. Essas mensagens são do tipo `trapAliveMsg`.

O gerente dispara um temporizador toda vez que o estado do equipamento muda para um estado ativo (segundo as convenções definidas na seção 4.3.3), e assume que o estado é desconhecido se o próximo sinal de vida (ou outra notificação de mudança de estado) não for recebido dentro do limite de tempo.

Obviamente, os agentes de equipamentos que se encontram em estados globais de baixo consumo não enviam esse "sinal de vida".

5.3.4 Avisos ao usuário

Embora o objetivo do sistema seja primariamente a redução do consumo de energia, não se pode simplesmente desconsiderar as necessidades dos usuários. Afinal, nos períodos em que a rede é abastecida por um banco de baterias através de um *no-break*, maximizar o tempo de autonomia significa, também, dar uma chance aos usuários para terminarem suas tarefas essenciais. Mesmo a alteração do estado de um único dispositivo pode inviabilizar a operação do computador pelo usuário (desligar o monitor de vídeo, por exemplo). Do mesmo modo, diminuir a velocidade do processador pode reduzir o consumo e estender o tempo que o *no-break* consegue manter o computador ligado, mas aumenta o tempo que as tarefas levam para serem executadas [LOR98]. Por este motivo, sugere-se que o agente exiba alguma espécie de aviso ao usuário para que ele tenha conhecimento da alteração iminente e possa optar por efetuá-la ou não.

Um modelo para janelas de aviso pode ser observado na Figura 5.3. Neste exemplo, a janela mostraria uma contagem regressiva para o instante da mudança, e o usuário teria como alternativas proceder imediatamente ou cancelar a operação, o que dispararia a exibição da segunda janela, avisando das possíveis conseqüências dessa ação (como desligamento abrupto do sistema, por exemplo).

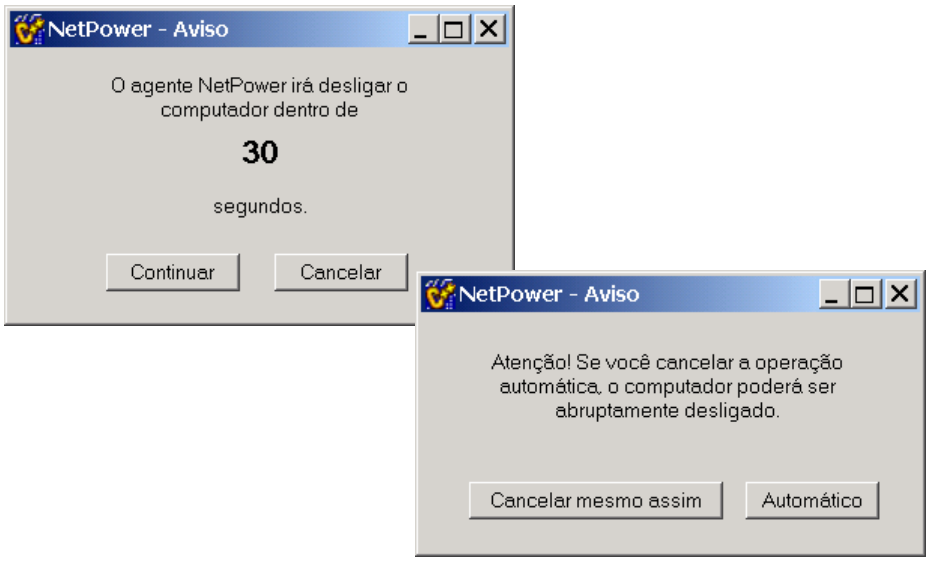


Figura 5.3 - Modelos de janelas de aviso

É importante apontar que, utilizando um temporizador, conforme se sugere acima, a troca de estado automática funciona normalmente nos períodos de ociosidade, quando não há usuários utilizando a máquina. Ao encerrar a contagem, o agente procede com a operação.

Quando há usuários operando o computador, o fato de dar-lhes o poder de decisão sobre as trocas de estado iminentes não deve interferir significativamente nos resultados pretendidos pelo sistema. Principalmente nos casos em que o objetivo é a proteção de dados (na iminência de corte de energia, por exemplo), o mais importante é que o computador seja de fato desligado, e não o modo como o desligamento foi iniciado. Portanto, se o usuário cancela a operação automática para terminar uma tarefa urgente, mas desliga o equipamento depois de executá-la, o efeito desejado ainda é obtido. Obviamente, isso depende da (boa) intenção do próprio usuário e de seu nível de esclarecimento quanto às possíveis conseqüências do desligamento abrupto da máquina.

5.3.5 Nota sobre agentes do tipo *"proxy"*

Agentes do tipo *proxy* funcionam como intermediários entre a aplicação de gerência e um dispositivo eletrônico qualquer que não dispõe dos recursos de *hardware* ou de *software* necessários para a comunicação direta com o gerente. Por exemplo, um equipamento cuja única interface de comunicação é uma porta serial, mas que suporta funções de gerência de energia (mesmo que sejam tão básicas quanto “ligar” e “desligar”), pode ser gerenciado pelo sistema se for conectado a um agente *proxy* capaz de se comunicar com ele através desta porta serial. Apesar de não dependerem do sistema operacional como agentes normais no que diz respeito às funções de gerência de energia, *proxies* também são escritos especificamente para um certo tipo de *hardware* e *software* (os do equipamento eletrônico em questão).

6 Implementação do gerente

Conforme se mencionou no primeiro capítulo, o foco da parte prática deste trabalho está na implementação do gerente. Neste capítulo, são apresentados detalhes sobre esta implementação. A não ser por uma pequena biblioteca usada para o envio de requisições ARP, implementada em C++ (ver seção 6.5.4), a aplicação foi completamente escrita em Java. Além dos pacotes de classes que fazem parte do SDK[1] da Sun Microsystems, algumas outras bibliotecas gratuitas foram utilizadas e serão identificadas oportunamente.

6.1 Funções e composição do gerente

O gerente é responsável pela execução das seguintes funções básicas[2]:

- monitorar eventos relativos ao fornecimento de energia reportados pelos *no-breaks*;
- disparar eventos programados nos horários pré-determinados;
- em resposta à ocorrência de eventos, disparar a execução de ações nos equipamentos da rede de acordo com a configuração;
- monitorar eventos dos agentes, atualizando o mapa da rede.

Cada uma dessas tarefas é delegada pelo gerente a um componente especificamente projetado para executá-la. Estes componentes são ilustrados na Figura 6.1, e o funcionamento de cada um deles é descrito em detalhe ao longo deste capítulo.

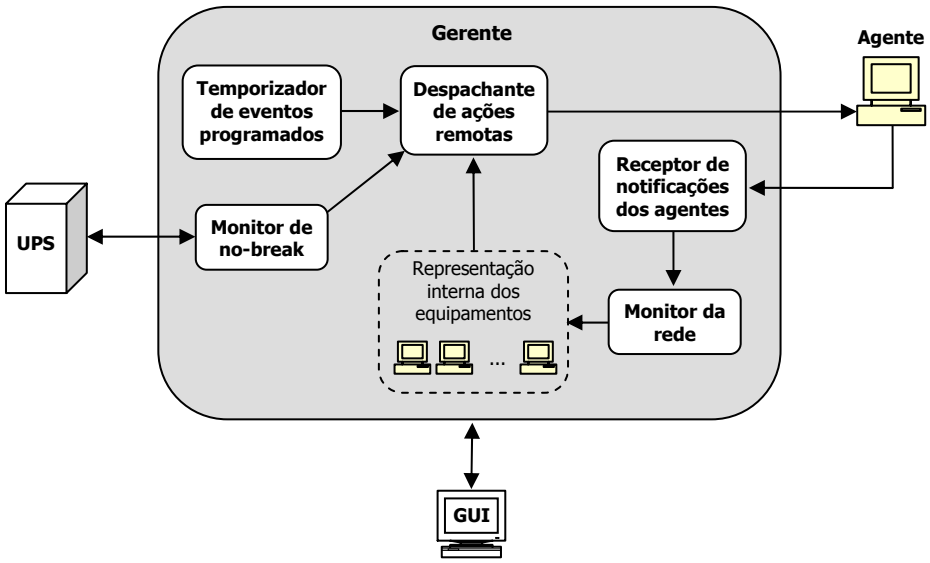


Figura 6.1 - Composição do gerente

6.2 Hierarquia de classes

A aplicação de gerência é construída a partir de uma série de pequenos componentes, que são definidos nos 7 pacotes de classes listados e descritos na Tabela 6.1.

---

[1] *Software Development Kit*, "*kit* de desenvolvimento de *software*".

[2] Para uma lista completa dos requisitos do gerente, consultar o Apêndice A.

Tabela 6.1 - Pacotes da implementação

Pacote	Descrição
`br.ufrgs.inf.netpower.auth`	Contém as classes relacionadas com a autenticação de usuários
`br.ufrgs.inf.netpower.conf`	Contém as classes responsáveis pelo tratamento do arquivo de configuração e sua conversão para a representação interna
`br.ufrgs.inf.netpower.manager`	Contém os componentes propriamente ditos do gerente
`br.ufrgs.inf.netpower.manager.gui`	Contém as classes que compõem a interface gráfica para operação remota do gerente
`br.ufrgs.inf.netpower.pm`	Contém as classes que tratam da gerência de energia (PM) propriamente dita, entre elas classes que representam os equipamentos gerenciados, os grupos de equipamentos e as ações que podem ser remotamente executadas sobre eles
`br.ufrgs.inf.netpower.ups`	Contém as classes que representam os *no-breaks*, os eventos de energia que eles podem reportar e três tipos de monitores de *no-break*
`br.ufrgs.inf.netpower.util`	Contém classes utilitárias com finalidades diversas

A documentação completa das classes, gerada automaticamente com a ferramenta `javadoc`, faz parte do Apêndice D do trabalho. A Figura 6.2, a seguir, apresenta um diagrama de classes UML (*Unified Modeling Language*) simplificado para o gerente[1].

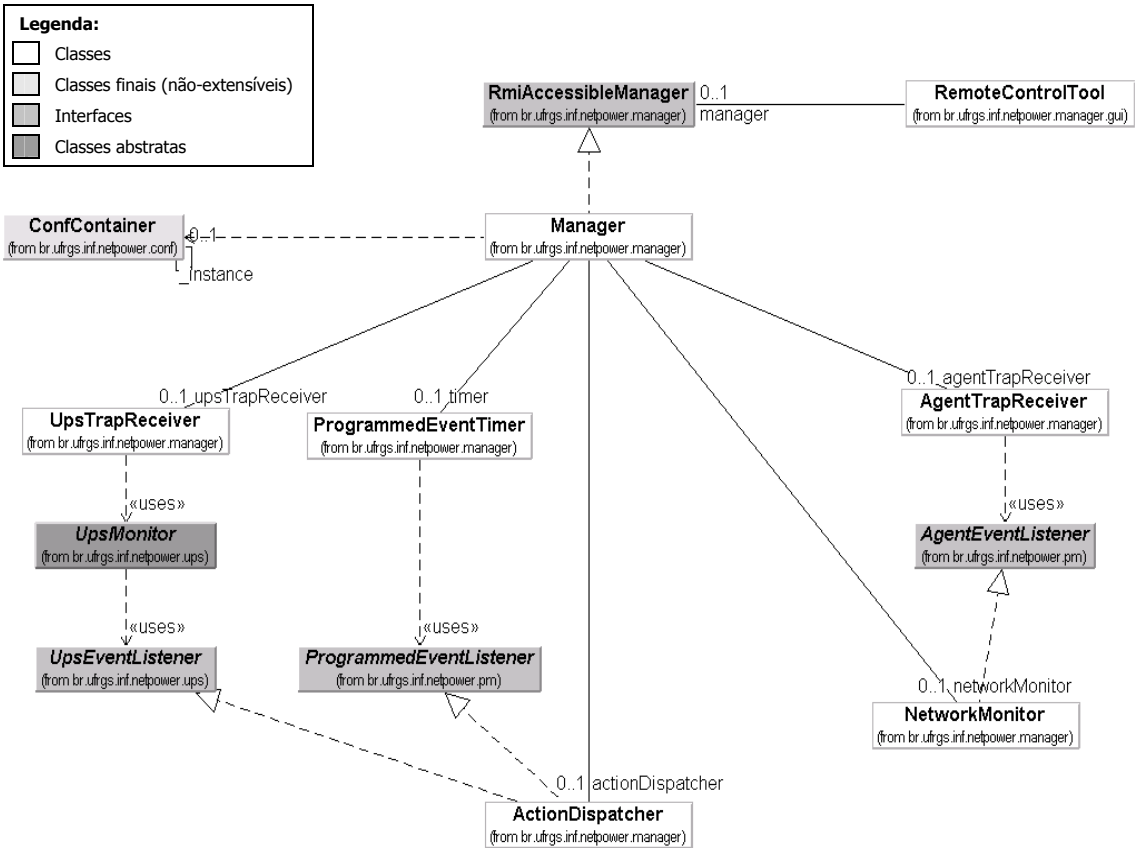


Figura 6.2 - Diagrama UML dos componentes do gerente

[1]O conjunto completo de diagramas de classes UML está disponível no Apêndice B.

6.3 Monitores de no-break e o receptor centralizado de notificações

Os monitores de *no-break* são os componentes responsáveis por detectar alterações nas condições de fornecimento de energia. São implementados como descendentes de uma classe abstrata chamada `UpsMonitor`, e diretamente conectados ao despachante de ações através de uma interface de eventos (`UpsEventListener`) que aquele componente implementa. Essa interface define métodos para tratar cada um dos quatro eventos de energia suportados pelo sistema (`UPS_ON_BATTERY`, `UPS_LOW_BATTERY`, `UPS_DEPLETED_BATTERY` e `EXTERNAL_POWER_RETURNED`).

A idéia por trás da classe abstrata é que os detalhes do protocolo de comunicação entre o *no-break* e o monitor não são relevantes, pois, internamente, o "dialeto" falado entre monitores e outros componentes interessados nos eventos de energia é único, como ilustra a Figura 6.3. Os três monitores implementados como parte deste trabalho utilizam o protocolo SNMP, mas, com pequeno esforço de programação, seria possível adicionar suporte a outros tipos de protocolos (comunicação serial, USB, etc.).

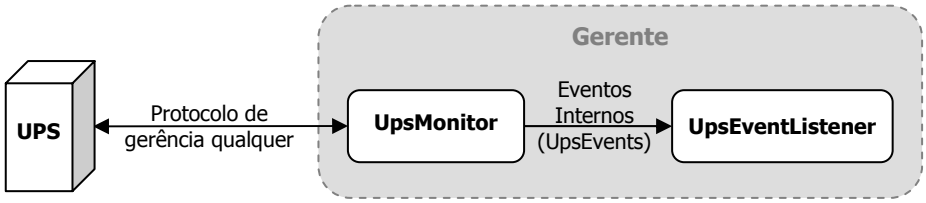


Figura 6.3 - Monitoramento de eventos da UPS

Teoricamente, o gerente tem a capacidade de monitorar um número indefinido de *no-breaks*, limitado apenas pelos recursos de memória e processamento do *host* em que ele é executado. No caso dos monitores SNMP implementados, que adotam um comportamento passivo, aguardando por notificações dos *no-breaks* para detectar alterações nas condições de fornecimento de energia, era necessário um mecanismo para organizar o acesso dos agentes SNMP de cada *no-break* ao gerente. Uma solução possível seria utilizar uma porta UDP diferente para cada agente, o que aumentaria a complexidade dos monitores, que precisariam gerenciar toda a comunicação SNMP com seus respectivos agentes. Utilizou-se, então, uma abordagem alternativa: um receptor de notificações único no gerente, que trata de todos os detalhes da comunicação SNMP com os agentes e apenas repassa as mensagens recebidas, já decodificadas, para os diferentes monitores. O receptor centralizado de notificações é implementado na classe `UpsTrapReceiver`.

Através dessa abordagem, uma única porta UDP pode ser utilizada para o recebimento de *traps* dos diversos agentes, o que é praxe em aplicações de gerência SNMP. Por outro lado, pode haver casos em que a MIB suportada pelo dispositivo não especifique o envio de notificações. Neste caso, o monitor precisaria adotar um comportamento ativo, realizando checagens periódicas do estado do *no-break*. Como a comunicação entre um monitor e os demais componentes se dá através de eventos internos, nada impede que ele envie requisições SNMP diretamente aos agentes e, baseado nas respostas obtidas, notifique os componentes interessados. Esses dois mecanismos de detecção de alterações nas condições de fornecimento de energia, e também o componente que centraliza a recepção de *traps*, são ilustrados na Figura 6.4.

Para diferenciar as notificações recebidas dos diversos *no-breaks*, o receptor de notificações utiliza um *identificador de agente*, composto do endereço IP do agente SNMP do

no-break e da *community string*[1] que o agente envia junto de cada pacote. Desta maneira, quando uma notificação chega ao receptor de notificações, ele é capaz de determinar sua origem e delegar o processamento da notificação para o monitor adequado. Obviamente, o monitor deve estar apto a "compreender" a mensagem recebida. Isto inclui ter conhecimento dos OIDs[2] das notificações e de seu significado, bem como do significado de quaisquer variáveis que possam estar inclusas na mensagem.

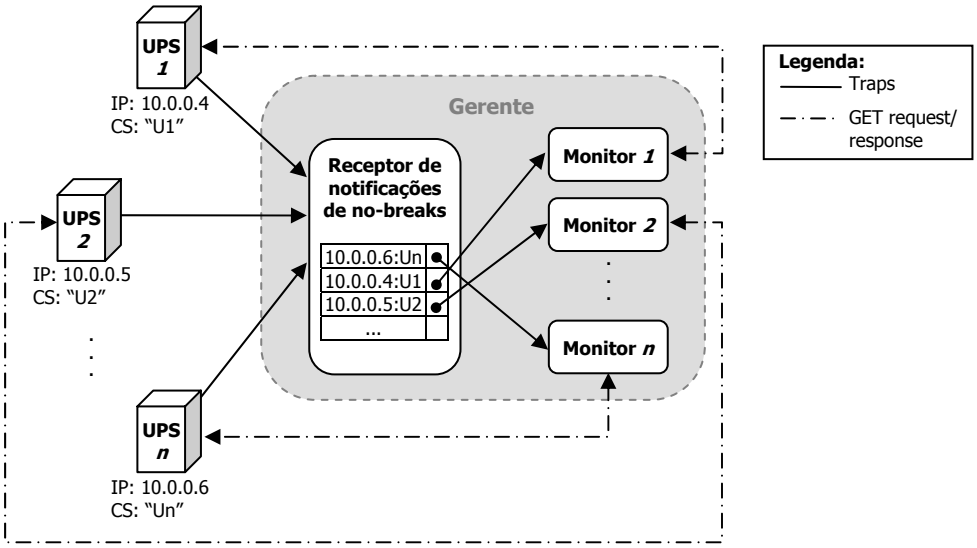


Figura 6.4 - Repasse de notificações para os diversos monitores

Os monitores implementados são descritos rapidamente a seguir. Todos suportam a versão 1 do protocolo SNMP (SNMPv1).

6.3.1 Monitor compatível com a UPS-MIB

Este monitor, implementado na classe `UpsMibCompliantMonitor`, é baseado na MIB genérica para gerência de *no-breaks* (definida na RFC 1628), e dispara os seguintes eventos:

- `UPS_ON_BATTERY`: quando a notificação `upsTrapOnBattery` é recebida e a UPS não estava operando em bateria até aquele instante (notificações subseqüentes do mesmo tipo são ignoradas para evitar que os eventos sejam duplicados);
- `UPS_LOW_BATTERY`: quando a notificação `upsTrapAlarmEntryAdded` é recebida e indica a adição do alarme `upsAlarmLowBattery` à tabela de alarmes;
- `UPS_DEPLETED_BATTERY`: quando a notificação `upsTrapAlarmEntryAdded` é recebida e indica a adição do alarme `upsAlarmDepletedBattery` à tabela de alarmes;
- `EXTERNAL_POWER_RETURNED`: quando a notificação `upsTrapAlarmEntryRemoved` é recebida e indica a remoção do alarme `upsAlarmOnBattery` da tabela de alarmes.

[1] *Community string:* uma "senha" enviada como parte de uma requisição SNMP e que pode ser usada para autenticação/autorização das requisições. É enviada como texto ASCII puro, sem qualquer tipo de criptografia, motivo pelo qual não é considerada um mecanismo "sério" de segurança – o que não a torna inapropriada para a identificação do *host* de origem em ambientes de redes locais, entretanto.

[2] OID, ou *Object Identifier* (identificador de objeto): uma série de números inteiros, separados por pontos, usada para identificar de maneira única um nodo específico da árvore de uma MIB.

6.3.2 Monitores para *no-breaks* da CP Eletrônica S.A.

Foram implementados dois tipos de monitor para *no-breaks* da CP Eletrônica. Cada um deles é compatível com um tipo diferente de MIB definido pela empresa.

O primeiro, implementado na classe `CpMonCompatibleMonitor`, é baseado na MIB para gerência de *clusters* de *no-breaks* da CP Eletrônica. Esse monitor é compatível com o *software* CP-Monitor Net, usado para supervisão de grupos de *no-breaks* microprocessados da linha Top. Ele é capaz de gerar os seguintes eventos:

- `UPS_ON_BATTERY`: quando a notificação `upsTrapAlarmOn` é recebida e indica a ocorrência do alarme `upsAlarmInputBad`, se a bateria não estiver esgotada ou em nível baixo;
- `UPS_LOW_BATTERY`: na mesma situação do evento anterior, se o nível da bateria estiver baixo;
- `UPS_DEPLETED_BATTERY`: na mesma situação do evento anterior, se a bateria estiver esgotada;
- `EXTERNAL_POWER_RETURNED`: quando a notificação `upsTrapPowerReturned` é recebida.

O segundo tipo de monitor é implementado na classe `CpCtrlCompatibleMonitor`, e é compatível com o agente SNMP que faz parte do programa CP-Ctrl, usado para supervisão de *no-breaks* de contato seco da linha Breakless. Esse monitor gera os seguintes eventos:

- `UPS_ON_BATTERY`: quando a notificação `upsTrapAcFail` é recebida;
- `UPS_LOW_BATTERY`: quando a notificação `upsTrapLowBat` é recebida;
- `EXTERNAL_POWER_RETURNED`: quando a notificação `upsTrapAcRestore` é recebida.

6.4 Temporizador de eventos programados

Além dos monitores de *no-break*, um outro componente do gerente também gera eventos que podem disparar a execução de ações nos equipamentos gerenciados: o temporizador (*timer*) de eventos programados, implementado na classe `ProgrammedEventTimer`.

Este componente é inicializado a partir de informações do arquivo de configuração que descrevem os eventos programados pelo administrador do sistema. Basicamente, um evento programado é um evento gerado automaticamente pela própria aplicação, e não por uma entidade externa (como no caso dos eventos de energia, gerados pelos *no-breaks* monitorados pelo gerente). Os eventos programados são caracterizados por um horário de início e uma possível taxa de repetição (ver seção 6.9.8). Ao contrário dos eventos de *no-break*, criados pelos monitores no momento de seu envio, os eventos programados são construídos pelo componente que interpreta o arquivo de configuração (`ConfContainer`) durante a inicialização do gerente (ver seção 6.9), e permanecem latentes até que o temporizador detecte que eles devam ser disparados. O gerente ainda provê um método para a adição de novos eventos programados depois da inicialização.

Depois de iniciado, o temporizador percorre o seu conjunto de eventos periodicamente (de acordo com a opção `timerInterval` do arquivo de configuração), comparando os horários de início de cada evento com o horário corrente do sistema. Quando os horários coincidem, o evento é disparado. O algoritmo que testa os eventos também leva em consideração a taxa de repetição especificada na configuração, que pode ser `"NEVER"` (executar o evento uma única

vez), `"DAILY"` (executar o evento diariamente), `"WEEKLY"` (semanalmente) ou `"MONTHLY"` (mensalmente). Se a taxa de repetição é compatível com a data corrente, o evento é testado; caso contrário, ele é ignorado. Os eventos cuja taxa de repetição é `"NEVER"` são excluídos depois de terem sido disparados.

O temporizador de eventos programados é conectado ao despachante de ações através da interface `ProgrammedEventListener`, que aquele componente implementa (ver Figura 6.2, pág. 65). Essa interface define um único método, que é chamado quando um evento programado deve ser processado.

6.5 Despachante de ações remotas

O despachante de ações remotas é o ponto central de processamento de eventos do gerente. Ele age em duas situações distintas:

- em resposta a eventos de energia, reportados pelos monitores de *no-break*;
- em resposta a eventos programados, reportados pelo temporizador.

Em ambos os casos, o despachante responde à ocorrência dos eventos de acordo com um mapa pré-configurado que associa cada evento a uma lista de ações correspondentes.

6.5.1 Ordem de despacho

As ações são programadas para despacho na ordem em que foram especificadas no arquivo de configuração. Em outras palavras, a ordem das entradas do arquivo que descrevem as ações remotas é importante. Por outro lado, quando o alvo de uma ação é um grupo de equipamentos, ela é desmembrada em ações equivalentes para cada um dos elementos do grupo (que não têm uma relação de prioridade entre si), e essas ações individuais são despachadas simultaneamente. Esse mecanismo de sincronização entre a execução de ações é ilustrado na Figura 6.5. Pode-se observar que o despacho de uma ação não tem início antes que o despacho da ação precedente tenha terminado. No caso de uma ação destinada a um grupo (como a Ação 2, na figura), o tempo de despacho da ação é igual ao maior tempo de despacho entre as ações derivadas para cada elemento do grupo, que são enviadas concorrentemente.

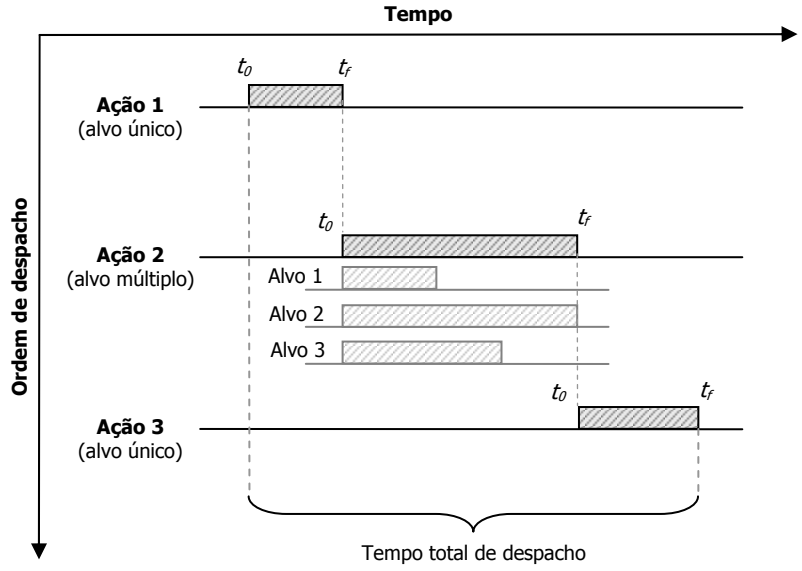


Figura 6.5 - Sincronização no despacho de ações

6.5.2 Tratamento de ações pendentes

A qualquer instante, pode haver ações pendentes no despachante (ações para as quais foi especificado um atraso que ainda não transcorreu completamente). As ações pendentes são associadas a seus respectivos alvos, e pode haver uma única ação pendente por alvo em um dado instante. Para decidir qual ação deve ser mantida e qual deve ser cancelada quando duas ações "colidem" (existia uma ação pendente para um determinado alvo e uma nova ação para o mesmo alvo foi disparada por um outro evento), utiliza-se o algoritmo apresentado na Figura 6.6:

```
Se há uma ação pendente para o mesmo alvo
da ação que está sendo processada, então
{
   Compare os níveis de prioridade das duas ações:
   Se uma tem alta prioridade e a outra não, então
   {
      Cancele aquela com baixa prioridade e execute
      aquela com alta prioridade;
   }
   Senão, se ambas têm alta prioridade, então
   {
      Compare os horários de execução das duas ações;
      Cancele a que seria executada por último;
   }
   Senão (ambas têm baixa prioridade)
   {
      Compare os horários de execução das duas ações;
      Cancele a que seria executada primeiro;
   }
}
```

Figura 6.6 - Algoritmo para remoção de ações em colisão

A prioridade de uma ação é determinada pelo tipo de evento ao qual ela está associada. Se a ação foi disparada por um evento de energia, reportado por um monitor de *no-break*, ela recebe alta prioridade. Caso contrário (se tiver sido disparada por um evento programado), recebe baixa prioridade.

Na ocorrência de eventos críticos, o algoritmo da Figura 6.6 garante que eventos programados, que têm baixa prioridade, não interfiram com a proteção da rede. Nessas situações, o algoritmo também garante que as ações mais urgentes sejam mantidas. Por exemplo, se um evento `UPS_ON_BATTERY` fosse configurado para executar uma ação com um atraso de 5 minutos, e um evento `UPS_LOW_BATTERY` fosse configurado para executar outra ação sem atraso, a segunda ação seria executada e a primeira cancelada. É claro que a noção de "urgência" é totalmente dependente da configuração. Se os atrasos das ações neste exemplo fossem configurados ao contrário, a ação associada ao evento mais crítico (`UPS_LOW_BATTERY`) seria cancelada, e aquela associada ao outro evento seria mantida.

Em situações de abastecimento normal de energia, durante as quais os monitores de *no-break* não reportam nenhum evento, a função do algoritmo é evitar que ações consecutivas que teriam efeitos inversos sobre um equipamento sejam despachadas. Por exemplo, no caso de um `SHUTDOWN` seguido de um `WAKEUP` para o mesmo alvo, o `SHUTDOWN` seria cancelado.

Considerando este mesmo exemplo, o algoritmo sozinho não evitaria que a ação `WAKEUP` fosse executada. Assumindo que o equipamento está num estado ativo, a execução dessa ação não teria efeito nenhum. Por isso, o sistema faz uso das convenções de estados de

consumo definidas no arquivo de configuração para determinar se, dado o estado global corrente de um equipamento, uma ação deve ou não ser executada. Os critérios para essa tomada de decisão são apresentados na Tabela 6.2.

Tabela 6.2 - Relação entre estados de consumo e ações possíveis

Ação \ Estado global	Desconhecido	Ativo	Baixo consumo	Desligado
RUN_COMMAND/ SHOW_MESSAGE	OK	OK	Impossível[1]	Impossível
SET_POWER_STATE	OK	OK	Impossível[1]	Impossível
SHUTDOWN	OK	OK	Impossível[1]	Não faz sentido
WAKEUP	OK	Não faz sentido	OK[2]	OK

Esta tabela pode ser lida da seguinte maneira:

- se o estado do equipamento é desconhecido, o gerente deverá fazer uma tentativa de executar a ação. Se o agente estiver em condições de atender à requisição e a ação tiver qualquer efeito sobre o estado do equipamento, ele enviará uma notificação em resposta, que o gerente usará para atualizar o mapa da rede. Caso contrário, o estado do equipamento continuará desconhecido. Se o estado do equipamento for conhecido, um dos critérios a seguir é aplicado;
- só é possível executar comandos (ação `RUN_COMMAND`) ou mostrar mensagens (ação `SHOW_MESSAGE`) em um equipamento se ele estiver em estado ativo;
- só é possível alterar o estado de um componente do equipamento se o equipamento estiver em um estado ativo (no qual o agente estará responsivo);
- só faz sentido desligar um equipamento se ele estiver em um estado ativo;
- só faz sentido "acordar" um equipamento se ele estiver desligado ou em estado de baixo consumo.

É importante frisar que os critérios listados na tabela são baseados no estado global do equipamento. Se, por exemplo, um agente notificar o gerente de que um componente (monitor de vídeo, por exemplo) entrará num estado de baixo consumo, isso não tem qualquer efeito sobre a percepção que o gerente tem do estado global do equipamento, apenas sobre o estado do componente em questão. Assim, neste exemplo, o gerente despacharia outras ações subseqüentes sem problema.

O procedimento de teste da validade de ações em relação ao estado dos equipamentos pode ser habilitado ou desabilitado no arquivo de configuração (ver seção 6.9.2).

6.5.3 Envio das requisições SNMP

Como foi mencionado anteriormente, com exceção da ação `WAKEUP`, todas as outras ações equivalem, no gerente, ao envio de um comando SNMP (uma mensagem do tipo `Set-`

[1] Como não se pode garantir que o agente está em condições de responder, a seqüência aceitável de operações seria despertá-lo do estado de baixo consumo (WAKEUP), aguardar notificação indicando que o equipamento passou a um estado ativo, e então executar outra ação.

[2] Depois de um WAKEUP, o equipamento passa para um estado ativo e pode executar outras ações.

`Request`) para alterar o valor de um objeto da MIB de um determinado agente. A Tabela 6.3 lista os objetos da MIB alterados por cada tipo de ação.

Nesta implementação, as mensagens são enviadas através de SNMPv1, a título de validar o funcionamento do sistema. Não foi possível encontrar uma biblioteca SNMP gratuita para a plataforma Java com suporte à versão 3 do protocolo, que permitiria validar a origem das mensagens nos agentes, conforme especificado no capítulo 4. Existem bibliotecas gratuitas em C e C++ que suportam SNMPv3, portanto uma solução possível seria implementar as funções de comunicação numa destas linguagens e construir um *wrapper*[1] que as acessasse através de JNI (*Java Native Interface*), a interface de programação nativa (dependente de plataforma) de Java. Mais informações sobre as bibliotecas utilizadas são dadas na seção 6.10.

Tabela 6.3 - Tradução de ações em comandos SNMP

Ação	Objeto	Valor
`RUN_COMMAND`	`powerMgmtMIB` ↳ `sysInteraction` ↳ `runCmd`	`String` contendo a linha de comando especificado na configuração
`SHOW_MESSAGE`	`powerMgmtMIB` ↳ `sysInteraction` ↳ `showMsg`	`String` contendo a mensagem especificada na configuração
`SET_POWER_STATE`	`powerMgmtMIB` ↳ `componentTable` ↳ `componentEntry` ↳ `componentCurrentState` A entrada alterada é aquela cujo índice (`componentId`) corresponde ao nome de componente especificado na configuração	`String` contendo o estado especificado
`SHUTDOWN`	`powerMgmtMIB` ↳ `componentTable` ↳ `componentEntry` ↳ `componentCurrentState` A entrada alterada é aquela cujo índice (`componentId`) é igual a `"global"`	`String` com o valor `"off"`

6.5.4 Ações WAKEUP: envio de *"magic packets"*

Ações do tipo `WAKEUP` são executadas através do envio de pacotes especiais, chamados de *"magic packets"*, para os equipamentos-alvo.

A seqüência de dados que identifica um *"magic packet"* consiste de 16 repetições do endereço MAC da interface de rede do equipamento em questão. Essa seqüência pode estar localizada em qualquer parte do pacote, mas deve ser precedida de uma "marca de sincronização", composta de 6 bytes de valor hexadecimal FF. Segundo a documentação da AMD, não há qualquer outra restrição em relação ao pacote que contém a seqüência. Por exemplo, não importa o protocolo usado para transmissão (TCP/IP, IPX, etc.) ou se o pacote sofreu roteamento ao longo do percurso [ADV95].

[1] Objeto que não implementa métodos diretamente, mas delega sua execução para uma outra entidade cujos detalhes de funcionamento não é necessário conhecer.

O envio dos *"magic packets"* pelo gerente foi implementado através do serviço de datagramas da plataforma Java (classes `java.net.DatagramSocket` e `java.net.DatagramPacket`). Usando esse tipo de serviço, é possível transmitir os pacotes sem que seja necessário estabelecer uma conexão com o computador-alvo – restrição existente quando o computador está em estado de baixo consumo. A Figura 6.7 mostra uma tela do programa LanExplorer [SUR2002] contendo a captura de um datagrama enviado pela aplicação, com a seqüência identificadora do *"magic packet"* para a interface de rede cujo endereço MAC é "`00:00:21:F3:EF:12`".

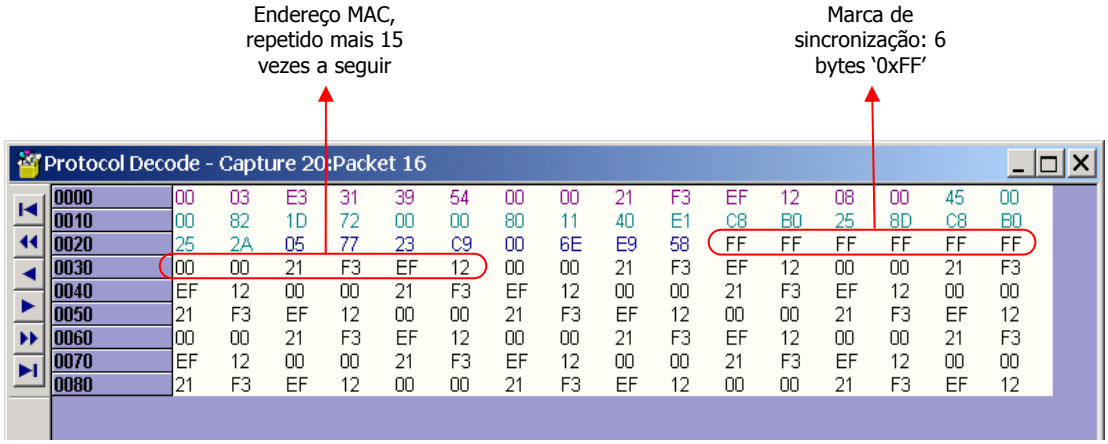


Figura 6.7 - Captura de *"Magic Packet"*

O endereço MAC do equipamento a ser "despertado" é obtido através de uma requisição do protocolo ARP[1]. Como não há uma implementação desse protocolo em Java, foi necessário implementar uma função em linguagem nativa capaz de executar a chamada de sistema operacional correspondente. O acesso à função é feito através da interface JNI. Essa interface possibilita que o código Java permaneça independente de plataforma, desde que sejam providas implementações para os métodos nativos em todas as plataformas-alvo em que a máquina virtual é capaz de executar (ou, no caso de um projeto específico, em todas as plataformas onde se espera executar a aplicação). Neste sentido, foram implementadas versões para Windows e para Linux da biblioteca que contém a função de pesquisa ARP.

Os sistemas operacionais tipicamente implementam uma *cache* ARP, cujo mecanismo de atualização e invalidação de entradas pode variar bastante. Por este motivo, e para aumentar as chances de que o endereço MAC de um equipamento esteja disponível quando uma ação `WAKEUP` tiver que ser enviada, o gerente possui sua própria *cache* ARP. Toda vez que o monitor da rede faz uma consulta SNMP completa ao agente de um equipamento (segundo definido na seção 6.6.2), ele antes envia uma requisição ARP para obter o seu endereço MAC. Se, por algum motivo, a chamada JNI retornar erro, o gerente tenta descobrir o endereço consultando o campo `macAddr` da MIB do agente. Depois de obter o endereço, o monitor verifica se ele está presente na *cache*. Se não estiver, ou se for diferente do endereço armazenado, a *cache* é atualizada. Esta *cache* é gravada periodicamente em disco, e carregada quando é acessada pela primeira vez.

[1] *Address Resolution Protocol*, protocolo usado para descobrir dinamicamente o endereço de enlace/físico (p. ex. *Ethernet*) que corresponde a um determinado endereço de rede (p. ex. IP). Especificado na RFC 826 [PLU82], era originalmente destinado a converter endereços de um protocolo de rede qualquer para endereços *Ethernet*, mas também foi implementado para outros tipos de padrões IEEE 802, como *Token Ring*.

6.6 O monitor da rede e o receptor de notificações dos agentes

No gerente, a recepção das notificações enviadas pelos agentes serve para atualizar o mapa da rede. O componente encarregado de tratar essas notificações é o monitor da rede, implementado na classe `NetworkMonitor`. Como se pode ver na Figura 6.2 (pág. 65), este componente implementa a interface `AgentEventListener`, que o conecta ao receptor de notificações dos agentes, implementado na classe `AgentTrapReceiver`.

A tarefa do receptor de notificações é simplesmente converter as *traps* SNMP recebidas dos agentes para a representação interna de eventos (`AgentEvent`), abstraindo o monitor da rede dos detalhes de comunicação. O receptor de notificações suporta SNMPv1.

A qualquer instante, cada equipamento monitorado pode encontrar-se numa das seguintes situações:

- o estado do equipamento é conhecido e é um dos estados ativos: neste caso, dentro de, no máximo, `aliveTimeout` minutos (ver seção 6.9), o monitor atualizará o estado do equipamento. A atualização da percepção que o monitor tem do estado do equipamento pode resultar da ocorrência dos seguintes eventos:
  - recepção de uma `trapAliveMsg` do agente, indicando que o equipamento continua em estado ativo;
  - recepção de uma `trapOperationError`, indicando que um erro ocorreu durante a execução de uma operação no agente – o estado do equipamento é considerado desconhecido;
  - recepção de outras *traps* que indicam mudança no estado do equipamento – o estado é alterado de acordo com o tipo de notificação;
- o estado do equipamento não é conhecido, ou é conhecido e pertence aos grupos de estados de baixo consumo ou desligados: neste caso, não há nenhum *timer* em efeito para o equipamento em questão. Assim que a próxima notificação for recebida, o estado do equipamento será alterado.

6.6.1 Constantes para estados de consumo

Além dos conjuntos de estados configuráveis (ver seção 4.3.3), a aplicação trabalha com as seguintes constantes:

Tabela 6.4 - Constantes para estados de consumo

Constante	Valor	Grupo
`STATE_ON`	`"on"`	Ativo
`STATE_IDLE`	`"idle"`	
`STATE_SLEEP`	`"sleep"`	Baixo consumo
`STATE_OFF`	`"off"`	Desligado
`STATE_UNKNOWN`	`"unknown"`	-

Estas constantes não são afetadas pela configuração das convenções de estados. Em outras palavras, estes valores sempre fazem parte das convenções de estados da aplicação, independente de estarem presentes no arquivo de configuração ou não. Elas são importantes na interpretação dos estados dos equipamentos e permitem que o sistema opere corretamente mesmo quando os valores recebidos dos agentes não são reconhecidos. Por exemplo, se uma

notificação `trapImminentSleep` for recebida, o agente está indicando, explicitamente, que o equipamento entrará num estado de consumo reduzido. Neste caso, se o estado indicado na notificação não for conhecido, o gerente não pode simplesmente ignorar o efeito lógico da mensagem. Assim, um valor padrão (`STATE_SLEEP`) é utilizado. Outros usos das constantes são descritos a seguir.

6.6.2 Interpretação das notificações

As notificações recebidas pelo gerente são interpretadas da seguinte maneira pelo monitor da rede:

- **`trapAliveMsg:`** não se assume nada a respeito dos componentes individuais (dispositivos cujo nome é diferente de "`global`" na tabela de componentes do agente). O estado global do componente é alterado para o valor anexado à notificação, se ele for um estado ativo conhecido (considerando as convenções de estados), ou para o estado ativo padrão (o valor da constante `STATE_ON`), caso o valor anexado não seja reconhecido. Na recepção de uma `trapAliveMsg`, o monitor verifica se o agente do equipamento já foi consultado anteriormente para preencher as estruturas internas. Se não foi, uma consulta é realizada e o objeto que representa o equipamento é atualizado com os valores obtidos;
- **`trapPowerOn:`** ao receber esta notificação, o monitor sempre realiza uma nova consulta ao agente e atualiza a representação interna do equipamento. Os estados de todos os componentes do equipamento são atualizados de acordo com as informações providas pelo agente;
- **`trapWakeUp:`** o estado de todos os componentes é alterado para o valor da constante `STATE_ON`. Se o agente do equipamento ainda não tinha sido consultado pelo monitor, a consulta é realizada e os estados dos componentes são atualizados de acordo com as informações recebidas;
- **`trapImminentComponentStateChg:`** o monitor altera o estado do componente identificado na notificação para o estado especificado. O estado dos outros componentes não é alterado.
- **`trapImminentSleep:`** se o estado anexado à notificação é conhecido, o estado do componente "`global`" é alterado para este valor. Caso o valor não seja conhecido, a constante `STATE_SLEEP` é utilizada. Independente do valor usado, o estado é propagado para os demais componentes – quando um equipamento está "dormindo", assume-se que todos os seus componentes também estão, até que outra notificação indique o contrário;
- **`trapImminentShutdown:`** o estado de todos os componentes é alterado para a constante `STATE_OFF`;
- **`trapOperationError:`** o estado de todos os componentes é alterado para desconhecido (constante `STATE_UNKNOWN`).

Obviamente, ao utilizar os valores das constantes para evitar que valores desconhecidos sejam atribuídos ao estado global dos equipamentos, o monitor garante que a visão lógica que possui da rede seja coerente com a sua própria capacidade de interpretação.

6.7 Operação local e remota

O gerente pode ser operado tanto a partir do *host* em que executa quanto a partir de um *host* remoto, através de RMI (*Remote Method Invocation*), o mecanismo de invocação remota de métodos de Java.

A grande vantagem desta abordagem é que ela permite a construção de diferentes ferramentas para operação e configuração do gerente, através de uma interface de acesso comum (`RmiAccessibleManager`, conforme ilustra a Figura 6.2). Como será apresentado adiante, implementou-se, a título de exemplo, uma aplicação gráfica capaz de executar algumas operações simples de visualização e configuração (ver seção 6.8). Mas as possibilidades são inúmeras. Conforme se mencionou no capítulo 4, uma ferramenta baseada em tecnologias *Web* foi desenvolvida num trabalho anterior, e poderia ser adaptada para operar com esta versão da aplicação de gerência.

6.7.1 Autenticação dos clientes RMI

A assinatura de todos os métodos em `RmiAccessibleManager` contém, no mínimo, um argumento que carrega as credenciais do usuário que está enviando a requisição, como ilustra o exemplo da Figura 6.8.

```
public void remoteStart(SimplePrincipal user) throws RemoteException;
```

Figura 6.8 - Assinatura de método RMI do gerente

Neste exemplo, o método `remoteStart()`, que permite disparar a execução do gerente remotamente, recebe um único argumento, que contém as informações de usuário. Todos os métodos que podem ser acessados remotamente possuem uma assinatura semelhante, e só são efetivamente executados se a autenticação dessas credenciais (nome de usuário e senha) for feita com sucesso.

A classe `SimplePrincipal`[1] contém apenas um nome de usuário e uma senha, que deve ser codificada através do algoritmo MD5[2]. Esse algoritmo não é um algoritmo de criptografia, portanto não é a escolha mais adequada para codificação de senhas. O problema mais óbvio é que duas senhas idênticas geram um código MD5 idêntico. No entanto, é uma maneira fácil de gerar um código ilegível a partir de uma senha, pois não exige o uso de chaves como algoritmos de criptografia, e serve ao propósito de evitar a transmissão de senhas como texto puro através da rede.

Os nomes de usuário e as respectivas senhas codificadas são armazenados em um arquivo no *host* do gerente. O arquivo tem um formato semelhante ao do arquivo `/etc/passwd` de sistemas Unix, como ilustra a Figura 6.9. A validação se dá simplesmente através da comparação das credenciais anexadas a uma requisição RMI com as informações gravadas neste arquivo. Se elas coincidem, o cliente é autorizado. Como se pode ver na figura, os usuários

[1] O termo *"principal"* (em inglês) é usado, no jargão de segurança, para se referir ao nome de um serviço, administrador ou usuário que é registrado numa base de dados e associado a privilégios de acesso variados.

[2] *Message Digest #5,* função de codificação de mensagens largamente utilizada na Internet para verificar se um arquivo foi corrompido depois de sua transmissão. O algoritmo produz uma "impressão digital" de 128 bits da mensagem original, independente do seu tamanho. O código gerado é chamado de *digest* ou *hash*.

`"luis"` e `"test"` possuem a mesma senha (embora seja difícil adivinhar qual é)[1]. Nesta versão da implementação, privilégios de administrador (o nome de usuário deve ser `"admin"`) são exigidos para execução de todos os métodos remotos.

```
admin:21232f297a57a5a743894a0e4a801fc3
luis:698dc19d489c4e4db73e28a713eab07b
test:698dc19d489c4e4db73e28a713eab07b
```

Figura 6.9 - Exemplo de arquivo de credenciais

Este mecanismo simples de validação de credenciais oferece um nível relativamente baixo de segurança, e foi implementado apenas a título experimental. Entre os problemas que ele apresenta pode-se citar, principalmente, a possibilidade de roubo de senhas (que são transmitidas e armazenadas no mesmo formato), e ataques do tipo *"man-in-the-middle"* (também chamado de *spoofing*), em que uma entidade não autorizada intercepta, possivelmente altera, e então retransmite mensagens, fazendo-se passar por um cliente autorizado. Para evitar esse tipo de ataque, seria necessário implementar autenticação e criptografia das mensagens num nível abaixo do protocolo RMI, o que aumentaria significativamente a complexidade da implementação. Sugestões para soluções de segurança na comunicação via RMI são dadas na conclusão do trabalho (capítulo 8).

6.8 Interface gráfica

A interface gráfica implementada consiste em uma aplicação JFC/*Swing* que permite visualizar o estado dos equipamentos gerenciados, bem como dos *no-breaks* monitorados. A aplicação também permite executar ações sobre os equipamentos e alterar o arquivo de configuração armazenado no *host* do gerente.

As Figuras 6.10, 6.11 e 6.12, a seguir, mostram algumas telas da aplicação.

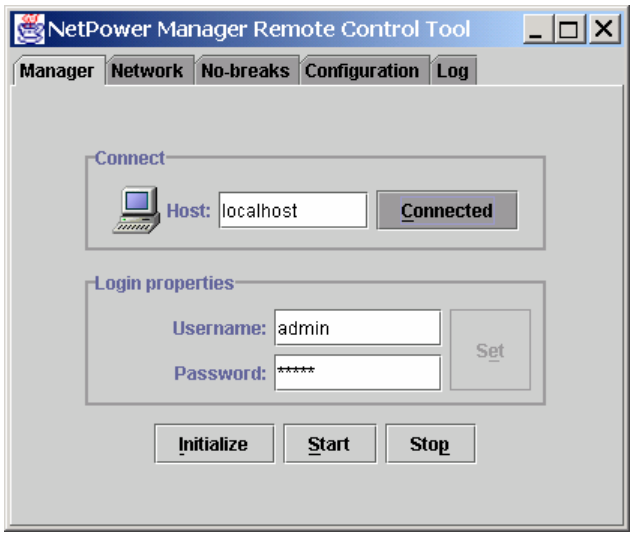


Figura 6.10 - Tela de conexão com o gerente

[1] Presume-se que a dificuldade de se conseguir duas mensagens que gerem o mesmo código é da ordem de 2^{64} operações, e que a dificuldade de se conseguir qualquer mensagem que gere um código pré-determinado é da ordem de 2^{128} operações. [RIV92]

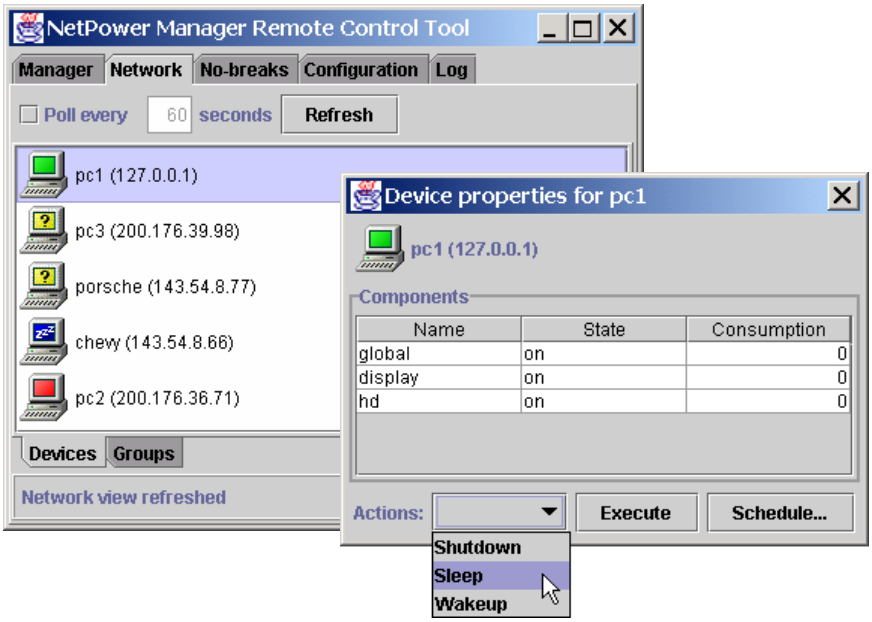


Figura 6.11 - Telas de visualização do estado dos equipamentos

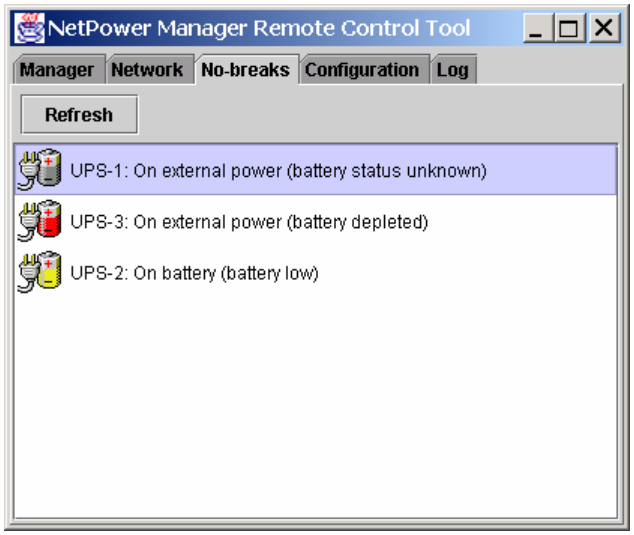


Figura 6.12 - Tela de visualização dos *no-breaks*

6.9 Configuração

O gerente é configurado através de um arquivo XML (*Extensible Markup Language* [W3C2000]). Este arquivo contém as opções de funcionamento do gerente (como números de portas TCP/IP, *timeouts*, etc.) e informações sobre os *no-breaks* monitorados e sobre os equipamentos gerenciados, além das regras que definem como o gerente deve reagir à ocorrência de eventos.

O uso de XML como linguagem de configuração de aplicativos já é praticamente um padrão, provavelmente graças à sua estrutura hierárquica, bastante adequada a este tipo de função, e sua facilidade de interpretação. Existem bibliotecas disponíveis para a interpretação de arquivos XML para virtualmente todas as linguagens de programação. A última versão do *kit* de

desenvolvimento Java da Sun Microsystems, na data de escrita deste trabalho (J2SE 1.4.0 SDK), já traz os pacotes de manipulação de XML integrados ao conjunto de classes-padrão.

Através do uso de uma biblioteca desse tipo, a aplicação é poupada de todo o trabalho de interpretação e validação sintática, e de grande parte do trabalho de validação semântica do arquivo de configuração. Obviamente, como qualquer outro tipo de *parser,* um interpretador XML converte o arquivo de entrada para uma representação em memória, que pode então ser utilizada pela aplicação. No caso da implementação apresentada neste trabalho, usou-se um *parser* que gera como resultado uma estrutura que segue o modelo DOM (*Document Object Model*), especificado pelo W3C [W3C2002]. Este modelo consiste, basicamente, de uma árvore de objetos cujos métodos permitem percorrer e modificar a estrutura hierárquica e o conteúdo dos nodos.

A seguir, são apresentados alguns detalhes e exemplos do arquivo de configuração. A gramática (DTD) completa para arquivos de configuração e um arquivo de exemplo podem ser encontrados no Apêndice D.

6.9.1 Estrutura geral do arquivo

O arquivo de configuração é dividido nas seguintes seções, que devem ser especificadas nesta ordem:

- **`<administrator>`**: contém informações sobre o administrador do sistema, como nome e endereço de e-mail;
- **`<options>`**: contém as opções de funcionamento do programa;
- **`<state-conventions>`**: define as convenções de estados de consumo com as quais o gerente deve trabalhar;
- **`<devices>`**: contém a descrição dos equipamentos gerenciados pelo sistema;
- **`<groups>`**: permite agrupar os equipamentos gerenciados segundo critérios arbitrários;
- **`<ups-list>`**: contém a descrição dos *no-breaks* monitorados pelo gerente;
- **`<action-lists>`**: contém definições de listas de ações que podem facilitar o trabalho de configuração de eventos;
- **`<events>`**: contém a descrição dos eventos de energia e dos eventos programados.

6.9.2 Opções de programa

As opções são especificadas na seção `<options>` do arquivo de configuração. Cada opção é configurada através de um nome e um valor, conforme o exemplo da Figura 6.13.

```
<option name="upsTrapPort" value="8162"/>
```

Figura 6.13 - Exemplo de opção de programa

O nome e o valor de cada opção especificada são checados pela aplicação durante a interpretação do arquivo. Opções não suportadas ou valores em formato incorreto são detectados e ocasionam erros não-fatais (a interpretação do arquivo continua).

Na versão atual, o gerente suporta as seguintes opções:

- **`agentPort:`** porta UDP usada pelos agentes para receber conexões SNMP;
- **`agentTrapPort:`** porta UDP usada pelo gerente para receber notificações (*traps*) dos agentes;
- **`aliveTimeout:`** tempo máximo, em minutos, que o gerente deve esperar entre duas mensagens *"I'm alive"* consecutivas de um agente, antes de assumir que o estado do equipamento correspondente é desconhecido;
- **`timerInterval:`** intervalo (ciclo de relógio), em segundos, que o temporizador de eventos espera entre checagens consecutivas dos eventos programados;
- **`logLevel:`** nível de *logging* (armazenamento de histórico da execução) desejado;
- **`logSize:`** tamanho máximo do arquivo de *log*. Zero é interpretado como ilimitado;
- **`snmpRetries:`** número máximo de tentativas de enviar cada mensagem SNMP para um agente na ocorrência de erro;
- **`snmpTimeout:`** tempo máximo, em milissegundos, para a transmissão de uma mensagem SNMP;
- **`upsTrapPort:`** porta UDP em que o gerente deve monitorar notificações dos *no-breaks*;
- **`controllerHost:`** nome ou endereço IP do *host* ao qual é permitido fazer requisições ao gerente via RMI. Esta checagem é feita antes da validação de credenciais da requisição;
- **`smtpServer:`** nome ou endereço IP do servidor SMTP (de envio de e-mail) que deve ser usado pelo gerente para enviar mensagens ao administrador do sistema;
- **`timestampFormat:`** formato para todos os campos que representam data e hora no arquivo de configuração. Deve estar de acordo com as definições da classe `java.text.SimpleDateFormat`;
- **`performActionChecks:`** determina se o gerente deve checar a "sanidade" das ações. Esta checagem é uma tentativa de determinar, dados o tipo da ação e o estado global de seu equipamento-alvo, se ela deve ser executada ou não (ver seção 6.5.2, Tabela 6.2);
- **`aliveTrapInterval:`** intervalo, em minutos, que os agentes devem esperar entre envios consecutivos de mensagens *"I'm alive"*;
- **`passwordFile:`** caminho completo para o arquivo de senhas usado para autenticação de usuários.

6.9.3 Especificação das convenções de estados

As convenções de estados são definidas conforme o exemplo da Figura 6.14. A finalidade destas convenções foi explicada em detalhe na seção 4.3.3.

```
<state-conventions>
   <active-states> on, working, active, idle </active-states>
   <sleep-states> standby, hibernate, suspend, sleeping </sleep-states>
   <off-states> off </off-states>
</state-conventions>
```

Figura 6.14 - Exemplo de definição de convenções de estados

6.9.4 Descrição dos equipamentos gerenciados

Os equipamentos gerenciados são descritos na seção `<devices>`. Cada entrada desta seção tem o formato exemplificado na Figura 6.15.

```
<device name="chevy" address="chevy.inf.ufrgs.br" mgmt-model="linux-apm"/>
```

Figura 6.15 - Exemplo de configuração de equipamento

Como se pode ver no exemplo, cada equipamento recebe um nome e um endereço. O nome é usado internamente como identificador para o equipamento e, portanto, deve ser único no contexto do arquivo. Para o campo `address`, deve-se usar preferencialmente o nome do *host*, pois o endereço IP freqüentemente é atribuído de maneira dinâmica, através de mecanismos como DHCP[1].

O atributo `mgmt-model` é opcional, e não é utilizado nesta versão da aplicação. A intenção original deste parâmetro era auxiliar o administrador a configurar ações para um equipamento, possibilitando-lhe escolher a partir de um conjunto pré-definido de ações associadas a um "modelo de gerência de energia". Essa descrição das ações que seriam suportadas por um determinado tipo de agente deveria ser gerada pelo programador responsável pela implementação do agente.

6.9.5 Definição de grupos

O mecanismo de agrupamento dos equipamentos gerenciados de acordo com um critério arbitrário foi discutido na seção 4.3.5. No arquivo de configuração, os equipamentos são agrupados conforme o exemplo da Figura 6.16.

```
<group name="pcsGroup">
   <group-member ref="pc1"/>
   <group-member ref="pc2"/>
   <group-member ref="pc3"/>
</group>
```

Figura 6.16 - Exemplo de configuração de grupos

Cada membro do grupo deve corresponder a um identificador atribuído a um equipamento na seção `<devices>` ou a um outro grupo definido anteriormente na seção `<groups>`. Não há relação de prioridade entre os elementos de um grupo. Se uma ação for destinada a um grupo, ela será desmembrada em ações equivalentes para cada equipamento e será despachada simultaneamente para todos os membros (conforme especificado na seção 6.5.1).

[1] *Dynamic Host Configuration Protocol*, protocolo que possibilita a configuração dinâmica de computadores ligados em rede a partir de um servidor.

6.9.6 Descrição dos *no-breaks*

Os *no-breaks* monitorados pelo gerente são descritos na seção `<ups-list>` do arquivo de configuração, conforme o exemplo da Figura 6.17.

A cada UPS é atribuído um nome e o tipo de protocolo de comunicação que ela suporta. Nesta versão da aplicação, o único valor suportado para o protocolo é "`snmp`". Opcionalmente, podem-se especificar parâmetros para a UPS. Estes parâmetros podem ser usados para construir uma representação interna mais precisa do dispositivo. No exemplo acima, o gerente detecta que o dispositivo suporta o protocolo SNMP e, portanto, tenta construir uma instância da classe `SnmpAwareUps` para representá-lo (ao invés da representação mais genérica provida pela superclasse `Ups`). Conforme ilustra a figura, os parâmetros necessários para *no-breaks* que suportam o protocolo "SNMP" são o endereço do agente (`agentIp`), a `communityString` e a porta UDP usada pelo agente (`snmpPort`).

```
<ups name="UPS-1" protocol="snmp">
   <ups-params>
      <param name="agentIp" value="127.0.0.1"/>
      <param name="communityString" value="UPS1"/>
      <param name="snmpPort" value="8161"/>
   </ups-params>
   <ups-monitor class="br.ufrgs.inf.netpower.ups.UpsMibCompliantMonitor"/>
</ups>
```

Figura 6.17 - Exemplo de configuração de UPS

Além disso, deve-se especificar o nome da classe Java capaz de efetuar o monitoramento do dispositivo, bem como quaisquer parâmetros necessários à sua configuração. Como foi explicado em detalhe anteriormente (seção 6.3), os monitores de *no-break* funcionam como pequenos *plug-ins* de *software* que podem ser acoplados ao gerente para monitorar tipos específicos de UPS, sem que seja necessário recompilar o próprio gerente.

6.9.7 Listas de ações

Para facilitar a configuração dos mapeamentos entre eventos e ações, é possível definir *listas de ações*, que permitem agrupar conjuntos de ações de maneira lógica. As listas de ações funcionam de maneira semelhante às funções de uma linguagem de programação procedural: permitem definir seqüências de comandos (ou, neste caso, de ações) e atribuir-lhes um nome, de modo que elas possam ser utilizadas repetidamente, em diferentes contextos. Por exemplo, pode-se definir uma lista de ações para desligar todos os servidores da rede, ou para colocar os monitores de todos os PCs em estado de baixo consumo, etc.

As listas de ações são definidas conforme o exemplo da Figura 6.18.

```
<action-list name="delayedShutdown" descr="Shutdown all devices after 5 min">
   <action target="printersGroup" type="SHUTDOWN" delay="5"/>
   <action target="serversGroup" type="SHUTDOWN" delay="5"/>
   <action target="pcsGroup" type="SHUTDOWN" delay="5"/>
</action-list>
```

Figura 6.18 - Exemplo de configuração de listas de ações

6.9.8 Descrição dos eventos dos *no-breaks* e dos eventos programados

É na seção `<events>` que os dois tipos de eventos suportados pelo sistema são associados a conjuntos de ações. A primeira parte dessa seção, `<ups-events>`, contém as definições dos eventos de energia, detectados pelos monitores de *no-break* conforme descrito na seção 6.3; a segunda, `<programmed-events>`, contém as definições dos eventos programados, disparados automaticamente pela aplicação nos horários determinados (conforme descrito na seção 6.4).

A sintaxe para configuração dos eventos e sua associação a listas de ações é exemplificada na Figura 6.19.

```
<events>
   <ups-events>
      <ups-event src="UPS-1" type="UPS_ON_BATTERY">
         <include-actions action-list="delayedShutdown"/>
      </ups-event>
      <ups-event src="UPS-1" type="UPS_LOW_BATTERY" warn-admin="true">
         <include-actions action-list="urgentShutdown"/>
      </ups-event>
      <ups-event src="UPS-1" type="EXTERNAL_POWER_RETURNED">
         <include-actions action-list="wakeupServers"/>
      </ups-event>
   </ups-events>
   <programmed-events>
      <programmed-event name="prgMonitorOff"
                        startTime="04/02/2002 19:00:00"
                        repeat="DAILY">
         <action target="pcsGroup" type="SET_POWER_STATE">
            <param name="component" value="display"/>
            <param name="state" value="off"/>
         </action>
      </programmed-event>
   </programmed-events>
</events>
```

Figura 6.19 - Exemplo de configuração de eventos

Como se pode observar no exemplo, para os eventos programados são definidos uma data e horário de início, e também a freqüência com que o evento deve ser repetido. O formato para os horários pode ser customizado através da opção `timestampFormat` (seção 6.9.2). A freqüência pode assumir os valores "`NEVER`", "`DAILY`", "`WEEKLY`" e "`MONTHLY`", significando, respectivamente, que o evento deve ser executado uma única vez na data e horário programados, ou repetido diária, semanal ou mensalmente no mesmo horário.

A definição das ações que serão executadas em resposta à ocorrência de um evento é feita da mesma maneira para os dois tipos de evento. Pode-se utilizar a cláusula `<include-actions>` para especificar uma lista de ações definida previamente, cláusulas `<action>` para definir ações individualmente, ou uma combinação das duas cláusulas.

Vale lembrar que não há uma correspondência forçada entre as ações efetivamente suportadas pelos agentes dos equipamentos e as ações definidas no arquivo de configuração. É tarefa de cada agente interpretar as ações recebidas, bem como possíveis parâmetros, e decidir se elas fazem sentido ou se podem ser executadas.

6.10 Ferramentas utilizadas

6.10.1 Compiladores e interpretador Java

Para compilar e executar os programas Java, utilizou-se o compilador e o interpretador (máquina virtual) providos como parte integrante do *kit* de desenvolvimento (SDK) da Sun Microsystems, versões 1.3.1 e 1.4.0. Este *kit*, que inclui várias outras ferramentas, está disponível para *download* no *site* da Sun, no seguinte endereço:

http://java.sun.com/j2se/

A biblioteca (DLL) usada para a obtenção dos endereços MAC dos equipamentos, implementada em C++, foi gerada com o compilador de linha de comando gratuito da Borland, disponível no seguinte endereço:

http://www.borland.com/bcppbuilder/freecompiler/

A versão para Linux dessa biblioteca foi compilada com o compilador GNU, que acompanha qualquer distribuição do sistema operacional.

6.10.2 Bibliotecas

As bibliotecas utilizadas para implementar o sistema são listadas na Tabela 6.5.

Tabela 6.5 - Bibliotecas Java utilizadas na implementação

Fonte, bibliotecas utilizadas e endereço	Finalidade
Sun Microsystems, Inc. ▪ JavaBeans™ Activation Framework 1.0.1 ▪ JavaMail™ API 1.2 http://java.sun.com/products/javamail/	Envio de mensagens de e-mail para notificação do administrador
OpenNMS ▪ JoeSNMP http://www.opennms.org/files/releases/joeSNMP/	Comunicação SNMP
Apache Software Foundation ▪ Xerces Java Parser 1.4.3 http://www.apache.org/	Interpretação de arquivos XML

6.10.3 Editores

Para criar a versão inicial da MIB de gerência de energia, utilizou-se o programa MibEditor, da AdventNet, Inc. Uma versão de avaliação pode ser obtida no seguinte endereço:

http://www.adventnet.com/products/javaagent/

Para criar e editar a versão inicial do arquivo de configuração, utilizou-se uma versão de avaliação do programa XMLSpy, da Altova, Inc. O programa está disponível para *download* no seguinte endereço:

http://www.xmlspy.com/download.html

Ambos os programas possuem um tempo limite para utilização. Como os arquivos não eram muito complexos, depois de ultrapassado o limite, foram utilizados editores de texto comuns para modificá-los.

6.10.4 Simuladores e outras ferramentas SNMP

Para simular o comportamento dos agentes e também dos *no-breaks*, foram utilizados os seguintes programas:

- *Simulation Toolkit* (versão de avaliação), AdventNet, Inc.
  http://www.adventnet.com/products/simulator/
- *MIB Browser Professional Edition* (versão de avaliação), MG-SOFT Corp.
  http://www.mg-soft.com/download.html
- *NET-SNMP Tools* (ferramentas gratuitas), NET-SNMP Project
  http://net-snmp.sourceforge.net/

O simulador SNMP que é parte integrante do *Simulation Toolkit* da AdventNet é uma ferramenta extremamente útil para testes. É altamente configurável e de fácil operação, bastando importar o arquivo da MIB do agente que se pretende simular para criar um ambiente de simulação básico.

Dadas as restrições de uso do simulador, duas outras ferramentas foram bastante úteis na simulação de eventos dos agentes e dos *no-breaks*: as ferramentas de linha de comando que integram o pacote da MG-SOFT, particularmente a de envio de *traps*, e a ferramenta equivalente do projeto NET-SNMP. Com essas ferramentas, foi possível criar pequenos *scripts* para os sistemas Windows e Linux, com funcionalidades equivalentes, que permitiam simular diferentes eventos dos agentes e dos *no-breaks*.

A Figura 6.20 mostra um exemplo desses comandos, usando a ferramenta da MG-SOFT, que serve para enviar uma `trapAliveMsg` (OID 1.3.6.1.3.1.6.6) para o endereço IP local (127.0.0.1), na porta 9162, contendo a variável anexa `globalState` (1.3.6.1.3.1.1.2) com o valor `"on"`.

```
sendtrap 127.0.0.1 -r9162 -e1.3.6.1.3.1.6.6 -o1.3.6.1.3.1.1.2 -mo"on"
```

Figura 6.20 - Exemplo de comando para simulação

6.11 Outra documentação

Além do texto desta dissertação, uma documentação completa para as classes Java que compõem o sistema foi gerada automaticamente através da ferramenta `javadoc`. Esta documentação está anexada ao trabalho em formato digital, no Apêndice D, e também pode ser encontrada no seguinte endereço da Web:

http://www.inf.ufrgs.br/~pollo/netpower/javadoc/index.html

Além da documentação `javadoc`, um conjunto de diagramas de classes UML pode ser encontrado no Apêndice B.

7 Testes e investigações

Foram realizados diversos testes e experimentos em laboratório com o objetivo de verificar o funcionamento da aplicação de gerência conforme o projeto, e também com a intenção de validar o próprio projeto. Esses testes ajudaram a descobrir pontos fracos do sistema e erros de implementação, que puderam então ser corrigidos. Adicionalmente, foram obtidas algumas medidas empíricas do ganho potencial proporcionado pelo uso do sistema.

Além disso, a fase de testes foi acompanhada de intensa investigação do suporte efetivo à gerência de energia em equipamentos e sistemas operacionais, o que permitiu estabelecer uma visão mais clara dos recursos que funcionam na prática e dos problemas que eventualmente impedem o seu funcionamento.

As seções seguintes trazem detalhes sobre esses testes e investigações, e considerações sobre os resultados obtidos.

7.1 Ambiente de testes

A grande maioria dos testes e experimentos apresentados neste capítulo foi realizada no Laboratório de Convênio CP Eletrônica - UFRGS, do Instituto de Informática da UFRGS. As características do ambiente de teste configurado no Laboratório são descritas a seguir.

7.1.1 Rede

O Laboratório possui uma pequena rede Ethernet composta de seis PCs e que opera nas velocidades de 10 e 100 Mbps, dependendo do modelo de adaptador de rede dos computadores envolvidos na troca de dados (as máquinas são conectadas através de um *switch*). Os computadores da rede são listados na Tabela 7.1.

Tabela 7.1 - Equipamentos da rede do Laboratório de Convênio CP Eletrônica - UFRGS

Nome	Processador	Memória RAM	Disco rígido	Sistemas operacionais	OSPM[1]
Degas	Pentium Pro 200 MHz	128 MB	1,6 GB	Mandrake Linux 8.0 / Windows NT 4.0	✗
Klimt	Pentium II 400 MHz	64 MB	3,4 GB	Conectiva Linux 8.0 / Windows 2000 Professional	✓
Rembrand	Pentium III 500 MHz	128 MB	6,4 GB	Conectiva Linux 8.0 / Windows 2000 Professional	✓
Renoir	Pentium III 550 MHz	128 MB	10 GB	Windows 2000 Server	✓
Rubens	Pentium Pro 200 MHz	32 MB	1,6 GB	Windows 98	✗
Vincent	Pentium Pro 200 MHz	64 MB	8,4 GB	Conectiva Linux 4.2	✗

[1] O critério, aqui, é se o computador pode ser colocado em um modo de baixo consumo ou desligado pelo sistema operacional. Isso requer não só suporte do SO e do BIOS, mas também que o computador seja equipado com um tipo adequado de fonte (ATX).

Conforme se pode observar na tabela, apenas três dos PCs da rede podem ser efetivamente colocados em modos de baixo consumo, ou desligados, pelo sistema operacional e, conseqüentemente, pelo sistema de gerência.

Dois destes três computadores possuem placas de rede que suportam a tecnologia *Wake-on-LAN*, possibilitando ao sistema despertá-los remotamente. As placas de rede em questão são fabricadas pela 3Com (modelo 3C905C). Maiores detalhes da configuração dos PCs usados para testes são apresentados na seção 7.5.

7.1.2 No-breaks

O Laboratório de Convênio dispõe de dois *no-breaks* da linha Breakless da CP, modelos 1220 e 1650, com potências de 2 e 5 kVA, respectivamente. Os dois *no-breaks* compartilham o mesmo grupo de 12 baterias, de 12 volts DC cada, totalizando 144 V de tensão DC.

O Instituto de Informática dispõe ainda de dois *no-breaks* da linha Top da CP. Estes modelos são mais adequados para o monitoramento das condições de energia da rede porque suportam um protocolo de gerência completo, ao contrário dos modelos da linha Breakless, que são capazes de informar um número muito pequeno de eventos. Os modelos Top são usados para abastecer a maior parte da rede do Instituto, de modo que não são adequados para a execução de alguns tipos de testes (como forçar a operação em bateria, por exemplo), sob risco de interromper o fornecimento de energia da rede.

Além disso, como será discutido adiante, o *software* usado para monitorar os *no-breaks* do Instituto, atualmente, não é adequado para a interação com o sistema de gerência implementado. Por estes motivos, os testes com modelos Top foram realizados na própria sede da CP Eletrônica S.A.

7.2 Aplicação de gerência

Uma diretriz presente ao longo de todo o desenvolvimento do trabalho foi a de construir uma implementação utilizável, com razoável estabilidade e completeza. Neste sentido, durante a fase de testes, procurou-se extrapolar as limitações impostas por ambientes 100% simulados, que tendem a mascarar os erros encontrados em situações reais de funcionamento. Dentro do possível, o sistema foi testado em condições de funcionamento próximas do real, utilizando-se *no-breaks* para a detecção de alterações no fornecimento de energia e um agente simplificado para controle dos PCs.

Por outro lado, é claro que não se pode desprezar a validade da simulação na execução de testes. Ela foi um recurso amplamente utilizado, principalmente nos estágios iniciais da fase de testes. As ferramentas usadas para simulação de agentes e *no-breaks* foram listadas no capítulo anterior (seção 6.10.4). Através dessas ferramentas, foi possível detectar e corrigir pequenos erros de implementação antes de utilizar equipamentos reais em testes mais avançados.

As diversas etapas do processo de teste da aplicação de gerência são descritas a seguir.

7.2.1 Detecção de eventos e despacho de ações

Os primeiros testes da aplicação de gerência tiveram o objetivo de validar o que pode ser considerado seu "núcleo lógico": o mecanismo de detecção de eventos e disparo das ações correspondentes. Como se poderia esperar, a implementação partiu da leitura e conversão do

arquivo de configuração para sua representação interna, e avançou a partir daí. Inicialmente, foram implementados dois dos componentes do gerente: o temporizador de eventos programados (ver 6.4) e o despachante de ações (ver 6.5).

Conforme foi explicado no capítulo 6, as ações remotas são enviadas por este elemento despachante único, independente do tipo de evento que gera seu disparo. Além disso, o mecanismo de associação de ações também é idêntico para os dois tipos de eventos (de energia e programados). Por isso, uma vez que o temporizador de eventos programados havia sido implementado, foi possível testar a correta interpretação dessas associações pelo despachante. Os resultados foram positivos, demonstrando a viabilidade do mecanismo.

7.2.2 Testes básicos de comunicação via SNMP

Depois de garantir que a lógica da parte central do sistema funcionava corretamente, o próximo passo foi implementar os componentes responsáveis por receber e tratar as notificações SNMP enviadas tanto pelos agentes quanto pelos *no-breaks*. Também era necessário habilitar o despachante de ações a requisitar, efetivamente, a execução das ações remotas pelos computadores gerenciados, através do envio das mensagens SNMP correspondentes.

Tendo completado esta fase da implementação, efetuaram-se testes da habilidade de comunicação via SNMP da aplicação de gerência. Para testar a recepção de notificações, foi suficiente gerá-las artificialmente (através de *scripts*) a partir de um computador qualquer da rede. Para testar a capacidade da aplicação de obter e alterar os valores da MIB do agente e das MIBs de gerência de *no-break* implementadas nos diversos monitores, utilizou-se, inicialmente, uma versão de avaliação do simulador SNMP da AdventNet [ADV2002]. Mais tarde, com a implementação de um agente-protótipo (ver seção 7.3), foi possível repetir e estender os testes da manipulação da MIB de gerência de energia do agente.

A pilha SNMP utilizada (JoeSNMP, apresentada no capítulo anterior) demonstrou ser confiável, não apresentando problemas de compatibilidade ou erros aparentes na implementação do protocolo.

7.2.3 Monitoramento de no-breaks

O monitor compatível com a MIB da CP Eletrônica utilizada para supervisão *no-breaks* da linha Breakless[1] foi testado diretamente com o agente disponibilizado pela empresa, no Laboratório de Convênio da UFRGS. Essa MIB, implementada pelo agente que integra o *software* CP-Ctrl 2.0, é bastante simples, devido à limitação de sinalização dos próprios *no-breaks*. Não é possível intervir na operação dos dispositivos através do agente SNMP, nem obter os valores de seus parâmetros de operação. Para a informação dos eventos de energia monitorados pelo sistema de gerência, entretanto, essa MIB apresenta uma estrutura quase ótima, definindo notificações que permitem identificar facilmente as situações de alteração no fornecimento com as quais o sistema se preocupa. Em decorrência disso, o monitor correspondente pôde ser facilmente implementado e testado. E demonstrou funcionar extremamente bem.

[1] Os modelos Breakless são do tipo comumente chamado de "contato-seco", não-microprocessados e, portanto, incapazes de se comunicar através de um protocolo de gerência. Eles possuem, entretanto, um sistema básico de sinalização de eventos através de linhas de uma porta serial. Esses sinais são recebidos pelo agente fornecido pela empresa e convertidos para as notificações SNMP correspondentes, definidas na MIB que acompanha o *software*.

Já o monitor compatível com a MIB da CP implementada no *software* CP-Monitor Net, usado para gerenciar modelos da linha Top (micro-processados), foi testado na própria empresa. Conforme mencionado anteriormente, o Instituto de Informática da UFRGS possui dois aparelhos desta linha, cujos modelos são capazes de informar uma série de valores a respeito do funcionamento do próprio *no-break e* do estado das baterias. No entanto, apenas um deles estava sendo monitorado durante a execução deste trabalho, e pela versão *stand-alone* do *software* de monitoramento, que serve apenas para supervisão local. A versão "Net" do CP-Monitor, que pode ser instalada em sistemas Windows NT ou compatíveis, faz o trabalho de conversão do protocolo serial suportado pelo *no-break* para o protocolo SNMP, segundo uma MIB específica. O computador utilizado para a supervisão, no Instituto, tinha uma versão inadequada de sistema operacional e, portanto, não tinha a versão servidora do programa de supervisão da CP[1].

Na sede da empresa, foi feita, inicialmente, uma investigação do comportamento do agente SNMP incluído no CP-Monitor Net, através de um *"MIB browser"*. Esta investigação teve por objetivo determinar as adequações necessárias para tornar o monitor implementado (ver seção 6.3.2) efetivamente compatível com o *software* da CP. Foi possível constatar que a implementação do agente divergia em alguns pontos das definições contidas na própria MIB (que havia servido como base para a implementação do monitor *NetPower*), bem como da especificação do protocolo SNMP em relação ao acesso de dados tabulares. Estes pontos foram discutidos com funcionários da empresa e com o responsável pela implementação do *software*, a título de sugestão para possíveis correções. Depois de fazer as devidas alterações no módulo monitor original, uma nova série de testes foi executada na CP, com resultados positivos.

Finalmente, o monitor compatível com a MIB padrão para gerência de *no-breaks* (UPS-MIB, RFC 1628), foi testado inteiramente através de *scripts* de simulação, por não haver no Instituto equipamentos que implementem esta MIB. Intuitivamente, pode-se concluir que, assim como no caso da MIB da CP Eletrônica, mencionado acima, testes com um agente real provavelmente indicariam a necessidade de alterações neste módulo. Embora a UPS-MIB seja razoavelmente bem documentada através das descrições dos objetos da árvore, que fazem parte do arquivo fonte, é impossível dizer, com certeza, que o monitor funcionaria completamente de acordo com o esperado. Além disso, também é possível que diferentes implementações da MIB sejam baseadas em diferentes interpretações da documentação e, por isso, comportem-se de maneiras diferentes.

7.2.4 Despertar remoto dos computadores gerenciados

Um recurso que mereceu atenção especial durante a etapa de testes foi o da capacidade do sistema de gerência de despertar os computadores remotamente. Como foi explicado anteriormente, este recurso é uma peça fundamental para a centralização do processo de gerência de energia, já que, sob o ponto de vista da administração da rede, a capacidade de ligar os computadores à distância é tão importante quanto a capacidade de desligá-los.

Foram efetuados dois testes simples de verificação do funcionamento do mecanismo de despertar implementado na aplicação de gerência: o envio do *magic packet* para computadores localizados fisicamente na mesma sub-rede do *host* do gerente, e para computadores localizados em outras sub-redes, de modo que o pacote passasse necessariamente por um ou mais roteadores. Nos dois casos, o processo funcionou sem problemas.

---

[1] Foi requisitado à administração dos laboratórios do Instituto que se instalasse um computador adequado à supervisão de qualquer dos dois *no-breaks* Top pelo CP-Monitor Net. Este equipamento está sendo providenciado, mas até a data de entrega do trabalho ainda não estava disponível.

7.3 Agente-protótipo

Para testar, efetivamente, a aplicabilidade do sistema proposto, o *software* gerente, sozinho, não seria suficiente. Era necessário ter um agente que, por mais simples ou específico, permitisse à aplicação de gerência atuar sobre os equipamentos do laboratório, executando as tarefas básicas de desligamento e entrada em estado de baixo consumo.

Assim, buscando levar o sistema ao estágio de maturidade desejado, implementou-se um agente-protótipo, destinado ao sistema operacional Microsoft Windows 2000. Ainda que bastante incompleto, se comparado com o conjunto de funções especificadas no capítulo 5, este protótipo permitiu transpor a barreira das simulações, como era desejado, e observar o sistema completo em atividade.

A principal razão para a escolha do Windows 2000 como plataforma de desenvolvimento foi que o suporte à API de gerência de energia é bastante limitado (ou até inexistente) em versões anteriores dos sistemas operacionais da Microsoft. Esta API fornece as funções necessárias para colocar o computador em estados de baixo consumo e define as mensagens do SO destinadas a informar as aplicações da ocorrência de eventos de gerência de energia – alterações no estado de consumo do sistema ou no modo operacional de um dispositivo ou do computador inteiro.

O agente foi implementado com uma aplicação Windows simples, a partir do esqueleto básico provido por um programa-exemplo do tipo *"Hello World"*. A única indicação visível da execução da aplicação é uma pequena janela que mostra o texto *"NetPower Agent"*, como se pode ver na Figura 7.1. Não há quaisquer outros controles de configuração. O endereço IP e a porta do gerente são fornecidos ao programa como parâmetros pela linha de comando.

Figura 7.1 - Tela do agente-protótipo

Inicialmente, fez-se uma tentativa de implementar o protótipo como uma aplicação de console, o tipo mais simples de programa que se pode escrever para um sistema Windows. Entretanto, o Windows 2000 possui modelos de eventos distintos para diferentes tipos de aplicações. O conjunto de eventos definidos para aplicações de console é muito pequeno e restrito, e não inclui os eventos de gerência de energia, ao contrário do que acontece no modelo destinado às aplicações gráficas. Do mesmo modo, serviços do Windows (provavelmente a maneira mais adequada de se implementar um agente de gerência de energia) possuem um terceiro modelo de eventos. O modelo usado para serviços inclui os eventos de gerência de energia, mas a implementação e a depuração desse tipo de aplicações é muito mais complexa do que a de aplicações do nível de usuário – desnecessariamente complexa para efeito de testes.

7.3.1 Funções implementadas

Foram implementadas as seguintes funções no agente-protótipo:

- detecção de mensagens do sistema operacional relacionadas com a gerência de energia;
- envio de notificações (*traps*) correspondentes às principais alterações de consumo de energia (ligação, entrada em modo de espera ou hibernação, desligamento, despertar);

- recepção e processamento de mensagens do gerente, com capacidade de execução de três ações básicas: desligamento, entrada em modo de espera (*standby*) e hibernação.

As mensagens e eventos do sistema operacional tratadas pelo agente são listadas na Tabela 7.2. Estas mensagens e eventos, bem como todas as funções de gerência de energia da API, estão documentadas no *Platform SDK* da Microsoft. Esta documentação está disponível na "biblioteca digital" da MSDN (*Microsoft Developer Network*) [MIC2002a].

Tabela 7.2 - Mensagens do Windows tratadas pelo agente

Mensagem / evento	**Significado (→) / comportamento do agente (←)**	
`WM_DESTROY`	→	Aplicação encerrada
	←	Notifica o gerente com `trapImminentShutdown` (em teoria, a aplicação deveria rodar sempre – se é encerrada, é porque o sistema está sendo desligado)
`WM_POWERBROADCAST`		Alteração nas condições de consumo de energia (ver eventos abaixo)
▪ `PBT_APMQUERYSUSPEND`	→	Permissão para entrar no modo de espera
	←	Concede permissão e suspende a comunicação
▪ `PBT_APMSUSPEND`	→	O sistema está entrando em modo de espera
	←	Notifica o gerente com `trapImminentSleep` e suspende a comunicação se necessário
▪ `PBT_APMRESUMECRITICAL` ▪ `PBT_APMRESUMESUSPEND` ▪ `PBT_APMRESUMEAUTOMATIC`	→	O sistema voltou à atividade
	←	Reinicia a comunicação e notifica o gerente com `trapWakeUp`
`WM_QUERYENDSESSION`	→	Permissão para encerrar o sistema
	←	Concede permissão
`WM_ENDSESSION`	→	Encerramento do sistema
	←	Notifica o gerente com `trapImminentShutdown` e finaliza a execução

Além das notificações listadas acima, o agente também envia uma `trapPowerOn` quando é carregado, sinalizando que o computador foi ligado (teoricamente, e conforme a especificação do agente, sempre que o computador está ligado o agente está rodando).

7.3.2 Detalhes de implementação

O agente foi implementado em C++ e compilado com o compilador gratuito da Borland [BOR2002]. Para a comunicação via SNMP, foram utilizadas duas bibliotecas gratuitas, SNMP++ e Agent++ [FOC2002]. A biblioteca SNMP++ original, destinada à programação de aplicações de gerência, foi implementada pela Hewlett-Packard [MEL94]. Essa biblioteca foi estendida pelo mantenedor atual da API Agent++ para facilitar a implementação de agentes.

O responsável pelas extensões da API SNMP++ e criação da API Agent++, o programador alemão Frank Fock, disponibiliza também uma ferramenta (AgentGen) para a geração automática da estrutura básica do código de um agente a partir da MIB que ele deve implementar. Essa ferramenta facilita enormemente a programação do agente, permitindo ao programador concentrar-se na parte lógica de seu funcionamento, ao abstrair os detalhes de tratamento da MIB e do próprio protocolo de comunicação.

O protótipo resultante possui uma única entrada (identificada pelo nome "global") na tabela de dispositivos do equipamento (componentTable). Esta entrada corresponde ao sistema inteiro. Quando o agente recebe uma requisição Set destinada ao objeto que corresponde ao estado atual do sistema (componentCurrentState), ele verifica se é um dos três estados suportados – "sleep", "hibernate" e "off". Se for um destes valores, faz a chamada à função apropriada da API do Windows para colocar o computador em modo de espera, hibernação ou desligá-lo, respectivamente.

O agente inicializa os objetos do grupo agentAttributes com valores padrão. O objeto globalState, do grupo devAttributes, é inicializado com o valor "on" e atualizado no decorrer da execução do agente para refletir as alterações no estado do equipamento. Este objeto está vinculado ao componente "global" descrito no parágrafo anterior – as requisições para o objeto globalState são efetivamente repassadas para o objeto correspondente na tabela. Os outros objetos da MIB, particularmente os do grupo devCapabilities, têm acesso restrito à leitura e papel informativo, e por isso não foram implementados. Os objetos do grupo sysInteraction também não foram implementados.

7.3.3 Avaliação

A implementação do agente-protótipo permitiu verificar não apenas a praticabilidade do modelo de comunicação proposto para o sistema de gerência, validando os mecanismos de atualização do mapa da rede e de controle dos computadores através de SNMP (descritos no capítulo 4), como também a da arquitetura concebida para os agentes, que utiliza o conceito de OSPM como premissa e, portanto, depende do sistema operacional para a execução da gerência de energia (conforme descrito no capítulo 5).

O modelo genérico de funcionamento especificado para os agentes, que propõe uma separação entre a execução propriamente dita das ações e o posterior envio das notificações que indicam sucesso ou falha da operação (disparadas assincronamente a partir de sinalização recebida do SO), mostrou-se perfeitamente adequado ao modelo de eventos do Windows 2000, sistema-alvo da implementação. Detalhes deste modelo, bem como outras informações sobre o suporte de diferentes sistemas operacionais à gerência de energia, serão apresentados a seguir.

7.4 Suporte dos sistemas operacionais para PCs à gerência de energia

Um dos resultados mais imediatos das investigações realizadas durante a fase de testes do trabalho foi determinar o "estado da arte" de duas famílias de sistemas operacionais para PCs (Linux e Windows) no que diz respeito ao suporte à gerência de energia. Constatou-se que esse suporte ainda é bastante limitado e desuniforme. Nas seções seguintes, são discutidos alguns pontos que sustentam essa constatação, acompanhados de breves descrições dos recursos de gerência de energia encontrados nos dois tipos de sistema.

7.4.1 Metodologia

Foram observados os seguintes itens dos sistemas operacionais analisados:

- suporte aos padrões APM e/ou ACPI;
- capacidade de detectar corretamente o padrão de gerência de energia implementado pelo *firmware* do computador;

- capacidade de detectar corretamente o suporte de dispositivos à gerência de energia, com ênfase para o adaptador de rede;
- capacidade de colocar o computador em um ou mais modos de baixo consumo de energia;
- capacidade de retomar o processamento corretamente ao retornar de um modo de baixo consumo;
- capacidade de colaboração com as aplicações para tomada de decisões referentes à gerência de energia;
- interação correta com *drivers* de dispositivos durante mudanças de estado;
- disponibilidade de mecanismos de configuração de políticas/esquemas de gerência de energia.

7.4.2 Windows

Nos sistemas da família Windows, da Microsoft, suporte relativamente completo a funções de gerência de energia aparece a partir do Windows 2000 [MIC2002b]. Em versões anteriores do sistema (Windows 95, 98 e NT), esse suporte é bem mais restrito. Com o Windows 2000, a Microsoft deu início à implantação de sua estratégia *OnNow*, que leva para o sistema operacional o controle da gerência de energia (modelo chamado de OSPM, como foi explicado anteriormente). O padrão ACPI é a peça-chave da implementação de OSPM nos sistemas Windows, e parte das funções de gerência de energia disponíveis nas versões mais recentes do sistema dependem dos recursos providos por *hardware* ACPI-compatível.

Arquitetura de gerência de energia do sistema

O modelo de eventos dos sistemas Windows é razoavelmente simples, baseado no registro, pela aplicação, de uma função de *callback* encarregada do tratamento das mensagens enviadas pelo sistema. Esta função possui uma assinatura (declaração) pré-definida, que especifica o retorno de um valor inteiro. Isto permite que a interação entre o sistema operacional e as aplicações seja bidirecional: não só as aplicações podem ser notificadas da ocorrência de eventos como também responder a requisições (como pedidos de permissão, por exemplo).

Na área específica de gerência de energia, o sistema faz a difusão de mensagens (do tipo `WM_POWERBROADCAST`) para todas as aplicações, avisando de alterações relevantes no estado da máquina, como a entrada em modo de baixo consumo ou o reinício da atividade. Dependendo do evento que dispara a entrada do computador num modo de espera, as aplicações podem ou não ser consultadas antes da operação ser executada. Neste caso, o sistema só prossegue se todas as aplicações respondem afirmativamente à consulta.

Além da mensagem `WM_POWERBROADCAST`, a API do Windows também define uma função (`SetThreadExecutionState`) que permite às aplicações informar o sistema operacional de quais dispositivos ou recursos elas necessitam para funcionar corretamente. Isso permite ao sistema otimizar a detecção de inatividade do computador e, conseqüentemente, a economia de energia. Por outro lado, permite também que as aplicações evitem a entrada do computador em modos de baixo consumo, se isto for necessário. Basicamente, o Windows não inicia uma transição para estados de baixo consumo enquanto houver atividade das aplicações. No entanto, nem todos os tipos de atividade podem ser detectados. Por exemplo, uma aplicação multimídia normalmente requer que o monitor permaneça ligado durante toda a apresentação, mas se ela for bastante longa e o usuário permanecer inativo durante a exibição, o sistema desligará o monitor – a não ser que a aplicação tenha indicado que ele deve permanecer ligado, através de uma chamada a `SetThreadExecutionState`.

Finalmente, a API do Windows também provê funções que permitem às aplicações obter e alterar o estado de consumo de energia de dispositivos (`GetDevicePowerState`) ou do sistema inteiro (`GetSystemPowerStatus`, `SetSystemPowerState`), bem como consultar e modificar as políticas e esquemas de gerência de energia definidos pelo usuário (`GetCurrentPowerPolicies`, `GetActivePwrScheme`, `SetActivePwrScheme`, etc.). Utilizando esta categoria de funções, as aplicações podem extrapolar o papel de coadjuvantes no processo de gerência de energia, e coordená-lo diretamente – o que vem a calhar para a implementação de um agente como o que foi proposto neste trabalho.

Documentação

Embora tenha sido importante para elucidar o funcionamento básico dos mecanismos de gerência de energia do Windows e seja bastante extensa, a documentação disponível na MSDN é vaga em muitos pontos, o que dificulta antecipar, em detalhe, o comportamento do sistema operacional no que se refere à gerência de energia. Isso é particularmente notável na documentação de algumas chamadas de sistema, cujo efeito é descrito em linhas gerais, abrindo espaço para muita especulação e incerteza[1].

Observações

O Windows possui, sem dúvida, o melhor suporte à gerência de energia disponível entre os sistemas operacionais disponíveis. No entanto, notam-se incompatibilidades entre o sistema operacional, *drivers* de dispositivos e *hardware/firmware*, que podem impedir a configuração adequada dos computadores para gerência de energia (ver seção 7.5.4).

Aparentemente, o grande apelo comercial da Microsoft força os fabricantes de *hardware* a implementarem soluções específicas para compatibilizar seus produtos com as peculiaridades dos sistemas Windows. Ainda que a própria Microsoft tenha sido co-autora da especificação ACPI, por exemplo, é notório seu histórico de sugerir pequenos "consertos" aos fabricantes de *hardware* para que a compatibilidade com o Windows seja garantida – o que costuma atrapalhar o funcionamento correto dos dispositivos em outros sistemas operacionais.

7.4.3 Linux

O suporte à gerência de energia no Linux ainda é bastante imaturo e, por isso, incompleto. Já existem *drivers* que fazem a ponte entre o *kernel* e o *firmware* APM ou ACPI, mas há um grande número de implementações de BIOS incompatíveis.

Inicialmente destinado quase que exclusivamente a computadores portáteis (que podem depender constantemente de baterias e, portanto, são especialmente beneficiados com a redução do consumo de energia), a implementação do padrão APM é bem mais completa do que a de ACPI. É claro que não se pode negligenciar o fato de que a especificação ACPI é muito mais complexa, requerendo, portanto, maior esforço de implementação. De qualquer modo, as funções mais "visíveis" da gerência de energia (e, provavelmente, mais úteis), como colocar o computador em um modo de espera, por exemplo, ainda não estão disponíveis na implementação de ACPI. A implementação de APM, em contrapartida, suporta a colocação do computador em dois modos de espera diferentes (*standby* e *suspend*).

[1] Como o código do Windows é fechado, nestas situações não resta ao programador sequer a alternativa de investigar o próprio código-fonte, como se pode fazer no Linux.

Arquitetura de gerência de energia do sistema

Tanto a implementação de APM quanto a de ACPI no Linux são divididas em duas partes: o *driver* propriamente dito, que interage com o *kernel* do sistema operacional e com o BIOS do computador, e um *daemon*, que roda em espaço de usuário e se comunica com o *driver*. Os *daemons* de ambas as implementações são configurados para responder à detecção de eventos informados pelo *driver* através da execução de ações (comandos) – uma arquitetura semelhante à implementada na aplicação de gerência proposta neste trabalho. Não há, entretanto, um mecanismo genérico de notificação das aplicações (como talvez fosse possível através de sinais POSIX, por exemplo). A configuração dos *daemons* permite a associação de um único comando à ocorrência de cada evento. Hipoteticamente, este comando poderia funcionar como um *proxy* e repassar as notificações para outras aplicações, mas isso dependeria não só da própria implementação deste *proxy* como da correta configuração do *daemon*, e ainda da definição de uma interface padrão de notificação e de sua implementação por todas as aplicações.

Curiosamente, existe no Linux uma API genérica de gerência de energia, mas destinada exclusivamente aos *drivers* de dispositivos. Essa API[1] permite que os *drivers* registrem uma função de *callback* com o *kernel*, sendo assim notificados dos eventos de entrada (*suspend*) e retorno (*resume*) de modos de consumo reduzido. Segundo a documentação, as implementações de padrões de gerência de energia, como APM e ACPI, devem utilizar essa API genérica para notificação dos *drivers*.

Como se poderia esperar, também não existe no Linux, ainda, a possibilidade de cooperação entre aplicações e sistema operacional nas decisões de gerência de energia. A API de *drivers* mencionada acima permite que qualquer um deles vete a operação de entrada em modo de espera, de maneira muito semelhante ao que acontece no Windows durante a difusão das mensagens de gerência de energia para as aplicações – se uma delas negar o pedido de autorização para entrar no modo de espera, o sistema aborta a operação. Infelizmente essa capacidade de interação com as aplicações ainda não foi implementada no Linux. Segundo um dos responsáveis pela implementação do padrão ACPI no Linux, entretanto, este suporte estará sendo adicionado em versões futuras do sistema [GRO2002].

Documentação

A documentação referente à gerência de energia no Linux é bastante escassa, e está concentrada no diretório que contém o código-fonte do *kernel*, e em algumas páginas de manual (*manpages*) e *"HOWTOs"*:

- Arquivos de documentação do código-fonte do *kernel*:
  - `/usr/src/linux/Documentation/Configure.help`;
  - `/usr/src/linux/Documentation/pm.txt`;
  - `/usr/src/linux/Documentation/power/pci.txt`;
- Páginas de manual:
  - `apm`, `apmd`, `apmsleep` e `xapm`;
  - `acpid`;
- *HOWTOs*:
  - Linux ACPI-HOWTO [GLE2001];

[1] As funções da API são definidas no arquivo /usr/src/linux/include/linux/pm.h, e implementadas no arquivo /usr/src/linux/kernel/pm.c.

- Linux Laptop-HOWTO [HEU2000];
- Linux Ecology-HOWTO [HEU2000a].

7.4.4 Comparação: Linux x Windows

A Tabela 7.3, a seguir, contém um resumo comparativo dos recursos de gerência de energia observados no Linux e no Windows.

Tabela 7.3 - Resumo de recursos de energia do Windows e do Linux

	Sistema	
Recurso	**Linux**	**Windows**
Suporte a APM	☺	☺
Suporte a ACPI	😐	☺
Detecção da implementação do BIOS	😐	☺
Detecção de recursos de gerência de energia de dispositivos	😐	😐
Modo de espera (*standby, suspend to RAM*)	😐 (APM)	☺
Hibernação (*suspend to disk*)	☹	☺
Retorno de modos de baixo consumo	☹	☺
Colaboração com aplicações	☹	☺
Interação com *drivers* de dispositivos	☺	☺
Mecanismos de configuração de políticas/esquemas de gerência de energia	😐 (APM)	☺

Legenda:
☺ Bom
😐 Razoável
☹ Ruim/inexistente
(APM) Apenas para APM

7.5 Suporte de hardware e firmware à gerência de energia

Como foi mencionado anteriormente, a gerência de energia num PC se dá através da colaboração de diversos elementos, dos quais os mais importantes são o sistema operacional e o *hardware*/*firmware* do equipamento. Assim, além de observar o comportamento do sistema operacional, foram feitos alguns testes com o objetivo de determinar o suporte de *hardware* e *firmware* à gerência de energia nos computadores do laboratório.

Foram considerados, primariamente, os recursos de gerência de energia de placas-mãe e placas de rede. A primeira categoria determina o suporte geral do PC à gerência de energia – se ele pode ser colocado em estados de baixo consumo (e quais são estes estados), se pode ser despertado por eventos externos, etc. A segunda determina se o PC pode ser despertado remotamente por uma aplicação de gerência.

7.5.1 Metodologia

Foram observados os seguintes itens nos equipamentos testados:

- placas-mãe:
  - suporte aos padrões APM e ACPI pelo *firmware*;
  - suporte a *Wake-on-LAN* (WOL);

  - presença de conector específico para *Wake-on-LAN*;
  - manutenção da energia para a placa de rede nas trocas para estados de baixo consumo ou desligamento do PC;
- placas de rede:
  - compatibilidade com ACPI;
  - suporte a despertar remoto (*remote wake-up*);
  - funcionamento do *driver*.

Para testar os itens listados acima, a seguinte seqüência de passos foi utilizada:

1) teste inicial: instalação "limpa" de um sistema operacional;
2) atualização do *firmware* (BIOS);
3) atualização do sistema operacional (*service packs*, *patches*, etc.);
4) atualização do *driver* do adaptador de rede;
5) instalação de um sistema operacional alternativo (mudando de Windows para Linux ou vice-versa).

A cada passo, foi observado o comportamento da máquina e, caso algum recurso não funcionasse corretamente, passava-se ao passo seguinte. Os passos 2 e 4 foram especialmente importantes para a solução de problemas, pois as versões originais dos *softwares* que acompanham dispositivos de *hardware* costumam apresentar deficiências, que podem ser corrigidas através de atualizações disponibilizadas pelos fabricantes.

7.5.2 Placas-mãe

A Tabela 7.4 apresenta as características observadas nas placas-mãe testadas. Os itens marcados com um ponto de interrogação não puderam ser determinados porque o equipamento não podia ser desmontado. Para as outras três placas, como se pode ver na tabela, foi possível observar que a energia para o adaptador de rede só era mantida quando o cabo WOL estava instalado. A função deste cabo é fornecer uma linha de energia para a placa de rede e permitir que ela sinalize a placa-mãe de um evento de *wake-up*. Segundo a documentação das placas de rede testadas, entretanto, se a placa-mãe for compatível com o padrão *PCI Local Bus Revision 2.2*, não há necessidade de usar o cabo porque a placa de rede recebe energia e sinaliza os eventos de *wake-up* através dos pinos do próprio conector PCI [TCO99]. Apesar dos manuais de todas as placas-mãe [INT99, INT99a, ASU2000] afirmarem que elas suportam o padrão PCI 2.2, as placas de rede não se mantinham ligadas quando o computador entrava num modo de baixo consumo se o cabo WOL não estivesse instalado.

Tabela 7.4 - Placas-mãe testadas

Configuração	**1) klimt** *	**2) rembrand** *	**3) renoir***	**4) benneton****
Placa-mãe	Intel SE440BX-2	Intel Desktop Board CC820	Asus P3V133	Dell CA81020A (chipset Intel 810E)
BIOS (fabricante e versão)	Phoenix BIOS Production Release 17.0	Intel/AMI BIOS P09-0015	Award BIOS Revision 1002	Intel/AMI BIOS Version A14
Suporte a APM e ACPI	Sim. Detectado corretamente no Windows e no Linux	Sim. Detectado corretamente no Windows. No Linux, o driver APM não reconhece o suporte do BIOS	Sim. Detectado corretamente no Windows e no Linux	Sim. Detectado corretamente no Windows e no Linux
Suporte a *Wake-on-LAN* (WOL)	Sim	Sim	Sim	Sim
Conector WOL	Sim	Sim	Sim	Sim
Cabo WOL instalado	Sim	Sim	Sim	?
Manutenção da energia para a placa de rede nas trocas de estado	Só com o cabo WOL instalado	Só com o cabo WOL instalado	Só com o cabo WOL instalado	?

**Laboratório de Convênio CP Eletrônica - UFRGS*
***Laboratório de Segurança*

7.5.3 Placas de rede e o processo de despertar remoto (*remote wake-up*)

Conforme foi mencionado anteriormente, a capacidade do computador de "acordar" a partir de um sinal enviado pela rede depende do suporte do *hardware* ao padrão *wake-on-LAN* ou algum mecanismo semelhante. Entretanto, o *hardware* é apenas uma pequena peça do processo de despertar, como será descrito brevemente a seguir.

Para verificar o funcionamento do processo na prática, foram testados dois modelos de placas de rede compatíveis com o padrão, que serão apresentadas adiante.

O processo de despertar remoto

A primeira etapa no processo de despertar remoto consiste, evidentemente, no desligamento do computador ou sua colocação num modo de baixo consumo. Associada à entrada do computador num destes modos de "dormência", está a habilitação de um modo especial de operação do adaptador de rede, o que geralmente é feito através da alteração de um bit num registrador do dispositivo. Neste modo de operação, o adaptador passa a varrer todos os pacotes recebidos à procura do padrão de dados apropriado (ver seção 6.5.4) e, caso detecte a sua ocorrência, notifica a placa-mãe com um sinal de *wake-up*.

A fase de configuração do adaptador de rede depende da correta implementação dos padrões de gerência de energia pelo sistema operacional e, ainda mais importante, pelo *driver* do dispositivo. Se ele não for colocado no estado apropriado antes do computador entrar num modo de baixo consumo ou ser desligado, o processo de despertar não funciona.

Quando o computador é finalmente desligado ou entra num modo de baixo consumo, a energia é cortada para a maior parte dos dispositivos do sistema. Obviamente, o adaptador de rede precisa continuar ativo para realizar a varredura dos pacotes e, portanto, não pode ser desligado. Isto depende do fornecimento de corrente pela placa-mãe, o que pode ser feito através do barramento PCI ou de um cabo especial, conforme descrito anteriormente (seção 7.5.2).

Neste ponto, se o adaptador houver sido corretamente habilitado para operar no modo especial de varredura de pacotes e continuar recebendo energia da placa-mãe, o computador encontra-se num estado passível de ser despertado remotamente. Quando, e se, a placa de rede detectar um *magic packet*, ela propaga o sinal de *wake-up* pela placa-mãe, eventualmente reativando o processador, que volta a executar algum estágio do sistema operacional (mesmo que seja partindo de um "*boot* frio", reinicializando completamente o sistema). Esta fase do processo é completamente dependente da implementação correta das especificações de gerência de energia pelo *firmware* (BIOS) [INT96, COM2000].

Testes das placas de rede

Foram testados os seguintes modelos de placa de rede:

- 3Com EtherLink 10/100 PCI for Complete PC Management (3C905C-TX-M);
- Intel Pro/100+.

Ambas as placas possuem o recurso de *remote wake-up* e são compatíveis com a especificação ACPI. Os dois modelos também são equipados com o conector WOL de 3 pinos, que permite conectá-las a placas-mãe compatíveis, como as listadas na seção anterior.

Os dois modelos testados permitiram despertar o computador em pelo menos uma situação, como depois de ele ter sido colocado em modo de espera (*standby*) ou desligado. Por outro lado, independente da configuração do computador ou do sistema operacional instalado, não foi possível despertar a máquina de todos os estados a partir dos quais a operação de *wake-up* deveria ser possível. Isso foi particularmente notável quando a máquina era colocada no estado de hibernação pelo Windows. Não foi possível despertar nenhum dos computadores testados a partir deste estado.

Esse tipo de problema não pode ser atribuído exclusivamente ao funcionamento dos adaptadores de rede. Como foi visto anteriormente, o adaptador de rede é responsável por uma parte muito pequena, apesar de essencial, do processo de despertar, e sua configuração para operar no modo de varredura de pacotes é bastante simples. Assim, é razoável deduzir que, tendo notificado corretamente o sistema a partir de um dado estado de "dormência" do computador, o adaptador de rede faria o mesmo a partir de qualquer outro estado possível. Em outras palavras, para o adaptador de rede, não faz diferença se o computador está desligado ou em algum modo de baixo consumo (ou em qual modo específico). Ele não possui nenhuma percepção do estado do resto do equipamento, apenas opera conforme foi configurado – neste caso, varrendo os pacotes recebidos. Portanto, é razoável assumir que o problema não é decorrente da operação da placa de rede, mas de falha de algum outro componente do sistema envolvido no processo, possivelmente o *driver* ou o sistema operacional (ao não colocarem o dispositivo no modo de operação correto antes da mudança de estado do computador), ou ainda o *firmware* da placa-mãe (ao omitir-se de reiniciar o computador ao ser notificado pela placa de rede).

Por outro lado, segundo o autor do *driver* Linux para a placa 3Com, Donald Becker, outra possibilidade é a de incompatibilidade entre a placa de rede e a placa-mãe. Segundo Becker, as primeiras implementações de *wake-on-LAN* não eram padronizadas, e o tipo de sinal

usado para notificação (nível, pulso, etc.) variava. À primeira vista, como foi mencionado no parágrafo anterior, pode parecer que o fato do computador despertar do estado de espera (*standby*) indicaria que essa incompatibilidade não existe. Entretanto, o retorno do estado de *standby* pode ser tratado inteiramente por *software* se a CPU não tiver sido desligada, de modo que ele não confirma o funcionamento correto do *hardware* [BEC2002].

Em suma, fica claro que a maior fonte de problemas é a incompatibilidade entre os diversos componentes envolvidos no processo, resultante de implementações divergentes das especificações pelos diversos fabricantes. Durante a execução dos testes, ficou evidente que, embora todos afirmem suportar um mesmo conjunto de padrões, na prática as implementações nem sempre conseguem interagir adequadamente.

Finalmente, cabe uma nota a respeito da documentação provida pelos fabricantes de *hardware*. Dificilmente encontram-se documentos específicos sobre os recursos de gerência de energia, e os que eventualmente se referem a estes recursos o fazem de maneira superficial. A maioria das informações úteis à solução de problemas é encontrada através de cansativas e repetidas pesquisas nas bases de dados de suporte dos fabricantes.

7.5.4 Outras incompatibilidades

Além das incompatibilidades mencionadas nas seções anteriores, foram observadas pelo menos duas outras que, por serem tão visíveis, merecem menção.

A primeira consiste na não detecção, pelo Windows 2000, da capacidade de *remote wake-up* da placa de rede 3Com quando instalada em um determinado computador (configuração 3 na Tabela 7.4), enquanto em outros dois computadores (configurações 1 e 2) ela havia sido detectada corretamente. Na Figura 7.2, pode-se observar a página *"Gerenciamento de energia"* do painel de configuração da placa de rede. O Windows só exibe esta página se detectar a compatibilidade da placa. No computador da configuração 3, a página não era exibida.

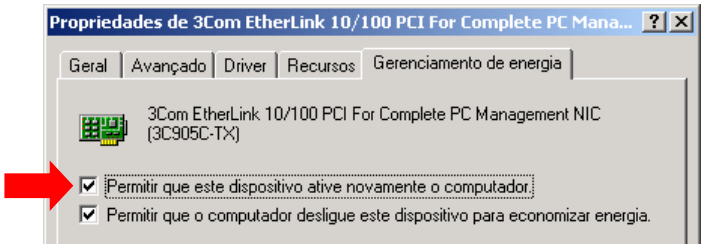


Figura 7.2 - Propriedades de gerência de energia da placa de rede no Windows

A segunda, observada no Linux, é a não detecção do suporte a APM do BIOS na configuração 2, embora o manual daquela placa-mãe afirme que o padrão é suportado pelo *firmware*. Neste caso, ainda que o computador disponha dos recursos necessários para a gerência de energia, ela não pode ser executada pelo sistema operacional em decorrência do que aparenta ser uma deficiência do *driver* APM do *kernel*.

As incompatibilidades entre os componentes envolvidos na gerência de energia nem sempre são tão visíveis. Determinar a causa do problema pode se tornar um trabalho tedioso e que muitas vezes não dá resultados. Já foi mencionado que a documentação dos fabricantes de *hardware* referente aos recursos de gerência de energia costuma ser bastante superficial, e as informações destinadas à solução de problemas geralmente estão espalhadas em repositórios *on-line* e são difíceis de localizar. Além disso, quando esse tipo de incompatibilidade é descoberto,

os fabricantes nem sempre oferecem uma solução para o problema (como atualizações de *software*, por exemplo).

7.6 Outros testes

A última etapa da fase de testes do trabalho teve o objetivo de obter, através de experimentação, medidas do ganho proporcionado pelo sistema numa rede abastecida por *no-break*, em situações de falha no fornecimento externo de energia. Estas medidas são apresentadas na conclusão do trabalho, no capítulo seguinte, junto de estimativas da economia que o sistema pode proporcionar durante os períodos de inatividade dos equipamentos da rede.

8 Conclusão

Este trabalho apresentou o projeto e a implementação de um sistema de gerência de energia para redes locais. Neste capítulo, é feita uma breve revisão das idéias que fundamentam a proposta do sistema, bem como de seus objetivos específicos e da arquitetura implementada. Também são apresentadas as contribuições do trabalho (incluindo algumas medidas e estimativas da economia proporcionada pelo sistema) e dificuldades encontradas durante sua execução. Por fim, são apresentadas uma avaliação crítica dos resultados e algumas sugestões para trabalhos futuros.

8.1 Contexto da proposta

Como foi demonstrado no início deste trabalho, os equipamentos de informática são responsáveis por uma fatia considerável do total de energia elétrica consumida atualmente. Mais importante, entretanto, é o fato de que este tipo de equipamento corresponde à fonte de consumo que tem a maior taxa de crescimento. Não é difícil perceber porque isto acontece. Virtualmente todos os setores da economia possuem, hoje, algum grau de automatização e, portanto, dependem de computadores e outros equipamentos eletrônicos. Destacam-se, em particular, os setores de serviços em diversas áreas de tecnologia (como telecomunicações, provedores de acesso à Internet, serviços de computação móvel, etc.), que não param de crescer, aumentando a demanda por novos equipamentos todos os anos.

Entretanto, como ficou evidente no capítulo 2, a indústria de informática vem investindo no desenvolvimento de padrões e mecanismos que permitem reduzir o consumo de energia por computadores e outros equipamentos nos períodos em que estão inativos. Neste sentido, os esforços mais recentes resultaram na especificação ACPI (*Advanced Configuration and Power Interface*), que serve como base para uma arquitetura de gerência de energia para computadores cujo elemento central é o sistema operacional, responsável pela aplicação de políticas que definem quando e como o equipamento deve ser colocado em estados de baixo consumo. Merece destaque, também, o programa *Energy Star,* da Agência de Proteção Ambiental norte-americana, que vem, desde o início da década de 90, incentivando a implementação de mecanismos de controle desse tipo em uma grande variedade de aparelhos elétricos, particularmente equipamentos de informática.

Obviamente, o fato desses mecanismos de controle de consumo estarem disponíveis não implica que eles sejam efetivamente utilizados. De fato, freqüentemente não são, o que constitui outra importante fonte de desperdício. Ainda na introdução do trabalho, foi apresentada uma estimativa da quantidade de energia que os EUA deixam de economizar anualmente em decorrência da má utilização da gerência de energia em equipamentos de informática.

Essa constatação de que os mecanismos de controle de consumo de energia são subaproveitados, associada à percepção de que é crescente a utilização de redes em ambientes computacionais, leva à idéia chave da proposta apresentada neste trabalho: a de que é possível otimizar o uso dos recursos de gerência de energia de equipamentos interligados em rede através de um elemento central, responsável por coordenar a estratégia de gerência a partir de uma visão panorâmica da rede, determinada pelo administrador do sistema. O objetivo, logicamente, é a redução do consumo de energia.

8.2 Revisão dos objetivos e da arquitetura do sistema

Duas situações claras em que essa centralização da estratégia de gerência de energia é útil foram apresentadas. Uma delas é na definição de políticas de consumo que podem ser aplicadas uniformemente a diversos equipamentos, em função de horários comuns de inatividade (como à noite e nos finais de semana, por exemplo), evitando desperdício de energia. A outra é na ocorrência de falha no fornecimento externo de energia, quando a rede é ligada a um *no-break* e passa a ser abastecida exclusivamente por um banco de baterias. Neste caso, reduzir temporariamente o consumo de alguns equipamentos significa aumentar o tempo de autonomia das baterias, o que pode evitar o desligamento de outros equipamentos mais importantes, caso a falha não se prolongue. O objetivo do sistema de gerência proposto é, exatamente, proporcionar economia de energia nestas duas situações, através de uma infra-estrutura baseada na associação entre a ocorrência de eventos e a execução de ações.

Conforme se descreveu em detalhe no capítulo 4, a arquitetura do sistema implementado consiste, basicamente, numa adaptação de um modelo cliente-servidor, utilizado em sistemas de monitoramento de *no-breaks*, ao modelo SNMP de gerência de redes. Nesse tipo de sistema de monitoramento, uma estação servidora é responsável por supervisionar um ou mais *no-breaks* e notificar estações clientes da iminência de condições críticas no abastecimento de energia, de modo que elas sejam adequadamente desligadas e se evite a perda de dados. De modo semelhante, na arquitetura proposta, uma *estação de gerência SNMP* (servidor) monitora as condições de fornecimento de energia e processa as políticas de consumo definidas pelo administrador, interagindo com *agentes SNMP* (clientes) instalados nos computadores da rede para alterar seu estado de consumo conforme necessário.

8.3 Resultados e contribuições

Dentre as contribuições do trabalho, destacam-se:

- uma implementação bastante satisfatória da aplicação de gerência, testada e depurada extensivamente. Acredita-se que a aplicação tenha atingido um nível de funcionalidade e de maturidade que permite, pelo menos, sua utilização em ambiente acadêmico. Exceto pelo uso de SNMPv3 para o envio de requisições para os agentes, todas as outras funções essenciais foram implementadas;
- uma documentação farta da implementação do gerente, composta da documentação `javadoc` e dos diagramas UML apresentados nos Apêndices, e de uma descrição textual que ocupou um capítulo inteiro da dissertação (cap. 6),
- uma especificação adequada do agente. Essa especificação, apresentada no capítulo 5, mostrou-se legível e razoavelmente simples de implementar ao servir de guia para a programação de um agente para o sistema Windows 98 por um aluno do curso de graduação em Ciência da Computação, em curto tempo de desenvolvimento [KER2002];
- a implementação de um protótipo de agente para o sistema Windows 2000, que permitiu estender os testes realizados com o sistema e pode servir de base para futuras implementações. Essa implementação foi descrita no capítulo 7;
- a documentação do "estado da arte" de dois dos principais sistemas operacionais para PCs, no que diz respeito à gerência de energia, também no capítulo 7.

Mais importantes do que as contribuições listadas acima, entretanto, são os benefícios práticos que a utilização do sistema implementado pode trazer. Para se ter uma idéia do ganho potencial que ele pode proporcionar, são apresentadas, a seguir, algumas medidas resultantes de experimentação e estimativa.

8.3.1 Medidas do ganho potencial em situações de falha no fornecimento de energia

Na fase final do trabalho, foram efetuados alguns testes com o objetivo de obter medidas empíricas do ganho proporcionado pelo sistema numa rede protegida por *no-break*, em situações de falha no fornecimento externo de energia. Estas medidas, apresentadas a seguir, correspondem ao tempo de autonomia das baterias do *no-break* em três diferentes cenários, decorrentes da configuração da aplicação de gerência para responder ao evento de início de operação em bateria de diferentes maneiras.

O ambiente de testes consistiu numa pequena rede Ethernet de dez PCs (um dos quais foi utilizado como servidor, para rodar o gerente), alimentados através de um *no-break* de 2kVA da CP Eletrônica (modelo Breakless 1220), operando com um conjunto de 12 baterias de 9Ah. O consumo médio de energia por PC é de 125 Watts em modo ativo, 30 Watts em modo de espera (*standby*) e 6,5 Watts em modo desligado (*soft-off*). Este ambiente de testes foi montado na sede da CP Eletrônica, que dispunha da infra-estrutura adequada à execução das medições.

No primeiro cenário, a aplicação não foi configurada para reagir ao evento de início de operação em bateria, o que equivale a uma medida da autonomia mínima do *no-break* para esta configuração de rede, já que todos os PCs permaneceram ativos até que as baterias estivessem esgotadas. No segundo cenário, a aplicação foi configurada para responder ao evento colocando 9 dos 10 computadores em modo de espera, mantendo ativo apenas o servidor. O terceiro e último cenário é semelhante ao anterior, com a diferença de que os computadores anteriormente colocados em modo de espera foram, desta vez, desligados.

Os valores de autonomia medidos nestes três cenários são ilustrados na Figura 8.1. Como se pode observar no gráfico, houve um aumento de 2,5 vezes na autonomia das baterias do cenário *a* para o cenário *b*. De *a* para *c*, observa-se um aumento de mais de 6 vezes. Em termos práticos, isso significa que haveria uma chance 2,5 e 6 vezes maior, respectivamente, de evitar o desligamento do servidor, que permaneceu ligado durante os testes. Se o fornecimento externo fosse restabelecido em qualquer instante da faixa entre 26min e 1h31min de operação em bateria, no primeiro caso, e entre 26min e 3h09min, no segundo, a rede poderia voltar a funcionar normalmente, sem a necessidade de reiniciar o servidor (o que costuma ser uma fonte constante de problemas para administradores de rede).

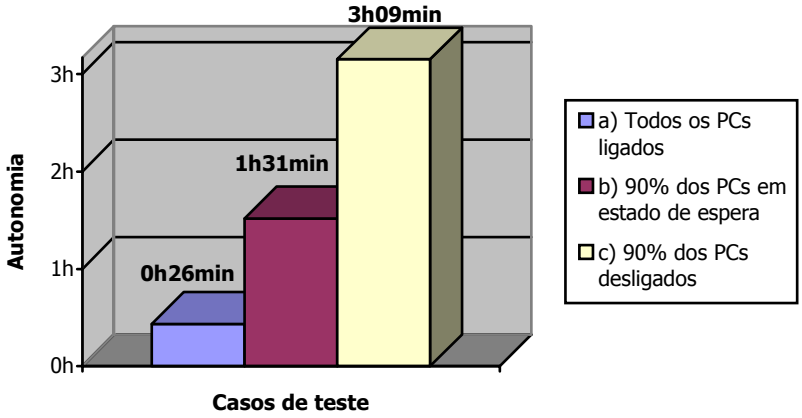


Figura 8.1 - Medidas de autonomia do *no-break* em diferentes cenários de consumo

Apesar destas medidas não serem inerentemente precisas, elas são bastante ilustrativas e ajudam a reforçar a idéia de que a centralização do processo de gerência de energia pode ser

lucrativa sob diversos aspectos, inclusive o de evitar grandes transtornos para recolocar uma rede em operação em caso de problemas no fornecimento externo de energia. É claro que, nesta configuração hipotética usada para os testes, foi omitida a possível intervenção dos usuários no processo de mudança de estado de consumo. Mas o simples fato de que os usuários poderiam ser automaticamente notificados da iminência do corte de energia, numa situação real, prevenindo perdas de dados e danos aos equipamentos, é outro indicativo da utilidade do sistema.

Infelizmente, não foi possível realizar testes semelhantes em outros laboratórios do Instituto de Informática, onde o ambiente de funcionamento do sistema não seria controlado. Como já foi dito anteriormente, os *no-breaks* do Instituto que abastecem estes laboratórios não estão sendo supervisionados por *software* adequado à comunicação via SNMP, o que impede seu monitoramento pela aplicação de gerência implementada.

8.3.2 Estimativa da economia potencial em horários de inatividade

Outra maneira de avaliar o ganho proporcionado pelo sistema proposto é estimar a quantidade de energia economizada em horários de inatividade dos equipamentos.

Considerando um horário de trabalho das 8 às 18 horas, restam, em cada dia, 14 horas durante as quais a maioria dos equipamentos permanece inativa. Considere-se ainda: o consumo médio de um computador e de um monitor em diferentes modos de consumo, conforme a Tabela 8.1; a tarifa média praticada no Brasil no primeiro semestre de 2002, nos setores comercial e residencial, que foi de R$0,15/MWh (vide Tabela 8.2); e que cada ano tem cerca de 250 dias úteis. A partir desses dados, pode-se estimar que:

- cada PC (computador + monitor) consome cerca de 1,25KWh de energia por dia, durante o horário típico de trabalho, se for mantido ativo todo o tempo. Isso equivale a um custo de R$0,73 por ano;
- se cada PC for mantido completamente ativo durante o período de ociosidade, há um consumo adicional de 1,75KWh por dia. Isso equivale a um gasto desnecessário de R$0,63 por ano, por computador;
- se o monitor for colocado em estado de baixo-consumo durante o período de ociosidade, serão economizados cerca de R$0,83 por ano, por PC;
- se, além do monitor, o computador também for colocado num estado de baixo-consumo durante o período de inatividade, o total economizado chega a R$0,56.

É interessante mencionar que, das 10 horas úteis de cada dia, estima-se que os computadores sejam efetivamente utilizados durante menos da metade do tempo [WEB2001], o que abre espaço para economia ainda maior.

Tabela 8.1 - Consumo médio de computadores e monitores (em Watts)

Modo de consumo / Equipamento	**Ativo**	**Baixo-consumo**	**Desligado**
Computador *desktop*	50	25	1,5
Monitor de vídeo 15″	75	5	0,5
Total	**125**	**30**	**2**

Fonte: LBNL, Universidade da Califórnia [KAW2001]

Tabela 8.2 - Tarifas médias por classe de consumo e região entre janeiro e maio de 2002 (R$/MWh)

Classe de Consumo	Norte	Nordeste	Sudeste	Sul	Centro-Oeste	Brasil
Residencial	182,43	181,18	213,34	203,15	193,60	203,75
Comercial	162,01	159,88	187,03	170,97	173,40	178,55
Tarifa Média	172,22	170,53	200,19	187,06	183,5	191,15

Fonte: ANEEL [ANE2002]

8.4 Dificuldades encontradas

As principais dificuldades encontradas durante a execução do trabalho estão relacionadas com a imaturidade das especificações e implementações de padrões de gerência de energia e com a escassez de documentação dos dispositivos de *hardware* e das APIs de sistema. Como foi visto no capítulo 7, um dos maiores empecilhos para a execução eficiente da gerência de energia em PCs é a incompatibilidade entre as implementações de diferentes fabricantes, que muito provavelmente decorre deste tipo de problema.

Outro problema menos evidente, mas igualmente frustrante, é a documentação incompleta de MIBs. A maioria das MIBs analisadas não era acompanhada de uma descrição substancial do comportamento esperado de um agente compatível. Assim, ainda que a definição de uma MIB caracterizasse de maneira inequívoca o significado dos objetos constituintes da árvore, muitas vezes não revelava o comportamento esperado do agente, explicando como e quando os valores dos objetos seriam modificados, em que situações específicas notificações seriam enviadas, etc.

Quanto à execução de testes, houve dificuldade em monitorar *no-breaks* fora do Laboratório de Convênio CP Eletrônica - UFRGS. Seria interessante ter monitorado aparelhos microprocessados (modelos da linha Top da CP Eletrônica), que permitiriam a coleta de uma série de dados que não estão disponíveis nos modelos de contato seco, que foram utilizados nos testes. Alguns dados importantes que poderiam ser obtidos são o consumo médio em diferentes horários e a duração média das falhas no fornecimento externo.

Quanto às ferramentas utilizadas para a implementação, não foi possível encontrar uma implementação gratuita e adequada da versão 3 do protocolo SNMP em Java, que seria necessária para garantir a segurança das operações de controle dos equipamentos da rede.

Finalmente, é preciso mencionar que muito do esforço empreendido na execução do trabalho, especialmente na fase inicial, foi destinado a revisar o projeto anterior, apresentado por Krolow [KRO2001]. É notável a quebra de continuidade criada pela saída de membros dos grupos de pesquisa. Muito tempo e esforço acabam sendo gastos para readquirir um conhecimento que já havia sido sedimentado em gerações anteriores do grupo, e que acaba sendo perdido pela falta de documentação e pela dificuldade de manter contato com os ex-alunos.

8.5 Avaliação crítica

Conforme foi enfatizado claramente no Plano de Estudos e Pesquisa onde se apresentou a proposta deste trabalho, o objetivo pretendido era não apenas reimplementar o protótipo apresentado no trabalho precedente, mas "obter uma implementação nova e mais completa que a anterior" [POL2001]. Para que o trabalho apresentado nesta dissertação se mantivesse fiel a esse objetivo, um período de mais de um ano foi dedicado à implementação do novo sistema. Muitas

das idéias introduzidas no trabalho anterior foram aproveitadas, mas o sistema foi, de fato, replanejado e implementado a partir do zero.

Possivelmente, o maior mérito do trabalho esteja na realização desta implementação. São mais de 8 mil linhas de código útil (mais de 2 vezes a quantidade de linhas desta dissertação, para se ter uma idéia), só na aplicação de gerência. O resultado, como já foi mencionado, é um sistema que tem condições de escapar da sina que, aparentemente, tem a maioria dos protótipos desenvolvidos em trabalhos acadêmicos: acabar engavetados e esquecidos.

Contudo, nem tudo são méritos. Como também é comum em trabalhos acadêmicos, este foi marcado por uma lenta curva de aprendizado, envolvendo muita pesquisa, antes que os resultados práticos finalmente aparecessem. Ao longo desse período, a quantidade de informações que precisam ser digeridas é enorme, o que pode levar a decisões equivocadas e à possibilidade de negligenciar as melhores (ou mais simples) soluções. Além disso, o segundo ano de um curso de mestrado costuma ser, sob muitos aspectos, uma empreitada solitária. Isso favorece o aparecimento de erros, especialmente na parte de programação, que possivelmente seriam detectados num trabalho em grupo.

Alguns desses erros e omissões acabam se tornando visíveis com a depuração das aplicações e a execução de testes. Do mesmo modo, decisões acertadas acabam sendo reveladas ao longo desse processo. As seções seguintes listam as principais descobertas neste sentido.

8.5.1 Modelo de funcionamento dos agentes

O modelo especificado para os agentes, que propõe um mecanismo de duas fases para a execução do ciclo completo de uma ação, obtidas através da separação entre a execução propriamente dita da ação e o posterior envio da notificação que indica sucesso ou falha da operação, demonstrou ser adequado e flexível. A implementação do agente-protótipo (ver seção 7.3) provou que essa separação simples permite o convívio do agente de gerência de energia com o controle de consumo nativo do sistema operacional, sem que um interfira na operação do outro.

Associada a este processo de duas fases, a decisão de desvincular, no gerente, o envio de uma requisição e a recepção de uma determinada notificação do agente (como confirmação da ação disparada) também provou ser adequada. Essa decisão simplificou bastante a implementação da aplicação, evitando a detecção de *timeouts* de confirmações. Na verdade, conforme foi explicado na seção 4.3.4, não há como prever o tempo que um agente levaria entre a recepção de uma requisição e o envio da notificação, de modo que não haveria, também, como escolher um limite de tempo adequado para as confirmações. Mas, da maneira como foi implementado, o monitor da rede consegue identificar os estados dos equipamentos perfeitamente bem.

8.5.2 Modelo de segurança

Quanto ao modelo de segurança implementado na aplicação de gerência, no que diz respeito à comunicação com interfaces gráficas ou outros componentes via RMI, é interessante notar que, apesar de não se tratar de um modelo à prova de falhas, ele previne a descoberta de senhas não-codificadas no ambiente da rede local. Parte-se do pressuposto de que a mesma senha original possa ser utilizada em outros serviços da rede. O método não tem qualquer vantagem sobre a transmissão sem codificação se as credenciais forem usadas exclusivamente na aplicação de gerência de energia, já que o roubo da senha codificada permitira a execução das operações por usuários não autorizados (a senha que trafega na rede é a mesma armazenada

no gerente). Ataques do tipo *"man in the middle"*, que esse modelo não é capaz de evitar, costumam ser uma preocupação menos prioritária em redes locais (basta observar a quantidade de serviços de rede que operam sobre protocolos não-seguros). Ainda assim, procurando alternativas para aumentar o nível de segurança nessa parte do sistema, foram consideradas algumas soluções, dentre as quais destacam-se:

- RMI sobre SSL [SUN2000];
- *Simple Authentication and Security Layer* (SASL) para Java [CRY2002];
- *Java Authentication and Authorization Service* (JAAS) [SUN2002];
- modelos de autenticação do tipo "desafio-resposta".

Estas soluções não foram implementadas porque aumentariam consideravelmente o tempo de desenvolvimento do sistema numa área que não era essencial para a prova de conceito.

No que se refere à segurança da comunicação entre o gerente e os agentes, a impossibilidade de usar SNMPv3 gera um problema fundamental: qualquer estação da rede poderia acessar a MIB de qualquer agente e alterar seu estado, sem que qualquer processo de validação de credenciais ou autorização fosse efetuado. Vale ressaltar, no entanto, que este é um pequeno detalhe de implementação, que pode ser solucionado facilmente com a utilização de uma biblioteca SNMP adequada (há produtos comerciais que possuem essa funcionalidade, por exemplo). Um método simples que pode ser utilizado para restringir a possibilidade de ataques desse tipo é descartar, nos agentes, qualquer mensagem SNMP não proveniente do endereço IP do gerente.

Uma vez que o suporte a SNMPv3 tenha sido incorporado, a habilitação do agente de gerência de energia num equipamento não tem impacto maior do que a habilitação de um servidor SSH, por exemplo. Alguém que disponha das credenciais e dos devidos privilégios de acesso poderia desativar o PC nos dois casos.

8.5.3 Mecanismo de cancelamento de ações

Conforme apresentado na seção 6.5.2, o mecanismo de cancelamento de ações do sistema é baseado num algoritmo estático de eliminação de ações pendentes em função dos tipos e horários de execução das ações em colisão, e do estado corrente do equipamento-alvo. Isso significa que, para uma ação ser cancelada, é necessário que outra ação tenha sido associada a um segundo evento, cujo significado é "cancele o evento anterior". Considere-se o seguinte cenário:

1. ocorre uma falha no fornecimento de energia da concessionária, e a rede passa a operar em bateria: o sistema programa uma ação de desligamento para execução em *t* minutos;
2. enquanto o tempo *t* está transcorrendo, o fornecimento de energia é restabelecido;
3. se uma ação estiver associada ao retorno do fornecimento, para execução imediata (*t = 0*), a ação de desligamento programada anteriormente seria cancelada; caso contrário, ela seria executada, desnecessariamente.

Neste tipo de situação, seria mais adequado poder configurar o sistema para cancelar eventos inteiros (todas as ações associadas a um determinado evento). Isto exigiria uma modificação no despachante de ações (`ActionDispatcher`) para incorporar essa noção de cancelamento entre eventos (e não apenas entre ações), que não existe no momento.

8.5.4 Configuração local de gerência de energia dos PCs

Um problema que o sistema não soluciona é o da má configuração local dos computadores para gerência de energia que, se corrigida, pode otimizar a economia além do que é possível fazer com a aplicação de gerência centralizada, pois o sistema operacional de cada máquina pode determinar, sem nenhum "atraso", quando ela está ociosa, e colocá-la num modo de consumo reduzido. Outro tópico interessante a abordar seria a configuração remota da gerência local de energia dos PCs, com a instalação de políticas de consumo baseadas em limites de ociosidade, e não só em eventos programados.

O ideal seria adaptar os agentes para que eles permitissem observar a configuração local de políticas de energia e definir operações para alteração remota dessas políticas. Deste modo, a aplicação de gerência poderia obter uma visão dinâmica da configuração de cada equipamento, permitindo ao administrador efetuar mudanças em tempo real, caso necessário.

8.5.5 Outras funções omitidas ou pendentes

Outras funções que não foram implementadas por não serem fundamentais, ou que foram omitidas inadvertidamente, ou ainda que foram concebidas durante a etapa de testes, são as seguintes:

- envio de requisições do gerente para os agentes via SNMPv3;
- desligamento automático também da estação de gerência, e não só dos clientes. No estágio atual, a aplicação não desliga o computador na iminência de um corte de energia, o que é desejável. Também não é feita nenhuma verificação da configuração para determinar se foram definidas ações cujo alvo é a própria estação de gerência. Obviamente, as ações destinadas à estação de gerência só terão algum efeito se um agente também estiver sendo executado na mesma máquina, o que não deixa de ser uma possibilidade;
- edição da configuração do sistema através de interface gráfica. A interface gráfica implementada permite editar os arquivos XML manualmente (um método não muito amigável);
- definição de "variáveis de ambiente" na configuração, de modo que se pudessem definir mensagens personalizadas. Por exemplo:

```
<ups-event src="nobreak1" type="UPS_ON_BATTERY">
   <action type="SHOW_MESSAGE" target="PCs">
      <param name="msg"
             value="A rede está operando em bateria. O tempo de
                    autonomia estimado é de $ESTIMATED_MINUTES_REMAINING
                    minutos. />
   </action>
</ups-event>
```

  Neste exemplo, a mensagem a ser mostrada pelos agentes conteria o tempo estimado de autonomia, que o gerente obteria dinamicamente do monitor do *no-break* associado ao evento;

- remoção do vínculo do monitor da rede (`NetworkMonitor`) com a biblioteca nativa usada para obtenção de endereços MAC. A atualização da *cache* depende de que os computadores estejam ligados, pois do contrário os adaptadores de rede não podem responder às requisições ARP. E, toda vez que um computador está ligado, assume-se que o seu agente de gerência de energia (que é capaz de prover o endereço MAC) está rodando, de modo que o gerente pode atualizar a *cache* a partir da consulta ao agente, removendo a única parte da aplicação que não é implementada em Java;

- substituição das notificações `trapPowerOn` e `trapWakeUp`, da MIB de gerência de energia, pelas notificações `coldStart` e `warmStart`, respectivamente, da MIB-II padrão.

8.6 Perspectivas futuras

No sentido de aperfeiçoar o sistema proposto neste trabalho e estender as contribuições do Grupo de Tolerância a Falhas na área de gerência de energia, espera-se que novas pesquisas venham a seguir. Especificamente, podem-se sugerir os seguintes tópicos para novos trabalhos:

- implementação das funções pendentes do gerente, especialmente a comunicação via SNMPv3;
- complementação do agente-protótipo para Windows 2000;
- implementação de um agente para Linux;
- integração do gerente com uma plataforma de gerência (p. ex. HP OpenView);
- pesquisa de campo para determinar o padrão de utilização de equipamentos do Instituto de Informática[1];
- monitoramento dos *no-breaks* micro-processados do Instituto para coleta de dados.

Além destes tópicos, há outras áreas indiretamente relacionadas com o trabalho que poderiam ser abordadas, entre as quais pode-se destacar:

- a implementação de partes do código ACPI/OSPM pendentes no sistema operacional Linux; e
- a implementação de suporte à versão 3 de SNMP à biblioteca JoeSNMP.

8.7 Considerações finais

As pesquisas na área de gerência de energia do Grupo de Tolerância a Falhas do PPGC da UFRGS vêm evoluindo continuamente. Conforme mencionado na introdução do trabalho, este é o mais recente de uma linha de trabalhos que investigaram possibilidades para gerência de energia em redes de computadores, procurando aproveitar, em particular, a infra-estrutura para monitoramento das condições do fornecimento de energia criada pela presença de *no-breaks*.

Como foi demonstrado anteriormente, há uma tendência de crescimento do consumo de eletricidade devido ao incremento da quantidade de equipamentos de informática em uso, o que acentua a relevância desta e de outras pesquisas do Grupo. Os resultados apresentados neste trabalho, em particular, são mais um indicativo de que este é um caminho importante a seguir.

[1] Um dos estudos do LBNL referenciados no texto apresenta uma metodologia para a realização de uma pesquisa de campo deste tipo [WEB2001].

Apêndice A - Requisitos do sistema

A.1 Visão geral

O propósito deste projeto é criar um sistema que permita centralizar a gerência de energia de computadores ligados em rede e, possivelmente, abastecidos através de *no-breaks* (UPSs).

A maioria dos computadores utilizados atualmente possui algum tipo de recurso para gerência de energia. Em ambientes de rede, no entanto, não há um mecanismo central para configuração remota desses recursos nos equipamentos ou para o monitoramento de seus estados de consumo de energia.

O sistema será composto basicamente de uma entidade controladora, o *gerente do sistema*, e de *agentes* instalados em cada equipamento que se deseja gerenciar. O gerente é, claramente, a parte mais complexa do sistema, e receberá maior ênfase neste projeto. Agentes completos, capazes de executar todas as funções especificadas em um certo padrão de gerência de energia, poderão ser objeto de pesquisa em trabalhos futuros.

A.2 Objetivos

Em síntese, os objetivos principais são prolongar o tempo de fornecimento de energia pelas baterias da(s) UPS(s) em situações de falha ou corte no abastecimento externo e economizar energia quando os equipamentos da rede estiverem ociosos durante o abastecimento regular. Mais especificamente, estão incluídos:

- monitoramento local (através de conexão serial) ou remoto (através de conexão TCP/IP) das UPSs que abastecem a rede;
- monitoramento remoto dos estados de consumo de energia de cada equipamento gerenciado e, possivelmente, de seus dispositivos internos;
- controle dos equipamentos gerenciados, com base em políticas de gerência de energia definidas em função de eventos relativos ao fornecimento de eletricidade ou de ações pré-programadas (p. ex., "desligar os monitores de todos os equipamentos, diariamente, às 22:00");
- configuração flexível do gerente, de modo a suportar diversas combinações de *hardware*, sistemas operacionais e padrões de gerência de energia nos agentes.

A.3 Funções do gerente

A.3.1 Funções de configuração e administração

Referência	Função	Categoria
F-1.1	Exibir informações de configuração do sistema em interface gráfica, permitindo sua alteração.	
F-1.2	Converter informações de configuração para formato XML e salvá-las em disco.	
F-1.3	Carregar arquivos de configuração durante a inicialização, convertendo as informações armazenadas em formato XML para as estruturas internas apropriadas.	
F-1.4	Exibir informações sobre o sistema (como estado das UPSs, estado dos	

Referência	Função	Categoria
	equipamentos da rede, etc.) na interface gráfica.	
F-1.5	Permitir controle remoto, através da interface gráfica, dos equipamentos.	🧍

Legenda: 💻 *- funções invisíveis ao usuário*
🧍 *- funções interativas*

A.3.2 Funções de segurança e *logging*

Referência	Função	Categoria
F-2.1	Cadastrar usuários e respectivas senhas para autenticação e criptografia das mensagens, através da interface gráfica.	🧍
F-2.2	Salvar cadastro de usuários em disco. As senhas devem ser criptografadas.	💻
F-2.3	Validar mensagens digitalmente assinadas e criptografadas, com base no cadastro de usuários.	💻
F-2.4	Salvar informações sobre os resultados das operações em arquivos de *log* (histórico das operações).	💻

A.3.3 Funções de monitoramento de UPSs

Referência	Função	Categoria
F-3.1	Monitorar eventos de energia (como interrupção do fornecimento externo, nível de bateria baixo, etc.) reportados pelas UPSs.	💻

A.3.4 Funções de monitoramento e controle dos agentes

Referência	Função	Categoria
F-4.1	Monitorar notificações dos agentes sobre alterações nos estados de consumo de energia dos equipamentos.	💻
F-4.2	Enviar pacote especial pela rede para despertar equipamentos em estado reduzido de consumo de energia.	💻
F-4.3	Comandar alteração do estado de consumo de energia em um determinado equipamento ou em seus dispositivos internos/periféricos.	💻
F-4.4	Comandar execução de comando ou *script* em determinado equipamento.	💻
F-4.5	Comandar desligamento do sistema operacional em um determinado equipamento, com possível desligamento da fonte de energia.	💻
F-4.6	Comandar a exibição de mensagem informativa para o usuário de determinado equipamento.	💻
F-4.7	Monitorar a lista de ações pré-configuradas para execução no tempo apropriado.	💻

A.4 Funções dos agentes

Os agentes são parcialmente dependentes da plataforma em que são executados. A parte encarregada da troca de mensagens com o gerente pode ser idêntica em todas as plataformas, mas aquela encarregada da comunicação com o sistema operacional subjacente precisa ser

implementada em uma linguagem de programação nativa, que disponha de acesso às funções da API do SO.

A.4.1 Funções de gerência de energia

Referência	Função	Categoria
F-5.1	Monitorar eventos relativos à gerência de energia reportados pelo sistema operacional do equipamento.	
F-5.2	Notificar o gerente sobre mudanças no estado de consumo do equipamento e de seus dispositivos internos/periféricos iniciadas pelo sistema operacional.	
F-5.3	Efetuar alteração do estado de consumo de energia do equipamento ou de seus dispositivos internos/periféricos.	
F-5.4	Executar comando ou *script* enviado pelo gerente.	
F-5.5	Efetuar desligamento do sistema operacional, com possível desligamento da fonte de energia.	
F-5.6	Mostrar mensagem informativa para o usuário (possivelmente requisitando-lhe algum tipo de confirmação/resposta).	

A.4.2 Funções de segurança e *logging*

Referência	Função	Categoria
F-6.1	Validar mensagens digitalmente assinadas e criptografadas, com base no cadastro de usuários.	
F-6.2	Salvar informações sobre os resultados das operações em arquivos de *log* (histórico das operações).	

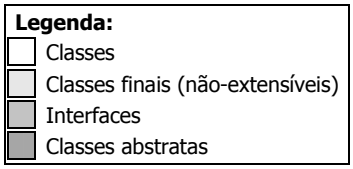
A.5 Atributos do sistema

Atributo	Detalhes e restrições
Tempo de resposta	O sistema assumirá que o estado de um agente é desconhecido se não receber notificação dentro do limite de tempo (*timeout*) especificado no arquivo de configuração.
Metáfora de interface	Interface baseada em elementos gráficos, como ícones e formulários, possivelmente disponibilizada através da *Web*.
Plataformas	Sempre que possível, a linguagem Java será utilizada para permitir a execução do software em diferentes plataformas.
Compatibilidade e extensibilidade	O sistema deve ser construído de maneira a suportar interação com os padrões de gerência de energia atuais e, possivelmente, com futuros padrões. Uma característica comum a todos os padrões de gerência de energia existentes é que eles são organizados de maneira a possibilitar o controle dos vários dispositivos internos dos equipamentos, bem como de dispositivos periféricos. O método é basicamente o mesmo nas diferentes especificações: para cada dispositivo, uma lista de possíveis estados de consumo é definida; cada estado equivale a um nível de energização diferente.
Segurança	As mensagens enviadas do gerente para os agentes devem ser digitalmente assinadas, de maneira a serem autenticadas ao atingirem o destino. Funções administrativas só podem ser executadas pelo administrador do sistema a partir de autenticação de credenciais.

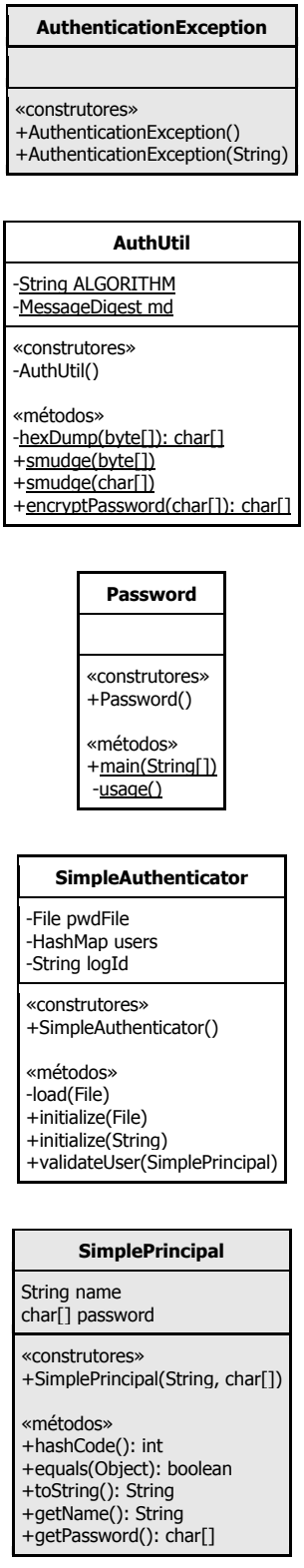
Apêndice B - Diagramas UML

Neste apêndice são apresentados diagramas UML para as classes mais importantes da aplicação de gerência. Inicialmente, é apresentado um diagrama de alto nível, que dá uma visão panorâmica dos relacionamentos entre as classes. Depois, são apresentados diagramas detalhados para cada uma das classes que aparecem no primeiro diagrama. Dentro do possível, as classes foram organizadas em ordem alfabética (a partir da seção B.2), embora, em algumas páginas, a ordem tenha sido quebrada para utilizar melhor o espaço disponível.

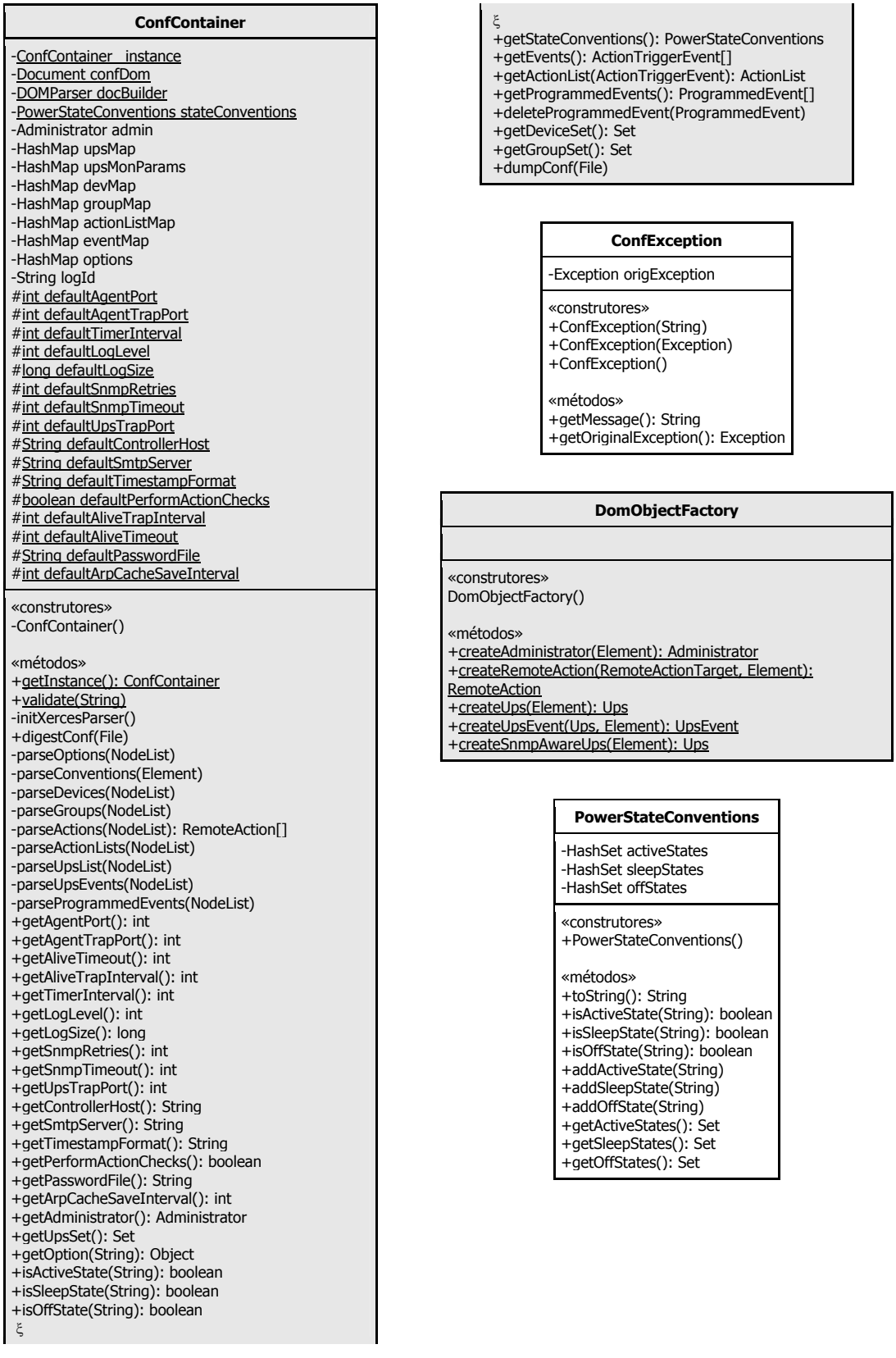
B.1 Visão geral



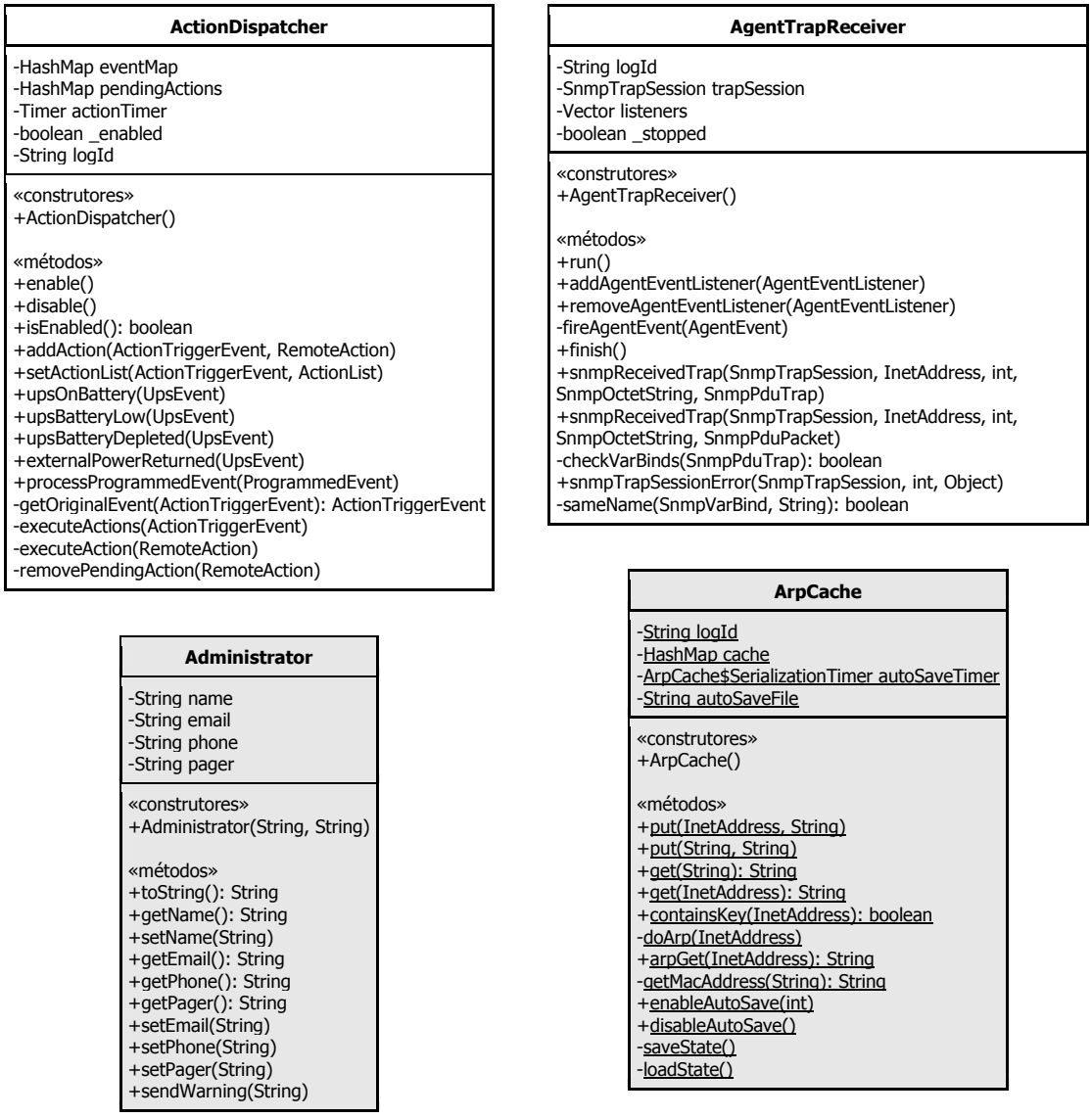
B.2 Pacote "br.ufrgs.inf.netpower.auth"

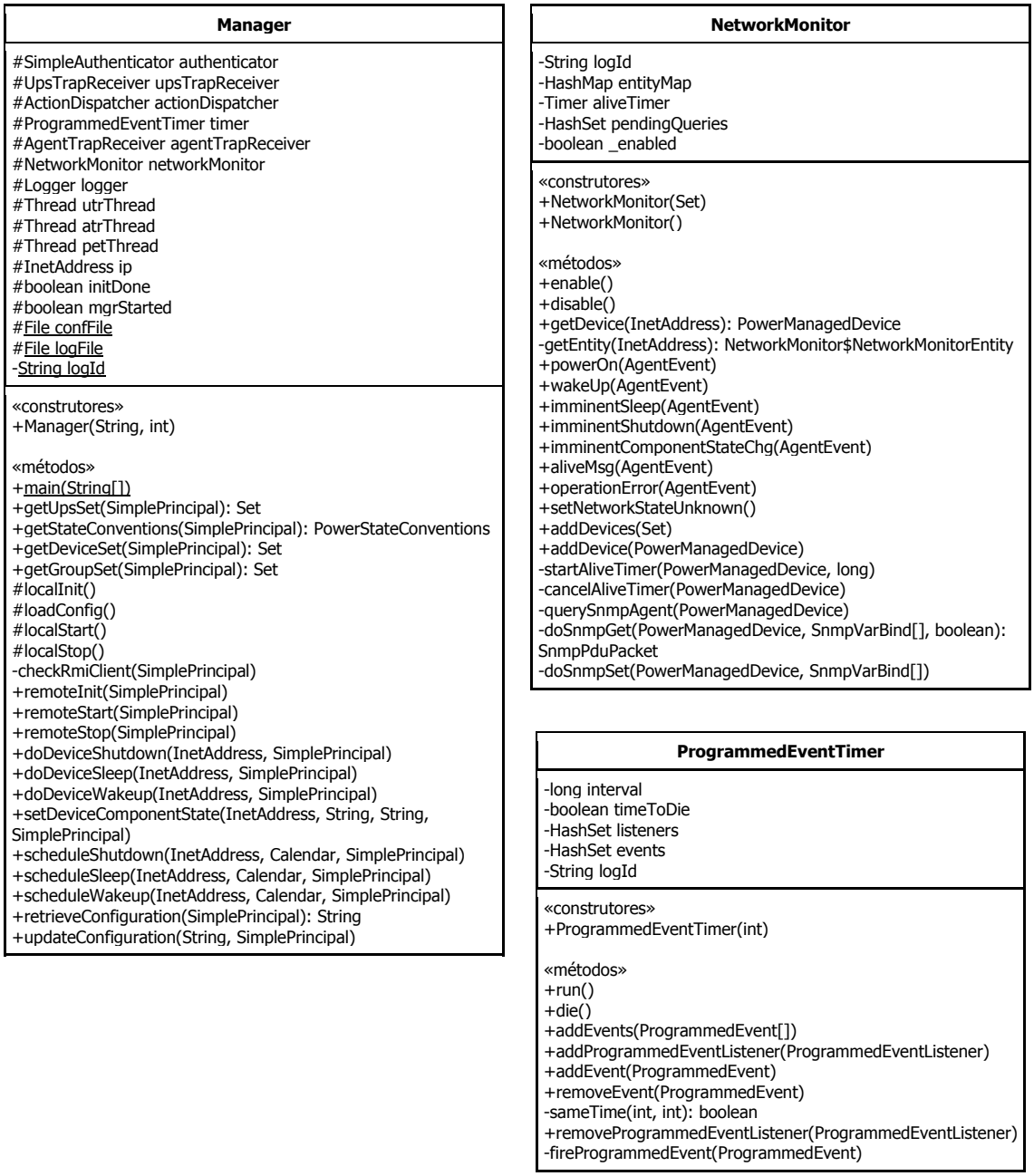


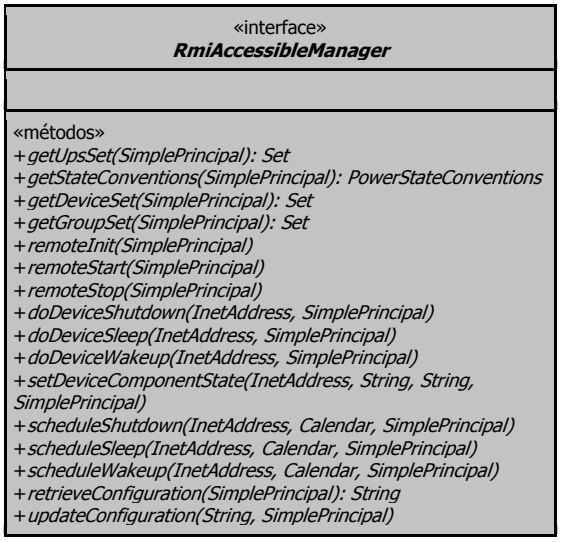
B.3 Pacote "br.ufrgs.inf.netpower.conf"

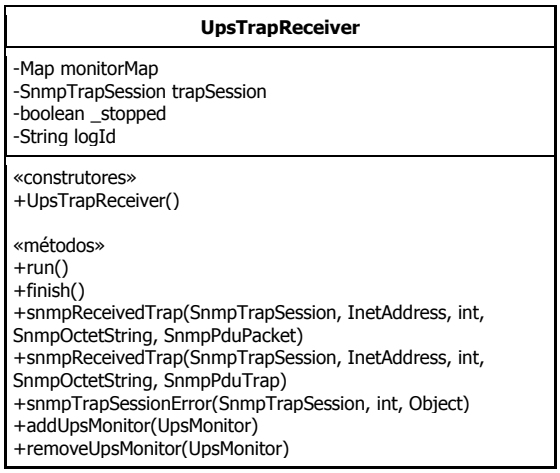


B.4 Pacote "br.ufrgs.inf.netpower.manager"

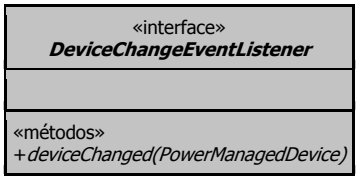


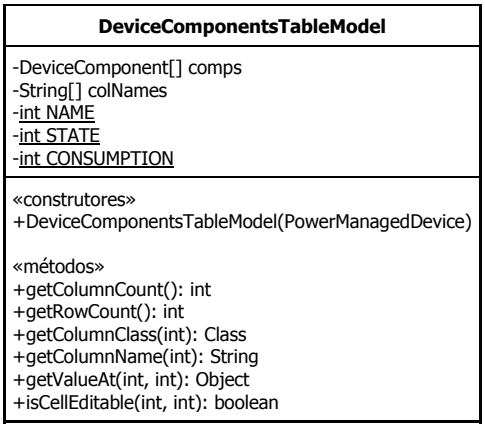

Manager
#SimpleAuthenticator authenticator
#UpsTrapReceiver upsTrapReceiver
#ActionDispatcher actionDispatcher
#ProgrammedEventTimer timer
#AgentTrapReceiver agentTrapReceiver
#NetworkMonitor networkMonitor
#Logger logger
#Thread utrThread
#Thread atrThread
#Thread petThread
#InetAddress ip
#boolean initDone
#boolean mgrStarted
#File confFile
#File logFile
-String logId
«construtores»
+Manager(String, int)
«métodos»
+main(String[])
+getUpsSet(SimplePrincipal): Set
+getStateConventions(SimplePrincipal): PowerStateConventions
+getDeviceSet(SimplePrincipal): Set
+getGroupSet(SimplePrincipal): Set
#localInit()
#loadConfig()
#localStart()
#localStop()
-checkRmiClient(SimplePrincipal)
+remoteInit(SimplePrincipal)
+remoteStart(SimplePrincipal)
+remoteStop(SimplePrincipal)
+doDeviceShutdown(InetAddress, SimplePrincipal)
+doDeviceSleep(InetAddress, SimplePrincipal)
+doDeviceWakeup(InetAddress, SimplePrincipal)
+setDeviceComponentState(InetAddress, String, String, SimplePrincipal)
+scheduleShutdown(InetAddress, Calendar, SimplePrincipal)
+scheduleSleep(InetAddress, Calendar, SimplePrincipal)
+scheduleWakeup(InetAddress, Calendar, SimplePrincipal)
+retrieveConfiguration(SimplePrincipal): String
+updateConfiguration(String, SimplePrincipal)
NetworkMonitor
-String logId
-HashMap entityMap
-Timer aliveTimer
-HashSet pendingQueries
-boolean _enabled
«construtores»
+NetworkMonitor(Set)
+NetworkMonitor()
«métodos»
+enable()
+disable()
+getDevice(InetAddress): PowerManagedDevice
-getEntity(InetAddress): NetworkMonitor$NetworkMonitorEntity
+powerOn(AgentEvent)
+wakeUp(AgentEvent)
+imminentSleep(AgentEvent)
+imminentShutdown(AgentEvent)
+imminentComponentStateChg(AgentEvent)
+aliveMsg(AgentEvent)
+operationError(AgentEvent)
+setNetworkStateUnknown()
+addDevices(Set)
+addDevice(PowerManagedDevice)
-startAliveTimer(PowerManagedDevice, long)
-cancelAliveTimer(PowerManagedDevice)
-querySnmpAgent(PowerManagedDevice)
-doSnmpGet(PowerManagedDevice, SnmpVarBind[], boolean): SnmpPduPacket
-doSnmpSet(PowerManagedDevice, SnmpVarBind[])
ProgrammedEventTimer
-long interval
-boolean timeToDie
-HashSet listeners
-HashSet events
-String logId
«construtores»
+ProgrammedEventTimer(int)
«métodos»
+run()
+die()
+addEvents(ProgrammedEvent[])
+addProgrammedEventListener(ProgrammedEventListener)
+addEvent(ProgrammedEvent)
+removeEvent(ProgrammedEvent)
-sameTime(int, int): boolean
+removeProgrammedEventListener(ProgrammedEventListener)
-fireProgrammedEvent(ProgrammedEvent)

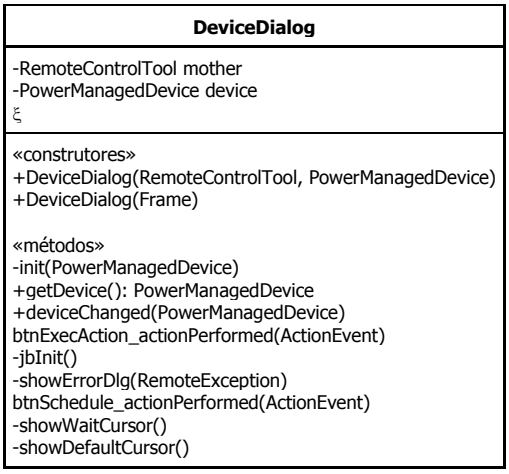

«interface»
RmiAccessibleManager
«métodos»
+getUpsSet(SimplePrincipal): Set
+getStateConventions(SimplePrincipal): PowerStateConventions
+getDeviceSet(SimplePrincipal): Set
+getGroupSet(SimplePrincipal): Set
+remoteInit(SimplePrincipal)
+remoteStart(SimplePrincipal)
+remoteStop(SimplePrincipal)
+doDeviceShutdown(InetAddress, SimplePrincipal)
+doDeviceSleep(InetAddress, SimplePrincipal)
+doDeviceWakeup(InetAddress, SimplePrincipal)
+setDeviceComponentState(InetAddress, String, String, SimplePrincipal)
+scheduleShutdown(InetAddress, Calendar, SimplePrincipal)
+scheduleSleep(InetAddress, Calendar, SimplePrincipal)
+scheduleWakeup(InetAddress, Calendar, SimplePrincipal)
+retrieveConfiguration(SimplePrincipal): String
+updateConfiguration(String, SimplePrincipal)

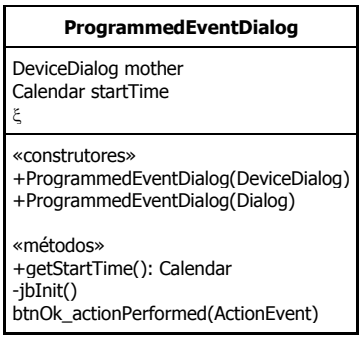

UpsTrapReceiver
-Map monitorMap
-SnmpTrapSession trapSession
-boolean _stopped
-String logId
«construtores»
+UpsTrapReceiver()
«métodos»
+run()
+finish()
+snmpReceivedTrap(SnmpTrapSession, InetAddress, int, SnmpOctetString, SnmpPduPacket)
+snmpReceivedTrap(SnmpTrapSession, InetAddress, int, SnmpOctetString, SnmpPduTrap)
+snmpTrapSessionError(SnmpTrapSession, int, Object)
+addUpsMonitor(UpsMonitor)
+removeUpsMonitor(UpsMonitor)

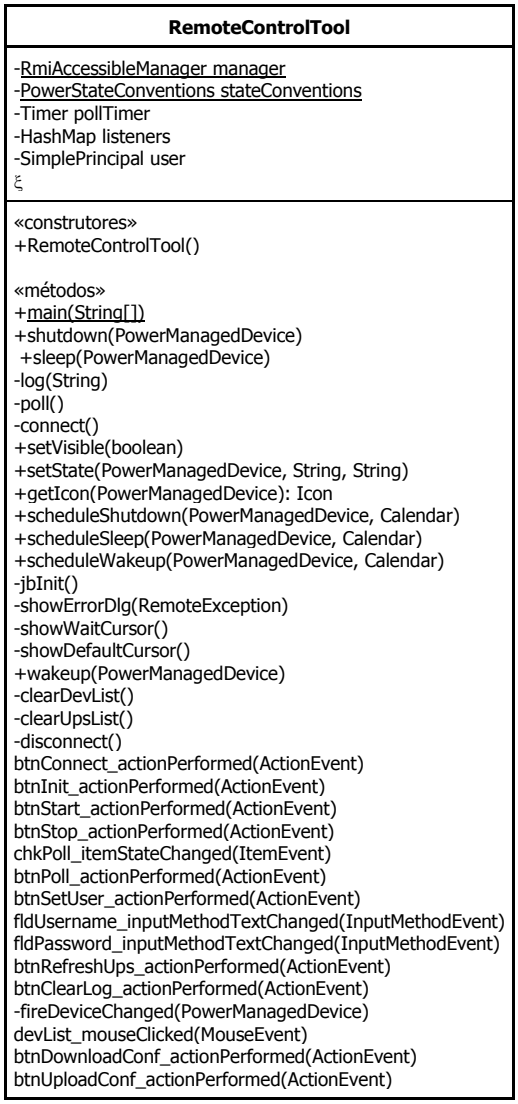
B.5 Pacote "br.ufrgs.inf.netpower.manager.gui"



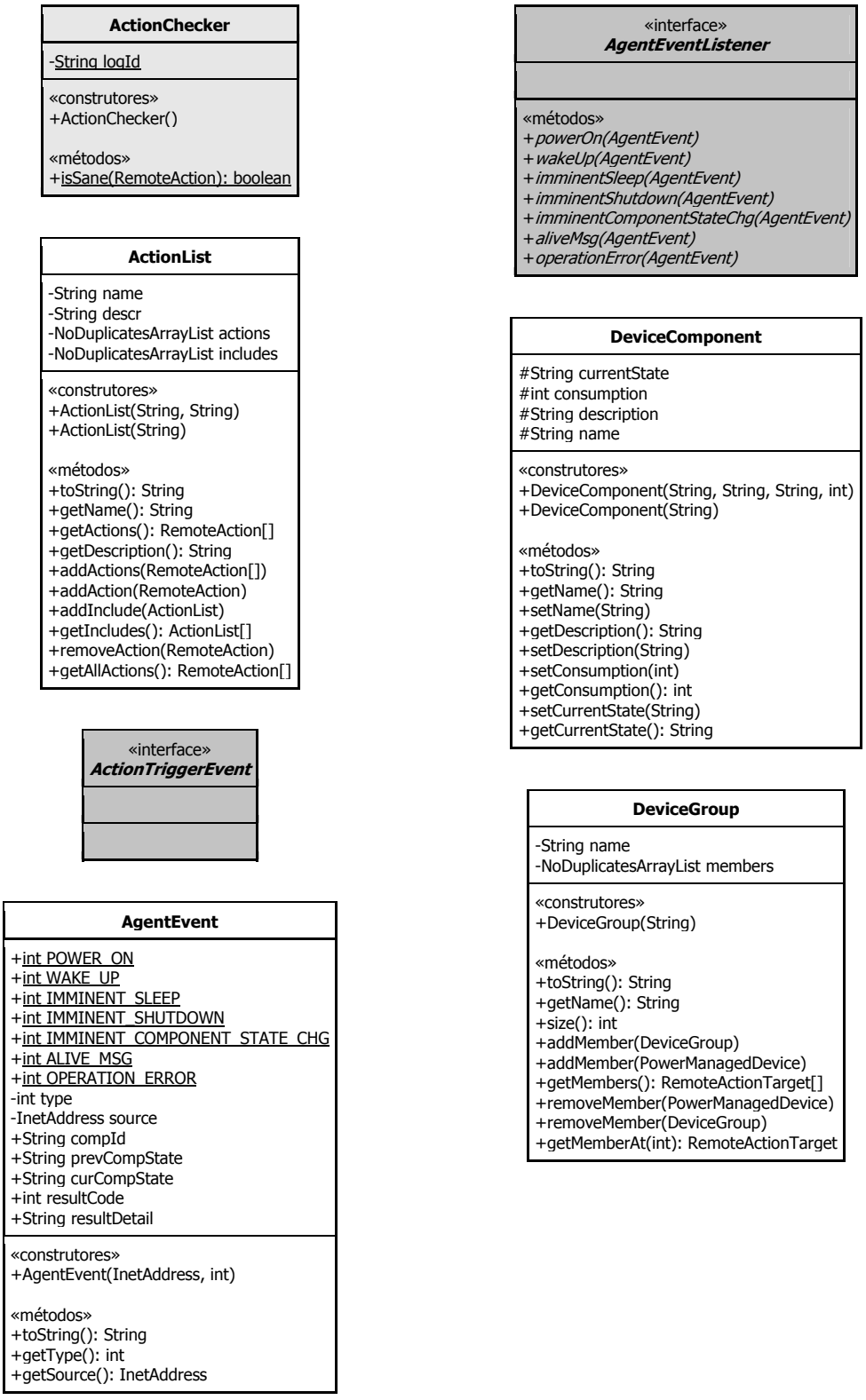


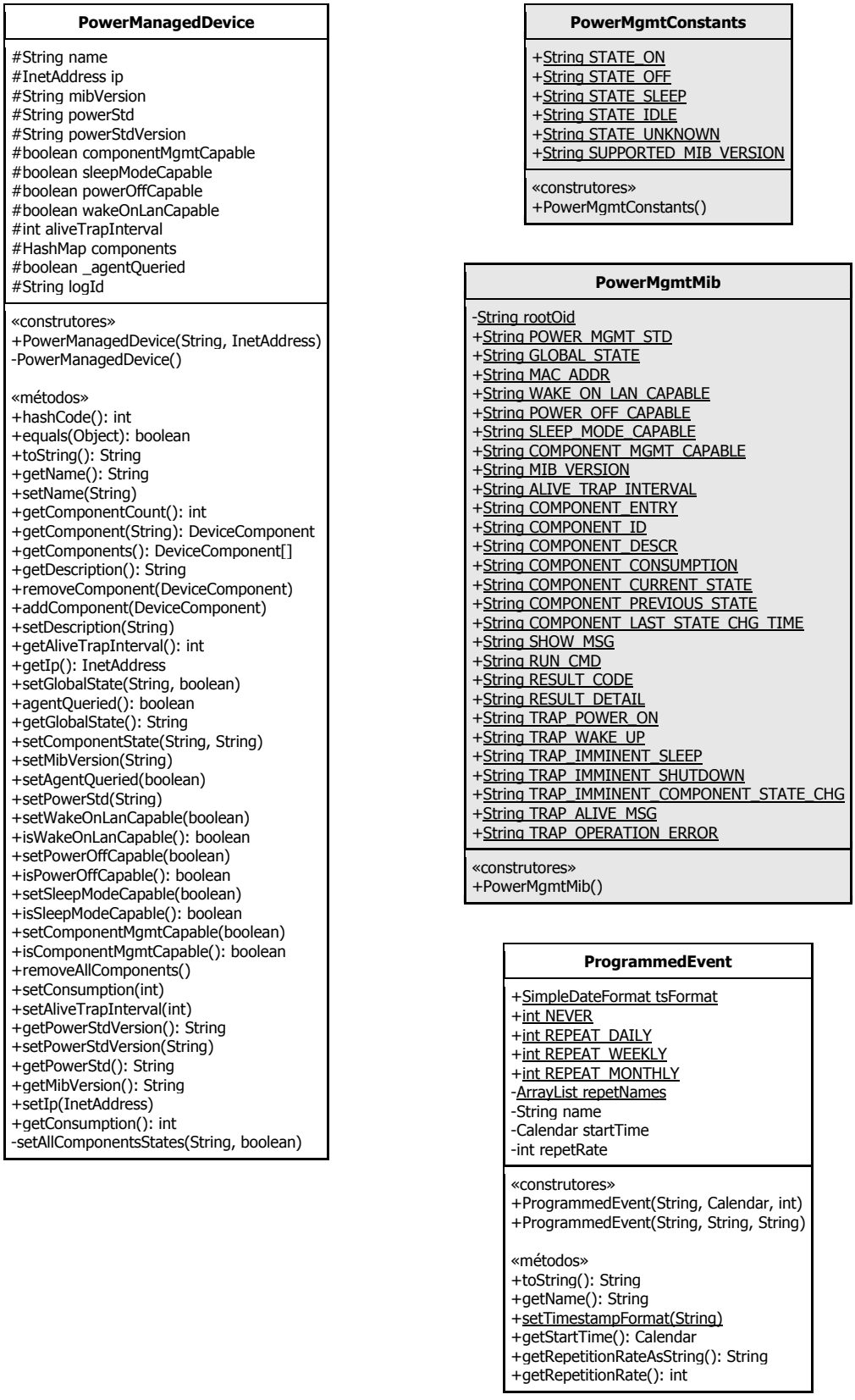


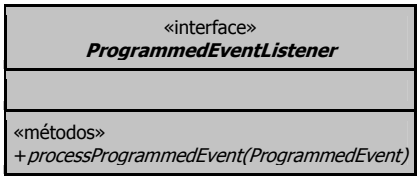


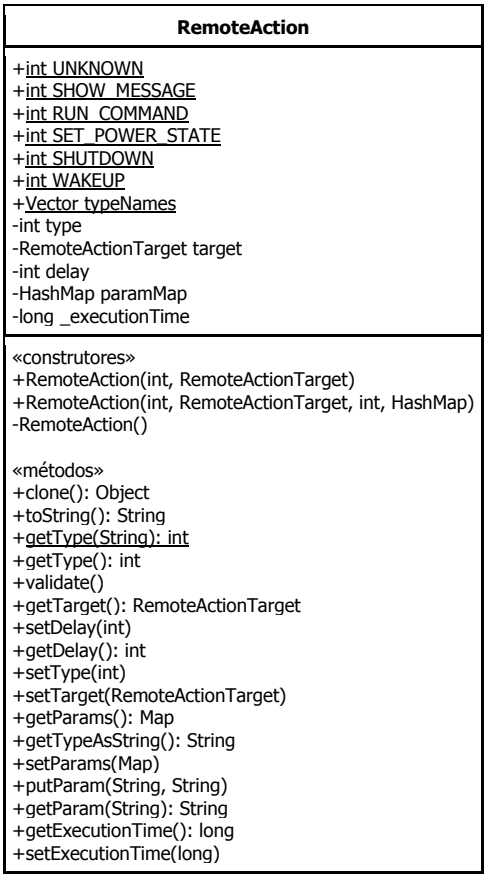


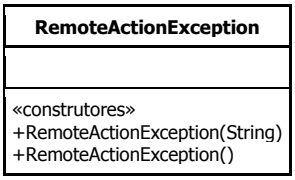
B.6 Pacote "br.ufrgs.inf.netpower.pm"

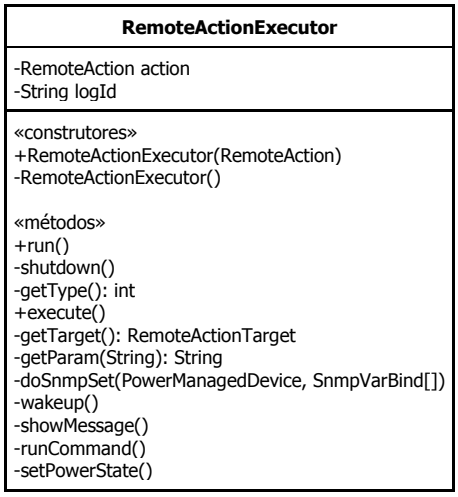


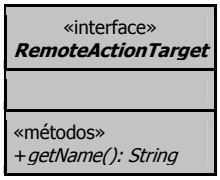

PowerManagedDevice
#String name
#InetAddress ip
#String mibVersion
#String powerStd
#String powerStdVersion
#boolean componentMgmtCapable
#boolean sleepModeCapable
#boolean powerOffCapable
#boolean wakeOnLanCapable
#int aliveTrapInterval
#HashMap components
#boolean _agentQueried
#String logId
«construtores»
+PowerManagedDevice(String, InetAddress)
-PowerManagedDevice()
«métodos»
+hashCode(): int
+equals(Object): boolean
+toString(): String
+getName(): String
+setName(String)
+getComponentCount(): int
+getComponent(String): DeviceComponent
+getComponents(): DeviceComponent[]
+getDescription(): String
+removeComponent(DeviceComponent)
+addComponent(DeviceComponent)
+setDescription(String)
+getAliveTrapInterval(): int
+getIp(): InetAddress
+setGlobalState(String, boolean)
+agentQueried(): boolean
+getGlobalState(): String
+setComponentState(String, String)
+setMibVersion(String)
+setAgentQueried(boolean)
+setPowerStd(String)
+setWakeOnLanCapable(boolean)
+isWakeOnLanCapable(): boolean
+setPowerOffCapable(boolean)
+isPowerOffCapable(): boolean
+setSleepModeCapable(boolean)
+isSleepModeCapable(): boolean
+setComponentMgmtCapable(boolean)
+isComponentMgmtCapable(): boolean
+removeAllComponents()
+setConsumption(int)
+setAliveTrapInterval(int)
+getPowerStdVersion(): String
+setPowerStdVersion(String)
+getPowerStd(): String
+getMibVersion(): String
+setIp(InetAddress)
+getConsumption(): int
-setAllComponentsStates(String, boolean)
PowerMgmtConstants
+String STATE_ON
+String STATE_OFF
+String STATE_SLEEP
+String STATE_IDLE
+String STATE_UNKNOWN
+String SUPPORTED_MIB_VERSION
«construtores»
+PowerMgmtConstants()
PowerMgmtMib
-String rootOid
+String POWER_MGMT_STD
+String GLOBAL_STATE
+String MAC_ADDR
+String WAKE_ON_LAN_CAPABLE
+String POWER_OFF_CAPABLE
+String SLEEP_MODE_CAPABLE
+String COMPONENT_MGMT_CAPABLE
+String MIB_VERSION
+String ALIVE_TRAP_INTERVAL
+String COMPONENT_ENTRY
+String COMPONENT_ID
+String COMPONENT_DESCR
+String COMPONENT_CONSUMPTION
+String COMPONENT_CURRENT_STATE
+String COMPONENT_PREVIOUS_STATE
+String COMPONENT_LAST_STATE_CHG_TIME
+String SHOW_MSG
+String RUN_CMD
+String RESULT_CODE
+String RESULT_DETAIL
+String TRAP_POWER_ON
+String TRAP_WAKE_UP
+String TRAP_IMMINENT_SLEEP
+String TRAP_IMMINENT_SHUTDOWN
+String TRAP_IMMINENT_COMPONENT_STATE_CHG
+String TRAP_ALIVE_MSG
+String TRAP_OPERATION_ERROR
«construtores»
+PowerMgmtMib()
ProgrammedEvent
+SimpleDateFormat tsFormat
+int NEVER
+int REPEAT_DAILY
+int REPEAT_WEEKLY
+int REPEAT_MONTHLY
-ArrayList repetNames
-String name
-Calendar startTime
-int repetRate
«construtores»
+ProgrammedEvent(String, Calendar, int)
+ProgrammedEvent(String, String, String)
«métodos»
+toString(): String
+getName(): String
+setTimestampFormat(String)
+getStartTime(): Calendar
+getRepetitionRateAsString(): String
+getRepetitionRate(): int


«interface»
ProgrammedEventListener
«métodos»
+processProgrammedEvent(ProgrammedEvent)

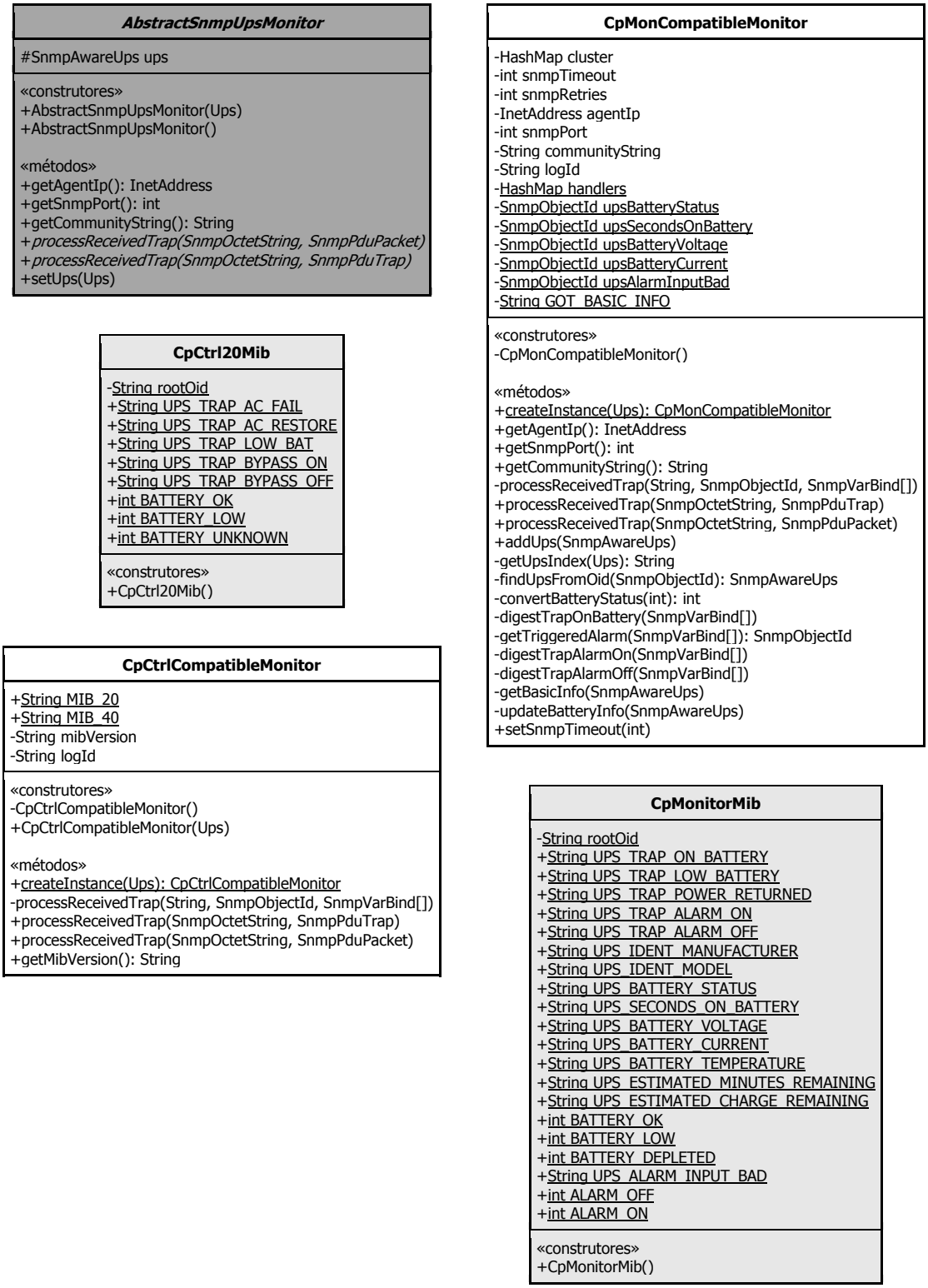

RemoteAction
+int UNKNOWN
+int SHOW_MESSAGE
+int RUN_COMMAND
+int SET_POWER_STATE
+int SHUTDOWN
+int WAKEUP
+Vector typeNames
-int type
-RemoteActionTarget target
-int delay
-HashMap paramMap
-long _executionTime
«construtores»
+RemoteAction(int, RemoteActionTarget)
+RemoteAction(int, RemoteActionTarget, int, HashMap)
-RemoteAction()
«métodos»
+clone(): Object
+toString(): String
+getType(String): int
+getType(): int
+validate()
+getTarget(): RemoteActionTarget
+setDelay(int)
+getDelay(): int
+setType(int)
+setTarget(RemoteActionTarget)
+getParams(): Map
+getTypeAsString(): String
+setParams(Map)
+putParam(String, String)
+getParam(String): String
+getExecutionTime(): long
+setExecutionTime(long)

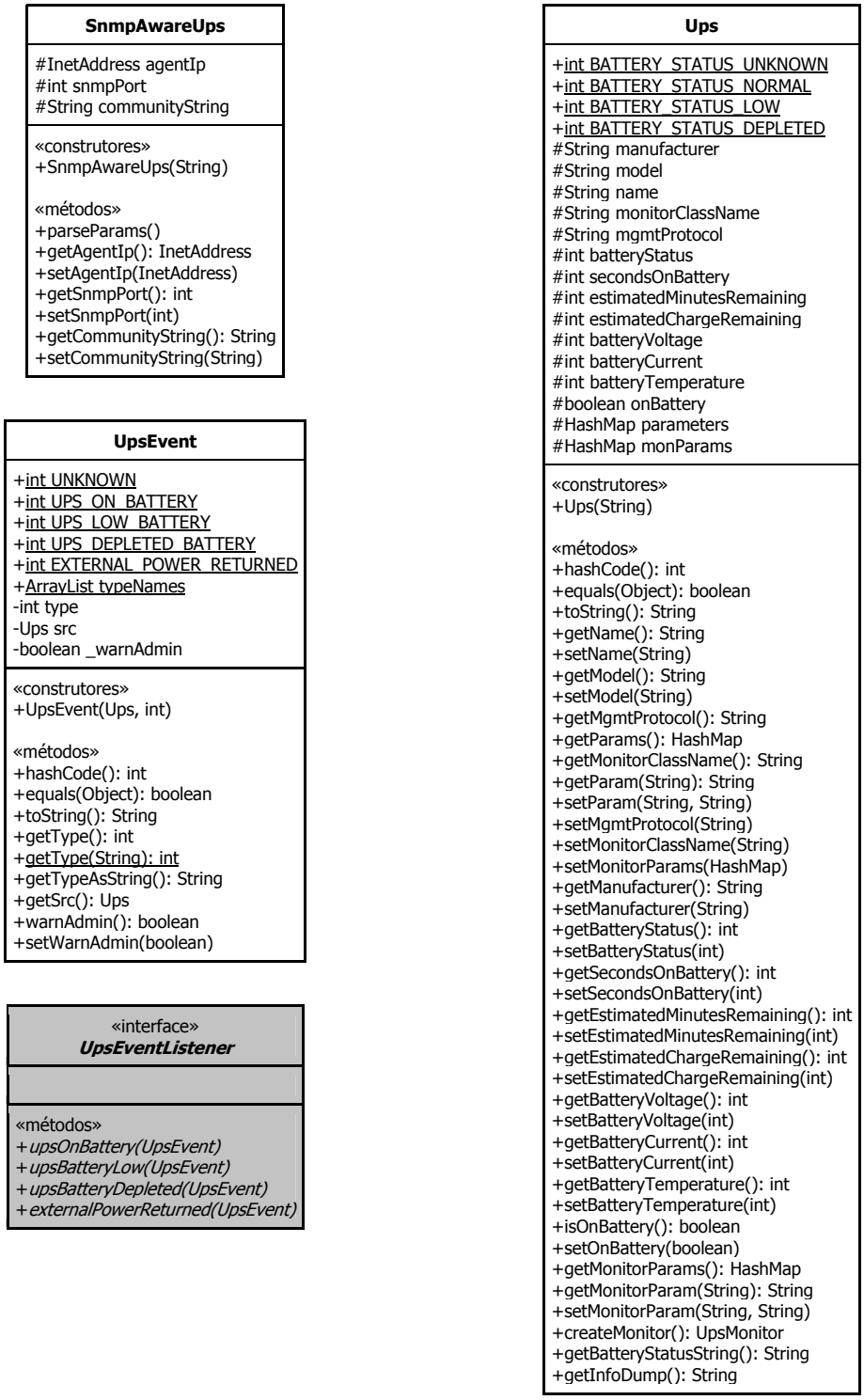

RemoteActionException
«construtores»
+RemoteActionException(String)
+RemoteActionException()

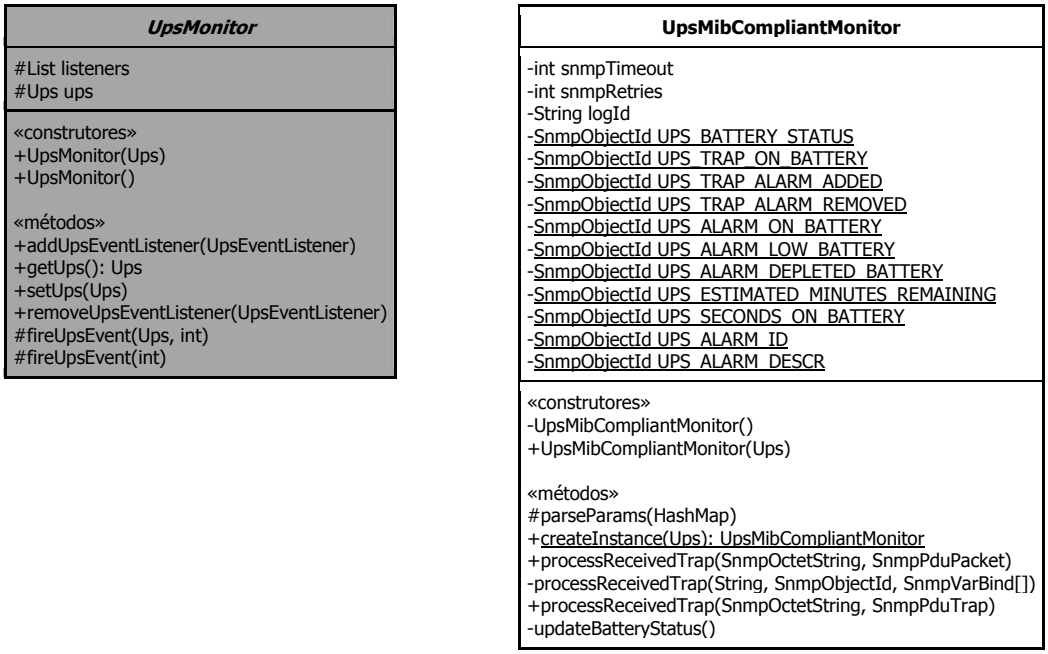

RemoteActionExecutor
-RemoteAction action
-String logId
«construtores»
+RemoteActionExecutor(RemoteAction)
-RemoteActionExecutor()
«métodos»
+run()
-shutdown()
-getType(): int
+execute()
-getTarget(): RemoteActionTarget
-getParam(String): String
-doSnmpSet(PowerManagedDevice, SnmpVarBind[])
-wakeup()
-showMessage()
-runCommand()
-setPowerState()


«interface»
RemoteActionTarget
«métodos»
+getName(): String

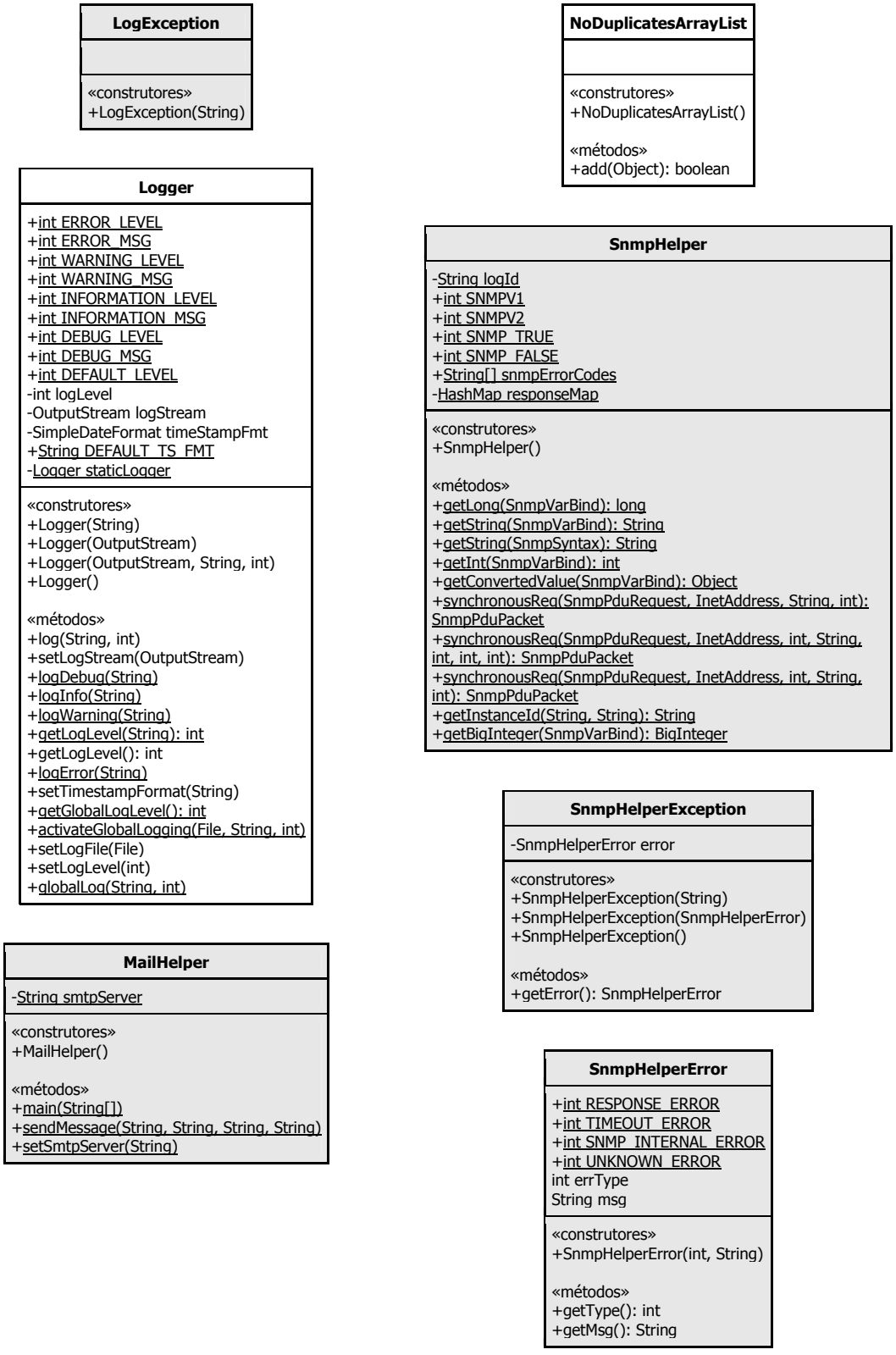
B.7 Pacote "br.ufrgs.inf.netpower.ups"

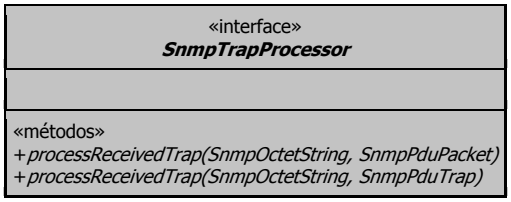


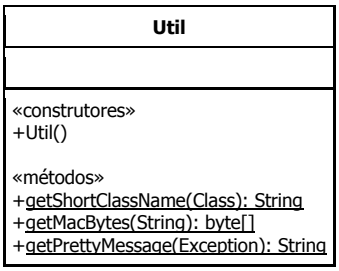

SnmpAwareUps
#InetAddress agentIp
#int snmpPort
#String communityString
«construtores»
+SnmpAwareUps(String)
«métodos»
+parseParams()
+getAgentIp(): InetAddress
+setAgentIp(InetAddress)
+getSnmpPort(): int
+setSnmpPort(int)
+getCommunityString(): String
+setCommunityString(String)
UpsEvent
+int UNKNOWN
+int UPS_ON_BATTERY
+int UPS_LOW_BATTERY
+int UPS_DEPLETED_BATTERY
+int EXTERNAL_POWER_RETURNED
+ArrayList typeNames
-int type
-Ups src
-boolean _warnAdmin
«construtores»
+UpsEvent(Ups, int)
«métodos»
+hashCode(): int
+equals(Object): boolean
+toString(): String
+getType(): int
+getType(String): int
+getTypeAsString(): String
+getSrc(): Ups
+warnAdmin(): boolean
+setWarnAdmin(boolean)
«interface»
UpsEventListener
«métodos»
+upsOnBattery(UpsEvent)
+upsBatteryLow(UpsEvent)
+upsBatteryDepleted(UpsEvent)
+externalPowerReturned(UpsEvent)
Ups
+int BATTERY_STATUS_UNKNOWN
+int BATTERY_STATUS_NORMAL
+int BATTERY_STATUS_LOW
+int BATTERY_STATUS_DEPLETED
#String manufacturer
#String model
#String name
#String monitorClassName
#String mgmtProtocol
#int batteryStatus
#int secondsOnBattery
#int estimatedMinutesRemaining
#int estimatedChargeRemaining
#int batteryVoltage
#int batteryCurrent
#int batteryTemperature
#boolean onBattery
#HashMap parameters
#HashMap monParams
«construtores»
+Ups(String)
«métodos»
+hashCode(): int
+equals(Object): boolean
+toString(): String
+getName(): String
+setName(String)
+getModel(): String
+setModel(String)
+getMgmtProtocol(): String
+getParams(): HashMap
+getMonitorClassName(): String
+getParam(String): String
+setParam(String, String)
+setMgmtProtocol(String)
+setMonitorClassName(String)
+setMonitorParams(HashMap)
+getManufacturer(): String
+setManufacturer(String)
+getBatteryStatus(): int
+setBatteryStatus(int)
+getSecondsOnBattery(): int
+setSecondsOnBattery(int)
+getEstimatedMinutesRemaining(): int
+setEstimatedMinutesRemaining(int)
+getEstimatedChargeRemaining(): int
+setEstimatedChargeRemaining(int)
+getBatteryVoltage(): int
+setBatteryVoltage(int)
+getBatteryCurrent(): int
+setBatteryCurrent(int)
+getBatteryTemperature(): int
+setBatteryTemperature(int)
+isOnBattery(): boolean
+setOnBattery(boolean)
+getMonitorParams(): HashMap
+getMonitorParam(String): String
+setMonitorParam(String, String)
+createMonitor(): UpsMonitor
+getBatteryStatusString(): String
+getInfoDump(): String


UpsMonitor
#List listeners
#Ups ups
«construtores»
+UpsMonitor(Ups)
+UpsMonitor()
«métodos»
+addUpsEventListener(UpsEventListener)
+getUps(): Ups
+setUps(Ups)
+removeUpsEventListener(UpsEventListener)
#fireUpsEvent(Ups, int)
#fireUpsEvent(int)
UpsMibCompliantMonitor
-int snmpTimeout
-int snmpRetries
-String logId
-SnmpObjectId UPS_BATTERY_STATUS
-SnmpObjectId UPS_TRAP_ON_BATTERY
-SnmpObjectId UPS_TRAP_ALARM_ADDED
-SnmpObjectId UPS_TRAP_ALARM_REMOVED
-SnmpObjectId UPS_ALARM_ON_BATTERY
-SnmpObjectId UPS_ALARM_LOW_BATTERY
-SnmpObjectId UPS_ALARM_DEPLETED_BATTERY
-SnmpObjectId UPS_ESTIMATED_MINUTES_REMAINING
-SnmpObjectId UPS_SECONDS_ON_BATTERY
-SnmpObjectId UPS_ALARM_ID
-SnmpObjectId UPS_ALARM_DESCR
«construtores»
-UpsMibCompliantMonitor()
+UpsMibCompliantMonitor(Ups)
«métodos»
#parseParams(HashMap)
+createInstance(Ups): UpsMibCompliantMonitor
+processReceivedTrap(SnmpOctetString, SnmpPduPacket)
-processReceivedTrap(String, SnmpObjectId, SnmpVarBind[])
+processReceivedTrap(SnmpOctetString, SnmpPduTrap)
-updateBatteryStatus()

B.8 Pacote "br.ufrgs.inf.netpower.util"




«interface»
SnmpTrapProcessor
«métodos»
+processReceivedTrap(SnmpOctetString, SnmpPduPacket)
+processReceivedTrap(SnmpOctetString, SnmpPduTrap)


Util
«construtores»
+Util()
«métodos»
+getShortClassName(Class): String
+getMacBytes(String): byte[]
+getPrettyMessage(Exception): String

Apêndice C - Código-fonte da MIB para gerência de energia

```
-- File Name : power-mgmt-MIB.mib
-- Date      : Sat Apr 20 21:35:00 GMT-03:00 2002
-- Author    : Luis Fernando Pollo - PPGC - UFRGS

PowerManagementMIB    DEFINITIONS ::= BEGIN
       IMPORTS
              TruthValue, DateAndTime, DisplayString, TimeInterval, MacAddress
                     FROM SNMPv2-TC
              experimental, Integer32
                     FROM SNMPv2-SMI
              MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE
                     FROM SNMPv2-SMI
              MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
                     FROM SNMPv2-CONF;

       powerMgmtMIB   MODULE-IDENTITY
              LAST-UPDATED   "200201022328Z"
              ORGANIZATION   "PPGC - UFRGS"
              CONTACT-INFO   "Luis Fernando Pollo
                                 Universidade Federal do Rio Grande do Sul
                                 Instituto de Informatica
                                 Av. Bento Goncalves, 9500 - Campus do Vale - Bloco IV
                                 Bairro Agronomia - Porto Alegre - RS - Brasil
                                 91501-970 Caixa Postal: 15064
                                 Phone: +55 (51) 3316-6159
                                 Fax: +55 (51) 3316-7308
                                 E-mail: pollo@inf.ufrgs.br"
              DESCRIPTION    "A MIB module to aid in remote power management of
                             electronic devices."

              REVISION       "200204202135Z"
              DESCRIPTION    "My last 2 cents to the MIB module... :-)"
              ::=  {  experimental  1  }

       devAttributes  OBJECT IDENTIFIER
              ::=  {  powerMgmtMIB  1  }

       devCapabilities       OBJECT IDENTIFIER
              ::=  {  powerMgmtMIB  2  }

       agentAttributes       OBJECT IDENTIFIER
              ::=  {  powerMgmtMIB  3  }

       sysInteraction OBJECT IDENTIFIER
              ::=  {  powerMgmtMIB  5  }

       traps  OBJECT IDENTIFIER
              ::=  {  powerMgmtMIB  6  }

       componentTable OBJECT-TYPE
              SYNTAX         SEQUENCE  OF  ComponentEntry
              MAX-ACCESS     not-accessible
              STATUS         current
              DESCRIPTION
                     "Component table. Lists the components (internal or peripheral
                     devices) that are part of the system. Has at least one entry that
                     represents the entire system ('global')."
              ::=  { powerMgmtMIB  4 }

       componentEntry OBJECT-TYPE
              SYNTAX         ComponentEntry
              MAX-ACCESS     not-accessible
              STATUS         current
              DESCRIPTION
                     "An entry of the components table."
              INDEX          {  componentId  }
              ::=  { componentTable 1 }

       ComponentEntry  ::=  SEQUENCE {
              componentId  DisplayString,
```

```
        componentDescr  DisplayString,
        componentConsumption  Integer32,
        componentCurrentState  DisplayString,
        componentPreviousState  DisplayString,
        componentLastStateChgTime  DateAndTime
        }


componentId    OBJECT-TYPE
        SYNTAX                  DisplayString (SIZE (0..20))
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
                        "Unique component identifier."
        ::=  {  componentEntry  1  }


componentDescr OBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
                        "Description of the component."
        ::=  {  componentEntry  2  }


componentConsumption  OBJECT-TYPE
        SYNTAX                  Integer32  ( -2147483648 .. 2147483647  )
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
                        "Current power consumption value for the component."
        ::=  {  componentEntry  3  }


componentCurrentState OBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-write
        STATUS                  current
        DESCRIPTION
                        "Current power state of the component."
        ::=  {  componentEntry  4  }


componentPreviousStateOBJECT-TYPE
        SYNTAX                  DisplayString
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
                        "Describes the previous power state of the component. This
                        is mostly for logging purposes."
        ::=  {  componentEntry  5  }


componentLastStateChgTime     OBJECT-TYPE
        SYNTAX                  DateAndTime
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
                        "Time of the last power state change."
        ::=  {  componentEntry  6  }

mibVersion     OBJECT-TYPE
        SYNTAX                  OCTET STRING
        MAX-ACCESS              read-only
        STATUS                  current
        DESCRIPTION
                        "Version of the MIB implemented by the agent."
        ::=  {  agentAttributes  1  }

aliveTrapInterval     OBJECT-TYPE
        SYNTAX                  TimeInterval
        MAX-ACCESS              read-write
        STATUS                  current
        DESCRIPTION
                        "Interval, in minutes, at which 'trapAliveMsg' is sent."
        DEFVAL                  { 1 }
```

```
        ::=  {  agentAttributes  2  }

powerMgmtStd   OBJECT-TYPE
        SYNTAX                DisplayString
        MAX-ACCESS            read-only
        STATUS                current
        DESCRIPTION
                      "Power management standard supported by this device.
                      Should include version information (e.g. 'ACPI 1.0')"
        ::=  {  devAttributes  1  }

globalState    OBJECT-TYPE
        SYNTAX                DisplayString
        MAX-ACCESS            read-write
        STATUS                current
        DESCRIPTION
                      "The current global state of the device. Must be the same
                      state of the entry named 'global' in 'componentTable'.
                      This variables allows us to send the current state of the
                      device along with 'alive' messages. A SET operation here is
                      equivalent to a SET with the same value to
                      componentTable.componentId='global'.componentCurrentState"
        ::=  {  devAttributes  2  }

macAddr        OBJECT-TYPE    -- do NOT change this to macAddress, it will
                              -- conflict with the SYNTAX definition
        SYNTAX                MacAddress
        MAX-ACCESS            read-only
        STATUS                current
        DESCRIPTION
                      "MAC address of the network interface used by the agent to
                      communicate with the management station"
        ::=  {  devAttributes  3  }

wakeOnLanCapable      OBJECT-TYPE
        SYNTAX                TruthValue
        MAX-ACCESS            read-only
        STATUS                current
        DESCRIPTION
                      "Indicates whether this device can be turned on remotely."
        DEFVAL                { false }
        ::=  {  devCapabilities  1  }

powerOffCapable       OBJECT-TYPE
        SYNTAX                TruthValue
        MAX-ACCESS            read-only
        STATUS                current
        DESCRIPTION
                      "Indicates whether the agent can turn this device's power
                      supply off."
        ::=  {  devCapabilities  2  }

sleepModeCapable      OBJECT-TYPE
        SYNTAX                TruthValue
        MAX-ACCESS            read-only
        STATUS                current
        DESCRIPTION
                      "Indicates whether this device supports one or more sleep
                      (standby) modes."
        ::=  {  devCapabilities  3  }

componentMgmtCapable  OBJECT-TYPE
        SYNTAX                TruthValue
        MAX-ACCESS            read-only
        STATUS                current
        DESCRIPTION
                      "Indicates whether this device is able to power
                      manage any of its hardware components (such as hard disks
                      and processors for PCs) or other peripherals (printers,
                      etc) individually."
        ::=  {  devCapabilities  4  }

showMsg OBJECT-TYPE
        SYNTAX                DisplayString
        MAX-ACCESS            read-write
        STATUS                current
        DESCRIPTION
```

```
                    "When this object is set, its value is showed to the
                    system user(s), if applicable."
        ::=  {  sysInteraction  1  }

runCmd  OBJECT-TYPE
        SYNTAX              OCTET STRING
        MAX-ACCESS          read-write
        STATUS              current
        DESCRIPTION
                    "When this object is set, the agent makes an attempt to
                    execute the specified command or script."
        ::=  {  sysInteraction  2  }

resultCode     OBJECT-TYPE
        SYNTAX              Integer32
        MAX-ACCESS          read-only
        STATUS              current
        DESCRIPTION
                    "A result code for the last operation executed by the
                    agent on the Operating System."
        ::=  {  sysInteraction  3  }

resultDetail   OBJECT-TYPE
        SYNTAX              DisplayString
        MAX-ACCESS          read-only
        STATUS              current
        DESCRIPTION
                    "Detailed information about the results of the last
                    operation performed on the Operating System."
        ::=  {  sysInteraction  4  }

trapPowerOn    NOTIFICATION-TYPE
        STATUS              current
        DESCRIPTION
                    "The system has booted, or the agent has been (re)started."
        ::=  {  traps  1  }

trapWakeUp     NOTIFICATION-TYPE
        STATUS              current
        DESCRIPTION
                    "The system has awakened from a sleep state."
        ::=  {  traps  2  }

trapImminentSleep     NOTIFICATION-TYPE
        OBJECTS             { globalState }
        STATUS              current
        DESCRIPTION
                    "The system is about to enter a sleep mode."
        ::=  {  traps  3  }

trapImminentShutdown  NOTIFICATION-TYPE
        STATUS              current
        DESCRIPTION
                    "The system is about to be shutdown."
        ::=  {  traps  4  }

trapImminentComponentStateChg NOTIFICATION-TYPE
        OBJECTS             { componentId , componentCurrentState ,
                              componentPreviousState }
        STATUS              current
        DESCRIPTION
                    "Indicates that a system component is about to have its
                     state changed. Preferably, this trap should NOT be used
                    for componentId='global'. Use one of the specialized traps
                    for notifying system-wide change instead (i.e.
                    trapImminentSleep, trapImminentShutdown, trapWakeUp)"
        ::=  {  traps  5  }

trapAliveMsg   NOTIFICATION-TYPE
        OBJECTS             { globalState }
        STATUS              current
        DESCRIPTION
                    "Sent every 'aliveTrapInterval' minutes to let the
                    manager know the device is active."
        ::=  {  traps  6  }

trapOperationError    NOTIFICATION-TYPE
```

```
                OBJECTS              { resultCode ,resultDetail  }
                STATUS               current
                DESCRIPTION
                                "Notifies the manager that something went wrong while
                                executing an operation."
                ::=  {  traps  7  }


-- conformance information

powerMgmtMIBConformance
                OBJECT IDENTIFIER ::= { powerMgmtMIB 7 }

powerMgmtMIBCompliances
                OBJECT IDENTIFIER ::= { powerMgmtMIBConformance 1 }

powerMgmtMIBGroups
                OBJECT IDENTIFIER ::= { powerMgmtMIBConformance 2 }


-- compliance statements

basicPowerMgmtCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "The compliance statement for agents that support basic
            handling of their hosts' power management."
    MODULE  -- this module
        MANDATORY-GROUPS { basicDevAttributesGroup, basicDevCapabilitiesGroup,
                           basicAgentAttributesGroup, basicComponentEntryGroup,
                           basicSysInteractionGroup, basicNotificationsGroup }

    ::= { powerMgmtMIBCompliances 1 }

fullPowerMgmtCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "The compliance statement for agents that support full
            handling of their hosts' power management."
    MODULE  -- this module
        MANDATORY-GROUPS { basicDevAttributesGroup, basicDevCapabilitiesGroup,
                           basicAgentAttributesGroup, fullComponentEntryGroup,
                           basicSysInteractionGroup, fullNotificationsGroup }

    ::= { powerMgmtMIBCompliances 2 }

-- units of conformance

basicDevAttributesGroup OBJECT-GROUP
    OBJECTS { powerMgmtStd,
              globalState,
              macAddr }
    STATUS  current
    DESCRIPTION
            "A collection of objects providing basic information about
             the managed device's power-management-related attributes."
    ::= { powerMgmtMIBGroups 1 }

basicDevCapabilitiesGroup OBJECT-GROUP
    OBJECTS { wakeOnLanCapable,
              powerOffCapable,
              sleepModeCapable,
              componentMgmtCapable }
    STATUS  current
    DESCRIPTION
            "A collection of objects providing basic hints about the
             the managed device's power management capabilities."
    ::= { powerMgmtMIBGroups 2 }

basicAgentAttributesGroup OBJECT-GROUP
    OBJECTS { mibVersion,
              aliveTrapInterval }
    STATUS  current
    DESCRIPTION
            "Objects that provide basic SNMP related information."
    ::= { powerMgmtMIBGroups 3 }
```

```
basicComponentEntryGroup OBJECT-GROUP
    OBJECTS { componentId,
              componentCurrentState }
    STATUS  current
    DESCRIPTION
            "Objects that provide basic power-management-related
            information about an internal (e.g. hard drive) or peripheral
            (e.g. printer) component of the device."
    ::= { powerMgmtMIBGroups 4 }

fullComponentEntryGroup OBJECT-GROUP
    OBJECTS { componentId,
              componentDescr,
              componentConsumption,
              componentCurrentState,
              componentPreviousState,
              componentLastStateChgTime }
    STATUS  current
    DESCRIPTION
            "Objects that provide detailed power-management-related
            information about an internal (e.g. hard drive) or peripheral
            (e.g. printer) component of the device."
    ::= { powerMgmtMIBGroups 5 }

basicSysInteractionGroup OBJECT-GROUP
    OBJECTS { showMsg,
              runCmd,
              resultCode,
              resultDetail }
    STATUS  current
    DESCRIPTION
            "Objects that provide for the management station to interact
            indirectly with the device's Operating System, and also for
            the agent to store operation result information."
    ::= { powerMgmtMIBGroups 6 }

basicNotificationsGroup NOTIFICATION-GROUP
    NOTIFICATIONS { trapPowerOn,
                    trapImminentShutdown,
                    trapAliveMsg,
                    trapOperationError }
    STATUS  current
    DESCRIPTION
            "Indispensable notifications, which the agent must be able to send."
    ::= { powerMgmtMIBGroups 7 }

fullNotificationsGroup NOTIFICATION-GROUP
    NOTIFICATIONS { trapPowerOn,
                    trapWakeUp,
                    trapImminentSleep,
                    trapImminentShutdown,
                    trapImminentComponentStateChg,
                    trapAliveMsg,
                    trapOperationError }
    STATUS  current
    DESCRIPTION
            "Set of notifications that a full-featured agent must be able to send."
    ::= { powerMgmtMIBGroups 8 }

END
```

Apêndice D - Código-fonte, MIBs e outros arquivos

O código-fonte completo da aplicação de gerência e do agente-protótipo implementados estão disponíveis no CD-ROM incluso, assim como todas as MIBs utilizadas, arquivos de configuração, documentos e outros arquivos relevantes.

Bibliografia

[ACP2002] The ACPI4LINUX Project. 2002. Disponível em: <http://phobos.fs.tum.de/acpi/index.html.en>. Acesso em: 12 jun. 2002.

[ADV95] ADVANCED MICRO DEVICES, INC. **Magic Packet technology white paper.** 1995. Disponível em: <http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/20213.pdf>. Acesso em: 05 mar. 2002.

[ADV2002] ADVENTNET, INC. **AdventNet Simulation Toolkit.** Disponível em: <http://www.adventnet.com/products/simulator/index.html>. Acesso em: 11 jun. 2002.

[ANE2002] ANEEL – AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA. **Tarifas referentes aos anos de 1999 e 2002.** Disponível em: <http://www.aneel.gov.br/aplicacoes/tarifamedia/Default.cfm>. Acesso em: 03 jul. 2002.

[ASU2000] ASUSTEK COMPUTER INC. **ASUS P3V133 User's Manual.** [S.l.], 2000.

[BEC2002] BECKER, Donald. **Informações sobre o driver da placa de rede 3C905C para Linux.** [Mensagem pessoal]. Mensagem recebida por <pollo@inf.ufrgs.br> em 13 maio 2002.

[BOG88] BOGGS, D.; MOGUL, J.; KENT, C. Measured Capacity of an Ethernet: Myths and Reality. **Computer Communication Review**, New York, v.18, n. 4, p. 222-234, Aug. 1988. Trabalho apresentado no Symposium on Communications Architectures and Protocols, SIGCOMM, 1988.

[BOR2002] BORLAND SOFTWARE CORPORATION. **Borland C++Builder Compiler.** Disponível em: <http://www.borland.com/cbuilder/cppcomp/index.html>. Acesso em: 11 jun. 2002.

[BUL98] BULIGON, Clairton; LEBOUTE, Mario M.; JANSCH-PÔRTO, Ingrid E. S. Gerenciamento de energia em redes locais usando CP-monitor. In: SALÃO DE INICIAÇÃO CIENTÍFICA, 10., 1998, Porto Alegre. **Livro de resumos**. Porto Alegre: UFRGS / PROPESQ, 1998. p. 33.

[CAM99] CAMPOS, Rui. **Informática no Brasil:** fatos e números. 3. ed. São Paulo: Fenasoft Feiras Comerciais, 1999. Disponível em: <http://www.mct.gov.br/sepin/palestras/infbr91.pdf>. Acesso em: 28 jan. 2002.

[CAS94] CASE, J. **UPS Management Information Base:** Request for Comments 1628. SNMP Research, Inc., 1994. Disponível em: <ftp://ftp.isi.edu/in-notes /rfc1628.txt>. Acesso em: 01 jul. 2002.

[CAS99] CASE, J. et al. **Introduction to version 3 of the Internet-standard Network Management Framework:** Request for Comments 2570, 1999. Disponível em: <ftp://ftp.isi.edu/in-notes /rfc2570.txt>. Acesso em: 01 jul. 2002.

[COM2000] COMPAQ COMPUTER CORPORATION; INTEL CORPORATION; MICROSOFT CORPORATION; PHOENIX TECHNOLOGIES LTD.; TOSHIBA CORPORATION. **Advanced Configuration and Power Interface Specification.** Revisão 2.0. 2000. Disponível em: <http://www.acpi.info/index.html>. Acesso em: 04 jan. 2002.

[CRA95] CRAMER, Michael. **The secondary savings of the Energy Star program.** Masters Thesis, Department of Mechanical Engineering, University of California, Berkeley, 1995.

[CRI91] CRISTIAN, F. Understanding fault-tolerant distributed systems. **Communications of the ACM**, New York, v. 34, n. 2, p. 57-78, Feb. 1991.

[CRY2002] THE CRYPTIX FOUNDATION LIMITED. **Cryptix SASL Library.** Disponível em: <http://www.cryptix.org/products/sasl/index.html>. Acesso em: 03 jul. 2002.

[DOC2000] DOC – UNITED STATES DEPARTMENT OF COMMERCE. **Computer equipment market in Brazil.** [EUA], 2000. Disponível em: <http://osecnt13.osec.doc.gov/public.nsf/TPMain/Brazil_Computer>. Acesso em: 28 jan. 2002.

[DOE2000] DOE – UNITED STATES DEPARTMENT OF ENERGY. **Annual Energy Outlook 2000 with Projections to 2020.** [EUA], 2000. Disponível em: <http://www.eia.doe.gov/oiaf/aeo/>. Acesso em: 28 jan. 2002.

[ENE2001] EIA – ENERGY INFORMATION ADMINISTRATION, UNITED STATES DEPARTMENT OF ENERGY. **International Electricity Information.** [EUA], 2000. Disponível em: <http://www.eia.doe.gov/emeu/international/electric.html#IntlConsumption>. Acesso em: 28 jan. 2002.

[EPA2000] EPA – UNITED STATES ENVIRONMENTAL PROTECTION AGENCY. **EPA Innovation Annual Report**. [EUA], 2000. Disponível em: <http://www.epa.gov/innovation/decade/2partnerships.htm>. Acesso em: 30 abr. 2002.

[EPA2002] EPA – UNITED STATES ENVIRONMENTAL PROTECTION AGENCY. **ENERGY Star.** [EUA], 2002. Disponível em: <http://www.energystar.gov>. Acesso em: 08 jan. 2002.

[FER99] FERREIRA, Aurélio Buarque de Holanda. **Aurélio século XXI:** o dicionário da língua portuguesa. 3. ed. rev. e ampl. Rio de Janeiro: Nova Fronteira, 1999.

[FOC2002] FOCK, Frank. **AGENT++ - SNMPv1/v2c/v3 Agent API for C++.** Disponível em: <http://www.agentpp.com/>. Acesso em: 11 jun. 2002.

[GLE2001] GLENN, Ariel. **Linux ACPI-HOWTO**. 2001. Disponível em: <http://www.columbia.edu/~ariel/acpi/acpi_howto.txt>. Acesso em: 18 jun. 2002.

[GRO2002] GROVER, Andrew. **Informações sobre a arquitetura de gerência de energia no sistema operacional Linux.** 13 jun. 2002. Informações obtidas por e-mail na lista de discussão sobre implementação de ACPI no Linux: <mailto:acpi-devel@lists.sourceforge.net>.

[HAY2001] HAYES, Bryan. The computer and the dynamo. **American Scientist**, [S.l.], v. 89, n. 5, p. 390-394, September-October 2001.

[HEU2000] HEUSER, Werner. **Linux Laptop-HOWTO**. v2.2a, 3 de novembro de 2000. Disponível em: <http://www.tldp.org/HOWTO/Laptop-HOWTO.html>. Acesso em: 18 jun. 2002.

[HEU2000a] HEUSER, Werner; HAMPTON, Wade W. **Linux Ecology HOWTO.** v0.8, 4 de novembro de 2000. Disponível em: <http://www.tldp.org/HOWTO/Ecology-HOWTO.html>. Acesso em: 18 jun. 2002.

[HUB99] HUBER, Peter; MILLS, Mark P. Dig more coal – the PCs are coming. **Forbes**, [S.l.], p. 70–72, May 31, 1999.

[INT96] INTEL CORPORATION & MICROSOFT CORPORATION. **Advanced Power Management (APM) BIOS Interface Specification.** Revisão 1.2. 1996. Disponível em: <http://www.microsoft.com/hwdev/archive/BUSBIOS/amp_12.asp>. Acesso em: 12 jan. 2002.

[INT99] INTEL CORPORATION. **SE440BX-2 Motherboard Technical Product Specification.** 1999. Disponível em: <http://support.intel.com/support/motherboards/desktop/se440bx2/>. Acesso em: 19 jun. 2002.

[INT99a] INTEL CORPORATION. **Intel Desktop Board CC820 Technical Product Specification.** 1999. Disponível em: <http://support.intel.com/support/motherboards/desktop/cc820/>. Acesso em: 19 jun. 2002.

[INT2001] INTEL CORPORATION. **Instantly available technology.** Disponível em: <http://developer.intel.com/technology/IAPC/index.htm>. Acesso em: 10 maio 2001.

[KAW2001] KAWAMOTO, Kaoru et al. **Electricity Used by Office Equipment and Network Equipment in the U.S.**: Detailed Report and Appendices. Berkeley: Lawrence Berkeley National Laboratory, University of California, 2001. Disponível em: <http://eetd.lbl.gov/ea/>. Acesso em: 28 jan. 2002.

[KER2002] KERBER, Fabiano Schommer. **Projeto e implementação de um agente de gerência de energia para redes de computadores.** 2002. Projeto de diplomação (Bacharelado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[KOR2002] KORYTOWSKI, Ivo. **Ivo Korytowski's English-Portuguese Translator's Dictionary.** Dicionário eletrônico para o programa Babylon. 2002. Disponível em: <http://info.babylon.com/cgi-bin/temp.cgi?id=17537&layout=gt.html>. Acesso em: 20 fev. 2001.

[KRO99] KROLOW, Roger; PERES, André; JANSCH-PÔRTO, Ingrid E. S. Management Information Base for Power Management in Computer Networks. In: LATIN AMERICAN NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, LANOMS, 1., 1999, Rio de Janeiro. **Proceedings...** Florianópolis: UFSC, 1999.

[KRO99a] KROLOW, Roger Al-Alam. **Gerência de energia em computadores pessoais.** 1999. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[KRO2001] KROLOW, Roger Al-Alam. **Sistema de Controle de Consumo para Redes de Computadores.** 2001. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[LEI96] LEINWAND, Allan; CONROY, Karen F. **Network Management.** 2nd ed. [S.l.]: Addison Wesley Longman, 1996.

[LOR98] LORCH, J. R.; SMITH, A. J. Software strategies for portable computer energy management. **IEEE Personal Communications**, New York, v.5, n.3, p.60-73, June 1998.

[LU99] LU, Y.-H.; SIMUNIC, T.; DE MICHELI, G. Software controlled power management. In: INTERNATIONAL WORKSHOP ON HARDWARE/SOFTWARE CODESIGN, 7., 1999, Roma. **Proceedings...** Roma: [s.n.], 1999. p. 157-161.

[MEL94] MELQUIST, Peter Erik. **SNMP++:** an object-oriented approach to developing network management applications. [S.l.]: Hewlett-Packard Professional Books / Prentice Hall, 1994.

[MIC98] MICROSOFT CORPORATION. **Microsoft Press dicionário de informática.** Tradução de Valeria Chamon. Rio de Janeiro: Campus, 1998.

[MIC2002] MICROSOFT CORPORATION. **OnNow and Power Management.** Disponível em: <http://www.microsoft.com/hwdev/tech/onnow/>. Acesso em: 10 maio 2002.

[MIC2002a] MICROSOFT CORPORATION. **Platform SDK:** Power Management Reference. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/power/power_3v6t.asp>. Acesso em: 28 jun. 2002.

[MIC2002b] MICROSOFT CORPORATION. **Windows Power Management.** Disponível em: <http://www.microsoft.com/hwdev/tech/onnow/winpowmgmt.asp>. Acesso em: 13 jun. 2002.

[MME2000] MME – MINISTÉRIO DE MINAS E ENERGIA. **Balanço Energético Nacional.** Brasília, 2000. Disponível em: <ftp://ftp.mme.gov.br/pub/Balanco/BEN/>. Acesso em: 28 jan. 2002.

[MOU86] MOURA, José A. B. **Redes locais de computadores:** protocolos de alto nível e avaliação de desempenho. São Paulo: Mcgraw-Hill, 1986.

[NOR96] NORDMAN, Bruce et al. Measured Energy Savings and Performance of Power-Managed Personal Computers and Monitors. In: ACEEE 1996 SUMMER STUDY ON ENERGY EFFICIENCY IN BUILDINGS, 1996, Washington, DC. **Proceedings…** Washington, DC: American Council for an Energy-Efficient Economy, 1996.

[NOR97] NORDMAN, Bruce et al. **User guide to power management for PCs and monitors.** Berkeley: Lawrence Berkeley National Laboratory, University of California, 2001. Disponível em: <http://eetd.lbl.gov/EA/BEA/LBLReports/39466/>. Acesso em: 05 dez. 2001.

[PEN2002] PENNARUN, Avery. **The Linux APM daemon.** Disponível em: <http://worldvisions.ca/~apenwarr/apmd/>. Acesso em: 12 jun. 2002.

[PER2000] PERES, André. **Definição e construção de uma base de informações de gerência para UPS.** 2000. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[PLU82] PLUMMER, David C. **An Ethernet address resolution protocol:** Request for Comments 826. Network Working Group. Novembro, 1982. Disponível em: <ftp://ftp.isi.edu/in-notes/rfc826.txt>. Acesso em: 01 jul. 2002.

[POL2000] POLLO, Luis F. **Implementação de uma interface gráfica Web para o sistema NetPower.** 2000. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

[POL2001] POLLO, Luis F. **Plano de Estudos e Pesquisa.** 2001. Instituto de Informática, UFRGS, Porto Alegre.

[POW2002] POWERWARE CORPORATION. **Power management software.** Disponível em: <http://www.powerware.com/Products/Software/Software.asp>. Acesso em: 22 fev. 2002.

[RAL97] RALOFF, Janet. Must We Pull the Plug? **Science News,** [S.l.], v. 152, n. 17, October 25, 1997.

[RIV92] RIVEST, R. L. **The MD5 message-digest algorithm:** Request for Comments 1321. Internet Activities Board, Internet Privacy Task Force, 1992. Disponível em: <ftp://ftp.isi.edu/in-notes/rfc1321.txt>. Acesso em: 01 jul. 2002.

[SHO84] SHOCH, John F.; HUPP, Jon A. Measured performance of an Ethernet local network. **Communications of the ACM**, New York, v. 23, n. 2, p. 711-21, Dec. 1980.

[SIQ98] SIQUEIRA, Tórgan F.; LEBOUTE, Mario M.; JANSCH-PÔRTO, Ingrid E. S. Utilização da arquitetura documento-visão no desenvolvimento de software para gerenciamento de energia em redes locais. In: SALÃO DE INICIAÇÃO CIENTIFICA, 10., 1998, Porto Alegre. **Livro de resumos.** Porto Alegre: UFRGS / PROPESQ, 1998. p. 33.

[SUN2000] SUN MICROSYSTEMS, INC. **The Scoop on RMI and SSL.** 2000. Disponível em: <http://java.sun.com/products/jdk/1.2/docs/guide/rmi/SSLInfo.html>. Acesso em: 03 jul. 2002.

[SUN2002] SUN MICROSYSTEMS, INC. **Java Authentication and Authorization Service (JAAS).** Disponível em: <http://java.sun.com/products/jaas/>. Acesso em: 03 jul. 2002.

[SUR2002] SUNRISE TELECOM, INC. **LanExplorer.** Disponível em: <http://www.sunrisetelecom.com/lansoftware/>. Acesso em: 02 maio 2002.

[TCO99] 3COM CORPORATION. **EtherLink PCI Network Interface Cards User Guide.** 1999. Disponível em: <http://support.3com.com/infodeli/tools/nic/3c905c/905cuser.pdf>. Acesso em: 21 jun. 2002.

[UNI2001] UNIVESITY OF MICHIGAN. **UM Guide to Green Computing, Energy Management.** Disponível em: <http://www.energymanagement.umich.edu/ems/Green_Computing.html>. Acesso em: 16 nov. 2001.

[VES2002] VESA – VIDEO ELECTRONICS STANDARDS ASSOCIATION. **VESA DPMS standard summary.** Disponível em: <http://www.vesa.org/dload/summary/sumdpms.htm>. Acesso em: 05 fev. 2002.

[W3C2000] W3C – WORLD WIDE WEB CONSORTIUM. **Extensible Markup Language (XML) 1.0 (second edition).** W3C Recommendation. 2000. Disponível em: <http://www.w3.org/TR/REC-xml>. Acesso em: 16 abr. 2002.

[W3C2002] W3C – WORLD WIDE WEB CONSORTIUM. **Document Object Model (DOM) level 3 core specification version 1.0.** W3C Working Draft. 2002. Disponível em: <http://www.w3.org/TR/DOM-Level-3-Core/>. Acesso em: 16 abr. 2002.

[WEB2001] WEBBER, Carrie A. et al. **Field Surveys of Office Equipment Operating Patterns.** Berkeley: Lawrence Berkeley National Laboratory, University of California, 2001. Disponível em: <http://eetd.lbl.gov/ea/>. Acesso em: 28 jan. 2002.