UNIVERSIDADE FEDERAL FLUMINENSE

ANDRÉ RICARDO DE CARVALHO SARAIVA

Reduzindo a latência de comunicação em múltiplos saltos dos mecanismos de duty cycle assíncrono baseados em escalonamento através de sincronização de baixa resolução

> NITERÓI 2017

UNIVERSIDADE FEDERAL FLUMINENSE

ANDRÉ RICARDO DE CARVALHO SARAIVA

Reduzindo a latência de comunicação em múltiplos saltos dos mecanismos de duty cycle assíncrono baseados em escalonamento através de sincronização de baixa resolução

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Sistemas de Computação

Orientador: Diego Gimenez Passos, D.Sc.

> NITERÓI 2017

André Ricardo de Carvalho Saraiva

Reduzindo a latência de comunicação em múltiplos saltos dos mecanismos de *duty cycle* assíncrono baseados em escalonamento através de sincronização de baixa resolução

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Sistemas de Computação

Aprovada em Agosto de 2017.

BANCA EXAMINADORA

Prof. Diego Gimenez Passos, D.Sc. - Orientador, UFF

Prof. Célio Vinicius Neves de Albuquerque, PhD, UFF

Prof. Rodrigo de Souza Couto, D.Sc., UERJ

Niterói 2017

Dedico este trabalho a minha família, amigos e professores.

Agradecimentos

A Deus que sempre iluminou minha caminhada.

A minha esposa Cintia e filhas Maria Eduarda e Maria Fernanda pela paciência, apoio, colaboração e compreensão ao longo desta jornada, compartilhando comigo das minhas ausências, conquistas e dificuldades durante esses dois anos.

Ao meu orientador Professor Diego Passos pelo enorme conhecimento que compartilhou comigo, pelo estímulo e atenção, sendo responsável por grande parte do meu crescimento profissional e da minha formação acadêmica.

Aos professores, amigos de laboratório, secretaria e coordenação do curso de pósgraduação em computação e funcionários do Instituto de Computação.

A todos que me incentivaram, contribuindo de forma direta ou indireta, para a minha formação e desenvolvimento deste trabalho.

Por fim, ao CNPq, CAPES, Proppi/UFF por todos os recursos disponibilizados para que esse trabalho pudesse ser realizado.

Resumo

Nas Redes de Sensores Sem Fio, nós tipicamente operam por baterias não recarregáveis ou substituíveis. Logo, a otimização do consumo energético é uma das principais preocupações. Dentre as várias soluções já propostas, os métodos de duty cycle assíncrono baseados em escalonamento se mostram os mais simples por não demandarem mecanismos, protocolos ou hardwares específicos para sincronização de relógio entre os nós. Esses métodos, no entanto, resultam em alta latência para comunicação de múltiplos saltos. Neste trabalho, demonstramos como os mecanismos assíncronos já existentes podem se beneficiar de um baixo nível de sincronismo, em resolução de *slots*. Mostramos com simulações numéricas e modelos matemáticos simplificados que o uso de offsets específicos entre os relógios de nós vizinhos, de acordo com suas distâncias ao nó sorvedouro, reduz drasticamente a latência. Além disso, implementamos sobre a plataforma TinyOS um protótipo de protocolo de roteamento com as funcionalidades propostas de sincronização de baixa resolução. Com base nesta implementação, realizamos simulações de rede utilizando o simulador TOSSIM. Os resultados dessas simulações corroboram os resultados das demais avaliações, mostrando ganhos representativos em termos de latência de comunicação.

Palavras-chave: Rede de Sensores Sem Fio, Duty Cycle, Mecanismos Assíncronos.

Abstract

In Wireless Sensor Networks, nodes typically employ batteries that cannot be recharged or replaced. Hence, optimizing energy consumption is a major concern. Among the several solutions already proposed, schedule-based asynchronous duty cycle methods are the simplest because they do not require mechanisms, protocols or specific hardware for clock synchronization between nodes. These solutions, however, result in high latency for multi-hop communication. In this work, we show how existing asynchronous mechanisms can benefit from a low level of synchronism, with resolution of slots. We show, using numerical simulations and simple mathematical models, that the use of specific offsets between clocks of neighboring nodes according to their distances to the sink node drastically reduces latency. Additionally, we implemented a routing protocol prototype with the proposed low resolution synchronization functionalities for the TinyOS plattaform. With this implementation, we were able to run network simulations using the well-known TOSSIM simulator. The results of these simulations agree with the results from the other evaluation methods, consistently showing representative gains in terms of communication delay.

Keywords: Wireless Sensor Networks, Duty Cycle, Asynchronous Mechanisms.

Lista de Figuras

1.1	Componentes de um nó sensor. Adaptado de [2]	3
2.1	Esquemas semi-síncronos. Adaptado de [36]	11
2.2	Mapeamento de cada bloco do <i>Block Design</i> $\{7,3,1\}$ para um padrão de <i>slots</i> ativos e inativos. À direita, ilustra-se o fechamento para rotação, com a interseção entre padrões de dois blocos distintos resultando em exatamente um <i>slot</i> em comum	16
2.3	Dois nós A e B operando de acordo com o padrão gerado por um <i>Grid</i> de ordem 5, com um <i>offset</i> relativo de 1 <i>slot</i> . O padrão garante ao menos dois <i>slots</i> ativos comuns em cada ciclo, independentemente do <i>offset</i> entre os nós.	16
2.4	Dois nós A e B operando de acordo com o padrão gerado por um <i>Torus</i> de ordem 5, com um <i>offset</i> relativo de 1 <i>slot</i> . O padrão garante pelo menos um <i>slot</i> ativo em comum, independentemente do <i>offset</i> relativo entre os nós.	17
2.5	Dois nós A e B operando com o padrão gerado pelo mesmo Disco $(\{5,7\})$ e com um <i>offset</i> relativo de 1 <i>slot</i> . O padrão garante a existência de <i>slots</i> ativos comuns independentemente do <i>offset</i> relativo entre os nós	18
3.1	Exemplo de caminho de múltiplos saltos no qual cada nó opera segundo um padrão de <i>duty cycle</i> baseado no <i>Block Design</i> $\{7,3,1\}$. O nó S é o sorvedouro da rede. São ilustradas duas possíveis sincronizações: os padrões à esquerda possuem <i>offset</i> = 0, enquanto os padrões à direita têm um <i>offset</i> relativo de 1 <i>slot</i> entre nós consecutivos no caminho até o sorvedouro	21
3.2	Comparação da latência fim-a-fim prevista pelas Equações apresentadas nesse capítulo, com <i>offset</i> aleatório e igual a 1, por padrões baseados em <i>Block Design, Grid, Torus</i> e <i>Disco</i> variando-se o número de saltos para <i>duty cycles</i> aproximados de 1,03%	24

4.1	Comparação da latência de comunicação obtida nas simulações numéricas com offset aleatório e igual a 1 por padrões baseados em Block Design, Disco, Grid e Torus variando-se o número de saltos do caminho com $p = 1$ para duty cycles aproximados de 1,03%	28
4.2	Comparação da latência de comunicação obtida nas simulações com offset aleatório e igual a 1 por padrões baseados em <i>Block Design</i> , <i>Disco</i> , <i>Grid</i> e <i>Torus</i> variando-se o número de saltos do caminho com $p = 1$ para <i>duty</i> <i>cycles</i> de aproximadamente 1,2%	29
4.3	Comparação da latência de comunicação para 7 saltos obtida nas simulações com <i>offset</i> aleatório e igual a 1 por padrões baseados em <i>Block Design</i> , <i>Grid</i> , <i>Torus</i> e <i>Disco</i> variando-se a probabilidade de entrega do quadro (p)	30
4.4	Comparativo da latência dos 4 métodos de escalonamento como uma função da probabilidade de entrega de quadros de cada enlace de um caminho de 7 saltos, com <i>offset</i> =1, normalizados pela latência do <i>Block Design.</i>	31
5.1	Comparação do tempo médio de convergência obtido com <i>offset</i> aleatório e igual a 1 por padrões baseados em <i>Block Design</i> variando-se a distância entre os nós na topologia em grade e a área do cenário para a topologia aleatória	36
5.2	Comparação do tempo médio de convergência obtido com <i>offset</i> aleatório e igual a 1 por padrões baseados em <i>Disco</i> variando-se a distância entre os nós na topologia em grade e a área do cenário para topologia aleatória	37
5.3	Comparação do tempo médio de convergência obtido com <i>offset</i> aleatório e igual a 1 por padrões baseados em <i>Grid</i> variando-se a distância entre nós na topologia em grade e a área do cenário para topologia aleatória	38
5.4	Comparação do tempo médio de convergência obtido com <i>offset</i> aleatório e igual a 1 por padrões baseados em <i>Torus</i> variando-se a distância entre nós na topologia em grade e a área do cenário para topologia aleatória	39
5.5	Comparação do tempo médio de convergência obtido com <i>offset</i> igual a 1 e aleatório por padrões baseados em <i>Grid</i> 193×193 e em <i>Torus</i> 145×145 variando-se a distância entre nós na topologia em grade	40

Lista de Tabelas

4.1	Comparação dos valores obtidos com as simulações numéricas e a modela-							
	gem matemática para 7 saltos e $p=1.\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots$	27						
5.1	Tabela de parâmetros usados nas simulações com base em [57]	34						
5.2	Profundidade da árvore para as topologias em grade e aleatória	40						

Lista de Abreviaturas e Siglas

ack	:	Acknowledgement;
GHz	:	Giga-Hertz;
GPS	:	Sistema de Posicionamento Global;
IEEE	:	Instituto de Engenheiros Eletricistas e Eletrônicos;
IoT	:	Internet das Coisas;
ISM	:	Faixa de freqüência para aplicações industriais, científicas e médicas;
Kbytes	:	Quilobyte;
$\rm kb/s$:	Quilobits por segundo;
LEACH	:	Low Energy Adaptive Clustering Hierarchy;
LPL	:	Low Power Listening;
LPP	:	Low Power Probing;
mA	:	Miliampère;
MAC	:	Media Access Control;
MANET	:	Mobile Ad Hoc Network;
Mbps	:	Megabits por segundo;
MEMS	:	Sistemas Microeletromecânicos;
MHz	:	Mega-Hertz;
NDT	:	Neighbor Discovery Time;
RAW	:	Random Asynchronous Wakeup Protocol;
RF	:	Rádio Frequência;
RSSF	:	Redes de Sensores Sem Fio;
STEM	:	Sparse Topology and Energy Management;
TCP	:	Protocolo de controle de transmissão;
TDMA	:	Time Division Multiple Access;
UCP	:	Unidade Central de Processamento;
μA	:	Microampère;

Sumário

1	Intro	odução	1
	1.1	Organização do Texto	5
2	Rev	são Bibliográfica	7
	2.1	Características das RSSF	7
	2.2	Duty Cycle	8
		2.2.1 Principais Mecanismos Síncronos	9
		2.2.2 Principais Mecanismos Semi-Síncronos	0
		2.2.3 Principais Mecanismos Assíncronos	2
		2.2.4 Block Design	15
		2.2.5 Grid	6
		2.2.6 Torus	17
		2.2.7 Disco	17
3	Red	zindo a latência em múltiplos saltos através da sincronização de slots 1	9
	3.1	Modelagem matemática da latência fim-a-fim para alguns casos específicos	21
	3.2	Comparação qualitativa com mecanismos síncronos baseados em escalona- mento	24
4	Aval	iação Numérica 2	26
	4.1	Parâmetros da Simulação	26
	4.2	Resultados para p = 1 e múltiplos saltos $\dots \dots \dots$	27
	4.3	Resultados para 7 saltos e p variando	29

5	Avali	ação do Ambiente de Simulação	32
	5.1	TinyOS	32
	5.2	MicaZ	33
	5.3	Parâmetros e Ambiente da Simulação	33
	5.4	Outros Aspectos da Simulação	35
	5.5	Métricas de Avaliação	38
	5.6	Tempo de Convergência	38
6	Cond	clusão	42
	6.1	Trabalhos Futuros	43
Re	ferên	cias	44
Ap	êndic	e A – Instalando o Sistema TinyOS 2.1.2 no Ubuntu 16.01 ou Debian 8	50

Capítulo 1

Introdução

As Redes de Sensores Sem Fio (RSSF) foram viabilizadas e se tornaram uma arquitetura emergente com o desenvolvimento de tecnologias de comunicação sem fio, de equipamentos de sensoriamento e de sistemas microeletromecânicos (MEMS). O estímulo ao desenvolvimento de sensores faz com que o mundo assista a uma revolução no sensoriamento remoto, inclusive com o advento do ramo da Internet da Coisas (IoT) [40, 34], uma proposta promissora, que incentiva o desenvolvimento de novas tecnologias de RSSF para otimizar seu uso e operação. Todos estes fatores fazem com que a produção de sensores em larga escala e com custo acessível seja viável tecnológica e financeiramente.

A capacidade de monitoramento remoto, flexibilidade de configuração e manutenção, robustez, ausência de cabeamento físico e facilidade de implantação em locais de difícil acesso fazem com que a arquitetura de RSSF se torne uma grande candidata para utilização em diversas aplicações. De fato, as Redes de Sensores Sem Fio vêm sendo cada vez mais utilizadas [31] para funções diversas, sendo algumas apresentadas a seguir:

- Desenvolvimento de ambientes inteligentes e Internet das Coisas: Como uma das premissas da Internet das Coisas é a criação de ambientes inteligentes, um bom representante dessas aplicações são as Cidades Inteligentes, que utilizam os sensores para reduzir congestionamentos, prover rápida resposta dos serviços de emergência, entre diversas outras possibilidades.
- Aplicações de Segurança: Em aplicações de segurança, os sensores podem ser usados em ambientes domésticos ou na indústria. Em [26] são descritas aplicações para indústria de petróleo e gás, como o monitoramento de equipamentos em refinarias de petróleo e o monitoramento da sala de máquinas de navios tanque.

- Aplicações Militares: Em aplicações militares a utilização de redes de sensores pode ser caracterizada principalmente pelo fato de não ser possível definir uma infra-estrutura de comunicação em meio a uma operação de guerra. As aplicações podem incluir a detecção de movimentos inimigos, explosões, a presença de material perigoso como gás, agentes biológicos ou radiação. Estudos sobre o uso de sensores em guerras têm avançado, conforme pode ser visto em [35].
- Aplicações Ambientais: As RSSF também são úteis para monitorar condições ambientais, como temperatura e umidade de um ambiente [47]. Na agricultura, o uso das RSSF foi descrito por [22], onde adota-se um mecanismo de controle de temperatura em vinhedos.
- Aplicações Médicas: Sensores podem ser utilizados em hospitais e clinicas para monitorar os movimentos dos pacientes e equipes multidisciplinares de saúde ou controlar determinadas funções do corpo através de sensores extracorpóreos ou intracorpóreos. Também podem ser aplicados para detectar a presença de substâncias, sinais ou alterações que indicam a presença ou surgimento de um problema médico, seja no corpo humano ou animal, como o aumento das taxas de açúcar presentes no sangue, em pacientes diabéticos. Em [23] relata-se o uso de redes de sensores para monitorar a pressão arterial e temperatura dos pacientes.
- Outras Aplicações: Existem diversas outras aplicações possíveis para as redes de sensores sem fio, como no monitoramento de aeronaves radiocontroladas [14], em entretenimento [43], em robótica móvel [39], em redes de sensores sem fio aquáticas [1] e na manutenção preventiva de máquinas [26]. Também existe a utilização de RSSF nas usinas nucleares [29] para monitorar a temperatura e vibração do reator, ruído do ambiente, radiação entre outros parâmetros necessários para que a segurança desses ambientes se torne possível.

Uma Rede de Sensores Sem Fio é caracterizada pelo uso de sistemas embarcados sem fio de baixa potência, com a capacidade de sensoriamento, processamento, atuação e comunicação. Em geral, as RSSF possuem baixos custos, os sensores possuem tamanho reduzidos, operam sem intervenção direta do homem e devem possuir mecanismos para auto-gerenciamento devido ao fato de, muitas vezes, serem usadas em áreas remotas. As RSSF podem ser caracterizadas como redes *ad hoc* [31].

As RSSF podem ser classificadas em relação à composição, organização, mobilidade, densidade e distribuição dos nós sensores [31]. Alguns sensores possuem características diferentes, como um nó sorvedouro com maior poder computacional, podendo ser mais utilizado que outros nós, já que a tendência é o tráfego de dados fluir dos demais nós para este sorvedouro.

Os sensores normalmente apresentam limitações de memória, processamento e severas restrições de energia. Conforme mostra a Figura 1.1, basicamente um sensor é composto de 4 componentes principais (outros três componentes adicionais, dependentes de aplicação, são ilustrados na figura) [2], a saber: uma unidade de sensoriamento, uma unidade de processamento, uma unidade de comunicação e uma unidade de energia.



Figura 1.1: Componentes de um nó sensor. Adaptado de [2].

A unidade de sensoriamento é composta de duas sub-unidades, os sensores propriamente ditos e os conversores Analógico/Digital. Sinais analógicos gerados pelos sensores são convertidos para sinais digitais pelo conversor A/D e utilizados como entrada para a unidade de processamento. A unidade de processamento geralmente é associada a uma unidade de armazenamento de dados (memória), e é responsável por controlar os procedimentos necessários para que os nós sensores possam realizar o processamento das tarefas associadas ao sensoriamento. A unidade de energia é responsável por alimentar todos os componentes do nó e é composta por células de energia limitadas, o que torna a eficiência energética uma grande área de pesquisa em RSSF. Quanto à unidade de comunicação, ela é responsável pela comunicação entre os elementos da RSSF, e é considerada uma das que mais demanda consumo de energia [3, 4].

Em geral a comunicação entre os nós é feita através de interfaces de RF (rádio frequência). Uma das tecnologias mais utilizadas para comunicação entre sensores é o IEEE 802.15.4, que pode operar em três faixas de frequência: 868 MHz com 1 canal, 915 MHz com 10 canais e 2.4 GHz com 16 canais. Todas estas são faixas ISM (*Industrial, Scientific and Medical*) nas quais não há necessidade de licenciamento para utilização da frequência. Segundo [5], o uso do *ZigBee*, baseado no padrão IEEE 802.15.4, é particularmente popular em RSSF.

Uma Rede de Sensores Sem Fio não exige uma infra-estrutura, podendo a posição de cada nó sensor ser predeterminada, como em grade, ou aleatória. A implantação de uma rede de sensores depende da aplicação a ser atendida, da topologia em que estes nós se encontram e da própria capacidade de comunicação de cada nó. Esta capacidade de comunicação pode ser prejudicada quando a fonte de energia que alimenta este nó atingir cargas insuficientes para tal.

As baterias que alimentam os sensores possuem um tempo de vida limitado e sua troca, na maioria das vezes, não é trivial ou mesmo possível (especialmente quando estas redes são implantadas em locais de difícil acesso). Ainda que as RSSF levem vantagem em diversas aplicações sobre as redes cabeadas, existem vários desafios a serem superados por esta tecnologia, principalmente em relação à eficiência energética dos nós sensores. Sendo assim, uma utilização otimizada das interfaces de rádio deve ser priorizada pelos protocolos de comunicação e gerenciamento da rede, tornando esses protocolos um vasto campo de pesquisa, já que, segundo [55], a execução de 3000 instruções gasta a mesma quantidade de energia que enviar 1 bit a 100 metros via rádio. Balancear os recursos limitados pode aumentar o tempo de vida da rede, como visto em trabalhos sobre diversos aspectos das RSSF, como protocolos de roteamento [61], topologia e arquitetura da rede [15] entre outros.

Em particular, o *duty cycling* da interface de rádio se torna necessário [3, 4], alternando períodos de atividade e inatividade. Entretanto, fazer com que todos os elementos de uma RSSF mantenham de forma coordenada seus períodos de atividade e inatividade, a fim de garantir que um nó sempre encontre um nó vizinho para transmitir seus dados com sucesso requer mecanismos de coordenação. Tais mecanismos podem ser síncronos, com um relógio de tempo comum compartilhado por todos os nós, semi-síncronos, nos quais os nós são agrupados em *clusters* ou assíncronos, onde um relógio comum não se faz necessário.

Mecanismos síncronos comumente requerem *hardware* adicional, geram um maior tráfego de controle e resultam em maiores custos [9]. Em contrapartida, mecanismos assíncronos não demandam *hardware* especializado ou a adição de grande volume de tráfego de controle, se mostrando, portanto, uma alternativa interessante no contexto de redes de sensores. Porém, segundo [62, 32], os mecanismos assíncronos apresentam uma latência excessiva ao longo de caminhos de múltiplos saltos. Mesmo assim, diversas propostas assíncronas baseadas em escalonamento podem ser encontradas na literatura, como *Block* Design [64], Grid [24], Torus [56] e Disco [16], embora elas não abordem a questão da elevada latência fim-a-fim.

O problema de *sleep waiting* e o problema de *data forwarding interruption* [32] são exemplos de fenômenos que contribuem para o aumento da latência, uma vez que um transmissor deverá aguardar certo tempo até que ele e o receptor pretendido fiquem com seus rádios ligados simultaneamente (o que constitui uma oportunidade de encontro). Estes fenômenos fazem com que o tempo de descoberta de vizinho (ou NDT, do inglês *Neighbor Discovery Time*) seja elevado.

Com base nessas características, neste trabalho apresenta-se uma análise dos benefícios de um certo grau de sincronismo de baixa resolução em mecanismos de *duty cycle* assíncrono, sem que haja a necessidade de *hardware* adicional ou aumento significativo de tráfego de controle, mas objetivando uma redução representativa da latência. Através do uso de simulações numéricas e modelos matemáticos simplificados, este trabalho demonstra que a utilização de certos valores específicos de *offset* entre os escalonamentos utilizados por nós vizinhos reduz consideravelmente a latência da comunicação em múltiplos saltos. Além disso, como forma de prova de conceito, implementou-se, em ambiente de simulação próprio para RSSF, um protótipo para sincronização dos nós em resolução de *slots* que permita que nós vizinhos sigam escalonamentos com *offsets* específicos (ao invés dos *offsets* aleatórios resultantes da natural dessincronização dos relógios dos nós). Com base neste protótipo, foi possível executar simulações de redes em cenários variados. Nos resultados encontrados para estas simulações, verificou-se que o uso do *offset* fixo proposto reduz a latência em até 45,65% para uma rede com características topológicas aleatória utilizando o método *Grid*.

1.1 Organização do Texto

Este trabalho está organizado da seguinte maneira. O Capítulo 2 introduz algumas características de RSSF e uma revisão bibliográfica dos principais dos mecanismos de *duty cycle* síncronos e assíncronos. O Capítulo 3 discute os benefícios de uma sincronização de *slots* na redução da latência para comunicações em múltiplo saltos. O Capítulo 4 discute a avaliação numérica dos conceitos introduzidos no Capítulo 3. O Capítulo 5 descreve os parâmetros utilizados nas simulações de rede, a metodologia empregada para avaliar o desempenho da proposta, as métricas selecionadas para comparar o desempenho e, por fim, discute os resultados obtidos. Finalmente, no Capítulo 6 são apresentadas as conclusões obtidas neste trabalho e as direções para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Neste capítulo apresentaremos uma breve revisão da bibliografia pertinente a esta dissertação. Começaremos descrevendo características gerais das RSSF e, em seguida, definiremos *duty cycle*, classificaremos os mecanismos síncronos, semi-síncronos e assíncronos.

2.1 Características das RSSF

Um conceito muito utilizado em RSSF é o de *cluster* [6]. Nesta organização, os nós que compõem a RSSF são classificados em três tipos: nós sensores (escravos), nós cabeça (*cluster head*) e os sorvedouros. Os nós sensores são todos os nó que pertencem a um *cluster* e são controlados por um nó cabeça, que tem como uma de suas funções manter a comunicação com outros *clusters* ou com o sorvedouro. O sorvedouro, por sua vez, é um nó de grande importância em Redes de Sensores Sem Fio, pois faz a conexão da RSSF com uma rede externa.

Como o sorvedouro é vital para toda a RSSF, se este falhar toda rede estará comprometida. Desta forma, algumas considerações importantes sobre a RSSF incluem:

- Tolerância a Falhas: a falha em um dos nós sensores, devido à falta de energia ou danos físicos, não deve atrapalhar o funcionamento normal de toda a rede. Assim os protocolos e mecanismos utilizados nestas redes devem ser projetados levando em consideração a tolerância a falhas.
- Escalabilidade: a RSSF deve ser flexível o suficiente para suportar uma grande quantidade de nós sensores, bem como a inclusão de novos nós.
- Topologia: as RSSF são projetadas para funcionar em áreas extensas e geralmente

de difícil acesso ou mesmo em áreas inacessíveis que impedem possíveis manutenções. Assim a topologia deve levar alguns requisitos básicos em consideração, como, por exemplo: redução do custo na instalação, eliminação da necessidade de um préplanejamento, aumento na escalabilidade da rede e tolerância a falhas.

- Restrição de Hardware e Tecnologia de Transmissão: as restrições de hardware impostas nestes equipamentos se dão, entre outros fatores, pela pequena capacidade de processamento. Por exemplo, um sensor do tipo MicaZ [13] possui microcontrolador ATmega128L e 512 Kbytes de memória *flash*. Além disso a fonte de energia é composta de baterias com carga limitada. As tecnologias de transmissão utilizadas devem atender às restrições de hardware e restrições físicas. Dependendo da aplicação, uma tecnologia pode ser escolhida a partir de suas características. Em [12], os autores comparam duas tecnologias de comunicação populares para este tipo de rede, *ZigBee* e *Bluetooth*.
- Eficiência Energética: as RSSF são projetadas com dispositivos com baixo consumo de energia. Mesmo assim, como citado anteriormente, a fonte de energia é composta de baterias com carga limitada. Assim, é importante a utilização de mecanismos que colaborem para a redução do consumo energético fazendo com que a rede tenha um período de vida maior.

2.2 Duty Cycle

Mecanismos de *duty cycling* de rádio têm como objetivo reduzir o tempo de escuta ociosa (i.e., intervalos em que o rádio se encontra ligado sem que haja transmissões ou recepções de quadros de interesse) dos nós de uma rede sem fio, reduzindo assim o consumo de energia e aumentando a vida útil da rede. Na literatura da área, define-se o*duty cycle*de um nó como o percentual de tempo que esse mantém seu rádio ligado [10]. Considerando os requisitos e capacidades atuais dos nós sensores,*duty cycles*inferiores a 1% são desejáveis [10]. Em contrapartida, a realização de*duty cycling*do rádio não é trivial, já que algum nível de coordenação é necessário para garantir que quaisquer dois nós vizinhos ainda tenham oportunidades de comunicação <math>(i.e., que exista algum período suficientemente longo em que os nós tenham seus rádios simultaneamente ligados).

O NDT é uma medida de desempenho de mecanismos de *duty cycle* definida como o tempo transcorrido a partir do momento em que o nó transmissor decide enviar uma mensagem para o receptor (o que pode, inclusive, ocorrer quando o nó transmissor está com seu rádio desligado) até o momento no qual efetivamente um *beacon* é recebido com sucesso pelo nó receptor, dada a comunicação entre quaisquer dois nós que operem sob um mesmo esquema de *duty cycle* [9]. A redução do *duty cycle* dos nós de uma rede geralmente é associada a um aumento do NDT [9].

Por estes motivos, a comunidade acadêmica tem realizado esforços para obter soluções viáveis com *duty cycles* baixos e, idealmente, baixo *overhead* de controle [64].

2.2.1 Principais Mecanismos Síncronos

Como forma de sincronizar o período de atividade dos rádios, reduzindo a latência, a escuta ociosa e a perda de pacotes, os esquemas síncronos introduzem um relógio comum entre todos os nós da rede. O estabelecimento deste relógio comum demanda a utilização permanente de protocolos de sincronização de relógio (já que, mesmo após uma sincronização inicial, o escorregamento dos relógios introduz falhas de sincronização) ou a adição de *hardware* especializado para manter a sincronização com o grau de precisão necessário. Esses mecanismos podem ser classificados em duas famílias: os esquemas estritamente síncronos, ou baseados em *Rendezvous* [48], e os esquemas baseados em escalonamento [3].

Os esquemas estritamente síncronos são os de mais fácil compreensão, uma vez que todos os nós deverão ter seus rádios ligados ou desligados ao mesmo tempo. No entanto, em múltiplos saltos esse esquema torna-se complexo, já que erros de sincronização entre os nós são comuns e tendem a se acumular ao longo dos saltos, dificultando a sincronização da rede como um todo e, muitas vezes, demandando elementos adicionais de *hardware* e certa quantidade de mensagens de controle [48].

A maioria das propostas de *duty cycling* para esquemas estritamente síncronos é projetada para ser incorporada na camada MAC [4]. Exemplos de esquemas estritamente síncronos encontrados na literatura são o RT-Link [44], que acrescenta um GPS para sincronização, e o TRAMA [42], que tem como princípio a eliminação das colisões e a suspensão dos nós que não participam da comunicação naquele instante de tempo. Ambos utilizam o protocolo de acesso múltiplo TDMA, do inglês *Time Division Multiple Access*, que é um protocolo livre de contenção, onde o tempo é dividido em *slots* atribuídos a nós, evitando colisões entre pacotes, transmissões desnecessárias e escutas ociosas, e consequentemente desperdício de bateria, características essas que estão entre as principais vantagens desses esquemas. Outras propostas são os esquemas baseados em escalonamento, que visam resolver o problema de interrupção de encaminhamento de dados (do inglês *Data Forwarding Interruption Problem*) – que segundo [32] ocorre quando o percurso de um pacote pela rede é interrompido devido ao próximo nó no caminho tornar seu rádio inativo – e reduzir o tempo de espera ociosa, uma vez que se pretende manter um sincronismo para que um nó não fique ativo prematuramente, aguardando até que seu nó vizinho se ative. Essas soluções utilizam uma topologia em árvore, cuja raiz é o nó sorvedouro e os demais nós são ativados em instantes de tempo determinados de acordo com sua profundidade [32, 37].

Outros esquemas desta família utilizam escalonamentos mais genéricos para determinar os períodos de atividade e inatividade dos nós. Em [59], propõe-se um mecanismo de coloração de grafo, no qual os nós agendam seu despertar de acordo com as cores atribuídas. Esta proposta garante que um nó sempre tenha pelo menos um vizinho operando sob a mesma cor (ou escalonamento).

Diversos outros esquemas baseados em escalonamento foram propostos na literatura, entre eles o SPEED-MAC [11], que introduziu um período de sinalização para notificação de eventos e preparo dos nós para transmitirem os dados recebidos de comunicações prévias, e o CUPID [27], que propõe um esquema bidirecional para o tráfego na árvore, dando suporte não só à existência de fluxo de dados dos sensores para o nó sorvedouro, mas também a um fluxo de configuração do nó sorvedouro para os sensores.

Uma dificuldade dos esquemas baseados em escalonamento é que, mesmo sendo menos restritivos em relação ao nível de sincronização dos nós como um todo (basta garantir um nível de sincronização entre nós vizinhos), ainda é necessária uma sincronização de relógio com alta resolução, o que vem associado à adição de *hardware* especializado e/ou alto *overhead* de sincronização. Além disso, no caso da entrada de novos nós na rede, estes deverão estar previamente sincronizados com o restante da rede (solução mais complexa) ou deverão entrar em um estado inicial de sincronização no qual seus rádios permanecerão constantemente ligados, induzindo um alto consumo energético até a sincronização.

2.2.2 Principais Mecanismos Semi-Síncronos

Nos mecanismos semi-síncronos [10], os nós são agrupados formando *clusters*. Dentro de um *cluster*, todos os sensores são sincronizados. Mas não se mantém sincronia de relógio entre diferentes *clusters*: esses *clusters* se comunicam com os outros grupos de forma assíncrona. Conforme observado na Figura 2.1, os sensores de um *cluster* se comunicam com o nó cabeça dentro do seu próprio *cluster*, mas apenas o nó cabeça realiza a comunicação com o sorvedouro. Essas propostas tentam utilizar os mecanismos síncronos e assíncronos em conjunto da melhor forma possível. A principal vantagem de utilizar essa abordagem é que a sincronização entre os nós vizinhos é mais fácil de alcançar do que a sincronização de toda a rede.

Em contrapartida, para eleger o nó cabeça pode ser necessário o uso de mecanismos de eleição de líder [21, 61] que exigem maior tráfego de controle e também podem ser inadequados para topologias dinâmicas. Também é possível, como uma consequência da sincronização dos nós, a criação espontânea de um *cluster*, conforme [62, 58, 63]. Esses mecanismos, portanto, podem ser classificados em duas famílias: esquemas de agrupamento espontâneo e os esquemas de eleição de nó cabeça.



Figura 2.1: Esquemas semi-síncronos. Adaptado de [36].

Os esquemas baseados em agrupamento espontâneo possuem requisitos de sincronização menos rigorosos. *Clusters* virtuais se formam espontaneamente à medida que cada nó transmite, em *broadcast*, seu escalonamento aos seus vizinhos. Se um nó X receber o escalonamento do seu vizinho Y antes de decidir seu próprio escalonamento, o nó X adota o escalonamento do nó Y. Se o nó X receber um escalonamento diferente do adotado por ele, ele adotará os dois escalonamentos, uma característica muitas vezes criticada como energeticamente ineficiente [10].

Alguns esquemas baseados em agrupamento espontâneo encontrados na literatura são o S-MAC (*SensorMAC* [62]) e o T-MAC (*Timeout-MAC* [58]). O S-MAC utiliza slots de tempo da ordem de 0,5 segundos e baseia sua sincronização apenas na troca de timestamps entre vizinhos diretos. Por esse motivo, as exigências de sincronização desse esquema são consideradas baixas. Problemas como o longo período de inatividade e o duty cycle elevado do S-MAC permitiram o surgimento de propostas como o T-MAC [58] que desativa dinamicamente o rádio de um nó sempre que o tráfego ao seu redor termina, minimizando a escuta ociosa e consequentemente o consumo de energia. Segundo [58], o T-MAC e S-MAC obtêm reduções de até 98% no consumo de energia. No entanto, devido à otimização de desligar o rádio na ausência de tráfego na vizinhança, o T-MAC pode resultar no problema do sono precoce, que ocorre quando um nó desliga seu rádio enquanto seu vizinho ainda possui algum fluxo de dados para ele.

Um das dificuldades encontradas por estas propostas é a eleição do nó cabeça. Além disso, encontrar mecanismos de coordenação intragrupo e formas eficientes de transmissão de tráfego intergrupos não é trivial, podendo resultar em tráfego de controle significativo [10]. Propostas como o GAF [61] visam facilitar a formação de *clusters* através uso de um sistema de coordenadas fornecidas por um GPS, o que aumenta o custo e o consumo de energia dos nós da rede.

O LEACH (*Low-Energy Adaptive Clustering Hierarchy* [21]) é um protocolo baseado em *clusters*, que utiliza a alternância aleatória dos nós cabeça para garantir a uniformidade do consumo de energia da rede. O LEACH usa uma coordenação localizada, na qual o nó cabeça coordena a atividade dos membros do *cluster* e incorpora a fusão do tráfego de dados para reduzir a quantidade de informações que devem ser transmitidas, melhorando assim a eficiência energética.

2.2.3 Principais Mecanismos Assíncronos

Diferentemente dos mecanismos síncronos, os mecanismo assíncronos são menos complexos, em termos de adição de *hardware*, mensagens de controle e custo. Por esta razão, a tendência ao assincronismo tem ganhado espaço na comunidade científica, em especial com os mecanismos baseados em escalonamento. Porém, problemas como a latência e a escuta ociosa continuam como uma preocupação.

O mecanismo de amostragem de preâmbulo [41], às vezes chamada de LPL (*Low Power Listening*) no contexto das redes de sensores, faz com que cada nó fique inativo de forma assíncrona, ativando-se periodicamente para escutar o meio. A ideia é reduzir a escuta ociosa, transferindo o gasto de energia para o transmissor. Assim, se ao escutar o meio, o nó receptor detecta um preâmbulo, esse permanece com seu rádio ligado para receber o quadro completo. Para evitar que um descasamento entre o período de transmissão do preâmbulo e o período de atividade do receptor, os preâmbulos são bastante longos, maiores que os períodos de atividade e inatividade combinados. As primeiras técnicas de amostragem de preâmbulo foram o B-MAC [41] e WiseMAC [17]. Porém, como um

longo preâmbulo implica problemas como o aumento do consumo de energia e do *overhead* de acesso ao meio, o X-MAC [7] foi proposto como uma técnica que substituiu o uso de preâmbulos longos por uma sequência de quadros curtos de sinalização transmitidos em curtos intervalos de tempo. Como vantagem, pode-se citar a possibilidade da rápida interrupção da sequência de sinalização quando um receptor estiver apto a receber a transmissão, o que reduz o consumo energético e o *overhead* de acesso ao meio para cada transmissão. Além disso, é possível inserir o endereço do receptor no preâmbulo, permitindo que os nós que não sejam o receptor voltem à inatividade.

Os autores em [54] classificam alguns mecanismos baseados na iniciativa do receptor. Nesses métodos, o transmissor aguarda um sinal do receptor, indicando que esse está apto a receber o dado. Nesse caso, novamente, a demanda energética é mais alta para o transmissor, que deve aguardar com seu rádio ligado até que o receptor acorde.

A ideia inicial das transmissões iniciadas pelo receptor era de um mecanismo para evitar de colisões [19]. Posteriormente foi descrito em [28] como uma proposta de LPP (*Low Power Probing*) onde cada nó envia um *beacon* no momento em que se encontra acordado e volta a dormir a menos que uma confirmação seja recebida para se manter acordado. Assim, o RI-MAC [54] propôs a comunicação iniciada pelo receptor como uma maneira de acordar os nós para configurar a rede.

O On-demand wakeup [46] assume a existência de uma segunda interface de rádio de baixo consumo que permanece sempre ligada através da qual o nó receptor pode ser avisado de que deve ligar sua interface principal para a transmissão de dados. Esta comunicação fora de banda faz com que a interface secundária de rádio envie uma interrupção para a UCP (Unidade Central de Processamento) que irá ativar a interface principal de rádio para a efetiva comunicação entre os nós. O mecanismo de On-demand wakeup pode ser vantajoso, porém manter uma segunda interface de rádio, por mais que seu consumo de energia seja baixo, ainda consome energia. Propostas de mecanismos On-demand podem ser encontradas na literatura, como em [46], que propõe o STEM (Sparse Topology and Energy Management). Nesta proposta, quando um nó possui dados para enviar, ele transmite uma sequência de beacons através da interface secundária e aguarda o receptor ativar o seu rádio e enviar uma confirmação de ativação. Em seguida, o transmissor envia os dados através da interface principal de rádio.

O *duty cycling* aleatório [38] adequa-se a RSSF densas. Os nós ligam e desligam seus rádios aleatoriamente, sem coordenação prévia. Quando um nó deseja transmitir um quadro, ele o faz independentemente de quais outros nós estejam ativos naquele momento.

Se a rede for densa o suficiente, há alta probabilidade de que pelo menos um receptor receba o quadro transmitido. O método, portanto, assume que as transmissões na camada de enlace não são endereçadas a um próximo salto específico. Um representante desta classe é o RAW (*Random Asynchronous Wakeup Protocol*) [38] que propõe um esquema de ativação aleatória em que o tempo de atividade de um nó seria inversamente proporcional ao número de seus vizinhos.

Finalmente, existem os mecanismos assíncronos baseados em escalonamento, tais como *Block Design* [64], *Grid* [24], *Torus* [24] e *Disco* [16], que são caracterizados por dividir o tempo em ciclos, por sua vez subdivididos em *slots* ativos e inativos. O grande diferencial destas propostas é que os padrões de *slots* ativos e inativos que compõem um ciclo são especialmente projetados com o intuito de garantir ao menos uma sobreposição de *slots* ativos por ciclo entre quaisquer dois nós, independentemente do *offset* relativo entre seus respectivos relógios locais (isto é, a diferença entre os relógios dos nós), uma característica conhecida na literatura como fechamento para rotação (do inglês, *rotation closure*) [9]. Ao contrário dos demais mecanismos assíncronos discutidos até aqui, estes mecanismos não assumem a existência de um segundo rádio de baixo consumo, nem requerem a escuta ao meio por um sinal ou uma grande densidade de nós. Para introduzirmos mais formalmente os mecanismos assíncronos baseados em escalonamento, observamos algumas definições dadas em [9] reproduzidas aqui por simplicidade:

- Dado um ciclo de *n slots* numerados de 1 a *n*, um escalonamento em *n*, denotado por $S[n] = \{a_0, ..., a_{m-1} \in [0, ..., n-1]\}$, é um conjunto de *slots* ativos em um ciclo.
- Um esquema em n, denotado por $\chi[n] = \{S[n]_1, S[n]_2, ..., S[n]_Z\}$ é um conjunto de escalonamentos em n.

Por exemplo, existe um esquema $\chi[5]$ de 5 *slots*, sendo 3 ativos, dado por: $\chi[5] = \{\{0, 1, 2\}, \{1, 2, 3\}, \{2, 3, 4\}, \{3, 4, 0\}, \{4, 0, 1\}\}.$

Observando o exemplo anterior, é possível definir um esquema simétrico como aquele onde todos os escalonamentos têm a mesma cardinalidade.

• A rotação de um escalonamento $S[n] = \{a_0, ..., a_{m-1}\}$ por $i \in \mathbb{Z}$, denotada por $\rho(S[n], i)$, é também um escalonamento $S[n]' = \{(a_k + i) \mod n, \forall a_k \in S[n]\}.$

Assim, o exemplo do escalonamento $S[5] = \{1, 2, 3\}$ citado anteriormente apresenta como uma possível rotação $S[5]' = \rho(S[5], 1) = \{2, 3, 4\}.$

• Um esquema $\chi[n]$ apresenta **fechamento para rotação** se e somente se $\forall S[n]_i, S[n]_j \in \chi[n] : S[n]_i \cap \rho(S[n]_j, k) \neq \emptyset, k = 0, ..., n - 1.$

Por exemplo, observa-se a aderência à propriedade de fechamento para rotação pelo esquema $\chi[4]_a = \{\{0, 1, 2\}, \{1, 2, 3\}, \{0, 2, 3\}\}.$

2.2.4 Block Design

Um Block Design [64] $\{v, k, \lambda\}$ é uma entidade matemática composta por um conjunto V de v elementos inteiros, juntamente com uma família de v subconjuntos de V chamados blocos, cada um com exatamente k elementos, tal que a interseção entre quaisquer dois blocos diferentes tenha exatamente λ elementos. No caso particular em que $\lambda=1$, os Block Designs recebem o nome de planos projetivos.

Um bloco de um *Block Design* pode ser mapeado para um padrão de *slots* ativos e inativos da seguinte forma. Cada *slot* do padrão corresponde a um elemento em V. Para cada elemento em V, se este pertence ao bloco, então ativa-se o *slot* correspondente no padrão. Caso contrário, o *slot* correspondente será inativo. A Figura 2.2 apresenta um exemplo com dois nós seguindo padrões baseados em blocos diferentes do *Block Design* $\{7,3,1\}$ onde temos V= $\{0,1,2,3,4,5,6\}$, k = 3, resultando nos seguintes blocos: $\{[0,1,3], [1,2,4], [2,3,5], [3,4,6], [4,5,0], [5,6,1], [6,0,2]\}$.

Note que os padrões gerados por cada bloco são simplesmente rotações dos padrões de outros blocos do mesmo *Block Design*. Logo, se dois nós de uma rede seguem um padrão resultante de um mesmo bloco, porém possuem seus relógios dessincronizados, isto é equivalente a cada nó utilizar um bloco diferente do mesmo *Block Design* sob uma base de tempo global. Com isso, os *Block Designs* garantem a existência de ao menos λ slots ativos coincidentes por ciclo, independentemente do offset relativo entre os relógios dos nós.

O duty cycle para um Block Design é dado por $\frac{k}{v}$. Em particular, é possível demonstrar que planos projetivos constituem os padrões de menor duty cycle com fechamento para rotação para um dado tamanho de ciclo [33].

Em [9], é derivada uma expressão que aproxima o NDT esperado para um *Block* Design $\{v, k, \lambda\}$ em função de p, a probabilidade de entrega de quadros do enlace:

$$E[NDT] = \frac{v+1}{p(\lambda+1)} - \frac{(v+1)(1-p)^{\lambda} - (\lambda+1)}{(\lambda+1)[(1-p)^{\lambda} - 1]}$$
(2.1)



Figura 2.2: Mapeamento de cada bloco do *Block Design* $\{7, 3, 1\}$ para um padrão de *slots* ativos e inativos. À direita, ilustra-se o fechamento para rotação, com a interseção entre padrões de dois blocos distintos resultando em exatamente um *slot* em comum.

2.2.5 Grid

O Grid [24] é um dos mecanismos baseados em sistemas de quórum. O Grid define uma lei de formação de padrões de slots ativos e inativos que, assim como os Block Designs, garante o fechamento para rotação. A Figura 2.3 mostra um exemplo de um padrão gerado por um Grid 5x5. Cada nó seleciona uma linha e uma coluna em uma matriz $n \times n$. Todos os slots da linha e da coluna selecionadas serão ativos, enquanto os demais slots serão inativos. O padrão resultante é obtido através da linearização da matriz, *i.e.*, o padrão é formado pela sequência de slots da primeira linha, seguidos dos slots da segunda e assim sucessivamente, resultando em um ciclo de n^2 slots. Mesmo o método funcionando com a escolha de linhas e colunas diferentes por cada nó, na prática comumente todos os nós escolhem as mesmas linha e coluna, resultando no mesmo padrão linearizado, embora, possivelmente, em offsets diferentes por conta da falta de sincronização, conforme ilustrado na Figura 2.3 (A e B seguem o mesmo padrão, mas com um offset relativo de um slot).



Figura 2.3: Dois nós A e B operando de acordo com o padrão gerado por um *Grid* de ordem 5, com um *offset* relativo de 1 *slot*. O padrão garante ao menos dois *slots* ativos comuns em cada ciclo, independentemente do *offset* entre os nós.

O duty cycle no Grid é dado por $\frac{2n-1}{n^2}$, onde *n* é a ordem da matriz. A Equação 2, extraída de [9], fornece uma estimativa do NDT esperado para um Grid de ordem *n* na comunicação por um enlace com probabilidade de entrega de quadros *p*:

$$E[NDT] = \frac{(3-p)n^2}{6p}$$
(2.2)

2.2.6Torus

O Torus [24], outro método da família dos sistemas de quórum, é uma proposta de melhoria do Grid em termos de duty cycle. Embora sua lei de formação de padrões seja similar à do Grid, ao invés de se ativarem todos os slots da linha selecionada, são ativados apenas metade mais um desses, resultando em um padrão de comprimento n^2 , mas com menos slots ativos em comparação ao Grid. A Figura 2.4 exemplifica essa lei de formação, mostrando os *slots* ativos coincidentes para dois nós com *offset* relativo de um *slot*.

O duty cycle de un padrão resultante de un Torus de ordem n é dado por $\frac{3}{2n}$, para n par, ou $\frac{3n-1}{2n^2}$, para valores de n ímpares. Já o NDT esperado, segundo [9], é dado por:

$$E[NDT] = \frac{(2-p)n^2}{2p}$$

$$(2.3)$$

$$B = 1$$

$$B$$

N 2 2

Figura 2.4: Dois nós A e B operando de acordo com o padrão gerado por um *Torus* de ordem 5, com um offset relativo de 1 slot. O padrão garante pelo menos um slot ativo em comum, independentemente do offset relativo entre os nós.

2.2.7Disco

O Disco [16], um mecanismo baseado em números primos, garante que se os slots ativos de dois nós A e B são múltiplos de q_1 ou de q_2 (os números primos que são parâmetros do método), sempre ocorrerá uma sobreposição de *slots* ativos, independentemente dos offsets relativos de seus relógios locais. A Figura 2.5 exemplifica a lei de formação do Disco, mostrando os slots ativos como múltiplos de um dos dois números e que mesmo com *offset* relativo, a sobreposição de *slots* ocorre.

Segundo [9], o NDT esperado resultante da utilização do *Disco* com os parâmetros q_1 e q_2 é aproximadamente dado por:

$$E[NDT] = \frac{q_1 q_2 (p^2 - 3p + 3)}{3p(2 - p)}$$
(2.4)



Figura 2.5: Dois nós A e B operando com o padrão gerado pelo mesmo Disco $(\{5,7\})$ e com um *offset* relativo de 1 *slot*. O padrão garante a existência de *slots* ativos comuns independentemente do *offset* relativo entre os nós.

Capítulo 3

Reduzindo a latência em múltiplos saltos através da sincronização de slots

A proposta desenvolvida neste trabalho [45] se baseia na possibilidade de escalonar o uso do rádio de acordo com a distância de cada nó ao sorvedouro (em número de saltos) e não aleatoriamente, com base no momento em que o mesmo é ligado e em outros fatores que resultam na dessincronização dos nós. A distância de cada nó em relação ao sorvedouro ou a própria árvore de menores caminhos entre os sensores e o sorvedouro são tipicamente obtidas pelos protocolos de roteamento. Logo, esse escalonamento pode ser feito, por exemplo, através de adição da informação do *slot* de tempo atual de um nó nos seus pacotes de controle relativos ao próprio protocolo de roteamento. Ao receber um destes pacotes de um dado vizinho, o nó verificaria se esse vizinho é, de acordo com as informações atuais de roteamento, seu próximo salto em direção ao sorvedouro. Em caso afirmativo, o nó alteraria seu *slot* de tempo atual para o *slot* atual do vizinho somado, possivelmente, de algum *offset* configurável. O *offset* irá estabelecer uma rotação entre o padrão de *duty cycle* no qual um nó deverá operar em relação aos seus vizinhos.

Assumindo que esse tipo de sincronização seja viável, a pergunta imediata passa a ser: qual é o valor ideal de *offset* entre os vizinhos? Uma primeira análise poderia sugerir que o valor ideal fosse 0, ou seja, que todos os nós da rede operassem exatamente sob o mesmo escalonamento. De fato, o emprego do *offset* = 0 é certamente ótimo para uma comunicação de um único salto, já que este *offset* maximiza o número de *slots* ativos coincidentes entre dois nós. No entanto, para o caso de comunicação em múltiplos saltos e sob hipóteses razoáveis na prática, isso não é necessariamente verdade, conforme discutiremos a seguir.

Neste trabalho, assume-se que um *slot* é longo o suficiente para a transmissão de um

quadro (incluindo a transmissão de um eventual ACK da camada de enlace, bem como outros tempos associados ao procedimento de acesso ao meio). No entanto, a duração de um ciclo dos padrões de *duty cycle* assíncronos baseados em escalonamento é diretamente proporcional à duração de cada *slot*. Logo, é desejável que um *slot* seja o mais curto possível. Assim, uma consequência desta hipótese é que a perfeita sincronização dos padrões dos nós ao longo de um caminho de múltiplos saltos (*i.e.*, o uso de um *offset* = 0) não é uma boa escolha porque, ao receber um pacote para encaminhamento, um nó intermediário terá que esperar pelo próximo *slot* ativo em seu padrão, o que pode demorar a ocorrer.

Na Figura 3.1, do lado esquerdo, é mostrado um exemplo de um caminho de múltiplos saltos no qual todos os nós têm seus padrões de *duty cycle* perfeitamente alinhados (*i.e.*, offset = 0). Observando o escalonamento dado pelo Block Design $\{7,3,1\}$, se o nó C estiver no *slot* três no momento da sua transmissão bem sucedida para o nó B, B precisará aguardar vários slots até que exista uma oportunidade de transmissão para A (*i.e.*, para que ambos estejam simultaneamente com seus rádios ligados). Por outro lado, se a transmissão bem sucedida de C para B se der no *slot* 0 do padrão, B poderá realizar uma tentativa de encaminhamento do pacote logo no slot subsequente, já que esse será um slotativo para ambos B e C. Mesmo nesse caso, no entanto, se existisse um terceiro salto neste caminho (e.q., um nó entre A e o sorvedouro), o nó A necessariamente teria que aguardar certa quantidade de *slots* para obter uma nova oportunidade de transmissão. Por fim, é importante notar que o Block Design $\{7, 3, 1\}$ possui quase 50% dos slots ativos, o que reduz o número de *slots* até a próxima oportunidade de transmissão. Se o padrão usado fosse o $\{9507, 98, 1\}$, com aproximadamente 1% dos *slots* ativos (um *duty cycle* muito mais adequado às aplicações atuais), a possibilidade de que um nó intermediário receba um pacote e possua uma oportunidade de transmissão logo no *slot* seguinte se torna bem mais baixa. De fato, quanto menor o duty cycle do padrão utilizado, maior o número esperado de *slots* inativos entre dois *slots* ativos consecutivos.

Ainda na Figura 3.1, do lado direito, é apresentado um esquema de escalonamento alternativo, no qual, ao invés de todos os nós operarem sob padrões perfeitamente alinhados, um offset de 1 slot é aplicado entre os padrões de nós pai e filho no caminho em direção ao sorvedouro. O nó C, o mais distante do sorvedouro, ativa seus slots 0, 1 e 3 (novamente, de acordo com o Block Design $\{7,3,1\}$). Já o nó B, próximo salto de C em direção ao sorvedouro, tem seu padrão atrasado em um slot em relação a C, resultando no escalonamento [0 + 1, 1 + 1, 3 + 1] = [1, 2, 4]. Da mesma forma, o nó A precisa de um offset relativo a B de um slot, resultando no padrão [1 + 1, 2 + 1, 4 + 1] = [2, 3, 5].



Figura 3.1: Exemplo de caminho de múltiplos saltos no qual cada nó opera segundo um padrão de *duty cycle* baseado no *Block Design* $\{7,3,1\}$. O nó S é o sorvedouro da rede. São ilustradas duas possíveis sincronizações: os padrões à esquerda possuem *offset* = 0, enquanto os padrões à direita têm um *offset* relativo de 1 *slot* entre nós consecutivos no caminho até o sorvedouro.

Esse escalonamento garante que, quando um nó intermediário recebe um pacote para encaminhamento, este terá uma oportunidade de realizá-lo logo no *slot* subsequente. Se os enlaces da rede possuem alta probabilidade de entrega de quadros, este esquema mitiga o problema da interrupção do encaminhamento, reduzindo a latência de comunicação fim-a-fim.

Embora a Figura 3.1 ilustre o uso do offset = 1 para um padrão específico, a mesma análise pode ser realizada para qualquer padrão de *duty cycling* com fechamento para rotação. Qualquer padrão que apresente fechamento para rotação necessariamente possui dois *slots* consecutivos ativos (do contrário, dois nós com *offset* relativo igual a 1 nunca teriam *slots* ativos coincidentes). Logo, através da aplicação do *offset* = 1 entre vizinhos, sempre que um nó intermediário B recebe um pacote para encaminhamento, seu próximo *slot* será, necessariamente ativo. Como o próximo nó no caminho, digamos A, por sua vez, também opera com um *offset* = 1 em relação a B, este também terá seu *slot* ativo. Este argumento pode ser facilmente estendido para um número arbitrário de saltos.

3.1 Modelagem matemática da latência fim-a-fim para alguns casos específicos

Em um caso extremo, no escalonamento proposto, em que todos os enlaces do caminho têm probabilidade de entrega de quadros igual a 1, a latência fim-a-fim é dada pelo NDT do primeiro salto somado de um *slot* para cada salto subsequente. Dadas as Equações 2.1, 2.2, 2.3 e 2.4, apresentadas no Capítulo 2, podemos derivar novas equações para a latência em um caminho de k saltos para o *Block Design, Grid, Torus* e *Disco*, respectivamente,

para o caso em que p = 1:

$$D[k]_{Block\ Design} = \frac{v+1}{(\lambda+1)} - 1 + k \tag{3.1}$$

$$D[k]_{Grid} = \frac{n^2}{3} + k \tag{3.2}$$

$$D[k]_{Torus} = \frac{n^2}{2} + k$$
(3.3)

$$D[k]_{Disco} = \frac{q_1 q_2}{3} + k \tag{3.4}$$

Em comparação, no caso de offsets aleatórios (*i.e.*, de uma rede sem qualquer sincronismo), a latência fim-a-fim se aproxima¹ do produto do NDT pelo número de saltos do caminho. Isso ocorre porque não temos como saber em que *slot* de tempo um nó e seu vizinho estariam no instante em que o pacote é recebido para encaminhamento. Logo, pode ser necessário aguardar até um ciclo inteiro até que uma nova oportunidade de transmissão ocorra no próximo salto. Assim, podemos derivar aproximações para a latência fim-a-fim dos quatro métodos com base nas Equações 2.1, 2.2, 2.3 e 2.4, multiplicando-as por k (número de saltos), para o Block Design, Grid, Torus e Disco, respectivamente:

$$D[k]_{Block\ Design} = \left(\frac{v+1}{\lambda+1} - 1\right) \cdot k \tag{3.5}$$

$$D[k]_{Grid} = \left(\frac{n^2}{3}\right) \cdot k \tag{3.6}$$

$$D[k]_{Torus} = \left(\frac{n^2}{2}\right) \cdot k \tag{3.7}$$

$$D[k]_{Disco} = \left(\frac{q_1 q_2}{3}\right) \cdot k \tag{3.8}$$

A Figura 3.2 mostra o resultado da latência obtida através das Equações 3.1 a 3.8 para

¹As Equações apresentadas no Capítulo 2 assumem que o transmissor pode estar em qualquer *slot* do padrão no instante em que tenta iniciar sua transmissão, incluindo *slots* inativos. Por outro lado, no cenário utilizado nesta análise, um nó intermediário certamente iniciará sua tentativa de transmissão pelo próximo salto em um *slot* imediatamente consecutivo a um *slot* ativo. Essa diferença pode introduzir algum tipo de viés estatístico na modelagem apresentada neste capítulo.

os quatro métodos, *Block Design* {9507, 98, 1}, *Grid* 193 × 193, *Torus* 145 × 145 e *Disco* {193, 197}, todos com *duty cycle* aproximado de 1,03%. Como esperado, os gráficos mostram uma taxa de crescimento bem mais lenta para a latência fim-a-fim quando o offset=1 é utilizado. À medida que o número de saltos até o sorvedouro aumenta, a redução da latência obtida com o uso do offset=1 se torna cada vez mais representativa. Nos Capítulos 4 e 5 validaremos esta modelagem através de simulações numéricas e de rede.

Na prática, enlaces sem fio são sempre susceptíveis a algum nível de perda, tornando a expectativa de enlaces com a probabilidade de entrega de quadros igual a 1 irreal. Entretanto, como será mostrado nos resultados da nossa avaliação, mesmo para valores realísticos de probabilidade de entrega de bons enlaces sem fio, o uso do offset = 1 resulta em grande redução na latência fim-a-fim. Para enlaces de baixa qualidade, o uso do offset=1 resulta ainda em valores competitivos de NDT, em relação ao offset aleatório.



Figura 3.2: Comparação da latência fim-a-fim prevista pelas Equações apresentadas nesse capítulo, com *offset* aleatório e igual a 1, por padrões baseados em *Block Design*, *Grid*, *Torus* e *Disco* variando-se o número de saltos para *duty cycles* aproximados de 1,03%.

3.2 Comparação qualitativa com mecanismos síncronos baseados em escalonamento

Por fim, é importante destacar as diferenças entre a presente proposta e mecanismos síncronos baseados em escalonamento, como o SPEED-MAC e o CUPID. Assim como estes dois mecanismos, a proposta apresentada neste trabalho associa o escalonamento dos nós a suas distâncias em relação ao sorvedouro. No entanto, ao contrário do que ocorre nos mecanismos síncronos, o uso de esquemas assíncronos, como *Block Design*, *Grid, Torus* e *Disco*, simplifica a entrada de novos nós na rede, sem a necessidade de um sincronismo de relógio prévio ou de uma fase preliminar em que os novos nós fiquem com seus rádios permanentemente ligados. Um novo nó, ao entrar na rede, pode simplesmente

seguir o padrão de *duty cycle* assíncrono, economizando energia e, ainda assim, tendo a garantia de que conseguirá se comunicar com seus vizinhos. Após algum tempo, através da troca de mensagens de controle com seus vizinhos, o nó realizará sua sincronização com os demais membros da rede, otimizando seu desempenho em relação a latência fim-a-fim.

Capítulo 4

Avaliação Numérica

Para validar e estender a análise teórica apresentada no capítulo anterior, realizamos simulações numéricas comparando o funcionamento dos mecanismos assíncronos baseados em escalonamento com *offsets* aleatórios, resultantes da falta de sincronização de relógio entre os nós da rede, e o *offset* = 1 entre nós subsequentes em um caminho até o sorvedouro. Na literatura sobre mecanismos assíncronos baseados em escalonamento, tais comparações se dão tradicionalmente em termos de métricas como consumo de energia e latência [25, 30].

4.1 Parâmetros da Simulação

Foram realizadas simulações considerando comunicações de 1 a 7 saltos e diversos valores de probabilidade de entrega de quadros para os enlaces (variando de 0,05 a 1, com incrementos de 0,05) — em uma dada simulação, assume-se que todos os enlaces do caminho possuem a mesma probabilidade de entrega. Para cada conjunto de parâmetros, as simulações foram repetidas 20000 vezes, permitindo a estimativa do NDT médio e o cálculo do intervalo de confiança de 95%. Embora nos resultados que se seguem os intervalos de confiança tenham sido plotados para todos os pontos de todos os gráficos, seus valores foram percentualmente muito baixos, dificultando sua visualização nas figuras. A menos que se diga o contrário, em todos os resultados apresentados neste capítulo são comparados o *Block Design* {9507, 98, 1}, Grid 193x193, Torus 145x145 e Disco {193, 197}, todos resultando em *duty cycle* aproximado de 1,03%. O simulador usado foi o mesmo utilizado nos experimentos apresentados em [8], implementado na linguagem de programação C, estendido neste trabalho com a capacidade de simular comunicações em múltiplos saltos utilizando *offsets* aleatórios ou fixos entre nós subsequentes em um caminho.

		Offset = 1		Offset Aleatório				
	Simulação	Modelagem	Diferença	Simulação	Modelagem	Diferença		
BD	4770,34	4759,00	0,237%	33269,42	33271,00	-0,004%		
Disco	9553,86	12679,67	-32,717%	88418,21	88715,67	-0,336%		
Grid	18492,38	12422,33	32,824%	76037,26	86914,33	-14,304%		
Torus	10448,33	10518,50	-0,671%	73245,01	73587,50	-0,467%		

Tabela 4.1: Comparação dos valores obtidos com as simulações numéricas e a modelagem matemática para 7 saltos e p = 1.

4.2 Resultados para p = 1 e múltiplos saltos

A Figura 4.1 mostra, para cada mecanismo de *duty cycle*, a latência de comunicação fim-a-fim como uma função do número de saltos do caminho para padrões com *duty cycle* de aproximadamente 1,03% e probabilidade de entrega de quadros p = 1. Os gráficos mostram que o uso do *offset* = 1 resultou em menores latências para todas as comunicações de pelo menos dois saltos em todos os quatro mecanismos avaliados. Tanto para o *offset* aleatório, quanto para o *offset* = 1, a latência fim-a-fim aumenta linearmente com o aumento do número de saltos. No entanto, a taxa de aumento é bem mais baixa para o *offset* = 1 (para este cenário, um aumento de apenas um *slot* por salto, conforme explicado no Capítulo 3). Ainda na Figura 4.1 é possível verificar o fato de que o *Block Design* resultou nas menores latências para todos os números de saltos e para ambos os *offsets* aleatório e 1, um resultado esperado dada a vasta literatura que mostra a vantagem em latência dos *Block Designs* em relação a outros mecanismos desta classe.

Comparando os resultados obtidos na Figura 3.2 com os da Figura 4.1 é possível ver a consistência qualitativa entre a modelagem matemática apresentada no Capítulo 3 e os resultados das simulações numéricas. Ainda é possível observar na Tabela 4.1 que as simulações numéricas apresentaram resultados quantitativos próximos da modelagem matemática, principalmente para o *offset* aleatório. Algumas discrepâncias podem ser observadas para o *offset* = 1 para o qual a modelagem matemática teve erros percentuais representativos, apresentado, no caso do *Grid*, uma diferença de 32,8% para 7 saltos. Já para o *Disco* a modelagem matemática se mostrou pessimista para o mesmo *offset* = 1, com uma diferença de -32,7%.

Para comunicações de um único salto, o offset = 1 resulta em latência mais alta em alguns dos mecanismos, já que embutido no caso aleatório está o offset = 0, que garante um número maior de oportunidades de encontro por ciclo. Esse comportamento é exacerbado no *Grid*, conforme mostra a Figura 4.1 (b). No entanto, à medida que os caminhos se



Figura 4.1: Comparação da latência de comunicação obtida nas simulações numéricas com offset aleatório e igual a 1 por padrões baseados em *Block Design*, *Disco*, *Grid* e *Torus* variando-se o número de saltos do caminho com p = 1 para *duty cycles* aproximados de 1,03%.

tornam mais longos, a diferença percentual de latência entre os dois métodos se torna cada vez mais pronunciada em favor do offset=1, já que o offset aleatório começa a resultar em interrupção do encaminhamento de dados. Ainda no caso de um único salto, percebe-se que o offset=1 é particularmente bom para o Disco (resultando em uma latência de um salto menor que o NDT usado nas equações do capítulo 3), o que explica porque o modelo matemático superestimou a latência em múltiplos saltos do Disco para este offset.

Na Figura 4.2 são apresentados os resultados para padrões com um valor de *duty cy*cle mais elevado (aproximadamente 1,20%), através do *Block Design* {6973, 84, 1}, *Grid* 166 × 166, *Torus* 124 × 124 e *Disco* {163, 167}. É possível observar que, embora quantitativamente diferentes, os resultados dessas simulações são qualitativamente similares aos das simulações com *duty cycles* mais baixos: o *Block Design* se mantém como o padrão



Figura 4.2: Comparação da latência de comunicação obtida nas simulações com *offset* aleatório e igual a 1 por padrões baseados em *Block Design*, *Disco*, *Grid* e *Torus* variandose o número de saltos do caminho com p = 1 para *duty cycles* de aproximadamente 1,2%.

de mais baixa latência, enquanto o aumento no número de saltos torna o uso do offset=1 cada vez mais vantajoso. Assim como para os padrões de *duty cycle* mais baixo, novamente percebe-se que a latência de um salto para o *Disco* é mais baixa quando se utiliza o offset=1 que no caso aleatório.

4.3 Resultados para 7 saltos e p variando

Já na Figura 4.3, são apresentados os resultados de latência fim-a-fim em função da probabilidade de entrega de quadros p, para uma comunicação em 7 saltos. É interessante notar que o *Grid* e o *Torus* apresentam uma latência com crescimento bem mais lento para o *offset* = 1, uma vez que os dois métodos selecionam *slots* em linha (*i.e.*, que se transformam em uma grande sequência de *slots* ativos no padrão linearizado). Quando

a entrega de um quadro é feita com sucesso em um salto, o *slot* seguinte provavelmente também será uma oportunidade de encontro no próximo salto. Embora isso seja válido para os quatro mecanismos estudados aqui (porque todos apresentam ao menos dois *slots* ativos consecutivos) no caso do *Grid* e do *Torus* com *offset* = 1 haverá até n - 1 ou n/2 - 1 oportunidades de encontro consecutivas, respectivamente. Ainda que a primeira tentativa não seja bem sucedida, os nós podem tentar novamente várias vezes em pouco tempo, mantendo a latência baixa mesmo quando p diminui.



Figura 4.3: Comparação da latência de comunicação para 7 saltos obtida nas simulações com *offset* aleatório e igual a 1 por padrões baseados em *Block Design*, *Grid*, *Torus* e *Disco* variando-se a probabilidade de entrega do quadro (p).

Este resultado é particularmente relevante porque a literatura acerca desses métodos consistentemente mostra que o *Block Design* é superior aos demais métodos em termos de NDT. No entanto, como ilustrado nestes resultados, no caso da latência fim-a-fim, *Grid* e *Torus* se mostraram a melhor opção para redes com enlaces de qualidade mais

baixa, desde que o offset = 1 seja garantido entre os nós consecutivos no caminho até o sorvedouro.



Figura 4.4: Comparativo da latência dos 4 métodos de escalonamento como uma função da probabilidade de entrega de quadros de cada enlace de um caminho de 7 saltos, com offset = 1, normalizados pela latência do *Block Design*.

Para que fique mais clara a vantagem do Grid e do Torus em relação aos outros dois métodos para valores mais baixos de p, a Figura 4.4 mostra os mesmos valores de latência exibidos na Figura 4.3 para o caso do offset=1, mas normalizados pela latência obtida pelo *Block Design* (BD). Na figura é possível ver que para enlaces com probabilidade de entrega de quadros inferior ou igual a 90% o *Torus* resulta em menor latência, se comparado ao *Block Design* e ao *Disco*. De forma similar, o *Grid* passa a resultar em uma latência inferior a do *Block Design* para p = 0.8. A partir destes valores, a redução da latência pelo *Grid* e pelo *Torus* se acentua rapidamente em relação aos dois outros mecanismos.

Voltando aos resultados da Figura 4.3, é possível notar que, para o offset=1, o BlockDesign e o Disco apresentam uma latência aumentando exponencialmente à medida que p cai, de forma similar ao que ocorre com o offset aleatório. Ainda assim, mesmo para esses mecanismos, o uso do offset = 1 resultou em redução na latência fim-a-fim para todos os valores de probabilidade avaliados.

Capítulo 5

Avaliação do Ambiente de Simulação

Neste capítulo estenderemos a avaliação da proposta apresentada nesta dissertação através do uso de um ambiente de simulação para Rede de Sensores Sem Fio, com 36 nós operando sob os quatro métodos de *duty cycle* assíncrono já discutidos nos capítulos anteriores, utilizando *offset* aleatório e *offset* = 1 entre cada nó subsequente em um caminho até o sorvedouro. O objetivo destas simulações é avaliar como efeitos particulares de redes sem fio de múltiplos saltos não modelados diretamente nas simulações numéricas, como colisões, por exemplo, podem influenciar o desempenho da proposta. Estas simulações também levam em consideração a baixa resolução da sincronização proposta: ao contrário do que ocorre nas simulações numéricas, não há um alinhamento perfeito entre os *slots* de dois nós vizinhos. Para isso, foi utilizado o simulador TOSSIM, baseado no sistema operacional para sensores TinyOS, simulando motes do tipo MicaZ.

$5.1 \quad TinyOS$

O sistema operacional *TinyOS* [50], desenvolvido pela Universidade de *Stanford*, utiliza um modelo baseado em eventos para suportar operações concorrentes consumindo uma quantidade pequena de memória e criando rapidamente tarefas associadas a um evento, sendo assim projetado para sistemas embarcados sem fio de baixo consumo energético. O *TinyOS* inclui um escalonador de tarefas simplificado e um conjunto de componentes. Cada componente declara seus comandos e eventos para permitir a modularidade e a interação com outros componentes. O escalonador de tarefas é uma simples fila visando a economia de energia.

A compatibilidade do *TinyOS* [51] com os *hardwares* dos sensores disponíveis no mercado se estende às plataformas eyesIFXv2, Imote2, Mica2, Mica2dot, MicaZ, Telosb,

TinyNode e BTNode3.

O *TinyOS* se encontra na versão 2.1.2. Sua distribuição pode ser obtida gratuitamente, porém o procedimento de instalação descrito em [53] já não é totalmente compatível com novos sistemas operacionais e novas versões de *softwares* como a linguagem de programação *Python*, sendo necessárias algumas mudanças. Um detalhamento sobre o processo de instalação em sistemas operacionais *Linux* baseados em Debian pode ser encontrado no Apêndice A.

Para escrever uma aplicação para o *TinyOS*, utiliza-se um dialeto da linguagem C baseado em componentes chamado NesC [20]. A compilação das aplicações é feita através do compilador GCC [18].

5.2 MicaZ

A arquitetura do MicaZ [13] permite o desenvolvimento de aplicações personalizadas e é otimizada para o baixo consumo de energia. Baseia-se no microcontrolador AT-MEGA128L de baixo consumo energético, que pode ser configurado para executar aplicativos de sensoriamento e manipular o rádio. O MicaZ possui um rádio com tecnologia ZigBee e taxa de transmissão de 250 kb/s, normalmente adequada a aplicações de sensoriamento. As simulações descritas neste capítulo foram baseadas nesta plataforma de *hardware*.

5.3 Parâmetros e Ambiente da Simulação

As simulações foram efetuadas no TOSSIM [52], um simulador de eventos discretos baseado no sistema operacional TinyOS. Foi utilizado um computador com o sistema operacional Debian 8 e um processador AMD FX9500+ 2,2 GHz com 24 GB de memória.

Os *scripts* de simulação foram escritos em *Python*, permitindo instanciar nós sensores e compor a RSSF simulada para testar a solução proposta. Dentre as restrições e limites deste simulador está o fato de que ele apenas simula a plataforma de *hardware* MicaZ [13, 52].

Os valores padrão para o modelo de rádio do TOSSIM são baseados no rádio CC2420, usado no MicaZ [13], e nas famílias Telos e Imote2. Além do modelo de propagação do sinal de rádio, o TOSSIM também simula o ruído de RF e a interferência que um nó

PARÂMETROS DO CANAL	VALORES UTILIZADOS
PATH_LOSS_EXPONENT	4.7
SHADOWING_STANDARD_DEVIATION	3.2
D0	1.0
PL_D0	55.4
PARÂMETROS DO RÁDIO	
NOISE_FLOOR	-105.0
S11	0.9
S12	-0.7
S21	-0.7
S22	1.2
WHITE_GAUSSIAN_NOISE	4
PARÂMETROS DE TOPOLOGIA	
TOPOLOGY	1 (grade) e 3 (aleatória)
GRID_UNIT	1.0 a 5.0 para Topology 1
NUMBER_OF_NODES	36
TERRAIN_DIMENSIONS_X	9.0 a 20.0 para Topology 3
TERRAIN DIMENSIONS Y	9.0 a 20.0 para Topology 3

Tabela 5.1: Tabela de parâmetros usados nas simulações com base em [57]. PARÂMETROS DO CANAL VALORES UTILIZADOS

sofre, tanto de outros nós quanto de fontes externas. Para simular o ruído, o simulador foi alimentado com um *trace* chamado *meyer-heavy*, que foi obtido da *Meyer Library* da Universidade de *Stanford*.

As topologias utilizadas nas simulações apresentadas neste capítulo foram obtidas através da ferramenta do próprio TOSSIM [49], descrita em [57]. Para gerar uma topologia de rede mais próxima da realidade, vários parâmetros relacionados ao canal, rádio e distância foram modificados, usando como referência para os valores os modelos descritos em [57], conforme ilustrado na Tabela 5.1. Neste trabalho foram usados os seguintes dois tipos de topologias:

- Grade: os nós são dispostos geograficamente de acordo com uma grade. Como exigência, o número de nós tem de ser um quadrado perfeito.
- Aleatória: os nós são colocados aleatoriamente em um terreno físico de dimensão n x n.

Assim, foram simuladas topologias em grade com distâncias entre os nós de 1, 2, 3, 4 e 5 metros e topologias aleatórias com áreas de 9×9 , 10×10 , 11×11 , 13×13 , 15×15 e 20×20 metros quadrados. É importante salientar que a distância de 1 metro nas topologias em grade e a área de 9×9 metros quadrados na topologia aleatória foram as menores medidas permitidas pelo gerador de topologias do TOSSIM. Já com distâncias acima de 5 metros entre os nós da topologia em grade e áreas maiores que 20×20 metros quadrados nas topologias aleatórias, os cenários gerados resultaram em redes particionadas.

Em ambos os tipos de topologia, foram utilizados 36 nós, atendendo à exigência da topologia em grade, sendo apenas um nó como sorvedouro. Todos os nós operam com slots de tempo com duração de 11 ms. Este valor foi escolhido levando em consideração o tempo necessário para a transmissão de um quadro de tamanho máximo pela interface sem fio do MicaZ. Como o tamanho máximo do pacote a ser transmitido é composto de 127 bytes de um pacote na camada MAC (já incluindo o cabeçalho de 24 bytes e o payload de 103 bytes) acrescido de 6 bytes na camada física [60], obtemos um pacote de 133 bytes. Como a taxa de transmissão do MicaZ é de 250 kb/s (31250 B/s), o atraso de transmissão seria de 133/31250 =4,256 ms.

Considerando que estes pacotes de dados serão transmitidos em *unicast* (ou seja, terão *ack*) e que existem também tempos de acesso ao meio somados a este valor, definiu-se o tempo de um *slot* para estas simulações como 10 ms (suficiente para uma transmissão e uma recepção de um quadro de dados de tamanho máximo) somando com 1 ms para a transmissão de um *beacon* (considerando-se os 30 bytes de cabeçalho das camadas MAC e física, acrescidos de 7 bytes de *payload* de um *beacon* - 37/31250 = 1,184 ms), totalizando 11 ms.

Os padrões utilizados para o *duty cycle* dos rádios foram os mesmos usados nas simulações numéricas, ou seja, *Block Design* {9507, 98, 1}, *Grid* 193x193, *Torus* 145x145 e *Disco* {193, 197}, todos resultando em *duty cycle* aproximado de 1,03%.

5.4 Outros Aspectos da Simulação

Cada nó é inicializado em um período aleatório compreendido entre 0 e a quantidade de slots do padrão utilizado multiplicada pela duração de um slot (i.e., 11 ms). Para que um sensor conheça seu próximo salto em direção ao sorvedouro, bem como as informações relevantes para a sincronização de slots proposta neste trabalho, foi implementado um protocolo de roteamento bastante simplificado, baseado em número de saltos. Cada nó da simulação mantém uma variável que indica sua distância, em saltos, até o sorvedouro. Como inicialmente os nós não possuem qualquer conhecimento sobre a topologia da rede, o sorvedouro a inicializa com 0, enquanto os demais nós a inicializam com a constante 255, indicando uma distância infinita.

Após a criação e inicialização de cada nó, começam-se a contar os ciclos de tempo. A cada novo *slot* de tempo, cada nó verifica se o *slot* atual está compreendido no conjunto de *slots* ativos do padrão de *duty cycle* utilizado. Em caso afirmativo, uma chamada à



Figura 5.1: Comparação do tempo médio de convergência obtido com *offset* aleatório e igual a 1 por padrões baseados em *Block Design* variando-se a distância entre os nós na topologia em grade e a área do cenário para a topologia aleatória.

rotina de ativar a interface de rádio é realizada e assim que a mesma se encontra ativa, um *beacon* é enviado em *broadcast* contendo o identificador do nó, o *slot* atual em que o transmissor se encontra e a distância dele ao sorvedouro.

Ao receber um *beacon*, o nó verifica se o número de saltos informado no pacote acrescido de 1 é inferior à sua melhor distância conhecida até o sorvedouro. Em caso afirmativo, o receptor atualiza sua melhor distância conhecida para o valor contido no *beacon* acrescido de uma unidade e armazena em uma variável o identificador do transmissor como seu pai na árvore.

Além de utilizados para roteamento, nas simulações do método proposto nesta dissertação, os *beacons* são usados também para a sincronização de *slots* entre os nós. Ao receber um *beacon* do seu pai na árvore, um nó altera seu *slot* atual para o *slot* do seu pai — informando no *beacon* — adicionado de uma unidade (resultando, assim, no *offset* relativo de 1). Mesmo após a primeira sincronização, os nós ainda usam os demais *beacons* para manter a sincronização, visto que deslizes nas janelas de tempo são comuns no decorrer de longos períodos, um aspecto prático importante em sensores.

Com intuito de simular uma Rede de Sensores Sem Fio o mais próxima o possível da realidade, cada nó, exceto o sorvedouro, gera um pacote de dados, propiciando a possibilidade de colisões entre os pacotes de dados e os *beacons*, a cada 5 segundos de simulação contados a partir da inicialização de cada nó. A partir do momento em que o pacote de dados é gerado, o nó aguarda até seu próximo *slot* ativo no seu padrão de *duty cycle* e efetua uma tentativa de transmissão em *unicast* endereçada a seu pai na árvore.



Figura 5.2: Comparação do tempo médio de convergência obtido com *offset* aleatório e igual a 1 por padrões baseados em *Disco* variando-se a distância entre os nós na topologia em grade e a área do cenário para topologia aleatória.

Apenas quando um ack é recebido, reconhecendo positivamente o recebimento do pacote pelo próximo nó no caminho, o transmissor encerra seu processo de transmissão.

Cada simulação possui a duração de 120000000 de eventos (o simulador TOSSIM é configurado por eventos), o que perfaz cerca de 1800 segundos para as simulações descritas neste capítulo. Para que os resultados apresentados na próximas seções sejam estatisticamente relevantes, cada simulação foi repetida 30 vezes para cada topologia.

Para a topologia aleatória, cada rodada de simulação gera posicionamentos potencialmente diferentes para cada nó, resultando em topologias efetivamente distintas. Já para as simulações com a topologia em grade, o TOSSIM mantém a mesma estrutura topológica, mas distribui aleatoriamente os identificadores dos nós. Nas simulações realizadas neste trabalho, foi arbitrado que o sorvedouro seria sempre o nó com identificador 0. Logo, o sorvedouro pode efetivamente estar em posições diferentes em simulações distintas usando a topologia em grade.

Cada vez que uma rodada de simulação termina, variáveis e arquivos de configuração do TOSSIM, como o app.xml, tossim.o e sim.o, que armazenam as configurações da topologia e do ruído para aquela simulação, são excluídos, reforçando a diferenciação e execução independente das várias repetições das simulações.

Todos os gráficos que apresentam a média das 30 rodadas de simulações incluem um intervalo de confiança de 95%.



Figura 5.3: Comparação do tempo médio de convergência obtido com *offset* aleatório e igual a 1 por padrões baseados em *Grid* variando-se a distância entre nós na topologia em grade e a área do cenário para topologia aleatória.

5.5 Métricas de Avaliação

A métrica usada para as avaliações deste capítulo foi a *média do tempo de convergência da rede.* Esta métrica é obtida através da média do tempo que cada simulação necessita para sincronizar a rede. Esta métrica permite avaliar se o uso do *offset*=1 tem algum impacto, positivo ou negativo, na inicialização da rede (que inclui a convergência do protocolo de roteamento utilizado). Como será discutido a seguir, este tempo de convergência é também fortemente correlacionado com o atraso de comunicação, servindo, portanto, como uma forma avaliação comparativa deste aspecto.

O cálculo do tempo de convergência da rede, nestas simulações, toma por base o momento em que todos os nós da rede se encontram sincronizados com seus respectivos pais. Não havendo mais alterações nesse sentido até o final da simulação.

5.6 Tempo de Convergência

As Figuras 5.1 a 5.4 apresentam o tempo de convergência médio para as topologia em grade e aleatória, aplicando-se o *offset* aleatório e o *offset* igual a 1. Como as figuras ilustram, para todos os 4 métodos avaliados o uso do *offset*=1 resultou em redução do tempo de convergência do protocolo de roteamento e sincronização. A maior redução do tempo de convergência entre o *offset* aleatório e o *offset*=1 para a topologia em grade com distância entre nós de 1 metro foi dada pelo *Grid*, com 36,8%. Para a distância entre



Figura 5.4: Comparação do tempo médio de convergência obtido com *offset* aleatório e igual a 1 por padrões baseados em *Torus* variando-se a distância entre nós na topologia em grade e a área do cenário para topologia aleatória.

nós de 5 metros, o *Block Design* apresentou uma maior redução, com 63,6%. No cenário onde a topologia aleatória foi utilizada, o *Grid* superou os demais métodos, com 47,6% de redução do tempo de convergência para a área de 9×9 metros quadrados e 66,6% para a área de 20×20 metros quadrados. Isso corrobora os resultados das simulações numéricas, no sentido de que a menor latência em múltiplos saltos verificada naquelas simulações para o *offset*=1 parecem ter se traduzido em uma convergência mais rápida. Este resultado é consistente para todas as distâncias/áreas avaliadas.

Nos resultados obtidos, o *Grid* e o *Torus* apresentaram desempenho melhor do que o *Block Design* e o *Disco* para o *offset*=1, o que novamente é consistente com os resultados das simulações numéricas, onde os mesmos dois métodos obtiveram resultados superiores, para probabilidade de entrega de quadros mais baixas que 0.85.

Vale ressaltar, no entanto, que em praticamente todos os pontos mostrados nos gráficos da Figura 5.5, as curvas do *Grid* e do *Torus* se encontram nos intervalos de confiança uma da outra. Logo, não é possível afirmar com confiança estatística que um dos métodos tenha sido efetivamente superior ao outro nos cenários avaliados.

Ainda na Figura 5.5 é apresentada uma comparação mais direta entre os tempos de convergência obtidos com o *Grid* e o *Torus* para a topologia em grade, com ambos os *offsets* 1 e aleatório. Os resultados foram estatisticamente muito próximos (vide intervalo de confiança), não sendo possível afirmar que um método é superior ao outro. Para o offset=1, o *Grid* apresentou o melhor tempo de convergência para a topologia em grade com distância entre os nós de 3, 4 e 5 metros, com valores aproximando-se dos 165, 103 e



Figura 5.5: Comparação do tempo médio de convergência obtido com *offset* igual a 1 e aleatório por padrões baseados em *Grid* 193×193 e em *Torus* 145×145 variando-se a distância entre nós na topologia em grade.

60 segundos respectivamente. Já para a topologia em grade com distância entre nós de 1 e 2 metros, o *Torus* apresentou o melhor tempo de convergência, tendo seu valor próximo a 189 e 209 segundos para o *offset*=1. Já para o *offset* aleatório, o *Torus* apresentou os melhores resultados para o tempo de convergência para a topologia em grade, com valores próximos a 230 segundos para a distância de 1 metro entre os nós e 203 segundos para a distância de 3 metros, enquanto o *Grid* apresentou convergência em 80 segundos para a distância de 5 metros entre os nós.

Um comportamento consistente observado na topologia em grade para todos os métodos foi a relação decrescente entre o tempo de convergência e a distância entre nós. Este resultado é de certa forma surpreendente, uma vez que o comportamento esperado seria o inverso, já que o número de saltos provavelmente aumenta com o aumento da distância. Para analisar melhor esse aspecto, a Tabela 5.2 provê, para cada topologia simulada, estatísticas sobre a profundidade da árvore resultante do protocolo de roteamento, ou seja, a maior distância entre o sorvedouro e um dos nós sensores. A tabela exibe os valores mínimo, máximo e médio da profundidade da árvore em cada topologia. De fato,

Topologia	Grade					Aleatória					
Topologia	1	2	3	4	5	9×9	10×10	11×11	13×13	15×15	20×20
Menor	4	4	Б	6	6	4	4	4	4	4	F
Profundidade	4	4	5	0	0	4	4	4	4	4	5
Profundidade	۲ŋ	ΕO	6.9	60	7.0	57	E G	۲Q	6 9	57	61
Média	$_{0,0}$	$^{-0,9}$	0,2	0,0	7,0	5,7	5,0	0,0	0,3	5,7	0,1
Maior	0	Q	0	0	0	7	0	8	0	8	8
Profundidade	Э	0	9	9	9		J	8	J	8	8

Tabela 5.2: Profundidade da árvore para as topologias em grade e aleatória.

conforme pode ser observado na Tabela 5.2, a profundidade média aumenta conforme a distância entre nós é incrementada, enquanto o tempo de convergência diminui. Assim, com base nos resultados das simulações numéricas apresentadas no capítulo anterior, é possível especular que esta redução no tempo de convergência se deva ao uso de enlaces com qualidade melhor nos cenários de maior distância, reduzindo, assim, o tempo de convergência mesmo em face de um número maior de saltos. Pretende-se, em trabalhos futuros, coletar mais informações acerca destas simulações (como a probabilidade de entrega de quadros nos enlaces usados nos caminhos até o sorvedouro) com o objetivo de comprovar ou refutar essa hipótese.

Capítulo 6

Conclusão

Este trabalho apresenta e avalia uma proposta para redução da latência de comunicação em múltiplos saltos dos mecanismos de *duty cycle* assíncrono baseados em escalonamento através de sincronização de baixa resolução. Demonstrou-se que este nível de sincronização pode ser obtido com baixo *overhead*, uma vez que é suficiente a inserção de um simples campo numérico nos pacotes de controle do protocolo de roteamento informando o *slot* de tempo em que o nó atual se encontra. Com base nestas informações e no conhecimento do próximo salto em direção ao sorvedouro, os nós podem alinhar seus padrões de *duty cycle*, no caso desta proposta, adicionando um *offset* entre eles. Para tanto, foi implementado um simples protocolo de roteamento baseado em números saltos, como forma de prova de conceito com objetivo de demonstrar a viabilidade prática desta proposta, possibilitando manter uma sincronização dos nós em resolução de *slots* que permita que nós vizinhos sigam escalonamentos com *offsets* específicos.

Através dos modelos matemáticos simplificados providos no Capítulo 3, mostramos que o uso do offset=1 proporciona uma redução da latência de comunicação em múltiplos saltos para os quatro mecanismos de *duty cycle* assíncronos testados, ao mesmo tempo em que garante que novos nós podem ser facilmente adicionados à rede, sem a necessidade de um processo de sincronização prévio ou de uma fase inicial de sincronização na qual o novo nó é obrigado a permanecer com seu rádio ligado todo o tempo, resultando em desperdício de energia. Através de simulações numéricas, mostramos que, de fato, a introdução do offset = 1 entre nós subsequentes em um caminho até o sorvedouro resulta em reduções representativas de latência fim-a-fim. Estes ganhos se tornam ainda mais expressivos à medida que o número de saltos aumenta. Outro resultado interessante é o fato dos mecanismos Grid e Torus manterem latências com crescimento muito mais lento em relação à probabilidade de entrega de quadros dos enlaces na comunicação de múltiplos

saltos com o uso do offset = 1, resultando em latências mais baixas que às obtidas pelo Block Design (um mecanismo conhecidamente superior em termos de NDT).

As simulações realizadas no TOSSIM corroboram os resultados das simulações numéricas e da modelagem matemática. Os resultados destas simulações mostram que o uso de um offset = 1 proporciona uma redução representativa do tempo de convergência da rede em múltiplos saltos. Como o simulador TOSSIM é um simulador que permite uma simulação de Redes de Sensores Sem Fio bastante detalhista, requerendo o desenvolvimento de código para o sistema operacional *TinyOS*, estes resultados também suportam a viabilidade prática do mecanismo proposto.

6.1 Trabalhos Futuros

Futuramente pretende-se aprofundar as simulações e experimentos avaliando como as métricas de desempenho aqui consideradas se comportam com o uso de outros valores fixos de *offset* entre nós. Pretende-se também explorar um pouco mais este protótipo de protocolo de sincronização de baixa resolução e roteamento, estendendo-o para considerar métricas de qualidade dos enlaces, evitando assim o uso de enlaces de qualidade muito baixa. Ainda em relação ao roteamento, um tópico de interesse é a criação de uma métrica de roteamento baseada na latência fim-a-fim que considere os modelos matemáticos apresentados no Capítulo 3. Neste sentido, o refinamento destes modelos, evitando as discrepâncias apontadas pelas simulações numéricas nos casos do *Disco* e *Grid*, deve ser estudado. Pretende-se ainda realizar testes em ambientes reais para permitir a avaliação direta do impacto do *overhead* de sincronização na comunicação de rede e no consumo energético dos nós.

Referências

- [1] AKYILDIZ, I. F.; POMPILI, D.; MELODIA, T. Underwater acoustic sensor networks: research challenges. Ad hoc networks 3, 3 (2005), 257–279.
- [2] AKYILDIZ, I. F.; SU, W.; SANKARASUBRAMANIAM, Y.; CAYIRCI, E. A survey on sensor networks. *IEEE Communications magazine* 40, 8 (2002), 102–114.
- [3] ANASTASI, G.; CONTI, M.; DI FRANCESCO, M.; PASSARELLA, A. Energy conservation in wireless sensor networks: A survey. Ad hoc networks 7, 3 (2009), 537–568.
- [4] BACHIR, A.; DOHLER, M.; WATTEYNE, T.; LEUNG, K. K. MAC essentials for wireless sensor networks. *IEEE Communications Surveys & Tutorials 12*, 2 (2010), 222–248.
- [5] BARONTI, P.; PILLAI, P.; CHOOK, V. W.; CHESSA, S.; GOTTA, A.; HU, Y. F. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards. *Computer communications 30*, 7 (2007), 1655–1695.
- [6] BU, S.; NAGHDY, F. Service discovery in wireless ad-hoc control networks. In Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on (2005), IEEE, pp. 157–162.
- [7] BUETTNER, M.; YEE, G. V.; ANDERSON, E.; HAN, R. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the* 4th international conference on Embedded networked sensor systems (2006), ACM, pp. 307–320.
- [8] CARRANO, R. C.; PASSOS, D.; MAGALHÃES, L. C.; ALBUQUERQUE, C. V. Nested block designs: Flexible and efficient schedule-based asynchronous duty cycling. *Computer Networks* 57, 17 (2013), 3316–3326.
- [9] CARRANO, R. C.; PASSOS, D.; MAGALHÃES, L. C.; ALBUQUERQUE, C. V. A comprehensive analysis on the use of schedule-based asynchronous duty cycling in wireless sensor networks. Ad Hoc Networks 16 (2014), 142–164.
- [10] CARRANO, R. C.; PASSOS, D.; MAGALHÃES, L. C.; ALBUQUERQUE, C. V. Survey and taxonomy of duty cycling mechanisms in wireless sensor networks. *IEEE Communications Surveys & Tutorials* 16, 1 (2014), 181–194.
- [11] CHOI, L.; LEE, S. H.; JUN, J.-A. SPEED-MAC: Speedy and energy efficient data delivery MAC protocol for real-time sensor network applications. In 2010 IEEE International Conference on Communications (ICC), (2010), IEEE, pp. 1–6.

- [12] COSTA, R. A. A.; MENDES, L. A. M. Evolução das redes sem fio: Um estudo comparativo entre bluetooth e zigbee, 2006. Disponível em http://ftp.unipac.br/ site/bb/tcc/tcc-a010b188f93af4c28ca9af23b9e3c476.pdf.
- [13] CROSSBOW TECHNOLOGY, INC. MicaZ, 2010. Disponível em http://www. openautomation.net/uploadsproductos/micaz_datasheet.pdf.
- [14] DE SOUSA, R. R.; MARGI, C. B. Sensoreamento sem fio em aeromodelos radiocontrolados, 2010. Disponível em http://www.larc.usp.br/~cbmargi/rodolfo/ FapespFinal.pdf.
- [15] DELICATO, F.; PIRES, P. F.; LAGES, A.; REZENDE, J. D.; PIRMEZ, L. Middleware orientado a serviços para redes de sensores sem fio. In Anais do 220 Simpósio Brasileiro de Redes de Computadores (2004).
- [16] DUTTA, P.; CULLER, D. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In *Proceedings of the 6th ACM conference on Embedded network sensor systems* (2008), ACM, pp. 71–84.
- [17] EL-HOIYDI, A.; DECOTIGNIE, J.-D. WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. In *International Symposium on Algorithms* and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (2004), Springer, pp. 18–31.
- [18] FREE SOFTWARE FOUNDATION, INC. GCC, the GNU compiler collection, 2017. Disponível em https://gcc.gnu.org/.
- [19] GARCIA-LUNA-ACEVES, J.; TZAMALOUKAS, A. Reversing the collision-avoidance handshake in wireless networks. In Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (1999), ACM, pp. 120–131.
- [20] GAY, D.; LEVIS, P.; CULLER, D.; BREWER, E. NESC 1.3 language reference manual, 2009. Disponível em https://github.com/tinyos/nesc/blob/master/doc/ ref.pdf.
- [21] HEINZELMAN, W. R.; CHANDRAKASAN, A.; BALAKRISHNAN, H. Energy-efficient communication protocol for wireless microsensor networks. In 2000. Proceedings of the 33rd annual Hawaii International Conference on System Sciences (2000), IEEE.
- [22] HOLLER, M. Update camalie networks wireless sensing, 2008. Disponível em http: //50.161.9.77/WirelessSensing/WirelessSensors.htm.
- [23] JAFARI, R.; ENCARNACAO, A.; ZAHOORY, A.; DABIRI, F.; NOSHADI, H.; SAR-RAFZADEH, M. Wireless sensor networks for health monitoring. In *MobiQuitous* 2005. The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (2005), IEEE, pp. 479–481.
- [24] JIANG, J.-R.; TSENG, Y.-C.; HSU, C.-S.; LAI, T.-H. Quorum-based asynchronous power-saving protocols for IEEE 802.11 ad hoc networks. *Mobile Networks and Applications 10*, 1-2 (2005), 169–181.

- [25] KANDHALU, A.; LAKSHMANAN, K.; RAJKUMAR, R. R. U-CONNECT: a lowlatency energy-efficient asynchronous neighbor discovery protocol. In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (2010), ACM, pp. 350–361.
- [26] KRISHNAMURTHY, L.; ADLER, R.; BUONADONNA, P.; CHHABRA, J.; FLANIGAN, M.; KUSHALNAGAR, N.; NACHMAN, L.; YARVIS, M. Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea. In *Proceedings of the 3rd international conference on Embedded networked sensor* systems (2005), ACM, pp. 64–75.
- [27] KRUGER, D.; PFISTERER, D.; FISCHER, S. CUPID-communication pattern informed duty cycling in sensor networks. In 2010 Fifth International Conference on Systems and Networks Communications (2010), IEEE, pp. 70–75.
- [28] LIANG, C.-J. M.; TERZIS, A., ET AL. Koala: Ultra-low power data retrieval in wireless sensor networks. In *Proceedings of the 7th international conference on Infor*mation processing in sensor networks (2008), IEEE Computer Society, pp. 421–432.
- [29] LIN, R.; WANG, Z.; SUN, Y. Wireless sensor networks solutions for real time monitoring of nuclear power plant. In 2004. WCICA 2004. Fifth World Congress on Intelligent Control and Automation (2004), vol. 4, IEEE, pp. 3663–3667.
- [30] LINK, J. Á. B.; WOLLGARTEN, C.; SCHUPP, S.; WEHRLE, K. Perfect difference sets for neighbor discovery: energy efficient and fair. In *Proceedings of the 3rd Extreme Conference on Communication: The Amazon Expedition* (2011), ACM, p. 5.
- [31] LOUREIRO, A. A.; NOGUEIRA, J. M. S.; RUIZ, L. B.; MINI, R. A. D. F.; NAKA-MURA, E. F.; FIGUEIREDO, C. M. S. Redes de sensores sem fio. In *Minicursos do* Simpósio Brasileiro de Redes de Computadores (2003), vol. 21, pp. 19–23.
- [32] LU, G.; KRISHNAMACHARI, B.; RAGHAVENDRA, C. S. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *Parallel* and Distributed Processing Symposium, 2004. Proceedings. 18th International (2004), IEEE, p. 224.
- [33] MAEKAWA, M. An \sqrt{N} algorithm for mutual exclusion in decentralized systems. ACM Transactions on Computer Systems (TOCS) 3, 2 (1985), 145–159.
- [34] MCEWEN, A.; CASSIMALLY, H. Designing the internet of things. John Wiley & Sons, 2013.
- [35] NEMEROFF, J.; GARCIA, L.; HAMPEL, D.; DIPIERRO, S. Application of sensor network communications. In *Military Communications Conference*, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE (2001), vol. 1, IEEE, pp. 336–341.
- [36] OGUNDILE, O. O.; ALFA, A. S. A survey on an energy-efficient and energy-balanced routing protocol for wireless sensor networks. *Sensors* 17, 5 (2017).

- [37] PARMAR, S. N.; NANDI, S.; CHOWDHURY, A. R. Power efficient and low latency MAC for wireless sensor networks. In 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (2006), vol. 3, IEEE, pp. 940–944.
- [38] PARUCHURI, V.; BASAVARAJU, S.; DURRESI, A.; KANNAN, R.; IYENGAR, S. S. Random asynchronous wakeup protocol for sensor networks. In 2004. BroadNets 2004. Proceedings. First International Conference on Broadband Networks (2004), IEEE, pp. 710–717.
- [39] PEREIRA, G. A.; SOARES, M. B.; CAMPOS, M. F. M. A potential field approach for collecting data from sensor networks using mobile robots. In *Intelligent Robots and* Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on (2004), vol. 4, IEEE, pp. 3469–3474.
- [40] PIRES, P. F.; DELICATO, F.; BATISTA, T.; BARROS, T.; CAVALCANTE, E.; PI-TANGA, M. Plataformas para a internet das coisas. *Minicursos do Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos* (2015).
- [41] POLASTRE, J.; HILL, J.; CULLER, D. Versatile low power media access for wireless sensor networks. In Proceedings of the 2nd international conference on Embedded networked sensor systems (2004), ACM, pp. 95–107.
- [42] RAJENDRAN, V.; OBRACZKA, K.; GARCIA-LUNA-ACEVES, J. J. Energy-efficient, collision-free medium access control for wireless sensor networks. *Wireless Networks* 12, 1 (2006), 63–78.
- [43] ROCHA, J. W. V. Rede de sensores sem fio, 2010. Disponível em http://www. teleco.com.br/pdfs/tutorialrssf.pdf.
- [44] ROWE, A.; MANGHARAM, R.; RAJKUMAR, R. RT-link: A global time-synchronized link protocol for sensor networks. Ad Hoc Networks 6, 8 (2008), 1201–1220.
- [45] SARAIVA, A. R. C.; PASSOS, D.; CARRANO, R. C.; ALBUQUERQUE, C. V. N. Reduzindo a latencia de comunicação em múltiplos saltos dos mecanismos de duty cycle assíncrono baseados em schedule através de sincronização de baixa resolução. In Anais do 35^o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (2017), SBC, pp. 673–686.
- [46] SCHURGERS, C.; TSIATSIS, V.; GANERIWAL, S.; SRIVASTAVA, M. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions On Mobile Coputing* 1, 1 (2002), 70–80.
- [47] SEIDEL, C.; FERREIRA, F. M.; DE OLIVEIRA, E. C. R. Aplicação de redes de sensores sem fio (RSSFs) para engenharia ambiental. In *Proceedings, International Conference on Engineering and Computer Education, ICECE 2007* (2007), ICECE, pp. 155–160.
- [48] SIVRIKAYA, F.; YENER, B. Time synchronization in sensor networks: a survey. *IEEE network 18*, 4 (2004), 45–50.

- [49] STANFORD UNIVERSITY. Configuring a network, 2013. Disponível em http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM#Configuring_ a_Network.
- [50] STANFORD UNIVERSITY. TinyOS documentation wiki, 2013. Disponível em http: //tinyos.stanford.edu/tinyos-wiki/index.php/Main_Page.
- [51] STANFORD UNIVERSITY. TinyOS platform hardware, 2013. Disponível em http: //tinyos.stanford.edu/tinyos-wiki/index.php/Platform_Hardware.
- [52] STANFORD UNIVERSITY. TOSSIM, 2013. Disponível em http://tinyos.stanford. edu/tinyos-wiki/index.php/TOSSIM.
- [53] STANFORD UNIVERSITY. Installing TinyOS, 2014. Disponível em http://tinyos. stanford.edu/tinyos-wiki/index.php/Installing_TinyOS.
- [54] SUN, Y.; GUREWITZ, O.; JOHNSON, D. B. RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems* (2008), ACM, pp. 1–14.
- [55] TILAK, S.; ABU-GHAZALEH, N. B.; HEINZELMAN, W. A taxonomy of wireless micro-sensor network models. ACM SIGMOBILE Mobile Computing and Communications Review 6, 2 (2002), 28–36.
- [56] TSENG, Y.-C.; HSU, C.-S.; HSIEH, T.-Y. Power-saving protocols for IEEE 802.11based multi-hop ad hoc networks. *Computer Networks* 43, 3 (2003), 317–337.
- [57] UNIVERSITY OF SOUTHERN CALIFORNIA. Building a network topology for TOSSIM, 2006. Disponível em https://github.com/tinyos/tinyos-main/blob/master/ doc/html/tutorial/usc-topologies.html.
- [58] VAN DAM, T.; LANGENDOEN, K. An adaptive energy-efficient MAC protocol for wireless sensor networks. In Proceedings of the 1st international conference on Embedded networked sensor systems (2003), ACM, pp. 171–180.
- [59] VASANTHI, N.; ANNADURAI, S. AWS: asynchronous wakeup schedule to minimize latency in wireless sensor networks. In 2006. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (2006), vol. 1, IEEE.
- [60] WORD, R. W. Zigbee MAC layer frames, 2012. Disponível em http://www. rfwireless-world.com/Tutorials/Zigbee-MAC-layer-frame-format.html.
- [61] XU, Y.; HEIDEMANN, J.; ESTRIN, D. Geography-informed energy conservation for ad hoc routing. In Proceedings of the 7th annual international conference on Mobile computing and networking (2001), ACM, pp. 70–84.
- [62] YE, W.; HEIDEMANN, J.; ESTRIN, D. An energy-efficient MAC protocol for wireless sensor networks. In INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (2002), vol. 3, IEEE, pp. 1567–1576.

- [63] YIN, C.; LI, Y.; ZHANG, D.; CHENG, Y.; YIN, M. DSMAC: An energy-efficient MAC protocol in event-driven sensor networks. In Advanced Computer Control (ICACC), 2010 2nd International Conference on (2010), vol. 1, IEEE, pp. 422–425.
- [64] ZHENG, R.; HOU, J. C.; SHA, L. Asynchronous wakeup for ad hoc networks. In Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (2003), ACM, pp. 35–45.

APÊNDICE A – Instalando o Sistema TinyOS 2.1.2 no Ubuntu 16.01 ou Debian 8

Como o simulador do *TinyOS* não apresenta mais suporte às versões atuais dos sistemas operacionais, foi necessário ajustar as instruções de instalação. A incompatibilidade do *TinyOS* com Python 3, por exemplo, faz com que o mesmo não funcione corretamente. Este apêndice apresenta um passo-a-passo detalhado, bem mais completo do que o encontrado em [53], para instalação do TinyOS 2.1.2 no Ubuntu 16 ou no Debian 8.

1. Atualizar os repositórios do gerenciador de pacotes

\$ sudo apt-get update

2. Instalar o Java Development Kit

\$ sudo add-apt-repository ppa:openjdk-r/ppa
\$ sudo apt-get update
\$ sudo apt-get install openjdk-8-jdk
\$ sudo update-alternatives --config java
\$ sudo update-alternatives --config javac

3. Instalar os pacotes de suporte

\$ sudo apt-get install flex \$ sudo apt-get install bison \$ sudo apt-get install swig \$ sudo apt-get install perl \$ sudo apt-get install graphviz \$ sudo apt-get install build-essential \$ sudo apt-get install python-dev \$ sudo apt-get install python-pygame

4. Instalar as bibliotecas

\$ sudo apt-get install libc6
\$ sudo apt-get install libgcc1
\$ sudo apt-get install libgmp10
\$ sudo apt-get install libmpfr4
\$ sudo apt-get install zlib1g

 Fazer o download dos seguintes pacotes em: http://packages.ubuntu.com/precise/ gcc-msp430 (observar a plataforma utilizada).

Os comandos abaixo realizam a instalação dos pacotes.

\$ sudo dpkg -i libmpc2_0.9-4_amd64.deb
\$ sudo dpkg -i msp430mcu_20110613-3_all.deb
\$ sudo dpkg -i msp430-libc_20110612-2_amd64.deb
\$ sudo dpkg -i binutils-msp430_2.22~msp20110716p5-1_amd64.deb
\$ sudo dpkg -i gcc-msp430_4.5.3~mspgcc-20110716-4_amd64.deb

6. Fazer o download dos seguintes pacotes do TinyOS em: http://tinyprod.net/ repos/debian/pool/main/n/nesc/ehttp://tinyos.stanford.edu/tinyos/dists/ ubuntu/pool/main/t/ (observar a plataforma utilizada) Os comandos abaixo realizam a instalação dos pacotes.

\$ sudo dpkg -i nesc_1.3.6-tinyprod_amd64.deb
\$ sudo dpkg -i tinyos-base_2.1-20080806_all.deb
\$ sudo dpkg -i tinyos-tools_1.4.2-20120808_amd64.deb

7. Instalar o *TinyOS* a partir do código fonte:

```
$ cd /opt
$ sudo wget http://github.com/tinyos/tinyos-release/
    archive/tinyos-2_1_2.tar.gz
$ sudo tar xf tinyos-2_1_2.tar.gz
$ sudo mv tinyos-release-tinyos-2_1_2 tinyos-main-2.1.2
$ sudo chmod 777 -R tinyos-main-2.1.2
$ sudo chown -R usuário:usuário tinyos-main-2.1.2
```

8. Definir variáveis de ambiente

\$ cd /opt/tinyos-main-2.1.2
\$ sudo gedit tinyos-env

9. Insira o seguinte conteúdo no arquivo tinyos-env criado:

```
export TOSROOT="/opt/tinyos-main-2.1.2"
export TOSDIR="$TOSROOT/tos"
export CLASSPATH=$CLASSPATH:$TOSROOT/support/sdk/java/tinyos.jar:.
export MAKERULES="TOSROOT/support/make/Makerules"
export PYTHONPATH=$PYTHONPATH:$TOSROOT/support/sdk/python
echo "Caminho do TinyOS:"$TOSROOT
$ cd
$ sudo gedit .bashrc
```

10. Insira o seguinte comando no arquivo .bashrc:

source /opt/tinyos-main-2.1.2/tinyos-env

- 11. Reinicie o terminal
- 12. Verificando a instalação

\$ tos-check-env

13. Alterando as versões do Python e do gcc para uso do TOSSIM:

```
$ cd /opt/tinyos-main-2.1.2/support/make
$ sudo gedit sim.extra
```

14. Comente a linha:

15. Insira a linha:

16. Salve e feche o arquivo sim.extra. No terminal:

```
$ sudo apt-get install python-software-properties
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test
$ sudo apt-get update
$ sudo apt-get install gcc-4.8
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/
gcc-4.8 50
```

17. Testando o TOSSIM:

```
$ cd /opt/tinyos-main-2.1.2/apps/Blink/
$ make micaz sim
```