Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Proposta para Redução de Consumo de Energia em Redes de Sensores Sem Fio

Paula Letícia Santos Lima

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Priscila A. Solís Mendez Barreto

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Prof.ª Dr.ª Alba Cristina M. A. de Melo

Banca examinadora composta por:

Prof.ª Dr.ª Priscila A. Solís Mendez Barreto (Orientadora) — CIC/UnB
Prof. Dr. Jacir Luiz Bordim — CIC/UnB
Prof. Dr. Georges Daniel Amvame Nze — ENE/UnB

CIP — Catalogação Internacional na Publicação

Lima, Paula Letícia Santos.

Uma Proposta para Redução de Consumo de Energia em Redes de Sensores Sem Fio / Paula Letícia Santos Lima. Brasília : UnB, 2015. 83 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2015.

1. rede de sensores, 2. energia, 3. TDMA, 4. CSMA, 5. região de intensidade.

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Proposta para Redução de Consumo de Energia em Redes de Sensores Sem Fio

Paula Letícia Santos Lima

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Priscila A. Solís Mendez Barreto (Orientadora)
CIC/UnB

Prof. Dr. Jacir Luiz Bordim
CIC/UnB

Prof. Dr. Georges Daniel Amvame Nze
ENE/UnB

Prof.ª Dr.ª Alba Cristina M. A. de Melo
Coordenadora do Mestrado em Informática

Brasília, 31 de Julho de 2015

Paula Letícia Santos Lima

Uma Proposta para Redução de Consumo de Energia em Redes de Sensores sem Fio

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre no Curso de Pós-graduação em Informática da Universidade de Brasília, pela Comissão formada pelos professores:

Orientadora: Prof. Dr.ª. Priscila América Solis Mendes Barreto (CIC/UnB)

Prof. Dr. Georges Daniel Amvame Nze (ENE/UnB)

Prof. Dr. Jacir Luiz Bordim (CIC/UNB)

Vista e permitida a impressão.
Brasília, 31 de julho de 2015.

Prof.ª Dr.ª Alba Cristina Magalhães Alves de Melo
Programa de Pós-Graduação em Informática
Departamento de Ciência da Computação
Universidade de Brasília

Dedicatória

Dedico esse trabalho a minha mãe Auxiliadora Lima.

Agradecimentos

Agradeço primeiramente a Deus, que ilumina meu caminho, por me dar sabedoria, força e coragem para enfrentar os desafios da vida e que colocou essas pessoas extraordinárias aqui citadas em minha vida.

Agradeço a Professora Priscila Solís pela orientação nesse trabalho.

Agradeço ao meu pai Laercio Lima (*in memorian*) por que sei de onde ele estiver ele está ao meu lado sempre e orgulhoso de mim.

Agradeço a minha mãe Auxiliadora Lima por todo incentivo, apoio, carinho, paciência e orações nesses anos que estou fora de casa. Obrigada por sempre me ouvir me dar bons conselhos, sem você eu não seria nada.

Ao meu irmão Laercio e meu sobrinho Lohan por todo o apoio, incentivo e carinho.

Agradeço ao meu primo Márcio Guerra por me acolher e me ajudar a trilhar esse caminho, sem a sua ajuda eu não teria conseguido, obrigada pelas risadas, pela companhia, pelos passeios, pela paciência... Você foi mais que um primo foi um irmão mais velho pra mim.

Agradeço a toda a minha família Santos e Lima que de certa maneira me ajudaram a chegar até aqui. Agradecimentos especiais a minha tia Nadir e minhas primas Márcia e Sylvia pelo incentivo dado desde antes do inicio dessa caminhada.

Agradeço aos meus amigos que fiz nesse tempo a Ariane, Daniel, Gustavo, Léia, Nilson e Vera pelos momentos que passamos juntos tanto os bons como os ruins, cada um de vocês tem um lugar no meu coração. Em especial ao Jeremias por todas as vezes que me ajudou, pelos momentos que passamos, pelas nossas conversas. Você foi um amigão na reta final deste trabalho!

Agradeço aos amigos Henrique e Kerlla pelas boas conversas de redes que tínhamos no laboratório, pelo incentivo e apoio mútuo que nos dávamos no dia a dia para poder seguir em frente.

Não poderia deixar de agradecer a Amanda e o Jonathan vocês foram essenciais na minha caminhada na UnB me ajudando desde o inicio com os nossos estudos em grupo, agradeço por todas as vezes que vocês abdicaram um pouco de seu tempo em prol do meu.

Jon, sem você eu não conseguiria terminar, vou te agradecer eternamente por tudo que você fez por mim, não tenho palavras para expressar a minha gratidão a você!

Amanda, você foi minha melhor amiga em Brasília, obrigada pela força, pelo apoio, pelo ombro amigo, pelo ouvido amigo, obrigada por tudo! Também não tenho palavras para descrever a minha gratidão a você.

Agradecimentos especiais a CAPES pelo apoio financeiro, ao Departamento de Ciência da Computação, em especial a Paula, e à UnB pela excelência nos seus cursos de graduação e pós-graduação, não poderia deixar de agradecer as Professoras Alba e Maria Emília por todas as vezes que precisei da ajuda de vocês sempre estiveram prontas a me ajudar, agradeço pelo enriquecimento pessoal e profissional que vocês me passaram.

Resumo

O desenvolvimento da tecnologia e da microeletrônica possibilitou a construção de sensores sem fio com tamanho reduzido e a baixo custo, tornando o estudo das Rede de Sensores Sem Fio (RSSF) cada vez mais interessante na comunidade acadêmica. Esse crescente interesse ao longo dos anos advém das inúmeras aplicações em que uma das limitações mais desafiadoras é o consumo de energia. As estratégias utilizadas para o controle do gasto de energia são normalmente implementados nas camadas de rede ou de enlace, ou em ambas. O excesso de quadros de controle, de colisões e de recepção de quadros são fontes de desperdício de energia na qual a subcamada MAC deve estar atenta. Consequentemente, os mecanismos de acesso ao meio podem procurar uma otimização para superar estas limitações. Neste trabalho propõe-se um método para a melhoria da eficiência energética em uma rede de sensores sem fio em topologia homogênea e hierárquica. A proposta especifica o uso de *clusters* e *clusters heads* (CHs) para o encaminhamento de dados ao *sink* através do roteamento utilizando-se uma abordagem do caminho mais curto, baseada no algoritmo de *Dijkstra*, para calcular a rota de menor energia até o *sink*. A rede é dividida em duas regiões, a região de não intensidade (RNI) e a região de intensidade (RI), onde na RNI é utilizado o algoritmo CSMA/CA para enviar quadros e o algoritmo TDMA é utilizado para otimizar a comunicação entre os CHs, ideias obtidas a adaptadas neste trabalho a partir de [1] e [18]. Os resultados experimentais da proposta mostram uma diminuição do gasto de energia média na RSSF e uma acentuada diminuição do número de nós mortos, assim como um aumento na vida útil na rede em comparação aos trabalhos propostos.

Palavras-chave: rede de sensores, energia, TDMA, CSMA, região de intensidade.

Abstract

The development of technology and microelectronics enabled the construction of wireless sensors with small size and low cost, making the study of Wireless Sensor Network (WSN) increasingly interesting in the academic community. This growing interest over the years comes from the numerous applications in which one of the most challenging limitations is the energy consumption. The strategies used for energy control are usually implemented in link or network layers, or both. Excess control frames, collisions and reception frames may waste energy in which the MAC sublayer should be aware. Consequently, medium access mechanisms may seek an optimization to overcome these limitations. In this paper we propose a method for improving energy efficiency in a wireless sensor network in homogeneous and hierarchical topology. The proposal specifies the use of clusters and cluster heads (CHs) for routing data to the sink through using an approach based on the shortest path Dijkstra's algorithm. The network is divided into two regions, non-intensity region (NIR) region and the intensity (RI) where RNI is used in CSMA/CA algorithm to send frames, and the TDMA algorithm is used to optimize the communication between the CHs, adapted from [1] and [18]. The experimental results show a decrease in the average energy expenditure in WSN and a sharp decrease in the number of dead nodes, as well as an increase in the network lifespan in the network compared to the works.

Keywords: wireless sensor network, energy, TDMA, CSMA, region of intensity.

Sumário

Lista de Figuras

Lista de Siglas

ARPANet *Advanced Research Projects Agency Network*, 4

AWACS *Airborne Warning and Control System*, 5

B-MAC *Berkeley MAC*, 32

CMAC *Convergent MAC*, 32

COTS *Commercial Off The Shelf*, 6

CSMA/CA *Carrier sense multiple access with collision avoidance*, 32

DARPA *Defense Advanced Research Projects Agency*, 4

DPS-MAC *Dual Preamble Sampling MAC*, 32

DW-MAC *Demand Wakeup MAC*, 30

IoT *Internet of Things*, 2

LEACH *Low-Energy Adaptive Clustering Hierarchy*, 36

LEACH-C *Low-Energy Adaptive Clustering Hierarchy Centralized*, 38

MAC *Media Access Control*, 29

MEMS *Microscale Electro-Mechanical Systems*, 4

MH-MAC *Multimode Hybrid MAC*, 32

MIMO *Multiple Input and Multiple Output*, 14

NEMS *Nanoscale Electro-Mechanical Systems*, 4

NOAA *National Oceanographic and Atmospheric Administration*, 4

PMC Preparata, Metze e Chien, 14

RNI *Região Intensidade*, iv
RNI *Região de não Intensidade*, iv
RSDs Redes de Sensores Distribuídos, 5
RSSFs Redes de Sensores Sem Fios, 1

S-MAC *Synchronous MAC*, 30
Sink Estação Base / Sorvedouro, 8
SOSUS *Sound Survillance System*, 4

T-MAC *Timeout-MAC*, 30
TCP/IP *Transmission Control Protocol/Internet Protocol*, 4
TDMA *Time Division Multiple Access*, 30
TRAMA *TRaffic-Adaptive Medium Access*, 30

WiseMAC *Wireless Sensor MAC*, 30

Z-MAC *Zebra MAC*, 30

Capítulo 1

Introdução

Uma Rede de Sensores Sem Fio (RSSF) (do inglês *Wireless Sensor Networks*) é uma coleção de dispositivos sem fio autônomos com recursos energéticos limitados que pode ser móveis ou fixos, e estão localizados aleatoriamente em um ambiente em mudança dinâmica conhecido como campo de detecção. Cada dispositivo integra computação, comunicação sem fio e capacidade de sensoriamento [44].

De acordo com [44], as RSSF têm o intuito de observar e possivelmente controlar um determinado ambiente, normalmente sem intervenção humana direta em escala espacial ou temporal. Os nós sensores[1] monitoram e coletam informações sobre fenômenos físicos (por exemplo, temperatura, umidade, vibração, aceleração, ou qualquer outro evento que seja de interesse ao observador), realizam processamento local e disseminam os dados usando a comunicação através de ondas de rádio, até que a informação seja entregue ao nó que solicitou os dados. De acordo com [2], uma RSSF tende a ser dependente da aplicação a que se objetiva, pois os requisitos de *hardware*, *software* e os mecanismos de operação podem variar de acordo com a necessidade de aplicação.

A principal fonte de energia de uma RSSF geralmente é a bateria, e sensores muitas vezes são destinados a serem implantados ou lançados em campos de detecção inóspitos (por exemplo, campo de batalha, ambientes de radiação), tornando assim financeiramente ou logisticamente inviável recarregar ou substituir as baterias de todos os sensores. Os trabalhos [44], [36] e [2] afirmam que mesmo com essa restrição energética, é extremamente desejável que a vida útil de uma RSSF seja a mais longa possível para qualquer aplicativo de monitoramento. Esse sistema é medido pelo tempo em que todas as baterias dos nós sensores se esgotem e seja possível a coleta e encaminhamento de informação.

[1] Neste texto, os termos nó e nós sensores serão usados como sinônimos. Do ponto de vista mais formal, o termo nó sensor ou nó numa RSSF indica um elemento computacional com capacidade de processamento, memória, interface de comunicação sem fio, além de um ou mais nós sensores do mesmo tipo ou não

Dentre as características (auto-organização, energia, segurança, entre outros) das RSSF, independente de sua aplicação, a questão energética é a que mais impõe restrição no tempo de vida da rede. Isso faz com que diversas soluções como algoritmos para a eficiência energética (LEACH [17], PEGASIS [28], entre outros), algoritmos de roteamento e que otimizem o processo de envio e recepção de quadros, entre outros, sejam apresentadas como forma de estender ao máximo a durabilidade da rede sem sacrificar a confiabilidade do sistema [36]. No projeto de qualquer solução para a eficiência energética em aplicações, os algoritmos e protocolos de roteamento não podem ser escolhidos avaliando apenas sua elegância, mas devem otimizar o consumo de energia e os outros recursos limitados do dispositivo.

1.1 Justificativa

Atualmente, o termo *Internet of Things* (IoT), tem tido um grande interesse da comunidade científica, em que o monitoramento de meio ambientes, a movimentação de pessoas, a segurança, entre outras aplicações, os sensores são considerados como parte integrante da estrutura que está interconectada à Internet. Embora os nós sensores não tenham uma padronização, existem sérios problemas para definir protocolos dentro do padrão de uma arquitetura em camadas, que viabilizem ainda uma otimização de recursos nas RSSF, tais como energia, processamento [14] e memória.

A IoT é uma tecnologia que viabiliza os ambientes inteligentes, que são espaços digitais nos quais a interação deixa os limites do computador e se torna embutida na infraestrutura. Estes ambientes podem ser combinados para criação de aplicações específicas tais como as cidades inteligentes, automatizando a coleta e o processamento de informações e no desenvolvimento de novos padrões comportamentais da sociedade [15].

Neste sentido este trabalho procura abordar um assunto específico dessa problemática que é melhorar a eficiência energética, visto que esse é um dos maiores problemas em RSSF.

1.2 Objetivo

O objetivo deste trabalho é propor um método para combinar protocolo MAC com roteamento e avaliar a sua eficiência em termos de custo energético de uma RSSF. A proposta tem como base os trabalhos [1] e [18] e uma proposta do roteamento com base no caminho mais curto. Os objetivos específicos deste trabalho são:

- Integração do protocolo *funneling*-MAC [1] com LEACH-C [18] em uma RSSF;

- Avaliação da proposta a partir de simulação e comparação com métodos já existentes;
- Definição de um método de roteamento do menor caminho entre os *cluster heads* de uma RSSF.

1.3 Contribuições

A principal contribuição deste trabalho é a proposta de um método de roteamento entre os *Clusters Heads* (roteamento intercluster) com base no algoritmo de *Djikstra* com a métrica de energia residual e a junção de duas propostas do estado da arte para acesso à camada de enlace em uma RSSF uma topologia homogênea e hierárquica.

1.4 Estrutura do Trabalho

Este trabalho está estruturado da seguinte maneira:

- O Capítulo 2 apresenta a definição bem como um breve histórico de RSSF. Além disso, são mostrados conceitos gerais e alguns dos desafios mais importantes nestas redes, como eficiência energética e roteamento.
- O Capítulo 3 descreve os trabalhos relacionados e os avanços relativos à camada MAC, roteamento e economia de energia em RSSF, o uso de *Time Division Multiple Access* (TDMA) e *Carrier Sense Multiple Access/Collision Avoidance* (CSMA/CA), assim como alguns protocolos MAC de RSSF. Por fim será apresentado o resumo de algumas topologias e configurações em RSSF, e os desafios desses protocolos.
- O Capítulo 4 descreve o estado da arte e os paradigmas usados como base para essa dissertação, incluindo suas propostas e funcionamento. A proposta deste trabalho centrado em uma abordagem para minimização de consumo de energia em RSSF, mostra as simulações, resultados e análises.
- O Capítulo 5 apresenta a conclusão e os trabalhos futuros.

Capítulo 2

Redes de Sensores Sem Fio

2.1 Definição de Redes de Sensores Sem Fio

Neste Capitulo será visto na Seção 2.1 a definição de redes de sensores e o histórico destas redes. Na Seção 2.2 serão descritos alguns conceitos gerais e a descrição do algoritmo mais curto. Na Seção 2.3 será feita uma descrição de alguns desafios como eficiência energética e roteamento de uma RSSF e alguns tipos de protocolos de roteamento.

2.1.1 Evolução das RSSF

Há quatro fases que caracterizam a evolução das redes de sensores sem fio [7].

A 1ª fase descreve como os nós sensores foram usados na Guerra Fria. A 2ª fase descreve as extensivas pesquisas patrocinadas pela *Defense Advanced Research Projects Agency* (DARPA), em que o objetivo era determinar se novos protocolos *Transmission Control Protocol/Internet Protocol* (TCP/IP) e o *Advanced Research Projects Agency Network* (ARPANet), o antecessor da Internet, poderiam ser usados no contexto de redes de sensores.

A 3ª fase a RSSF se concentra nas aplicações militares e finalmente a 4ª fase descreve os nós sensores atuais, compactos e de baixo custo baseado em um número de tecnologias de alta densidade, incluindo Sistemas Eletromecânicos em Escala Micrométrica (do inglês, *Microscale Electro-Mechanical Systems*-MEMS), e Sistemas Eletrônicos em Escala Nanométrica (do inglês, *Nanoscale Electro-Mechanical Systems*-NEMS). A seguir, as 4 fases serão descritas com mais detalhes.

Fase 1: Redes de Sensores na Fase da Guerra Fria – Em [7], durante a Guerra

Fria, extensivas redes de nós sensores acústicos[1] foram desenvolvidas pelos Estados Unidos para vigilância submarina. Alguns desses nós sensores (hidrofones) ainda são usados pela *National Oceanographic and Atmospheric Administration* (NOAA), para detecção de submarinos inimigos. Este programa foi chamado de SOSUS (*Sound Survillance System*) e é atualmente utilizado para monitorar acontecimentos no fundo do oceano (movimento de animais, atividade sísmica, entre outros.)[48].

Nos Estados Unidos e no Canadá, ainda durante a guerra fria, foram instalados redes de defesa aérea com radares, para proteger contra ataques de aviões, onde os nós sensores faziam ao longo dos anos os sistemas de defesa aéreos tornaram-se cada vez mais sofisticados, incluindo aeróstatos[2] com nós sensores e aviões AWACS (*Airborne Warning and Control System*).

Fase 2: Iniciativas da DARPA – Um expressivo incremento em pesquisas em redes de sensores sem fio aconteceu no início da década de 1980, por meio de programas patrocinados pela DARPA. As pesquisas em Redes de Sensores Distribuídos (RSDs) objetivavam determinar se os novos protocolos TCP/IP e o ARPAnet poderiam ser usados no contexto de redes de sensores.

A pesquisa e desenvolvimento das RSDs proporcionaram a produção de vários nós sensores de baixo custo que foram distribuídos para operarem de forma colaborativa, ainda que cada nó tenha sua própria automaticidade, pois eles fazem seu próprio processamento, enviam e recebem pacotes/dados.

As RSDs focaram em computação distribuída, processamento de sinais e rastreamento. As principais tecnologias incluíam nós sensores acústicos, protocolos de comunicação em alto nível, processamento de algoritmos (e.g., algoritmos de auto-localização para nós sensores) e programas distribuídos (possibilidade de modificação dinâmica de sistemas distribuídos) [7].

Segundo os pesquisadores da Universidade de Carnegie Mellon [41] realizaram projetos para o desenvolvimento de sistemas de rede para acesso transparente flexível visando distribuir recursos; pesquisadores do Instituto Tecnológico de Massachusetts [32] focaram em técnicas baseadas em processamento de sinais.

Testes foram desenvolvidos para o rastreamento de múltiplos alvos em um ambiente distribuído e todos os componentes foram projetados e fabricados especificamente para esses testes. Vários outros trabalhos ao longo da década de 80 resultaram no desenvolvimento de um algoritmo de rastreamento de múltiplas hipóteses [35] para solucionar

[1] Os nós sensores acústicos são um caso especial em que um transdutor é usado para gerar uma onda a partir da compressão de um fluido médio (ar ou água)

[2] Aérostatos: é a designação dada às aeronaves mais leves que o ar. A atividade e o estudo dos aeróstatos é levada a cabo por um ramo da Aeronáutica denominado Aerostação.

problemas envolvendo alta densidade de alvos, detecções perdidas e alarmes falsos [46].

Fase 3: Aplicações Militares nas décadas de 80 e 90 – Essa fase é a primeira geração relativa a produtos comerciais ligados a tecnologias de RSSFs. Baseado nos resultados gerados pelas pesquisas e os testes desenvolvidos pela DARPA em RSDs, projetistas militares decidiram nas décadas de 80 e 90 adotar a tecnologia de redes de sensores, tornando-a um fator importante em cenários de guerra. Um esforço foi feito para iniciar o emprego das tecnologias *Commercial Off The Shelf* (COTS) [3] e interfaces de redes comuns, de modo a reduzir o custo e tempo de desenvolvimento.

Em ambientes de guerra, as redes de sensores podem melhorar o desempenho de detecção e rastreamento, com o uso de múltiplas observações, diversidade geométrica, região de detecção extendida e tempo de resposta mais rápido [7]. Nessa geração, o tempo de vida médio de operação das redes de sensores podia alcançar vários dias.

Os avanços principais neste período foram a consolidação do protocolo TCP/IP nas redes NSFNET, ANSNET e vBNS [7]. Entretanto, não foi definido ou consolidado nenhum padrão nas RSSF que permitisse a sua integração na internet.

Fase 4: Redes de Sensores sem Fio na época atual – Segundo [7], essa fase é conhecida como a segunda geração comercial de redes de sensores sem fio. Avanços nos sistemas de comunicações e de processamento computacional que ocorreram no fim da década de 90 e começo do ano 2000, resultaram em uma nova geração de tecnologias de redes de sensores. Essa evolução está relacionada à evolução dos nós sensores tradicionais, nós sensores compactos e de baixo custo baseado em um número de tecnologias de alta densidade, incluindo MEMS e NEMS.

O surgimento dos NEMS proporcionou o desenvolvimento de nós sensores de baixo custo e consumo de energia. Avanços em redes baseadas no IEEE 802.11 a/b/g e outros sistemas sem fio, como o *Bluetooth* [51], ZigBee [43] e WiMax [37] facilitam a conectividade de forma confiável e pervasiva.

2.1.2 Redes WPAN 802.15

A norma WPAN 802.15 define o padrão de rede de área pessoal sem fios, onde se prevê a ligação entre dispositivos distanciados até 300 metros [13]. O padrão 802.15 existe para garantir as especificações de redes pessoais com baixa potência e custo reduzido. A norma define os seguintes tipos de redes:

- **802.15.1-** É uma rede que se baseia na tecnologia *Bluetooth* e que permite a ligação sem fios de dispositivos fixos, portáteis e móveis (celulares e PDA's);

[3] COTS é usado para tecnologias de equipamentos e produtos de computação em geral, que estão à venda, financiados ou licenciados para o público geral.

- **802.15.2**-É uma rede que permite facilitar a coexistência das redes WPAN com redes WLAN;
- **802.15.3**- Rede dimensionada para velocidades elevadas (11-55Mbit/s) para aplicações multimídia e que necessitem de uma qualidade de serviço bastante elevada;
- **802.15.4**- É uma rede de complexidade muito reduzida que funciona com velocidades baixas. Esta característica permite um consumo reduzido permitindo que a bateria dure bastantes meses ou anos. As redes 802.15.4 podem ser utilizadas em dispositivos *ZigBee* com alcance máximo de 300 metros [13];
- **802.15.5**- Esta norma permite a coexistência numa rede em malha de uma rede de baixa velocidade e outra de velocidade elevada garantindo a intemporalidade, estabilidade e também a escalabilidade entre ambas as redes;
- **802.15.6**- A norma 802.15.6 foca-se nas redes *Body Area Network* (BAN) que utilizam sensores espalhados pelo corpo humano, na roupa ou mesmo debaixo da pele. Desta forma o corpo humano torna-se um meio de transmissão ou recepção [13];
- **802.15.7**- A norma 802.15.7 define a camada física e a camada de acesso ao meio de uma rede que permite em espaço livre a comunicação óptica utilizando a luz visível. Este tipo de rede surgiu em Janeiro de 2009;
- **802.15.8**- Registado para um padrão da próxima geração de redes sem fios. Permite facilitar e estimular as apresentações e discussões sobre novas tecnologias sem fios que pode permitir criar novas redes 802.15.

2.1.3 Redes de Sensores Sem Fio

Uma RSSF é composta por vários nós sensores e, normalmente, uma estação base (*sink*). Os nós sensores coletam as informações e as enviam para a estação base, que pode se comunicar com outras redes, para que os dados de interesse sejam analisados [27].

2.1.4 Nó Sensor

Um nó sensor é composto por cinco principais componentes: bateria, memória, processador, transceptor e dispositivo de sensoriamento [25]. A bateria armazena a energia do sensor, que além de ter capacidade limitada, tem pouca possibilidade de reposição, pois geralmente ficam em lugares inóspitos. A capacidade da memória e do processador são reduzidas devido ao tamanho, a Figura 2.1 ilustra a arquitetura de um nó sensor.

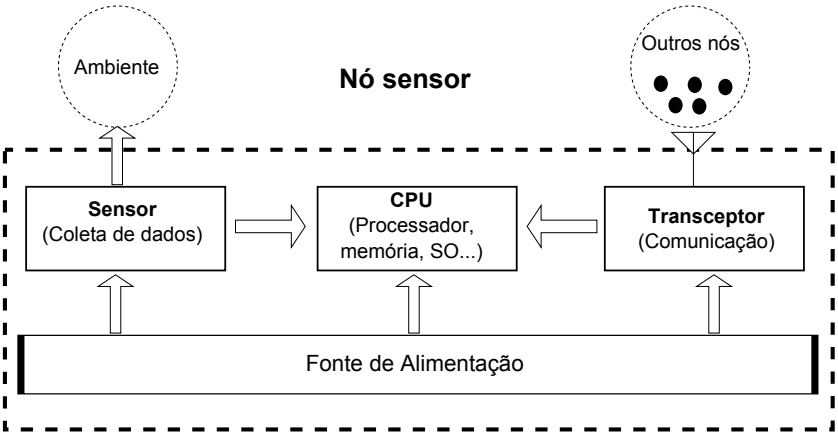


Figura 2.1: Arquitetura do nó sensor, adaptado de [25]

2.1.5 Estação Base/Sink

Um *sink* é uma entidade para os nós sensores enviarem as informações que coletam. Pode ser um computador pessoal, um servidor ou um *gateway*[4], que fornece conexão física com a Internet [25]. O *sink* não possui as restrições de um nó sensor, tendo grande poder de processamento, alta capacidade de armazenamento e sem restrições de bateria. A Figura 2.2 mostra um exemplo de RSSF na qual o *sink* envia os dados para o usuário através da internet.

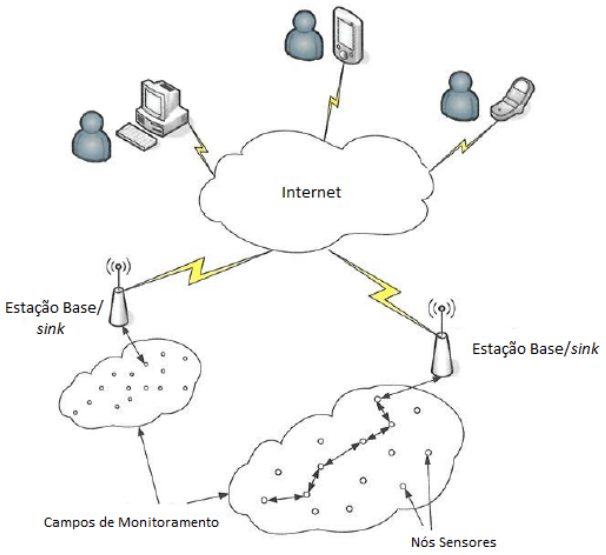


Figura 2.2: Exemplo de uma Rede de Sensores Sem Fio, adaptado de [10]

[4]Gateway é um intermediário geralmente destinado a interligar redes, separar domínios de colisão, ou mesmo traduzir protocolos. Exemplos de *gateway* podem ser os roteadores, celulares e *firewalls*, já que ambos servem de intermediários entre o utilizador e a rede.

2.2 Conceitos Fundamentais de RSSF

Em uma RSSF os nós sensores são distribuídos em uma área e buscam comunicar-se um com o outro para um objetivo comum que é entregar os pacotes de dados ao *sink*. Nessas redes ocorrem alguns problemas ao longo do tempo como a ineficiência energética, onde alguns nós podem morrer, falhar na entrega dos pacotes, sobrecarga de comunicação, entre outros.

Em [10], muitos nós sensores conectam-se uns com os outros e processam suas estações diretamente, fazendo a coleta dos dados para o *sink*. Isto é importante para muitas aplicações da rede que requerem centenas ou milhares de nós sensores, muitas vezes usados em áreas remotas e de difícil acesso. Portanto, um sensor sem fio não tem apenas a detecção do evento, mas também um processamento *on-board*, comunicação e capacidade de armazenamento.

Com essas melhorias de processamento e comunicação, um nó sensor não é só responsável pela coleta dos dados, como também pela análise dentro da rede, a fusão dos seus próprios dados e dos dados de outros nós sensores. Quando muitos nós sensores de forma cooperativa monitoram um ambiente, eles formam uma RSSF.

Os nós sensores não somente comunicam-se entre si, mas também se comunicam com o *sink* usando comunicação sem fio, o que lhes permite divulgar seus dados para o processamento, análise e armazenamento. Por exemplo, a Figura 2.3 mostra dois campos sensores monitorando duas diferentes regiões geográficas e conectando-se à Internet através do *sink*. A capacidade de um nó dentro de uma RSSF pode variar largamente, isto é, um nó pode monitorar um fenômeno físico único, enquanto dispositivos mais complexos podem combinar diferentes técnicas de sensoriamento (e.g., acústica, ótica, magnética).

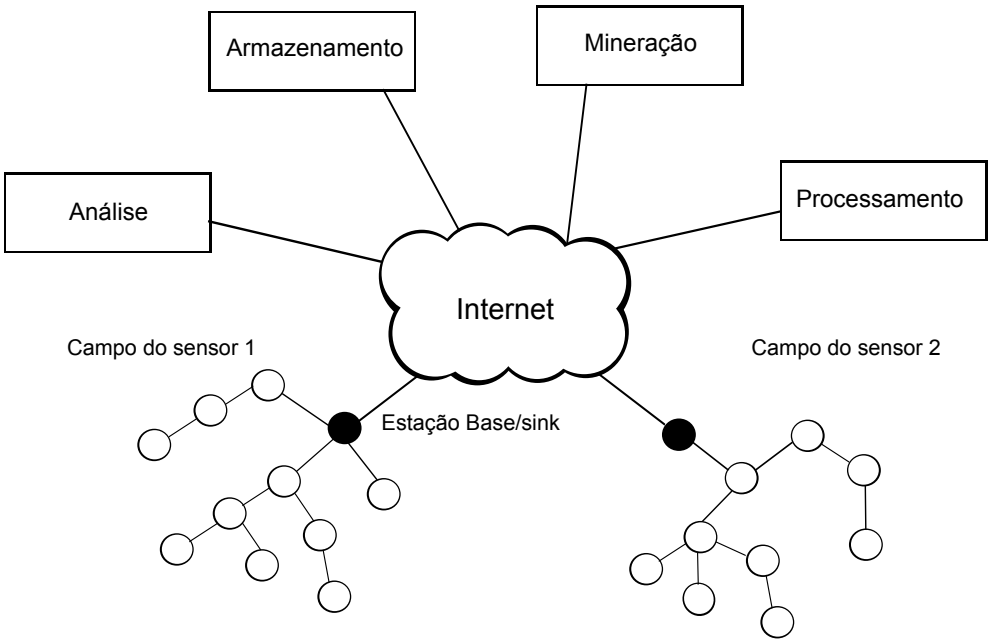


Figura 2.3: Redes de Sensores Sem Fio, adaptada de [10]

Os nós sensores podem também diferenciar suas capacidades de comunicação, por exemplo, usando ultrassom [12], infravermelho [9], entre outros. Enquanto um nó pode somente coletar e divulgar a informação sobre o ambiente observado, dispositivos mais poderosos (os dispositivos com grande processamento, energia e capacidade de armazenamento) podem também realizar extensas funções de processamento e agregação.

Tais dispositivos assumem responsabilidades adicionais em RSSF, podendo formar comunicações em *backbones*[5] que pode ser usado em outros recursos dentro da rede.

2.2.1 Comunicação em RSSF

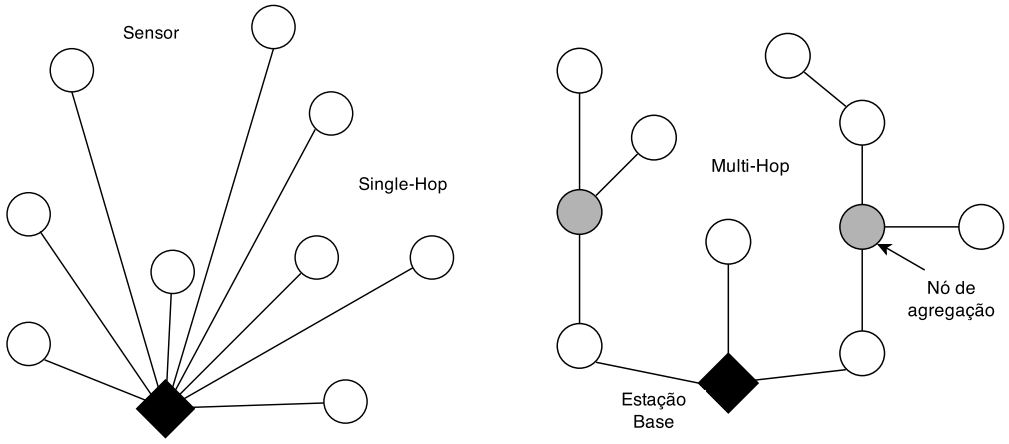
O padrão IEEE 802.11 foi introduzido em 1997, utiliza bandas de frequência diferentes, por exemplo, a banda de 2,4GHz usada em IEEE 802.11b e IEEE 802.11g, enquanto o protocolo IEEE 802.11a usa 5GHz de banda. O IEEE 802.11 foi frequentemente usado nas primeiras redes de sensores sem fio e ainda pode ser encontrado em redes atuais quando as demandas de largura de banda são elevadas (por exemplo, para sensores de multimídia) [10]

Quando os intervalos de transmissão das rádios de todos os nós sensores são grandes o suficiente e os sensores podem transmitir seus dados diretamente para o *sink*, eles podem formar a topologia estrela como mostra a Figura 2.4a. Nessa topologia, cada nó se comunica diretamente com o *sink* usando apenas um salto (*single-hop*). No entanto, redes de sensores muitas vezes cobrem uma grande área geográfica e a potência de transmissão

[5] *Backbones* designa o esquema de ligações centrais de um sistema mais amplo de elevado desempenho

de rádio deve ser mantida no mínimo, a fim de economizar energia; consequentemente, comunicação usando múltiplos saltos (*multi-hop*) é o caso mais comum para RSSF, como mostra a Figura 2.4b.

Os exemplos anteriores induzem a perceber que o roteamento, que é a tarefa de encontrar um caminho *multi-hop*, ou seja com vários saltos de um nó para o *sink*, é um dos desafios mais importantes. Quando um nó funciona como um retransmissor para várias rotas, muitas vezes ele tem a oportunidade de analisar os dados do sensor e pré-processamento na rede, isso pode levar à eliminação de informações redundantes.



(a) Comunicação *single-hop* (b) Comunicação *multi-hop*

Figura 2.4: Comunicação em RSSF, adaptado de [10]

2.2.2 Padrões de camada física de RSSF

O IEEE 802.15.4 [24] (Figura 2.5) é uma padronização das camadas física e de acesso ao meio para dispositivos de baixo custo, que possuem limitações severas de energia e que enviam dados a baixas taxas [24]. O 802.15.4 é uma alternativa ao *Bluetooth*, pois aumenta o número de dispositivos suportados, e possui implementação mais simples e de menor custo. Foi desenvolvido para aplicações de automação doméstica e industrial, monitoração ambiental e entretenimento, além de RSSF . São permitidos até 2^{64} dispositivos na rede, distribuídos em uma ou mais redes, coexistindo até 2^{16} redes em uma região.

O padrão especifica duas frequências de operação: 2.4 GHz com banda total de 250 kbps, e 868/915 MHz operando a 20/40 kbps, respectivamente. A frequência de operação também determina outras características, como área máxima de cobertura do sinal, interferência e modulação [6]. A banda é dividida em canais, sendo todos estes utilizados por padrão, mas que podem ser selecionados dinamicamente por outras camadas, como uma forma de atenuar interferências com outras redes ou evitar canais com ruído alto.

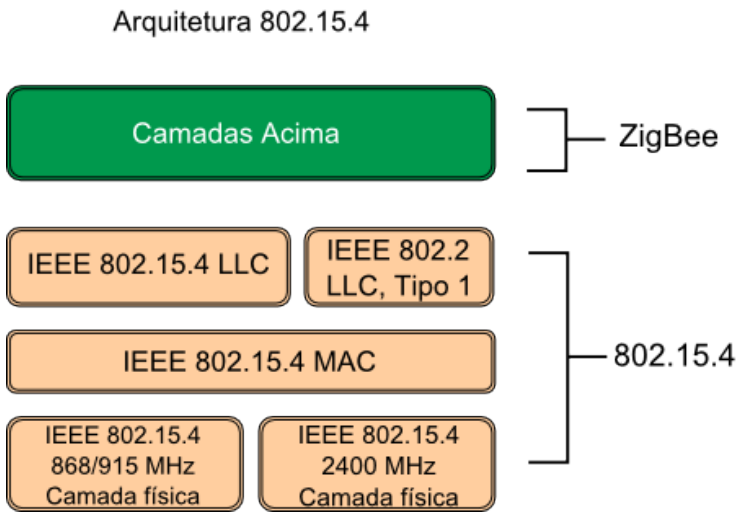
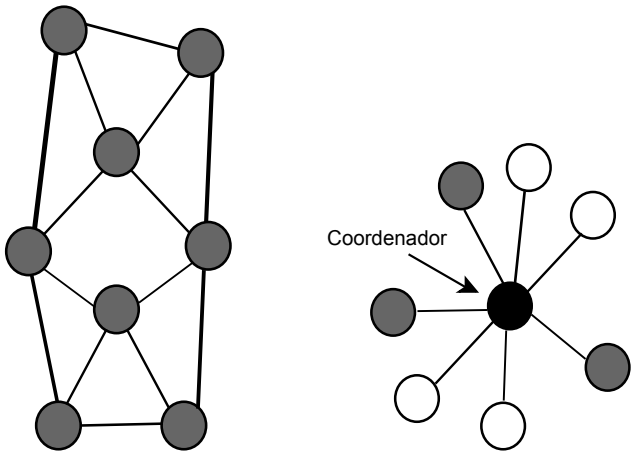


Figura 2.5: Camada física do padrão 802.15.4, adaptado de [49].

O padrão foi desenvolvido para duas topologias: estrela e ponto-a-ponto, definidos pela camada de controle de acesso ao meio, ilustradas na Figura 2.6. A configuração da rede permite o uso de dispositivos mais simples, permitindo assim redes de baixo custo.



(a) Topologia Ponto-a-Ponto (b) Topologia Estrela

Figura 2.6: Organizações de uma rede no padrão IEEE 802.15.4, adaptado de [10].

Redes de sensores sem fio normalmente operam em meios de comunicações com restrições de largura de banda. Alguns dispositivos de nós sensores são baseados em um único canal de rádio frequência operando em 916 MHz. Há sensores que utilizam um transceptor compatível com a tecnologia *Bluetooth* em 2,4 GHz, com um sintetizador de frequência integrado. Outros sistemas utilizam também 2,4 GHz (tecnologia IEEE 802.11b), 5,0 GHz (tecnologia IEEE 802.11a), ou possivelmente outras bandas (IEEE 802.15.4 ou IEEE 802.16).

2.2.3 Auto configuração

As características de uma RSSF são influenciadas pela aplicação e pela variabilidade das condições do ambiente. Além disso, a estrutura da rede tende a se modificar durante o seu tempo de vida, seja por falha ou desvanecimento de energia dos nós. Logo, os protocolos de controle de acesso ao meio devem possuir mecanismos que permitam seu ajuste à variabilidade do ambiente afim de otimizar seu funcionamento e economizar energia. As características desses protocolos podem ser subdividas quanto ao modo de adaptação em:

- **Estático:** não permitem à aplicação ajustar os parâmetros de configuração dos protocolos em tempo de execução. Os parâmetros são definidos no momento de compilação do código ou durante a programação do nó, se mantendo inalterados durante todo o tempo de vida da rede. Esses parâmetros dos protocolos estáticos são simples de programar e em geral demandam menos recursos de memória e processamento, permitindo seu uso em ambientes com restrição de recursos.

  Os parâmetros podem ser estáticos devido a uma decisão de projeto ou limitações de *hardware*, como por exemplo frequência e potência de transmissão do rádio. Além disso, o emprego de parâmetros estáticos limita a aplicabilidade do protocolo a um tipo de rede ou cenário específico (só o cenário padrão por exemplo).

- **Reconfigurável:** permitem a mudança de parâmetros pelo operador ou pela aplicação em tempo de execução. A mudança dos parâmetros pode ser desencadeada pela aplicação ou por recebimento de um comando do operador. A reconfiguração aumenta a aplicabilidade do protocolo a vários cenários.

  Para isto a aplicação deve adicionar uma lógica que ajuste os parâmetros em tempo de execução às condições da rede. Outra forma de reconfiguração é o envio de comandos do operador, permitindo a tomada de decisão fora da rede, utilizando algoritmos mais complexos. Essa abordagem demanda mais recursos de *hardware* e está sujeita a falhas, como por exemplo erros na programação da aplicação ou da transmissão do comando do operador.

- **Auto-configurável:** protocolos que permitem a mudança automática dos parâmetros, ajustando-se à variabilidade do ambiente sem a influência do operador. Esses ajustes são realizados por uma lógica interna pré-definida no protocolo. Caso essa lógica seja modificada durante o tempo de execução, consideramos que o protocolo é reconfigurável e auto-configurável [43].

  Em [43], a auto-configuração permite ao protocolo se ajustar às condições adversas do ambiente e à múltiplas configurações em pontos distintos da rede. Para tanto o

código é mais complexo e geralmente necessita de mais recursos em comparação com a reconfiguração, pois deve levar em conta todos os estados possíveis de operação. A autoconfiguração pode ser baseada em algoritmos distribuídos[30], portanto os nós estarão sujeitos a falhas bizantinas decorrentes de problemas de sincronização.

Algoritmo do Caminho Mais Curto

O algoritmo de *Dijkstra* é um clássico algoritmo computacional para o cálculo da distância mínima entre um vértice e todos os demais vértices de um grafo. Foi desenvolvido pelo premiado matemático holandês Edsger Dijkstra (1930 - 2002) e publicado na edição número 1 do periódico *Numerical Mathematics*, em 1959 (Dijkstra, 1959).

Uma das razões de não se usar *Dijkstra* em RSSF é que a rede é dinâmica, um exemplo disso é quando os nós estiverem encaminhando os dados, se o nó morrer ao longo da rota o caminho irá se perder o que torna a eficiência e vida útil da rede ruim. Logo *Dijkstra* não é uma boa opção para RSSF. Os algoritmos de RSSF tentam tirar vantagem do fato de poder se comunicar com vários ao mesmo tempo e nessa comunicação tentam rotear da melhor forma.

Alguns trabalhos tem aplicado esse paradigma para roteamento. Em [34], apresenta um novo mecanismo para a identificação do caminho mais curto e gastando menos energia. Essa abordagem faz uso de roteamento pró-ativo de modo que cada nó conhece o seu caminho em direção ao nó CAC (do inglês *Closest Adjacency Condition*), esse nó fica próximo ao *gateway* e ativo o tempo todo, enquanto os outros nós permanecem no modo de sono. Quando um nó tem que enviar suas informações, antes de enviá-la vai acordar os nós ao longo de sua rota. Quando os nós CAC recebem essa informação transmitem os dados e mandam os nós daquela rota entrarem em modo de suspensão. A questão crítica nesta solução é que um nó CAC permanece ativo durante todo o tempo, a sua fonte de energia estará esgotada rapidamente. Esse problema foi resolvido usando uma bateria secundária.

Em [8], o *Optimized Link State Routing Protocol* (OLSR) é um protocolo de roteamento desenvolvido para operar em redes móveis Ad Hoc. É um protocolo do tipo proativo, no qual os nós trocam informações de topologia periodicamente. O protocolo OLSR adota o esquema de encaminhamento de pacotes baseado em tabela de rotas. Para diminuir a intensidade de inundação (*flooding*) da rede com mensagens de sinalização, cada nó seleciona um *Multipoint Relay* (MPR) para propagar suas mensagens.

No OLSR, somente os nós eleitos como MPRs são responsáveis por encaminhar o tráfego de controle, permitindo a diminuição no número dessas mensagens na rede. Os nós selecionados como MPRs anunciam para a rede informações de alcançabilidade a respeito dos nós que os selecionaram como MPR. Os nós da rede selecionam seus MPRs entre seus

vizinhos de um salto. Além das funcionalidades clássicas dos protocolos de roteamento Ad Hoc, o OLSR suporta funcionalidades complementares de extensões, como a redistribuição de rotas de outras redes e a redundância de MPRs. Na operação do protocolo, cada nó escolhe um conjunto de vizinhos para retransmitir suas mensagens (denominados MPRs).

Os nós "nãoMPRs" recebem as mensagens de vizinhos de um salto e não as encaminham. Desse modo, os MPRs devem ser escolhidos de forma que, em termos de alcance, todos os nós a dois saltos recebam as mensagens de atualização. A seleção de MPRs pode ser descrita pelo mecanismo de escolha e notificação de MPRs é implementado por meio da troca de mensagens HELLO, em que cada nó possui e mantém informações de um conjunto de MPRs (MPR Set), para armazenamento dos MPRs do nó, e de quais nós estão usando o roteador local como MPR, respectivamente. Seus pontos críticos são topologia da rede que não é estável, assim o protocolo apresenta um baixo desempenho. Isso ocorre devido à disseminação de informações da topologia, que necessita ser feita frequentemente causando um *overhead* de comunicação.

2.3 Desafios nas RSSF

Nesta Seção serão descritos os desafios de pesquisa mais importantes em RSSF, baseados nos trabalhos desenvolvidos [1], [47], [3] e [10].

2.3.1 Eficiência Energética

O primeiro e muitas vezes o mais importante desafio em se tratando de RSSF é a eficiência energética. Este requisito permeia todos os aspectos de nó sensor e da rede. Por exemplo, as escolhas feitas na camada física de um nó sensor afeta o consumo de energia de todo o dispositivo e a concepção de protocolos de nível mais alto.

O tempo de vida do sensor exibe uma forte dependência do tempo de vida da bateria, provocando falhas em rotas estabelecidas previamente e também alterando dinamicamente o trajeto do roteamento. Em vários casos, o nó sensor sem fio possui uma fonte de energia limitada (e.g., $\leq$ 500 mAh, 1,5 V) e a recarga ou substituição dessa fonte também pode ser limitada ou inviável.

A operação das baterias para sensores usadas em aplicações comerciais é baseada no uso de uma célula de lítio AA [42], e ultimamente vem sendo usado também placas solares para potencializar o tempo de vida do sensor [50]. Em [38] é utilizado um mote[6] chamado SunSPOT (fabricado pela antiga SUN, hoje Oracle). O mote é a peça central tanto nos

[6]Mote é o termo que descreve um dispositivo equipado por um micro controlador e mais alguns sensores analógicos, como o de temperatura.

nós da rede quanto no *sink*, eles são responsáveis por transmitir ou receber as informações captadas pelos sensores analógicos.

O gerenciamento e conservação de energia são funções críticas em redes de sensores e há a necessidade de projetos de algoritmos e protocolos que otimizem a utilização de energia nos nós sensores. A conservação de energia deve ser considerada como uma restrição básica ao projetar uma RSSF, uma vez que regula a vida útil da rede.

2.3.2 Roteamento em RSSFs

Para que os dados que trafegam por uma rede cheguem ao seu destino é necessário o calculo de rotas entre a origem e o destino. Os protocolos responsáveis por determinar estas rotas são chamados de protocolos de roteamento. Os protocolos de roteamento para RSSFs têm que ser projetados para serem tolerantes a falhas de nós individuais enquanto minimizam o consumo de energia.

Como a banda é limitada e tem de ser dividida entre todos os sensores, os protocolos de roteamento têm de ser capazes, usando colaborações locais, de reduzir os requerimentos por banda [17]. Além disso, os ambientes em que os nós necessitam operar, são muito dinâmicos com mudanças rápidas dos parâmetros físicos. Alguns dos parâmetros que podem variar dependendo da aplicação são:

- **Energia disponível:** Os nós sensores são alimentados por bateria, e tem a capacidade de energia limitada. A energia representa um grande desafio para os projetistas de redes em ambientes hostis, por exemplo, um campo de batalha, onde é impossível acessar os sensores e recarregar as baterias. Além disso, quando a energia de um sensor atinge um determinado limiar, o sensor irá tornar defeituoso e não será capaz de funcionar adequadamente, o que terá um impacto importante sobre o desempenho da rede. Assim, protocolos de roteamento projetados para sensores devem ser o mais eficiente possível como, prolongar a vida útil da rede, garantindo bom desempenho global.

- **Posições do sensor:** Outro desafio que enfrenta o projeto de protocolos de roteamento é gerenciar as posições dos sensores. A maioria dos protocolos propostos assumem que os sensores estão equipados com sistema de posicionamento global (GPS) ou usam alguma técnica de localização [5] para saber mais sobre suas localizações.

- **Recursos de hardware limitados:** Além de capacidade de energia limitada, os nós sensores têm também capacidade limitada de processamento e armazenamento, e, portanto, só pode executar funcionalidades computacionais limitadas. Estas restrições de hardware apresentam muitos desafios no desenvolvimento de *software* e

projeto de protocolo de rede para RSSF, que deve considerar não só a restrição de energia em nós sensores, mas também as capacidades de processamento e armazenamento de nós sensores.

- **Requisitos das aplicações de sensoriamento diversos:** Redes de sensores têm uma ampla gama de aplicações diversas. O protocolo de rede pode cumprir os requisitos de todas as aplicações. Portanto, os protocolos de roteamento devem garantir a entrega de dados e sua precisão para que o *sink*.
- **Escalabilidade:** Protocolos de roteamento devem ser capaz de escalar com o tamanho da rede. Além disso, os sensores podem não necessariamente ter as mesmas capacidades em termos de energia, processamento, sensoriamento, e em particular de comunicação. Assim, a comunicação entre os sensores pode não ser simétrica, ou seja, um par de sensores pode não ser capaz de ter uma comunicação em ambos os sentidos.

2.3.3 Tipos de Protocolos de roteamento

Protocolos baseados em localização

Em protocolos baseados em localização, nós sensores são abordados por meio de suas localizações. Essas informações de localização dos nós sensores é necessária para RSSF pela maioria dos protocolos de roteamento para calcular a distância entre dois nós específicos de modo a que o consumo de energia possa ser calculado. A seguir será apresentado alguns trabalho com esse tipo de roteamento:

- ***Geographic and Energy-Aware Routing* (GEAR):** GEAR [52] é um protocolo de roteamento eficiente de energia proposto para o encaminhamento de consultas para segmentar regiões no do campo sensor. Em GEAR, assume-se que os sensores são equipados com GPS, para que eles saibam suas posições atuais.

  Além disso, os sensores estão conscientes da sua energia residual, bem como os locais e energia residual de cada um dos seus vizinhos. GEAR usa heurísticas conscientes de energia que são baseadas em informações geográficas para selecionar sensores para rotear um pacote para a sua região de destino. Em seguida, o GEAR utiliza um algoritmo de encaminhamento geográfica recursivo a disseminar o pacote dentro da região de destino.

- ***Trajectory-Based Forwarding* (TBF):** TBF [33] é um protocolo de encaminhamento que requer uma rede suficientemente densa e a presença de um sistema de coordenadas, por exemplo, um GPS, de forma que os sensores podem posicionar-se

e estimar a distância aos seus vizinhos. A fonte especifica a trajetória em um pacote, mas não indica explicitamente o caminho em uma base *hop-by-hop*. Com base na informação sobre a localização dos seus vizinhos, um sensor de encaminhamento faz uma decisão ávido para determinar o próximo salto que é a mais próxima da trajetória fixa por o sensor de origem.

A manutenção da rota em TBF não é afetada pela mobilidade do sensor dado que uma rota de origem é uma trajetória que não inclui os nomes dos sensores de encaminhamento. A fim de aumentar a confiabilidade e capacidade da rede, é também possível implementar o roteamento *multipath* em TBF, onde um caminho alternativo é apenas uma outra trajetória. TBF também pode ser utilizado para descoberta de recursos. Uma outra aplicação interessante do TBF é proteger o perímetro da rede.

Protocolos de dados centralizados

Protocolos de dados centralizados diferem dos protocolos de endereços tradicionais centradas na maneira que os dados são enviados a partir de sensores de origem para o *sink*. Nos protocolos de endereço centrado, cada sensor fonte que tem os dados apropriados respondendo e enviando seus dados para o *sink*, independentemente de todos os outros sensores.

No entanto, em protocolos de dados centralizados, quando os sensores de origem enviam seus dados para o *sink*, os sensores intermediários podem executar alguma forma a agregação dos dados provenientes de vários sensores de origem e enviar os dados agregados para o *sink*. Este processo pode resultar em economia de energia por causa de menos transmissões necessárias para enviar os dados desde a origem até o *sink*. Como será apresentado a seguir:

- ***Sensor Protocols for Information via Negotiation* (SPIN):** O protocolo SPIN [16, 26] foi projetado para melhorar protocolos de *flooding* clássicos e superar os problemas que eles podem causar, por exemplo, implosão e sobreposição. Os sensores que executam os protocolos SPIN são capazes de calcular o consumo de energia necessária para calcular o envio e recebimento de dados através da rede. Assim, eles podem tomar decisões para o uso eficiente de recursos próprios.

  Os protocolos SPIN são baseados em dois mecanismos principais nomeadamente negociação e adaptação de recursos. O SPIN permite que os sensores negociem uns com os outros antes de qualquer divulgação dos dados, pode ocorrer a fim de evitar a inclusão de informações não úteis e redundante na rede. SPIN usa meta-dados como

os descritores dos dados que os sensores querem divulgar. A noção de meta-dados evita a ocorrência de sensores de sobreposição.

Pode-se notar que o tamanho dos meta-dados deve ser definitivamente inferior do que a dos dados de sensores correspondentes. Ao contrário da técnica de *flooding*, cada sensor está ciente do seu consumo de recursos com a ajuda de seu próprio gerenciador de recursos que é testado pela aplicação antes de qualquer processamento ou transmissão de dados. Isso ajuda os sensores para monitorar e adaptar-se a qualquer alteração em seus próprios recursos.

- ***Directed Diffusion*:** O protocolo *Directed Diffusion* [21, 22] é um protocolo de roteamento centrado para o sensor consultar a difusão e processamento dos dados. Ele atende às principais exigências da RSSF como a eficiência energética, escalabilidade e robustez. No início do processo de difusão dirigida, o *sink* especifica uma baixa taxa de dados para os eventos de entrada.

  Depois disso, o *sink* pode reforçar um sensor específico para enviar eventos com uma taxa de dados maior, reenviando a mensagem original, com um intervalo menor. Da mesma forma, se um sensor vizinho recebe esta mensagem encontra interesse e que o interesse do remetente tem uma velocidade de dados maior do que antes, e esta taxa de dados é maior do que a de qualquer gradiente existente, ele irá reforçar um ou mais dos seus vizinhos.

Protocolos Hierárquicos

Muitos projetos de pesquisa nos últimos anos têm explorado clusterização hierárquica em RSSF partir de diferentes perspectivas [45]. A clusterização é um protocolo de comunicação eficiente em termos de energia que pode ser utilizada pelos sensores para relatar seus dados detectados até o *sink*.

A seguir será descrito alguns protocolos em que uma rede é composta de vários aglomerados (*clusters*) de sensores. Cada *cluster* é gerenciado por um nó especial, chamada de *cluster head*, que é responsável pela coordenação das atividades de transmissão de dados de todos os sensores em seu *cluster*.

- ***Low-energy adaptive clustering hierarchy* (LEACH):** Em LEACH [17, 18], é o primeiro e mais popular algoritmo de agrupamento hierárquico eficiente em termos de energia para RSSF que foi proposto para reduzir o consumo de energia. Em LEACH, a tarefa de agrupamento é rodado entre os nós, com base na duração. A comunicação direta é usada por cada *cluster head* (CH) para transmitir os dados para a estação base (BS). Ele utiliza *clusters* para prolongar a vida útil da RSSF.

LEACH é baseado numa técnica de agregação (ou de fusão), que combina ou agrega os dados originais para um tamanho menor de dados que transportam apenas uma informação significativa para todos os sensores individuais, divide uma rede em vários conjunto de sensores, que são construídos usando coordenação localizada e controlam não só a redução de quantidade de dados que são transmitidos para o *sink*, mas também para fazer o roteamento e divulgação de dados mais escalável e robusta.

LEACH utiliza uma randomização de CHs de alta energia em vez de selecionar um em modo estático, para dar a possibilidade de todos os sensores para atuar como CHs e evitar o esgotamento da bateria de um sensor individual de morrer rapidamente. A operação de LEACH é dividido em *rounds* com duas fases cada a saber: (i) uma fase de configuração para organizar a rede em *clusters*, anuncia o CH , e a criação de programação de transmissão e (ii) uma fase estável de estado para agregação de dados, compressão e transmissão para o *sink*.

- ***Power-Efficient Gathering in Sensor Information Systems* (PEGASIS):** PEGASIS [29], é um protocolo para RSSF baseado no conceito de cadeias. Cada nó troca informações apenas com os vizinhos mais próximos formando uma cadeia entre os nós e apenas um nó é escolhido a cada momento para transferir as informações coletadas ao nó *gateway* (estação base).

  Portanto, o número de trocas de mensagens será baixo e a comunicação será realizada entre nós próximos uns dos outros. Espera-se com isso que a energia gasta seja menor, se comparada a outros protocolos que requerem muitas trocas de mensagens para eleger líderes e formar grupos, e protocolos em que os nós constantemente trocam mensagens com o nó *gateway* de forma direta (o *gateway* geralmente se encontra distante dos nós). Isto implica um tempo de vida maior para cada nó e um consumo menor da largura de banda da rede. O PEGASIS assume o seguinte:

- O nó *gateway* situa-se estacionado a uma distância fixa da rede;
- Os nós são capazes de transmitir dados diretamente para o nó *gateway* e para qualquer outro nó;
- Cada nó possui informação de localização dos outros nós;
- Os nós são homogêneos e com o nível de energia uniforme;
- Os nós não são móveis. A cada *round* um nó é escolhido para transmitir a informação à estação base.

Resumo do capítulo

Neste Capítulo foram abordados os conceitos principais de uma RSSF e um breve histórico de como surgiu essa tecnologia. Foram abordados alguns conceitos gerais como roteamento e como é feita a comunicação na rede que se dá através do *single-hop* e *multi-hop*, em que no *single-hop* os nós se comunicam diretamente com o *sink* e no *multi-hop* os nós se comunicam por mais de um salto para conseguir entregar os dados ao *sink*.

Capítulo 3

Comunicação em RSSF

Neste Capítulo será visto na Seção 3.1 o funcionamento dos protocolos de acesso ao meio compartilhado (MAC), e os desafios desses protocolos. Na Seção 3.2 será descrito o estado da arte.

3.1 Protocolos MAC

Os principais desafios para o projeto de protocolos MAC para RSSFs é que essas redes diferem de redes sem fio tradicionais em muitos aspectos, tais como: energia, disposição dos nós, densidade e sensoriamento. Por não tratar de vários desses aspectos, os protocolos MAC tradicionais não podem ser utilizados em RSSFs.

As RSSF devem ser eficientes em relação ao consumo de energia, uma vez que os nós sensores são operados por bateria e, devido à alta densidade e aos ambientes inóspitos onde estas redes podem ser implementadas, a recarga dessas baterias é praticamente inviável. Por isso, a característica que deve infligir maior atenção ao se projetar tais redes é o consumo de energia, permitindo o prolongamento no tempo de vida da rede.

Em muitas aplicações, os nós são depositados de maneira aleatória e devem se auto-organizar para estabelecer rotas de comunicação. Isso requer que essas redes sejam adaptáveis a mudanças e escaláveis em relação ao número de nós [11]. Finalmente, a maior parte do tráfego nessas redes é desencadeada por sensoriamento de eventos externos que podem ocorrer de uma forma inesperada.

3.1.1 Protocolos Síncronos

Em protocolos MAC síncronos, há um nó sensor de escuta para o canal durante um certo período de tempo. Se não ouvir qualquer horário a partir de outros nós sensores, ele determina sua próxima hora de despertar e transmite sua programação. Isso faz com

que o nó sensor seja um sincronizador. Se um nó sensor recebe uma programação de um vizinho antes de escolher o seu próprio horário, segue-se a programação recebida, o que o torna um seguidor.

Geralmente, todos os nós sensores em um *cluster*[1] são sincronizados com um sincronizador que é um ou alguns saltos de distância. Se um nó sensor recebe uma agenda diferente depois que ele define sua própria agenda (ou seja, qualquer um tem transmitido a sua própria agenda ou seguiu uma agenda previamente recebida), é adotada essa agenda para que nó possa ser uma ponte entre dois *clusters*.

Em [39], o *Synchronous MAC* (S-MAC), *Timeout-MAC* (T-MAC), *Demand Wakeup MAC* (DW-MAC) são exemplos de protocolos síncronos, *Zebra MAC* (Z-MAC), *TRaffic-Adaptive Medium Access* (TRAMA), *Wireless Sensor MAC* (WiseMAC), esses são exemplos de protocolos MAC de contenção [2].

Os protocolos síncronos em RSSF são baseados em agendas em que cada participante da comunicação possui um intervalo de tempo agendado exclusivo para sua transmissão. De posse dessa agenda, os potenciais destinatários ficam ativos para receber transmissões nos períodos pré-agendados. Demais remetentes também respeitam a agenda e esperam sua vez para transmitir. Logo, protocolos síncronos são capazes de tanto mitigar o problema de transmissão-em-vão, como o de colisões.

Por outro lado, eles precisam trocar quadros de controle para "acordar" e/ou anunciar uma agenda comum, o que resulta em sobrecarga de comunicação, se o nó sensor não usar o seu tempo, o canal fica ocioso. Ademais, as agendas são usualmente fixas e essa inflexibilidade pode acarretar maior atraso. Como exemplo podemos citar o protocolo *Time Division Multiple Access* (TDMA)[40] que é uma técnica de acesso múltiplo que permite que diversos usuários compartilhem recursos simultaneamente em intervalos de tempo (*slots*) pré determinados.

Em [47], é utilizado o protocolo TDMA onde cada nó acorda o seu *slot* correspondente e verifica se há alguma "mensagem" para ele, caso não haja ele desliga o rádio e vai para o modo sono economizando energia. Mas durante uma situação de emergência que ele participe muda o seu comportamento, permitindo a contenção em *slots* TDMA.

3.1.2 Protocolos Assíncronos

Nos protocolos assíncronos cada nó sensor escolhe sua programação ativa de forma autônoma. Sem pagar o preço para os horários de sincronização dos vizinhos, os protocolos

[1] Cluster é um conjunto de nós sensores.

[2] protocolos de acesso aleatório, cada estação é logicamente independente, ou seja, podem acessar o meio a qualquer instante de tempo. Sendo assim, duas ou mais estações podem transmitir simultaneamente, provocando a perda de informações devido a colisões.

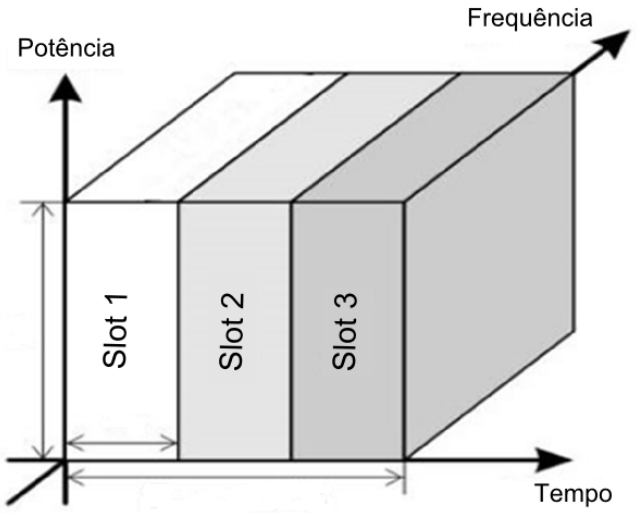


Figura 3.1: Exemplo de intervalo de tempo síncrono, adaptado de [19]

assíncronos podem conseguir um ciclo de trabalho muito baixo, mas tem que procurar maneiras eficientes para estabelecer a comunicação entre dois nós.

Nas RSSF, um nó sensor passa a maior parte do tempo no modo "sono" e "acorda" periodicamente para verificar se há pacotes para ele. Para indicar que existe uma transmissão de dados iminente, um remetente precede os dados com um preâmbulo, que é o tempo suficiente para ser detectado por todos os potenciais receptores.

Este projeto é adequado para aplicações de baixa carga de tráfego, onde as transmissões de dados ocasionais não impliquem muita sobrecarga em que a contenção do canal não é grave. Devido à transmissão do preâmbulo[3], o canal é ocupado e impede os nós vizinhos de transmissão, e a taxa de transferência possível é limitada. Para obter um maior rendimento, os protocolos assíncronos começaram a adotar que o receptor inicia uma sondagem para liberar mais espaço para a transmissão de dados.

Os protocolos assíncronos são baseados na disputa do meio de transmissão. Tais protocolos também tiveram que empregar novos mecanismos para lidar com a alternância de modos de RSSFs. Em diversos protocolos, por exemplo, o remetente precede o envio de um quadro com um preâmbulo ligeiramente mais longo que o período de "sono" do destinatário. Este último, ao "acordar", nota pelo preâmbulo a intenção do remetente de enviar um quadro e se mantém ativo.

Por outro lado, caso não escute preâmbulos, o potencial destinatário volta rapidamente a "dormir". Tais protocolos possuem a vantagem de minorar o problema de escuta ociosa e de não precisarem de sincronização, o que diminui a sobrecarga de comunicação. Contudo, eles sofrem tanto de transmissão em vão como de escuta desnecessária, já que todos os

[3] O preâmbulo é um tempo de espera e sincronismo que precede a transmissão de cada *frame*.

sensores devem esperar o preâmbulo para que possam verificar se o quadro é ou não destinado a eles.

Como exemplo cita-se o *Carrier Sense Multiple Access/Collision Avoidance* (CSMA/CA) [40] que é um método de acesso simples, utilizado pelas redes sem fio, que tenta evitar as colisões através do *backoff time*[4]. Em [39], Berkeley MAC (B-MAC), X-MAC, *Multimode Hybrid MAC* (MH-MAC), *Dual Preamble Sampling* MAC (DPS-MAC) e *Convergent MAC* (CMAC) usam CSMA em seus protocolos.

3.2 Camada de Controle de Acesso ao Meio (MAC)

Nesta Seção serão abordados os últimos avanços relativos a camada MAC, roteamento e economia de energia em RSSF. Na Seção 3.2.1 será discutida a proposta apresentada em [1] e o uso de TDMA e CSMA/CA. Nas Seções 3.3.1 e 3.3.2 serão apresentados os procedimentos desenvolvidos em [17] e [18] para fazer a clusterização e as formas de escolha dos nós.

Como foi visto na Seção 2.2, a camada MAC difere conforme o padrão IEEE 802.15.4 utilizado e permite que os dispositivos compartilhem a capacidade de transmissão de uma rede, minimizando as colisões de dados com a transmissão de dados aos nós vizinhos. Dessa forma, o esforço principal é poupar o consumo de energia devido a uma transmissão de pacote perdido, além de manter uma tabela dos endereços físicos dos dispositivos.

3.2.1 Protocolo *Funneling-MAC*

Redes de sensores apresentam um efeito afunilamento único, que é um evento de muitos para um, tráfego salto por salto é encontrado em redes de sensores e resulta em um aumento significativo na intensidade de tráfego, colisão, congestionamento, perda de pacotes e a energia se esgota com os eventos se aproximando do *sink*. Embora as técnicas de aplicação (por exemplo, agregação) e rede (por exemplo, controle de congestionamento) podem ajudar a combater este problema mas não podem aliviá-la totalmente. Em [1], foi adotada uma abordagem diferente, mas complementar para resolver este problema do que é encontrada na literatura, a implementação e avaliação de um *sink* localizado e orientado.

O afunilamento de eventos leva ao aumento da intensidade de tráfego e de atraso na rede com os eventos se aproximando ao *sink*, o que resulta em colisões significativas de pacotes, de congestionamento e consequentemente na perda desses pacotes; no melhor dos

[4] *backoff time* tempo aleatório de envio.

casos, isso leva a uma aplicação limitada medida no *sink*, e na pior hipótese haveria um congestionamento na rede [20].

Além disso, os sensores próximo ao *sink*, tipicamente dentro de um pequeno número de saltos, perdem um número desproporcional de pacotes (esta região do funil foi chamada de região de intensidade), tal como ilustrado na Figura 3.2 e consomem muito mais energia do que os sensores mais distantes do *sink*, portanto, encurtando o tempo de vida da rede. Reduzir o efeito de afunilamento representa um desafio importante em rede de sensores.

Em [1], é proposto um protocolo que é capaz de reduzir o efeito de afunilamento (Figura 3.2), devido a disputa que os nós fazem para chegar ao *sink*. O CSMA/CA é utilizado em todo o campo de sensor e apenas na parte chamada região de intensidade[5] é que o TDMA é usado para ter controle das entradas de pacotes e para diminuir perdas de pacotes (o que pode acontecer eventualmente devido a concorrência pelo canal a ser acessado).

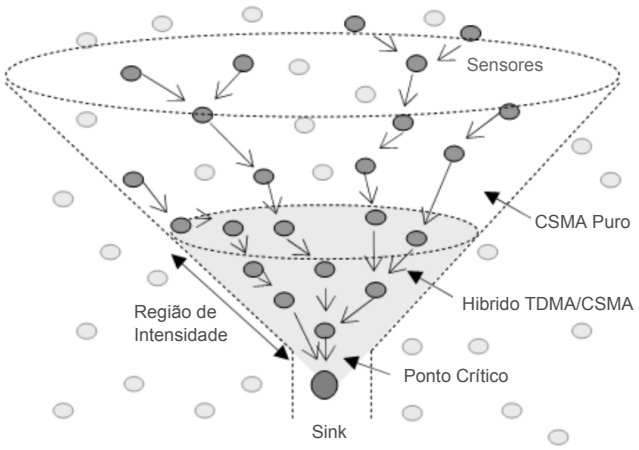


Figura 3.2: Efeito afunilamento em redes de sensores, adaptado de [1].

Quando um nó não tem uma rota, envia os dados do evento para o canal de transmissão. Esse processo contribui para o congestionamento da rede e degrada a produtividade ainda mais. Quanto mais nós de origem adquirirem rotas, o *funneling*-MAC ganha desempenho em termos de taxa de transferência.

O trabalho propõe que o protocolo MintRoute [4] escolha essas rotas. Ele consiste em estabelecer um caminho do nó de origem do evento para o *sink*, a partir da construção de uma árvore para encontrar o melhor nó para o encaminhamento de mensagens em direção ao *sink* e constrói uma tabela de roteamento. Para construir a tabela de roteamento, ele

[5]Região de intensidade é a parte em que os nós sensores ficam concentrados/disputando através do hibrido de TDMA/CSMA o canal para chegar ao *sink*

calcula a distância entre os nós vizinhos através da troca de mensagem de *beacon*[6]. O estabelecimento desse caminho é o fator mais crítico na rede.

O monitoramento do tráfego que chega no *sink* é de responsabilidade do mesmo. Através de uma base por caminhos agregados e calcula o cronograma TDMA com base no tráfego monitorado (inicialmente com base em apenas novos eventos CSMA e, posteriormente, incluindo o tráfego existente TDMA) para todos os caminhos, e distribui essa programação, transmitindo um pacote de programação com a mesma potência de transmissão utilizada pelos *beacons*.

O caminho de agregação é uma fusão de dois ou mais caminhos, ao entrar na região de intensidade. Por exemplo o *funneling*-MAC mantém informações associadas dos caminhos G-B-F-E-D e H-B-F-E-D, que forma como caminho principal B-F-E-D. Esse caminho agregado é tratado como uma entrada de caminho único,como exemplificado na Figura 3.3.

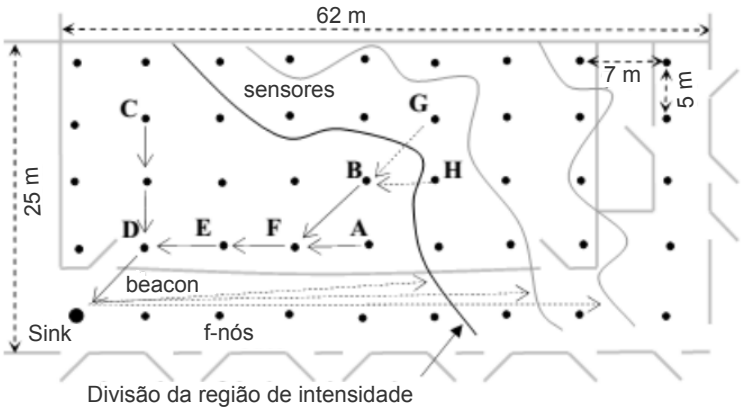


Figura 3.3: Caminho de agregação, adaptado de [1].

Na Figura 3.3 podemos ver a região de intensidade de forma mais ampliada, um espaço de $25 \times 62\ m^2$, onde cada sensor tem uma distância de 7m horizontalmente e 5m verticalmente. Na Figura 3.3, os sensores formam um caminho de agregação em que são excluídos os caminhos redundantes e se forma um caminho único onde são levadas as informações ao *sink*.

O *funneling*-MAC funciona bem porque o número de caminhos agregados que entram na região de intensidade é limitado pelo número de nós nessa região. Dessa forma, na proposta do algoritmo é dada principal atenção ao método de acesso ao meio e a dinâmica de encaminhamento dos quadros.

[6] *Beacon* é uma mensagem periódica utilizada para diversas finalidades como notificar aos nós vizinhos que a origem está ativa e também para descobrir os vizinhos e trocar alguma informação útil, dependendo do contexto

3.3 Clusterização

Em [23], o termo clusterização define a agrupação de dados, com base na similaridade entre eles, em grupos disjuntos chamados *clusters*. Isso significa que dados em um mesmo *cluster* são mais similares do que dados pertencentes a *clusters* diferentes. O ato de agrupar os dados representa uma ferramenta importante no aprendizado e compreensão a respeito dos mesmos. Pode ser definido também como um problema de aprendizado não-supervisionado, e é necessário distinguir aqui clusterização (classificação não-supervisionada) de análise discriminante (classificação supervisionada):

- em classificação supervisionada, são fornecidos padrões rotulados (pré-classificados) e o problema é rotular novos padrões, ainda não-rotulados.
- em classificação não-supervisionada, o problema é agrupar um conjunto de padrões não-rotulados em *clusters* que possuam algum significado, ou seja, de tal modo que os padrões apresentem alguma propriedade comum. Sendo assim, uma vez definidos os *clusters*, os padrões também estarão "rotulados", mas o rótulo é ditado pelos próprios padrões que compõem cada *cluster*.

Como não existem rótulos iniciais, o objetivo da clusterização é encontrar uma organização válida e conveniente dos dados, ao invés de separá-los em categorias. As aplicações de clusterização incluem caracterização de diferentes grupos de clientes baseado nos padrões de compra, categorização de documentos na *World Wide Web* (www), agrupamento de genes e proteínas que possuem funcionalidades similares, agrupamento de localizações geográficas propensas a terremotos através de dados sismológicos, entre outros.

3.3.1 *Low-Energy Adaptive Clustering Hierarchy* (LEACH)

Segundo [17], o LEACH é um protocolo de agrupamento adaptativo de auto-organização que utiliza a randomização para distribuir a carga de energia uniformemente entre os sensores da rede. Os nós se organizam em *clusters* locais, com um nó atuando como *cluster head*.

No algoritmo LEACH, cada sensor possui uma probabilidade de se tornar *cluster head* e faz um *broadcast* com a sua decisão para os demais nós. Os sensores que não forem atuar como *cluster head* se juntam ao *cluster* cujo *cluster head* pode ser atingido com o menor custo de energia, geralmente sendo o mais próximo.

A função de *cluster head* tem um custo mais alto do que a função de um sensor comum, já que além de realizar o monitoramento, o *cluster head* também é responsável por executar algoritmos de processamento de dados simples e também serve de intermediário entre

todos os sensores do *cluster* e o *sink*. Assim, a energia dos sensores que são eleitos *cluster heads* tende a se esgotar muito mais rapidamente, o que põe em risco a funcionalidade da rede, já que caso um *cluster head* sem energia, todo o *cluster* perde a capacidade de comunicação com o *sink*.

Para amenizar esse problema, o LEACH utiliza o conceito de *rodadas*. Uma *rodada* pode ser definida como um período de tempo que a rede permanece com uma determinada topologia de modo que, ao início de cada *rodada*, uma nova topologia com novos *cluster heads* é formada. Uma *rodada* é composta por duas fases distintas: a fase de *setup*, seguida pela fase de transmissão.

A primeira fase é responsável pela eleição dos *cluster heads* e formação dos *clusters*, ou seja, a topologia da rede. A fase de transmissão é onde a simulação da rede realmente é executada, ou seja, os sensores realizam o monitoramento, coletam os dados, repassam-nos para seu *cluster head* que envia os dados para o *sink*. No LEACH, cada *rodada* tem a duração de 20 segundos. Basicamente, a cada simulação do LEACH, diversas *rodadas* são executados, até que a rede seja considerada inativa. A Figura 3.4 apresenta as etapas do LEACH durante uma simulação.

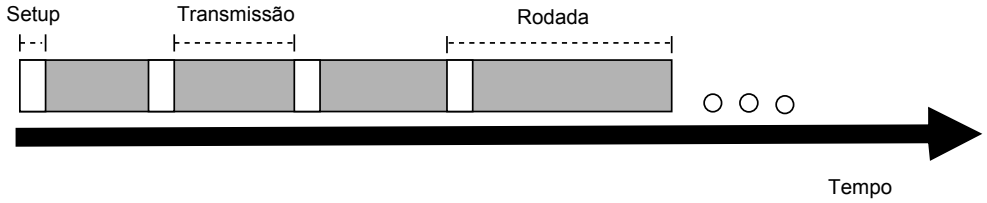


Figura 3.4: Gráfico da simulação do protocolo LEACH, adaptado de [17].

A cada fase de *setup*, o LEACH obriga os sensores a elegerem *cluster heads* diferentes, evitando que os mesmos sensores sejam eleitos repetidas vezes. Esse tipo de rotatividade visa buscar uma distribuição da carga entre todos os sensores de maneira uniforme. O número esperado de sensores que não foram eleitos *cluster heads* nas primeiras r rodadas pode ser estimado como $n - (p \times r)$, onde n é a quantidade de sensores da rede, e p é o número esperado de *cluster heads* eleitos em cada *rodada*. Assim, espera-se que em $\frac{n}{p}$ *rodadas*, todos os sensores da rede já tenham exercido a função de *cluster head*. Para formar os *clusters head*, cada sensor sorteia um número aleatório T entre 0 e 1, e o sensor i seria *cluster head* se:

$$T(i) = \begin{cases} \frac{p}{n - p \times (r mod \frac{n}{p})}, & \text{se } i \in G, \\ 0, & \text{caso contrário.} \end{cases}$$

onde n é a quantidade de sensores da rede, p é a quantidade de *clusters* que deseja-se formar, r a *rodada* atual e G o conjunto de sensores que não foram eleitos *cluster heads*

nos últimos $\frac{n}{p}$ rodadas. Com isso, passados $\frac{n}{p}$ rodadas, espera-se que todos os sensores da rede possuam aproximadamente a mesma quantidade de energia.

Para tal, o LEACH assume que a quantidade inicial de energia dos sensores seja igual e todos os sensores que transmitem quantidades iguais de dados durante a fase de transmissão. Assim, em redes onde o envio de informações seja orientado a eventos, a função de probabilidade deve levar em conta a energia residual dos sensores, priorizando a escolha daqueles que possuam maior quantidade de energia para serem eleitos como *cluster head*.

3.3.2 *Low-Energy Adaptive Clustering Hierarchy Centralized* (LEACH-C)

Em [18], o LEACH-C foi desenvolvido como uma extensão do LEACH para resolver os problemas de seu antecessor (LEACH). Formar os *clusters* de maneira distribuída é vantajoso por não necessitar de intervenção externa, porém o LEACH não garante uma quantidade fixa de *clusters* nem uma distribuição de forma igual dos *cluster heads* pela rede.

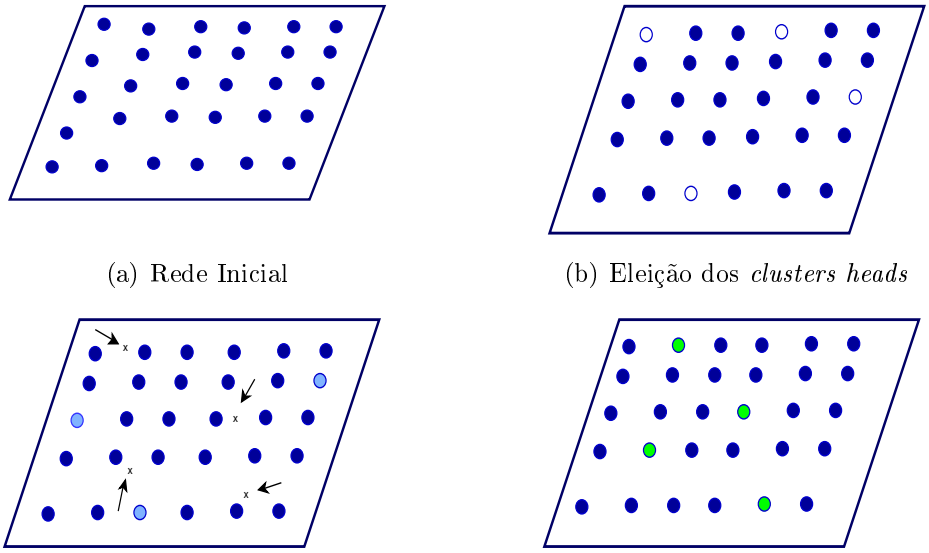
O LEACH-C contorna o problema de descentralização do LEACH executando um algoritmo centralizado no *sink* para eleger os *cluster heads*. Enquanto a fase de transmissão do LEACH-C é idêntica a do LEACH, na fase de *setup*, todos os sensores da rede enviam informações para o *sink* como sua energia residual e posição (em caso de redes móveis).

O *sink* então executa a meta-heurística *simulated annealing* [31] para eleger os *cluster heads* de maneira eficiente. O algoritmo leva em conta a quantidade de energia de cada sensor, formando uma lista de sensores elegíveis para serem *cluster heads*. Um sensor i só é considerado elegível se:

$$E(i) > \frac{\sum_{j=0}^{n} E(j)}{n} \tag{3.1}$$

onde E é a função que retorna a quantidade de energia de um dado sensor e n é a quantidade de sensores da rede. Isso garante que apenas os sensores que possuírem mais energia terão rodadas para atuarem como *cluster head*.

Uma lista *CH* (*Cluster Head*) é formada contendo p sensores a serem rodadas de forma aleatória conforme *EL* (Eleição) (Figura 3.5-b). Como pode ser visto nas linhas 5 e 6 do algoritmo 1, para cada sensor de *CH* é feita uma perturbação em suas coordenadas, somando algum valor aleatório, obtendo uma nova coordenada no terreno (Figura 3.5-c).



(a) Rede Inicial

(b) Eleição dos *clusters heads*

(c) Mapeamento de novas coordenadas a partir dos *clustersheads*

(d) Os sensores mais próximos das novas coordenadas são escolhidos como novos *clusters heads*

Figura 3.5: Obtenção de uma nova solução do LEACH-C

Como essa nova coordenada gerada pode ser um ponto onde não há nenhum sensor, a função MAP_CH faz o mapeamento da nova coordenada para o sensor mais próximo a ela, elegendo-o como novo *cluster head* (Figura 3.5-d). Se a nova solução obtida for melhor do que a anterior, então é feita a substituição das soluções.

Porém, se a nova solução não for melhor, ainda existe uma probabilidade (pr) de ser feita a substituição das soluções. Essa probabilidade é maior nas primeiras iterações, diminuindo a cada iteração executada do algoritmo. O critério de parada do algoritmo é por número de iterações executadas, retornando a lista CH ao final.

Com essa meta-heurística para eleger os *cluster heads* em LEACH-C mostrou ser mais inteligente e eficiente para prolongar o tempo de vida da rede. O Algoritmo 1 descreve passo a passo a meta-heurística do LEACH-C.

Algoritmo 1: ALGORITMO LEACH-C

```
1  início
2      EL ← constrói a lista de sensores;
3      CH ← p sensores aleatórios de EL;
4      enquanto i<iters faça
5          para para cada c ∈ CH faça
6              Coord[c] ← Coord[c] + rand();
7              NEW_CH ← MAP_CH(CH);
8          fim
9          se f(NEW_CH)<f(CH) então
10             CH ← NEW_CH;
11         senão
12             pr ← prob();
13             se rand()<pr então
14                 CH ← NEW_CH;
15             fim
16         fim
17     fim
18     retorna CH;
19 fim
```

3.3.3 Análise do *funneling*-MAC e LEACH-C

O protocolo *funneling*-MAC apresenta diversos problemas na presença de mobilidade dos nós da rede, de entre os quais se destaca o fato da transição do modo de funcionamento CSMA para o modo TDMA depender do período do *beacon*. Como apenas os nós que recebem o *beacon* funcionam em modo TDMA, um nó que funcione utilizando CSMA e que acabou de entrar na vizinhança, apenas efetua a transição quando receber o próximo *beacon*. Sendo o período de envio do *beacon* elevado (20s, o valor mínimo utilizado), a espera é considerável, este aspecto prejudica o desempenho do protocolo.

No LEACH-C por ser um protocolo de comunicação direta e os nós estarem distantes da estação base, será consumida uma grande quantidade de energia para cada nó. Este fato irá rapidamente esgotar a bateria dos nós e reduzir o tempo de vida da rede. Entretanto, como a única recepção acontecerá no *sink*, caso o *sink* esteja próximo ao nó, este pode ser um método aceitável de comunicação.

A junção dos dois protocolos permite obter vantagens ao se organizar a rede em *clusters* em duas regiões, de intensidade ou não intensidade. A identificação do nó como *cluster*

head permite que este tenha uma previsibilidade para o envio de informação que precisa ser encaminhada ao *sink*, sem depender então de um envio de *beacon*.

Resumo do capítulo

Neste Capítulo foram descritos os desafios de pesquisa mais importantes nestas redes, baseados nos trabalhos desenvolvidos sobre redes de sensores sem fio nos últimos anos. Entre estes trabalhos se destacam, os problemas relativos à economia de energia, roteamento e protocolos da camada MAC.

Foi mencionado sobre a topologia em uma RSSF, que ao contrário do que acontece nas redes de computadores tradicionais, pode ser alterada dinamicamente, devido a falha de nós, inserção de novos nós e alterações no comportamento da rede. Finalmente, sobre a taxonomia da topologia de rede foi descrito um resumo das principais características de uma RSSF segundo a sua configuração.

Foram descritos alguns trabalhos do estado da arte na área de RSSF. Em [1] o *funneling*-MAC é baseado em CSMA/CA que é implementado em toda a rede, com um algoritmo de TDMA localizado na região de intensidade. Neste sentido, o *funneling*-MAC representa uma abordagem híbrida, mas não tem os problemas de escalabilidade associadas com a *funneling* em toda a rede TDMA.

No *funneling*-MAC o TDMA faz o agendamento de eventos do sensor na região de intensidade que passam os dados para o nó *sink*. Com isso os nós sensores são "orientados ao *sink*", e é "localizado" porque TDMA só funciona localmente na região de intensidade perto do *sink* e não em todo o campo do sensor.

Em [18] foi proposto um protocolo de agrupamento adaptativo de auto-organização que utiliza a randomização para distribuir a carga de energia uniformemente entre os sensores da rede. Os nós se organizam em *clusters* locais, com um nó atuando como *cluster head*. O LEACH-C forma os *clusters* de maneira distribuída é vantajoso por não necessitar de intervenção externa, porém o seu antecessor LEACH não garante uma quantidade fixa de *clusters* nem uma distribuição de forma igual dos *cluster heads* pela rede. O LEACH-C contorna essa "fraqueza" do LEACH.

Capítulo 4

Proposta de Trabalho e Análise Experimental

Neste Capítulo será descrita a proposta deste trabalho, que é baseada na integração do *funneling*-MAC [1] e do LEACH-C [18], integradas através do roteamento do caminho mais curto baseada no algoritmo de *Dijkstra*. A Seção 4.1 apresenta a proposta de trabalho chamada de *Reduction Energy Algorithm in WSN* (REA-WSN). Na Seção 4.2 são apresentados os resultados da proposta e é feita uma análise crítica e comparativa dos resultados.

4.1 Proposta

Na proposta de trabalho, a rede é dividida em duas regiões, a região de intensidade (RI) e região de não intensidade (RNI). Na RNI os nós se comunicam utilizando CSMA e na RI utilizam TDMA. Dessa forma todo nó que estiver na RNI é um nó híbrido, pois todos tem chances de se tornar *cluster head*, e passará a se comunicar utilizando o TDMA com os outros *cluster heads* vizinhos, conforme figura 4.1. Considerando-se os trabalhos de [1] e [18], alguns pontos que podem ser melhorados são os seguintes:

a) uma estratégia de roteamento *multihop* entre os *clusters heads*, o que irá melhorar a economia de energia, uma vez que a quantidade de energia usada para enviar pacotes diretamente para o *sink* é maior do que o necessário para enviar pacotes para um nó mais próximo;

b) os *clusters* perto do *sink*, ou seja, na região de intensidade, podem reduzir o efeito de afunilamento usando TDMA e isso vai ser mais eficiente para *slots* de agendamento quando diferentes *clusters heads* são selecionados.

Tal como proposto no LEACH-C [18], ou seja, o *sink* centraliza as informações da rede: posições dos nós e os *cluster heads*, sincronização e agendamento. Uma adaptação para LEACH-C da proposta de trabalho é que os*cluster heads* são selecionados quando têm energia residual miníma para completar, pelo menos, um ciclo de encaminhamento de dados. A comunicação entre os *cluster heads* usa TDMA e o algoritmo REA-WSN calcula o caminho mais curto para o *sink* baseado no algoritmo de *Dijkstra* em que a métrica é a energia residual.

Algoritmo *Dijkstra* funciona através de pesquisa na rede de modo uniforme e o processo resultante pode visitar um grande número de nós. O algoritmo calcula o caminho mais curto entre dois pontos de uma rede usando um gráfico composto por nós e arestas. Ele atribui a cada nó um valor de custo, definindo zero para o nó fonte e infinito para todos os outros nós. O algoritmo irá procurar todos os caminhos únicos na rede para encontrar um caminho viável.

Para a proposta, uma vez que a rede é dividida em *clusters*, cada *cluster head* irá atuar como um *gateway* e irá encaminhar pacotes para o *sink* usando uma rota *multihop* calculado pelo REA-WSN, em vez de enviar os dados diretamente para o *sink*, o que pode resultar na economia de energia e uma escalabilidade superior. Uma vez que o processo de clusterização está centralizada no *sink*, o *sink* vai sincronizar o procedimento de atualização de cluster que terá impacto sobre o processo de roteamento.

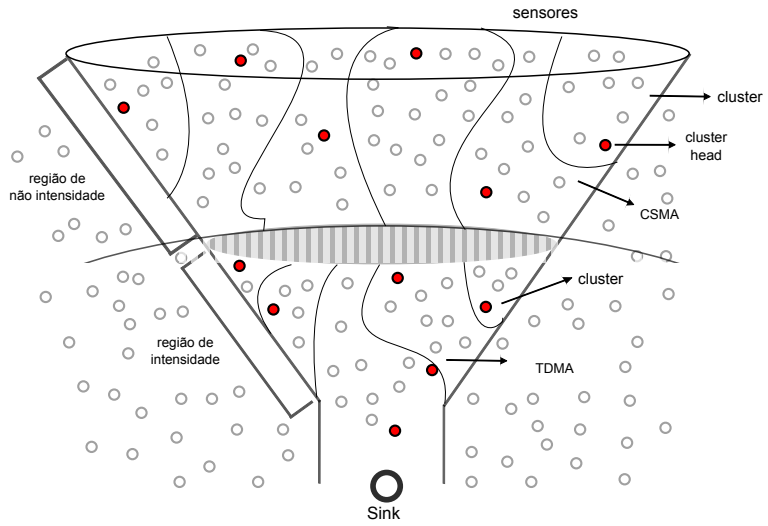


Figura 4.1: Proposta de trabalho REA-WSN.

Este trabalho assume um modelo de uma RSSF densamente formada, onde os nós sensores alcançam com um salto o CH. Este trabalho assume uma topologia de dinâmica controlada, pois com o tempo, alguns nós da rede podem deixar de funcionar devido a danos, ou por ter suas fontes de energias esgotadas. Otimizar o consumo de energia média da rede é o principal objetivo desta proposta.

Primeiramente neste trabalho foi feita a clusterização de toda a rede através do LEACH-C [18], e em seguida a rede é dividida em regiões chamadas região de não intensidade (RNI) e região de intensidade (RI). Essas regiões visam melhorar a economia de energia na rede (conforme Figura 4.1). Depois disso os nós sensores saberão se irão trabalhar no modo CSMA (aqueles que estão fora da RI) ou TDMA (aqueles que estão dentro da RI). Na RNI o acesso ao meio é realizado com o CSMA/CA e apenas na RI é utilizado o TDMA. Em toda a rede os nós sensores são agrupados em seus *clusters* e em cada *cluster* um nó é eleito como *cluster head*. Os nós enviam seus dados diretamente para o *cluster head* do seu grupo através do LEACH-C.

Com essa divisão, o roteamento entre os CHs é feito pelo REA-WSN. Os CHs mandam uma mensagem em *broadcast* aos outros CHs vizinhos e verificam qual a sua energia residual. É escolhido o *cluster head* vizinho com maior energia, até se comunicar com um *cluster head* de dentro da RI, ao fazer essa comunicação o *cluster head* de dentro da RI e esse enviará os dados ao *sink*.

Os algoritmos baseados em *Dijkstra* (por exemplo OLSR, OSPF, entre outros) usam a estabilidade de rede para definir as rotas, entretanto em uma RSSF existe a movimentação dos nós (caso sejam móveis) ou a "morte" precoce de um nó. No algoritmo proposto, o REA-WSN, a rede é homogênea e hierárquica o que torna viável, pois são criados *clusters* e são escolhidos para CHs o nó de maior energia residual o que garante que durante o período de roteamento o nó escolhido não irá "morrer". Logo, este tipo de roteamento proposto é viável em uma RSSF com essas características.

Uma restrição do algoritmo é a dinâmica de troca de CHs que só vai acontecer depois do encaminhamento dos dados, pois não é possível mudar os CHs no meio deste processo. Quando os nós que estão como CH trabalham em TDMA é feita a sincronização e o encaminhamento dos dados, somente depois pode haver a alternância dos CHs.

Como contribuição neste trabalho, foram desenvolvidas algumas variações dos trabalhos [1] e [18], a saber:

1) Na RNI os nós trabalham com CSMA e de forma clusterizada utilizando o LEACH-C.

2) Na RI os nós trabalham com TDMA e de forma clusterizada utilizando o LEACH-C.

3) Otimização do roteamento através do REA-WSN. Os CHs detectam o caminho que gastaria menor energia entre os CHs na RNI até um CH na RI. Utilizando o algoritmo proposto, é feito um pré-calculo de quanto de energia seria necessário pra enviar de CH para outro CH. Então usa-se esses valores como peso das arestas, escolhe o menor, ou seja o caminho com menos gasto de energia.

O REA-WSN é descrito a seguir:

Na linha 2 está a função dos parâmetros (ver Anexo 1 Algoritmo 4) de configuração da rede, o que diz respeito a quantidade de nós, tamanho da área, energia inicial etc. Na linha 3 é onde está inicializando a rede, onde os nós recebem a energia, tamanho da área e as coordenadas de onde vai ficar no campo.

Nas linhas de 4 a 6 delimita a RI em relação ao *sink* e calcula a distância nó até o *sink*, isso vai fazer com que o *sink* saiba se o nó está dentro da RI ou na RNI.

Nas linhas 7 a 16 é um vetor que guarda quantos CHs tiveram por rodada, onde acontece a configuração da rede, onde se verifica se há algum nó morto (ver Anexo 1 Algoritmo 5), onde escolhe-se os *clusters* e *cluster heads*, onde faz-se o roteamento através de *Dijkstra*, calcula a energia gasta. As funções parâmetros, verifica nó morto, escolha de CH, cria caminhos, calcula energia e associação de nós, pode ser vista com detalhes no anexo 1.

Algoritmo 2: *Reduction Energy in WSN* (REA-WSN)

```
1  início
2      Parametros da Rede;
3      Inicialização dos nós;
4      para para cada nó ∈ n faça
5          S(i).sinkDist=√((sink.x − S(i).xd)² + (sink.y − S(i).yd)²)
6      fim
7      para para cada rodada ∈ rmax faça
8          disp(['r = ',num2str(r)]);
9          se  mod (r, round(1/p))==0 então
10             para para cada nó ∈ n faça
11                 S(i).G=0;
12             fim
13             C = [];
14         fim
15         Verifica Nó Morto;
16         Escolha de CHs; Cria Caminhos; Calcula Energia; Associação de Nós;
           Sincronização;
17     fim
18 fim
```

4.2 Análise Experimental

4.2.1 Definição dos cenários

A validação do algoritmo proposto foi feita em um cenário de simulação com as seguintes características:

a) Uma RSSF homogênea,ou seja, os sensores terão as mesmas configurações, cada sensor tem uma energia inicial de 0.5J (*Joule*). Com o número de nós estáticos dispostos aleatoriamente de 100 a 250 numa área de interesse, essa variação entre $100 \times 100\ m^2$ a $300 \times 300\ m^2$ depende do tamanho da área de cobertura da rede;

b) Para definir a RI e RNI foi utilizado um parâmetro de afunilamento de rede, definindo que 30% da rede é dedicada a RI, esse parâmetro foi baseado no trabalho de [1], onde através de um *beacon* é definido a RI e RNI.

c) Um conjunto de métricas para avaliação da eficiência energética na rede que serão descritas na Seção 4.3.4

d) Foi definido um modelo analítico com base em [1] e [18], apresentados na Seção 3.2.1 e Seção 3.3.2 para melhorar a eficiência energética em uma RSSF. Em cada modelo foi assumido que o tamanho do pacotes de dados é de 500 *bytes*.

4.3 Resultados

Esta Seção apresenta as simulações do algoritmo REA-WSN, e os resultados comparados com os dos protocolos [1] e [18].

4.3.1 Ambiente Utilizado

O ambiente utilizado foi o sistema operacional *windows* 8.1, com processador core i5 da intel, 500 GB de HD e 6GB de memória RAM, o simulador utilizado foi o MATLAB2012B, o mesmo no qual foi o desenvolvido os outros algoritmos para fazer as simulações.

4.3.2 Parâmetros

Os parâmetros utilizados neste trabalho tem como referência os utilizados em [18], onde cada nó na rede possui 0.5 *Joule* de energia e o tamanho dos pacotes enviados é de 500 *bytes*. Nos parâmetros tem-se as seguintes configurações:

- Envio: ato de enviar os quadros de um nó a outro.
- Recepção: ato de receber os quadros de um nó a outro.
- Agregação: ato de agregar os dados redundantes dentro do *cluster*
- Sincronização: ato de sincronizar os quadros de dados os nós híbridos para envio de quadros em modo TDMA.
- Processamento: tempo de decisão dos CH em relação a rota.

Os parâmetros utilizados são apresentados na Tabela 4.1:

Tabela 4.1: Configuração do cenário

Parâmetros gerais	
Topologia de rede	plana
Sensores	estáticos e homogêneos
Número de sensores	$100 \sim 250$
Área	$100 \times 100m^2 \sim 300 \times 300m^2$
Posição do *sink*	centralizado
Energia inicial	0.5J
Parâmetros de consumo para todos os tipos de nós	
Envio	$50\times\ 10^{-9}$ nJ
Recepção	$50\times\ 10^{-9}$ nJ
Parâmetros de consumo para os nós quando CHs	
Envio	$50\times\ 10^{-9}$ nJ
Recepção	$50\times\ 10^{-9}$ nJ
Agregação	$5\times\ 10^{-9}$ nJ
Sincronização	$50\times\ 10^{-9}$ nJ
Processamento	variável

Os parâmetros gerais baseiam-se na proposta do LEACH-C [18], onde rede é plana, com nós estáticos e homogêneos o número de nós varia de 100 a 250 isso depende do tamanho da área, pois em uma área muito grande se tiver poucos nós a rede fica sobrecarregada e morrerá mais rápido porque terá que ter uma potência maior de comunicação entre os nós, a área varia de $100 \times 100m^2$ a $300 \times 300m^2$ para análises experimentais, a posição do *sink* é centralizada e a energia inicial é de 0.5 *Joules*. Os parâmetros de consumo para todos os tipos de nós foi baseado na proposta do LEACH-C [18], onde nesse trabalho eles usam os parâmetros de envio e transmissão de $50\times\ 10^{-9}$ nJ para a comunicação nos nós.

Os parâmetros de consumo para os nós quando CH também baseiam-se na proposta de LEACH-C [18], os parâmetros de envio e transmissão de $50\times\ 10^{-9}$ nJ para a comunicação, o gasto de energia da agregação é de $5\times\ 10^{-9}$ nJ, não é foco deste trabalho, mas como todo CH faz a agregação de dados para poder enviar os dados, é calculado esse gasto. A sincronização é de $50\times\ 10^{-9}$ nJ, onde é feita a comunicação dos CHs para troca de informações. O cálculo de gasto de energia para o processo de roteamento se baseia na complexidade do algoritmo de caminho mais curto em cada CH definida por $O(n^2 + n + log_n)$, a ser calculada em cada CH.

4.3.3 Cenário de simulação

O cenário foi baseado em [18] com algumas variações para termos de comparação. A Tabela 4.2 mostra os cenários utilizados para as simulações:

Tabela 4.2: Cenários utilizados nas simulações.

Cenário	Quantidade de nós	Área (m x m)	Número de rodadas
1	100	100 x 100	200
2	150	100 x 100	250
3	200	200 x 200	300
4	200	300 x 300	500
5	250	300 x 300	700

O intervalo de confiança foi definido como $\mu - k \leq x \leq \mu + k$, tal que k é o valor que resolve a seguinte equação:

$$\int_{\mu-k}^{\mu+k} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = \frac{95}{100} \tag{4.1}$$

Foram feitas 10 simulações de cada algoritmo e em cada cenário usando a expressão acima para o cálculo de intervalo de confiança em cada simulação, onde foi feito o cálculo da variância junto com os valores médios de cada métrica para se obter o valor de 95% de intervalo de confiança.

4.3.4 Métricas para Avaliação

Nas métricas tem-se as seguintes configurações:

- Na energia relativa.
  - $E_R(t)$: energia relativa em um instante de tempo t.
  - $\frac{E_T(t)}{E_T(0)}$: razão da energia total inicial pela energia total final em um instante de tempo t
- Na energia total.
  - $E_T(t)$: energia total em um instante de tempo t.
  - $\sum_{i=1}^{n}$: somatório de todos os nós.
  - $E.no_i(t)$: energia de cada nó em um instante t.

Para a avaliação dos resultados, são consideradas as seguintes métricas:

- Energia relativa: É a razão da energia total no instante t pela energia do instante inicial. Dado pela equação:

$$E_R(t) = \frac{E_T(t)}{E_T(0)} \tag{4.2}$$

- Energia total: É a energia de todos os nós em um dado intervalo de tempo. A seguinte equação é o cálculo da energia total:

$$E_T(t) = \sum_{i=1}^{n} E.no_i(t) \tag{4.3}$$

- Números de nós inativos: quantidade de nós que tem energia residual igual a 0 (zero) no instante t.

4.3.5 Resultados

Esta Seção analisa os resultados das simulações e compara os resultados obtidos pela proposta apresentada em [1] e [18]. Os parâmetros e métricas utilizados nas simulações são os apresentados em 4.3.2 e 4.3.4. As Seções seguintes mostram os resultados para cada métrica utilizada. Tem-se como cenário padrão, o cenário 1 da Tabela 4.2, que foi o mesmo cenário utilizado para mostrar os resultados do protocolo LEACH-C em [18].

Cenário 1

No cenário é composto por 100 nós sensores distribuídos aleatoriamente em uma área de 100×100 com 200 rodadas de simulação.

Energia relativa

No gráfico da Figura 4.2 pode-se observar que o algoritmo LEACH-C em relação ao REA-WSN teve um gasto de energia, em média, 2% superior. O algoritmo *funneling-MAC* teve um gasto de energia 44% superior ao REA-WSN. Esses dados mostram que o algoritmo de REA-WSN foi superior nesta métrica.

Energia total

A Figura 4.3, pode-se observar que o algoritmo LEACH-C teve um gasto de energia média de 8.6902e-04 J, enquanto REA-WSN teve 8.5064e-04 J, o que faz com que LEACH-C tenha gasto 2% a mais de energia que REA-WSN, em relação ao algoritmo *funneling-MAC* que tem um gasto de energia média de 0.0012 J, o que faz com que LEACH-C tenha gasto 44% a mais de energia que REA-WSN. Esses dados mostram que o algoritmo de REA-WSN foi superior nesta métrica.

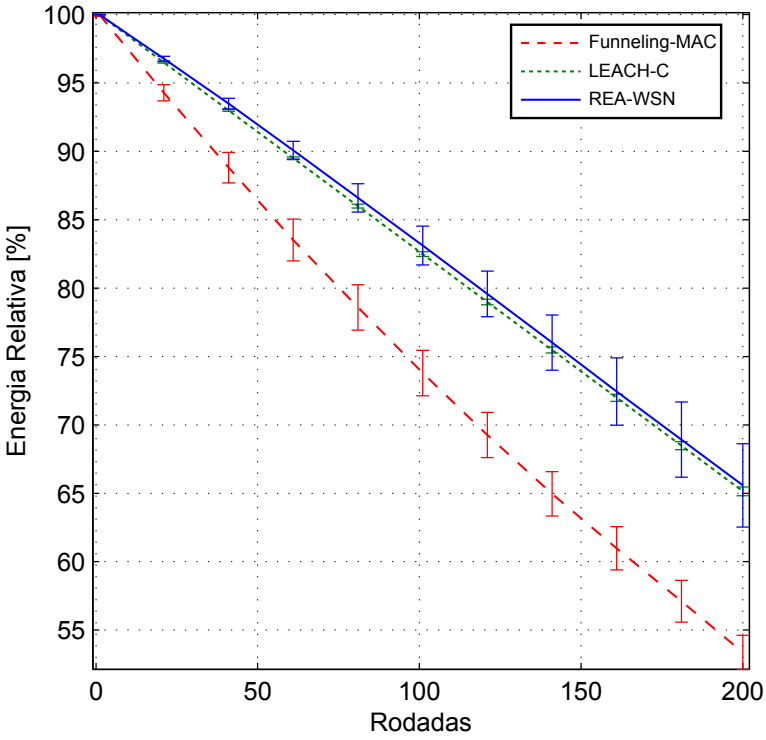


Figura 4.2: Energia Relativa.

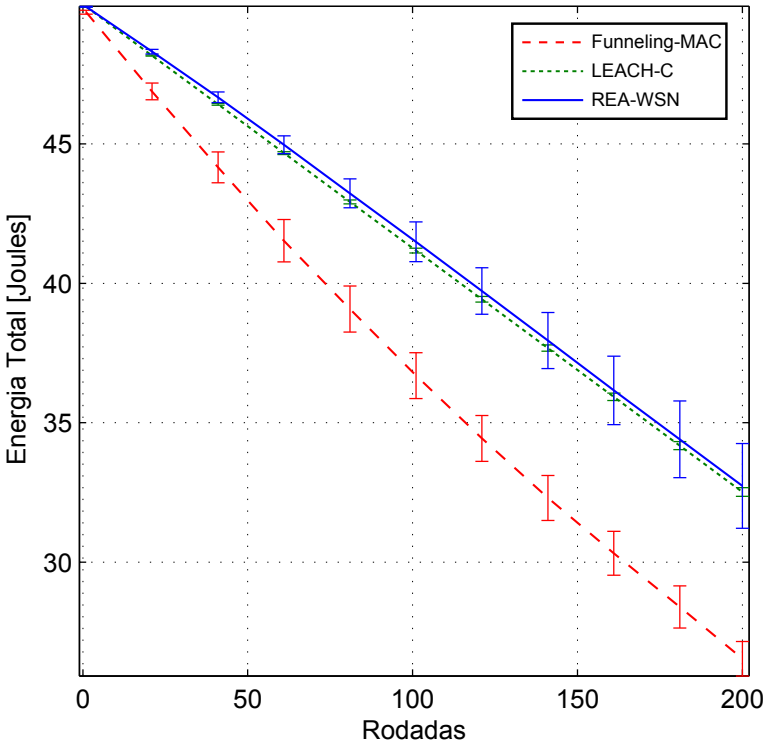


Figura 4.3: Energia Total.

Nós inativos

Na Figura 4.4 pode-se observar que os algoritmos LEACH-C e o REA-WSN ao final das 200 rodadas não morreu nenhum nó. Enquanto que no algoritmo de *funneling*-MAC

morreram 20 nós, o que corresponde a 20% do total de nós. Indicando que em relação ao número de nós inativos LEACH-C e REA-WSN foram superiores nesta métrica, mas em relação ao consumo de energia REA-WSN leva vantagem por consumir menos energia média em comparação aos demais algoritmos.

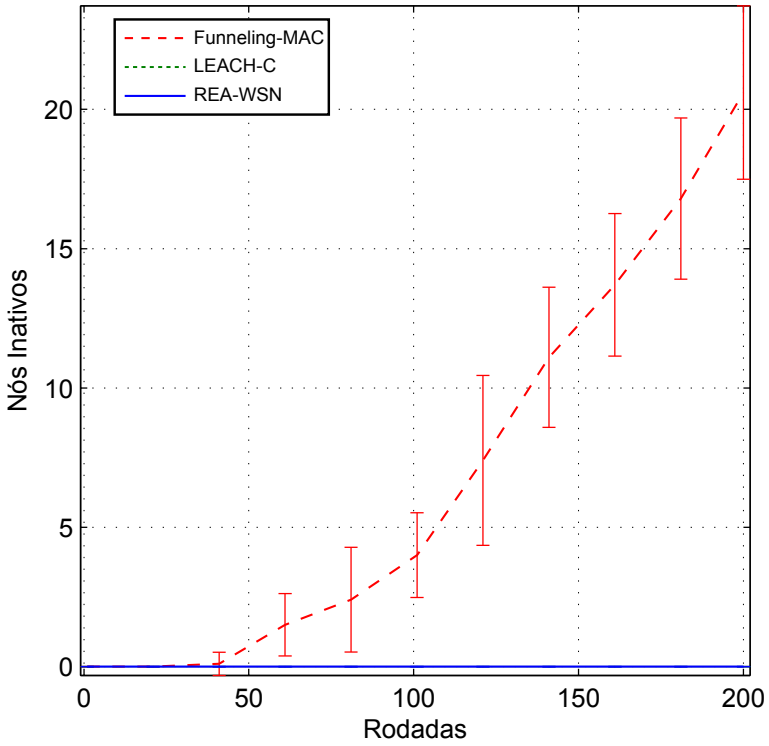


Figura 4.4: Nós inativos.

Cenário 2

O cenário é composto por 150 nós sensores distribuídos aleatoriamente em uma área de 100×100 com 250 rodadas de simulação.

Energia relativa

No gráfico da Figura 4.5 pode-se observar que o algoritmo LEACH-C em relação ao REA-WSN teve um gasto de energia, em média, 4% superior. O algoritmo *funneling-MAC* teve um gasto de energia 55% superior ao REA-WSN. Esses dados mostram que o algoritmo de REA-WSN foi superior nesta métrica.

Energia total

A Figura 4.6 apresenta o algoritmo LEACH-C teve um gasto de energia média de 8.6663e-04 J, enquanto REA-WSN teve 8.3146e-04 J o que faz com que LEACH-C tenha

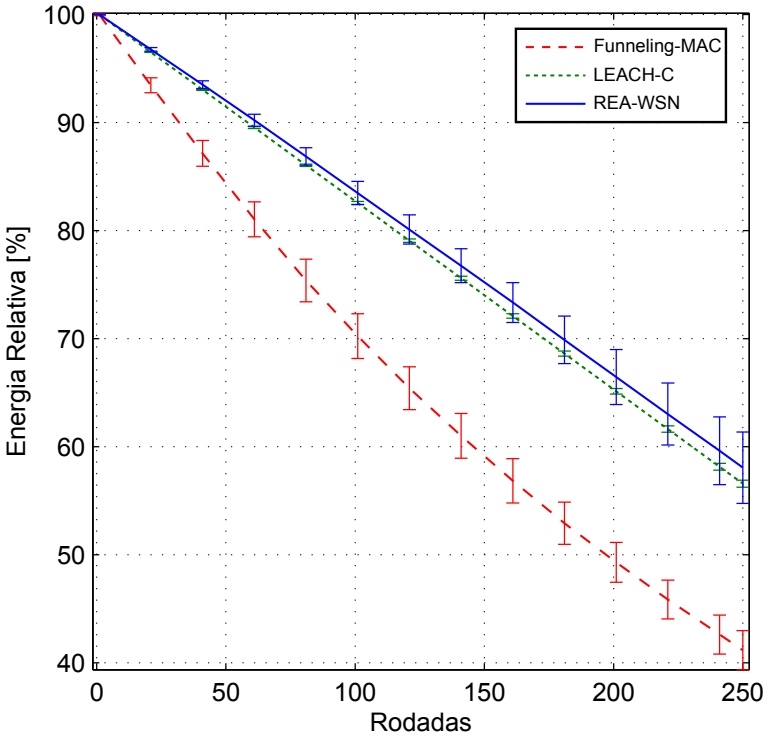


Figura 4.5: Energia Relativa.

gasto 4% a mais de energia que REA-WSN, em relação ao algoritmo *funneling-MAC* que tem um gasto de energia média de 0.0012 J o que faz com que LEACH-C tenha gasto 55% a mais de energia que REA-WSN. Esses dados mostram que o algoritmo de REA-WSN foi superior neste métrica.

Nós inativos

Na Figura 4.7 pode-se observar que os algoritmos LEACH-C e o REA-WSN ao final das 250 rodadas não morreu nenhum nó. Enquanto que no algoritmo de *funneling*-MAC morreram 49 nós, o que corresponde a 33% do total de nós. Indicando que em relação ao número de nós inativos LEACH-C e REA-WSN foram superiores nesta métrica, mas em relação ao consumo de energia REA-WSN leva vantagem por consumir menos energia média em comparação aos demais algoritmos.

Cenário 3

O cenário é composto por 200 nós sensores distribuídos aleatoriamente em uma área de 200×200 com 300 rodadas de simulação.

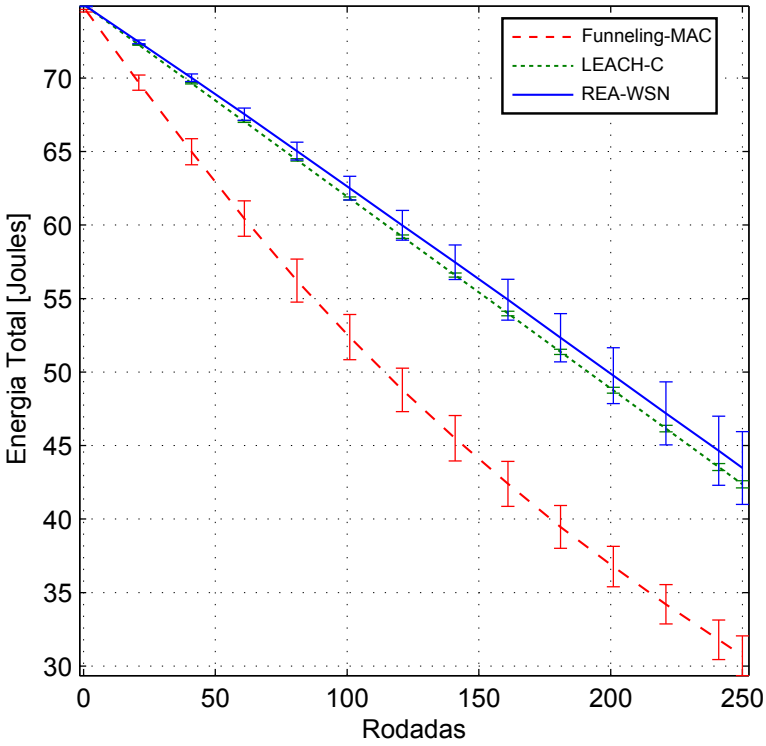


Figura 4.6: Energia Total.

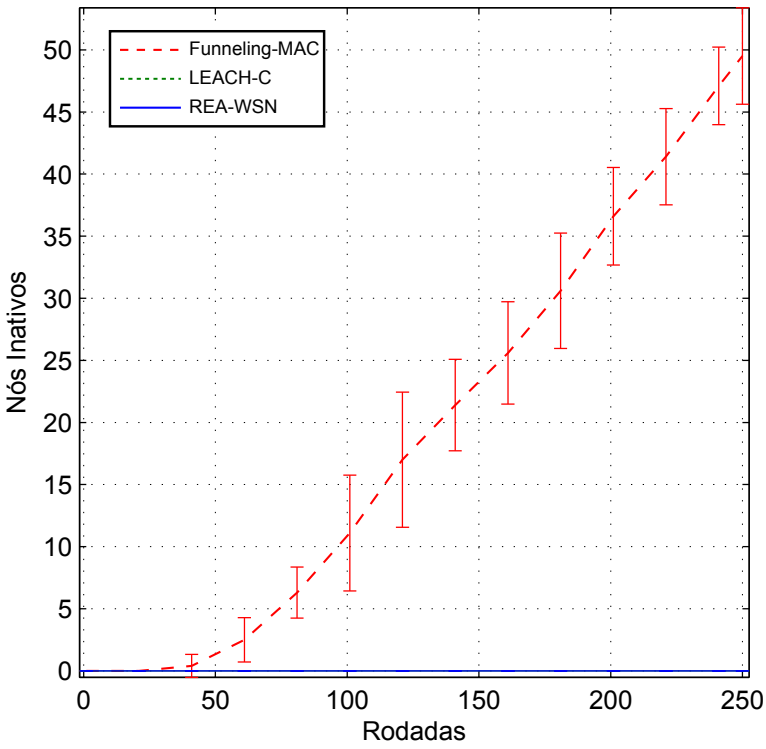


Figura 4.7: Nós inativos.

Energia relativa

Na simulação, a Figura 4.8 pode-se observar que o algoritmo LEACH-C em relação ao REA-WSN teve um gasto de energia, em média, 20% superior. O algoritmo *funneling-*

MAC teve um gasto de energia 41% superior ao REA-WSN. Esses dados mostram que o algoritmo de REA-WSN foi superior nesta métrica.

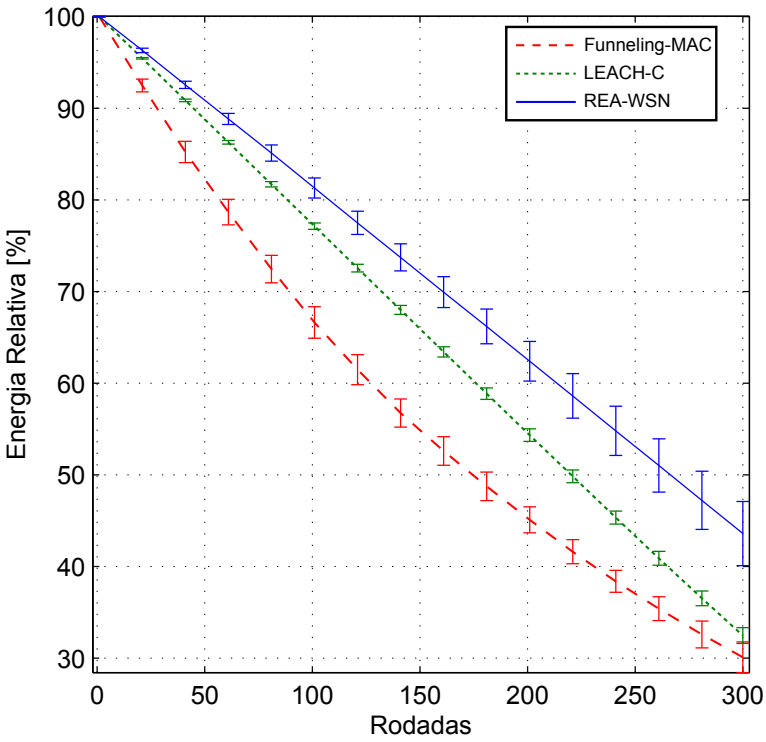


Figura 4.8: Energia Relativa.

Energia total

A Figura 4.9 apresenta o algoritmo LEACH-C teve um gasto de energia média de 0.0011 J enquanto REA-WSN teve 9.3601e-04 J, o que faz com que LEACH-C tenha gasto 20% a mais de energia que REA-WSN, em relação ao algoritmo *funneling-MAC* que tem um gasto de energia média de 0.0013 J, o que faz com que LEACH-C tenha gasto 41% a mais de energia que REA-WSN. Esses dados mostram que o algoritmo de REA-WSN foi superior neste métrica.

Nós inativos

Na Figura 4.10 pode-se observar que no algoritmo REA-WSN ao final das 300 rodadas não morreu nenhum nó. Enquanto que no algoritmo de *funneling*-MAC morreram em média 85 nós, o que corresponde a 45% do total de nós e no algoritmo LEACH-C morreram em média 10 nós, o que corresponde a 5% do total de nós. Isso indica que em relação ao número de nós inativos REA-WSN foi superior nesta métrica em relação aos demais algoritmos.

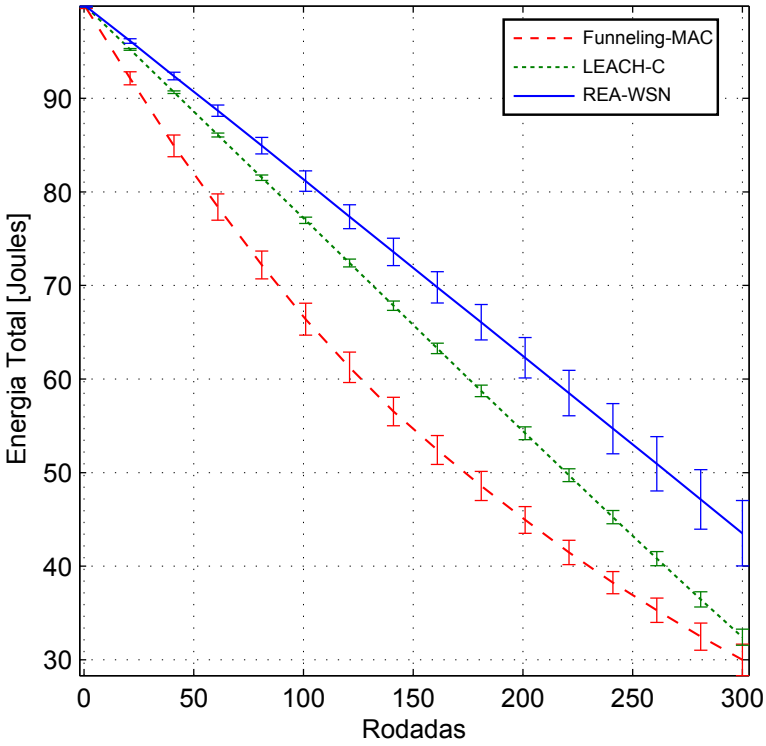


Figura 4.9: Energia Total.

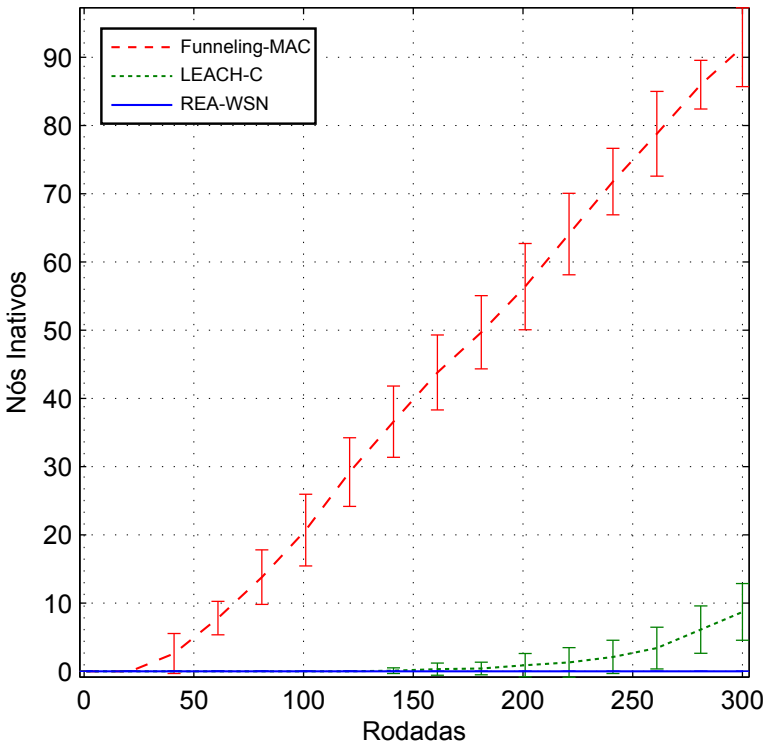


Figura 4.10: Nós inativos.

Cenário 4

No cenário é composto por 200 nós sensores distribuídos aleatoriamente em uma área de 300×300 com 500 rodadas de simulação.

Energia relativa

Na Figura 4.11 pode-se observar que o algoritmo LEACH-C em relação ao REA-WSN teve um gasto de energia, em média, 14% superior. O algoritmo *funneling-MAC* teve um gasto de energia 3% superior ao REA-WSN, mas ao final das 500 rodadas o algoritmo REA-WSN mostra-se inferior aos demais, isso pode ser justificado pelo consumo médio de energia nos nós, apesar que REA-WSN consome menos energia, mas os nós tem mais tarefas de decisão, sincronização, entre outros mostrando que isso pode acarretar no final das rodadas um gasto mais elevado na energia. Esses dados mostram que em média o algoritmo de REA-WSN foi superior nesta métrica.

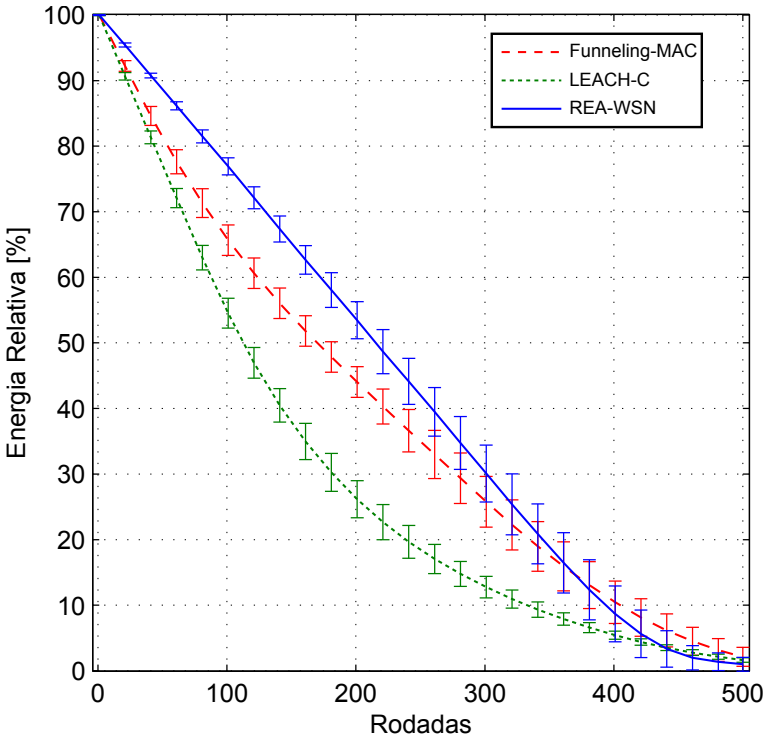


Figura 4.11: Energia Relativa.

Energia total

A Figura 4.12 apresenta o algoritmo LEACH-C teve um gasto de energia média de 0.0013 J, enquanto REA-WSN teve 0.0011 J, o que faz com que LEACH-C tenha gasto 14% a mais de energia que REA-WSN, em relação ao algoritmo *funneling-MAC* que tem um gasto de energia média de 0.0012 J, o que faz com que LEACH-C tenha gasto 3% a mais de energia que REA-WSN. Esses dados mostram que o algoritmo de REA-WSN foi superior neste métrica, mas ao final das 500 rodadas o algoritmo REA-WSN mostra-se inferior aos demais, isso pode ser justificado pelo consumo médio de energia nos nós,

apesar que REA-WSN consome menos energia, mas os nós tem mais tarefas de decisão, sincronização, entre outros mostrando que isso pode acarretar no final das rodadas um gasto mais elevado na energia.

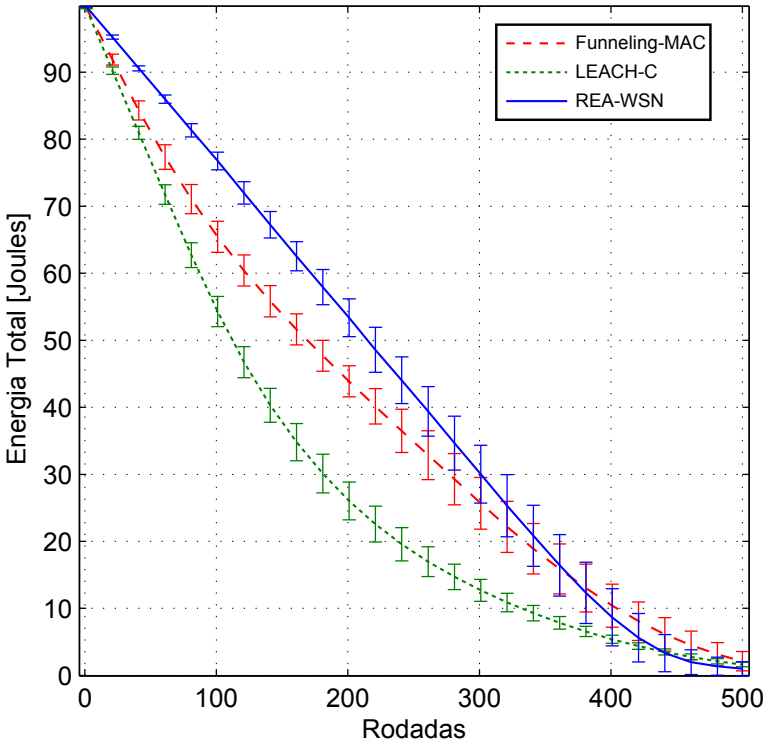


Figura 4.12: Energia Total.

Nós inativos

Na Figura 4.13, pode-se observar que no algoritmo REA-WSN ao final das 500 rodadas morreram em média 165 nós, o que corresponde em torno de 95% do total de nós, assim como no algoritmo LEACH-C. Enquanto que no algoritmo de *funneling*-MAC morreram em média 190 nós, o que corresponde em torno de 90% do total de nós. Isso indica que em relação ao número de nós inativos REA-WSN houve empate nesta métrica com o LEACH-C.

Cenário 5

No cenário é composto por 250 nós sensores distribuídos aleatoriamente em uma área de 300×300 com 700 rodadas de simulação.

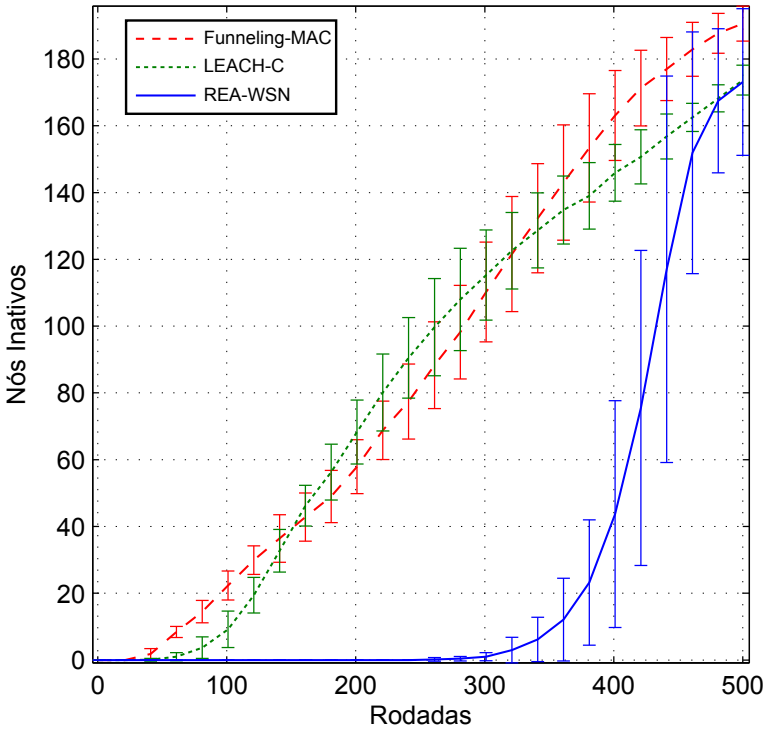


Figura 4.13: Nós inativos.

Energia relativa

Na simulação, a Figura 4.14 apresenta o algoritmo LEACH-C em relação ao REA-WSN teve um gasto de energia, em média, 7% superior. O algoritmo *funneling-MAC* teve um gasto de energia 3% superior ao REA-WSN, mas em torno de 600 rodadas os algoritmos mostram-se semelhantes chegando ao esgotamento da energia.

Energia total

A Figura 4.15 apresenta o algoritmo REA-WSN teve um gasto de energia média de 9.1961e-04 J, enquanto LEACH-C teve 9.8730e-04 J, o que faz com que LEACH-C tenha gasto 7% a mais de energia que REA-WSN, em relação ao algoritmo *funneling-MAC* que tem um gasto de energia média de 9.5280e-04 J, o que faz com que LEACH-C tenha gasto 3% a mais de energia que REA-WSN. Esses dados mostram que o algoritmo de REA-WSN foi superior neste métrica, e que ao final das 700 rodadas os algoritmos mostram-se iguais, pois em torno da rodada 600 a energia é próxima ao seu esgotamento.

Nós inativos

Na Figura 4.16, pode-se observar que no algoritmo REA-WSN ao final das 700 rodadas morreram em média 240 nós, o que corresponde em torno de 95% do total de nós assim

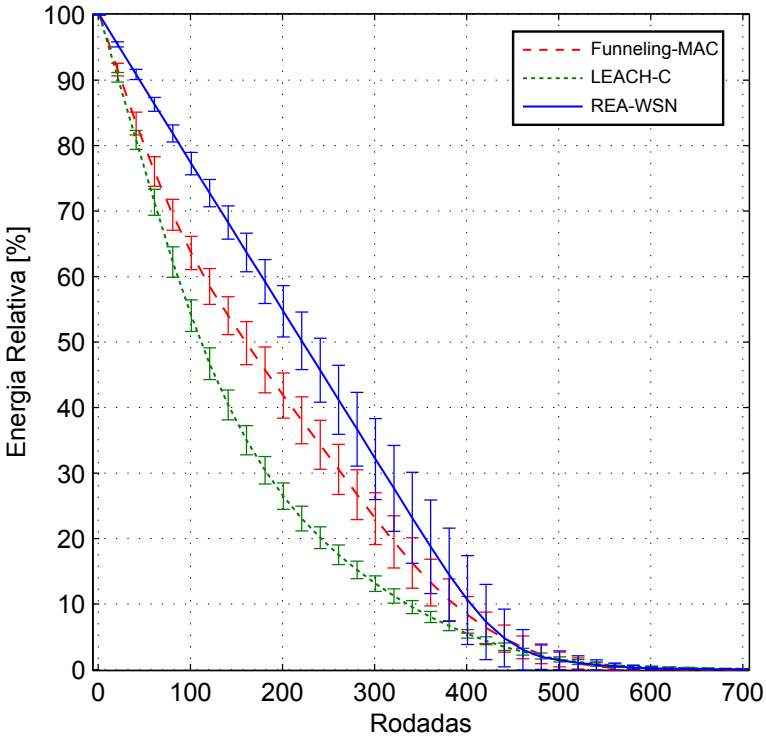


Figura 4.14: Energia Relativa.

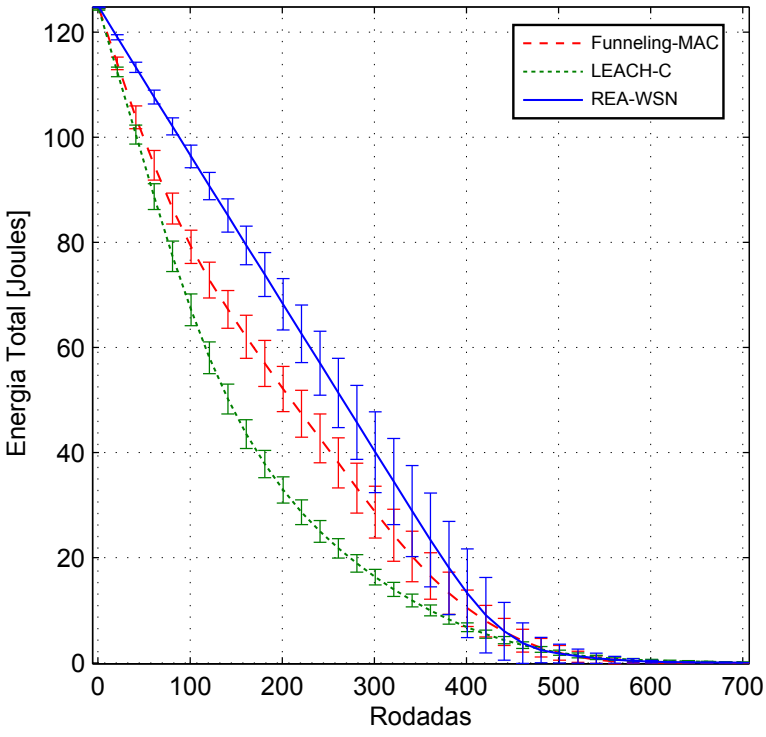


Figura 4.15: Energia Total.

como no algoritmo LEACH-C que morreu em média 235 nós, o que corresponde em torno de 90% do total de nós. Enquanto que no algoritmo de *funneling*-MAC morreram todos

os nós em torno da rodada 570. Isso indica que em relação ao número de nós inativos REA-WSN foi 5% inferior ao LEACH-C nesta métrica.

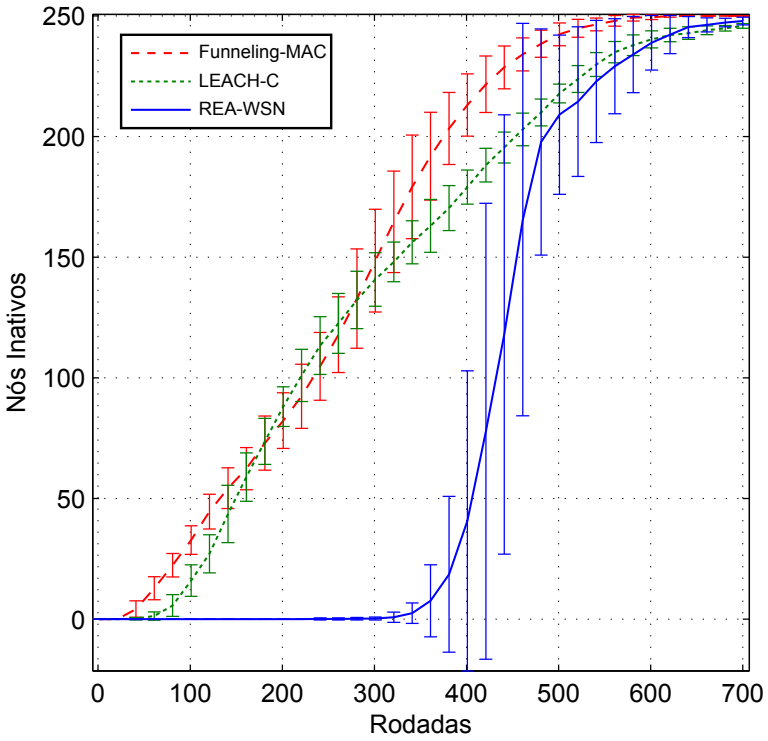


Figura 4.16: Nós inativos.

4.3.6 Análise dos gráficos

Ao analisar os gráficos observamos que no cenário 1 o algoritmo REA-WSN tem uma vantagem de 2% em comparação ao LEACH-C e 44% em relação ao *funneling*-MAC, mostrando-se superior nas métricas de energia relativa e total e em relação ao número de nós inativos mostrou-se igual ao LEACH-C, pois não morreu nenhum nó até o final das 200 rodadas, mas no *funneling*-MAC morreram em torno de 20 nós.

No cenário 2 foi observado que o algoritmo REA-WSN tem uma vantagem de 4% em comparação ao LEACH-C e 55% em relação ao *funneling*-MAC, mostrando-se superior nas métricas de energia relativa e total e em relação ao número de nós inativos mostrou-se igual ao LEACH-C, pois não morreu nenhum nó até o final das 250 rodadas, mas no *funneling*-MAC morreram em torno de 49 nós.

No cenário 3 foi observado ser o melhor cenário para o REA-WSN, pois foi onde mais teve vantagem sobre os demais algoritmos. O algoritmo REA-WSN tem uma vantagem de 21% em comparação ao LEACH-C e 40% em relação ao *funneling*-MAC, mostrando-se superior nas métricas de energia relativa e total e em relação ao número de nós inativos não morreu nenhum nó até o final das 300 rodada. No LEACH-C morreram em torno de

10 nós e no *funneling*-MAC morreram em torno de 90 nós. Isso se deve ao tamanho da área e número de nós que foi ideal para os experimentos com o algoritmo REA-WSN.

No cenário 4 foi observado que o algoritmo REA-WSN tem uma vantagem de 14% em comparação ao LEACH-C e 3% em relação ao *funneling*-MAC, mostrando-se superior nas métricas de energia relativa e total e em relação ao número de nós inativos morreram 170 nós no REA-WSN assim como no LEACH-C e no *funneling*-MAC morreram em torno de 190 nós ao final das 500 rodadas.

Por fim no cenário 5 foi observado que o algoritmo REA-WSN tem uma vantagem de 7% em comparação ao LEACH-C e 3% em relação ao *funneling*-MAC, mostrando-se superior nas métricas de energia relativa e total e em relação ao número de nós inativos morreram 240 nós no REA-WSN assim como no LEACH-C e no *funneling*-MAC morreram em torno de 250 nós ao final das 700 rodadas.

Ao observar os gráficos conclui-se que algoritmo REA-WSN é superior em todos os cenários, pois o consumo de energia é menor em comparação aos outros algoritmos. Esse resultado se justifica por que a descoberta de rota é feita nos CH. A desvantagem de usar esse algoritmo seria que ele tem uma restrição na alternância dos CH que tem que esperar concluir a rota para ter a alternância, não podendo alternar no meio do caminho. O cenário 3 mostrou-se ser o melhor cenário para o algoritmo REA-WSN, pois foi o cenário onde obteve-se resultados mais vantajosos que os demais algoritmos e o cenário 1 é padrão, pois é o mesmo cenário utilizado em LEACH-C [18] usado como base nesses experimentos.

Resumo do Capitulo

Neste Capítulo foi descrita a proposta deste trabalho, como junção de [1] e [18] com o aditivo do *Reduction Energy in WSN* (REA-WSN). Os resultados do algoritmo apresentado REA-WSN e pode-se concluir que em alguns cenários os gráficos do LEACH-C e do REA-WSN são semelhantes. Em *funneling*-MAC o gasto energético é superior em comparação aos outros algoritmos e REA-WSN mostrou-se ser superior em relação aos demais algoritmos em todos os cenários apresentados.

Capítulo 5

Conclusão

5.1 Conclusão e trabalhos futuros

Neste trabalho foi proposta um novo algoritmo para economia de energia em uma RSSF. A proposta se baseia na modificação do *funneling-MAC* [1], e na modificação do LEACH-C [18] com relação ao seu roteamento onde foi implementado o algoritmo de REA-WSN, que calcula o caminho de menor custo de energia até ao *sink*, com base no algoritmo de *Dijkstra*.

A rede é dividida em duas áreas (intensidade e não intensidade) e os nós são organizados em *clusters*. Em cada *cluster*, com intervenção do *sink* é calculado um *cluster head* que atuará como *gateway* do *cluster*. Os *cluster heads* da região de não intensidade farão a escolha dos *cluster heads* dentro da região de intensidade e enviaram os dados ao *sink* com base no algoritmo proposto REA-WSN.

Os resultados das simulações mostram que a proposta apresentada nessa dissertação é equivalente em alguns cenários ao LEACH-C, porém na média se mostrou superior em todas as simulações. Um dos problemas do LEACH-C é que, como os *clusters heads* enviam os dados de seu *cluster* diretamente para o *sink*, os *clusters heads* que estão mais distantes gastam mais energia no envio destes dados ao *sink*. Na proposta deste trabalho é proposto o encaminhamento ao *cluster head* com mais energia relativa e esse enfoque parece ter promovido uma melhor eficiência energética na rede.

Em relação a escalabilidade da rede é possível ver que quanto maior a escalabilidade, os resultados da proposta se mostram melhores em relação aos demais algoritmos. Pois LEACH-C quanto mais distante estiver o nó do *sink*, mais potência terá que gastar para enviar dados. No caso do *funneling*-MAC, quanto mais distante estiver do *sink*, mas potência e mais caminhos terá que usar para passar os dados ao *sink*.

As limitações da proposta desenvolvida nesta dissertação se concentram no controle centralizado dos *clusters* e *clusters heads* pelo *sink*, o que pode limitar o campo de apli-

cação da RSSF. Assim também foram adotadas as premissas de uma rede homogênea e com baixa movimentação dos sensores. Como trabalho futuro se propõe um maior aprofundamento nessas limitações para avaliar parâmetros de escalabilidade e movimentação dos nós.

Referências

[1] G. Ahn, S. Hong, E. Miluzzo, A. T. Campbell, and F. Cuomo. Funneling-MAC: A localized, sink-oriented MAC for boosting fidelity in sensor networks. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 293–306, New York, NY, USA, 2006. ACM. viii, ix, xii, 2, 15, 25, 26, 27, 33, 34, 36, 38, 41, 53, 54

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002. 1

[3] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella. Energy conservation in wireless sensor networks: A survey. *Ad Hoc Networks*, 7(3):537–568, 2009. 15

[4] M. Baek, K. Kim, and S. Cho. A revised mint-route protocol in wireless sensor networks. In *Information and Communication Technology Convergence (ICTC), 2010 International Conference on*, pages 258–259, Nov 2010. 26

[5] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *Personal Communications, IEEE*, 7(5):28–34, Oct 2000. 16

[6] E. Callaway, P. Gorday, L. Hester, J.A. Gutierrez, M. Naeve, B. Heile, and V. Bahl. Home networking with IEEE 802.15.4: a developing standard for low-rate wireless personal area networks. *Communications Magazine, IEEE*, 40(8):70–77, Aug 2002. 11

[7] C. Chee-Yee and S.P. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, Aug 2003. 4, 5, 6

[8] S. Choudhary and A. Bhatt. A survey of optimized link state routing (OLSR) networks. *International Journal of Scientific Research in Science, Engineering and Technology*, 1(2):408–415, 2015. 14

[9] E. Costa. *Jogo interativo para reabilitaçao de pacientes com cancro da mama*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2015. 10

[10] W. Dargie and C. Poellabauer. John Wiley & Sons, Ltd, 2011. xii, 8, 9, 10, 11, 12, 15

[11] I. Demirkol, C. Ersoy, and F. Alagoz. MAC protocols for wireless sensor networks: A survey. *Communications Magazine, IEEE*, 44(4):115–121, April 2006. 22

[12] E. Diniz and F. Demarchi. *Desenvolvimento de um sistema para auxílio à locomoção de deficientes visuais através da implementação em arquiteturas reconfiguráveis da transformada Census para estimação de distância usando visão estéreo.* PhD thesis, Universidade de Brasilia, 2014. 10

[13] L. do Couto Antunes. *Identificação de pessoas numa portaria virtual.* PhD thesis, Instituto Superior de Engenharia de Lisboa, 2012. 6, 7

[14] G. Fortino and A. Liotta. Internet of things: Technology, communications and computing. In C. Aggarwal, editor, *Internet of Things*. Springer US, 2014. 2

[15] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services; Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond. 2

[16] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, pages 174–185, New York, NY, USA, 1999. ACM. 18

[17] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10 pp. vol.2–, Jan 2000. xii, 2, 16, 19, 25, 28, 29

[18] W.R. Heinzelman, A.P. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *Wireless Communications, IEEE Transactions on*, 1(4):660–670, Oct 2002. viii, ix, 2, 19, 25, 30, 33, 34, 35, 36, 38, 39, 41, 53, 54

[19] Xavier G. Sales M. Holanda, C. Redes de sensores sem fio. `http://www.gta.ufrj.br/grad/11_1/rssf/`, 2011. xii, 24

[20] B. Hull, K. Jamieson, and H. Balakrishnan. Mitigating congestion in wireless sensor networks. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, SenSys '04, pages 134–147, New York, NY, USA, 2004. ACM. 26

[21] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 56–67, New York, NY, USA, 2000. ACM. 19

[22] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, February 2003. 19

[23] A. Jain and R. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Inc., 1988. 28

[24] Z. Jianliang and M.J. Lee. Will IEEE 802.15.4 make ubiquitous networking a reality?: a discussion on a potential low power, low bit rate standard. *Communications Magazine, IEEE*, 42(6):140–146, June 2004. 11

[25] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. Wiley-Interscience, 2007. xii, 7, 8

[26] J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wirel. Netw.*, 8(2/3):169–185, March 2002. 18

[27] M. Labrador and P. Wightman. *Topology Control in Wireless Sensor Networks*, volume 412. Springer, 2009. 7

[28] S. Lindsey and C. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *Aerospace conference proceedings, 2002. IEEE*, volume 3, pages 3–1125. IEEE, 2002. 2

[29] S. Lindsey and C.S. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *Aerospace Conference Proceedings, 2002. IEEE*, volume 3, pages 3–1125–3–1130 vol.3, 2002. 20

[30] D. Militani. Um algoritmo para auto-configuração do protocolo de roteamento OLSR. 2015. 14

[31] T. Murata and H. Ishibuchi. Performance evaluation of genetic algorithms for flowshop scheduling problems. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 812–817 vol.2, Jun 1994. 30

[32] C. Myers, A. Oppenheim, R. Davis, and W. Dove. Knowledge based speech analysis and enhancement. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84.*, volume 9, pages 162–165, Mar 1984. 5

[33] B. Nath and D. Niculescu. Routing on a curve. *SIGCOMM Comput. Commun. Rev.*, 33(1):155–160, January 2003. 17

[34] N. Nema. Gateway based shortest path detection using dijkstra's algorithm and closest adjacency condition (DA-CAC) in WSN. `http://www.ijarcsse.com/thesis_publication.php`, 2014. 14

[35] I. Oliveira. *Rastreamento de indivíduos em sistema de monitoramento*. PhD thesis, Universidade de Brasilia, 2013. 5

[36] N.A. Pantazis, S.A. Nikolidakis, and D.D. Vergados. Energy-efficient routing protocols in wireless sensor networks: A survey. *Communications Surveys Tutorials, IEEE*, 15(2):551–591, Second 2013. 1, 2

[37] D. Pareit, B. Lannoo, I. Moerman, and P. Demeester. The history of wimax: A complete survey of the evolution in certification and standardization for IEEE 802.16 and wimax. *Communications Surveys Tutorials, IEEE*, 14(4):1183–1211, Fourth 2012. 6

[38] M. Pechoto, J. Ueyama, and J. de Albuquerque. E-noé: Rede de sensores sem fio para monitorar rios urbanos. *Instituto de Ciências Matemáticas e de Computação(ICMC-USP)*, 2012. 15

[39] H. Pei, X. Li, S. Soltani, M.W. Mutka, and X. Ning. The evolution of MAC protocols in wireless sensor networks: A survey. *Communications Surveys Tutorials, IEEE*, 15(1):101–120, First 2013. 23, 25

[40] T. Rappaport. *Comunicações sem fio: Princípios e Práticas*. Pearson Prentice Hall, 2009. 23, 25, 62, 66

[41] R. Rashid and G. Robertson. *Accent: A Communication Oriented Network Operating System Kernel*. SOSP '81. ACM, New York, NY, USA, 1981. 5

[42] L. Rodrigues, C. Montez, P. Portugal, and F. Vasques. Escalonamento de sono e efeito recuperação em baterias de nodos em redes de sensores sem fio. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC*, 2014. 15

[43] P. P. Saraswala. A survey on routing protocols in zigbee network. *International Journal of Engineering Science and Innovative Technology (IJESIT) Volume*, 2, 2013. 6, 13

[44] R. Shelke, G. Kulkarni, R. Sutar, P. Bhore, D. Nilesh, and S. Belsare. Energy management in wireless sensor network. In *Computer Modelling and Simulation (UKSim), 2013 UKSim 15th International Conference on*, pages 668–671. IEEE, 2013. 1

[45] S. Singh, M. Singh, and D. Singh. A survey of energy-efficient hierarchical cluster-based routing. In *in Wireless Sensor Networks Int. J. of Advanced Networking and Applications (2010), VoL: 02, Page: 570-580 www.ijmer.com 489 | Page*, 2010. 19

[46] K. Sohraby, D. Minoli, and T. Znati. *Wireless Sensor Networks: Technology, Protocols, and Applications*. John Wiley & Sons, Inc., 2007. 6

[47] G.M. Tamilselvan and S. Kiruthika. An energy efficient data aggregation based medium access control protocol using centre at nearest source approach for sensor networks in a lattice topology. In *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, pages 1–6, Feb 2012. 15, 23

[48] O. Taneja. *Novel Approaches for increasing Energy Efficiency in Wireless Sensor Network*. PhD thesis, Thapar University Patiala, 2014. 5

[49] Setting the Standard for Automation. ISA develops. http://www.isa.org/standards-and-publications/isa-publications/intech-magazine/2004/may/networking-and-communications-zigbee-short-on-power-by-design, 2015. xii, 12

[50] A. B. Verona. *Simulação e Análise de Redes de Sensores Sem Fio Aplicadas à Viticultura*. PhD thesis, Dissertation-MSc in computer science. v548s. State University of Maringá-Brazil, Graduate Program in Computer Science, 2010. 15

[51] W. Xiaohang et al. Video streaming over bluetooth: A survey. *Institute for Infocomm Research (I2R)/School of Computing, NUS*, 2011. 6

[52] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. Technical report. 17

Apêndice A

Anexo 1

Algoritmo 3: Função de Parâmetros

```
1  início
2      xm=200; ym=200;
3      RI_raio = 0.3×(xm + ym)/2;
4      sink.x=0.5 × xm;
5      sink.y=0.5 × ym;
6      n=100;
7      p=0.1;
8      minNumClusters=2; maxNumClusters=30;
9      Eo=0.5;
10     ETX=50× 10^-9;
11     ERX=50× 10^-9;
12     Efs=10×10^-13;
13     Emp=0.0013×10^-13;
14     EDA=5× 10^-9;
15     ESX=50× 10^-9;
16     tamMsg=4000;
17     clock=737× 10^-4;
18     tempo do clock=1/737× 10^-4;
19     energia ativa por segundo=33*× 10^-7;
20     rmax=200;
21 fim
```

O Algoritmo 4 tem-se as dimensões do campo (linha 2), a delimitação da RI (linha 3) que aqui sempre vai ser 30% do tamanho da rede, as coordenadas do *sink* (linhas 4 e 5), número de nós no campo (linha 6), probabilidade de eleger *clusters heads* (linha 7) isso vem do algoritmo do LEACH-C que também tem essa probabilidade, o minimo e o máximo de *clusters* dentro da rede foi uma condição adotada na proposta, pois tem que

ter pelo menos um *cluster* dentro da RI e um fora para fazer a comunicação dos dados, a energia inicial dos nós.

Foi adotado o mesmo modelo de dissipação de energia do LEACH-C (linhas 11-15), onde o transmissor dissipa energia para funcionar o sistema eletrônico de rádio e o amplificador de potência, e o receptor dissipa energia para funcionar o sistema eletrônico de rádio. Para os experimentos descritos, tanto o espaço livre (perda de energia) e o *multipath* (perda de potência) foram utilizados modelos de canal, de acordo com a distância entre o transmissor e receptor [40]. O controle de alimentação pode ser utilizada para inverter esta perda, definindo adequadamente a potência do amplificador se a distância for menor do que um limiar ($d0$), o modelo (FS) espaço livre (do inglês *Free Space*) é utilizado; caso contrário, o modelo (mp) *multipath* é usado. O EDA é o gasto de energia da agregação também adotado do LEACH-C. Nas linhas 16 a 18 são usados para contabilizar o processamento do CH. Por fim é contabilizado o número das rodadas (linha 19), ou seja quantas vezes irá se repetir o algoritmo.

Algoritmo 4: VERIFICA NÓ MORTO

```
1  início
2      para para cada nó ∈ n faça
3          se S(i).E<=0 então
4              dead_t=dead_t+1;
5              se S(i).ENERGY==0 então
6                  dead_n=dead_n+1;
7              fim
8          fim
9          se S(i).E>0 então
10             S(i).type='N';
11         fim
12     fim
13 fim
```

O algoritmo 5 temos a verificação se existe algum nó morto na rede, se tiver algum contabiliza nos contadores.

O algoritmo 6 escolhe a configuração da rede e quem são os CHs, verifica se a rede está boa, ou seja, para a configuração da rede ficar boa, tem umas regras: tem que ter entre o minimo e o máximo de CHs, e pelo menos 1 dentro da RI. Na linha 5 é uma variável que conta o número de CHs escolhidos nesta rodada, e também é o índice do CH corrente, dentro do laço.

Nas linhas de 6 a 20 percorre todos os nós e verifica se não tem nenhum nó morto, faz escolha de um número aleatório pra usar no próximo SE, para entrar no SE eh por que o nó i foi escolhido como CH. Escolhido como CH. verifica qual a sua coordenada,

Algoritmo 5: Escolhe a configuração da rede: quem são os cluster heads

```
1  início
2      C = [];
3      RedeRuim=true;
4      enquanto RedeRuim faça
5          countCHs=0;
6          para para cada nó ∈ n faça
7              se S(i).E>0 & S(i).G==0 então
8                  temp_rand=rand;
9                  se temp_rand ≤ (p/(1-p × mod(r_aux,round(1/p)))) então
10                     countCHs=countCHs+1;
11                     S(i).type='C';
12                     S(i).G=round(1/p)-1;
13                     C(countCHs).xd=S(i).xd;
14                     C(countCHs).yd=S(i).yd;
15                     C(countCHs).node=i;
16                     distance= √((S(i).xd − (S(n+1).xd))² + (S(i).yd − (S(n+1).yd))²);
17                     C(countCHs).distance=distance;
18                     C(countCHs).id=i;
19                 fim
20             fim
21         fim
22         soma = sum([S([S(:).E] > 0).E]); dentro_RI=0;
23         para para cada cluster head ∈ countCH faça
24             se C(i).distance < RI_raio então
25                 dentro_RI=dentro_RI+1;
26             fim
27         fim
28         r_aux=r_aux+1;
29         se dentro_RI>0 & countCHs > minNumClusters & countCHs ≤
           maxNumClusters então
30             RedeRuim=false;
31         senão se  mod (r_aux, round(1/p))==0 então
32             para  para cada nó ∈ n faça
33                 S(i).G=0;
34             fim
35             C = [];
36         fim
37     fim
38 fim
```

seu número de identificação e verifica a distância euclidiana do CH em relação ao *sink*. Verifica-se a quantidade de energia dos nós (linha 22). Para cada CH se estiver dentro da RI, adiciona mais um da variável, dizendo que tem mais um CH dentro da RI (linhas 23-27). O auxiliar do número de rodadas conta todas as rodadas (linha 28), mesmo aquelas

que deram uma configuração de rede ruim.

Nas linhas 29 a 37 verifica-se a condição de ser uma boa configuração de rede: pelo menos 1 CH dentro da RI e entre o minimo e o máximo de CH no total, se entrar na condição, só irá sair por que a rede esta boa, seguindo as condições definidas. Quando não entrar no SE, ele vai voltar no ENQUANTO, por que a RedeRuim ainda é true. Se mod de r(numero da rodada que está), round de 1/p(0.1) for igual a 0 (zero) entra no PARA, percorre todos os nós e G=0 aos nós. Por fim tem-se um vetor de CHs, que é zerado pois vai escolher novos CH, por que a rede escolhida não é boa.

Algoritmo 6: Associação de Cluster Head aos nós

```
1  início
2      para para cada nó ∈ n faça
3          se S(i).type=='N' && S(i).E>0 && countCHs ≥ 1 então
4              min_dis=xm+ym;
5              para para cada cluster head ∈ countCHs faça
6                  se (S(i).sinkDist<RI_raio && C(c).distance<RI_raio) || (S(i).sinkDist ≥
                   RI_raio && C(c).distance ≥ RI_raio) então
7                      aux_dis=√((S(i).xd − C(c).xd)² + (S(i).yd − C(c).yd)²); se
                       aux_dis<min_dis então
8                          min_dis=aux_dis;
9                          min_dis_cluster=c;
10                     fim
11                 fim
12             fim
13             S(i).myCH=min_dis_cluster;
14             se min_dis>do então
15                 S(i).E=S(i).E- (ETX × tamMsg + Emp × tamMsg × (min_dis)⁴);
16             fim
17             se min_dis ≤ do então
18                 S(i).E=S(i).E-(ETX × tamMsg + Efs × tamMsg × (min_dis)²);
19             fim
20             se min_dis>0 então
21                 S(C(min_dis_cluster).id).E = S(C(min_dis_cluster).id).E-((ERX +
                   EDA) × tamMsg );
22             fim
23             S(i).min_dis=min_dis;
24             S(i).min_dis_cluster=min_dis_cluster;
25         fim
26     fim
27 fim
```

No algoritmo 7 nas linhas 2 a 12 percorre todos os nós se do tipo normal e ao mesmo tempo sua energia estiver maior que 0 (zero) e se countCHs for maior ou igual a 1, ou seja, se não tiver nenhum CH busca o CH de menor distancia, verifica se estão na mesma região seja RI ou RNI, calcular o minimo da distância para cada CH, utiliza pitágoras para verificar a distancia entre o nó e o CH, se aux_min for menor que o minimo então esse passa a ser a menor distancia e esse(c) é o CH. Escreve na figura o número do CH correspondente mais perto e preenche o CH do nó com o (c) escolhido.

Nas linhas 13 a 22 calcula-se a energia dissipada nos nós, como já foi citado no Algoritmo 4 foi adotado o mesmo modelo de dissipação de energia do LEACH-C, onde o transmissor dissipa energia para funcionar o sistema eletrônico de rádio e o amplificador de potência, e o receptor dissipa energia para funcionar o sistema eletrônico de rádio. Para os experimentos descritos, tanto o espaço livre (perda de energia) e o *multipath*

(perda de potência) foram utilizados modelos de canal, de acordo com a distância entre o transmissor e receptor [40]. O controle de alimentação pode ser utilizada para inverter esta perda, definindo adequadamente a potência do amplificador se a distância for menor do que um limiar ($d0$), o modelo (FS) espaço livre é utilizado; caso contrário, o modelo (mp) *multipath* é usado. O EDA é o gasto de energia da agregação também adotado do

LEACH-C. Nas linhas 23 e 24 só atualiza as distâncias minimas.

Algoritmo 7: FUNÇÃO DE CRIAÇÃO DE CAMINHOS E CALCULO DE ENERGIA;

```
1  início
2      para para cada nó i ∈ countCHs faça
3          C(i).caminho=[0];
4          se C(i).distance<RI_raio então
5              C(i).caminho=[i C(i).caminho];
6              se C(i).distance>do então
7                  S(C(i).node).E=S(C(i).node).E-((ETX+EDA) × tamMsg + Emp ×
                   tamMsg × (C(i).distance)^4);
8              fim
9              se C(i).distance ≤ do então
10                 S(C(i).node).E=S(C(i).node).E-((ETX+EDA) × tamMsg + Efs ×
                   tamMsg × (C(i).distance)^2);
11             fim
12         senão
13             RI_min_dist=xm+ym;CH_RI_min_dist=0;
14             C(i).perto=[];cj=0; CH da RI mais Perto da RNI;
15             para para cada nó i ∈ countCHs faça
16                 se √((C(i).xd − C(j).xd)^2 + (C(i).yd − C(j).yd)^2) ≤ RI_min_dist
                   então
17                     C(i).perto=[ C(i).perto j];cj=cj+1;
18                 fim
19             fim
20             Cria Caminho CH_RNI_RI;
21             Caminho=C(i).perto(paths);C(i).caminho=[Caminho C(i).caminho];
22             para para cada j ∈ length(C(i).caminho)-2 faça
23                 idx_1=0;
24                 para para cada ∈ cj faça
25                     se C(i).perto(l)==C(i).caminho(j) então
26                         idx_1=l;
27                     fim
28                 fim
29                 idx_2=0;
30                 para l=1:cj faça
31                     se C(i).perto(l)==C(i).caminho(j+1) então
32                         idx_2=l;
33                     fim
34                 fim
35                 S(C(C(i).perto(idx_1)).node).E=S(C(C(i).perto(idx_1)).node).E-
                   C(i).E_temp(idx_1,idx_2);
36             fim
37         fim
38     fim
39 fim
```

No algoritmo 8 faz a criação do caminho do CH C(i) ate o *sink*, O *cluster* 0 (zero) é considerado o *sink* o caminho termina sempre nele. Nas linhas 4 e 5 verifica se os nós estão dentro da RI e cria o caminho de cada nó. Nas linhas 6 a 11 calcula o gasto de energia conforme o modelo de dissipação do LEACH-C dentro da RI, ou seja, se a distância for maior do que o limiar da potência d_0 então aplica-se o modelo de *multipath* senão aplica-se o modelo de espaço livre. Nas linhas 12 a 21 se fora da RI escolhe o CH mais próximo de dentro da RI, o CH de dentro da RI vai estar mais perto que esse valor, inicia com zero o *cluster* da RI escolhido, inicia a lista de todos os CH que estão perto, para poder fazer o caminho até o *sink* (esse caminho vai com certeza passar por alguns CH que estão perto, mas precisa de todos do *Dijkstra* para calcular).

Acha o índice do CH dentro da RI e sua distância, até aqui já se sabe o CH escolhido da RI que ta mais perto do CH i. Preenche o vetor de CH da RNI que estão perto do CH i(lembre-se que i ta na RNI), para poder calcular o caminho até o CH da RI que já foi escolhido. Para todos os CHs mais perto do CH corrente que o CH de dentro da RI, adiciona o j na lista dos que estão perto de i, adiciona um elemento no tamanho da lista de caminhos perto e aumenta 1 no contador. Nas linhas 22 a 36 faz a criação do caminho da RNI até o *sink*, percorre o caminho e calcula as energias, busca o CH mais próximo de j da lista de caminho perto de i. Por fim calcula a energia e subtrai do CH escolhido e o

próximo CH no caminho.

Algoritmo 8: Algoritmo do menor caminho

início
  M=zeros(C(i).cj,C(i).cj);C(i).E_temp=zeros(C(i).cj,C(i).cj);
  para *para cada nó* $j \in C(i).cj$ **faça**
    para *para cada nó* $k \in C(i).cj$ **faça**
      se *if k!=j* **então**
        $M(j,k) = \sqrt{((C(C(i).perto(j)).xd - C(C(i).perto(k)).xd)^2 + (C(C(i).perto(j)).yd - C(C(i).perto(k)).yd)^2)};$
        se $M(j,k) > d_0$ **então**
          $C(i).E\_temp(j,k) = ((ETX + EDA) \times (tamMsg) + Emp \times tamMsg \times (M(j,k)^4));$
        senão
          $C(i).E\_temp(j,k) = ((ETX + EDA) \times (tamMsg) + Efs \times tamMsg \times (M(j,k)^2));$
        fim
      fim
    fim
  fim
  A_temp=1-eye(C(i).cj);instrucao=0; comeco=0;
  para *para cada nó* $l \in C(i).cj$ **faça**
    se *C(i).perto(l)==i* **então**
      comeco=l;
    fim
  fim
  instrucao=instrucao+C(i).cj; fim=0;
  para *para cada nó* $l \in C(i).cj$ **faça**
    se *C(i).perto(l)==CH_RI_min_dist* **então**
      fim=l;
    fim
  fim
  instrucao=instrucao+C(i).cj; instrucao=instrucao+$C(i).cj^2 + C(i).cj\times$ log(C(i).cj);
  TempoProcessamento=instrucao × tempo_clock;
  ProcessamentoEnergia=TempoProcessamento × energiaAtivaPorSegundo;
  S(C(i).id).E=S(C(i).id).E-ProcessamentoEnergia;
  [costs,paths]=dijkstra(A_temp,C(i).E_temp,comeco,fim);
  S(C(i).id).E=S(C(i).id).E-C(i).E_temp(comeco,fim);
  para *para cada nó* $l \in C(i).cj$ **faça**
    se *C(i).perto(l) =i* **então**
      S(C(C(i).perto(l)).id).E=S(C(C(i).perto(l)).id).E-((ERX) × (tamMsg));
      S(C(C(i).perto(l)).id).E=S(C(C(i).perto(l)).id).E-C(i).E_temp(l,comeco);
      S(C(i).id).E=S(C(i).id).E-((ERX) × (tamMsg));
    fim
  fim
fim

No algoritmo 9 faz o custo do caminho mais curto até chegar ao *sink*. Na linha 2 a matriz que contêm as distâncias entre todos os CH da lista perto e a matriz que contêm o quanto que gastaria de energia entre todos os CH da lista perto, são preenchidas no próximo PARA. Nas linhas de 3 a 14 percorre as linhas e as colunas da matriz, se linha for diferente de coluna preenche M, a matriz de distancias e preenche a matriz E_temp com as energias. Na linha 15 matriz identidade invertida onde todos se conectam e inicializa instrução e começo. Nas linhas 16 a 25 percorre a lista de CH e verifica qual o CH que tem o inicio (começo) da rota e o que o final(fim) da rota a ser percorrida. Na 27 a 30 faz o calculo do tempo de processamento e energia gasta em cada CH. Na linha 31 faz o Dijkstra onde se obtêm a lista de caminhos da rota do caminho de menor energia. Nas linhas 33 a 39 calculo da energia para a comunicação inicial, seria um pulso par saber quem está próximo e sincronização dos dados.