Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Abordagem Orientada a Serviços para o Gerenciamento de Energia em Redes Pervasivas

Paulo Rômulo Alves Barros

Campina Grande, Paraíba, Brasil

Novembro – 2011

Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Coordenação de Pós-Graduação em Ciência da Computação

Abordagem Orientada a Serviços para o Gerenciamento de Energia em Redes Pervasivas

Paulo Rômulo Alves Barros

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação

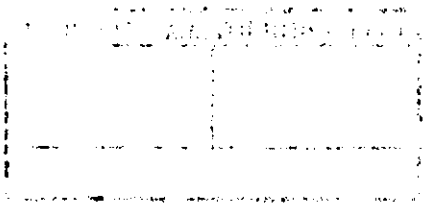
Linha de Pesquisa: Engenharia de Software

Orientadores

Hyggo Oliveira de Almeida

Angelo Perkusich

Campina Grande, Paraíba, Brasil



FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

B277a Barros, Paulo Rômulo Alves.

Abordagem orientada a serviços para o gerenciamento de energia em redes pervasivas / Paulo Rômulo Alves Barros. - Campina Grande, 2012.

82 f.: il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

Orientadores: Prof. Dr. Hyggo Oliveira de Almeida e Prof. Dr. Angelo Perkusich.

Referências.

1. Gerenciamento de Energia. 2. Universal Plug and Play. 3. Redes AD HOC. I. Título.

CDU 004.725.5(043)

ORDAGEM ORIENTADA A SERVIÇOS PARA O GERENCIAMENTO DE ENERGIA EM REDES PERVASIVAS"

PAULO RÔMULO ALVES BARROS

DISSERTAÇÃO APROVADA EM 28.11.2011

HYGGO OLIVEIRA DE ALMEIDA, D.Sc
Orientador(a)

ANGELO PERKUSICH, D.Sc
Orientador(a)

LEANDRO DIAS DA SILVA, D.Sc
Examinador(a)

MARCOS RICARDO ALCÂNTARA MORAIS, D.Sc
Examinador(a)

CAMPINA GRANDE - PB

Resumo

O crescimento do uso de dispositivos móveis no cotidiano das pessoas, aliado à difusão das tecnologias de comunicação sem fio, dá subsídios para se afirmar que cenários de um mundo onde a computação está embutida de forma "invisível" na vida cotidiana são perfeitamente possíveis. Conhecimento e serviços são providos no lugar e hora certos, de maneira transparente.

Essa autonomia das aplicações se tornou um dos pilares da Computação Pervasiva, onde esse tipo de "consciência" pôde ser introduzido a partir do momento em que se tornou possível capturar e tratar informações relativas ao ambiente e aos próprios dispositivos. Nesse contexto, um aspecto de importância significativa é o consumo de energia desses aparelhos. Redes pervasivas, em sua maioria, são compostas principalmente por dispositivos móveis que possuem uma bateria como sua fonte primária de energia, o que pode comprometer a disponibilidade dos serviços oferecidos.

Neste trabalho, tem-se como objetivo desenvolver um novo método de gerenciamento de energia em ambientes pervasivos. Propõe-se otimizar o consumo dos dispositivos considerando todo o conjunto de pares da rede, através de uma abordagem transversal e orientada a serviços. O método faz uso de políticas distribuídas de gerenciamento, buscando uma melhoria global do consumo.

Abstract

The growing use of mobile devices in people's everyday life, along with the difusion of wireless communication technologies, give subsidies to claim that scenarios of a world where computing is embedded on an invisible way in life are quite possible. Knowledge and services are provided in the right place and time, transparently.

This autonomy of applications has become one of the pillars of the Pervasive Computing field. Applications' awareness could be introduced in the moment when it is possible to capture and manipulate information about the environment and the devices themselves. In this context, one significant aspect is the power consumption of devices. Pervasive networks are mostly formed by mobile devices with a battery as primary source of energy. This limitation can compromise the availability of the provided services.

The goal of this work is to suggest a new method of power management in pervasive networks. The core of the idea relies in optimizing devices consumption considering the whole network and interactions, in a transversal and service-oriented way. Distributed policies for power management are used, in order to achieve a reduction of global consumption.

Agradecimentos

Primeiramente, gostaria de agradecer a todos os professores que contribuíram para minha formação, tanto na graduação quanto no mestrado. Em especial, deixo aqui registrado meu agradecimento a meus orientadores, profs. Hyggo Almeida e Angelo Perkusich. Sem as contribuições (e paciência) deles, este trabalho não seria possível.

Agradeço também a todos os amigos que fiz durante esse tempo, desde a minha chegada em Campina Grande. Eles também são responsáveis pelo sucesso desta empreitada. Cada gole de cerveja teve seu propósito. :-)

À minha mãe Albertina, que sempre me apoia nos momentos complicados e me mantém focado no que realmente importa. Ela é e sempre será minha referência como ser humano.

Aos membros da banca pelas justas críticas e contribuições.

Por fim, agradeço a todas as pessoas dentro e fora da Universidade Federal de Campina Grande que direta ou indiretamente tornaram tudo isto possível.

À CAPES, pelo apoio financeiro.

Muito obrigado.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

Em um mundo cada vez mais conectado, dispositivos como *smart phones*, *tablets* e *netbooks* têm sido massivamente adotados pelas pessoas. A diversidade de funções, públicos-alvo, capacidade de processamento e memória nesses aparelhos impulsiona a disseminação da computação móvel, um paradigma que avança de forma consistente e rápida, fazendo com que o conjunto de possíveis aplicações direcionadas a esse tipo de plataforma seja muito grande.

O crescimento do uso de dispositivos móveis no cotidiano das pessoas aliado à formalização e padronização de diversas tecnologias de comunicação sem fio como os padrões IEEE 802.11 (WiFi), IEEE 802.16 (WiMax) e conexão 3G dá subsídios para se afirmar que as ideias semeadas por Mark Weiser no seu clássico artigo de 1991 [34] estão começando a se tornar realidade. No seu trabalho, Weiser vislumbra cenários de um mundo onde a computação está embutida de forma "invisível" nos mais variados aspectos da vida cotidiana, "tecendo-se na fábrica da vida até se tornar indistinguível dela", segundo o próprio. Com o advento da **computação pervasiva**, situações como a de ser lembrado pela geladeira de que o leite acabou ou ser avisado pelo carro a respeito de promoções de pneus novos uma vez que os atuais estão gastos são perfeitamente possíveis.

Esse tipo de "consciência" pode ser introduzido nas aplicações pervasivas a partir do momento em que se é possível capturar e tratar informações relativas ao ambiente onde o dispositivo se encontra, como também dados a respeito das preferências do usuário com relação a diversos aspectos. Com isso, pode-se disponibilizar serviços e conhecimento no lugar e na hora certos. É a chamada *ciência de contexto* [18], onde *contexto* pode ser definido

como qualquer informação necessária a uma aplicação para que a mesma tenha condições de tomar decisões transparentes ao usuário. Tomando o exemplo do carro supracitado, o veículo poderia sugerir a loja de pneus que estivesse mais próxima e que praticasse os melhores preços. Nesse caso, as principais informações de contexto utilizadas seriam relativas à localização (*location-awareness* ou ciência de localização). Essa autonomia das aplicações se tornou um dos pilares da computação pervasiva, na qual, por se tratar de uma área de pesquisa relativamente nova, vários problemas novos e desafiadores surgem a cada momento.

Com relação às tecnologias que fazem da computação pervasiva uma realidade mais observável, um padrão aberto tem se mostrado uma solução interessante na implementação de ambientes e aplicações pervasivas. Trata-se do *Universal Plug and Play* (UPnP), um esforço conjunto de empresas como Nokia, Microsoft e Intel, entre outras, as quais definiram um padrão de conectividade com o mínimo de configuração. Através de uma pilha de protocolos já consolidados como o HTTP e o SOAP, é possível estabelecer uma rede *ad hoc* na qual os dispositivos podem disponibilizar vários tipos de serviços entre si, tornando possíveis desde cenários de automação simples como o controle da temperatura de um ambiente, até aplicações mais complexas como casas inteligentes que se adaptam ao hábitos e preferências de seus donos.

Nesse contexto, um aspecto de importância significativa diz respeito ao consumo de energia dos dispositivos. A computação pervasiva, por ser uma área oriunda de tópicos como computação móvel, redes *ad hoc* e sistemas distribuídos, herda todos os problemas relacionados à essa questão, principalmente pelas suas características de mobilidade. Redes pervasivas, em sua maioria, são compostas principalmente por dispositivos móveis que possuem uma bateria como sua fonte primária de energia. Em redes *ad hoc* e de sensores, por exemplo, uma das maiores preocupações é o tempo de vida da rede, já que estas podem estar instaladas em áreas onde a recarga não é viável ou mesmo possível. Por isso, diversas estratégias são adotadas pelos fabricantes para que os dispositivos consumam o mínimo possível de bateria tanto em uso quanto em períodos de inatividade [23] [36] [2].

Por sua vez, aplicações cientes do consumo de energia (*energy-awareness*) podem utilizar informações de contexto para "agirem" de uma forma ou de outra dependendo do estado da energia de um dado aparelho. Pode ser interessante, por exemplo, saber se um *netbook* está sendo alimentado por sua bateria ou pela rede elétrica em um dado momento. No pri-

meiro caso, a aplicação poderia desativar os efeitos gráficos e diminuir o brilho da tela, enquanto que no segundo caso essas configurações poderiam permanecer em um estado menos restritivo.

Isso pode ser útil para os casos em que os dispositivos conectados cooperam de forma a fornecer um determinado serviço. A ideia é que com o passar do tempo e com os avanços em áreas como computação utilitária, serviços oferecidos em redes pervasivas compartilhem uma quantidade cada vez maior de recursos de dispositivos na rede. Como exemplo, pode-se citar a integração de sensores e aparelhos de ar condicionado com a finalidade de se ajustar a temperatura de um determinado ambiente.

Porém, a maioria das abordagens de gerenciamento de consumo estão embutidas no *hardware* e/ou no sistema operacional do dispositivo [28] [3], sendo concebidas para os casos em que o mesmo é utilizado de maneira isolada. Quando em rede, a tendência é que esses aparelhos permaneçam ligados por todo o tempo em que o serviço deve estar disponível, inclusive nos períodos de inatividade. Com isso, dispositivos dependentes de bateria não otimizam o consumo da mesma, enquanto que os alimentados pela rede elétrica continuam a consumir energia, acarretando em um custo monetário que poderia ser evitado.

1.1 Problemática

Atualmente pode-se encontrar na literatura várias técnicas de gerenciamento de energia, variando de abordagens que atuam no nível do hardware [12] [13] a métodos implementados a nível de sistema operacional [14] [15]. Técnicas como *clock-gating* [5] adaptam os circuitos para que reduzam seu consumo. Já o ACPI [3], é um padrão implementado por sistemas operacionais (como o Linux) o qual visa um maior controle do consumo através de *software*.

Esses e muitos outros esforços visam diminuir o consumo de um dispositivo durante períodos de inatividade ou de baixa taxa de transmissão/recepção de dados. Em suma, a maioria das técnicas e padrões foram concebidos do ponto de vista de um dispositivo utilizado de maneira isolada, visando a otimização do consumo de uma forma independente do restante da rede pervasiva. No entanto, cenários onde os dispositivos estão conectados e executando serviços colaborativos não são considerados. Nesse contexto, faz-se necessária uma abordagem transversal de gerenciamento de energia desses dispositivos, com o intuito

de otimizar a execução dos serviços em relação à energia necessária a cada nó integrante da rede.

A arquitetura *Low Power* do padrão UPnP fornece uma solução parcial do problema, definindo estados de economia de energia que podem ser implementados pelos dispositivos. Um dispositivo é dito compatível com a especificação *Low Power* quando implementa dois ou mais estados de economia de energia. Isso, em conjunto com a figura do *proxy*, definida também pelo padrão, permite que os nós da rede que estiverem em algum estado de "dormência" continuem alcançáveis. Na chegada de uma requisição por determinado serviço oferecido por um dispositivo em estado de economia, o mesmo pode ser "acordado" pelo *proxy* a fim de atender àquela solicitação.

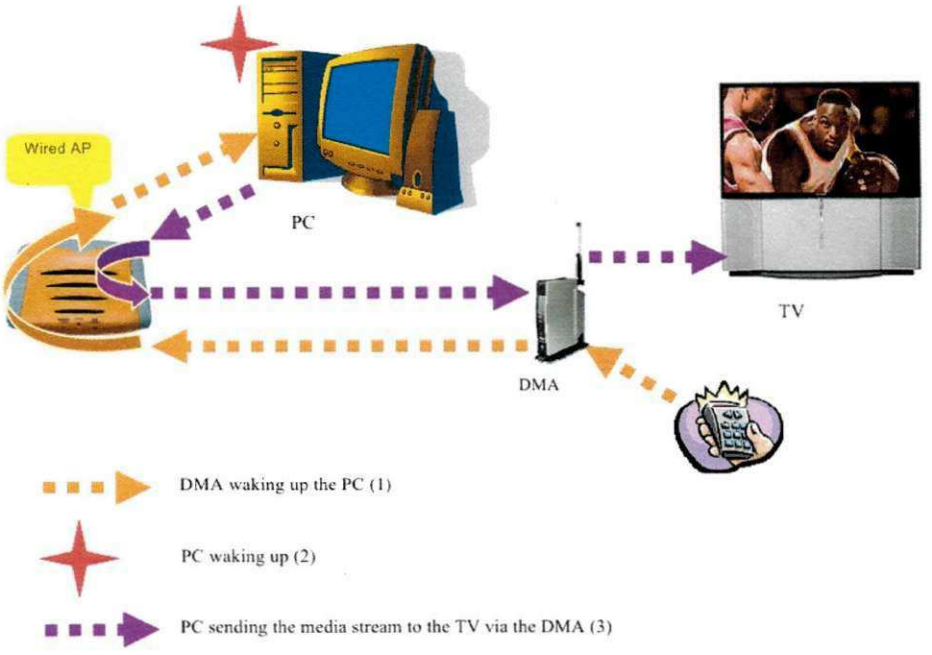


Figura 1.1: *Low Power Architecture*

A figura 1.1 ilustra um cenário de utilização da arquitetura. O usuário pretende exibir no televisor um arquivo de vídeo armazenado em seu computador pessoal, o qual se encontra em estado de economia. O dispositivo móvel requisita ao *proxy*, que neste caso consiste do ponto de acesso, informações sobre servidores em estado de espera e solicita que o computador em questão seja "acordado". O *proxy*, o qual sabe como despertar dispositivos, acorda o computador, que através do dispositivo DMA gera o fluxo de dados do vídeo e o transmite à

TV.

No entanto, a solução prevista na especificação *Low Power* não prevê, por exemplo, políticas de otimização na escolha do(s) provedor(es) de serviço(s) nem informações contextuais como preferências do usuário. Ainda considerando o cenário acima, como proceder no caso de dois servidores possuírem o mesmo arquivo? Como escolher, do ponto de vista de consumo, o "melhor" servidor a ser despertado? Se o dispositivo móvel estiver com pouca carga em sua bateria, seria viável sugerir que o vídeo fosse exibido em outro renderizador presente na rede, como um *laptop*, por exemplo?

Considere-se, por exemplo, uma organização que dispõe de um serviço de impressão comum a todos os funcionários. Há várias impressoras conectadas entre si e compartilhadas na rede através do padrão UPnP. Para evitar que todas as impressoras fiquem ligadas o tempo todo, apenas algumas localizadas em posições estratégicas permanecem ativas, enquanto as outras são mantidas em um estado de espera. A decisão de quem é ligado ou não cabe ao *proxy*, que poderia ser um ponto de acesso sem fio, dispositivo este que deve permanecer ligado todo o tempo. Em momentos de alta demanda, o *proxy* poderia ativar outra impressora a fim de diminuir a espera por impressões. Por exemplo, se um usuário envia um documento que demanda muito tempo de processamento da impressora (muitas páginas e/ou muitas figuras, por exemplo), o *proxy* poderia acordar a outra impressora mais próxima para que o próximo trabalho não aguarde por muito tempo para ser processado. Apesar da figura do *proxy* já estar definida na especificação *Low Power*, esse tipo de coordenação não é prevista pela arquitetura original. Através da inserção de certas ações em cada etapa da comunicação entre os dipositivos UPnP, é possível dotar o *proxy* e os próprios nós provedores de serviço de uma certa autonomia, fazendo com que cenários como o supracitado sejam possíveis de serem implementados.

1.2 Objetivo

Este trabalho tem como objetivo desenvolver um método para gerenciamento de energia em redes pervasivas baseadas no padrão UPnP. Propõe-se otimizar o consumo dos dispositivos considerando todo o conjunto de pares da rede que cooperam provendo um certo serviço. O método faz uso de políticas distribuídas de gerenciamento buscando uma melhoria global

do consumo. Para isso, uma abordagem transversal será incorporada ao que já está definido na arquitetura UPnP *Low Power*, fazendo uso de mecanismos já existentes e realizando modificações na especificação quando necessário.

Nesse contexto, propõe-se uma abordagem orientada a serviço a qual se baseia na figura do *proxy* como principal elemento coordenador no processo de ativação de serviços pervasivos, onde através de regras pré-deteminadas e/ou informações contextuais pode-se determinar qual a melhor configuração de um ponto de vista que o consumo seja o menor possível no momento.

No que diz respeito às etapas de implementação e validação, foi utilizado o *framework* BRisa [11], justificando a sua escolha pelo fato de o mesmo seguir o modelo de desenvolvimento em código aberto, além de prover meios para construção de dispositivos e serviços compatíveis com o padrão UPnP de uma maneira simplificada para o programador. Para atingir esse objetivo, fez-se necessário também a implementação da especificação *Low Power* utilizando o referido arcabouço.

1.3 Relevância

Em cenários pervasivos, os dispositivos em rede frequentemente atuam de forma colaborativa e distribuída. Pode-se verificar esse fato em cenários como o serviço de IPTV (*Internet Protocol Television*) [1], que pode envolver dispositivos como computares pessoais, pontos de acesso sem fio, *set-top boxes*, televisores, *tablets*, entre outros. No entanto, a grande maioria das abordagens de gerenciamento de energia disponíveis para estes tipos de dispositivos não levam em consideração a natureza colaborativa das aplicações, fazendo com que os dispositivos permaneçam ligados continuamente, inclusive em períodos de inatividade dos serviços. Com isso, gera-se um consumo desnecessário, assim como custos inerentes a este.

Um outro aspecto que vale ser destacado é o fato de que as informações relativas ao consumo de dispositivos são também informações de contexto. Com elas, é possível criar aplicações e infraestruturas "cientes de energia", podendo-se gerar, por exemplo, padrões de consumo em relação a como os usuários utilizam os serviços. Isso pode, no futuro, otimizar a interação destes usuários com os ambientes pervasivos, diminuindo assim custos relativos

[1] http://www.openiptvforum.org

ao consumo energético dos dispositivos.

Por fim, este trabalho está inserido no contexto do projeto Percomp do Laboratório de Sistemas Embarcados e Computação Pervasiva [2] da Universidade Federal de Campina Grande [3], contribuindo para o avanço do estado da arte na área.

1.4 Organização do Documento

Este documento segue organizado da seguinte forma:

- No Capítulo 2, é apresentada a fundamentação teórica. Nela, todos os conceitos necessários ao entendimento do trabalho são apresentados separadamente. Especificamente, são abordados em detalhes o padrão UPnP para conectividade de dispositivos sem fio, bem como sua especificação de economia de energia, a arquitetura UPnP *Low Power*. São apresentados também conceitos gerais em gerenciamento de energia
- No Capítulo 3, expõem-se técnicas de controle de consumo de energia já propostas, algumas em cenários pervasivos e outras em ambientes similarmente conectados, como redes de sensores sem fio e redes *ad hoc*, discutindo as principais diferenças entre elas e mostrando que limitações destas técnicas tenta-se tratar neste trabalho.
- No Capítulo 4, descreve-se a solução proposta para gerenciamento de energia em redes pervasivas UPnP.
- No Capítulo 5, são apresentados um estudo de caso e simulações.
- No Capítulo 6, considerações finais e trabalhos futuros.

[2] http://www.embeddedlab.org

[3] http://www.ufcg.edu.br

Capítulo 2

Fundamentação Teórica

O presente capítulo tem como objetivo apresentar os conceitos nos quais este trabalho se baseia, provendo o conhecimento que se julga necessário ao entendimento da solução aqui proposta.

2.1 *Universal Plug and Play* (UPnP)

Proposto pela Microsoft no ano de 1999, o *Universal Plug and Play* (UPnP) é um modelo de arquitetura de rede que provê conectividade *ad-hoc* entre dispositivos de forma distribuída, transparente, independente de *driver* ou plataforma e sem a necessidade de qualquer tipo de configuração. Uma de suas principais características é a facilidade em se disponibilizar serviços na rede, possuindo um protocolo de descoberta de dispositivos (e, consequentemente, de serviços) bem definido e de fácil implementação. Além disso, é um padrão aberto que se baseia em tecnologias e especificações já consolidadas, tais como a arquitetura TCP/IP e os protocolos HTTP e SOAP.

Outras grandes empresas como Nokia, Intel e HP também colaboraram com a concepção do padrão, fazendo surgir dessa parceria o *UPnP Forum* [1]. Este consórcio de empresas, que hoje conta com mais de 900 membros, é responsável por garantir a interoperabilidade do UPnP, atualizando e definindo novas especificações, as quais garantem a aplicabilidade do modelo em diversos cenários, seja em ambiente de rede doméstico ou corporativo. Para isso, os participantes do fórum são divididos em comitês, sendo alguns listados abaixo:

[1] http://upnp.org

- **Áudio/Vídeo**: armazenamento e disponibilização de conteúdo multimídia entre dispositivos;
- **Automação e Segurança Residencial**: padrões para automação residencial, através de dispositivos como câmeras, sensores, lâmpadas, entre outros;
- **Sincronização de Conteúdo**: gerenciamento de conteúdo armazenado em diversas fontes;
- **Gerenciamento de Dispositivos**: manutenção e configuração de dispositivos UPnP;
- ***e-Health***: gerenciamento de informações de saúde e bem-estar;
- ***Internet Gateway***: operação de dispositivos de conectividade, tais como roteadores, modems e pontos de acesso;
- **Telefonia**: visa garantir maior interação entre telefones e outros dispositivos UPnP;
- ***Low Power***: define um padrão onde dispositivos expõem informações relacionadas a gerência de energia, como por exemplo estados de economia (ativo, *stand-by*, etc.).

As próximas subseções apresentarão a arquitetura UPnP, mostrando seus componentes e as interações entre eles. Serão apresentados também alguns cenários de aplicação do padrão.

2.1.1 Rede UPnP

Tecnicamente, uma rede UPnP consiste de uma rede *ad-hoc*, na qual os nós são capazes de anunciar sua chegada e sua saída, além de poderem buscar por serviços de interesse disponibilizados pelos dispositivos que estão ativos.

Neste contexto, existem dois tipos de dispositivos UPnP: aqueles que provêem os serviços e aqueles que utilizam os serviços disponibilizados pelos primeiros. Àqueles dá-se o nome de **dispositivo controlável**, ou apenas dispositivo UPnP, e os últimos denominam-se **pontos de controle**. Dispositivos controláveis se comportam como servidores, enquanto que pontos de controle agem como clientes. Um dispositivo UPnP pode ser controlável e se comportar como ponto de controle ao mesmo tempo, facilitando a implementação de serviços distribuídos e colaborativos. Por exemplo, um telefone compatível com o padrão UPnP

pode ser controlável ao disponibilizar um serviço de GPS e ao mesmo tempo ser ponto de controle de um serviço de automação residencial.

Figura 2.1: *Rede UPnP*

É importante salientar que dispositivos são entidades lógicas aninháveis, formando uma estrutura similar a uma árvore. Assim, um dispositivo UPnP pode conter outros, e cada um deles pode disponibilizar determinados serviços em um contexto próprio. **Dispositivo raiz** é aquele que não está associado a nenhum "pai". Toda essa estrutura de dispositivos aninhados, além de informações sobre serviços, fabricante, entre outras, são disponibilizadas em um arquivo XML, o qual é utilizado pelos pontos de controle para interagir com dispositivos controláveis.

2.1.2 Disponibilização de Serviços UPnP

Um serviço pode ser definido como um conjunto de ações que podem ser requisitadas por um ponto de controle a um determinado dispositivo. Fazendo uma analogia com o paradigma de programação orientado a objetos, pode-se comparar uma ação a um método, que como tal pode ser invocado e receber parâmetros, retornando ou não um valor após sua execução. Serviços também possuem variáveis que armazenam seu estado, as quais podem ter eventos associados a mudanças em seus valores, notificando entidades interessadas da atualização desses valores. Por exemplo, uma impressora que implementa o padrão UPnP que

disponibiliza um serviço de impressão compartilhada e possui uma ação *enqueue_job*, a qual enfileira uma impressão na fila e uma variável de estado *job_queue_status*, que armazena o estado atual da fila e sinaliza mudanças em seu estado através de eventos. A descrição dos serviços, ações e variáveis de estado de um determinado dispositivo fica armazenada em um arquivo XML.

2.1.3 Protocolos de Comunicação

A pilha de protocolos adotada pelo padrão UPnP faz uso de tecnologias já difundidas e amplamente utilizadas em diversos contextos, a exemplo do HTTP [2], do TCP [3] e do SOAP [9], os quais configuram-se como padrões no desenvolvimento de aplicações para Internet ou em outras situações se faz necessária a comunicação em rede. Com isso, a interoperabilidade entre dispositivos e serviços é praticamente garantida e a curva de aprendizado para a construção destes é aumentada. A figura 2.2 resume a pilha de protocolos do padrão UPnP.

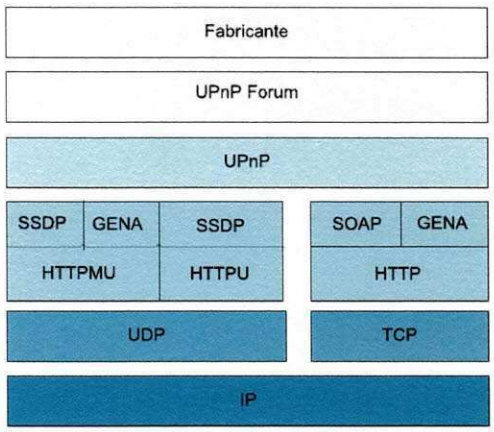


Figura 2.2: *Protocolos de Comunicação do padrão UPnP*

2.1.4 Etapas de Conectividade

O processo de estabelecimento de uma rede UPnP é subdividido em seis etapas, a saber, **endereçamento**, **descoberta**, **descrição**, **controle**, **evento** e **apresentação**. A seguir, apresenta-se o detalhamento de cada uma delas.

[2]http://www.ietf.org/rfc/rfc2616.txt

[3]http://www.ietf.org/rfc/rfc793.txt

Etapa 0 - Endereçamento

A primeira etapa a ser desempenhada na formação de uma rede UPnP é a de endereçamento. Dado que na camada mais inferior da pilha de protocolos tem-se o padrão IP (*Internet Protocol*) [4] como base, nesta etapa são atribuídos aos dispositivos os seus respectivos endereços IP, tal qual é feito usualmente em redes locais do tipo LAN (*Local Area Network*). Há duas possibilidades para a distribuição de endereços: em redes autogerenciadas, através de um servidor de DHCP (*Dynamic Host Configuration Protocol*) [5], ou fazendo uso de um mecanismo de AutoIP [12]. Para o primeiro caso, assume-se que existe um servidor executando algum tipo de servidor DHCP. No momento que um novo dispositivo UPnP, o qual, por padrão, também executa um cliente de DHCP, junta-se à rede, este servidor fica responsável por atribuir um endereço IP válido para aquela rede. No segundo caso, o próprio dispositivo é encarregado de escolher um endereço válido. Se durante o processo de endereçamento o dispositivo obtiver também um nome de domínio de um servidor de DNS (*Domain Name System*), este deve ser usado nas operações de rede subsequentes. Caso contrário, o próprio endereço IP é utilizado.

Etapa 1 - Descoberta

Uma vez atribuído um endereço IP, o próximo passo é a etapa de descoberta. O protocolo de descoberta usado no padrão UPnP é um dos únicos próprios do padrão e é denominado SSDP (*Simple Service Discovery Protocol*. Quando um dispositivo se junta à rede UPnP, o SSDP permite que o mesmo anuncie sua presença e seus serviços para os pontos de controle presentes na rede (os quais também passam pela etapa de endereçamento normalmente). Analogamente, através do SSDP é possível que um ponto de controle recém chegado procure por dispositivos e/ou serviços de interesse. Em ambos os casos, mensagens *multicast* SSDP são trocadas entre os dispositivos, nas quais são embutidas informações a respeito dos serviços que se deseja anunciar ou critérios para busca por dispositivos e serviços específicos.

No caso de dispositivos recém chegados na rede UPnP, o protocolo obriga que os mesmos anunciem sua chegada através de uma mensagem *multicast* específica do tipo *ssdp:alive*. Esta mensagem de "boas vindas" contém informações básicas a respeito do dispositivo, tais

[4] http://www.ietf.org/rfc/rfc791.txt

[5] http://www.ietf.org/rfc/rfc2131.txt

como nome e uma URL indicando um endereço de uma descrição mais detalhada. Um exemplo deste tipo de mensagem pode ser visto na Listagem 2.1. É importante salientar que a mensagem *ssdp:alive* também é usada como meio de notificar que o dispositivo ainda não deixou a rede, utilizando o modelo de *heartbeat*, onde cada mensagem possui um tempo de validade, e passado esse tempo outra notificação deve ser enviada. Caso contrário, quando ocorre o *timeout*, assume-se que o dispositivo desligou-se da rede em condições anormais.

Listagem 2.1: Mensagem de Anúncio de Presença

```
1 NOTIFY * HTTP/1.1
2 HOST: 239.255.255.255.250:1900
3 CACHE-CONTROL: Tempo máximo de validade
4 LOCATION: URL para descrição do dispositivo
5 NTS: ssdp:alive
6 //campos omitidos
```

Quando um ponto de controle faz uma busca por dispositivos/serviços, é enviada uma mensagem *multicast* do tipo *M-Search*, na qual podem ser inseridos alguns critérios de busca, como por exemplo o tipo de serviço que se procura, como um serviço de impressora. Quando nenhum critério de busca é especificado na mensagem, assume-se que se quer descobrir todos os dispositivos ativos na rede UPnP. Na Listagem 2.2, tem-se um exemplo de mensagem *M-SEARCH*.

Listagem 2.2: Mensagem de Descoberta

```
1 M-SEARCH * HTTP/1.1
2 HOST: 239.255.255.250:1900
3 MAN: ssdp:discover
4 MX: 10
5 ST: ssdp:all
6 //campos omitidos
```

Todos os dispositivos UPnP que se alinham com os critérios especificados na mensagem de busca *M-Search* (ou todos os dipositivos da rede, de fato) devem responder à descoberta através de mensagens *unicast* UDP enviadas diretamente ao ponto de controle remetente. Essas mensagens são enviadas pelo dispositivo raiz, o qual envia separadamente descrições de seus dispositivos embutidos e os respectivos serviços. Um exemplo de mensagem de resposta à descoberta pode ser visto na Listagem 2.3.

Listagem 2.3: Mensagem Resposta de Descoberta

```
1 HTTP/1.1 200 OK
2 CACHE-CONTROL: max-age=1800
3 EXT:
4 LOCATION: http://10.0.0.138:80/IGD.xml
5 SERVER: SpeedTouch 510 4.0.0.9.0 UPnP/1.0 (DG233B00011961)
6 ST: urn:schemas-upnp-org:service:WANPPPConnection:1
7 USN: uuid:UPnP-SpeedTouch510::urn:schemas-upnp-org:service:
    WANPPPConnection:1
```

Não necessariamente durante a etapa de descoberta um dispositivo UPnP deve notificar à rede que está deixando de fazer parte da mesma. Para isso, utiliza-se uma mensagem *multicast* do tipo *ssdp:byebye*, a qual pode ser vista na Listagem 2.4. Diferente de quando se deixa de enviar mensagens de *heartbeat*, esta é a forma de o dispositivo comunicar que está deixando a rede em condições normais.

Listagem 2.4: Mensagem de Saída de Dispositivo

```
1 NOTIFY * HTTP/1.1
2 HOST:239.255.255.250:1900
3 NT: urn:schemas-microsoft-com:nhed:presence:1
4 NTS: ssdp:byebye
5 LOCATION: *
6 USN: uuid:00000000-0000-0000-0200-00125A8A0960::urn:schemas-microsoft-com:
    nhed:presence:1
```

Etapa 2 - Descrição

Após descobrir um dispositivo UPnP, o ponto de controle ainda possui muito pouco conhecimento acerca do mesmo. Não é possível ainda interagir de fato, nem requisitar nenhum tipo de serviço. Para isso, o ponto de controle deve buscar pelo arquivo de descrição do dispositivo raiz a fim de poder recuperar informações detalhadas sobre ele, bem como suas funcionalidades. O referido arquivo de descrição, formatado em XML, está contido na URL disponibilizada anteriormente na mensagem de descoberta. Nele, encontram-se dados específicos do dispositivo como fabricante, modelo, entre outras. O arquivo de descrição também inclui a lista de todos os dispositivos embutidos naquele dispositivo raiz. Na Listagem 2.5

pode-se ver um exemplo de arquivo de descrição de dispositivo.

Listagem 2.5: Descrição de Dispositivo

```
1  <?xml version="1.0"?>
2  <root xmlns="urn:schemas-upnp-org:device-1-0">
3    <specVersion>
4      <major>1</major>
5      <minor>0</minor>
6    </specVersion>
7    <URLBase>base URL for all relative URLs</URLBase>
8    <device>
9      <deviceType> urn:schemas-upnp-org:device:CDPlayer:1</deviceType>
10     <friendlyName>short user-friendly title </friendlyName>
11     <manufacturer>manufacturer name</manufacturer>
12     <manufacturerURL>URL to manufacturer site </manufacturerURL>
13     <modelDescription>long user-friendly title </modelDescription>
14     <modelName>model name</modelName>
15     <modelNumber>model number</modelNumber>
16     <modelURL>URL to model site </modelURL>
17     <serialNumber>manufacturer's serial number</serialNumber>
18     <UDN>uuid:UUID</UDN>
19     <UPC>Universal Product Code</UPC>
20     <iconList>
21       <icon>
22         <mimetype>image/format</mimetype>
23         <width>horizontal pixels</width>
24         <height>vertical pixels</height>
25         <depth>color depth</depth>
26         <url>URL to icon</url>
27       </icon>
28       XML to declare other icons, if any, go here
29     </iconList>
30     <serviceList>
31       <service>
32         <serviceType>urn:schemas-upnp-org:service:SwitchPower:1</
               serviceType>
33         <serviceId>urn:upnp-org:serviceId:SwitchPower</serviceId>
34         <SCPDURL>URL to service description</SCPDURL>
```

```
35          <controlURL>URL for control</controlURL>
36          <eventSubURL>URL for eventing</eventSubURL>
37        </service>
38        <service>
39          <serviceType>urn:schemas-upnp-org:service:ChangeDisc:1</
                serviceType>
40          <serviceId>urn:upnp-org:serviceId:ChangeDisc</serviceId>
41          <SCPDURL>URL to service description</SCPDURL>
42          <controlURL>URL for control</controlURL>
43          <eventSubURL>URL for eventing</eventSubURL>
44        </service>
45        <service>
46          <serviceType>urn:schemas-upnp-org:service:PlayCD:1</serviceType>
47          <serviceId>urn:upnp-org:serviceId:PlayCD</serviceId>
48          <SCPDURL>URL to service description</SCPDURL>
49          <controlURL>URL for control</controlURL>
50          <eventSubURL>URL for eventing</eventSubURL>
51        </service>
52        <service>
53          <serviceType>urn:schemas-upnp-org:service:Audio:1</serviceType>
54          <serviceId>urn:upnp-org:serviceId:Audio</serviceId>
55          <SCPDURL>URL to service description</SCPDURL>
56          <controlURL>URL for control</controlURL>
57          <eventSubURL>URL for eventing</eventSubURL>
58        </service>
59        Declarations for other services added by UPnP vendor (if any) go
              here
60      </serviceList>
61      <presentationURL>URL for presentation</presentationURL>
62    </device>
63  </root>
```

Além do supracitado, o arquivo de descrição de dispositivos UPnP também indica URLs para os arquivos de descrição dos serviços contidos em cada dispositivo (raiz e embutidos). Estes também são arquivos do tipo XML que indicam, para cada serviço, a lista de ações e seus respectivos parâmetros, e a lista das variáveis de estado, as quais armazenam o estado de execução de um serviço. Cada variável de estado é descrita em termos do seu tipo de dado,

intervalo aceitável baseado naquele tipo e eventos associados. Eventos UPnP serão abordados em subseção posterior. A Listagem 2.6 mostra um exemplo de arquivo de descrição de um serviço UPnP.

Listagem 2.6: Descrição de Serviço

```
1  <?xml version="1.0"?>
2  <scpd xmlns="urn:schemas-upnp-org:service-1-0">
3    <specVersion> <!-- UPnP version 1.0 -->
4      <major>1</major>
5      <minor>0</minor>
6    </specVersion>
7    <actionList>
8      <action>
9        <name>AddDisc</name>
10     </action>
11     <action>
12       <name>NextDisc</name>
13     </action>
14     <action>
15       <name>PrevDisc</name>
16     </action>
17     <action>
18       <name>RandomDisc</name>
19     </action>
20     <action>
21       <name>HasTrayDisc</name>
22       <argumentList>
23         <argument>
24           <name>HasDisc</name>
25           <relatedStateVariable>TrayHasDisc</relatedStateVariable>
26           <direction>out</direction>
27         </argument>
28       </argumentList>
29     </action>
30     <action>
31       <name>OpenDoor</name>
32     </action>
```

```
33      <action>
34        <name>CloseDoor</name>
35      </action>
36      <action>
37        <name>ToggleDoor</name>
38      </action>
39      <action>
40        <name>IsDoorOpen</name>
41        <argumentList>
42          <argument>
43            <name>IsOpen</name>
44            <relatedStateVariable>DoorIsOpen</relatedStateVariable>
45            <direction>out</direction>
46          </argument>
47        </argumentList>
48      </action>
49      <!-- Declarations for other actions implemented by an -->
50      <!--   UPnP vendor (if any) go here. -->
51    </actionList>
52    <serviceStateTable>
53      <stateVariable sendEvents="yes">
54        <name>DoorIsOpen</name> <!-- whether disc tray is open -->
55        <dataType>boolean</dataType> <!-- 1 if open -->
56          <defaultValue>0</defaultValue>
57      </stateVariable>
58      <stateVariable sendEvents="yes">
59        <name>TrayHasDisc</name> <!-- is there a disc in the tray -->
60        <dataType>boolean</dataType> <!-- 1 if a disc -->
61      </stateVariable>
62      <!-- Declarations for other state variables implemented by an -->
63      <!--   UPnP vendor (if any) go here. -->
64    </serviceStateTable>
65  </scpd>
```

Etapa 3 - Controle

Uma vez tendo conhecimento das descrições dos dispositivos/serviços UPnP desejados, o ponto de controle pode, a partir das URLs de controle indicadas nos arquivos de descrição, invocar as ações disponibilizadas por cada serviço. Isso é feito através de chamadas padronizadas pelo protocolo SOAP [9], usado na implementação de *Web Services* [10]. Uma invocação a uma ação pode ser comparada a uma chamada remota de método, onde parâmetros de entrada são passados na chamada e parâmetros de saída são retornados ao final da operação, se completada com sucesso. Caso contrário, mensagens de erro são retornadas. Mensagens de controle são encapsuladas em pacotes do tipo HTTP usando o método POST. Na Listagem 2.7, é mostrada estrutura de uma mensagem de controle enviada de um ponto de controle UPnP para um dispositivo.

Listagem 2.7: Mensagem de Saída de Dispositivo

```
1  HOST: servidor:porta
2  CONTENT-LENGTH: bytes do corpo
3  CONTENT-TYPE: text/xml; charset="utf-8"
4  <!-- identificador do serviço e da ação invocada -->
5  SOAP-ACTION: urn:schemas:upnp-org:service:serviceType:v#actionName
6  <s:Envelopexmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
7  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
8  <s:Body>
9    <!-- Informações de parâmetros do serviço executado -->
10   <u:actionName xmlns:u="urn:schemasupnporg:service:serviceType:v">
11   <argumentName>valor</argumentName >
12   </u:actionName >
13 </s:Body>
14 </s:Envelope>
```

Etapa 4 - Notificações de Eventos

Muitas vezes, serviços UPnP necessitam armazenar o seu estado. Para isso, são utilizadas o que se chama de variáveis de estado. Por sua vez, invocações de serviços UPnP podem alterar os valores dessas variáveis, modificando o estado do serviço propriamente dito. Tomando o exemplo da impressora UPnP, seu serviço de impressão tem o estado alterado de

"inativo" para "imprimindo" no momento em que é submetido um trabalho a ser impresso. Similarmente, quanto mais trabalhos são submetidos para a impressora, a sua fila de impressões pendentes aumenta. Isto também pode ser representado por uma variável de estado tamanho_da_fila, por exemplo. A fim de informar os pontos de controle acerca das mudanças ocorridas nas variáveis de estado, notificações de evento são enviadas aos mesmos para que tomem conhecimento e atualizem possíveis variáveis locais dependentes desses eventos. Por exemplo, um ponto de controle com interface gráfica pode ter o interesse em atualizar o ícone de impressora quando esta estiver inativa ou imprimindo.

As notificações são realizadas através do protocolo GENA [19], e os dados são formatados em XML. Pontos de controle que estiverem interessados em mensagens de eventos devem registrar-se através de mensagens HTTP do tipo SUBSCRIBE. Neste momento, o dispositivo envia notificações *unicast* de evento ao ponto de controle contendo os nomes e os valores atualizados de todas as variáveis de estado. Para manter a inscrição ativa, pontos de controle devem renová-la antes que expire. Este mecanismo permite que caso um ponto de controle seja desconectado da rede sem solicitar explicitamente o cancelamento do registro através de mensagens UNSUBSCRIBE, a rede UPnP permaneça consistente.

Etapa 5 - Apresentação

Opcionalmente, o último passo previsto no estabelecimento de uma rede UPnP consiste do que se chama apresentação. Se um dispositivo UPnP possui uma página de apresentação, os pontos de controle interessados podem, caso possuam essa funcionalidade, carregar essa URL em um navegador. Pode ser possível também efetuar operações de controle de forma mais direta, simplificando o processo de invocação de ações, as quais seriam feitas através de algum tipo de interface gráfica *web*. Tudo isso depende das implementações específicas tanto do lado do dispositivo UPnP (servidor) quanto do lado do ponto de controle (cliente). Por exemplo, uma lâmpada com grau de luminosidade configurável pode possuir uma página de apresentação que poderia ser acessada por um *smartphone* com o intuito de controle da sua intensidade em um determinado local.

2.2 UPnP *Low Power Architecture* (LPA)

O padrão UPnP, assim como diversas arquiteturas existentes, prevê que os dispositivos que fazem parte da rede estejam sempre ligados. Em cenários de redes pervasivas, nos quais vários dispositivos são móveis e consequentemente alimentados por baterias, o desempenho na disponibilização de serviços pode ser prejudicado pelo fato de não haver energia suficiente para manter o serviço por muito tempo.

Com o objetivo de reduzir o impacto dessa limitação, surgiu a UPnP *Low Power Architecture* (LPA), uma dentre muitas outras especificações complementares ao padrão elaboradas pelo UPnP Forum. Esta arquitetura consiste em permitir que dispositivos UPnP possam economizar energia sem perder a conectividade com o restante da rede quando necessário. Assim, mantém-se a característica principal da rede UPnP que é a de os seus dispositivos poderem descobrir e serem descobertos. A ideia é que dispositivos implementem diferentes estados de conservação de energia e anunciem na rede a respeito das transições entre esses estados. Além disso, deve ser possível descobrir dispositivos que por ventura estejam "adormecidos" e invocar serviços destes quando necessário sem que a rede perca a sua consistência.

2.2.1 Dispositivos Compatíveis e Não Compatíveis com a LPA

A definição de uma nova especificação complementar ao padrão acarreta em mudanças no protocolo, principalmente no que diz respeito às mensagens. Cada nova especificação pode adicionar à pilha de protocolos mensagens específicas, bem como novos campos aos arquivos de descrição de dispositivos e/ou serviços. O padrão UPnP exige que dispositivos que implementam novas especificações sejam compatíveis com aqueles que não o fazem.

No contexto de *Low Power*, podem haver dois tipos de dispositivos: aqueles que implementam a especificação e aqueles que não implementam. Aqueles que são compatíveis com a LPA conseguem enviar e receber mensagens específicas relativas aos mecanismos de gerenciamento de energia (e.g. mensagens de transição de um estado de conservação para outro). Dispositivos compatíveis com a LPA entendem mensagens enviadas por aqueles que não implementam (mensagens de busca M-SEARCH, por exemplo) a especificação, porém o contrário pode não se verificar se a mensagem for específica da LPA.

2.2.2 Estados de Conservação de Energia

Tipicamente, dispositivos como computadores pessoais, celulares, TVs, entre outros, possuem um ou mais estados de economia de energia (*stand-by*). Diversos fatores podem engatilhar esses modos de conservação, como por exemplo relógios internos a cada dispositivo ou ativações por controle remoto. A especificação *Low Power* do padrão UPnP prevê a existência de cinco tipos de estados de economia, a saber: ***Active***, ***Transparent Sleep***, ***Deep Sleep Online***, ***Deep Sleep Offline*** e ***Disconnected***. Uma descrição de cada um deles é dada a seguir.

Estado *Active*

Dispositivos no estado *Active* são aqueles que estão em pleno funcionamento. Eles são visíveis aos outros dispositivos UPnP e, logo, podem ser descobertos por buscas M-SEARCH. Dispositivos nesse estado enviam periodicamente mensagens do tipo NOTIFY para renovar sua presença na rede. São as chamadas mensagens de *heartbeat* do protocolo SSDP, especificamente mensagens **ssdp:alive**. A partir desse estado um dispositivo pode entrar em qualquer um dos outros quatro estados, seja através de relógios internos, mensagens enviadas por outros dispositivos ou ações de usuário.

Estado *Transparent Sleep*

Esse é o primeiro estado onde realmente há alguma economia de energia. Seria equivalente ao *stand-by* de uma TV ou um aparelho de microondas, por exemplo. Um dispositivo no estado *Transparent Sleep* só é visível para outros dispositivos que implementam a especificação *Low Power*. Ou seja, mensagens de *heartbeat* só serão recebidas pelos dispositivos compatíveis com a LPA. No entanto, dispositivos em *Transparent Sleep* respondem às buscas M-SEARCH normalmente (quando normalmente são "acordados" e voltam ao estado *Active*), bem como a outras mensagens de controle enviadas por clientes. A partir desse estado, um dispositivo UPnP pode ir para qualquer um dos outros estados.

Estado *Deep Sleep Online*

Um dispositivo UPnP no estado *Deep Sleep Online* permanece conectado à rede, porém não responde à nenhuma mensagem de controle, exceto mensagens específicas de *wake up* (despertar). Isto é, uma vez neste estado, o dispositivo apenas receberá mensagens que o façam transitar para o estado *Transparent Sleep* ou *Active*. Estar em *Deep Sleep Online* significa também que o dispositivo possui um endereço IP, no entanto não envia as mensagens de *heartbeat* para o restante da rede, ficando assim, invisível aos outros nós. Transições deste estado para todos os outros são permitidas.

Estado *Deep Sleep Offline*

O estado *Deep Sleep Offline* faz com que o dispositivo não possua mais um endereço IP na rede, sendo assim, invisível a todos os outros. Logo, mensagens de NOTIFY não são enviadas, nem respostas à buscas M-SEARCH. As únicas formas de se "despertar" um dispositivo nesse estado seriam relógios internos ou mecanismos *out of band*, como por exemplo o *Wake on Lan* (WoL) [6]. Não são permitidas transições do estado *Deep Sleep Offline* para o estado *Deep Sleep Online*, pois não seria possível ao dispositivo anunciar a sua presença na rede UPnP, nem responder a possíveis buscas. Todas as outras transições são permitidas.

Estado *Disconnected*

Como o próprio nome sugere, dispositivos no estado *Disconnected* estão totalmente desativados e, consequentemente, fora da rede UPnP. Transições para este estado podem ocorrer pelo simples fato de o usuário pressionar o botão de desligar ou através de relógios internos. Neste caso, dispositivos podem ser "despertados" através de métodos definidos pelo fabricante. Não são permitidas transições do estado *Disconnected* para os estados *Deep Sleep Online* e *Deep Sleep Offline*, pois não seria possível ao dispositivo anunciar a sua presença na rede UPnP, nem responder à possíveis buscas. Transições para os outros estados são permitidas. A figura 2.3 mostra um diagrama de transição que resume o modelo de estados da especificação *Low Power* do padrão UPnP.

[6] http://gsd.di.uminho.pt/jpo/software/wakeonlan/mini-howto/wol-mini-howto-2.html

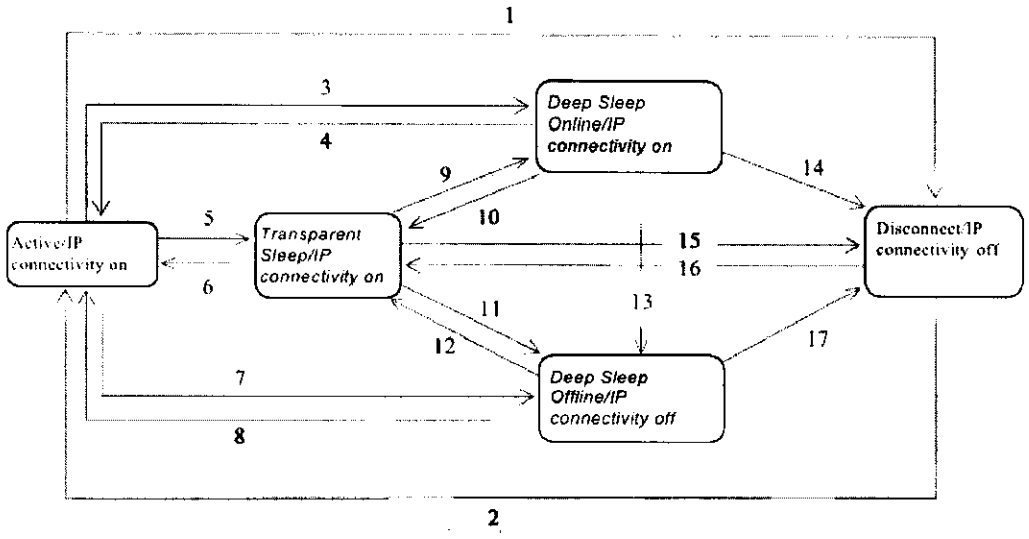


Figura 2.3: Máquina de Estados UPnP *Low Power*

2.2.3 Entidades do Modelo UPnP *Low Power*

A arquitetura UPnP *Low Power* define três entidades básicas que interagem com o intuito de permitir que dispositivos UPnP implementem estados de conservação de energia, mantenham os outros dispositivos cientes das transições entre estados e que, principalmente, os mesmos continuem acessíveis de alguma forma.

Pontos de Controle *Low Power*

O **ponto de controle *Low Power*** pode monitorar as transições de estado dos outros nós da rede que implementam a especificação LPA, além de também estar ciente da entrada e saída de dispositivos da rede. Essas informações podem ser persistidas para uso futuro, por exemplo. Além disso, um ponto de controle *Low Power* tem a habilidade de diferenciar dispositivos UPnP compatíveis com a LPA daqueles que não são compatíveis, podendo fazer buscas específicas e enviar, por exemplo, mensagens de *wake up* para dispositivos nos estados de *Transparent Sleep* e *Deep Sleep Online*.

Dispositivos *Low Power*

Um dispositivo UPnP que implementa a LPA informa à rede a respeito das suas mudanças de estado de conservação de energia, bem como informa a respeito de sua entrada ou saída

da rede. Um dispositivo UPnP *Low Power* pode ser classificado de acordo com as seguintes categorias:

- *Sleep*-autônomo: Um dispositivo que pode entrar em um estado de *stand by* através de mecanismos internos definidos pelo fabricante (e.g. *timers* internos).
- *Sleep*-controlado: Dispositivo que entra em um estado de conservação de energia ao receber uma mensagem de controle externa de um ponto de controle ou outro dispositivo UPnP *Low Power*.
- *Wake up*-autônomo: Aquele dispositivo que "desperta" através de mecanismos definidos internamente (e.g. relógio).
- *Wake up*-controlado: Um dispositivo que necessita de uma interação externa, como uma mensagem WoL, para poder sair de um estado de *stand by*.

Proxy Low Power

Denomina-se *proxy* um dispositivo UPnP que pode "responder" por dispositivos que estejam "adormecidos", ou seja, em algum estado de conservação, como o *Deep Sleep Online*. O papel do *proxy* é garantir que os nós em *stand by* que por ventura estejam invisíveis aos demais dispositivos continuem presentes na rede. Apesar de sua presença na rede ser opcional, a figura do *proxy* é de extrema importância para que dispositivos nos estados *Deep Sleep Online* e *Deep Sleep Offline* continuem acessíveis, já que ele trata de responder a possíveis buscas que forem realizadas por esses dispositivos.

2.2.4 Gerenciamento de Dispositivos Utilizando o *Proxy Low Power*

O principal objetivo do *proxy* é permitir que dispositivos UPnP possam assumir qualquer um dos estados de conservação de energia, com exceção do estado *Disconnected*, e permanecerem como parte da rede UPnP. Um *proxy* é capaz de fazer buscas por dispositivos que sejam capazes de assumir estados de *stand by*, ou seja, aqueles que implementam a especificação *Low Power*. Isso é feito através de uma mensagem do tipo M-SEARCH, que é enviada no momento em que o *proxy* se junta à rede. Essas informações são então armazenadas em um

banco de dados interno, e usadas no momento que um destes dispositivos anuncia sua entrada em um estado de conservação de energia. Vale salientar que outros *proxies* podem fazer parte de uma mesma rede, porém as informações armazenadas em cada um não são sincronizadas, podendo haver inconsistências. Um outro aspecto a respeito do *proxy* é que sendo este também um dispositivo UPnP, o mesmo não entra em um estado de conservação. Por isso, recomenda-se que *proxies* sejam aparelhos que raramente ou nunca sejam desligados, como roteadores ou pontos de acesso.

Tecnicamente, um *proxy* é um dispositivo UPnP que disponibiliza um serviço chamado *LowPowerProxy*. Este serviço possui duas ações principais, a saber: *SearchSleepingDevices* e *WakeupDevice*. A primeira é usada para recuperar a lista de dispositivos UPnP que anunciaram sua ida para um estado de conservação de energia, tendo sido assim armazenados no banco de dados interno do *proxy*. Esta ação pode ser usada principalmente por pontos de controle interessados em saber quais dispositivos estão efetivamente em *stand by*. A ação *WakeupDevice* é usada pelo *proxy* para "despertar" um dispositivo em *stand by* quando alguma busca M-SEARCH efetuada por um ponto de controle possui critérios compatíveis com algum dos dispositivos armazenados na lista. Dispositivos no estado de *Deep Sleep Offline* devem indicar no arquivo de descrição de dipositivo qual método deve ser utilizado (e.g. *Wake on Lan*). Além de ser utilizada pelo próprio *proxy*, a ação *WakeupDevice* também pode ser invocada diretamente por um ponto de controle que deseja "despertar" um determinado dispositivo.

2.2.5 Cenários

As próximas subseções mostram cenários onde a especificação LPA poderia ser aplicada a fim de economizar energia em ambientes de rede domésticas.

Cenário 1 - Sem a Presença do *Proxy Low Power*

Neste cenário, a rede UPnP é composta por um computador pessoal (PC), um roteador de rede sem fio, um receptor de mídia digital (DMR) e um televisor com capacidade de recepção de dados através de conexões de rede sem fio. A figura 2.4 ilustra esta configuração.

1. O PC anuncia que está entrando em um estado de conservação de energia. Esta infor-

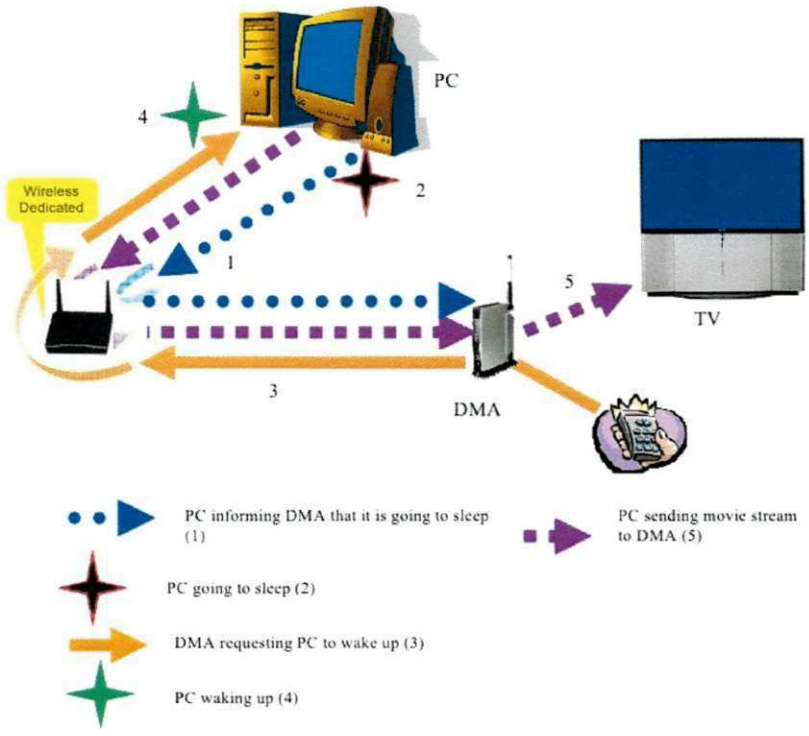


Figura 2.4: *Rede UPnP Low Power sem Proxy*

mação é armazenada pelo DMR, que é também um ponto de controle UPnP.

2. O PC completa sua transição para o estado de *stand by*, ficando assim inacessível para os outros dispositivos.

3. O usuário deseja assistir a um filme em sua TV. O filme está armazenado no PC, o qual está em um modo de conservação de energia. O DMR tem conhecimento que o arquivo está armazenado no PC (configuração feita anteriormente). O DMR então "acorda" o PC baseado nas informações de gerenciamento de energia enviadas pelo PC no momento do estabelecimento da rede UPnP. O DMR então mostra na TV uma mensagem ao usuário indicando que o PC está sendo acessado.

4. Após ser "acordado", o PC então se torna operacional e envia a lista de arquivos para o DMR, que envia para a TV. O usuário então procura pelo filme desejado.

5. Selecionado o arquivo, o PC envia o stream do filme ao DMR, que por sua vez se encarrega de exibi-lo no televisor.

Obviamente, as informações relativas ao procedimento de *wake up* do PC devem ser passadas ao DMR antes de ser necessário acordá-lo. Isso pode ser feito através da invocação de uma ação chamada *GetPowerManagementInfo*, disponibilizada pelos dispositivos UPnP que implementam a arquitetura *Low Power*.

Cenário 2 - Com a Presença do *Proxy Low Power*

No segundo cenário, a rede UPnP consiste de um computador pessoal (PC), um ponto de acesso de rede sem fio (AP), um receptor de mídia digital (DMR) e um dispositivo móvel com capacidade de reprodução de vídeos, como um *smartphone*. Assume-se que o dispositivo que faz o papel de *proxy* é o DMR. Este cenário pode ser visto na figura 2.5.

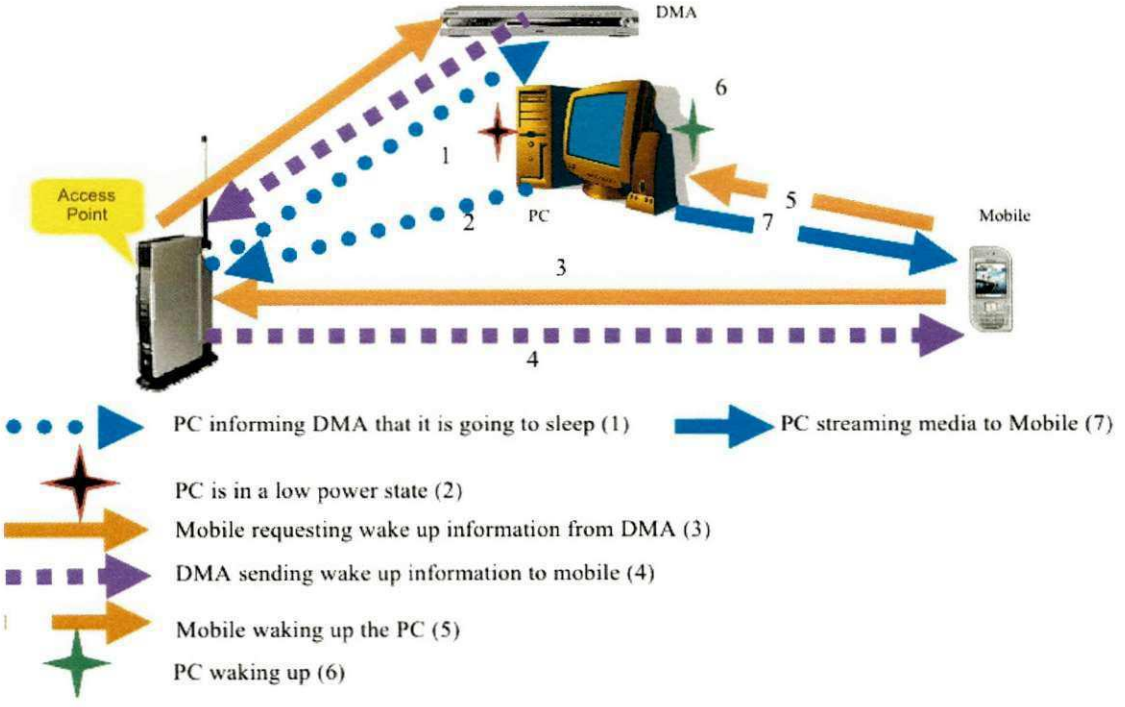


Figura 2.5: *Rede UPnP Low Power com Proxy*

1. O PC anuncia que está entrando em um estado de conservação de energia. Esta informação é armazenada pelo DMR, que, como *proxy*, tem o dever de armazenar o estado atual de todos os dispositivos UPnP compatíveis com a LPA que estão presentes na rede.

2. O PC completa sua transição para o estado de *stand by*, ficando assim inacessível para os outros dispositivos.

3. O usuário deseja ver um determinado vídeo em seu *smartphone*. Este vídeo está armazenado no PC, que se encontra em *stand by*, e o *smartphone* é ciente dessa informação. No entanto, o *smartphone* desconhece as informações relativas à forma de se "despertar" o PC, pois juntou-se a rede após a entrada do PC em modo de energia baixa. Por isso, o *smartphone* requisita ao *proxy* (DMR) a lista de todos os dispositivos "adormecidos" naquele momento, através da ação *SearchSleepingDevices*.

4. O DMR envia a lista ao *smartphone*, o qual mostra ao usuário a lista de servidores de mídia disponíveis, incluindo aqueles que por ventura estejam em modos de conservação de energia.

5. O usuário então escolhe o PC no qual está armazenado o vídeo desejado. É exibida uma mensagem indicando que o dispositivo em questão está em *stand by* e que será "acordado".

6. Após ser "acordado", o PC então se torna operacional e o usuário pode então procurar pelo vídeo desejado através do *smartphone*.

7. Selecionado o arquivo, o PC envia o stream do vídeo ao *smartphone*, que por sua vez se encarrega de exibi-lo na tela para o usuário.

Analogamente, as informações relativas ao procedimento de *wake up* do PC devem ser passadas ao *proxy* antes de ser necessário acordá-lo. Geralmente, o *proxy* invoca a ação *GetPowerManagementInfo* no momento em que detecta que um dispositivo UPnP que compatível com a LPA se junta à rede.

2.3 Gerenciamento de Energia em Redes sem Fio

A diferença mais notável entre redes cabeadas e redes sem fio é justamente o tipo de meio por onde a comunicação é feita. A ideia de uma rede totalmente sem fio é permitir a mobilidade total dentro de um determinado raio de alcance, sem manter os nós presos a uma

determinada posição. Sem o devido gerenciamento do consumo desses dispositivos móveis, a energia necessária para mantê-los conectados à rede por longos períodos de tempo se esgota rapidamente. Neste contexto, usuários são vistos procurando mais por fontes de energia do que pela rede em si, ficando assim presos novamente a uma posição fixa.

Muitas abordagens de gerenciamento de energia foram desenvolvidas nos últimos anos com o intuito de resolver o problema exposto acima. Soluções foram propostas em todas as camadas da pilha de rede tradicional, cada uma com sua própria contribuição na economia de energia. As próximas subseções mostram algumas dessas soluções, bem como padrões que emergiram dessas abordagens.

2.3.1 Técnicas de Gerenciamento de Energia

Esta seção apresenta algumas técnicas usadas em padrões de rede sem fio conhecidos, como aqueles da série IEEE 802 [7]. Serão discutidas técnicas que existem desde a camada de aplicação até a camada de rede, levando em consideração a pilha tradicional de protocolos definida no modelo OSI [29]. Abordagens relativas às camadas de enlace e física não serão expostas por estarem em um nível muito baixo do ponto de vista do que é proposto neste trabalho.

Camada de Aplicação

Na camada de aplicação, muitas abordagens podem ser usadas para diminuir o consumo de dispositivos com comunicação sem fio. Uma conhecida técnica chamada **particionamento de carga** [15] permite que uma aplicação tenha todas as suas operações as quais exigem uma carga maior de energia processadas remotamente. O dispositivo simplesmente envia uma requisição e espera pelo resultado. Outras abordagens utilizam eventos de bateria para alertar aplicações e fazer com as mesmas trabalhem em modos mais econômicos. Programas com interfaces gráficas mais elaboradas podem suspender efeitos visuais mais sofisticados, economizando, assim, a bateria do aparelho.

Algumas aplicações mais comuns merecem abordagens que reduzam o custo total de executá-las. Dois destes casos incluem operações em bancos de dados e processamento de

[7] http://www.ieee802.org/

vídeo. No caso de programas que fazem uso de bancos de dados, as técnicas mais usadas são aquelas que visam reduzir o consumo durante a recuperação dos dados, indexação e operações de consulta. A ideia é que se reduza a quantidade de transmissões/recepções necessárias a essas operações. Em aplicações que processam vídeos, as principais técnicas tentam comprimir os arquivos com o intuito de reduzir a quantidade de bits transmitidos através do canal. Como a compressão em si pode consumir uma quantidade significativa de energia, outras técnicas exploram tornar a qualidade do vídeo um pouco mais baixa para tentar compensar a economia gerada pela compressão. Algumas abordagens relacionadas à camada de aplicação podem ser vistas em [15] e [22].

Camada de Transporte

A maioria das técnicas de gerência de energia na camada de aplicação visa reduzir o número de retransmissões devido à perda de pacotes ocasionada por conexões sem fio de baixa qualidade. Em redes cabeadas, a taxa de perda de pacotes é usada para medir o quão congestionada se encontra a rede, fazendo com que os algoritmos da camada de transporte usem suas estratégias de contenção. No entanto, em redes sem fio picos de perdas podem ocorrer esporadicamente, não sendo interpretadas imediatamente como indicativos de congestionamento. Algumas abordagens como [33] e [38] foram desenvolvidas com esse conhecimento levado em consideração, tentando garantir a entrega de dados fim-a-fim com alto rendimento e baixo consumo.

Camada de Rede

Os métodos provenientes da camada de rede têm como objetivo prover um roteamento em redes *multi-hop* mais eficiente de um ponto de vista de consumo de energia. [21], [16]. As abordagens nesse caso podem ser divididas basicamente em orientadas a ***backbone***, orientadas a **controle de topologia** ou técnicas híbridas das duas anteriores.

Em métodos baseados em *backbones*, alguns nós são escolhidos para permanecerem ativos o tempo todo (nós do *backbone*), enquanto outros são configurados para "dormirem" periodicamente. Os nós do *backbone* são então usados para que sejam estabelecidos caminhos entre todas as fontes e todos os destinos na rede. Para que isso seja possível, no entanto, todos os nós da rede devem estar a um *hop* de pelo menos um nó do *backbone*, incluindo

os próprios. A economia de energia aí é alcançada por permitir que os nós que não são do *backbone* entrem em modo de economia de energia periodicamente, bem como mudando a configuração do próprio *backbone* com alguma frequência.

Por sua vez, os algoritmos baseados em controle de topologia conseguem economizar energia de uma forma diferente. Sua meta é reduzir o poder de transmissão de todos os nós em uma rede de uma forma que todos permaneçam conectados, mas operando com o mínimo possível de transmissões. Em redes homogêneas, isto significa que o poder de transmissão de todos os nós é ajustado para que seja possível alcançar o vizinho mais próximo que esteja a distância de um *hop*. Para o caso de redes heterogêneas, os poderes de transmissão dos nós são ajustados de acordo com as necessidades de cada rede. O esquema mostrado na figura 2.6 mostra uma das classificações citadas na literatura para abordagens baseadas em topologia.

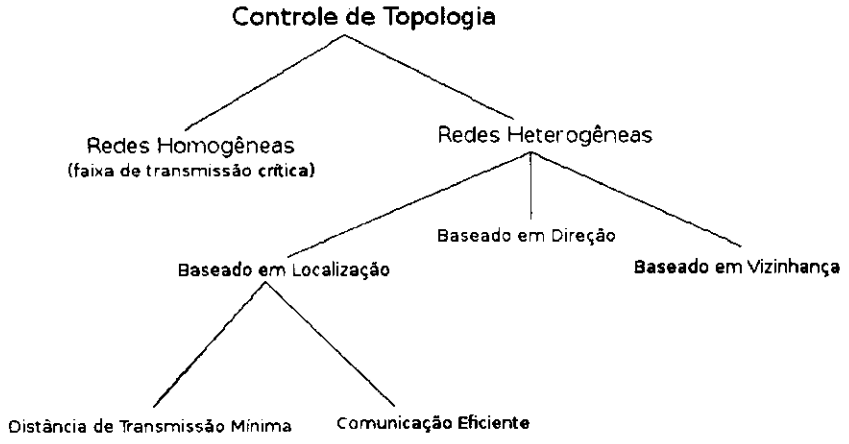


Figura 2.6: *Abordagens Baseadas em Topologia*

Por fim, existem também as abordagens híbridas que combinam elementos tanto dos algoritmos orientados a *backbones* como aqueles que são orientados ao controle de topologia. Alguns exemplos podem ser vistos em [7] e [8].

2.3.2 Padrões da Série IEEE 802

Os padrões definidos na série 802 do IEEE para redes sem fio são um esforço de muito tempo em evoluir as tecnologias *wireless* e permitir uma variedade de dispositivos com grande interoperabilidade. A figura 2.7 reúne alguns dos principais padrões da série, dos quais

serão destacados nesta seção aqueles que abrangem as redes locais sem fio (WLANs) e redes pessoais sem fio (WPANs), por terem uma relação mais próxima com o contexto do trabalho aqui apresentado. Um subconjunto importante das WPANs que vale ser destacado é o das redes de sensores sem fio (WSNs), área na qual há um esforço de pesquisa significativo no que diz respeito a métodos de economia de energia.

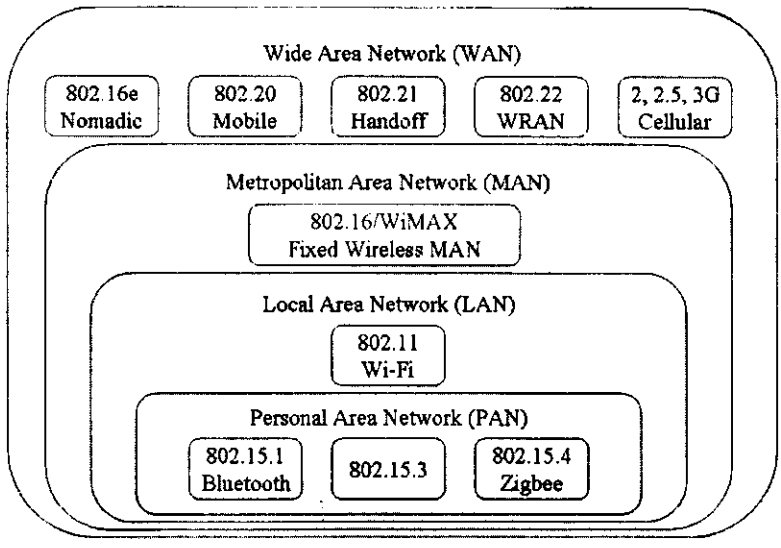


Figura 2.7: *Padrões IEEE 802*

Todos os padrões acima prevêem em suas definições algumas das abordagens discutidas nas seções anteriores. Além disso, evoluções nessas técnicas são implementadas em versões mais atualizadas das respectivas especificações visando gerenciar ainda melhor a energia consumida pelos dispositivos conectados através destes protocolos. As próximas subseções abordam alguns dos principais esforços na gerência de consumo de energia em WLANs, WPANs (*Bluetooth*) e WSNs.

IEEE 802.11 - Redes Locais Sem Fio (WLANs)

O padrão IEEE 802.11 especifica como ocorre a comunicação entre nós de uma rede local sem fio (WLAN). Uma parte deste padrão define o que se chama de ***Power Save Mode*** (PSM), que pode estar disponível em dispositivos conectados em uma infraestrutura baseada no 802.11. O PSM é baseado em uma política de escalonamento de períodos de *stand by*, onde os nós (estações) são capazes de alternar entre um modo **ativo** e um modo ***sleep***. Assim que um nó se junta à WLAN, ele deve notificar ao ponto de acesso ao qual se associou que

ele está com o PSM habilitado ou não. O ponto de acesso então se sincroniza com esta nova estação, permitindo que o mesmo execute seu esquema de escalonamento. Quando pacotes são endereçados a dispositivos que implementam o PSM, o ponto de acesso os armazena em um *buffer* até que a estação volte ao estado ativo novamente, notificando-a sobre esses pacotes recebidos no período de *sleeping*.

Apesar de ser uma funcionalidade presente na especificação 802.11, o PSM não é muito utilizado. Afirma-se que a perda em rendimento na rede não compensa o que se ganha em economia de energia [27].

IEEE 802.15.1 - Redes Pessoais Sem Fio (WPANs)

Especificamente, o padrão IEEE 802.15.1 [8], também conhecido como *Bluetooth*, provê seus próprios métodos de gerenciamento de energia. Nós em uma rede *Bluetooth* são organizados em subgrupos (*piconets*), onde um nó é escolhido como *master* e os outros como *slaves*. Estas subredes podem conter até sete nós ativos ao mesmo tempo, podendo possuir 256 nós potenciais (249 inativos). Todos estes nós operam com escalonamento de tempos de *sleep* e *caching* de pacotes também é feito de maneira similar ao que é feito em WLANs, com o nó *master* fazendo o papel do ponto de acesso neste caso.

O padrão *Bluetooth* também define oito modos de operação, onde três deles são dedicados a manter um baixo consumo de energia. São os modos *sniff*, *hold* e *park*. Quando em *sniff*, um nó simplesmente aumenta seu tempo de *sleeping*, escutando menos no contexto da *piconet* da qual faz parte. Quando muda para o estado *hold*, o dispositivo suspende todas as suas capacidades de comunicação, mas permanece ocupando uma das sete "vagas" na *piconet*. Já um dispositivo em modo *park* também desabilita toda a sua comunicação, com a diferença que deixa de ocupar o *status* de ativo no contexto da *piconet*.

IEEE 802.15.4 - Redes de Sensores Sem Fio

O padrão de redes de sensores sem fio IEEE 802.15.4 provê um baixo fluxo de dados e comunicação com baixo consumo, o que é ideal para aplicações que fazem uso de sensores sem fio. Apesar de também se basear no conceito de períodos intercalados de *sleeping*, aqui o cenário se difere dos anteriores no que diz respeito à frequência com que os nós "despertam",

[8] http://www.bluetooth.org/spec

bem como a quantidade de dados que é transmitida. Nesse contexto, o protocolo *Zigbee* [1] se destaca por ser a solução *low power* mais utilizada em ambientes de redes de sensores sem fio, possuindo uma grande aceitação da indústria e, consequentemente, uma grande variedade de dispositivos compatíveis.

Capítulo 3

Trabalhos Relacionados

O presente capítulo apresenta algumas abordagens existentes relacionadas ao gerenciamento de energia de dispositivos conectados. São apresentadas tanto técnicas gerais para dispositivos conectados quanto métodos para domínios específicos mas similares, como redes de sensores sem fio e redes *ad hoc*. Como não são muitos os registros de abordagens puramente orientadas a redes pervasivas, o estudo buscou suporte em áreas correlatas, como as citadas anteriormente.

3.1 Gerenciamento de Energia em Nível de Rede para Dispositivos de Redes Domésticas

Em dois trabalhos, tendo um deles sido publicado em 2007 [13] e outro em 2009 [17], Han et. al propuseram uma abordagem em nível de rede para o gerenciamento de energia de redes pervasivas, mais especificamente redes domésticas. Os autores afirmam que, devido ao surgimento constante de diversos tipos de dispositivos, as redes locais domésticas se tornam maiores e mais complexas. Hoje, além dos computadores pessoais, é fácil encontrar em algumas residências dispositivos como pontos de acesso sem fio, televisores digitais, *set-top boxes*, impressoras, *smartphones*, *tablets*, entre outros. Não são raros, também, os cenários onde esses dispositivos atuam colaborativamente com o objetivo de prover algum serviço. Um exemplo disso pode ser o *streaming* de mídias armazenadas em um computador e exibidas em um televisor, que pode envolver vários aparelhos, como o computador onde o arquivo

está contido, o ponto de acesso, o *set-top box* e o próprio televisor. Uma outra aplicação poderia receber o fluxo de dados de mídia diretamente em um telefone ou *tablet*. Tem-se que dispositivos podem assumir diversas funções em contextos diferentes. Espera-se que em um futuro próximo os serviços em redes domésticas compartilhem cada vez mais das diferentes funcionalidades dos dispositivos, onde cada um seria um provedor de recursos funcionais baseado no serviço provido em determinado momento.

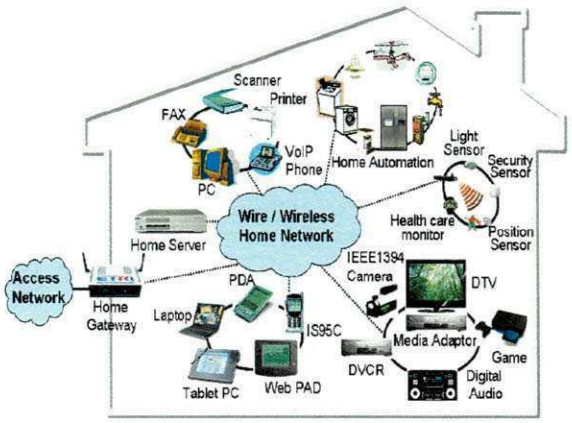


Figura 3.1: *Rede Doméstica Pervasiva*

Cenários como os citados acima sofrem do problema de que os dispositivos participantes (ou provedores de recursos) devem permanecer ligados o tempo todo. Ou seja, se um determinado serviço necessita de GPS para funcionar e só o *smartphone* alimentado por bateria pode fornecer essa funcionalidade, o serviço inteiro fica limitado pela disponibilidade deste aparelho. No entanto, se o GPS está ligado, o próprio *smartphone* está ligado, e o *Bluetooth* pode estar habilitado, também consumindo energia da bateria. Diversas outras funções do aparelho podem estar habilitadas naquele momento, não sendo necessárias para o serviço que só necessita do GPS. Nesse contexto, os autores propõem que para cada "colaborador" em um determinado serviço, apenas as unidades funcionais (GPS, Bluetooth, Wi-Fi) realmente necessárias estejam habilitadas, fazendo com que o restante das funcionalidades do dispositivo estejam desligadas ou em estado de espera. Essas unidades funcionais são denominadas elementos de controle (PCEs). A arquitetura proposta pode ser vista na figura 3.2.

A solução consiste de dois tipos de dispositivos: os servidores de controle e os clientes de controle. Um exemplo de servidor seria um *home gateway* ou um computador comum.

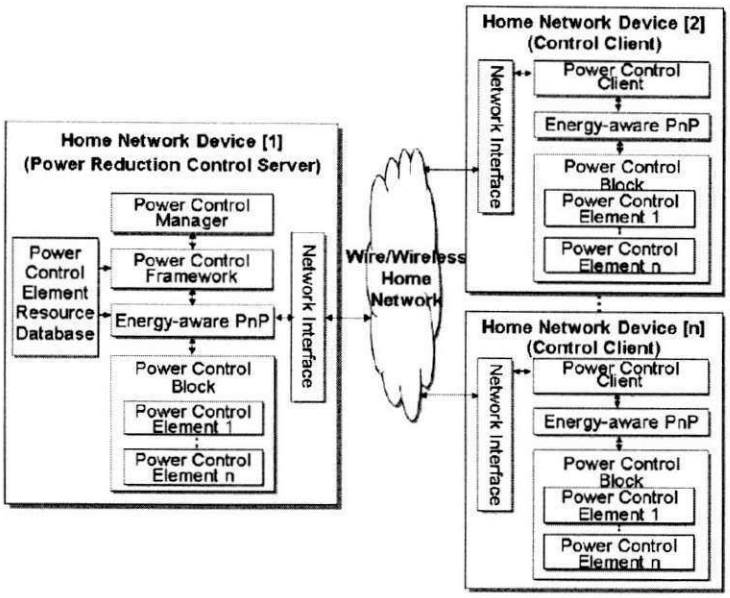


Figura 3.2: *Arquitetura*

O servidor de controle é o responsável por gerenciar a utilização de todos os clientes de controle, requisitando a lista de PCEs de cada um e efetuando a ativação de serviços baseada na disponibilidade de elementos de controle nos clientes. Todas as mensagens de controle são trocadas pelos dispositivos utilizando-se um protocolo chamado *Energy-aware Plug and Play* (EPnP). Não é referenciada nenhuma especificação formal desse protocolo, bem como não são providas maiores informações a seu respeito. A figura 3.3 mostra um fluxo de execução entre cliente e servidor através do protocolo EPnP.

Apesar de ser uma solução perfeitamente viável em teoria, a principal assunção presente no trabalho, que é o particionamento de recursos pontuais, resultando nos PCEs, não é factível nos dispositivos atuais. Apesar de hoje ser possível o controle independente de algumas funções em dispositivos como *smartphones* (e.g. GPS e *Bluetooth*), o mesmo não pode ser verificado em outros aparelhos como impressoras e televisores, por exemplo. Esta limitação se torna explícita quando se constata que todos os resultados são obtidos a partir de simulações, não sendo possível implementar nenhum cenário de teste com dispositivos reais.

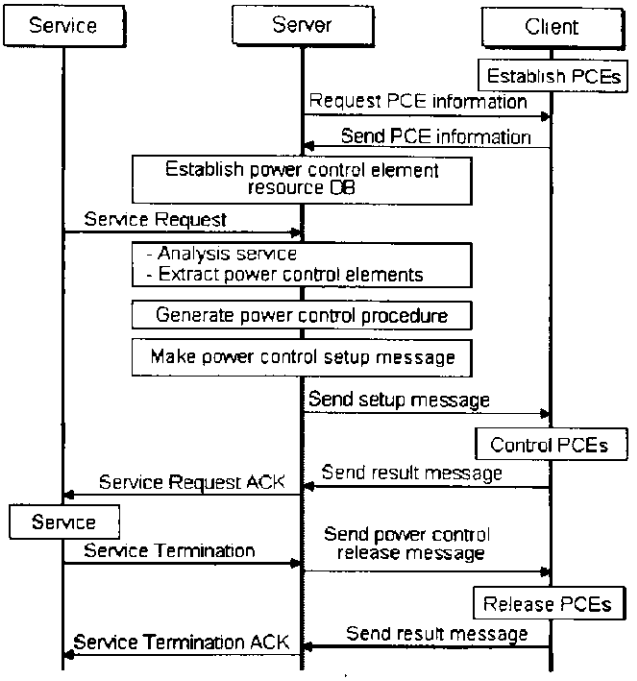


Figura 3.3: *Energy-aware Plug and Play*

3.2 Arquitetura Pervasiva para Gerenciamento de Energia em Redes Domésticas

Trompos et. al, nos anos de 2008 [30] e 2009 [31] publicaram dois trabalhos apresentando uma arquitetura orientada a redes domésticas para monitoramento, estimação e gerenciamento de consumo energético, utilizando uma abordagem genérica. Resultados alcançados anteriormente pelo consórcio AIM [6] e por pesquisas realizadas em países europeus mostram o grande potencial de crescimento na utilização e consequentemente no custo energético gerado por redes domésticas, as quais tendem a interligar uma quantidade cada vez maior de aparelhos. Além das tecnologias mais conhecidas como ZigBee e WiFi, é possível verificar que os fabricantes dos chamados *white goods* (como geladeiras e máquinas de lavar) têm dotado esse tipo de dispositivo com a capacidade de se comunicar em rede, através, por exemplo do protocolo KNX (ISO/IEC 14543). Nesse contexto, os autores propõem um modelo genérico de comunicação entre os integrantes da rede e um *gateway* gerenciador, através de um protocolo próprio baseado em IP. Para isso, é introduzida uma entidade lógica chamada EMD (sigla em inglês para dispositivo monitor de energia), o qual "traduz"

as mensagens de gerenciamento enviadas pelo *gateway* utilizando o protocolo genérico para o tipo de mensagem específica que é entendida pelo aparelho em si. A figura 3.4 mostra a arquitetura definida pelos autores.

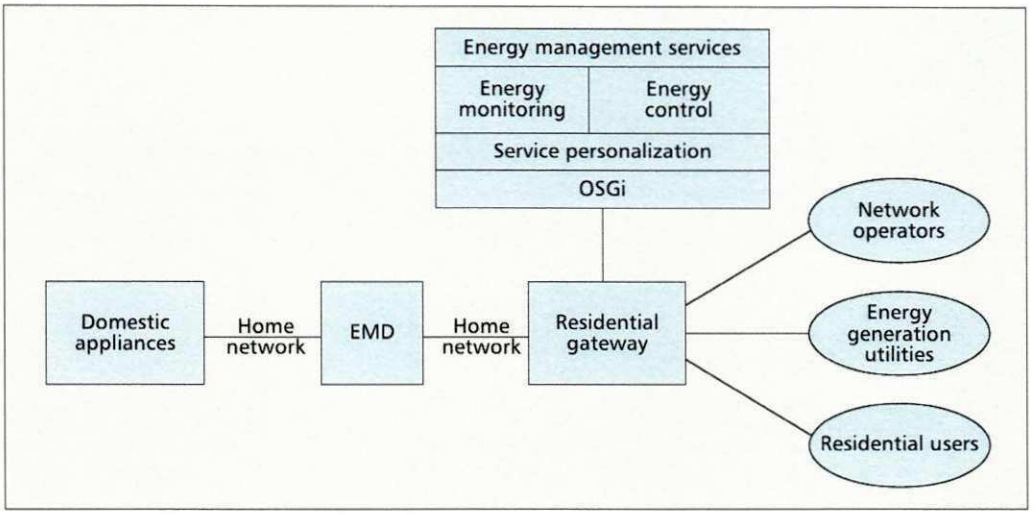


Figura 3.4: *Arquitetura da Solução utilizando EMDs*

Além do papel de intermediador entre o módulo gerenciador e as entidades gerenciadas, o EMD também é responsável pelas atividades de monitoramento dos dispositivos, mantendo um registro de seu estado (ativo ou em espera), o qual é armazenado em um banco de dados acessado pelo *gateway*.

A implementação da comunicação entre as aplicações, o *gateway* e os dispositivos gerenciados é totalmente baseada no protocolo IP, visando garantir interoperabilidade em diferentes cenários. *Web services* são usados para efetuar a comunicação entre o usuário (aplicações) e o *gateway*, através de interfaces *web* acessíveis nos navegadores de Internet mais utilizados, seja em ambiente *desktop* ou móvel. A interação do *gateway* com os elementos gerenciados é realizada através de dois protocolos próprios, também baseados em IP. O protocolo **máquina-para-máquina** (M2M) tem o papel de prover uma API única conhecida pelo *gateway* e os elementos gerenciados, unificando a comunicação e facilitando a implementação das funções de gerenciamento e monitoramento por parte dos dispositivos conectados. Já o **protocolo universal de gerenciamento de energia** (UEM) é uma forma unificada de se implementar os comandos de controle e as mensagens trocadas entre o *gateway* e o EMD. A figura 3.5 resume as pilhas de protocolos utilizadas nos contextos de elemento doméstico

gerenciado (*domestic appliance*), *gateway* residencial e aplicações de usuário. O formato de mensagem trocada entre o EMD e o *gateway* é mostrado na figura 3.6.

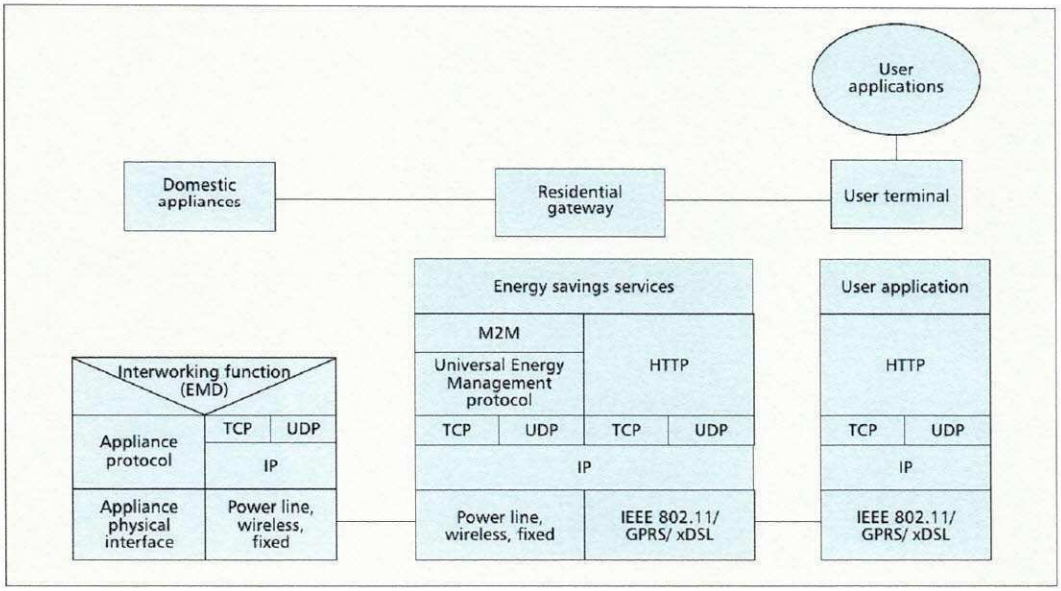


Figura 3.5: *Pilha de Protocolos*

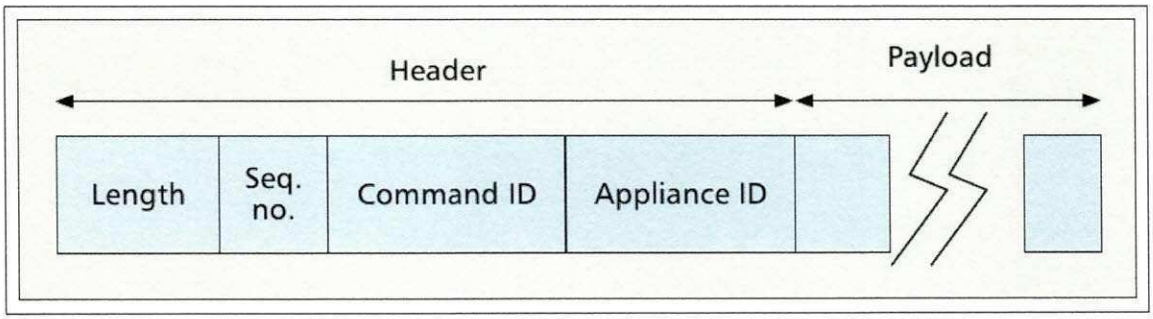


Figura 3.6: *Formato da Mensagem*

A arquitetura proposta, apesar de consistente e muito bem definida peca em alguns aspectos. A grande dependência do protocolo KNX é um ponto crítico, pois mesmo sendo um padrão definido pela ISO, este não possui um nível de adoção significativo entre os fabricantes de *white goods*, compromentendo assim a eficiência de toda a técnica. Uma outra limitação observada no trabalho é o fato de não haver estudo de caso verificando a validade do método apresentado, nem indicações de que o modelo foi implementado de fato. Mesmo que não fossem utilizados dispositivos reais, simulações poderiam mostrar os benefícios da

técnica, ou pelo menos indicar o seu comportamento em um ambiente real. Isso também se deve à generalidade do modelo, o qual deixa em aberto potenciais tecnologias que poderiam ser utilizadas em uma implementação.

3.3 SANDMAN - Um *Middleware* Ciente do Consumo de Energia para Computação Pervasiva

No fim de 2008, Schiele e Handte apresentaram o SANDMAN [25], um *middleware* para redes pervasivas *ad-hoc* (MANETs) com ciência de consumo de energia e foco em **clusterização**. Baseado em um trabalho anterior, denominado BASE [4], o SANDMAN dá suporte a comunicação *peer-to-peer* com foco na redução do consumo de energia dos dispositivos principalmente quando estão inativos à espera de requisições de clientes.

O SANDMAN foi projetado baseado em dois objetivos principais: reduzir a energia consumida na transferência de dados selecionando o protocolo mais eficiente para isso em um determinado momento; e fazer com que dispositivos inativos permaneçam em um estado de *stand by* a fim de aumentar a vida útil de suas baterias. Segundo os autores, o primeiro deles pode ser atingido a partir do trabalho desenvolvido no BASE, através de sua habilidade de selecionar *plug-ins* dinamicamente. Isto é, adicionando as devidas informações na descrição do *plug-in* de cada protocolo da pilha, a estratégia de seleção baseada na eficiência do consumo pode ser facilmente implementada.

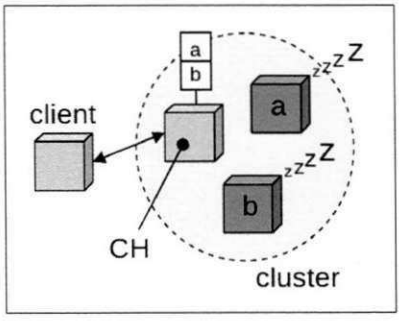


Figura 3.7: *SANDMAN*

Com relação ao protocolo de desativação de dispositivos inativos, é utilizada uma abordagem baseada em clusterização que visa a manutenção da conectividade da rede mesmo

com diversos nós em estado de *sleep*. Uma vez estabelecida a rede, a mesma é dividida em *clusters*, os quais possui um líder, cada um denominado *cluster head* (CH). O agrupamento de nós é feito baseado em padrões de mobilidade (por exemplo, dispositivos pertencentes a um mesmo usuário tendem a compartilhar um padrão de mobilidade) e vizinhança, permitindo que seja possível total comunicação inter-*cluster*. Com os *clusters* formados, é implementada uma política do tipo *round-robin* entre os dispositivos inativos, onde cada um fica "adormecido" por um período fixo negociado entre o dispositivo e o CH. Para cada *cluster*, o CH permanece ativo e fica responsável por realizar o *advertise* dos dispositivos de seu grupo, também respondendo a possíveis buscas. Se uma busca por algum serviço "casa" com algum dispositivo "adormecido" do *cluster*, o cliente é notificado porém deve esperar até que o nó desejado "desperte" e fique disponível para atender à requisição. A figura abaixo resume o protocolo do SANDMAN, onde há um *cluster* com dois dispositivos n_1 (CH) e n_2, além de um cliente n_3. Neste cenário, n_2 negocia um tempo t_s com n_1 e adormece. Nesse meio tempo, n_3 efetua uma busca, a qual é recebida pelo CH e processada. Este, por sua vez, percebe que a busca "casa" com os serviços oferecidos por n_2 e responde a n_3 com o tempo restante que ele deve esperar para que n_2 fique disponível (zero para o caso de o dispositivo estar ativo). Após esse tempo, o cliente pode contactar n_2 diretamente e usar o serviço desejado. Uma visão geral do protocolo adotado pelo SANDMAN pode ser vista na figura 3.8.

Algumas limitações foram observadas no trabalho. A primeira delas diz respeito ao método de agrupamento dos nós da rede. Pelos resultados mostrados no trabalho, redes onde a clusterização resulta em grupos com poucos nós têm um desempenho de economia bem inferior, chegando a ser pior do que não aplicar a técnica no caso de *clusters* unitários. Em redes pervasivas e até mesmo em redes *ad hoc* de propósito não específico, a heterogeneidade dos nós é significativa, e nem sempre é possível agrupá-los em grandes grupos. Um outro ponto crítico é o fato de os clientes terem obrigatoriamente que esperar pelo fim do período de *stand by*. O ideal seria haver uma forma de o CH poder "acordar" os nós do *cluster* quando necessário, minimizando a latência quando, por exemplo, um cliente requisita um serviço de um nó que acabou de entrar em estado de *sleep*. Um outro aspecto que vale ser destacado é o fato de, no caso de haverem dois serviços iguais sendo oferecidos, não ser possível escolher aquele dispositivo que tem melhor condições de oferecê-lo. Por exemplo, se há dois nós que

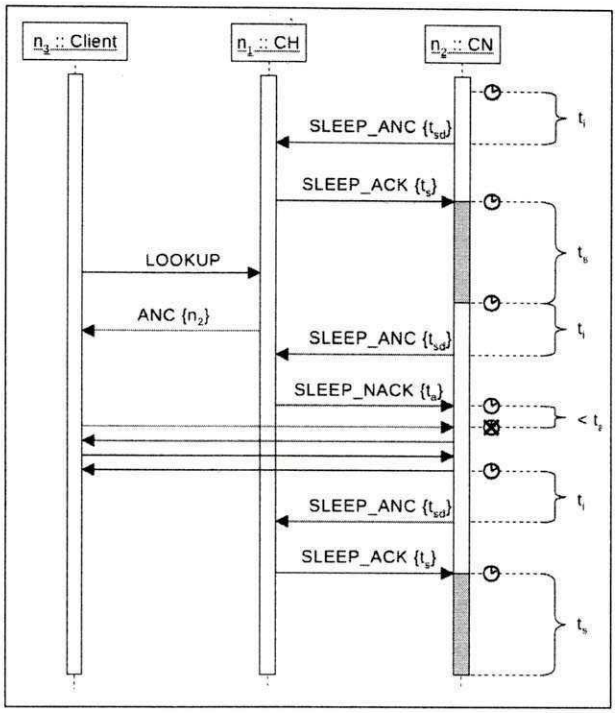


Figura 3.8: *Protocolo SANDMAN*

fornecem acesso à Internet, um com nível de bateria muito baixo e outro com mais reserva de energia, o cliente escolherá o primeiro que estiver disponível. Nesse caso, um critério de seleção baseado no conceito de "mais capaz" poderia ser adotado a fim de selecionar o melhor provedor de serviço. Melhoras nesse e em outros aspectos são previstas nos trabalhos futuros pelos autores.

3.4 *Middleware* para Ciência de Consumo de Energia de Dispositivos Móveis

Em 2009, um artigo [35] publicado por Xiao et. al. apresentou uma proposta de arquitetura de *middleware* para habilitar ciência de consumo (*energy-awareness*) em dispositivos móveis. Os autores definem uma estrutura que provê classificação de aplicações em categorias de consumo, estimação do custo de execução de cada aplicação, bem como definição de políticas de economia de energia baseadas em regras predefinidas. A ideia é realizar um gerenciamento de consumo global baseado no comportamento específico de cada aplicação, utilizando para isso técnicas de aprendizado e autogerenciamento de recursos orientado por

eventos e regras.

O trabalho se propõe em dotar as aplicações com capacidades de adaptação a diferentes condições com o objetivo de economizar energia. O seguinte cenário é considerado: um usuário está viajando de trem e vendo um vídeo *online* através de seu *smartphone*. Quando passa por túneis, o sinal de Internet fica indisponível e nesse meio tempo o usuário lê notícias que estão sendo baixadas por um cliente de *feed* RSS. O dispositivo, dotado de funcionalidades adaptativas, toma medidas com o objetivo de maximizar a duração da bateria, desativando a interface de rede nos túneis para evitar tentativas de transmissão e recepção desnecessárias. Analogamente, quando a bateria chega a um nível médio, a luz da tela tem o seu brilho diminuído, visando reduzir o consumo na reprodução do vídeo. Quando o nível de bateria alcança patamares mais críticos, a aplicação sugere ao usuário que suspenda a reprodução do vídeo, ao mesmo tempo que diminui a frequência com que o leitor de RSS agrega novas notícias.

Neste contexto, o *framework* proposto assume o papel de tomar decisões como as mencionadas anteriormente, tornando transparentes as heterogeneidades de cada plataforma, e provendo uma interface uniforme de decisões que podem ser tomadas pelas aplicações em determinados momentos. O modelo abstrai as funções responsáveis por recuperar informações e controlar elementos de *hardware* do sistema em um serviço **gerenciador de recursos**. Além disso, existe um componente de aprendizado responsável por classificar as aplicações baseado em suas características de consumo, auxiliando o componente **gerenciador de políticas** a definir operações de adaptação que devem ser invocadas em condições específicas (e.g. nível de bateria abaixo de 25%). As políticas de adaptação são gerenciadas e escalonadas automaticamente, e as condições de ativação dessas políticas incluem não somente eventos vindos do gerenciador de recursos, mas também conhecimento gerado pelos componentes **classificador de aplicações** e **estimador de consumo**. Este último se baseia em medidas obtidas de monitores responsáveis por coletar dados de carga de dispositivos como GPS, interfaces de rede, entre outros. A figura 3.9 mostra um modelo arquitetural do *framework* proposto.

O trabalho foi avaliado através de um estudo de caso com uma aplicação que baixa e reproduz vídeos no formato *Flash* da Internet, comparando seu desempenho com e sem a aplicação das políticas de adptação de consumo. Para este caso, o fluxo funcional entre os

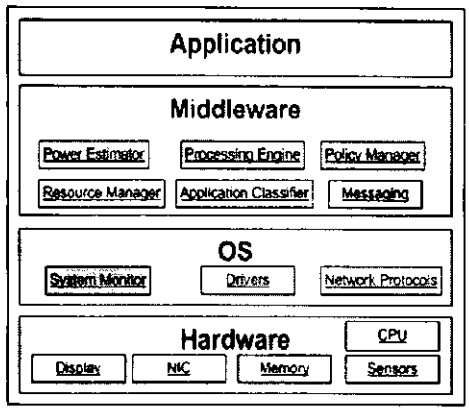


Figura 3.9: *Arquitetura*

componentes do *framework* pode ser resumido da seguinte forma:

1. No *boot* do sistema, o **engenho de processamento** se prepara para ser notificado quando o processo de aplicação executar e o *download* do vídeo for iniciado.

2. O algoritmo classificador de aplicações então classifica o processo baseado nas monitorações vindas do gerenciador de recursos. De acordo com a proposta de categorias existente, o processo é classificado como uma aplicação com intensa transmissão/recepção de dados.

3. O gerenciador de recursos então sinaliza eventos como condições da rede e nível de bateria, fazendo com que o engenho de processamento requisite uma estimação do consumo ao componente estimador.

4. Baseado na medida encontrada pelo estimador de consumo, é requisitado ao componente gerenciador de políticas que defina ações a serem tomadas de acordo com os eventos recebidos. No caso específico desta aplicação, uma regra foi definida para monitorar a medida da razão entre a qualidade do sinal e o ruído no mesmo (*Signal-to-Noise Ratio - SNR*). O nível de bateria também é checado, e ao ser atingido um certo limite de SNR, o *download* é interrompido, sendo retomado quando melhores condições de SNR são verificadas.

Apesar de ter uma arquitetura consistente e os resultados obtidos no estudo de caso indicarem uma melhora no consumo de bateria do dispositivo, o trabalho peca em diversos

pontos. Primeiramente, o estudo de caso em si é insuficiente, pois utiliza apenas um dispositivo em condições de uso bem favoráveis. Apenas a execução de uma única aplicação é considerada para validar a arquitetura. A maioria dos dispositivos móveis atuais dá suporte à execução multitarefa, e este fato deveria ter sido observado pelos autores quando da definição do estudo de caso. A escalabilidade também não é verificada e nem mesmo posta à prova. Com relação ao modelo em si, os autores afirmam usar métodos de aprendizagem de máquina para realizar a classificação das aplicações. No entanto, este algoritmo não é detalhado, ficando em aberto, entre outros aspectos, a questão da fonte dos dados de treinamento para o componente classificador. Uma outra limitação observada é que não se considera a possibilidade de execução de aplicações distribuídas. A arquitetura não prevê cenários onde um dispositivo usa um recurso de outro, por exemplo. O que se pode concluir ao analisar o modelo é que apenas são considerados casos relativamente triviais de aplicações e dispositivos únicos, que rodam em um único processo, sem concorrência e com grande disponibilidade de recursos. Cenários de uso real com essas características nem sempre são factíveis.

3.5 Outros Trabalhos Relacionados

Foram encontrados algumas outras abordagens relacionadas, como [26], [37], [24] e [20], principalmente em áreas correlatas como redes *ad hoc* e redes de sensores sem fio. Muitas delas partem dos mesmos princípios, aplicando técnicas bem semelhantes às vistas nos trabalhos acima detalhados, salvo diferenças julgadas irrelevantes. Ocorre também que muitas ideias utilizadas em abordagens aplicadas a outras áreas se valeram de recursos específicos de ambientes como redes de sensores (e.g. aplicação de DPM [28] e controle de topologia em redes de sensores sem fio). Por isso, decidiu-se não incluir subseções específicas nestes casos.

Capítulo 4

Abordagem Orientada a Serviço para o Gerenciamento de Energia em Redes Pervasivas UPnP

Neste capítulo, apresenta-se a solução desenvolvida para o gerenciamento de energia orientado a serviço em redes pervasivas. Primeiro, é feita uma contextualização do problema a ser resolvido. Logo após, são descritos os requisitos necessários para que a abordagem seja aplicável. Por fim, apresenta-se a solução, consistindo de uma visão geral da mesma e o seu comportamento em dois tipos de cenários previstos.

4.1 Contextualização

Como discutido nos capítulos 1 e 3, verifica-se que a maioria dos esforços que visam diminuir o consumo de energia de dispositivos foram definidos partindo de um ponto de vista de utilização isolada, visando a otimização do consumo de uma forma independente do restante da rede. Assim, cenários onde os dispositivos estão conectados e executando serviços distribuídos não são considerados. Nesse contexto, o que se quer propor é uma abordagem transversal de gerenciamento de energia desses dispositivos, com o intuito de otimizar a execução dos serviços em relação à energia necessária para disponibilizá-los, mantendo um nível de eficiência aceitável.

A arquitetura *Low Power* do padrão UPnP fornece uma solução parcial do problema,

definindo estados de economia de energia que podem ser implementados pelos dispositivos. Um dispositivo é dito compatível com a especificação *Low Power* quando implementa dois ou mais estados de economia de energia. Isso, em conjunto com a figura do *proxy*, definida também pelo padrão, permite que os nós da rede que estiverem em algum estado de *stand by* continuem alcançáveis. Na chegada de uma requisição por um serviço oferecido por algum dos dispositivos em estado de economia, o mesmo pode ser "acordado" pelo *proxy* a fim de atender àquela solicitação.

Baseado nisso, propõe-se aqui um modelo de gerenciamento de energia de dispositivos UPnP compatíveis com a especificação *Low Power Architecture*, o qual parte de um ponto de vista orientado a serviço. Portanto, o que se quer é otimizar a execução de serviços UPnP, garantindo que o mínimo possível de energia seja consumida na disponibilização dos mesmos. Este objetivo pode ser alcançado adotando algumas práticas já definidas na arquitetura *Low Power* original, bem como também implementando melhorias julgadas necessárias ao longo da pesquisa.

4.2 Requisitos de Aplicabilidade

Primeiramente, a solução aqui proposta se baseia no padrão UPnP, versão 1.1, de Outubro de 2008. Isto significa que a sua aplicação se limita a ambientes formados por dispositivos que implementam a pilha de protocolos do padrão UPnP de alguma forma. Além disso, o modelo assume que **todos** os dispositivos são compatíveis com a especificação *Low Power Architecture* (LPA), versão 1.0, publicada pelo UPnP Forum em Agosto de 2007. Um dispositivo é compatível com a LPA se ele implementa o serviço *LowPowerDevice:1* [1]. Opcionalmente, o serviço *LowPowerProxy:1* [2] deve ser implementado por aqueles dispositivos que serão ou poderão ser *proxies* na rede. A definição de *proxy* e o funcionamento da arquitetura UPnP *Low Power* foram discutidos no capítulo 2.

Nas próximas subseções, é discutido em detalhes o funcionamento dos serviços *LowPowerDevice Service* e *LowPowerProxy Service*.

[1] http://upnp.org/specs/lp/UPnP-lp-LowPower-v1-Service.pdf

[2] http://upnp.org/specs/lp/UPnP-lp-LowPowerProxy-v1-Service.pdf

4.2.1 *LowPowerDevice Service* - LPDS

Diz-se que um dispositivo UPnP é compatível com a especificação *Low Power* quando o mesmo disponibiliza um serviço denominado *LowPowerDevice Service*. O documento que especifica este serviço faz parte da arquitetura e pode ser consultado na página do UPnP Forum. Esta subseção tem como objetivo mostrar o seu funcionamento.

O LPDS provê ações para que pontos de controle compatíveis com a especificação possam consultar e gerenciar dispositivos UPnP com capacidades de economia de energia. Para isso, o dispositivo deve suportar os *headers* adicionais necessários às mensagens SSDP e ser compatível com as transições definidas pela máquina de estados mostrada no capítulo 2.

Extensões ao Protocolo SSDP

Como visto no capítulo 2, o SSDP é o protocolo responsável pelo anúncio e descoberta de serviços/dispositivos adotado pelo padrão UPnP. Tendo em vista tornar um dispositivo compatível com a arquitetura *Low Power*, alguns campos novos devem ser adicionados às mensagens de descoberta e anúncio do protocolo. A tabela 4.1 resume quais campos são esses e seus possíveis valores.

Campo	Descrição
PowerState	Indica o novo estado energético do dispositivo
SleepPeriod	Indica o período que o dispositivo permanecerá em um certo estado, em segundos.

Tabela 4.1: Novos campos SSDP

Variáveis de Estado

O serviço LPDS mantém algumas variáveis de estado as quais podem ser usadas por pontos de controle e *proxies* com o objetivo de auxiliar no gerenciamento e controle de dispositivos de baixo consumo que estejam presentes na rede. São providas algumas informações úteis, como métodos específicos de *wake up* (para o caso de *Deep Sleep Offline*, principalmente), período de tempo que o dispositivo passará em um estado de economia de energia, entre outras. A tabela 4.2 indica que variáveis de estado são mantidas pelo serviço *LowPowerDevice*.

Variável	Descrição
BatteryLow	Indica se a bateria está em nível crítico
PowerSupplyStatus	Indica o estado atual da fonte de alimentação
WakeupMethod	Método a ser usado para despertar o dispositivo
ExternalPowerSupplySource	Indica se a fonte atual de energia é a rede elétrica ou interna
SleepPeriod	Período que o dispositivo permanecerá em um certo estado
PowerState	Indica o estado de energia atual do dispositivo

Tabela 4.2: Variáveis de Estado do LPDS

Ações e Operações de Controle

Um dispositivo deve disponibilizar informações a respeito de suas próprias características de gerenciamento de energia através da ação ***GetPowerManagementInfo***. Quando invocada, esta ação retorna um arquivo XML com informações que podem incluir detalhes sobre métodos específicos de *wake up*, quais estados de economia que são implementados por aquele dispositivo, nível atual de bateria e suporte a fontes externas de alimentação. A Listagem 4.1 mostra um trecho do que seria um possível retorno dessa ação.

Listagem 4.1: Informações de Gerenciamento de Energia do Dispositivo

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <p:PowerSupplyStatus xmlns:p="urn:schemas-upnp-org:lp:PowerSupplyStatus"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
4    <p:ExternalPowerSupply>
5      <p:ExternalPowerSourceInfo>AC</p:ExternalPowerSourceInfo>
6      <p:IsConnected>true</p:IsConnected>
7    </p:ExternalPowerSupply>
8    <p:InternalPowerSupply>
9      <p:PowerRemaining>95</p:PowerRemaining>
10     <p:TimeRemaining>P00DT10H30M</p:TimeRemaining>
11   </p:InternalPowerSupply>
12 </p:PowerSupplyStatus>
13 <p:WakeupMethod xmlns:p="urn:schemas-upnp-org:lp:WakeupMethod"
14 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
15   <p:BearerWakeupMethod>
16     <p:IanaTechnologyType>71</p:IanaTechnologyType>
```

```
17 <p:WakeupPattern>0008744C7F1D0008744C7F1D0008744C7F1D0008744C7F1D0008
18 744C7F1D0008744C7F1D0008744C7F1D0008744C7F1D0008744C7F1D0008744C7F1D0
19 008744C7F1D0008744C7F1D0008744C7F1D0008744C7F1D0008744C7F1D0008744C7F
20 1D</p:WakeupPattern>
21     <p:AdditionalBearerInfo>
22       <p:Address>0008744C7F1D</p:Address>
23       <p:Bearer_Info>Adsilon </p:Bearer_Info>
24     </p:AdditionalBearerInfo>
25   </p:BearerWakeupMethod>
26   <p:NonBearerWakeupMethod>
27     <p:BearerType>Infrared </p:BearerType>
28     <p:VendorNonBearerInfo>Kolhips </p:VendorNonBearerInfo>
29   </p:NonBearerWakeupMethod>
30 </p:WakeupMethod>
```

Pontos de controle podem invocar a ação ***Wakeup*** para solicitar que um determinado dispositivo "acorde" de um estado de economia (*Transparent Sleep*, *Deep Sleep Online*). A ação ***GoToSleep*** tem o funcionamento simétrico, permitindo que um ponto de controle (ou *proxy*) requisite ao dispositivo que entre em um estado econômico por um certo período de tempo. Um detalhe neste caso é que o dispositivo em questão pode ou não aceitar a sugestão de tempo de *sleep* vinda do ponto de controle. O retorno desta ação é uma tupla com o estado para o qual será feita a transição e o período de tempo que o dispositivo permanecerá neste estado.

Sempre que há uma transição entre um estado de economia e outro, esta deve ser anunciada à rede através de mensagens SSDP, utilizando as devidas extensões nas mensagens. Algumas considerações são feitas a seguir:

- Quando um LPD entra no estado *Active*, o mesmo passa a enviar mensagens do tipo **ssdp:alive** *header* ***PoweState*** definido para **active**. As mensagens normais de *heartbeat* (do tipo **ssdp:keepalive** previstas pela especificação UPnP continuam sendo enviadas periodicamente.

- Transições para o estado *Transparent Sleep* devem ser anunciadas através de mensagens **ssdp:alive** com o campo *PowerState* definido para o valor **transparent**. Neste estado, mensagens de *heartbeat* também continuam sendo enviadas periodicamente.

- Quando um dispositivo entra em *Deep Sleep Online* ou *Deep Sleep Offline*, é enviada uma mensagem do tipo **ssdp:byebye** com o campo *PowerState* definido com o valor correspondente. Nesses estados, mensagens **ssdp:keepalive** têm o seu envio suspenso.

- Recomenda-se que transições de estados menos ativos para mais ativos aconteçam de forma gradual, passando por todos os estados mais ativos primeiro. Por exemplo, para sair de *Deep Sleep Offline* para *Deep Sleep Online*, o dispositivo deve primeiro efetuar a transição para *Active* e depois para *Transparent Sleep*, finalmente chegando no estado desejado. Apesar de não ser obrigatória, essa prática visa manter a consistência do banco de dados de dispositivos mantido por *proxies* ou pontos de controle, já que há estados onde mensagens de anúncio *multicast* não são enviadas para a rede.

- Transições de qualquer estado para *Disconnected* devem ser anunciadas por mensagens **ssdp:byebye** usuais, sem a necessidade de definir o campo *PowerState* próprio da especificação LPA.

As listagens acima mostram exemplos de mensagem anunciando transições entre estados de economia. Nota-se a presença de um novo campo na mensagem SSDP usual, denominado *PowerState*.

Ao receber uma busca M-SEARCH, um dispositivos no estado *Active* ou *Transparent Sleep* responde com o estado atual correspondente embutido na mensagem de descoberta. Quando no estado *Deep Sleep Online*, LPDs só respondem a invocações da ação *Wakeup*, que podem ser feitas por *proxies* ou pontos de controle. Dispositivos no estado *Deep Sleep Offline* não respondem a descobertas nem invocações de ações, podendo ser "despertados" apenas por métodos dependentes de fabricante (e.g. Wake-on-LAN) ou por relógios internos. Neste caso, a forma como um dispositivo nesse estado pode ser acordado deve ser disponibilizada na resposta da ação *GetPowerManagementInfo* e/ou no seu arquivo de descrição.

Vale salientar que a ação *GoToSleep* tem como parâmetro um número que indica uma quantidade de tempo na qual o dispositivo passará em um determinado estado. Todos os dispositivos devem eventualmente despertar e renovar sua presença na rede quando este tempo se esgota. Por renovar a presença, entenda-se transitar para um estado ativo e anunciar para o restante da rede esta transição. Isso contribui com a consistência da rede, sendo possível para *proxies* e pontos de controle tomarem conhecimento de dispositivos que por ventura

deixaram a rede enquanto estavam em um estado de economia.

4.2.2 Low Power Proxy Service - LPPS

Além da figura do dispositivo UPnP *Low Power*, foi discutido no capítulo 2 o papel de uma entidade denominada *proxy* (*Low Power Proxy - LPP*). Em uma rede UPnP compatível com a especificação LPA, o *proxy* é o responsável descobrir dispositivos *Low Power* presentes na rede e facilitar a descoberta daqueles que estejam em um estado de economia e, consequentemente, invisíveis aos pontos de controle. Para isso, é mantido um registro dos dispositivos que implementam o LPDS e os seus respecitivos estados.

O LPPS também provê meios de se conseguir "despertar" um determinado dispositivo que esteja em um estado econômico. Para isso, as informações de *wake up* específicas que são disponibilizadas no LPDS são usadas no momento que um ponto de controle solicitar que um dado LPD seja trazido a um estado ativo.

Assim, nota-se que o *proxy* é uma entidade que possui comportamento tanto de ponto de controle (cliente) quanto de dispositivo (servidor). O LPP invoca ações dos dispositivos para manter seu registro interno e também provê ações para que pontos de controle tomem conhecimento dos LPDs presentes na rede.

Ações e Operações de Controle

Um LPP mantém registro das informações de gerenciamento de energia de todos os dispositivos *Low Power* presentes na rede. Essas informações são recuperadas no momento do anúncio da chegda do dispositivo e armazenadas em uma variável de estado chamada ***DeviceListInfo***. A ação *GetPowerManagementInfo* disponibilizada no LPDS é fundamental para este passo. Além disso, o estado atual de cada LPD também é mantido, sendo este atualizado sempre que um dispositivo anunciar uma transição para outro estado. Um ponto de controle pode solicitar ao *proxy* o valor atual dessa variável de estado através da ação ***SearchSleepingDevices***. Esta ação pode receber como parâmetro um certo filtro de pesquisa (como tipo de dispositivo, tipo de serviço e estado atual), retornando apenas os dispositivos compatíveis com os critérios especificados. O retorno desta ação é um arquivo XML que deve conter no mínimo os seguintes campos:

- Informações específicas do dispositivo/serviço, as mesmas enviadas em respostas SSDP;
- O estado de economia atual do dispositivo (*Transparent Sleep*, *Deep Sleep Online* ou *Deep Sleep Offline*);
- O tempo no qual o dispositivo permanecerá nesse estado;
- Se é possível que o dispositivo seja "despertado" do estado onde se encontra antes que o tempo anunciado anteriormente expire;
- O método usado para acordar o LPD para o caso de ele se encontrar no estado *Deep Sleep Offline* (sem conectividade IP).

O LPP provê outra ação, chamada ***WakeupDevice***. Como o próprio nome sugere, esta ação é invocada por um ponto de controle quando se deseja despertar um determinado LPD do seu estado de economia, e recebe como parâmetro o identificador único (UUID) de dispositivo. Quando o *proxy* recebe uma requisição desse tipo, primeiro é verificado o estado atual no qual se encontra o LPD. Se ele estiver no estado *Deep Sleep Offline* e o *proxy* "sabe" como despertá-lo através de seu método específico (e.g. Wake-on-LAN), faz-se a tentativa. Se o LPD estiver no estado *Deep Sleep Online* ou *Transparent Sleep*, invoca-se a ação *Wakeup* referente ao LPDS. Caso não seja recebida nenhuma resposta, em ambos os casos tenta-se mais uma vez usar o método específico. Em não havendo retorno mesmo assim, assume-se que o dispositivo foi para o estado *Disconnected* e uma mensagem de erro é enviada ao ponto de controle.

4.2.3 Relação com o Padrão UPnP

A dependência do padrão UPnP não é estrita. Mesmo que a abordagem apresentada aqui tenha sido desenvolvida levando em consideração esse tipo de ambiente, as ideias aqui expostas podem ser aplicadas a outras arquiteturas de rede, desde que existam entidades equivalentes àquelas que são consideradas nesta solução.

4.3 Visão Geral da Solução

Considerando o contexto de redes UPnP com suporte à presença de dispositivos de baixo consumo (ou seja, aqueles que implementam o LPDS e/ou o LPPS), foi desenvolvida uma solução que procura diminuir o consumo total da rede baseado em uma abordagem orientada a serviço. Os objetivos foram:

1. Garantir o funcionamento da rede mantendo sempre a maior quantidade possível de disposistivos em estado de economia, sem prejudicar seu desempenho;
2. Melhorar a disponibilização de serviços distribuídos escolhendo sempre o melhor dispositivo provedor em um determinado momento;
3. Prover estimativas de consumo total da rede, podendo manter um comportamento compatível com limiares pré-definidos;
4. Permitir a definição de regras de gerenciamento de serviços, as quais podem ser aplicadas em determinadas situações.

Para isso, foi definido um novo serviço UPnP chamado *LowPowerManagement Service*, o qual faz uso dos serviços *LowPowerDevice* e *LowPowerProxy*. Este serviço é parte fundamental na tentativa de se gerenciar de forma transversal o consumo de energia de toda a rede UPnP.

O serviço *LowPowerManagement* faz uso dos mecanismos implementados na especificação UPnP *Low Power* e provê ações de controle e gerenciamento da rede através de suas ações e eventos. Na figura 4.1, ilustra-se onde se localiza o serviço de *LowPowerManagement* de um ponto de vista de arquitetura em nível de camadas.

Para fins de implementação, uma camada de *middleware* deve ser adicionada abaixo da camada de aplicações. O Capítulo 5 aborda um exemplo de *middleware* que implementa o padrão UPnP, denominado BRisa.

4.3.1 O Papel do *Proxy* no Modelo de Gerenciamento de Energia

A figura do *proxy* é de extrema importância para que a solução aqui proposta seja aplicável. Dispositivos que disponibilizam o serviço de *proxy* podem responder por aqueles que estão

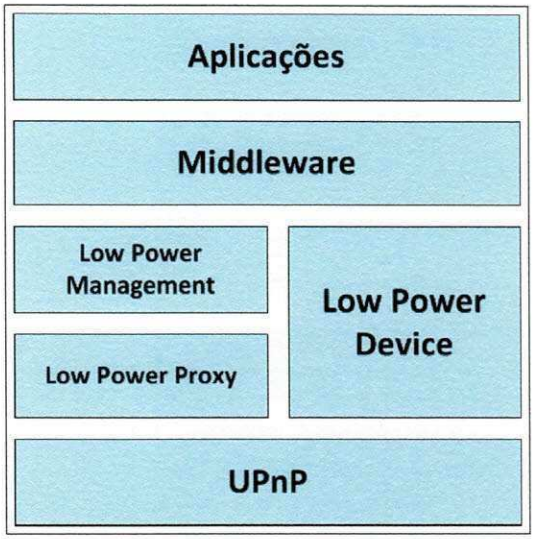


Figura 4.1: *Serviço LowPowerManagement*

"adormecidos" em estados de economia, atuando tanto como ponto de controle como dispositivo controlável. Assim, é fácil ver que o papel de *proxy* deve ser assumido idealmente por um dispositivo que nunca "dorme", ou seja, que nunca é desligado ou entra em modo econômico.

Neste contexto, surge o seguinte questionamento: **que nó deve ser escolhido para ser *proxy***? A resposta para essa pergunta tem duas possibilidades: uma para redes UPnP infraestruturadas e outra para redes UPnP em modo *ad hoc*. Este e outros aspectos são abordados a seguir.

Escolha do *Proxy* - Redes UPnP Infraestruturadas

Uma rede é dita **infraestruturada** quando todos os seus nós estão conectados a um dado **ponto de acesso** (AP). Pontos de acesso são cientes de toda a comunicação advinda de ou endereçada a um certo nó conectado, atuando como roteadores do tráfego da rede. Pode haver mais de um AP em uma determinada rede, e cada um deles pode gerenciar sua própria sub-rede.

Em redes infraestruturadas, pontos de acesso são cientes de todo o tráfego gerado e recebido pelos dispositivos a eles associados. Também são responsáveis por manter dois ou mais dispositivos conectados entre si, intermediando a comunicação entre os pares. Por isso,

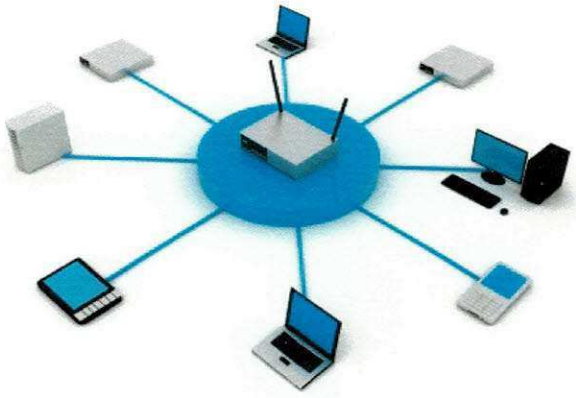


Figura 4.2: *Rede Infraestruturada*

os APs devem estar sempre ligados e ativos, garantindo o funcionamento de toda a rede e a alcançabilidade de todos os nós.

Assim, para o caso de redes UPnP infraestruturadas, todos os pontos de acesso são potenciais *proxies*, e um deles deve ser escolhido para tal, assumindo que este estará sempre ativo. A escolha é arbitrária, haja vista que todo dispositivo UPnP de baixo consumo anuncia seu estado e transições para toda a rede através de mensagens *multicast*. Em [14], foi usada uma abordagem similar, onde uma figura equivalente ao *proxy* UPnP pode ser instalada em dispositivos como pontos de acesso, interfaces de controle de rede (NICs), entre outros.

Escolha do *Proxy* - Redes UPnP *Ad Hoc*

Ao contrário de uma rede infraestruturada, nas redes *ad hoc* não há a presença de pontos de acesso, e os nós dependem uns dos outros para manter a rede conectada. Nesse caso, a comunicação entre os pares acontece de forma direta (P2P), sem a necessidade de intermediação por um terceiro nó. Por esse motivo, redes *ad hoc* são indicadas principalmente em situações onde não se pode, ou não faz sentido, instalar uma rede fixa.

Com isso, não se tem necessariamente um dispositivo que esteja ligado e ativo todo o tempo, dificultando assim a indicação de um ou mais *proxies*. Neste caso, foi definido um método de "eleição" distribuída de *proxies*, no qual cada dispositivo que implementa o LPDS é um potencial candidato. Foi definida também uma métrica de capacidade e, baseado nela, um dentre os potenciais *proxies* é escolhido e assume o papel de gerenciar os demais dispo-

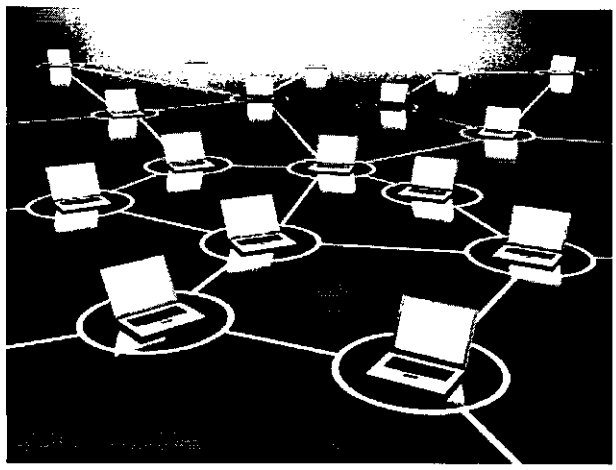


Figura 4.3: *Rede Ad Hoc*

sitivos de acordo com a especificação do serviço *LowPowerProxy* discutido anteriormente. O método de escolha do *proxy* em uma rede UPnP *ad hoc* será abordado com detalhes na próxima seção.

LowPowerManagement

Como discutido anteriormente, a arquitetura UPnP *Low Power* fornece uma solução parcial para o gerenciamento de dispositivos conectados em uma rede UPnP. Assim, foi definido um modo de adicionar funcionalidades ao *proxy* através da disponibilização de um novo serviço UPnP chamado *LowPowerManagement*. Assim, é necessário que dispositivos que venham a ser potenciais *proxies* implementem ambos os serviços para que as funcionalidades adicionais tenham efeito. A seguir, é detalhado o funcionamento do serviço *LowPowerManagement* no contexto de toda a solução.

4.4 Funcionamento da Solução

Esta seção aborda o funcionamento da solução de gerenciamento de energia, baseada na especificação UPnP *Low Power Architecture* e no novo serviço UPnP denominado *LowPowerManagement*, o qual depende da referida arquitetura. Em um primeiro momento, será abordado o caso mais simples, envolvendo redes UPnP infraestruturadas. Logo após, será discutido o caso de redes UPnP totalmente *ad hoc*. Para ambos os casos, assume-se que

todos os dispositivos UPnP que se quer gerenciar implementam o serviço *LowPowerDevice*, discutido nas seções anteriores.

4.4.1 Gerenciamento de Energia Orientado a Serviço em Redes UPnP Infraestruturadas

Em redes infraestruturadas, a escolha do *proxy* é direta. Um dos pontos de acesso deve implementar o serviço *LowPowerProxy*, e deve também implementar o novo serviço *LowPowerManagement*. Outro requisito necessário ao *proxy* é que ele mesmo nunca entre em um estado de economia, permanecendo ativo. Assim, é possível tomar ciência de todos os dispositivos presentes e manter registro dos seus estados atuais.

Entrada de Dispositivos

Quando um dispositivo entra na rede, cumprem-se todas as etapas previstas no padrão UPnP, e o anúncio de sua entrada é enviado através de uma mensagem *multicast*. Nesta mensagem, é enviado o estado atual do dispositivo (*Active*), através de um campo específico da mensagem SSDP usado pelos LPDs.

Assim que recebe uma mensagem de anúncio, o *proxy* invoca a ação *GetPowerManagementInfo* do dispositivo, a fim de recuperar as suas informações de gerenciamento. De posse desses dados, a presença do dispositivo é registrada na variável de estado ***LowPowerDevices***, definida no novo serviço *LowPowerManagement*. Esta variável consiste de um arquivo XML que armazena **todos** os dispositivos de baixo consumo presentes na rede, e contém as seguintes informações:

- UUID do dispositivo;
- Estado atual no qual se encontra;
- Informações de gerenciamento de energia;
- Serviços disponibilizados.

Foram incluídos dois novos campos ao retorno da ação *GetPowerManagementInfo*, que são o consumo (em kWh) do dispositivo quando ativo e o consumo médio (em kWh) do

dispositivo quando em *stand by*. Estes valores não estão previstos na especificação *Low Power* original e serão sugeridos como adição ao padrão ao UPnP Forum.

Após o registro da presença do novo dispositivo, este permanece em estado ativo durante um certo tempo. Caso durante esse intervalo não haja nenhuma requisição feita por algum ponto de controle, o mesmo entra em estado de *Deep Sleep Online*, anunciando a transição para o restante da rede. O *proxy*, ciente desta transição, registra este dispositivo em sua lista *DeviceListInfo*, como discutido na seção que abordou o serviço *LowPowerProxy*.

O princípio é simples: manter todos os dispositivos que não estejam atendendo alguma requisição devidamende "adormecidos", e solicitá-los quando algum ponto de controle interessado efetue uma busca.

Política de Seleção e Ativação de Serviços

Pontos de controle fazem buscas por dispositivos e serviços através de mensagens *multicast* do tipo M-SEARCH. No contexto de redes UPnP que implementam a arquitetura *Low Power*, os pontos de controle devem ser cientes da presença do *proxy* na rede. De acordo com a especificação, esses pontos de controle requisitam diretamente ao *proxy* as informações de dispositivos "adormecidos" e solicitam que os mesmos sejam trazidos ao estado ativo novamente.

A presente abordagem define também o serviço *LowPowerManagement*. Pontos de controle devem ser cientes deste novo serviço e interagir com o *proxy* através dele, e não mais através do LPPS diretamente. O serviço *LowPowerManagement* dispõe de uma ação chamada ***SearchForService***, que recebe como parâmetro um filtro de busca baseado em serviços. O retorno desta ação são todos os serviços compatíveis com o filtro de busca, independente de sua natureza do dispositivo que o provê.

Baseado nisso, um ponto de controle pode solicitar que um dispositivo seja escolhido para prover um determinado serviço. Isso pode ser feito através da ação ***RequestService***, que recebe como parâmetro o identificador do serviço que se quer ativar. A escolha pelo dispositivo provedor é feita exclusivamente pelo *proxy*.

Ao receber uma invocação à ação *RequestService*, o *proxy* faz pesquisa em seu banco de dados por todos os dispositivos que oferecem aquele serviço. Para cada um, é feito o cálculo da **capacidade** que o dispositivo tem de oferecer aquele serviço naquele momento.

Capacidade aqui é definida como a medida de o quão eficiente um dispositivo pode prover um determinado serviço levando em consideração o consumo de energia. Esse cálculo é feito baseado em algumas das informações de gerenciamento de energia que foram coletadas quando o dispositivo anunciou sua chegada à rede.

O valor da capacidade *cp* de um dispositivo UPnP é calculado da seguinte forma:

$$cp = \begin{cases} f_{ac} + \frac{f_b}{c_a}, & \text{se } f_b > 0, \\ f_{ac} + \frac{1}{c_a}, & \text{se } f_b = 0, \end{cases} \tag{4.1}$$

onde f_{ac} é um valor binário (0 ou 1) que indica se o dispositivo está sendo alimentado pela rede elétrica ou não; f_b é o nível atual de bateria (0 quando o dispositivo não possui bateria); e c_a é o consumo em kWh do dispositivo quando em estado ativo. O valor correspondente ao nível de bateria está disponível em uma variável de estado do serviço *LowPowerDevice*, e é atualizado através de um evento UPnP ao qual o *proxy* deve estar inscrito. O mesmo se aplica à informação acerca do dispositivo estar sendo alimentado pela rede elétrica ou não. Quanto maior o valor de *cp*, mais "apto" um dispositivo está para prover um determinado serviço.

Após calcular o valor da capacidade de cada dispositivo, o *proxy* retorna para o ponto de controle o UUID do dispositivo com o maior valor de *cp*. Logo após, o *proxy* solicita que o dispositivo em questão "acorde", invocando a ação *WakeupDevice* do serviço *LowPowerProxy*. Ao acordar, o dispositivo anuncia sua presença através de uma mensagem *multicast* do tipo **ssdp:alive**, a qual é recebida pelo ponto de controle, que compara o UUID do dispositivo recém-chegado com o que foi retornado pelo *proxy*. Sendo iguais, o ponto de controle pode invocar o serviço desejado diretamente ao dispositivo controlável. Se após um certo tempo nenhum dispositivo com aquele UUID não anunciar presença, assume-se que o mesmo foi para o estado *Disconnected* e o ponto de controle invoca novamente a ação *RequestService*, até que uma mensagem de erro indicando que não existe dispositivo para atender à requisição seja retornada pelo *proxy*.

4.4.2 Gerenciamento de Energia Orientado a Serviço em Redes UPnP *Ad Hoc*

Para o caso de redes UPnP *ad hoc*, o funcionamento das etapas de entrada de dispositivos e seleção e ativação de serviços acontece de maneira similar às redes infraestruturadas, discutidas na subseção anterior. A diferença entre os dois ambientes ocorre na escolha do *proxy*. Em redes que possuem pontos de acesso presentes, a escolha daquele que será o *proxy* é direta. No caso de redes *ad hoc* não existe necessariamente um nó que esteja ativo o tempo todo. Havendo a definição de qual dispositivo deve agir como *proxy* em um determinado momento, o comportamento da rede é o mesmo de uma rede UPnP infraestruturada. Por isso, foi definido um protocolo de eleição distribuída do *proxy*, o qual é apresentado a seguir.

Escolha do *Proxy* em uma Rede UPnP *Ad Hoc*

Como discutido na subseção anterior, um *proxy* é um dispositivo UPnP que implementa dois serviços, a saber: *LowPowerProxy*, definido originalmente na especificação *Low Power* e o novo serviço *LowPowerManagement*, o qual foi definido no presente trabalho, adicionando funcionalidades à figura do *proxy*. Assim, em uma rede UPnP *ad hoc*, os potenciais candidatos a *proxy* são aqueles que atendem esses requisitos, além de implementarem o próprio serviço *LowPowerDevice*.

Por ser um serviço UPnP, o melhor candidato a *proxy* é aquele com a maior **capacidade** de oferecer este serviço. Logo, usa-se o conceito de capacidade definido anteriormente para classificar os potenciais *proxies*. Ao se anunciar na rede, cada dispositivo procura pelo *proxy* através de uma busca M-SEARCH. Caso encontre e não haja nenhuma requisição para atender, o dispositivo entra no estado de *Deep Sleep Online*. Caso não encontre, este dispositivo efetua uma outra busca M-SEARCH em busca de todos os dispositivos de baixo consumo ativos, atualizando sua lista como discutido na subseção anterior. A partir de então, este dispositivo atuará como *proxy* daquela rede, até que, por alguma razão, ele próprio deixe a rede ou entre em um estado de economia diferente de *Transparent Sleep* (este não requer uma mensagem do tipo **ssdp:byebye**). Todos os outros dispositivos que não estiverem atendendo requisições permanecem em estado de economia.

Uma característica da especificação *Low Power* é que todo dispositivo que implementa o

LPDS permanece em um estado econômico por um determinado intervalo de tempo, o qual é anunciado no momento da transição. Ao "despertar" para o estado *Active*, os dispositivos aguardam um tempo por requisições e voltam a "dormir". Outro aspecto do padrão UPnP é que dispositivos ativos devem enviar mensagens de *heartbeat* para o restante da rede para renovar sua presença (do tipo **ssdp:alive**).

A cada "despertar" periódico, os potenciais *proxies* devem esperar no mínimo o tempo de *timeout* para atualizarem suas listas de dispositivos presentes. Ao fazer isso, devem verificar se o *proxy* atual ainda está presente. Caso não esteja, o dispositivo atual calcula sua própria capacidade e a compara com a dos potenciais *proxies* da lista. Caso possua a maior, ele será o novo *proxy*. Caso contrário, o dispositivo volta para o estado de *Deep Sleep Online*. Este processo deve acontecer também ao recebimento de uma mensagem do tipo **ssdp:byebye** vinda do atual *proxy*.

A fim de controlar o estado de cada potencial *proxy*, é usada uma variável de estado de tipo booleano denominada ***CurrentlyProxy***, a qual armazena o estado atual do dispositivo, ou seja, se ele atualmente é *proxy* ou não. Havendo modificação no valor dessa variável, um evento UPnP é gerado, comunicando ao restante da rede a presença de um novo *proxy*.

Capítulo 5

Estudo de Caso

Este capítulo tem como objetivo apresentar um estudo de caso onde a abordagem de gerenciamento de energia discutida no capítulo 4 foi aplicada e avaliada. Para isso, foi usado o *framework* BRisa, um projeto de *software* de código aberto que implementa o padrão UPnP. Algumas adaptações ao BRisa foram necessárias para que o estudo de caso fosse devidamente realizado. As próximas seções mostram os cenários considerados e os resultados obtidos.

5.1 BRisa

O *framework* BRisa [1] provê suporte ao desenvolvimento de soluções baseadas no padrão UPnP e é desenvolvido no Laboratório de Sistemas Embarcados e Computação Pervasiva[1] da Universidade Federal de Campina Grande. Através dele, é possível a construção de dispositivos UPnP e pontos de controle, abstraindo do desenvolvedor a complexidade dos passos de conectividade UPnP. Dentre as funcionalidades previstas no *framework*, destacam-se:

- Definição e geração da descrição do dispositivo e dos serviços: é disponibilizada uma API para definir as propriedades de um dispositivo, bem como seus serviços, com os respectivos parâmetros de entrada, saída e geração de eventos UPnP. Automaticamente, o BRisa constrói o conteúdo XML da descrição do dispositivo e do serviço com base em informações definidas programaticamente.

[1] http://brisa.garage.maemo.org

- API para a construção de pontos de controle: o framework disponibiliza também uma API para a construção de pontos de controle, oferecendo os métodos necessários para a descoberta e invocação de serviços UPnP, além dos mecanismos para inscrever-se em eventos UPnP de entrada e saída de dispositivos.

Além disso, o BRisa disponibiliza também a implementação de dois dispositivos UPnP presentes na especificação UPnP A/V. O BRisa *Media Server* e o BRisa *Media Renderer*. O primeiro define uma arquitetura baseada em *plug-ins* para consulta de conteúdos multimídia de diferentes fontes de dados, os quais podem ser reproduzidos no dispositivo BRisa *Media Renderer*.

5.2 Descrição do Estudo de Caso

Neste estudo de caso, foi avaliado o desempenho da rede com relação ao consumo com e sem o uso da abordagem de gerenciamento. Para isso, foram implementados 3 cenários, 2 deles considerando uma rede UPnP infraestruturada e 1 considerando uma rede UPnP *ad hoc*. Foram usados alguns dispositivos reais e outros simulados em um computador executando uma distribuição Linux. A seguir, a lista dos dispositivos utilizados:

- Dois *internet tablets* Nokia N800, rodando a distribuição Linux Maemo;
- Um *smartphone* Nokia N900, rodando a distribuição Linux Maemo;
- Um *laptop* com processador Intel Celeron, rodando a distribuição Arch Linux;
- Um PC/servidor com processador Intel Xeon, rodando a distribuição Arch Linux.

Com o intuito de verificar o comportamento da solução proposta, comparou-se o desempenho da rede sem o uso da abordagem com o desempenho nas mesmas condições incluindo o suporte ao gerenciamento de energia. Para isso,foram usadas duas versões do BRisa: a comum, disponibilizada no repositório, e a modificada, que inclui o suporte à especificação *Low Power* e ao serviço *LowPowerManagement*. Em ambos os casos, o experimento foi executado pela mesma quantidade de tempo e o consumo médio da rede foi estimado através de um mecanismo de *logging* que registrava o tempo que o dispositivo ficou em estado ativo

durante todo o experimento. Com essa informação foi possível estimar o consumo de toda a rede através das informações de consumo disponibilizadas pelos próprios aparelhos.

5.3 Cenários

Nesta seção são apresentados os cenários usados na implementação do estudo de caso.

5.3.1 Redes UPnP Infraestruturadas

Foram desenvolvidos dois cenários para o caso de uma rede UPnP infraestruturada. No primeiro deles, avaliou-se o comportamento da rede para o caso de diversos dispositivos oferecendo o serviço de *Media Renderer* com o intuito de reproduzir um arquivo de vídeo armazenado em um *Media Server*. O segundo cenário, por sua vez, simula a presença de diversas impressoras UPnP na rede, através do serviço *Printer Basic Service*, definido pelo UPnP Forum.

Cenário 1 - Serviço UPnP *Media Renderer*

O primeiro cenário consiste de vários dispositivos oferencendo o serviço de *Media Renderer*, e um ponto de controle requisitando este serviço a fim de reproduzir um arquivo de vídeo armazenado em um dispositivo *Media Server*. Foi utilizada a seguinte configuração:

- O PC como sendo o *proxy* da rede, quando utilizada a solução de gerenciamento;
- Um *tablet* disponibilizando o serviço de *Media Renderer*;
- Dois dispositivos simulados pelo *laptop* também disponibilizando o serviço de *Media Renderer*;
- O *smartphone* disponibilizando o serviço de *Media Server*;
- Um *tablet* atuando como ponto de controle UPnP.

A figura 5.1 mostra o esquema montado para o caso de redes com suporte a gerenciamento de energia. O caso contrário apenas exclui a presença do *proxy*.

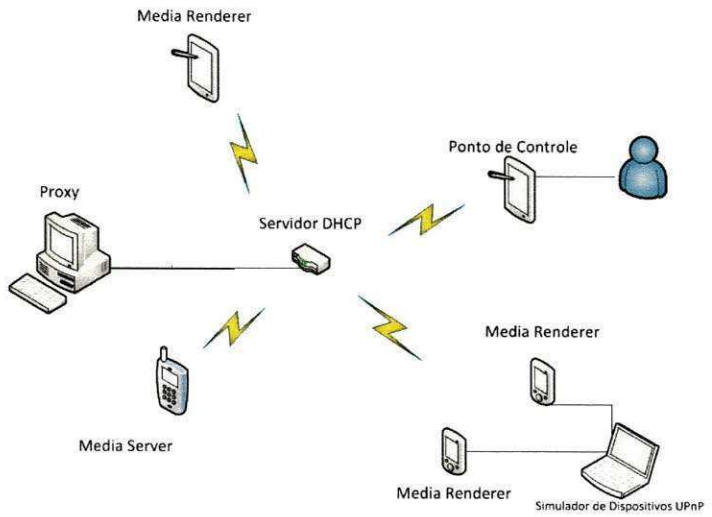


Figura 5.1: *Cenário de Teste 1*

Cenário 2 - Serviço UPnP *Printer Basic*

Neste segundo cenário, foi simulada a presença de impressoras UPnP, as quais oferecem o serviço *Printer Basic Service*, definido pelo UPnP Forum. Foram usados dois pontos de controle que requisitam os serviços de impressão e verificado o comportamento da rede. Os dispositivos foram configurados da seguinte forma:

- O PC como sendo o *proxy* da rede, quando utilizada a solução de gerenciamento;
- Os dois *tablets* e o *smartphone* simulando o serviço de impressora UPnP;
- *Laptop* simulando dois pontos de controle.

A figura 5.2 mostra o esquema montado para o caso de redes com suporte a gerenciamento de energia. O caso contrário apenas exclui a presença do *proxy*.

5.3.2 Redes UPnP *Ad Hoc*

Para o caso de rede UPnP *ad hoc*, foi implementado um cenário para o estudo de caso, o qual consiste em vários dispositivos oferecendo o serviço de *Media Renderer*.

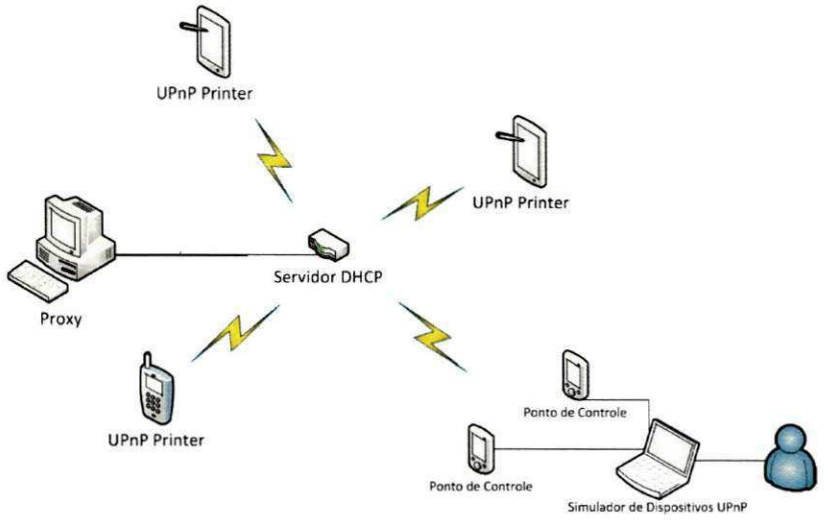


Figura 5.2: *Cenário de Teste 2*

Cenário 3 - Serviço UPnP *Media Renderer*

Para este caso, foram configurados dois pontos de controle requisitando o serviço de UPnP *Media Renderer*, com o objetivo de reproduzir um arquivo de música armazenado em um dispositivo *Media Server*. A seguinte configuração foi montada para este caso:

- Um *tablet* disponibilizando o serviço de *Media Renderer*;
- Dois dispositivos simulados pelo *laptop* também disponibilizando o serviço de *Media Renderer*;
- O *laptop*, simulando também um ponto de controle;
- O *smartphone* disponibilizando o serviço de *Media Server*;
- Um *tablet* atuando como ponto de controle.

A figura 5.6 mostra o esquema montado tanto para o caso de redes com suporte a gerenciamento de energia como redes não-gerenciadas. A diferença reside no fato de que em redes com suporte a baixo consumo, todos os dispositivos podem ser potenciais *proxies*.

A diferença entre este cenário e o usado como primeiro caso para redes infraestruturadas consiste no fato de que, quando utilizada a solução de gerenciamento, todos os dispositivos

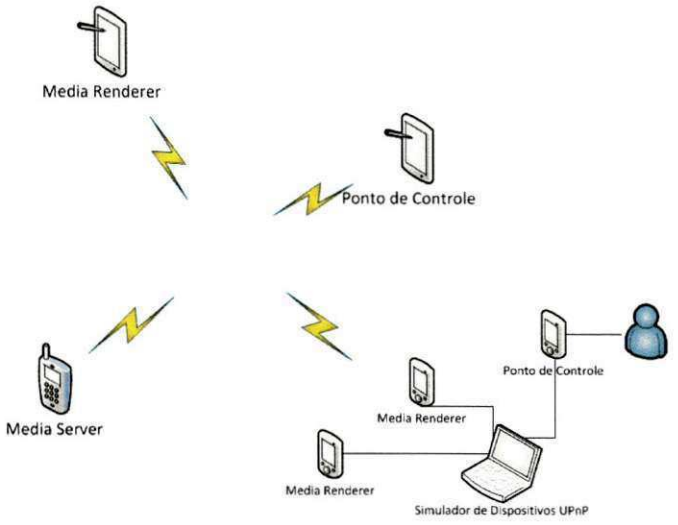


Figura 5.3: *Cenário de Teste 3*

(exceto os pontos de controle) implementam os serviços *LowPowerProxy* e *LowPowerManagement*, sendo todos eles potenciais *proxies*.

5.4 Execução dos Experimentos

Cada experimento consistiu em montar a rede UPnP utilizando as duas implementações do BRisa, uma com suporte a gerenciamento de energia e outra sem utilizar a solução combinada entre a especificação *Low Power Architecture* e o serviço *LowPowerManagement*. Foi avaliado o comportamento da rede com relação ao consumo a cada 3 minutos, para cada cenário.

Os pontos de controle foram configurados para realizar requisições periodicamente, com tempo variável entre cada requisição, até que fossem atendidos. Ao anunciar sua presença, os pontos de controle esperavam um tempo de 1 minuto para iniciar o ciclo de requisições. Ao final de cada invocação de serviço realizada com sucesso, o ponto de controle aguardava um tempo aleatório entre 1 e 3 minutos para efetuar outra requisição.

O consumo da rede foi estimado através de *logs* gerados sempre que um dispositivo transitava entre o estado *Active* e estados econômicos, para o caso de rede com suporte a gerenciamento de energia. No caso de rede UPnP sem solução de economia, foram armazenados os *timestamps* dos anúncios de entrada e saída dos dispositivos. Ao final de cada

execução, foi comparado o consumo total estimado da rede.

5.5 Resultados

Nesta seção, serão apresentados os resultados das execuções dos experimentos descritos anteriormente. Considera-se como unidade de consumo o kilowatt-hora e minuto como unidade de tempo.

5.5.1 Cenário 1

Após a execução do primeiro cenário, notou-se que nos primeiros minutos, ambas as redes se comportam de maneira similar. Isso acontece pelo fato de que os dispositivos que implementam os mecanismos de baixo consumo permanecem em estado ativo por algum tempo antes de entrar em modo econômico. A partir daí, nota-se a diferença entre os dois gráficos, como pode-se verificar na figura 5.4.

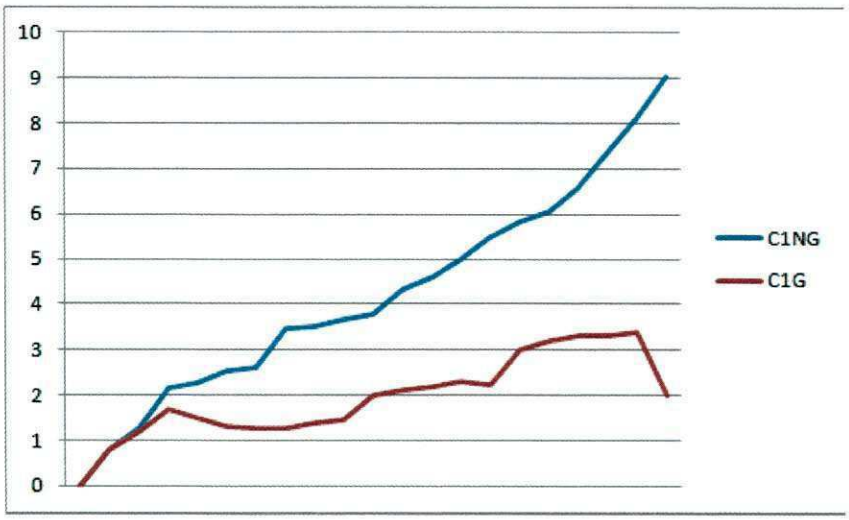


Figura 5.4: *Resultados Cenário de Teste 1*

Quando a rede UPnP já está estabelecida, para o caso onde não há gerenciamento de energia, C1NG, a curva de consumo cresce de uma forma pracicamente linear, enquanto que a rede gerenciada C1G tem o seu consumo reduzido pelo fato de todos os dispositivos ociosos entrarem automaticamente em um estado de economia. Mesmo assim, o consumo

continua crescente, pois o *proxy* permanece ativo durante todo o tempo.

Pode-se observar também alguns picos de consumo no gráfico da rede gerenciada. Isso acontece quando algum ponto de controle solicita ao *proxy* que um dispositivo seja "despertado" para atender uma requisição. Após a requisição atendida, o dispositivo volta ao modo econômico e o consumo é reduzido.

5.5.2 Cenário 2

Considerando o segundo cenário, mais uma vez verificou-se que nos primeiros instantes do estabelecimento da rede o consumo tanto para o caso de rede gerenciada quanto para não-gerenciada é bastante similar. O motivo desse comportamento é o mesmo observado no experimento anterior. A figura 5.5 mostra o resultado da execução do segundo cenário.

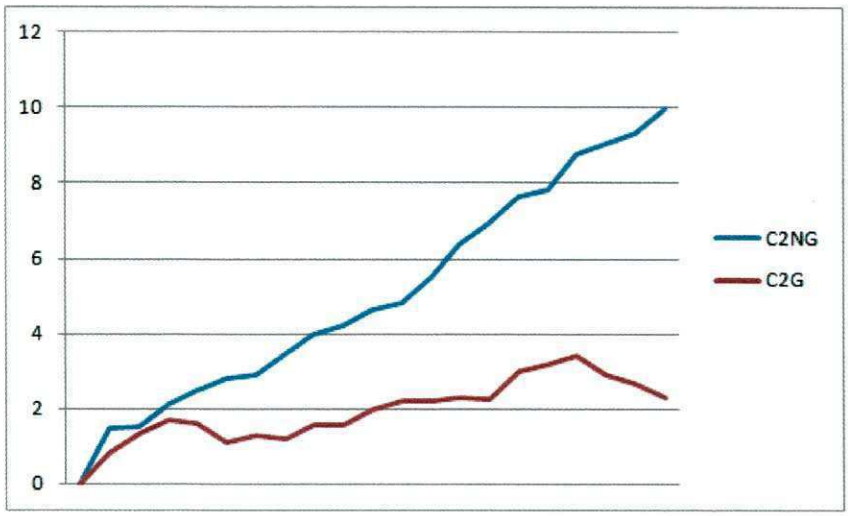


Figura 5.5: *Resultados Cenário de Teste 2*

Uma vez passado esse tempo inicial, observa-se mais uma vez que a rede não-gerenciada tem o seu consumo crescente a uma taxa praticamente constante, enquanto que a rede gerenciada apresenta um consumo geral menor com picos devidos às requisições atendidas.

5.5.3 Cenário 3

Este cenário, por se tratar de rede *ad hoc*, possui comportamento diferente quando gerenciado. Um dos picos de consumo ocorre quando se desconecta o *proxy* atual da rede. Sabe-se

que os potenciais *proxies* acordam periodicamente para evitar que a rede fique sem *proxy* por um grande período de tempo. Assim, quando todos acordam, o consumo tem um pico, voltando a estabilizar após a eleição do novo *proxy*.

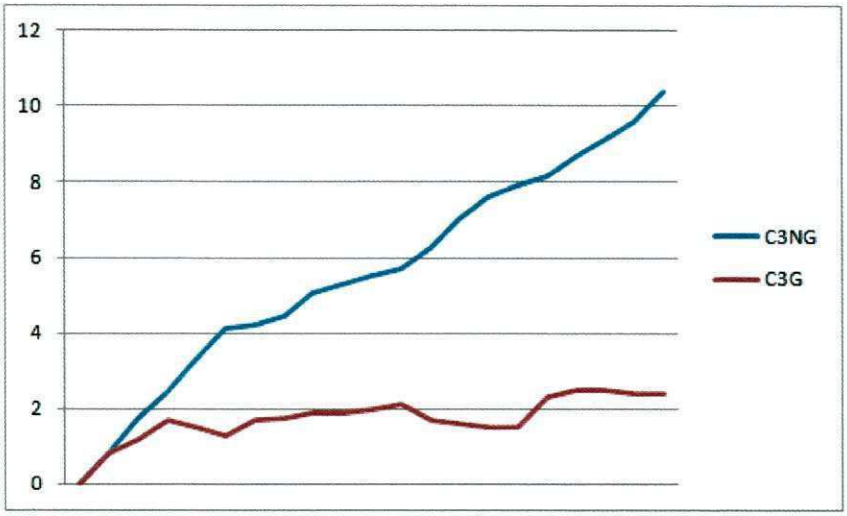


Figura 5.6: *Resultados Cenário de Teste 3*

Similarmente aos experimentos anteriores, outros picos de consumo foram verificados quando pontos de controle requisitam o serviço de *Media Renderer* ao *proxy* corrente.

Quando não gerenciado, o comportamento observado segue o que já foi visto no caso de redes infraestruturadas. Não há grandes diferença, pois os dispositivos permanecem ativos todo o tempo. A diferença nesse caso é somente em como os dispositivos adquirem o seu endereço IP, já que não há ponto de acesso presente.

5.6 Conclusão

Ao final dos experimentos, foi verificada uma tendência de redução no consumo de energia considerando os cenários gerenciados, em comparação com os casos de redes UPnP sem suporte ao método de gerenciamento de energia proposto no Capítulo 4.

Capítulo 6

Considerações Finais

No presente trabalho, foi proposta uma abordagem orientada a serviço para o gerenciamento de energia em redes pervasivas baseadas no padrão UPnP. Foi tomada como base a especificação *Low Power Architecture* do próprio padrão, por já prever certas características necessárias à solução, como por exemplo a implementação de estados de economia de energia pelos dispositivos. A solução baseia-se em manter possível a descoberta e ativação de serviços de um modo que se mantenha o maior número possível de dispositivos em modo econômico. Para isso, foi utilizado o conceito de *proxy*, que consiste de um dispositivo que está preferencialmente sempre ativo e tem ciência de quais nós da rede estão em estado de economia e quais serviços estes provêem. Procurou-se melhorar a escolha por dispositivos provedores de serviços, tentando sempre escolher o dispositivo mais "capaz" para disponibilizar aquele serviço naquele momento. Para isso, foi definida uma métrica de capacidade, usada pelo *proxy* para escolher dispositivos no momento que uma busca é feita por um dispositivo cliente, ou ponto de controle.

Com relação ao *proxy*, um dos principais questionamentos é a escolha pelo dispositivo que atuará como tal. Idealmente, o *proxy* deve ser um dispositivo que esteja sempre ativo, pois ele deve manter um registro de quais dispositivos de baixo consumo estão presentes e principalmente quais estão em estado de economia. Dois cenários de infraestrutura de rede foram considerados para a escolha do *proxy*. Em redes UPnP infraestruturadas, o *proxy* é um dos pontos de acesso obrigatoriamente presente nesse tipo de ambiente. Para o caso de redes UPnP totalmente *ad hoc*, é feito um processo de "eleição" distrubuída do *proxy* entre os nós candidatos.

A especificação *Low Power Architecture* por si só não provê uma solução para o problema exposto neste trabalho. Por isso, foi definido um novo serviço UPnP denominado *Low Power Management*, o qual usa as funcionalidades disponíveis na especificação e adiciona as suas próprias, como o cálculo da capacidade de cada dispositivo, além de tentar abstrair certas particularidades do padrão, sendo uma única interface entre os pontos de controle e os dispositivos controláveis. De fato, a própria arquitetura *Low Power* teve que ser ligeiramente alterada para que a presente abordagem fosse possível. Dados de consumo dos dispositivos quando ativos e em modo econômico não são originalmente incluídos no padrão. Essas informações são necessárias para o cálculo da capacidade de cada dispositivo de prover um determinado serviço em um dado momento.

Foi realizado também um estudo de caso, utilizando o *framework* BRisa Qt, uma implementação livre do padrão UPnP. Para que o estudo de caso fosse possível, alterações tiveram que ser feitas no código do *framework* as quais contemplam o suporte a dispositivos e *proxies Low Power* e também a implementação do serviço *LowPowerManagement* aqui definido. O estudo de caso foi feito com dispositivos UPnP simulados em dois computadores, um *smartphone* e um *internet tablet*. Resultados mostraram uma tendência de redução de consumo da rede quando utilizada a técnica de gerenciamento de energia aqui apresentada.

6.1 Contribuições

Apesar de o trabalho ter sido desenvolvido considerando um ambiente de rede UPnP, esta restrição não é estrita. A abordagem aqui apresentada pode ser aplicada em outros tipos de redes pervasivas, bastando para isso que entidades como os dispositivos de baixo consumo e o *proxy* possam ser mapeadas do especificado aqui para outros tipos de redes pervasivas.

O novo serviço UPnP *LowPowerManagement* permite que sejam usadas tanto as funcionalidades disponibilizadas pelo *proxy* como pelos dispositivos de baixo consumo com o objetivo de se conseguir um gerenciamento de energia realmente orientado a serviço. Pontos de controle, ao tomarem conhecimento da existência do *proxy* na rede, requisitam serviços ao invés de buscarem por dispositivos específicos. Fica a cargo do *proxy* descobrir qual dispositivo mais indicado para atender às requisições, baseado nas informações que já são providas pela especificação *Low Power Architecture*. Pretende-se que o serviço *LowPowerManage-*

ment seja incorporado à arquitetura *Low Power*. Para isso, será elaborado um documento denominado *Service Template*, sendo possível submetê-lo ao UPnP Forum como adição a uma possível nova versão da especificação, a qual contemplaria também a obrigatoriedade dos dados de consumo dos dispositivos no arquivo de informações de gerenciamento de energia que é disponibilizado por todo dispositivo que implementa o serviço *LowPowerDevice Service*.

Para que o estudo de caso fosse implementado, modificações foram necessárias no código do *framework* BRisa Qt, a saber:

- Suporte a dispositivos de baixo consumo - UPnP *LowPowerDevice Service*;
- Suporte à *proxy* de baixo consumo - UPnP *LowPowerProxy Service*;
- Suporte ao novo serviço *LowPowerManagement*

Com essas modificações, é possível detectar dispositivos de baixo consumo, ou seja, aqueles que implementam dois ou mais dos estados de economia previstos na especificação *Low Power*, bem como fazer uso das funcionalidades do *proxy*. Além disso, pode-se fazer uso do que foi definido no *LowPowerManagement Service*, bastando incorporá-lo ao dispositivo que oferece o serviço de *LowPowerProxy*. Essas modificações serão eventualmente incorporadas ao código do projeto e disponibilizadas no repositório do mesmo.

6.2 Trabalhos Futuros

Muitas são as possibilidades de trabalhos futuros a partir do que foi apresentado no presente trabalho. A definição de regras customizadas no *proxy* tornaria possível um monitoramento específico do consumo da rede, permitindo, por exemplo, o estabelecimento de limiares de consumo global e por serviço. Considerando uma rede doméstica, o uso de regras permitiria não somente otimizar o consumo de bateria dos dispositivos como também gerar uma economia real no consumo de energia da própria residência. A ciência de diversos contextos poderia ser usada para, por exemplo, que todos os equipamentos fossem mantidos desligados ou em *stand by* no período de tempo que o dono da casa não está presente, sendo todos ligados quando detectada a presença do mesmo, ou de pessoas previamente autorizadas. O

uso de regras poderia também fazer restrições quanto à utilização de um determinado dispositivo no provimento de algum serviço. Se uma impressora UPnP multifuncional está com seu *scanner* com defeito, mas imprimindo normalmente, pode ser desejado que o serviço de *scanner* seja excluído do cálculo de capacidade, tornando o dispositivo incapaz de prover aquele serviço naquele momento.

Com relação ao cálculo da capacidade, outras métricas poderiam ser levadas em consideração, como padrões de mobilidade [32], reputação baseada em desempenho, localização, entre outras. Atualmente só se consideram informações sobre a atual alimentação do dispositivo (rede elétrica ou bateria) no referido cálculo. Muitas outras informações contextuais poderiam ser usadas no cálculo da capacidade de um dispositivo de oferecer um certo serviço.

Outros estudos de caso devem ser realizados, a fim de avaliar o comportamento quando há muitos pontos de controle efetuando requisições por um único serviço. Além disso, situações de falhas inesperadas de dispositivos devem ser exploradas com mais profundidade. Experimentos com mais dispositivos (reais e/ou simulados) e mais serviços oferecidos ao mesmo tempo necessitam ser executados, também. Uma análise formal dos resultados de experimentos mais aprofundados pode ser feita para auxilar na validação do método aqui proposto.

O caso de redes UPnP *ad hoc* merece uma melhor investigação no que diz respeito à eleição do *proxy*. Falhas inesperadas do *proxy* corrente podem levar à situação onde todos os dispositivos estão adormecidos sem ninguém que "responda" por eles até que um potencial *proxy* "desperte" e assuma seu lugar. Um outro ponto crítico que foi identificado é o caso de falhas inesperadas dos dispositivos que estão em modo econômico. O *proxy* não tomará conhecimento até que tente "acordar" o dispositivo e não conseguir. Isso pode levar a inconsistências graves para o caso de serviços que são providos por apenas um dispositivo.

Por fim, pretende-se escrever um artigo científico abordando tudo o que foi exposto, e submetê-lo para um evento relevante. Com o *feedback* recebido, será possível melhorar e evoluir a solução, podendo a mesma servir de base para outros trabalhos que porventura se proponham a resolver o mesmo problema ou problemas similares.

Bibliografia

[1] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless mesh networks: a survey. *Comput. Netw.*, 47:445–487, March 2005.

[2] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, and Andrea Passarella. An adaptive and low-latency power management protocol for wireless sensor networks. In *Proceedings of the 4th ACM international workshop on Mobility management and wireless access*, MobiWac '06, pages 67–74, New York, NY, USA, 2006. ACM.

[3] Saisanthosh Balakrishnan and Jyothir Ramanan. Power-aware operating systems using acpi, 2001.

[4] Christian Becker, Gregor Schiele, Holger Gubbels, and Kurt Rothermel. Base "a micro-broker-based middleware for pervasive computing. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, PERCOM '03, pages 443–, Washington, DC, USA, 2003. IEEE Computer Society.

[5] Los Angeles Ca, Qing Wu, Massoud Pedram, and Xunwei Wu. Clock-gating and its application to low power design of sequential circuits. *Proc. of the IEEE Custom Integrated Circuits Conference*, 47:415–420, 2000.

[6] Antonio Capone, Maria Barros, Halid Hrasnica, and Spyridon Tompros. A New Architecture for Reduction of Energy Consumption of Home Appliances. In *TOWARDS eENVIRONMENT, European conference of the Czech Presidency of the Council of the EU e-Envi2009*, Prague, Czech Republic, 2009.

[7] Alberto Cerpa and Deborah Estrin. Ascent: Adaptive self-configuring sensor networks topologies. *IEEE Transactions on Mobile Computing*, 3:272–285, July 2004.

[8] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wirel. Netw.*, 8:481–494, September 2002.

[9] Kenneth Chiu, Madhusudhan Govindaraju, and Randall Bramley. Investigating the limits of soap performance for scientific computing. In *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, page 246, Washington, DC, USA, 2002. IEEE Computer Society.

[10] Schahram Dustdar and Wolfgang Schreiner. A survey on web services composition. *Int. J. Web Grid Serv.*, 1:1–30, August 2005.

[11] A.L.V. Guedes, D.F.S. Santos, J.L. do Nascimento, L.M. Sales, A. Perkusich, and H.O. Almeida. Brisa upnp a/v framework. In *ICCE 2008: Digest of Technical Papers. International Conference on Consumer Electronics*, pages 1–2. IEEE Computer Society, 2008.

[12] Insu Jeong, Hyunjun Choi, and Joongsoo Ma. Study on address allocation in ad-hoc networks. In *Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science*, ICIS '05, pages 604–609, Washington, DC, USA, 2005. IEEE Computer Society.

[13] Youn-Kwae Jeong, Intark Han, and Kwang-Roh Park. A network level power management for home network devices. *Consumer Electronics, IEEE Transactions on*, 54(2):487–493, May 2008.

[14] Miguel Jimeno and Ken Christensen. A network connection proxy to enable hosts to sleep and save energy, 2008.

[15] Christine E. Jones, Krishna M. Sivalingam, Prathima Agrawal, and Jyh Cheng Chen. A survey of energy efficient network protocols for wireless networks. *Wirel. Netw.*, 7:343–358, September 2001.

[16] Holger Karl. An overview of energy-efficiency techniques for mobile communication systems, 2003.

[17] Jong-Hoon Lee, Jung-Tae Kim, Eui-Hyun Paik, Intark Han, and Kwang-Roh Park. Design and implementation of a service oriented power saving system for the home network. *Computers in Education, International Conference on*, 0:1–2, 2009.

[18] Emerson Loureiro, Loreno Oliveira, and Hyggo Almeida. Improving flexibility on host discovery for pervasive computing middlewares. In *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, pages 1–8, New York, NY, USA, 2005. ACM.

[19] Elena Meshkova, Janne Riihijärvi, Marina Petrova, and Petri Mähönen. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Comput. Netw.*, 52:2097–2128, August 2008.

[20] Shivajit Mohapatra, Nikil Dutt, Alexandru Nicolau, and Nalini Venkatasubramanian. Dynamo: A cross-layer framework for end-to-end qos and energy optimization in mobile handheld devices. *IEEE Journal on Selected Areas in Communications*, 25(4):722–737, 2007.

[21] C. Siva Ram Murthy and B.S. Manoj. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[22] Luca Negri, Domenico Barretta, and William Fornaciari. Application-level power management in pervasive computing systems: a case study. In *Proceedings of the 1st conference on Computing frontiers*, CF '04, pages 78–88, New York, NY, USA, 2004. ACM.

[23] Joseph A. Paradiso and Thad Starner. Energy scavenging for mobile and wireless electronics. *IEEE Pervasive Computing*, 4:18–27, January 2005.

[24] Sachs, D. G., Yuan, W., Hughes, C. J., Harris, A., Adve, S. V., Jones, D. L., Kravets, R. H., Nahrstedt, and K. Grace: A hierarchical adaptation framework for saving energy, 2004.

[25] Gregor Schiele and Christian Becker. SANDMAN: an Energy-Efficient Middleware for Pervasive Computing. Technical report, University of Mannheim, Karlsruhe, Germany, October 2007.

[26] Gregor Schiele, Christian Becker, and Kurt Rothermel. Energy-efficient cluster-based service discovery for ubiquitous computing. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, EW 11, New York, NY, USA, 2004. ACM.

[27] T. Simunic. Power saving techniques for wireless lans. In *Proceedings of the conference on Design, Automation and Test in Europe - Volume 3*, DATE '05, pages 96–97, Washington, DC, USA, 2005. IEEE Computer Society.

[28] Tajana Simunic, Luca Benini, Peter Glynn, and Giovanni De Micheli. Dynamic power management for portable systems. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 11–19, New York, NY, USA, 2000. ACM.

[29] Andrew Tanenbaum. *Computer Networks*. Prentice Hall Professional Technical Reference, 4th edition, 2002.

[30] Spyridon Tompros, Nikolaos Mouratidis, Michael Caragiozidis, Halid Hrasnica, and Anastasius Gavras. A pervasive network architecture featuring intelligent energy management of households. In *Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*, PETRA '08, pages 75:1–75:6, New York, NY, USA, 2008. ACM.

[31] Spyridon Tompros, Nikolaos Mouratidis, Maurice Draaijer, Andreas Foglar, and Halid Hrasnica. Enabling applicability of energy saving applications on the appliances of the home environment. *Netwrk. Mag. of Global Internetwkg.*, 23:8–16, November 2009.

[32] Alicia Triviño Cabrera, Raúl Morales-Berrocal, and Eduardo Casilari. Simulation of realistic mobility patterns for mobile ad hoc networks. In *Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Computer Science - Volume 7*, pages 227–231, Stevens Point, Wisconsin, USA, 2007. World Scientific and Engineering Academy and Society (WSEAS).

[33] V. Tsaoussidis and H. Badr. Tcp-probing: towards an error control schema with energy and throughput performance gains. In *Proceedings of the 2000 International Confe-*

rence on Network Protocols, ICNP '00, pages 12–, Washington, DC, USA, 2000. IEEE Computer Society.

[34] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991.

[35] Yu Xiao, Ramya Sri Kalyanaraman, and Antti Ylä-Jääski. Middleware for energy-awareness in mobile devices. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, COMSWARE '09, pages 13:1–13:6, New York, NY, USA, 2009. ACM.

[36] Kuo-Qin Yan, Shu-Ching Wang, Mao-Lun Chiang, and Lin-Yu Tseng. A fuzzy-based power-aware management for mobile ad hoc networks. *Comput. Stand. Interfaces*, 31:209–218, January 2009.

[37] Ossama Younis and Sonia Fahmy. HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, 2004.

[38] C. Zhang and V. Tsaoussidis. Tcp-real: improving real-time capabilities of tcp over heterogeneous networks. In *Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '01, pages 189–198, New York, NY, USA, 2001. ACM.