Redes de sensores sem-fio sob a perspectiva do EPOS

- - Minicurso - -

Arliones Hoeller Jr , Antônio Augusto Fröhlich

[1]Universidade Federal de Santa Catarina
Laboratório de Integração Software/Hardware
Cx. Postal 476, 88040-900, Florianópolis-SC, Brasil
{arliones,guto}@lisha.ufsc.br

Abstract. *Redes de sensores sem-fio e outros sistemas pervasivos são cada vez mais comuns em nosso dia a dia. Esta tecnologia traz, junto das novas possibilidades de aplicações, um grande conjunto de desafios, o que deu margem ao surgimento de novas soluções de integração de componentes de processamento (CPU) e rádio, novos mecanismos de comunicação, variadas implementações de controle de acesso ao meio (MAC), sempre atendendo às rígidas restrições em termos de capacidade de processamento, potência de transmissão e, de modo especial, consumo de energia. Para tratar tudo isso, uma série de novos sistemas operacionais focou no domínio de redes de sensores sem-fio , abstraindo as funcionalidades de sensoriamento e comunicação, com o objetivo de agilizar o processo de desenvolvimento de aplicações. Este minicurso visa apresentar as principais características das redes de sensores sem-fio focando em temas importantes a serem considerados durante o desenvolvimento de aplicações, utilizando-se para isso dos trabalhos desenvolvidos para implementação do* EPOS *[Fröhlich 2001] e do* EPOSMOTE *[LISHA 2010c].*

1. Redes de sensores sem-fio

Avanços recentes nos projetos de dispositivos eletrônicos e a miniaturização levou ao surgimento de um novo conjunto de aplicações para computadores na forma de microsensores sem-fio de baixa potência. Estes microsensores são equipados com dispositivos de sensoriamento analógicos ou digitais (e.g., temperatura, campo magnético, som), um processador digital, um transceptor de comunicação sem-fio (e.g., rádio de baixa potência, infra-vermelho) e um módulo de alimentação (e.g., bateria, célula foto-sensível). Cada sensor, individualmente, é capaz de obter uma visão local de seu ambiente e de coordenar e se comunicar com outros sensores para criar uma visão global do objeto de estudo alvo da aplicação.

A ideia de uma rede auto-gerenciada composta por dispositivos autônomos que coletam e enviam dados através de um enlace sem-fio traz à tona uma série de novos desafios ao projeto das plataformas (hardware). Para que não sejam intrusivos e operem autonomamente por longos períodos de tempo, os nodos sensores precisam ser pequenos e consumir pouca energia. Além disso, nodos sensores precisam ser modulares e permitir uso de diferentes tipos de dispositivos sensores para permitir que uma única plataforma possa ser empregada nos diversos tipos de aplicações, podendo assim serem adaptados de acordo com as especificidades de cada aplicação. De modo similar, o hardware de comunicação deve permitir ampla configuração do canal de dados, possibilitando que diferentes aplicações se beneficiem de diferentes esquemas de modulação ou de controle

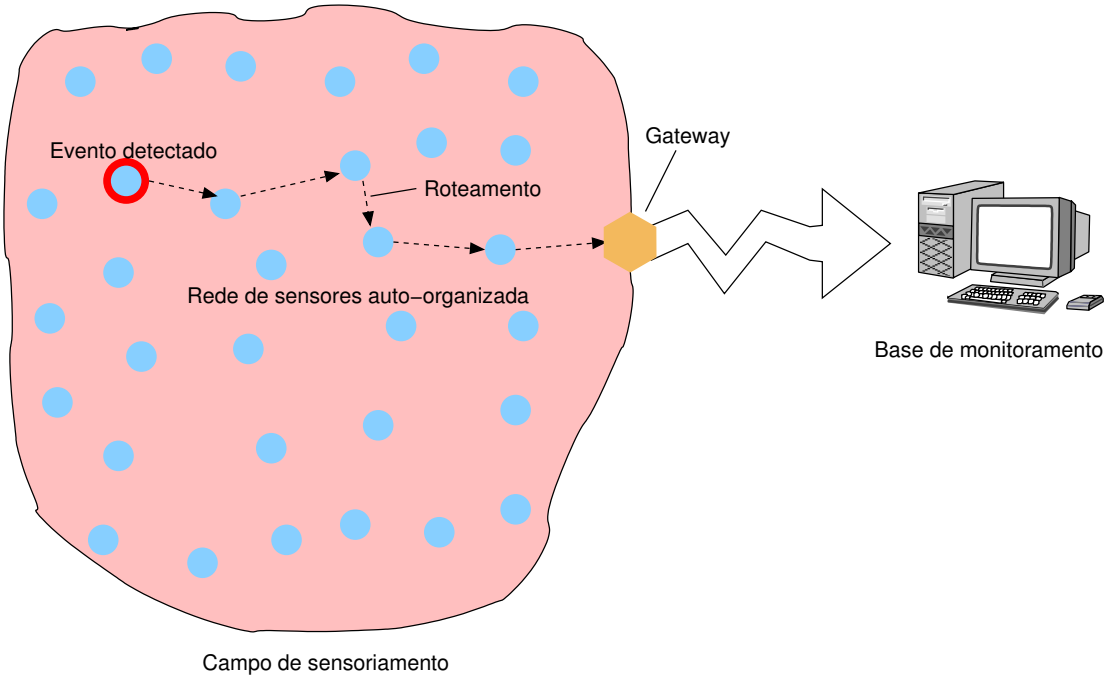

Figura 1. Redes de sensores sem-fio .

de acesso ao meio (MAC). Com o aumento da complexidade das tecnologias de redes de sensores sem-fio a necessidade de software para suporte à execução composto por sistemas operacionais e componentes abstratos de alto nível (e.g., middleware) se torna essencial.

Neste contexto, desde o ano de 2003 o LISHA (Laboratório de Integração Software/Hardware) vem trabalhando com redes de sensores sem-fio . Trabalhos desenvolvidos no LISHA no escopo do Projeto EPOS [LISHA 2010c] desenvolveram suporte de sistema operacional para a abstração dos mecanismos de:

- aquisição de dados [Wanner et al. 2006];
- comunicação [Wanner et al. 2007];
- gerência do consumo de energia [Hoeller et al. 2006a];
- outros serviços comuns de sistema operacional como alocação de memória e escalonamento de tarefas [Marcondes et al. 2006];

Mais recentemente, o LISHA passou a dedicar esforços no desenvolvimento de plataformas próprias de redes de sensores sem-fio , o que culminou com o projeto dos módulos de sensoriamento EPOSMOTE I, baseado em uma arquitetura AVR de 8 bits, em 2009 e EPOSMOTE II, baseado em uma arquitetura ARM7 de 32 bits, em 2010.

Este curso prevê uma revisão dos aspectos das tecnologias de redes de sensores sem-fio , buscando descrever as tecnologias e apresentar como estas foram tratadas tanto na implementação do sistema operacional EPOS quanto no projeto dos módulos de sensoriamento EPOSMOTE I e II. Para isso, esta primeira seção apresenta uma contextualização da tecnologia seguido de exemplos de aplicações das redes de sensores sem-fio . A seção 2 apresenta requisitos comumente procurados em módulos de redes de sensores sem-fio e apresenta alternativas de módulos disponíveis. A seção 3 apresenta tipos de sensores, métodos de interfaceamento e técnicas de aquisição de dados. A seção 4 revisa

Aplicação	Objetivo	Interação	Mobilidade	Espaço	Tempo
Monitoramento de habitat	AS	*-1	E	G	P
Resgate em avalanches	AS	*-*	E	R	P
Navegação de robô	AS	*-1	CM	R	BE
Controle de tráfego veicular	SR	*-*	E	R	P

Tabela 1. Exemplos de aplicações classificadas segundo a taxonomia proposta por Mottola [Mottola and Picco 2010].

técnicas de controle de acesso ao meio, apresenta topologias e descreve alguns dos MACs mais utilizados em redes de sensores sem-fio . A seção 5 apresenta características de sistemas operacionais para redes de sensores sem-fio . A seção 6 lista alguns exercícios para utilização do EPOS na plataforma EPOSMOTE .

1.1. As aplicações de redes de sensores sem-fio

Novas tecnologias, sistemas ou plataformas surgem, quase que exclusivamente, por um único motivo: atender a uma determinada demanda, ou seja, uma aplicação! No caso das redes de sensores sem-fio não poderia ser diferente. A criação das redes de sensores sem-fio foi motivada por aplicações militares como vigilância de campos de batalha e são hoje utilizados em diversas áreas de aplicações civis e industriais, incluindo monitoramento e controle de processos industriais, supervisão de maquinário, monitoramento de ambientes ou habitats, automação doméstica, entre outras.

Esta variada gama de aplicações implica na existência de grande variação no conjunto de requisitos que uma redes de sensores sem-fio precisa atender, incluindo requisitos possivelmente conflitantes, ou seja, características das quais um determinado tipo de aplicação se beneficia, eventualmente, pode tornar o uso da tecnologia proibitiva a outra aplicação. Para dar conta desta variabilidade, redes de sensores sem-fio precisam ser amplamente configuráveis. E, para ser configurável, um sistema de redes de sensores sem-fio precisa levar em consideração as características específicas dos diferentes tipos de aplicações existentes.

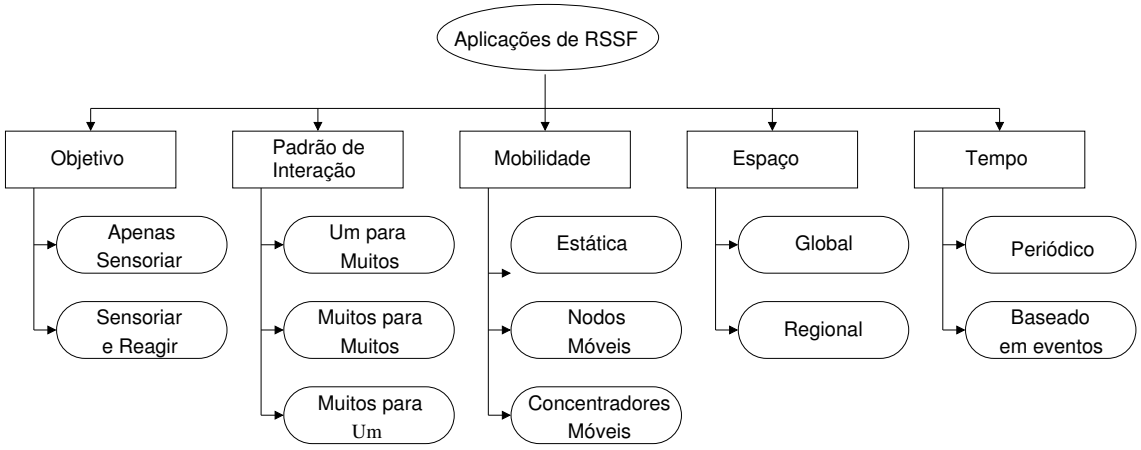

Figura 2. Uma taxonomia para aplicações de redes de sensores sem-fio [Mottola and Picco 2010]

Motolla e Picco [Mottola and Picco 2010] publicaram um excelente estudo em que, dentre outras coisas, classificaram diferentes aplicações de redes de sensores sem-fio

com o objetivo de destacar as diferenças entre estas aplicações. A Figura 2 apresenta uma taxonomia pela qual as características das aplicações podem ser classificadas. Neste trabalho, os autores ainda analisam uma série de aplicações publicadas em canais científicos, classificando-as segundo a taxonomia proposta. Alguns exemplos destas classificações estão na Tabela 1. Cada uma das características da taxonomia por eles proposta podem ser assim interpretadas:

- *Objetivo:* dependendo do objetivo das redes, isto é, apenas monitorar um objeto de estudo ou, além de monitorar, também atuar sobre este objeto, a topologia empregada pode sofrer modificações, especialmente devido à inclusão, neste último tipo de objetivo, de módulos atuadores capazes de interagir com o objeto de estudo.
- *Padrão de interação:* é, normalmente, dependente do *objetivo*. "Muitos-para-um", mais comumente empregado, é utilizado quando dados de diversos nodos são coletados por um nodo central. Comunicação "um-para-muitos" é normalmente utilizada para envio de comandos de configuração nas redes, e "muitos-para-muitos" é mais comum em situações onde há múltiplos concentradores de informação, o que geralmente ocorre na presença de atuadores.
- *Mobilidade:* em configurações "estáticas" nenhum elemento da rede se move após a implantação. Em configurações com "nodos móveis" alguns elementos das redes são móveis, como, por exemplo, em aplicações de monitoramento de habitats em que alguns nodos estão implantados em animais. Já em configurações com "concentradores móveis", normalmente, são indiferentes quanto à mobilidade dos demais nodos já que, neste caso, a diferença básica reside no fato de que a coleta de dados é realizada de modo oportunístico quando os concentradores se aproximam dos demais nodos.
- *Espaço:* diz respeito à semântica dos dados recolhidos. Configurações "globais" são aquelas em que dados de sensores individuais não são úteis, sendo úteis apenas as informações extraídas da análise das medições realizadas por todos os sensores em uma rede. Já configurações "regionais" são aquelas o objeto de interesse está localizado em uma região limitada, sendo, neste caso, leituras individuais relevantes.
- *Tempo:* do ponto de vista temporal a operação dos nodos em uma redes de sensores sem-fio pode ser classificada em "periódica", quando nodos realizam leituras de seus sensores e enviam os dados para processamento na rede periodicamente, ou "orientada a eventos", quando nodos sensores permanecem em modo quiescente observando os dados lidos por seus sensores e enviam dados na rede apenas quando um determinado evento é detectado (e.g., o valor lido em um determinado sensor ultrapassa um limite pré-definido).

2. Módulos de sensoriamento

Este capítulo aborda a arquitetura dos módulos de sensoriamento. Será discutida a arquitetura básica, comumente composta de um conjunto de sensores, um processador e um transceptor de rádio [Barr et al. 2002, Pottie and Kaiser 2000]. Serão apresentadas abordagens comerciais recentes de integração destes componentes em *single-package* ou em *single-die*. Também serão discutidos os requisitos buscados em um módulo de sensoriamento em termos de dimensões, consumo de energia, modularidade e adaptabilidade do canal de comunicação.

2.1. Arquitetura de módulos de sensoriamento

Em redes de sensores sem-fio , diversos nodos sensores, compostos por um conjunto de sensores analógicos e digitais, um microcontrolador, um transceptor sem-fios e bateria, coordenam-se e trocam informações de maneira a prover uma visão global de um dado objeto de estudo. Cada nodo individual possui capacidade limitada, mas a comunicação e processamento cooperativo na rede permitem a obtenção de dados mais precisos. Com base na pesquisa e aplicações atuais, e possível definir que a arquitetura básica de um nodo de sensor, composta por um microcontrolador e a um transceptor sem-fios deve [Wanner 2006]:

- Ter dimensões físicas reduzidas.
  Para poderem ser instalados de maneira não intrusiva, os nodos sensores devem ter dimensões reduzidas. Dado o constante avanço das técnicas de miniaturização de hardware, o tamanho dos componentes eletrônicos utilizados nos nodos tende a diminuir constantemente. Entretanto, a miniaturização dos nodos sensores pode estar limitada ao tamanho da fonte de energia (seja na forma de baterias ou dispositivos para captura de energia ambiente).
- Ser capaz de operar por um longo tempo com quantidade limitada de energia.
  A necessidade de operação autônoma de um nodo sensor, e a capacidade limitada de energia disponível ao mesmo, fazem com que o baixo consumo de energia seja um fator determinante no projeto de hardware. Sendo assim, o projeto de um nodo sensor priorizará componentes de baixa potência e com suporte a gerencia do consumo de energia (e.g., microcontroladores, transceptores de baixa potência) em detrimento de componentes direcionados a alta capacidade de processamento, desempenho ou potência.
- Ter um projeto modular, permitindo a conexão com sensores específicos para diferentes aplicações.
  Os serviços de uma rede de sensores tendem a ser específicos, e utilizar somente o hardware necessário aos requisitos de cada aplicação. Desta forma, é importante que o projeto seja modular, e permita a remoção e inclusão de sensores conforme as necessidades da aplicação.
- Permitir a mais ampla configuração possível do canal de transmissão de dados.
  O transceptor de dados sem-fios é, em geral, o componente com maior consumo de energia em um nodo sensor. Desta forma, é importante que este transceptor passe a maior parte do tempo desligado. Por outro lado, aplicações específicas terão padrões de comunicação específicos, e poderão se beneficiar de diferentes técnicas de modulação de dados e controle de acesso ao meio, que permitam o controle do consumo de energia sem comprometer a comunicação de dados. Desta forma, o transceptor deve permitir a maior configuração do canal de dados possível.

Para atender a esta série de requisitos, módulos de sensoriamento focaram no uso de componentes de baixa potência de pequeno tamanho. É o caso, por exemplo, da família módulos de sensoriamento Mica da Crossbow, Inc. [Crossbow 2005]. Estes *motes* (como o da Figura 3) são baseados em processadores AVR, da Atmel, Inc., que são processadores RISC de baixa potência. Além dos processadores AVR, estes dispositivos apresentam variados modelos de transceptores de rádio, como o caso do CC1000 no Mica2 e do CC2400 no MicaZ, ambos transceptores do fabricante Chipcom, Inc. O CC1000 é um dispositivo de rádio que opera com modulação FM (*Frequency Modulation*) nas faixas

Módulo	MicaZ	ZigBit	MC13224V
Encapsulamento	módulo	single-package	single-die
Processador	ATMEGA128L	ATMEGA1281V	ARM7TDMI
RAM	4 kB	8 kB	96 kB
Flash	128 kB	128 kB	128 kB
Potência de transmissão	0 dBm	3 dBm	20 dBm
Corrente Tx*	17.4 mA	18 mA	29 mA
Corrente Rx*	19.7 mA	19 mA	22 mA
Corrente *sleep*	15 μA	6 μA	0.85 μA
Dimensões (mm)	58 x 32 x 7	24 x 13,5 x 2	9,5 x 9,5 x 1,2
Preço (USD)	$ 144,00 (em 2005)	$ 18,25	$ 4,86

* Corrente drenada quando operando na potência máxima de transmissão.

Tabela 2. Comparação de características do MicaZ, ZigBit e MC13224V. Todos os três apresentam um rádio compatível com IEEE 802.15.4.

de frequência 315, 433, 868 e 915 MHz. Este dispositivo implementa apenas a camada física de um transceptor FM, sendo que a camada de enlace (MAC e LLC) são implementadas em software. A popularização do uso deste tipo de transceptor deu origem a uma série de protocolos de MAC criados para dar melhor suporte a certas categorias de aplicação. Já o CC2400 implementa as camadas física e de enlace (MAC) do padrão IEEE 802.15.4. Este padrão, embora diminua a flexibilidade no uso da camada física, permitiu o desenvolvimento de abstrações de mais alto nível estáveis, já que os desenvolvedores passaram a trabalhar sobre versões estáveis das camadas 1 (física) e 2 (enlace) da pilha de comunicação. É exemplo destes desenvolvimentos o consórcio $ZigBee^{TM}$, que implementa uma série de protocolos das camadas de 3 (rede) e 4 (transporte) para uso em dispositivos das chamadas PAN (*Personal Area Network*). Embora descrever a tecnologia $ZigBee^{TM}$ não seja objetivo deste minicurso, os diferentes MACs utilizados em redes de sensores sem-fio , incluindo o IEEE 802.15.4, são melhores descritos na seção 4.

Figura 3. Mica2 mote da Crossbow Inc.

Para atender as demandas por maior poder de processamento e maior potência de rádio em plataformas com menores dimensões e menor consumo de energia, os projetos modulares caminharam para a integração de microcontrolador, transceptor e outros componentes como antena e reguladores de tensão em uma abordagem *single-package*. Como exemplo pode ser citado o projeto $ZigBit^{TM}$ da MeshNetics [Meshnetics 2007].

Buscando um desempenho ainda maior, abordagens *single-die* permitem a integração de microcontrolador, rádio, e diversos outros dispositivos em um único circuito integrado, como, por exemplo, os dispositivos da família MC1322X da Freescale, ou os dispositivos STM32W da STMicroelectronics. A Tabela 2 mostra como dispositivos que representam cada uma das abordagens apresentam melhor desempenho junto com melhores características de tamanho e consumo de energia.

2.2. A arquitetura do EPOSMOTE

O projeto EPOSMOTE tem por objetivo desenvolver uma família de módulos de sensoriamento que permita ampla configurabilidade tanto da plataforma quanto do ambiente de software (sistema operacional). O EPOSMOTE , apresentado na Figura 4, foi concebido com um projeto modular, sendo previstos três módulos, entre os quais foram estabelecidas interfaces padrão, permitindo o uso intercambiável de diferentes versões dos módulos. A Figura 5 apresenta os três tipos de módulos, que são os seguintes:

Figura 4. O EPOSMOTE ao lado de uma moeda de uma libra.

- **Módulo de Base:** o módulo base incorpora as funcionalidades de processamento e de comunicação. O projeto EPOSMOTE desenvolveu duas versões deste módulo, uma utilizando o *single-package* $ZigBit^{TM}$ e outra utilizando o SOC (*single-die*) MC13224V, da Freescale. Características específicas de cada versão do módulo base são apresentadas abaixo. Este módulo deve implementar detalhes de do suporte a estes dispositivos, como a regulação de tensão e o dimensionamento da antena, além rotear os pinos dos dispositivos de modo a manter o padrão das interfaces de alimentação e de entrada e saída.
- **Módulo de Entrada e Saída:** no módulo de entrada e saída devem ser implementadas as interfaces necessárias de entrada e saída, podendo um novo módulo destes ser desenvolvido para cada aplicação que se pretende desenvolver, permitindo o emprego dos sensores ou atuadores desejados para uma aplicação específica. O projeto EPOSMOTE desenvolveu um módulo de entrada e saída ao qual deu o nome de *start-up board*. Esta placa incorpora uma interface USB, sensor de temperatura, acelerômetro de 3 eixos, alguns LEDs e botões [LISHA 2010c].
- **Módulo de Alimentação:** de modo a permitir o emprego de diferentes fontes de alimentação, uma interface de alimentação foi implementada. Módulos que se conectam a esta interface podem ser tão simples como uma bateria alcalina AA, ou tão complexas quanto um sistema com bateria de lítio recarregável ou com painéis solares. A interface de alimentação ainda disponibiliza uma interface I2C, permitindo que o módulo de alimentação se comunique com o de processamento. O projeto EPOSMOTE ainda não desenvolveu nenhum módulo de alimentação

específico, mas trabalhos em andamento estão explorando tecnologias de captação de energia utilizando esta interface.

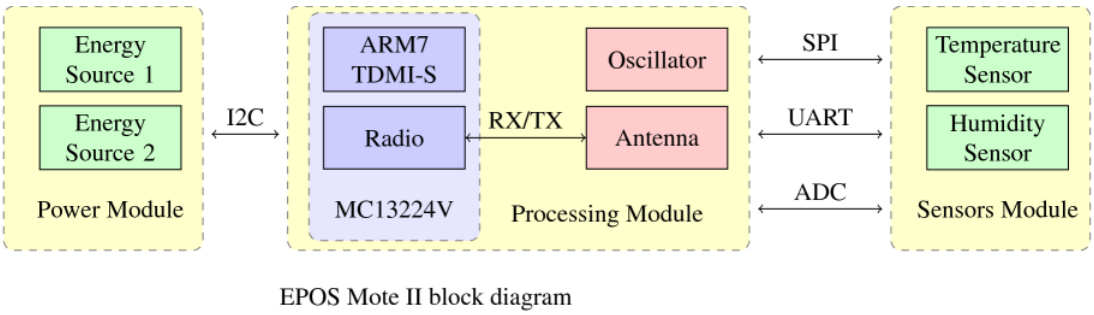

Figura 5. Arquitetura básica do EPOSMOTE

As interfaces padrão definidas pelo projeto EPOSMOTE para interconexão dos módulos desenvolvidos são as seguintes:

- **Interface de Entrada e Saída:** 32 pinos, sendo 2 para alimentação e outros 30 que podem ser utilizados ou como GPIO, ou com funções específicas que inclui ADC, UART e SPI.
- **Interface de Alimentação:** disponibiliza pinos aos quais devem ser conectados o terra e alimentação do módulo de alimentação. O módulo de processamento devolve ao módulo de alimentação o sinal com tensão regulada. Também existem os 2 pinos empregados na comunicação I2C.

3. Sensores e aquisição de dados

Um sensor é um dispositivo que responde a estímulos físicos (e.g., luz, temperatura, pressão, campo magnético ou movimento) e transmite um impulso resultante. Um sensor normalmente interage com um sistema digital, provendo informações sobre o mundo analógico. Essa informação pode ser fornecida através de um sinal analógico, que é convertido para valores digitais através de um conversor analógico-digital externo ao sensor, ou através de uma interface digital que converte internamente os sinais analógicos. Interfaces de dados de sensores podem variar amplamente mesmo dentro da mesma classe de sensores [Wanner 2006].

3.1. Tipos de sensores

Devido ao pequeno porte e às suas limitações em termos de disponibilidade de energia, módulos de redes de sensores sem-fio normalmente empregam sensores eletrônicos ou óticos com relativa eficiência energética. É o caso de, por exemplo, termistores para medir temperatura, foto-didos para medir luminosidade ou acelerômetros para medir deslocamento.

De maneira a possibilitar um melhor entendimento dos desafios envolvidos no projeto e implementação de um sistema de aquisição de dados de sensores, esta seção apresenta alguns dispositivos sensores usados em nodos de sensor contemporâneos. Esta não é uma lista exaustiva, mas deve prover um caso geral de uso de dispositivos sensores em redes de sensores sem-fio .

3.1.1. Termistores

Um termistor é um resistor cuja resistência varia com mudanças de temperatura. A equação de Steinhart-Hart é uma aproximação de terceira ordem amplamente utilizada para determinar a curva de resposta de um termistor:

$$T = \frac{1}{a + b.lnR_t + c.(lnR_t)^3} \tag{1}$$

onde a, b, e c são parâmetros Steinhart-Hart específicos para cada termistor, T é a temperatura Kelvin, e R_t é a resistência em Ohms apresentada pelo termistor na temperatura atual. Um termistor normalmente é ligado a um conversor analógico-digital através de um circuito divisor de tensão simples. A estimativa de temperatura baseada em leituras do ADC pode depender do cálculo em tempo de execução de funções de aproximação complexas (e.g., a equação Steinhart-Hart), ou pode fazer uso de tabelas de conversão previamente calculadas. Termistores diferentes podem ter constantes de tempo e precisão diferentes, bem como diferentes constantes de dissipação de energia.

3.1.2. Sensores digitais de temperatura

A família SHT1x [Sensirion 2005] de sensores de umidade e temperatura provê um exemplo de sensores digitais de temperatura usados em redes de sensores sem-fio . O sensor é fabricado pela Sensirion, e provê leituras digitais calibradas por uma interface SPI de dois fios. Um microcontrolador pode e ler dados do sensor enviando um comando específico solicitando uma medida temperatura para o sensor pela interface SPI. Ao completar da leitura, o sensor envia um sinal de dados prontos. O microcontrolador pode, então, ler os dados acompanhados de um código CRC para validação. Depois dessa transmissão, o sensor entra em modo inativo. Os coeficientes de calibração são programados na memória interna do sensor, e são usados internamente durante as leituras para calibrar os sinais lidos. Dados sensoriais providos pelo SHT1x podem ser convertidos para valores de temperatura através de uma e função linear. Um registrador de status provê uma interface de detecção de baixa tensão, configura a resolução da leitura (por exemplo, 8 bits, 16 bits) e controla um aquecedor interno.

3.1.3. Foto-resistores e foto-diodos

Um foto-resistor é um componente eletrônico cuja resistência diminui com o aumento da intensidade de luz incidente. A maioria dos foto-resistores são implementados através de células de sulfeto de cádmio, que usam a habilidade desse material de variar sua resistência (por exemplo, apresentando $2k\Omega$ em condições de baixa luminosidade e 500Ω quando exposto à luz). Essas células também são capazes de reagir a uma ampla gama de frequências, incluindo infravermelho, luz visível ultravioleta. Como no caso dos termistores, foto-resistores normalmente fazem interface com um conversor analógico-digital através de um circuito divisor de tensão.

Um foto-diodo é um semicondutor que responde a estímulo óptico. Foto-diodos operam pela absorção de fótons que geram uma variação na corrente que flui através

deles. Um circuito RC (resistor-capacitor) auxiliar tem sua fonte de corrente afetada por este sensor, o que permite a detecção da presença ou ausência de quantidades diminutas de luz através de medidas de tensão no resistor desse circuito auxiliar.

3.1.4. Sensores digitais de luminosidade

O sensor TSL2550 [TAOS 2005], fabricado pela TAOS, provê um exemplo de sensor digital de luminosidade usado em redes de sensores sem-fio . Ele combina dois foto-diodos e um conversor analógico-digital de 12 bits num circuito integrado para prover medidas de luz com uma sensitividade parecida com a do olho humano. Um dos foto-diodos é sensível a luz visível e infravermelha, enquanto o segundo foto-diodo é sensível primariamente a luz infravermelha.

Dados de sensoriamento são lidos através de uma interface de dois fios SMBus. Um registrador de controle gere o dispositivo, e dois registradores de ADC armazenam os dados de para leitura. As saídas dos dois canais ADC podem ser usadas em uma função linear para se obter um valor que aproxima a resposta do olho humano na unidade comumente utilizada de Lux.

3.1.5. Magnetômetros

Magnetômetros são sensores capazes de medir força e/ou direção de campos magnéticos. Magnetômetros podem ser divididos em magnetômetros escalares, que medem a força total do campo magnético ao qual eles são expostos, e magnetômetros de vetor, que têm a capacidade de medir a componente do campo magnético em determinada direção. Um conjunto de magnetômetros de vetor podem ser combinados para permitir a definição de força, declinação e inclinação de um campo magnético. A família Honeywell HMC100x [Honeywell 2005] de sensores magnetoresistivos são dispositivos de pontes resistivas simples que requerem apenas uma fonte de tensão para medir campos a magnéticos. Esses dispositivos são capazes de medir qualquer campo magnético ambiente ou aplicado em um eixo sensível. A tensão de saída do sensor é linear em relação ao campo magnético aplicado.

3.1.6. Acelerômetros

Acelerômetros são usados para medir mudanças na velocidade. Na sua forma mais simples, um acelerômetro é composto por uma massa suspensa e um dispositivo sensível à deflexão. A família Analog Devices ADXL345BCCZ-RL7 [Analog-Devices 2009] provê um exemplo de acelerômetros com 2 eixos e baixo consumo de energia. O sensor provê tanto saídas analógicas quanto digitais. O valor de saída do sensor é linear em relação à aceleração aplicada, e calibração para baixas gravidades podem usar o campo gravitacional da Terra como referência.

3.2. Calibração de sensores

Dados lidos diretamente de sensores precisam ser calibrados para garantir um determinado nível de qualidade. Imprecisões nas medições podem surgir por diversas

razões [Albertazzi and de Sousa 2008]:

- **Definição do mensurando:** diz respeito ao conhecimento disponível acerca da grandeza física que é objeto de estudo.
- **Procedimento de medição:** diz respeito aos procedimentos utilizados para realizar a medição. Neste caso se aplica, por exemplo, a precisão dos modelos matemáticos sendo utilizados para a medição de uma determinada grandeza.
- **Condições ambientais:** diz respeito a condições que podem interferir no resultado de uma medição, como por exemplo, interferência eletromagnética ou de pressão atmosférica sobre o sistema de medição.
- **Sistema de medição:** diz respeito a interferência gerada no resultado de uma medição pelos equipamentos utilizados no processo. Esta interferência pode vir, por exemplo, da incerteza do sensor utilizado ou erros de arredondamento na conversão analógico-digital.
- **Operador:** diz respeito a erros que possam ser ocasionados pela ação humana quando o procedimento de medição depende da ação de um operador. Normalmente, este fator tem pouco efeito em aplicações de redes de sensores sem-fio , já que estas aplicações tendem a operar de modo autônomo.

Estes fatores agem sobre o sistema formado pelo mensurando (objeto de estudo) e pelo sistema de medição que, no caso das redes de sensores sem-fio , é formado pelo sensor escolhido, pelo conversor analógico-digital ou pelo microcontrolador no caso de um sensor digital. O que resulta destas interferências é uma variação no valor indicado. Para tornar as leituras mais confiáveis é necessário, de algum modo, compensar o efeito destas interferências.

As interferências geradas por estes agentes no processo de medição podem ser agrupados de modo a formar duas variáveis de erro: o erro sistemático e o erro aleatório. O erro sistemático é aquela parcela do erro que se mantém constante entre várias medições. Para permitir uma correção das medidas, é possível definir uma estimativa do erro sistemático chamada de *Tendência*. A *Tendência* pode ser determinada através de medições sucessivas de um mensurando com valores conhecidos. Por exemplo, para calibrar um sensor de temperatura, pode-se realizar leituras sucessivas deste sensor em um ambiente com temperatura constante conhecida. A *Tendência* é dada, neste caso, pela diferença entre a média das leituras realizadas e o valor do mensurando. Este procedimento é conhecido calibração do sensor.

Não há meio de se estimar o valor exato do erro aleatório, mas é possível, através de tratamento estatísticos de uma série de amostras realizadas sobre um mensurando com valores conhecidos, determinar a repetitividade do sistema de medição e, consequentemente, seu erro máximo. Do ponto de vista de redes de sensores sem-fio é importante analisar o erro máximo do sistema definido para verificar sua adequação à aplicação que se pretende desenvolver. Outra característica importante do erro aleatório é que a soma dos erros aleatórios de medidas sucessivas de um mesmo mensurando tende a se anular no infinito. Logo, a média de sucessivas medições pode diminuir o efeito deste erro, gerando resultados mais confiáveis.

3.3. Sensores no EPOSMOTE

Como apresentado na Seção 2.2, o EPOSMOTE apresenta uma interface padrão para um módulo de entrada e saída que deve ser utilizada para conectar os sensores a serem uti-

lizados pela aplicação em questão. No atual estágio de desenvolvimento, o EPOSMOTE conta com a *startup board*, um módulo de entrada e saída que, além de componentes como interface USB, botões e LEDs, conta com um sensor de temperatura do tipo termistor, modelo ERT-J1VG103FA [Panasonic 2004], e um acelerômetro digital de 3 eixos, modelo ADXL345BCCZ-RL7 [Analog-Devices 2009]. A fim de exemplificar o processo de calibração de um sensor, vamos aqui demonstrar como foi realizada a calibração do sensor de temperatura da *startup board* do EPOSMOTE .

O mediador[1] do sensor de temperatura foi implementado utilizando a Equação de Steinhart-Hart (1). Os valores dos parâmetros de Steinhard-Hart podem ser calculados [Steinhart and Hart 1968] a partir de dados fornecidos pelo fabricante do termistor, e são substituídos na Equação 1 de modo a montar o modelo matemático do sensor em uso:

$$a = 1,0750492 \times 10^{-3} \tag{2}$$

$$b = 0.27028218 \times 10^{-3} \tag{3}$$

$$c = 0.14524838 \times 10^{-6} \tag{4}$$

Para efetuar este cálculo, é necessário conhecer a resistência apresentada pelo termistor. Como o termistor está conectado ao ADC por um circuito divisor de tensão a resistência pode ser calculada da seguinte forma:

$$V_s = I.(R + R_t) \tag{5}$$

$$V_o = I.R \tag{6}$$

$$com\ I\ constante\ no\ divisor,\ logo,\ \frac{V_s}{R + R_t} = \frac{V_o}{R} \tag{7}$$

$$isolando\ R_t,\ R_t = R.\left(\frac{V_s}{V_o} - 1\right) \tag{8}$$

onde V_s é a tensão de entrada do circuito de divisão de tensão, I é a corrente que passa pelas resistências (igual tanto sobre o resistor do divisor de tensão quanto sobre o termistor), R é a resistência conhecida do resistor do divisor de tensão, V_o é a tensão de saída do divisor, que é lido pelo conversor analógico-digital, e R_t é o valor atual da resistência do termistor, que se deseja obter.

Ao final, substituindo o valor da resistência atual do termistor definido pela Equação (8), e os parâmetros de Steinhart-Hart definidos em (2), (3) e (4) na Equação (1) (Steinhart-Hart), e ainda, sabendo que o valor do resistor utilizado no circuito divisor de tensão é de $10\ k\Omega$, podemos chegar à equação para o cálculo da temperatura amostrada pelo termistor estudado.

A calibração do sensor de temperatura foi realizado por comparação com outro sensor de temperatura, modelo SHT-11 [Sensirion 2005], previamente calibrado. Ambos os sensores foram conectados a um EPOSMOTE , e os dois EPOSMOTE foram colocados em uma caixa hermeticamente fechada. Os EPOSMOTE utilizados tiveram seus timers configurados igualmente, e foram conectados a uma mesma fonte de alimentação. Eles foram programados para realizar 1000 medidas de temperatura espaçadas de 10 segundos

[1]Mediador é o artefato de software que realiza interface do sistema operacional com o hardware no Projeto EPOS

cada e armazená-las em sua memória interna. Ao final das medições, as leituras foram enviadas por rádio para uma estação-base, de onde a tendência do sistema foi extraída. A correção, definida como o inverso da tendência ($C = -Td$), foi então inserida no sistema, ficando a equação final de temperatura do termistor assim definida:

$$T = \left\{ \frac{1}{a + (b \times ln\, R_t) + [c \times (ln\, R_t)^3]} \right\} + C \tag{9}$$

com a indicação T expressa em Kelvin. A implementação deste modelo, na linguagem C++, para o sistema operacional EPOS , é apresentada na Figura 6.

É importante destacar que as arquiteturas utilizadas no EPOSMOTE , AVR e ARM7-TDMI-S, não apresentam unidade de ponto flutuante (FPU), logo, o sistema carece de software adicional para executar as operações em ponto fixo, chegando a resultados semelhantes, porém com muito mais demanda de processamento. Por isso, é comum utilizar, em algumas aplicações, versões de mediadores para o termistor que utilizem tabelas pré-calculadas de conversão das leituras do ADC para um valor equivalente de temperatura, ao custo, neste caso, de um consumo extra de espaço de armazenamento. De qualquer modo, o processo de calibração descrito acima continua sendo necessário, neste caso, para a construção da tabela que ficará armazenada na memória do nodo.

3.4. Abstrações de sensores no EPOS

Do ponto de vista do programador da aplicação, abstrair questões específicas como o procedimento de calibração de sensores descrito acima é de grande importância. Neste contexto, o EPOS fornece suporte de sensoriamento às aplicações através de uma interface de software/hardware que abstrai famílias de sensores de forma uniforme [Wanner et al. 2006]. O sistema define classes de dispositivos baseado em sua finalidade (e.g. medir aceleração ou temperatura), e estabelece um substrato comum para cada classe. Para cada dispositivo são armazenadas propriedades e parâmetros operacionais, de maneira similar ao TEDS (*Transducer Electronic Data Sheet*) do padrão IEEE 1451. Uma camada fina de software adapta dispositivos individuais (e.g., a converte leituras de ADC em valores contextualizados, aplica as correções) para adequá-lo às características mínimas da sua classe de sensores. Desta forma, um termistor simples é exportado para a aplicação exatamente do mesmo modo que um sensor de temperatura digital complexo [Wanner and Fröhlich 2008].

A Figura 7 apresenta um diagrama de classes com as abstrações de sensoriamento do EPOS . No subsistema de sensoriamento do EPOS , métodos comuns a todos dispositivos de sensoriamento são definidos pela interface `Sensor_Common`. O método `get()` provê leituras para um único sensor em um único canal (i.e. habilita o dispositivo, espera os dados estarem disponíveis, lê o sensor, desabilita o dispositivo e retorna a leitura convertida em unidades físicas previamente determinadas). Os métodos `enable()`, `disable()`, `data_ready()` e `get_raw()` permitem que o sistema operacional e as aplicações realizem controle de grão fino sobre leituras de sensores (e.g., realizar leituras sequenciais, obter dados não convertidos de sensores). O método `convert(int v)` pode ser utilizado para converter valores não processados de sensores em unidades científicas ou de engenharia. O método `calibrate()` executa calibragem específica para cada sensor.

```
class ADC {
// ...

static const unsigned int div_resistor = 10000; // 10 kohm
static const unsigned int adc_max_value = 1023; // 10-bits ADC

public:
    float getResistance() {
        return div_resistor * ((float)adc_max_value / this->read() - 1);
    }

// ...
};

class Thermistor {
// ...

static const float A = 0.0010750492;
static const float B = 0.00027028218;
static const float C = 0.00000014524838;
static const float Correction = 2.37;
static const float Kelvin_to_Celsius = -273.15;

public:

float sample() {
    float logR = logf(adc.getResistance());
    float T = (1 / (A + B*logR + C*logR*logR*logR)) + Correction;
    if (mode == CELSIUS) return T + Kelvin_to_Celsius;
    else return T;
}

};
```

Figura 6. Código para implementação do termistor calibrado em C++ para o EPOS .

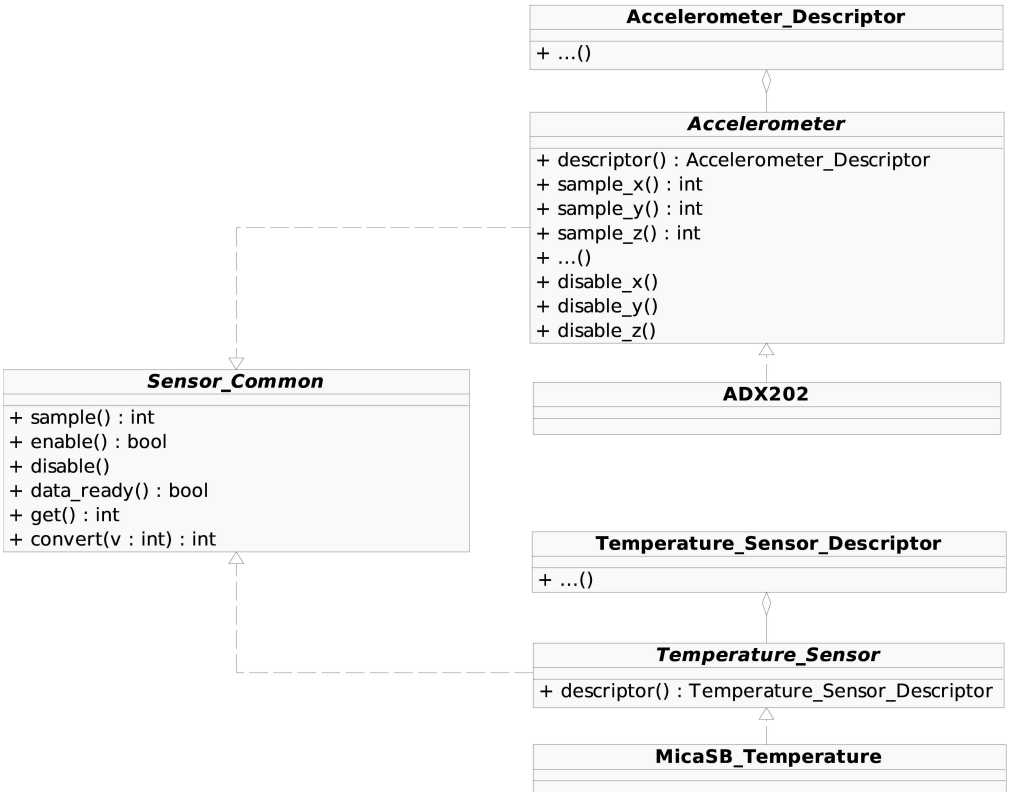

Figura 7. Diagrama de classes das abstrações de sensoriamento do EPOS .

Cada família de sensor pode especializar a interface `Sensor_Common` para abstrair adequadamente características específicas da família. A família `Magnetometer` pode adicionar, por exemplo, métodos para realizar leituras em diferentes eixos de sensibilidade. A família `Thermistor`, por outro lado, provavelmente não precisará estender a interface comum. Cada família também define uma estrutura `Descriptor` específica, que define campos como precisão, dados para calibração e unidades físicas. Cada dispositivo sensor implementa uma das interfaces definidas, e preenche a estrutura `Descriptor` da família com valores específicos do sensor. Valores padrão de configuração para cada dispositivo (e.g., frequência, ganho, etc.) são armazenados em uma estrutura de *traits de configuração*.

Sempre que o sistema operacional ou uma aplicação precisam fazer referência a um dispositivo de sensoriamento, estes podem utilizar um *dispositivo específico* e realizar operações específicas do dispositivo, ou utilizar a *classe do dispositivo*, e restringir-se às operações definidas por aquela classe. Uma realização utilizando metaprogramação estática da classe do dispositivo agrega os dispositivos disponíveis em uma configuração do sistema.

A Tabela 3 apresenta características de tamanho de código e dados do subsistema de sensoriamento do EPOS em comparação com as estruturas equivalentes dos sistemas TinyOS e MANTIS OS. A Tabela 4 mostra as taxas máximas de amostragem possíveis no EPOS , também em comparação com os outros sistemas. O baixo sobrecusto e alta taxa de amostragem no EPOS são resultado direto do projeto do sistema, que minimiza dependências entre componentes de sensoriamento e o resto do sistema. No EPOS , um

	TinyOS		MANTIS OS		EPOS	
	Código	Dados	Código	Dados	Código	Dados
System	10188	455	25500	596	7046	213
AVR ADC	550	4	538	9	64	3
ADXL202	722	4	936	10	266	9
Thermistor	1366	12	1050	11	1064	3
Photocell	1366	12	1050	11	1064	3
HMC1002	748	7	910	10	246	9

Tabela 3. Tamanhos de código e dados de componentes de sensoriamento (em bytes).

Sensor	TinyOS	MANTIS OS	EPOS
AVR ADC	8084	3685	24597
ADXL202	7657	3401	21711
Thermistor	5766	3107	10999
Photocell	6009	3117	11121
HMC1002	7494	3408	23024

Tabela 4. Taxas de amostragem dos sistemas de sensoriamento (em Hz).

componente que abstrai um sensor analógico normalmente depende apenas do conversor analógico-digital da plataforma e do seu subsistema de I/O, que são abstraídos por operadores metaprogramados *inline*. Estes mecanismos envolvidos no desenvolvimento do EPOS serão melhores descritos na Seção 5.1.

4. Controle de acesso ao meio (MAC)

Rádios de banda estreita suportam, tipicamente, esquemas simples de modulação, deixando a cargo do software o controle da comunicação, o que incorre em maior custo de processamento. Por outro lado, rádios de banda larga empregam técnicas de modulação sofisticadas, como *Direct-Sequence Spread Spectrum* (DSSS) e *Phase Shift Keying* (PSK), que são mais resistentes a ruído e interferências, mas apresentam pouca flexibilidade e impõem sobrecustos em termos de consumo de energia. Neste contexto, rádios de baixa potência se apresentam como uma alternativa viável para comunicação sem-fio para sistemas embarcados como redes de sensores sem-fio .

Nesta seção faremos uma breve revisão de técnicas de controle de acesso ao meio para redes sem-fio. Também serão apresentadas alternativas existentes de implementações de MACs específicos para aplicações de redes de sensores sem-fio .

4.1. Protocolos de acesso ao meio

Redes de sensores sem-fio apresentam um canal de comunicação único e compartilhado em que duas ou mais transmissões podem ocorrer simultaneamente, gerando interferência e invalidando a comunicação. Para viabilizar a comunicação neste meio, um *protocolo de múltiplo acesso* se faz necessário. Este tipo de protocolo implementa um algoritmo distribuído de controle de acesso ao meio (MAC - *Medium Access Control*) qeu determina como as estações compartilha o canal, ou seja, determinam quando uma estação pode iniciar uma transmissão.

Abordagens de MAC para controle de acesso múltiplo podem, geralmente, ser classificados em três classes:

- **Particionamento de canal:** estas abordagens dividem o canal disponível e aloca uma parte do canal para uso exclusivo de um determinado nodo. Exemplos deste tipo de abordagem é o FDMA, que divide o canal em faixas de frequência, e o TDMA, que divide o canal em fatias de tempo.
- **Acesso aleatório:** estas abordagens permitem a ocorrência de colisões. Diferentes abordagens constituem diferentes modelos pelos quais é possível, ou não, identificar, recuperar, evitar, ou eliminar colisões.
- **Passagem de permissão:** nestes protocolos, o compartilhamento do canal é estritamente coordenado para evitar colisões. Normalmente implementam técnicas de passagem de ficha (*token-passing*).

É comum que implementações de protocolos de controle de acesso ao meio utilizem mais de uma destas técnicas em conjunto, permitindo uma melhor exploração do canal disponível e oferecendo diferentes características de transmissão de dados às aplicações que o utilizam.

4.1.1. TDMA - Time Division Multiple Access

No TDMA, dispositivos compartilham o mesmo canal de frequência pela divisão do sinal em fatias de tempo (*time slots*). Neste protocolo, cada nodo de uma rede transmite, exclusivamente, dentro da fatia que foi previamente alocada a ele. Esta tecnologia é utilizada, por exemplo, em sistemas 2G de telefonia celular digital, como no GSM. O emprego de *time slots* também é bastante explorada em redes de sensores sem-fio , especialmente em aplicações que querem evitar contenção de dados e colisões. Uma abordagem TDMA é utilizada pelo MAC do padrão IEEE 802.15.4 para em seu período livre de contenção (CFP - *Contention-Free Period*).

Uma característica peculiar das abordagens TDMA diz respeito a necessidade de manter a sincronização dos nodos na rede de modo a garantir que todos concordam com as bordas das fatias de tempo do protocolo. No padrão IEEE 802.15.4 esta sincronização é realizada através de uma técnica chamada *beaconing*, em que um nodo "mestre" emite, periodicamente, *beacons*, pacotes especiais que contêm a configuração do canal (e.g., quais fatias de tempo estão alocadas para quais nodos), e também são utilizados como baliza para determinar a borda de início de um período de transmissão no protocolo.

4.1.2. FDMA - Frequency Division Multiple Access

No FDMA, nodos compartilham um canal dividindo o espectro do canal em bandas de frequência. Através desta divisão, cada banda de frequência pode ser alocado a um único nodo, ou a um conjunto diferente de nodos, eliminando ou, ao menos, diminuindo a ocorrência de contenção. A divisão do espectro de frequência em vários canais é bastante utilizado em redes sem-fio, inclusive sendo seu uso previsto nos padrões IEEE 802.11 (WiFi) e IEEE 802.15.4.

Um dos principais problemas relacionados ao FDMA é o *crosstalk*. Este fenômeno ocorre em algumas situações em que o sinal transmitido em uma banda de frequência interfere no sinal de uma banda de frequência adjacente. Este problema é geralmente causado por erros ou na emissão do sinal no transmissor ou na filtragem no

receptor, e geralmente é minimizado com o emprego de dispositivos moduladores e demoduladores de melhor qualidade.

4.1.3. ALOHA e *slotted* ALOHA

O protocolo ALOHA (também referenciado como ALOHA Puro, ou *Pure ALOHA*) foi implementado como protocolo de controle de acesso ao meio da ALOHAnet, que foi a primeira demonstração de uma rede sem-fio de dados [Abramson 1970]. O protocolo original (Puro) é bem simples: se um nodo tem dados a enviar, ele envia; se há uma colisão, o nodo tenta novamente no futuro. No ALOHA, portanto, não há verificação de sinal ocupado. Outro problema do protocolo advém da determinação de quanto tempo um nodo deve esperar antes de efetuar uma retransmissão, ficando a qualidade do canal de comunicação dependente da eficiência do algoritmo de *backoff* utilizado.

Para reduzir o volume de colisões do ALOHA há uma variação chamada *slotted* ALOHA. Nesta variação, o tempo é dividido em fatias de tamanho igual, que também deve equivaler ao tempo máximo de transmissão de um pacote na rede. Nodos que desejam transmitir iniciam transmissão apenas no início de cada fatia de tempo, garantindo que apenas ocorrerão colisões caso dois nodos tenham dados prontos para enviar no início da fatia de tempo. Se um nodo inicia a transmissão de uma pacote sozinho no início de uma fatia de tempo é garantido que terminará a transmissão sem colisão. Como no ALOHA Puro, a eficiência do algoritmo de *backoff* continua sendo de crucial importância para impedir que colisões ocorram com muita frequência.

4.2. CSMA - Carrier Sense Multiple Access, e variações CD e CA

O CSMA é um protocolo MAC probabilístico. Este protocolo verifica, sempre, o estado da rede no momento do envio de um pacote, ou seja, ele apenas transmite se o meio estiver livre. Caso o dispositivo encontre o meio livre, ele transmite seu pacote com uma probabilidade p, caso esteja ocupado, ele aguara por um período de tempo e tenta novamente. No momento da transmissão, "transmitir com uma probabilidade p" significa que nem sempre que o protocolo encontrar o meio livre ele transmitirá o pacote. Esta medida visa a redução de colisões, já que mais de um nodo podem decidir transmitir em um mesmo instante, sentindo, simultaneamente, o meio livre. Na versão CSMA p-persistente, p define a probabilidade de o protocolo utilizar o meio caso o detecte livre. No caso especial CSMA 1-persistente, o MAC vai transmitir pacotes sempre que encontrar o meio livre (1, neste caso, equivale a uma probabilidade de 100%).

Duas modificações do CSMA são largamente utilizadas em redes padronizadas: o CSMA/CD (CSMA with Collision Detection - Detecção de Colisão) e o CSMA/CA (CSMA with Collision Avoidance - Prevenção de colisão). O CSMA/CD é utilizado no padrão IEEE 802.3. Ao detectar uma colisão, o MAC CSMA/CD pode, eventualmente, tentar se recuperar da colisão ou, caso não seja possível recuperar, reiniciar a transmissão. Este MAC é bastante utilizado em redes cabeadas. Já o CSMA/CA é utilizado em redes sem-fio, incluindo os padrões IEEE 802.11 (WiFi) e IEEE 802.15.4.

No algoritmo CSMA/CA (*Carrier-Sense Multiple Access with Collision Avoidance*) uma estação que deseja realizar uma transmissão verifica o meio para determinar se está livre. Caso o meio estiver livre, transmite-se o quadro. Do contrário, a

estação aguarda um intervalo de tempo aleatório antes de verificar o meio novamente. O CSMA/CA busca solucionar os problemas clássicos de transmissão sem-fio conhecidos como "problema da estação escondida" e "problema da estação exposta". Basicamente, o CSMA/CA utiliza sinalizações RTS (*Ready to Send*) e CTS (*Clear to Send*) para verificar a existência de atividade de rádio no canal em uso na região do receptor.

4.3. MACs para redes de sensores sem-fio

Um protocolo de controle de acesso ao meio (MAC) decide quando um nodo de rede pode acessar o meio, tentando garantir que nodos não interfiram nas transmissões uns dos outros. No contexto de redes de sensores sem-fio , protocolos MAC são ainda responsáveis por implementar o uso eficiente do rádio, que é frequentemente o componente mais crítico em termos de consumo de energia. Um MAC neste cenário normalmente considera métricas tradicionais de rede como latência, vazão e disponibilidade menos importantes que o baixo consumo de energia. Com isso, os principais fontes de sobrecusto em comunicação via rádio (i.e., escuta ociosa, colisões, escuta desnecessária e flutuações no tráfego) definem metas seguidas por, praticamente, todos os MACs neste contexto [Langendoen and Halkes 2005].

O *B-MAC* é um protocolo MAC para redes de sensores sem-fio com *carrier sense*, ou seja, um protocolo que observa o uso do meio antes de utilizá-lo para envio de dados [Polastre et al. 2004]. Ele provê uma interface que permite reconfiguração online, o que permite que os serviços de rede ajustem seus mecanismos. Estas reconfigurações incluem aspectos como ligar e desligar o uso de CCA (*Clear Channel Assessment*), o envio de mensagens de reconhecimento (ACKs), ajuste do tamanho do preâmbulo e do intervalo de escuta. Uma limitação do B-MAC é que um receptor tem que esperar até que o preâmbulo seja completamente transmitido para iniciar a troca de dados, mesmo se o receptor já estiver acordado no início da transmissão. Além deste atraso, isto ainda implica num problema conhecido como escuta desnecessária (*overhearing*), onde receptores permanecem acordados (e, portanto, consumindo energia) até o final do preâmbulo para, apenas após este momento, descobrir que o pacote não era endereçado a eles. Estas limitações são resolvidas pelo *X-MAC*, que usa preâmbulos curtos dentro dos quais o endereço do receptor está "escondido" [Buettner et al. 2006]. Assim, um receptor pode identificar se um pacote é destinado a ele antes de receber o pacote inteiro, podendo ou simplesmente desligar o rádio, caso não seja o destinatário, ou enviar um ACK ao transmissor, avisando-o de que este pode parar de enviar o preâmbulo e iniciar o envio do pacote de dados. Como ambos os protocolos são baseados no CSMA eles sofrem do problema da estação escondida.

S-MAC é um protocolo MAC para redes de sensores sem-fio também é baseado no CSMA [Ye et al. 2002], mas ele utiliza um mecanismo de RTS/CTS para evitar o problema da estação escondida. Nodos vizinhos trocam informação de sincronização para que acordem simultaneamente para se comunicar. Uma grande limitação do S-MAC é que ele não permite nenhum tipo de configuração, nem estática, nem dinâmica, apresentando um período de atividade (*duty cycle*) fixo que pode terminar por desperdiçar energia (*idle listening*). O *T-MAC*, que é uma versão melhorada do S-MAC, trata deste problema e adapta, dinamicamente, seu *duty cycle* através de um mecanismo refinado baseado em *timeouts*. Nestes protocolos, a troca de informação necessária para manter os nodos sincronizados produz um certo sobrecusto.

O *Z-MAC* é um protocolo híbrido, que combina TDMA e CSMA [Rhee et al. 2008]. Ele usa um escalonamento TDMA, mas permite que nodos disputem as fatias alocadas a outros nodos utilizando CSMA. O protocolo dá aos nodos aos quais as fatias de tempo foram alocadas exclusividade para iniciar transmissões logo no início do período alocado. Caso o nodo dono da fatia não inicia sua transmissão, outros nodos começam a disputar o uso do meio através de um CSMA. O Z-MAC prevê o cálculo e atribuição das fatias de tempo no momento da implantação da rede, o que limita sua adaptabilidade.

O MAC do padrão *IEEE 802.15.4* controla o acesso dos protocolos de mais alto nível à camada física por dois modos distintos [IEEE Computer Society 2006]. Em um modo de operação básico, ele utiliza CSMA/CA e pacotes de reconhecimento (ACK) para tratar colisões. Em outro modo, conhecido como modo com *beacons*, o MAC utiliza quadros chamados *beacons* para sincronizar os dispositivos na rede. Neste modo, o período de atividade da rede é divido em duas partes: um com contenção (CAP - *Contention Access Period*) e um livre de contenção (CFP - *Contention Free Period*). No CAP, os nodos operam normalmente no CSMA/CA, com possibilidade de colisões durante a contenção. No CFP, contudo, o protocolo divide o meio em fatias de tempo (GTS - *Guaranteed Time Slots*), que são alocadas a nodos que podem, então, se comunicar sem a possibilidade de colisões. O frame de *beacon* é transmitido periodicamente pelo nodo coordenador da rede e, além de servir como baliza para sincronização da rede, transporta informações de configuração como, por exemplo, a tabela de alocação dos GTSs, informação sobre dados pendentes para leitura, que são utilizados na sincronização de dados da rede, e a definição do período de atividade da rede, o que permite que dispositivos entrem em modo de baixo consumo de energia, desligando seus rádios, durante a fase de inatividade. O modo *beacon* permite uma melhor sincronização entre os dispositivos, baixando o consumo de energia, mas ao preço de uma menor vazão.

4.4. C-MAC

o *C-MAC* (*Configurable MAC*) é um protocolo MAC altamente configurável para redes de sensores sem-fio implementado como um arcabouço de estratégias de controle de acesso ao meio que podem ser combinadas para produzir protocolos específicos a uma determinada aplicação [Wanner et al. 2007]. Ele permite que programadores de aplicação configure diversos parâmetros de comunicação (e.g., sincronização, contenção, detecção de erros, reconhecimento, empacotamento, etc) para ajustar o protocolo especificamente para a aplicação sendo desenvolvida. Embora altamente configurável, as instâncias do C-MAC configuradas para operar conforme o B-MAC produziram melhores resultados que a implementação original deste protocolo em termos de tamanho de código e dados, desempenho e eficiência na utilização da rede. Isto se deve às técnicas de metaprogramação estática usadas na implementação do C-MAC em C++, que permitem ao compilador gerar uma série de otimizações reduzindo, especialmente, sobrecusto introduzido por chamadas de funções e por polimorfismo.

A versão original do C-MAC, no entanto, definiu os elementos configuráveis do protocolo de modo relativamente grosseiro. Por exemplo, sincronização foi modelada como um único e grande componente, que precisava ser reimplementado para todo novo protocolo, mesmo sabendo que aspectos como geração de preâmbulo e sincronização de relógio são comuns a, praticamente, qualquer protocolo. Uma revisão do projeto foi

então realizada para refinar os componentes do C-MAC [Steiner et al. 2010]. Nesta revisão do C-MAC foi realizada uma decomposição dos protocolos MAC tradicionais para obter uma máquina de estados generalizada para cada um das três principais categorias de MAC [Klues et al. 2007]: *channel pooling*, *scheduled contention* e TDMA. As Figuras 8, 9 e 10 apresentam as máquinas de estados desenvolvidas. É importante notar que estas máquinas de estado incluem funcionalidades exclusivas de alguns tipos de MAC. Dependendo da configuração adotada, estados das máquinas de estados podem ser suprimidos. Como exemplo, pode-se citar o estado `TX ACK PREAMBLE` na máquina de estados da Figura 8, que é utilizada pelo X-MAC;

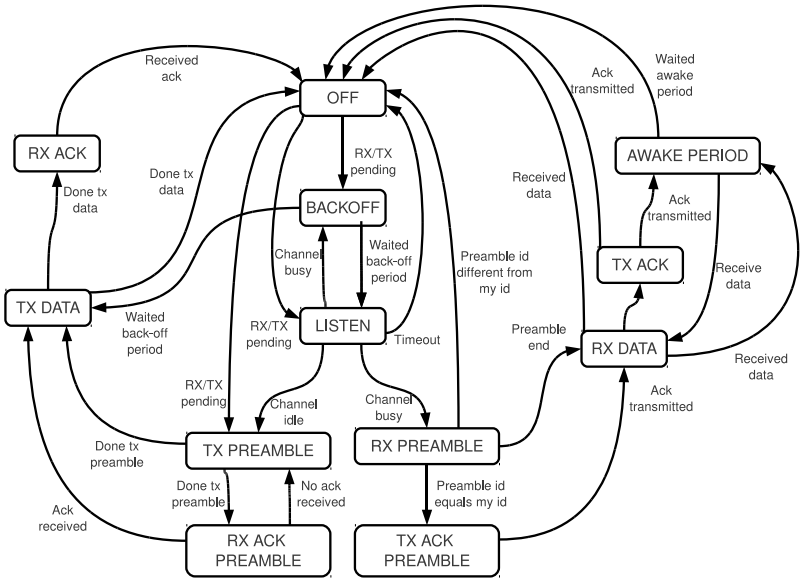

Figura 8. Máquina de estados para protocolos baseados em *channel polling*.

Uma análise cuidadosa destas máquinas de estado levou à definição da máquina de estados para o C-MAC (Figura 11). Cada estado representa um microcomponente que pode ter diferentes implementações. Estes microcomponentes junto das transições de estados podem ser combinados para produzir protocolos específicos para cada aplicação. A implementação desta máquina de estados utiliza técnicas de metaprogramação estática (templates C++) para remover estados que não fazem parte de uma determinada instância do protocolo. Quando um estado é removido da máquina de estados, as entradas do estado removido são encaminhadas diretamente ao(s) próximo(s) estados conectado(s) a ele, mantendo a semântica original das transições. Além de ser capaz de acomodar protocolos representativos em qualquer das três categorias estudadas, a máquina de estados do C-MAC ainda suporta protocolos híbridos, como Z-MAC e IEEE 802.15.4.

Na máquina de estados apresentada na Figura 11 existem quatro "macro estados": `SYNCHRONOUS SYNC`, `ASYNCHRONOUS SYNC`, `RX CONTENTION` e `TX CONTENTION`. Estes estados, na verdade, são abstrações de outras máquinas de estados específicas para estas funções complexas executadas pelo protocolo, e são arpesentados na Figura 12, 13, 14 e, 15.

Através desta máquina de estados é possível prover um maior número de pontos configuráveis num framework com um alto nível de reuso. Os principais pontos configuráveis do C-MAC incluem:

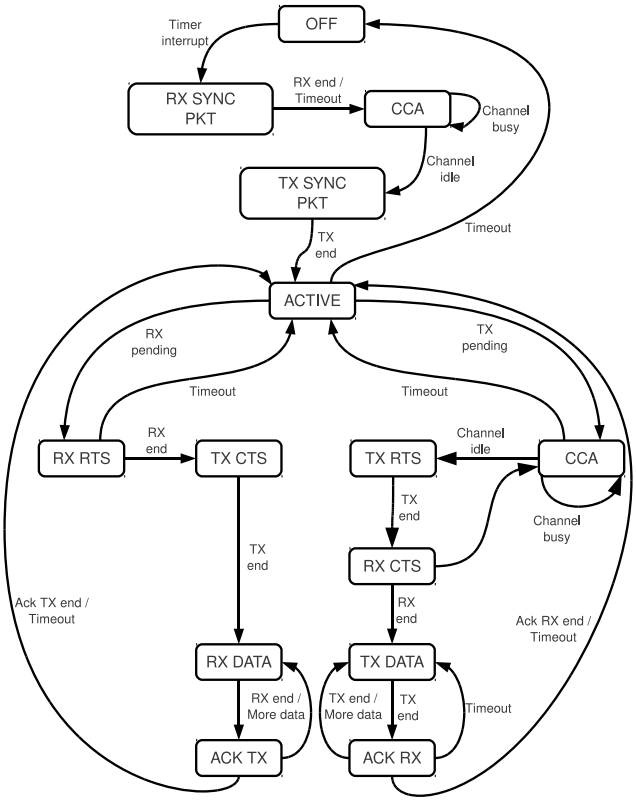

Figura 9. Máquina de estados para protocolos baseados em *scheduled contention*.

- **Configuração da camada física:** através da disponibilização de funcionalidades que adaptam o hardware abaixo do MAC para atender às funcionalidades incluídas na instância do C-MAC em uso (e.g., frequência, potência de transmissão, taxa de transmissão);
- **Sincronização e organização:** permite configurar os dispositivos para enviar ou receber dados de sincronização para organizar a rede e sincronizar dados ou os períodos de atividade (*duty cycle*);
- **Mecanismo de prevenção de colisões (colision avoidance):** define os mecanismos de contenção usados para prevenir colisões. Pode incluir um algoritmo CSMA-CA, a troca de pacotes de controle de contenção (RTS/CTS) ou uma combinação dos dois;
- **Mecanismo de reconhecimento:** a troca de pacotes "ACK" para determinar o sucesso de uma transmissão, incluindo reconhecimento de pacotes de dados e de preâmbulo.
- **Tratamento de erros e segurança:** determinar que mecanismos serão usados para garantir a consistência dos dados (e.g., CRC, checksum) e segurança dos dados.

Testes realizados com o C-MAC apresentaram desempenho ligeiramente superior a outros protocolos configurados de modo equivalente, porém apresentou um menor tamanho [Wanner et al. 2007]. Esta vantagem, contudo, é aumentada pelo sistema de configuração do C-MAC, que permite a criação de protocolos específicos para aplicações, que terminam por ter apenas os sobrecustos estritamente necessários.

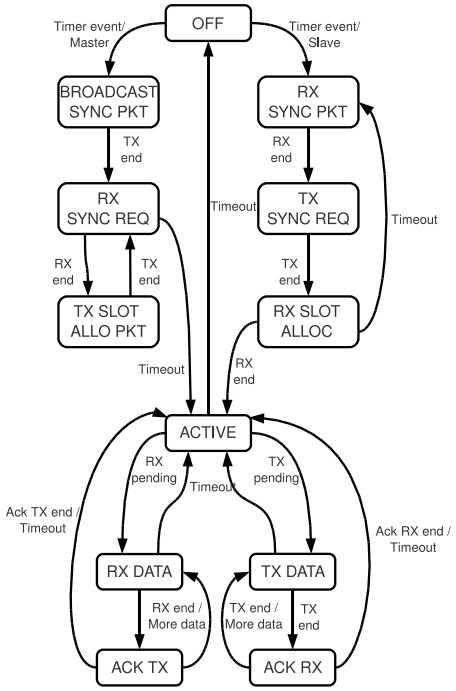

Figura 10. Máquina de estados para protocolos baseados em TDMA.

5. Sistemas Operacionais para Redes de sensores sem-fio

Esta seção apresenta uma revisão de características necessárias aos sistemas operacionais de redes de sensores sem-fio . Ao final será apresentado o EPOS , um sistema operacional desenvolvido no LISHA para plataformas embarcadas que implementa muitas das funcionalidades utilizadas em redes de sensores sem-fio .

Em uma redes de sensores sem-fio , requisitos específicos das aplicações em desenvolvimento guiam todo o projeto de hardware, incluindo a capacidade de processamento, largura de banda e taxas de transmissão do rádio, e módulos de sensores, requerendo que o projeto seja modular. Estes requisitos, contudo, leva a uma grande variedade de componentes de hardware, tornando os dispositivos de redes de sensores sem-fio não apenas modulares, mas heterogêneos. Neste cenário, uma aplicação de sensoriamento desenvolvida para uma dada plataforma dificilmente será portável a uma plataforma diferente, a não ser que o sistema de suporte de tempo de execução nestas plataformas provejam mecanismos que abstraiam e encapsulem a plataforma de sensoriamento de modo adequado. Ao mesmo tempo, os recursos limitados tipicamente encontrados no hardware para redes de sensores sem-fio forçam qualquer sistema para estes dispositivos a serem eficientes e não usar recursos em excesso.

A necessidade por conectividade, abstração de hardware e gerenciamento de recursos limitados torna imperativo o suporte de um sistema operacional para aplicações de redes de sensores. Considerando a pesquisa, tecnologia e aplicações atuais é enumera-se uma série de requisitos para sistemas operacionais de redes de sensores sem-fio [Wanner and Fröhlich 2008]:

1. **Prover funcionalidade básica de sistema operacional:** Para não restringir funcionalidade e portabilidade das aplicações, um sistema operacional para redes

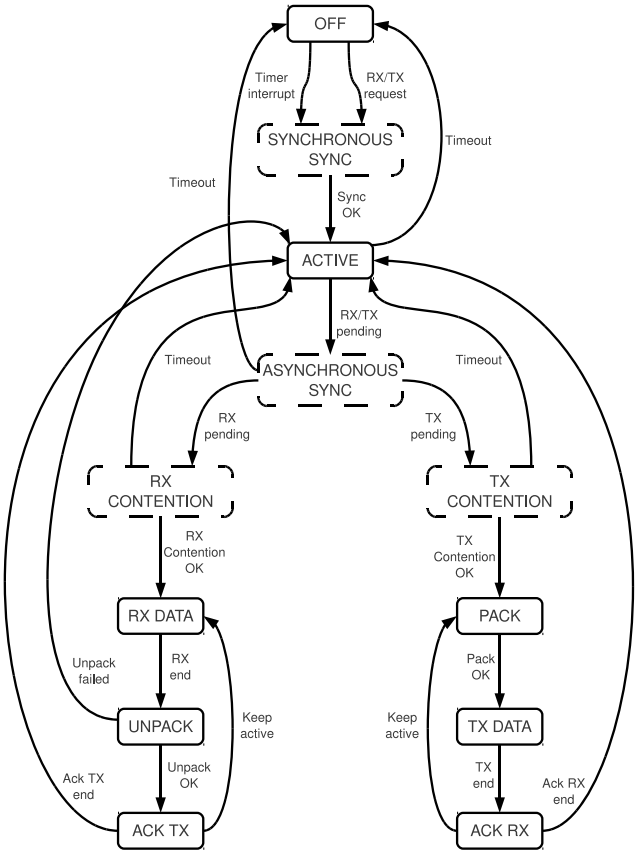

Figura 11. Máquina de estados do C-MAC.

de sensores sem-fio deve prover serviços tradicionais de sistema operacional como: abstração de hardware, gerenciamento de processos (geralmente seguindo o prisma "monotarefa, multi-thread"), serviços de temporização e gerenciamento de memória.

2. **Prover mecanismos eficientes para gerenciar consumo de energia:** Gerência de energia dos nodos de sensoriamento é um fator determinante da vida útil de uma redes de sensores sem-fio . O sistema de suporte em tempo de execução para aplicações de redes de sensores deve prover mecanismos de gerência de energia para as aplicações, assim como usar o mínimo de energia possível para prover seus serviços.
3. **Prover mecanismos de reprogramação em campo:** Dado que os nodos de redes de sensores sem-fio podem estar localizados em regiões inóspitas e que requisitos e parâmetros das aplicações podem mudar com o tempo, reprogramação em campo através da rede de comunicação é um serviço importante neste tipo de sistema. Um sistema operacional para redes de sensores sem-fio deve idealmente prover, para aplicações já implantadas, mecanismos de reprogramação total ou parcial em campo.
4. **Abstrair hardware de sensores heterogêneos de modo uniforme:** Como já dito acima, os requisitos específicos de uma aplicação de redes de sensores sem-fio faz do seu hardware não apenas modular, mas também heterogêneo, fazendo com que seja difícil portar uma aplicação de sensoriamento de uma plataforma para outra distinta. Além das diferenças arquiteturais, os próprios sensores (e.g., tem-

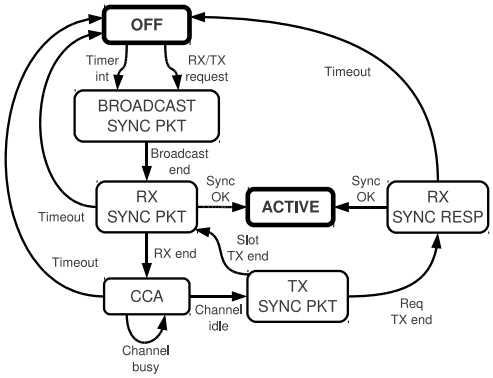

Figura 12. Máquina de estados `SYNCHRONOUS SYNC`.

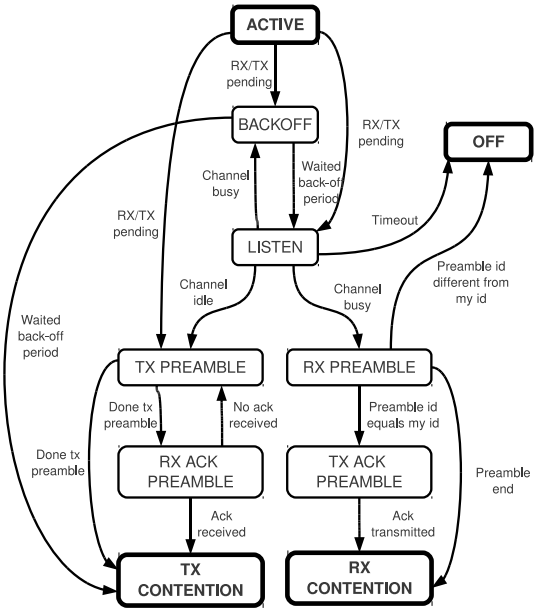

Figura 13. Máquina de estados `ASYNCHRONOUS SYNC`.

peratura, luz, sensores de movimento) apresentam uma variabilidade ainda maior. Módulos de sensores apresentando a mesma funcionalidade frequentemente variam sua interface de acesso, características operacionais e parâmetros. Um sistema de suporte de tempo de execução apropriadamente projetado pode liberar os programadores das aplicações destas dependências arquiteturais e promover a portabilidade das aplicações a diferentes plataformas de sensoriamento.

5. **Prover uma pilha de protocolos de comunicação configurável:** Dados os requisitos específicos de comunicação que diferentes aplicações apresentam, o hardware de comunicação para redes de sensores sem-fio deve apresentar um certo nível de configurabilidade. O sistema operacional deve prover meios para configurar a pilha de protocolos de comunicação, a partir dos protocolos de controle de acesso ao meio (MAC), garantindo assim que aplicações possam explorar as características do dispositivo de comunicação empregado do modo que melhor lhe convir.
6. **Operar com recursos limitados:** Como nodos de redes de sensores sem-fio pre-

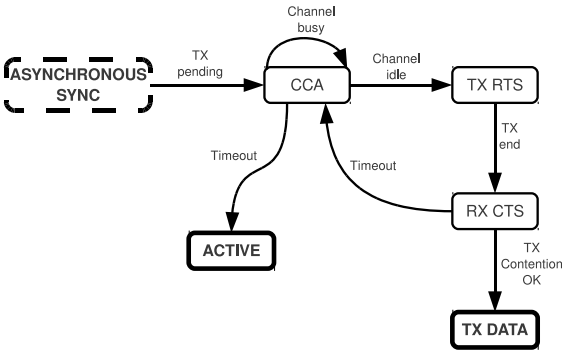

Figura 14. Máquina de estados `TX CONTENTION`.

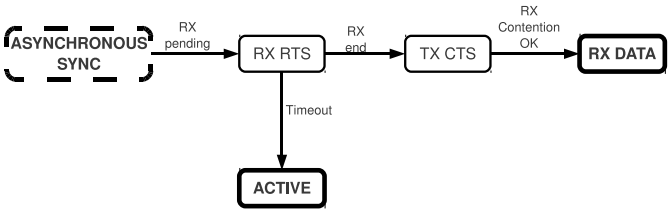

Figura 15. Máquina de estados `RX CONTENTION`.

cisam consumir pouca energia, o projeto do hardware destes nodos deve trocar, sempre que possível, sua capacidade de computação por mais baixa potência. Assim, os nodos terão recursos de processamento e memória limitados. Um sistema operacional para redes de sensores sem-fio deve entregar os serviços requisitados pela aplicação sem utilizar uma quantidade elevada dos recursos computacionais disponíveis.

Sistemas operacionais típicos para sistemas embarcados, como VxWorks, QNX, OS-9, WinCE e μClinux, provêm um ambiente de programação similar àqueles existentes em computadores tradicionais, normalmente através de serviços compatíveis com POSIX. Muitos destes sistemas operacionais provêm e requerem suporte em hardware para proteção de memória. Embora estes sistemas sejam adequados para outros aplicações embarcadas complexas como telefones celulares, set-top-boxes, seus requisitos em termos de capacidade de memória e de processamento torna impossível seu uso em redes de sensores sem-fio . Vários sistemas foram projetados especialmente para estas redes, incluindo MagnetOS [Barr et al. 2002], Contiki [Dunkels et al. 2004] e AmbientRT [Hofmeijer et al. 2004]. Os mais proeminentes são, contudo, os sistemas TinyOS [Hill et al. 2000], MANTIS OS [Abrach et al. 2003] e SOS [Han et al. 2005].

O TinyOS é um sistema operacional baseado em eventos. O sistema é organizado em uma coleção de componentes. Cada configuração do TinyOS é composta por uma aplicação e os serviços de sistema operacional por ela requiridos, consistindo de um escalonador e um grafo de componentes. Cada componente é composto por comandos, tratadores de eventos, tarefas e quadro de execução. Cada componente ainda declara os comandos aos quais responde e os eventos que ele sinaliza. Comandos são chamadas de métodos não bloqueantes e são tipicamente usados para iniciar requisições de software e hardware e, condicionalmente, iniciar tarefas. tratadores de eventos são usados para tratar interrupções de hardware e podem chamar comandos ou disparar tarefas.

O sistema possui um modelo de concorrência simplificado, baseado num modelo em que tarefas executam até completar, sendo apenas preemptadas apenas por interrupções. Este modelo traz consequências tanto positivas quanto negativas. em um modelo tradicional baseado em threads, onde cada thread tem sua própria pilha, cada thread precisa reservar espaço na memória, já limitada, do nodo para seu contexto de execução. Dependendo da arquitetura, troca de contexto pode ser uma operação extensa. Restringindo este modelo, o TinyOS reduz grande parte deste sobrecusto, mas também perde a maior parte das características de um modelo multi-thread. Esta restrição de concorrência pode ainda inibir a capacidade do sistema em tratar restrições de tempo-real. O TinyOS não provê mecanismos de alocação dinâmica de memória. Serviços de temporização são providos por uma interface de Timer. O modelo de componentes do TinyOS, junto de seu sistema simplificado de concorrência, permite ao sistema operar em plataformas com menos de 1 kilobyte de memória RAM.

Gerenciamento de energia no TinyOS é implementado pelo escalonador de tarefas, que faz uso da interface `StdControl` para iniciar e para componentes. Quando a fila do escalonador está vazia, o processador principal é colocado em um modo *sleep*, de baixa potência. Deste modo, novas tarefas somente serão disparadas na execução de um tratador de interrupção, que "acorda" o processador. Este método permite bons resultados para o microcontrolador principal, mas deixa métodos mais agressivos de gerência de energia a cargo da aplicação.

O TinyOS apresenta uma arquitetura de abstração de hardware composta de três camadas: *Hardware Presentation Layer* (HPL), *Hardware Adaptation Layer* (HAL) e *Hardware Interface Layer* (HIL) [Handziski et al. 2004]. A HPL agrupa componentes de dispositivos específicos em modelos específicos de cada domínio, como *Alarm* ou *ADC Channel*. A HAL provê a melhor abstração possível em termos de uso efetivo de recursos, mas ainda tenta não inibir a portabilidade da aplicação. A HIL usa componentes adaptados para implementar abstrações independentes de plataforma. O desenvolvedor de aplicações para o TinyOS pode escolher usar qualquer um dos níveis de abstração disponíveis, trocando portabilidade da aplicação por uso eficiente dos recursos.

A pilha de comunicação do TinyOS é baseada no protocolo controle de acesso ao meio B-MAC [Polastre et al. 2004]. O protocolo é implementa em duas camadas: controle de hardware (LLHC - *low-level hardware control*) e lógica do protocolo (PL - *protocol logic*). A camada de controle de hardware permite configuração estática e dinâmica dos parâmetros básicos de comunicação (e.g., frequência, potência de transmissão). O sistema também permite algum nível de configuração da camada da lógica do protocolo (e.g., *duty cycle*, algoritmo de detecção de canal livre, uso de reconhecimento).

O MANTIS OS (Multimodal networks of in situ sensors) [Abrach et al. 2003] é um sistema operacional multi-thread que utiliza uma API (*Application Programming Interface*) inspirada em POSIX e adaptada às necessidades e restrições das redes de sensores sem-fio . A API é preservada entre diferentes plataformas e o kernel do sistema é composto de um escalonador e drivers de dispositivos. Uma pilha de comunicação e um servidor de comandos são disponibilizados a serviços de nível de usuário.

O escalonador do MANTIS OS provê um subconjunto do pacote POSIX threads, com escalonamento round-robin baseado em prioridades. O sistema suporta alocação de

memória na estática e dinâmica para as threads. O escalonador é acionado periodicamente por um timer, através de operações em semáforos. Uma thread *idle* de baixa prioridade é utilizada como ponto de entrada para as políticas de gerência de energia do sistema, que põem o processador em modo *sleep* sempre que não há threads aptas para execução. Serviços de temporização e de sincronização são providos através de interfaces similares às definidas pelo padrão POSIX. O mecanismo de escalonamento complexo utilizado no MANTIS OS acarreta um sobrecusto maior que o acarretado por modelos mais simples, baseados em eventos. Logo, o sistema necessita de mais memória, tanto RAM quanto de código, que, por exemplo, o TinyOS. O sistema ainda é, contudo, adequado para uso em protótipos atuais de redes de sensores sem-fio .

O MANTIS usa uma camada de abstração de hardware (HAL - *Hardware Abstraction Layer*), com as funções `dev_read()`, `dev_write()`, `dev_mode()` e `dev_ioctl()`. Cada função toma um dispositivo como parâmetro e uma tabela de funções redireciona chamadas gerais aos drivers dos dispositivos. A lista de parâmetros para as funções `dev_mode()` e `dev_ioctl()` são específicas para cada dispositivo e não há uma abstração unificada para os sensores (cada driver de dispositivo apresenta semânticas específicas).

O sistema provê uma interface de comunicação unificada através de threads em nível de usuário. Há um formato unificado de pacote para diferentes interfaces de comunicação (e.g., RS-232, USB, rádio). Esta camada de comunicação gerencia a sincronização de pacotes e *buferização*. Sob a API de comunicação, o MANTIS OS usa drivers de dispositivos tradicionais. As aparentes vantagens de um ponto de entrada único para comunicação é diminuída devido à semântica e parâmetros específicos dos métodos de comunicação para cada interface.

O SOS [Han et al. 2005] é um sistema operacional dinamicamente reconfigurável para redes de sensores sem-fio . O kernel do sistema inclui serviços de passagem de mensagens, alocação dinâmica de memória e carga dinâmica de módulos. O SOS é organizado em uma série de módulos binários que implementam tarefas específicas. Estes componentes são comparáveis em funcionalidade aos componentes do TinyOS. Uma aplicação é composta por uma série de módulos que interagem entre si, apresentando tanto interface por chamada de métodos quanto por passagem de mensagens. Passagem de mensagens é assíncrona e coordenada por um escalonador que usa uma fila ordenada por prioridades. Chamadas diretas a funções são utilizadas para operações síncronas entre os módulos. O sistema integra alocação dinâmica de memória e *garbage collection*. Como no TinyOS e no MANTIS OS, SOS coloca o processador em modo *sleep* sempre que não há mensagens para escalonar. O modelo de reconfiguração dinâmica do SOS implica em sobrecustos consideravelmente maiores quando comparado com outros sistemas. Contudo, este sobrecusto ainda é aceitável aplicações de redes de sensores sem-fio] [Han et al. 2005].

SOS provê serviços para, dinamicamente, incluir, atualizar e remover módulos em programas previamente implantados. O sistema divide a memória de programa em páginas, e mantém estruturas de estado e contexto na memória RAM para cada módulo.

O sistema usa os mecanismos de módulos de kernel carregáveis nas abstrações de hardware de sensoriamento. Através desta arquitetura, drivers de dispositivos podem registrar seus serviços e associá-los a um nome, permitindo às aplicações acessar com-

ponentes através destes nomes. Por exemplo, um driver de um sensor analógico pode se associar a um canal do ADC (Analog-to-Digital Converter) e registrar um sensor de um tipo específico (e.g., PHOTO - luminosidade, TEMP - temperatura, etc). Quando a aplicação requer dados de, por exemplo, PHOTO, o kernel usa o driver registrado para obter a leitura apropriada. Esta abstração semântica das leituras de sensores promove portabilidade das aplicações. Contudo, como o sistema operacional precisa manter uma tabela de ponteiros para funções indexada por nomes, o registro de drivers implica em algum sobrecusto de memória.

5.1. EPOS - Embedded Parallel Operating System

Requisitos de sistema para redes de sensores sem-fio incluem funcionalidades básicas de sistema operacional (e.g., gerência de memória, escalonamento), gerência de energia, mecanismos de reprogramação dinâmica, abstração dos dispositivos de sensoriamento (sensores) e uma pilha de comunicação configurável. A capacidade limitada do hardware de redes de sensores sem-fio requer que estes sistemas operem com recursos limitados, o que faz do uso e adaptação de sistemas operacionais tradicionais impossível. Vários projetos de pesquisa [Abrach et al. 2003, Barr et al. 2002, Dunkels et al. 2004, Han et al. 2005, Hill et al. 2000, Hofmeijer et al. 2004] focaram na solução do problema de suporte de sistema para redes de sensores. A maioria deles, contudo, falhou no tratamento de dois requisitos: configuração transparente do canal de comunicação e abstração eficiente e unificada da camada de abstração de sensores.

O EPOS (Embedded Parallel Operating System) [Fröhlich 2001, Marcondes et al. 2006] é um arcabouço baseado em componentes para a geração de ambientes dedicados de suporte de tempo de execução. O arcabouço do EPOS permite que programadores desenvolvam aplicações independentes de plataforma, e ferramentas de análise permitem que componentes sejam adaptados automaticamente para atender a requisitos destas aplicações particulares. Por definição, uma instância do sistema agrega todo suporte necessário para a sua aplicação dedicada, e nada mais.

O projeto modular do EPOS foi guiado pela metodologia Projeto de Sistemas Embarcados Guiado pela Aplicação (ADESD - *Application-Driven Embedded System Design*). A ADESD baseia-se nas conhecidas estratégias de decomposição de domínio por trás do Projeto Baseado em Famílias (FBD - *Family-Based Design*) e Orientação a Objetos (OO) como, por exemplo, análise de atributos comuns e variabilidade, para adicionar o conceito de identificação e separação de aspectos ainda nos estágios iniciais do projeto [Fröhlich 2001]. Deste modo, a ADESD guia a engenharia de domínio para famílias de componentes, das quais dependências de cenários de execução são fatoradas na forma de aspectos e relacionamentos externos são capturados em um arcabouço de componentes. Esta estratégia de engenharia de domínio trata consistentemente algumas das questões mais relevantes do desenvolvimento de software baseado em componentes: reuso, gerenciamento da complexidade e composição.

A Figura 16 mostra o processo de decomposição de domínio no projeto de sistemas embarcados guiado pela aplicação. Abstrações são identificadas a partir do domínio do problema e organizadas em famílias, de acordo com suas características comuns. Dependências de cenário são modeladas como aspectos que podem ser aplicados através de adaptadores de cenário. Famílias de abstrações são visíveis para aplicações através de

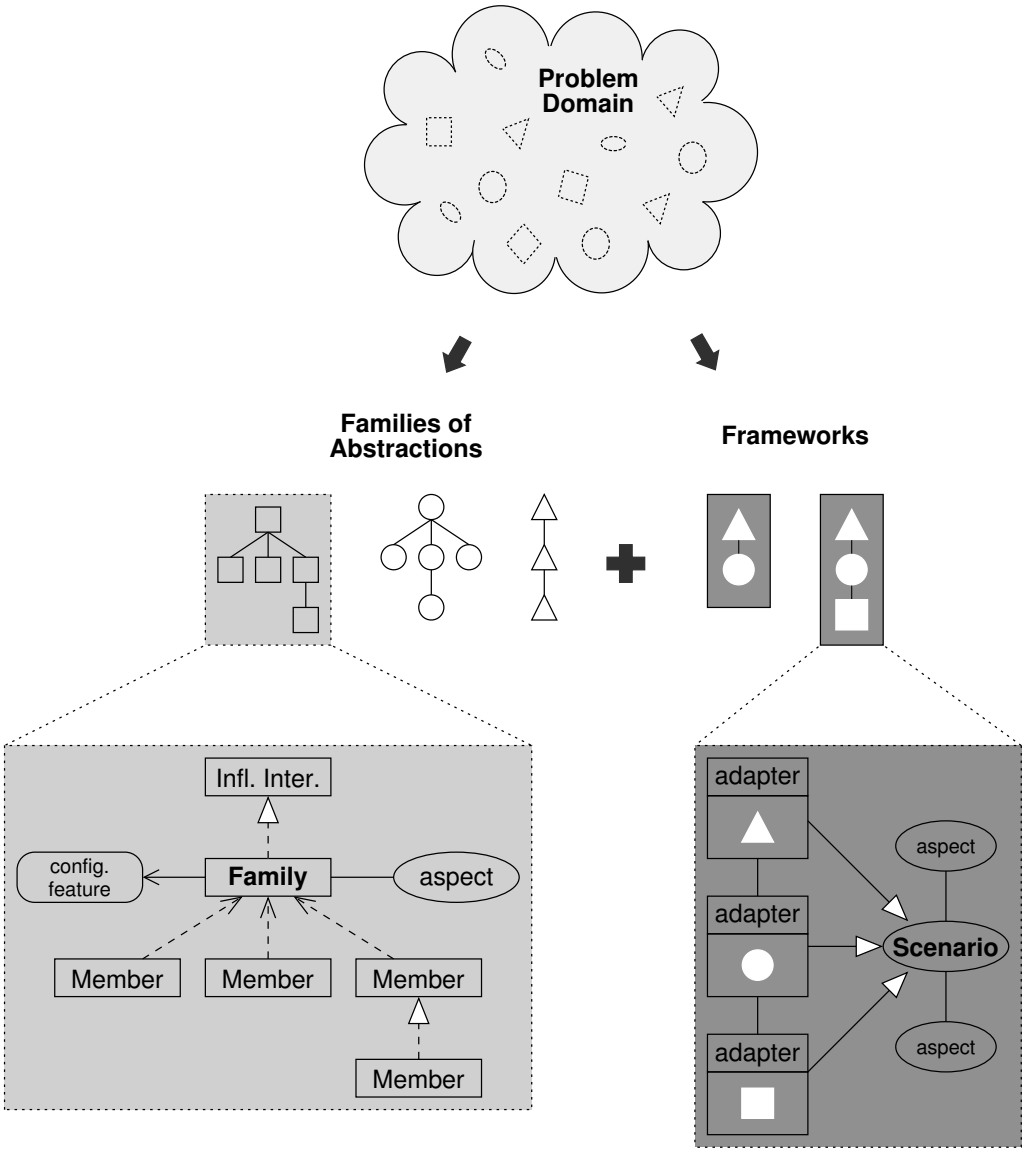

Figura 16. Processo de decomposição de domínio da ADESD.

interfaces infladas, que exportam seus membros como um único "super-componente". As arquiteturas de sistema são capturadas em arcabouços de componentes, que são definidos em termos de aspectos de cenário.

Famílias de abstrações no EPOS representam abstrações tradicionais de sistemas operacionais e implementam serviços como gerenciamento de memória e de processos, sincronização de processos, gerenciamento de tempo e comunicação. Abstrações são projetadas e implementadas de modo independente de seus cenários de execução e arquiteturas. Todas unidades de hardware dependentes de arquitetura são abstraídas como mediadores de hardware, que exportam, através de interfaces independentes de plataforma, a funcionalidade necessária às abstrações. Devido ao uso de metaprogramação estática e *function inlining*, mediadores de hardware implementam suas funcionalidades sem formar uma camada de abstração de hardware (HAL - *Hardware Abstraction Layer*) tradicional. Através do uso de mediadores de hardware, as abstrações do EPOS atingiram

um nível de reuso que permite, por exemplo, a mesma família de abstrações de threads ser utilizada em ambientes tanto monotarefa quanto multitarefa, como parte de um μkernel ou meramente ligada à aplicação, tanto em microcontroladores de 8-bits quanto em um processador de 64-bits.

No EPOS , processos são gerenciados pelas abstrações `Thread` e `Task`. Cada thread armazena seu contexto em sua própria pilha. A abstração de contexto defini o conjunto de dados que precisa ser armazenado para um fluxo de execução e, deste modo, cada arquitetura define seu próprio contexto.

Tempo é tratado pela família de abstrações `Timepiece`. Estas abstrações são suportadas através dos mediadores `Timer`, Timestamp Counter (`TSC`) e Real-Time Clock (`RTC`). A abstração `Clock` é responsável por um controle estrito de tempo e está disponíveis em sistemas que possuam um dispositivo de relógio de tempo real (RTC). A abstração `Alarm` pode ser utilizada para gerar eventos que acordem uma thread ou chamem uma função. Alarmes têm ainda um evento mestre de altíssima prioridade que está associado com um período de tempo pré-definido. Este evento mestre é utilizado para acionar o algoritmo de escalonamento do sistema quando o *quantum* de escalonamento é atingido, nos casos em que uma configuração com um escalonador ativo é utilizada. Finalmente, a abstração `Chronometer` é utilizada para realizar operações de medição de tempo.

A família de abstrações `Synchronizer` provê mecanismos que garantem a consistência de dados em ambientes com processos concorrentes. O membro `Mutex` implementa um mecanismo de exclusão mútua que entrega duas operações atômicas: `lock` e `unlock`. O membro `Semaphore` implementa, como o próprio nome diz, um semáforo, que é uma variável inteira cujo valor apenas pode ser manipulado indiretamente através das operações atômicas `p` e `v`. O membro `Condition` realiza uma abstração de sistema inspirada no conceito de variável de condição, que permite a uma thread esperar que um predicado se torne válido.

No EPOS , detalhes referentes a proteção e tradução de espaços de endereçamento, assim como alocação de memória, são abstraídas através da família de mediadores de hardware `MMU` (*Memory Management Unit*). A abstração `Address Space` é um contêiner para "fatias" de memória física chamados de `Segments`. Ele não implementa tarefas de proteção, tradução ou alocação de endereços de memória, deixando este papel para o mediador da MMU. O membro `Flat Address Space` define um modelo de memória no qual endereços lógicos e físicos coincidem, eliminando a necessidade de hardware para MMU. Em plataformas que não dispõem de MMU, um mediador para MMU simplesmente mantém o contrato de interface com o `Flat Address Space`, realizando implementações vazias de métodos quando necessário. Métodos relativos a alocação de memória operam sobre bytes de modo similar ao que é feito pela função de alocação de memória da libc.

Controle de entrada e saída (I/O) de dispositivos periféricos é disponibilizado no EPOS pelo mediador de hardware correspondente. O mediador `Machine` armazena as regiões de I/O e trata o registro dinâmico de interrupções. O mediador `IC` (*Interrupt Controller*) trata a ativação ou desativação de interrupções individuais. Para lidar com as diferentes interrupções existentes em diferentes plataformas e contextos, EPOS atribui

um nome e uma sintaxe independente de plataforma a interrupções pertinentes ao sistema (e.g., interrupção de timer, interrupção de conversão completa no ADC).

5.2. O EPOS para redes de sensores sem-fio

Aplicações de redes de sensores sem-fio apresentam requisitos específicos que vão além dos atendidos pelos serviços tradicionais de sistemas operacionais. Estes incluem gerenciamento de energia eficiente, reprogramação em campo, abstração uniforme de sensores e serviços de comunicação configuráveis. O EPOS foi estendido de modo a atender estes requisitos extras [Wanner and Fröhlich 2008].

O EPOS provê serviços de gerência de energia dirigida pela aplicação que permite um consumo consciente de energia em sistemas profundamente embarcados sem comprometer a portabilidade da aplicação e sem gerar sobrecustos excessivos. O objetivo do subsistema de gerenciamento de energia do EPOS é permitir que aplicações expressem quando determinados componentes de software não estão sendo utilizados, permitindo que o sistema migre dispositivos (hardware) associados a estes componentes de software para modos de operação que consumam menos energia. Disto emergiram várias questões que dizem respeito a diferenças arquiteturais entre dispositivos distintos e ao acesso concorrente aos recursos de hardware por diferentes componentes de software. Para tratar destas questões, foram concebidas (1) uma interface genérica para gerenciamento de energia, (2) um mecanismo de propagação de mensagens e (3) um modelo de formalização das trocas entre modos de operação [Hoeller et al. 2006b].

Nesta estratégia de gerenciamento de energia, espera-se que o programador da aplicação especifique, em seu código-fonte, quando certos componentes não serão utilizados. Assim, uma API uniforme que permite gerenciamento de energia foi definida. Esta interface permite a interação entre aplicação e sistema, entre componentes do sistema e dispositivos de hardware, além de permitir que a aplicação acesse diretamente o hardware se assim desejar o programador. Para que o programador da aplicação não necessite "acordar" explicitamente os componentes do sistema sempre que eles forem utilizados, o mecanismo de gerenciamento de energia intercepta acessos aos componentes de hardware desativados, retornando estes componentes a seu estado operacional anterior sempre que estes são utilizados.

A aplicação pode, como demostrado na Figura 17, acessar um componente global (`System`) que possui referências para todos os outros componentes no sistema, disparando ações globais de troca de modo de operação no sistema. A aplicação ainda pode utilizar a interface de gerenciamento de energia através de subsistemas (e.g., comunicação, processamento, sensoriamento). Deste modo, mensagens são propagadas apenas para componentes usados na implementação de cada subsistema. A aplicação pode ainda acessar o hardware diretamente, utilizando a API disponível nos drivers de dispositivos (e.g., *Network Interface Card* (NIC), CPU, Thermistor). A mesma API também é utilizada para troca de mensagens de gerenciamento de energia entre componentes do sistema.

Para garantir a portabilidade da aplicação e para facilitar o desenvolvimento da aplicação API de gerenciamento de energia foi definida com um conjunto mínimo de métodos e um conjunto de modos de operação universais, que possuem uma semântica unificada para todo o sistema. Portabilidade é alcançada pelo fato de a aplicação não necessitar implementar procedimentos específicos para cada dispositivo de modo a trocar

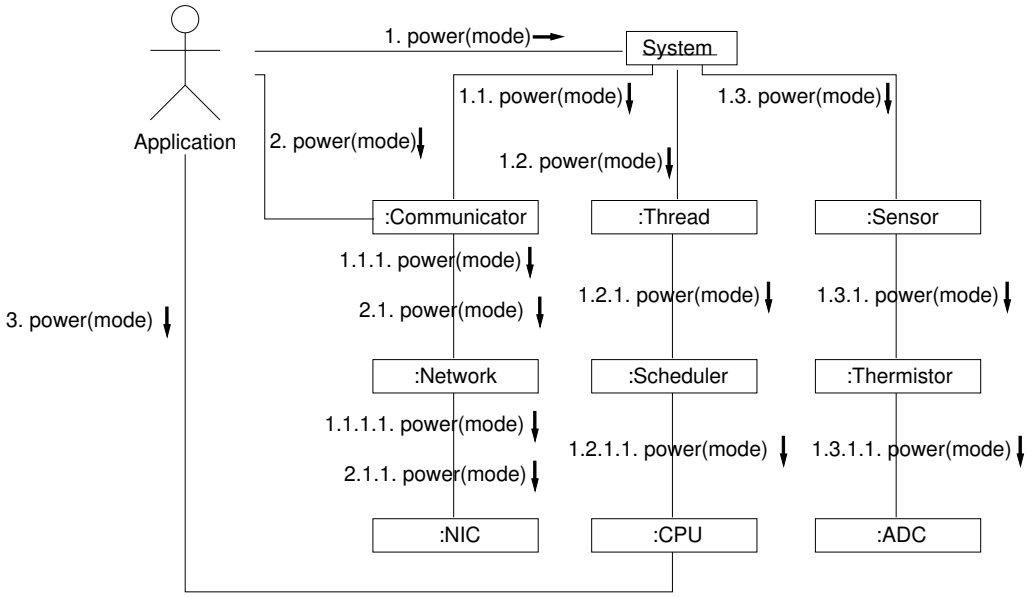

Figura 17. Hierarquia da API de gerenciamento de energia do EPOS .

seu modo de operação. Estes procedimentos são abstraídos pela API. A facilidade de uso vem do fato de que o programador da aplicação não necessita analisar documentação específica do hardware para identificar os modos de operação disponíveis, os procedimentos para utilizar estes modos e as consequências destas mudanças.

Estudos de caso [Hoeller et al. 2006b] demonstram que é possível alcançar uma economia significante de energia com mínima intervenção pela aplicação. Esta estrutura de gerenciamento de energia do EPOS é também utilizado por um gerenciador de energia ativo, que executa periodicamente ou quando não á threads a serem escalonadas e, oportunisticamente, altera o modo de operação de componentes e dispositivos para diminuir o consumo de energia do sistema. Este gerenciador de energia verifica o período de inatividade de cada componente e emprega heurísticas configuráveis que decidem os momentos apropriados para trocar seus modo de operação. Em sua versão mais simples, o gerenciador de energia põe todos os componentes ociosos por um determinado tempo em um modo de baixo consumo.

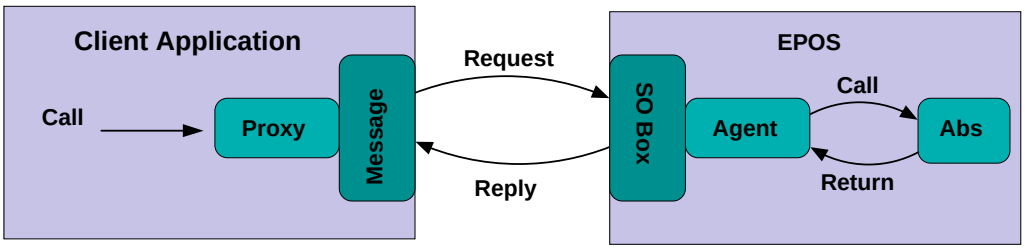

Figura 18. ELUS - o mecanismo de reprogramação em campo do EPOS .

Reprogramação em campo é realizada no EPOS através de um mecanismo de indireção similar ao RPC (*Remote Procedure Calls*), chamado ELUS (EPOS Live Update System) [Gracioli and Fröhlich 2008]. Nesta infraestrutura, apresentada na Figura 18, a invocação de um método de um componente por uma aplicação cliente passa por um

`Proxy`, que envia uma mensagem a um `Agent`. Após a execução do método, uma mensagem com o valor de retorno é devolvido à aplicação. Nesta estrutura, um nível de indireção é criado entre as chamadas de métodos pela aplicação e o sistema, fazendo com que o componente `Agent` seja o único ciente da posição do componente do sistema na memória. O `Agent` controla o acesso aos métodos dos componentes através de um sincronizador (`Semaphore`), não permitindo chamadas a um componente que esteja passando por uma atualização. Uma thread do sistema é responsável por receber requisições de atualização e o novo código para os componentes. Estas requisições é enviada ao agente, que sobrescreve o código antigo para incluir o novo. A infraestrutura de reprogramação é transparente à aplicação é pode ser "desligada", eliminando as indireções e o ocasional sobrecusto que surge tanto em volume de código e dados (memória) e atrasos de processamento.

O EPOS ainda provê suporte específico para aplicações de sensoriamento, definido interfaces software/hardware capazes de abstrair famílias de dispositivos sensores de modo uniforme [Wanner et al. 2006]. O subsistema de sensoriamento do EPOS já foi discutido na Seção 3.4 deste texto.

A infraestrutura de comunicação do EPOS para redes de sensores sem-fio é implementada pelo protocolo C-MAC, *Configurable MAC*, que provê suporte a comunicação de baixo nível (MAC - *Medium Access Control*) [Wanner et al. 2007]. O C-MAC já foi discutido na Seção 4.4 deste texto.

6. Conclusão

Este texto foi desenvolvido para dar suporte a um curso sobre conceitos básicos de redes de sensores sem-fio em que se relata as experiências dos autores no desenvolvimento dos projetos EPOS e EPOSMOTE [LISHA 2010b]. O principal foco do curso está no suporte de sistema operacional para o hardware de redes de sensores sem-fio , incluindo os subsistemas de processamento e memória, comunicação e sensoriamento.

O texto descreve em detalhes os resultados alcançados com o desenvolvimento no EPOS do C-MAC, um MAC configurável para redes de sensores sem-fio , e de um conjunto de abstrações para o hardware de sensoriamento. Estes desenvolvimentos, junto com os serviços básicos de sistema operacional do EPOS (escalonamento, gerenciamento de memória e gerenciamento de energia) possibilitaram a criação de um ambiente para desenvolvimento de aplicações de redes de sensores sem-fio independentes de plataforma, enquanto ainda mantendo bons resultados em termos de tamanho de código e dados, desempenho e consumo de energia.

Agradecimentos

Os autores gostariam de agradecer a todos os colaboradores do LISHA que contribuíram para o desenvolvimento do EPOS e do EPOSMOTE ao longo de quase 10 anos de trabalho.

Referências

Abrach, H., Bhatti, S., Carlson, J., Dai, H., Rose, J., Sheth, A., Shucker, B., Deng, J., and Han, R. (2003). Mantis: System support for multimodal networks of in-situ sensors.

In *2nd ACM International Workshop on Wireless Sensor Networks and Applications*, pages 50 – 59, San Diego, CA.

Abramson, N. (1970). The aloha system - another alternative for computer communicatons. In Press, A., editor, *Proceedings of the AFIPS Fall Joint Computer Conference*.

Albertazzi, A. J. and de Sousa, A. R. (2008). *Fundamentos de Metrologia Científica e Industrial*. Manole, Barueri, 1 edition.

Analog-Devices (2009). *3-Axis, +/-2 g / +/-4 g / +/-8 g / +/-16 g Digital Accelerometer (ADXL345) Datasheet*. Analog Devices.

Barr, R., Bicket, J. C., Dantas, D. S., Du, B., Kim, T. W. D., Zhou, B., and Sirer, E. G. (2002). On the need for system-level support for ad hoc and sensor networks. *ACM SIGOPS Operating Systems Review*, 36(2):1–5.

Buettner, M., Yee, G. V., Anderson, E., and Han, R. (2006). X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, New York, NY, USA. ACM.

Crossbow, T. (2005). *MTS/MDA Sensor and Data Acquisition Board User's Manual*. Crossbow Technology, Inc, San Jose, CA.

Dunkels, A., Grönvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455 – 462.

Fröhlich, A. A. (2001). *Application-Oriented Operating Systems*. GMD - Forschungszentrum Informationstechnik, Sankt Augustin.

Gracioli, G. and Fröhlich, A. A. (2008). An Operating System Infrastructure for Remote Code Update in Deeply Embedded Systems. In *First ACM Workshop on Hot Topics in Software Upgrades (HotSWUp)*, Nashville, USA.

Han, C.-C., Kumar, R., Shea, R., Kohler, E., and Srivastava, M. (2005). A dynamic operating system for sensor nodes. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 163–176, New York, NY, USA. ACM.

Handziski, V., Polastre, J., Hauer, J.-H., and Sharp, C. (2004). Flexible hardware abstraction of the ti msp430 microcontroller in tinyos. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 277–278, New York, NY, USA. ACM.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. (2000). System architecture directions for networked sensors. In *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104, Cambridge, Massachusetts, United States.

Hoeller, A. J., Wanner, L. F., de Oliveira, A. B., Immich, R., and Fröhlich, A. A. (2006a). Gerenciamento de Energia em Sistemas de Sensoriamento Remoto. In *Third Brazilian Workshop on Operating System*, pages 46–58, Campo Grande, Brazil.

Hoeller, A. J., Wanner, L. F., and Fröhlich, A. A. (2006b). A Hierarchical Approach For Power Management on Mobile Embedded Systems. In *5th IFIP Working Conference on Distributed and Parallel Embedded Systems*, pages 265–274, Braga, Portugal.

Hofmeijer, T., Dulman, S., Jansen, P., and Havinga, P. (2004). Ambientrt - real time system software support for data centric sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, pages 61 – 66.

Honeywell, S. S. E. C. (2005). *HMC1001/1002 1- and 2-Axis Magnetic Sensors Datasheet*. Honeywell, Plymouth, MN.

IEEE Computer Society (2006). IEEE Standard 802 Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Technical report, The Institue of Electrical and Electronics Engineers.

Klues, K., Hackmann, G., Chipara, O., and Lu, C. (2007). A component-based architecture for power-efficient media access control in wireless sensor networks. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 59–72, New York, NY, USA. ACM.

Langendoen, K. and Halkes, G. (2005). *Embedded Systems Handbook*, chapter Energy-Efficient Medium Access Control. CRC Press.

LISHA (2010a). Epos online user's guide. Internet. http://epos.lisha.ufsc.br/EPOS+User+Guide.

LISHA (2010b). Epos project website. Internet. http://epos.lisha.ufsc.br.

LISHA (2010c). Eposmote website. internet. http://epos.lisha.ufsc.br/EPOSMote.

Marcondes, H., Hoeller, A. J., Wanner, L. F., and Fröhlich, A. A. (2006). Operating Systems Portability: 8 bits and beyond. In *11th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 124–130, Prague, Czech Republic.

Meshnetics (2007). *Ultra-Compact 2.4GHz 802.15.4/ZigBee Modules for Wireless Networking Applications Product Datasheet*. Meshnetics, Moscow.

Mottola, L. and Picco, G. P. (2010). Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Computing Surveys*. To Appear.

Panasonic (2004). *ERTJ Multilayer Chip NTC Thermistors Datasheet*. Panasonic.

Polastre, J., Hill, J., and Culler, D. (2004). Versatile low power media access for wireless sensor networks. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107, New York, NY, USA. ACM.

Pottie, G. J. and Kaiser, W. J. (2000). Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58.

Rhee, I., Warrier, A., Aia, M., Min, J., and Sichitiu, M. L. (2008). Z-mac: a hybrid mac for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 16(3):511–524.

Sensirion (2005). *SHT1x / SHT7x Humidity and Temperature Sensor Datasheet*. Sensirion, Stäfa, Switzerland.

Steiner, R. V., Mück, T. R., and Fröhlich, A. A. (2010). A Configurable Medium Access Control Protocol for IEEE 802.15.4 Networks. In *Proceedings of the International Congress on Ultra Modern Telecommunications and Control Systems*, Moscow.

Steinhart, J. S. and Hart, S. R. (1968). Calibration curves for thermistors. *Deep Sea Research and Oceanographic Abstracts*, 15(4):497–503.

TAOS (2005). *TSL2550 Ambient Light Sensor with SMBus Interface Datasheet*. Texas Advanced Optoelectronic Solutions, Plano, Texas.

Wanner, L. F. (2006). Suporte de Sistema Operacional para Redes de Sensores Sem Fios. Master's thesis, Federal University of Santa Catarina, Florianópolis. M.Sc. Thesis.

Wanner, L. F., de Oliveira, A. B., and Fröhlich, A. A. (2007). Configurable Medium Access Control for Wireless Sensor Networks. In *International Embedded System Symposium*, pages 401–410, Irvine, CA, USA.

Wanner, L. F. and Fröhlich, A. A. (2008). Operating System Support for Wireless Sensor Networks. *Journal of Computer Science*, 4(4):272–281.

Wanner, L. F., Hoeller, A. J., de Oliveira, A. B., and Fröhlich, A. A. (2006). Operating System Support for Data Acquisition in Wireless Sensor Networks. In *11th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 582–585, Prague, Czech Republic.

Ye, W., Heidemann, J., and Estrin, D. (2002). An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the IEEE INFOCOM*, volume 3, pages 1567–1576. Citeseer.

Anexo: Exercícios práticos

O EPOS suporta uma variedade de arquiteturas, variando de 8 a 32 bits. Atualmente, estas arquiteturas são:

- IA32: single and multi-core;
- AVR8: atmega16, atmega128, atmega1281, and at90can128 microcontrollers;
- PPC32: including ml310;
- MIPS32: including plasma soft-core;
- ARM7: incluindo o MC13224V (Freescale) e o AT91SAM7 (Atmel).

O EPOS é software aberto, livre para uso não comercial. Mais detalhes sobre a licença do EPOS podem ser encontrados no sítio do projeto [LISHA 2010b].

Esta seção apresenta os passos básicos para obter o EPOS , instalar seu ambiente de desenvolvimento, compilá-lo e executá-lo em um EPOSMOTE .

Baixando e instalando o ambiente de desenvolvimento

O EPOS foi concebido prevendo seu desenvolvimento utilizando o ambiente GNU/Linux. Assim, o EPOS é um sistema cross-compilado no GNU/Linux para várias plataformas alvo, utilizando o *GNU Compiler Collection* (GCC). Versões binárias deste compilador para as plataformas que o EPOS suporta estão disponíveis para serem baixadas no sítio do EPOS [LISHA 2010b]. O primeiro passo é baixar o compilador e instalá-lo conforme as instruções constantes no sítio.

O EPOS também está disponível para baixar no sítio do projeto. Após baixar o sistema, é necessário descompactá-lo e configurar as seguintes variáveis de ambiente em seu sistema:

```
export EPOS=/path/to/epos
export PATH=$PATH:$EPOS/bin
```

Configurando o EPOS

A configuração do EPOS é realizada através da definição de variáveis no arquivo `$EPOS/makedefs`, com as seguintes opções:

- **MODE:** configura a arquitetura do sistema operacional:
  - *Library:* sistema é ligado à aplicação;
  - *Builtin:* sistema e aplicação estão no mesmo espaço de endereçamento;
  - *Kernel:* sistema e aplicação estão em espaços de endereçamento diferentes com uma camada de chamadas de sistema (*System Call*) entre eles.
- **ARCH:** configura a arquitetura do processador na plataforma alvo:
  - *ARCH_IA32:* Intel x86 32-bits Architecture;
  - *ARCH_AVR8:* Atmel AVR 8-bits Architecture;
  - *ARCH_PPC32:* IBM PowerPC 32-bits Architecture;
  - *ARCH_MIPS32:* MIPS 32-bits Architecture;
- **MACH:** configura a máquina para a qual o sistema será gerado:
  - *MACH_PC:* computadores pessoais;
  - *MACH_ATMEGA16:* microcontrolador Atmel ATMega16;
  - *MACH_ATMEGA128:* microcontrolador Atmel ATMega128;
  - *MACH_ATMEGA1281:* microcontrolador Atmel ATMega1281;
  - *MACH_AT90CAN128:* microcontrolador Atmel AT90CAN128;
  - *MACH_ML310:* Xilinx ML310 and ML403 Evaluation Boards;
  - *MACH_PLASMA:* microprocessador Plasma;

Compilando o EPOS

Com o EPOS configurado, para compilar o sistema basta executar um comando `make all` dentro do diretório em que o sistema foi instalado (`$EPOS`). Devido a algumas variações nas distribuições GNU/Linux utilizadas, alguns problemas podem surgir durante este processo. Soluções para estes problemas estão documentados no guia de usuário do EPOS , disponível no sítio do projeto [LISHA 2010a].

Compilando uma aplicação para o EPOS

Para compilar aplicações para o EPOS , o sistema provê uma ferramenta específica: o `eposcc`. Esta ferramenta analisa a aplicação, busca a configuração do sistema e realiza os comandos de compilação adequados utilizando o GCC. O primeiro passo é compilar a aplicação através do seguinte comando:

```
$ eposcc app/your_app.cc -o app/your_app.o
```

Feito isso, basta ligar o arquivo objeto da aplicação com o sistema operacional, pelo seguinte comando:

```
$ eposcc -o app/your_app app/your_app.o
```

Agora, o arquivo `app/your_app` contém o binário de sua aplicação já ligado ao sistema operacional. Para ter uma imagem binária pronta para ser carregada em seu hardware, basta utilizar a ferramenta `eposmkbi`, que é responsável por gerar uma imagem completa do sistema com as aplicações para execução na plataforma alvo. Para isso, basta executar o seguinte comando:

```
$ eposmkbi img/your_app.img app/your_app
```

Caso a imagem gerada para o sistema seja multitarefa, e você tenha compilado mais de uma aplicação para o sistema, basta listar todas as aplicações geradas no comando de construção da imagem de boot, deste modo:

```
$  eposmkbi img/your_app.img app/your_app1 app/your_app2...
```

Hello World!

Para simplificar o processo de desenvolvimento da aplicação, o `makefile` do EPOS está configurado para construir automaticamente o sistema para uma aplicação específica que pode ser informada pela linha de comando da seguinte forma:

```
$  make APPLICATION=helloworld
```

Deste modo, o `makefile` construirá o EPOS e, em seguida, compilará a aplicação informada e preparará a imagem de boot pronta para o sistema. Por exemplo, para criar uma aplicação "Hello World!" no EPOS , basta criar um arquivo, por exemplo, `helloworld.cc` no diretório `$EPOS/app`. Deste modo, a execução do comando acima resultará na imagem do sistema com a aplicação informada, pronta para ser carregada na plataforma alvo.

Programando no EPOS

A API de programação do EPOS é composta de dois tipos de componentes de software: abstrações e mediadores. As abstrações são classes C++ independentes de plataforma com interface e comportamento bem definidos. Suporte específico para plataformas é implementado através de mediadores de hardware, que são, funcionalmente, equivalentes aos *device drivers* do Unix, mas sem utilizar uma estrutura tradicional de HAL (*Hardware Abstraction Layer*). Ao invés disto, eles mantém um contrato de interface entre as abstrações e dispositivos de hardware através de técnicas de metaprogramação estática (C++ templates), o que implica na dissolução do código dos mediadores nas abstrações em tempo de compilação, eliminando sobrecustos de chamada de função. O EPOS ainda oferece uma série de componentes "utilitários", como implementações de estruturas de dados (lista, fila, mapa, etc).

A documentação da API do EPOS é mantida atualizada no sítio do projeto [LISHA 2010a].