UNIVERSIDADE FEDERAL DO CEARÁ
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DANIEL RIBEIRO MATOS

UM ALGORITMO DE COBERTURA DISTRIBUÍDO PARA ESCALONAMENTO DO RÁDIO DE SENSORES EM RSSF

FORTALEZA, CEARÁ

2013

DANIEL RIBEIRO MATOS

UM ALGORITMO DE COBERTURA DISTRIBUÍDO PARA ESCALONAMENTO DO RÁDIO DE SENSORES EM RSSF

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Miguel Franklin de Castro

Co-Orientador: Prof. Dr. Gabriel Antoine Louis Paillard

FORTALEZA, CEARÁ

2013

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Ciências e Tecnologia

M381a Matos, Daniel Ribeiro.
Um algoritmo de cobertura distribuído para escalonamento do rádio de sensores em RSSF / Daniel Ribeiro Matos. – 2013.
60f. : il. color., enc. ; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Departamento de Computação, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2013.
Área de Concentração: Ciência da Computação.
Orientação: Prof. Dr. Miguel Franklin de Castro.
Coorientação: Prof. Dr. Gabriel Antoine Louis Paillard.

1. Rádio - Transmissores e transmissão. 2. Sinais e sistemas. 3. Algorítmos computacionais. 4. Sistemas de comunicação sem fio. I. Título.

CDD 005

DANIEL RIBEIRO MATOS

UM ALGORITMO DE COBERTURA DISTRIBUÍDO PARA ESCALONAMENTO DO RÁDIO DE SENSORES EM RSSF

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação, da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação. Área de concentração: Ciência da Computação

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Dr. Miguel Franklin de Castro
Universidade Federal do Ceará - UFC
Orientador

Prof. Dr. Gabriel Antoine Louis Paillard
Universidade Federal do Ceará - UFC
Co-orientador

Prof. Dr. Angelo Roncalli Alencar Brayner
Universidade de Fortaleza - Unifor

Aos meus Pais, por tudo. À minha esposa, pelas várias noites ausentes enquanto concluía este trabalho. Aos meus orientadores, professores Miguel e Gabriel, pela compreensão e principalmente, pela atenção recebida e pelo conhecimento adquirido. Aos meus colegas e chefes de trabalho, em especial ao Vlademiro e aos professores Joaquim Bento e José Maria, pela compreensão e apoio para que eu dispusesse apenas de parte de minha dedicação ao trabalho, enquanto o resto era aplicado à conclusão desta dissertação.

“Ninguém ignora tudo. Ninguém sabe tudo. Todos nós sabemos alguma coisa. Todos nós ignoramos alguma coisa. Por isso aprendemos sempre”

(Paulo Freire)

RESUMO

Redes de Sensores Sem Fio (RSSF) são utilizadas em diversos tipos de aplicações: desde casas inteligentes a aplicações militares. RSSF possuem, em geral, severas restrições energéticas - um sensor geralmente possui uma quantidade limitada de bateria e este não é substituível. Os sensores podem possuir uma certa redundância de uma área sensoreada, uma vez que, quando os sensores são distribuídos de forma aleatória, alguns sensores acabam ficando muito próximos, ou mesmo quando são depositados de maneira determinística, uma certa redundância é necessária para prever a falha de alguns destes sensores. Neste trabalho, propomos um algoritmo distribuído que faz um escalonamento de sensores ativos, de forma a reduzir a redundância dos dados coletados e aumentar o tempo de vida da rede de sensores.

Palavras-chave: Redes de Sensores Sem Fio. Gerência de Cobertura. Escalonamento por Inversão de Arestas.

ABSTRACT

Wireless Sensor Networks (WSNs) are used in a lot of applications: from smart homes to military enviromnets. In general, WSNs has severe energy restrictions - a sensor usualy has a limited batery and it's not replaceable. Distributing the sensor in a random mander can lead to a redundancy of some areas and this is desirable to support fail of some sensors. In this work, we propose an distributed algorithm to schedule active sensors to reduce the redundancy of data obtainned by the network and prolong the network lifetime.

Keywords: Wireless Sensor Networks. Coverage problem. Scheduling by edge reversal.

SUMÁRIO

1 INTRODUÇÃO

Os avanços na área de dispositivos eletrônicos - com novas técnicas de miniaturização, redução de consumo de energia e aumento no poder de processamento, vem tornando possível a popularização do uso de dispositivos de baixo custo, com boa capacidade de processamento e comunicação sem fio. As Redes de Sensores Sem Fio - RSSF, fazem amplo uso deste tipo de dispositivo para as mais variadas aplicações, como monitoramento ambiental, agricultura, monitoramento de saúde, estudo e monitoramento do clima, aplicações militares dentre outras. Além das tradicionais aplicações de RSSF, temos toda uma demanda de aplicações da chamada "Internet das Coisas"(Internet of Things - IoT), onde temos um conjunto de tecnologias que proveem a todas as "coisas", conectividade a toda hora e em todos os locais (ATZORI; IERA; MORABITO, 2010). Na figura 1.1 temos exemplos de aplicações de RSSF na IoT

Figura 1.1: Exemplos de aplicações de Redes de Sensores Sem Fio na Internet das Coisas (IoT). Adaptação do infográfico "A INTERNET e as COISAS"obtida em (CISCO, 2013). © 1992 - 2011 Cisco Systems, Inc.

Neste cenário de computação ubíqua e pervarsiva - a computação em todo lugar, onde *smartphones* podem comunicar-se com a geladeira, sensores de iluminação e com o restaurante da esquina, a pesquisa em redes de sensores sem fio - bastante explorada nos últimos

anos, ganha uma nova abordagem - onde temos sensores bastante heterogêneos em capacidade de comunicação, processamento e energia.

Embora os dispositivos continuem dando saltos em tecnologia, especialmente na área de processamento e armazenamento, o mesmo não se pode dizer de sua capacidade energética. A inovação na área de armazenamento de energia não tem acompanhado a demanda crescente dos dispositivos eletrônicos, tornando ainda mais importante o uso eficiente dos recursos computacionais disponíveis. Além disto, mesmo com o aumento do processamento disponível a um custo cada vez mais baixo, ainda se faz necessário o uso de dispositivos com custo extremamente reduzido e restrições energéticas ainda mais altas - como, por exemplo, sensores a serem embutidos em paredes de modo a prevenir o colapso de estruturas em um prédio ou ajudar na busca por sobreviventes de um eventual desastre.

Uma das técnicas mais utilizadas em RSSF para que o tempo de vida da rede seja prolongado é o gerenciamento de cobertura da rede. A cobertura pode ser definida por quanto da área a ser sensoreada está coberta por sensores. Como, em geral, o custo dos dispositivos em uma RSSF deve ser baixo, costuma-se preencher a área a ser monitorada com uma boa quantidade de sensores, implantados - na maioria das vezes, de maneira aleatória, o que pode causar redundância nos dados obtidos por sensores muito próximos. A gerência de cobertura atua ativando ou desativando sensores, de modo a poupar energia, eliminando ou reduzindo a redundância dos dados medidos. É necessário entretanto, tomar cuidado ao desligar nós, pois isto pode afetar a conectividade da rede, deixando sensores sem comunicação. O desenvolvimento de soluções na área deve considerar alguns objetivos de alto-nível, como robustez, escalabilidade e simplicidade (WANG; XIAO, 2006). As pesquisas na área vem utilizando-se de técnicas cada vez mais complexas para obter uma boa gerência de cobertura. (SHARMILA; SUJITHA; RAJKUMAR, 2013) utilizam uma técnica conhecida como *Virtual Backbone Scheduling* para balancear e diminuir o gasto de energia, aumentando o tempo da vida de uma rede com sensores redundantes. Em (CHEN; ZHANG; KUO, 2013) os autores voltam-se para o problema da cobertura em RSSF multimídia, onde temos como sensores: câmeras, microfones e outros dispositivos multimídia. O trabalho baseia-se em uma métrica descrita como *Overlap-Sense Ratio* - OSR, para calcular a área de sobreposição de um sensor multimídia. Para decidir quais nós serão desligados devido à redundância, os autores propõem uma estratégia baseada em prioridades - definidas pelo valor do OSR, onde nós com maior OSR tem maior prioridade.

Nesta dissertação, trabalhamos a ideia do Algoritmo de Inversão de Arestas (BARBOSA; GAFNI, 1989), para utilizá-la no escalonamento do rádio dos nós em uma RSSF. Esta adaptação trás alguns desafios, que serão discutidos a seguir. A solução foi combinada a um algoritmo de roteamento hierárquico, de forma a analisar o real impacto na obtenção de dados em uma RSSF. Um dos principais requisitos almejados foi manter uma solução simples, escalável e robusta.

1.1 Estrutura da Dissertação

No capítulo 2 é apresentada a fundamentação teórica em RSSF, cobertura, roteamento e apresentados trabalhos relacionados na área. No capítulo 3 descrevemos em detalhes a solução proposta nesta dissertação. No capítulo 4 são apresentados os resultados obtidos nas simulações. Por fim, no capítulo 5, temos a conclusão do trabalho e propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Redes de Sensores Sem fio

O número de RSSF tem aumentado consideravelmente no cotidiano da sociedade atual. Há inúmeras aplicações de RSSF como casas inteligentes (*smart homes*), agricultura, pecuária, monitoramento de saúde, estudo e monitoramento do clima, diversos usos militares, detecção de intrusão, dentre vários outros. Sensores em uma RSSF possuem, em geral, restrições de processamento, armazenamento e, principalmente, restrições de energia. Muita pesquisa tem sido feita em busca de técnicas para minimizar o efeito destas restrições nas aplicações de RSSF.

As RSSF diferem-se das tradicionais redes sem fio em vários aspectos como, por exemplo, na restrição de energia: em uma rede sem fio tradicional - na maioria dos casos, possui uma fonte de energia inesgotável ou abundante, enquanto um sensor em uma RSSF possui, quase sempre, severas restrições energéticas, devendo reduzir ao máximo suas transmissões ao mesmo tempo em que deve executar corretamente a função para a qual foi planejado. Embora muitas das técnicas utilizadas em redes convencionais possam ser aplicadas às RSSF, suas características intrínsecas devem ser levadas em conta no seu desenvolvimento, por isso a pesquisa de algoritmos específicos para os diferentes tipos de RSSF é uma área em evidência nos últimos anos.

Um dos principais objetivos no desenvolvimento de uma RSSF, é lidar com a comunicação de forma a prolongar o tempo de vida da rede, além de prevenir a degradação da comunicação, através de técnicas de gerenciamento de energia (AL-KARAKI; KAMAL, 2004). Redes sem fio já existiam antes de RSSF, porém as características destas apresentam novos desafios. Por exemplo, devido à elevada quantidade de nós em uma RSSF, em geral, não é possível manter esquemas de endereçamento global para todos os nós, uma vez que sua capacidade de armazenamento e processamento é limitada, logo, lidar com muitos números de identificação não é uma opção - e a rede é descentralizada, ou seja, não há um dispositivo central que possa realizar este trabalho. Por conta destas restrições, também é necessário um bom gerenciamento dos recursos, de forma a não esgotá-los antes do tempo necessário para a aplicação realizar o seu trabalho. Ainda diferentemente de uma rede convencional, a ocorrência de falhas frequentes nos nós precisa ser considerada, o que pode afetar o desempenho de protocolos convencionais de Redes sem Fio em RSSF.

Na figura 2.1, temos a estrutura de um nó em uma RSSF, onde cada nó possui tarefas de sensoreamento, processamento, transmissão e unidade de energia. Também é possível ver a arquitetura de comunicação de uma RSSF - os sensores podem comunicar-se entre si, ou diretamente com uma estação base (*Base Station - BS na figura*). Em alguns tipos de sensores, podem ainda existir módulos de localização (como um dispositivo GPS) e módulos responsáveis pela locomoção do sensor (em casos de sensores móveis).

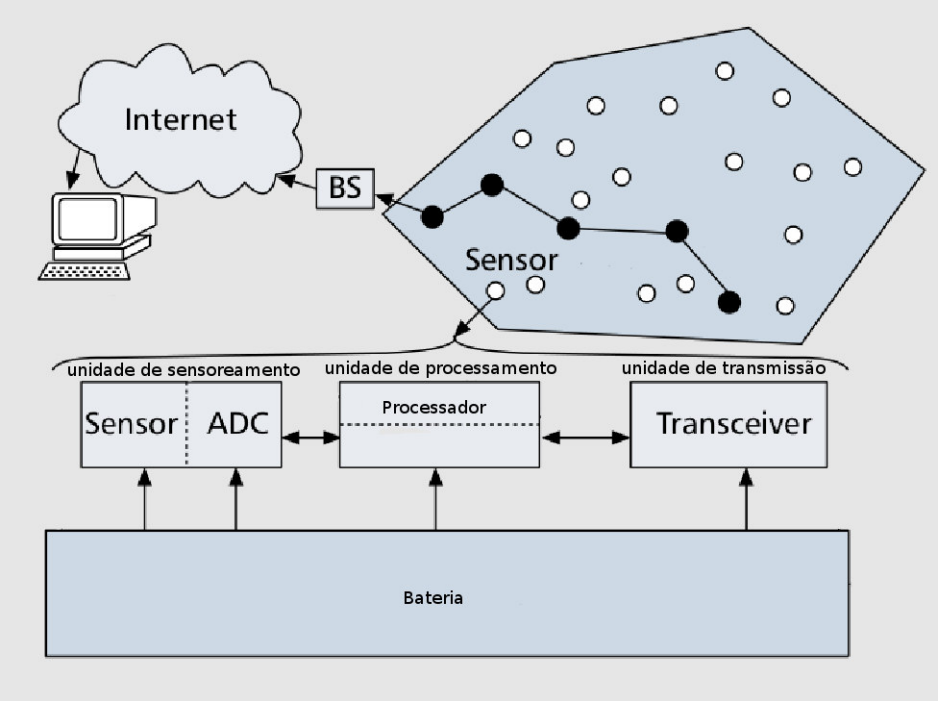

Figura 2.1: Componentes de um sensor. Adaptado de (AL-KARAKI; KAMAL, 2004)

2.2 Cobertura em Redes de Sensores Sem Fio

A cobertura é uma métrica que indica quanto da área monitorada por uma RSSF está sendo sensoreada pela mesma. A cobertura reflete em quão bem monitorada está a área coberta por uma RSSF (WANG, 2011). Quando temos um ponto no espaço coberto por k diferentes sensores, dizemos que o ponto está k-coberto ou possui uma k-cobertura. Na figura 2.2(b) e (d) temos os pontos denotados por uma estrela com uma 3-cobertura.

2.2.1 Modelos de Cobertura em RSSF

Modelos de cobertura em RSSF, medem a capacidade de sensoriamento - e a qualidade dos dados sensoreados, usando relações geométricas entre um dado ponto no espaço e a localização física dos sensores (WANG, 2011). Os principais modelos de cobertura podem ser classificados da seguinte maneira, conforme (WANG, 2011):

Setores Boleanos: Motivado pela ideia de uma câmera direcional – modelo utilizado em Redes de Sensores Sem Fio Multimídia (RSSF especializadas em transmissão de sons, imagens e outros dados multimídia), pode ser descrito como um setor circular, onde, qualquer ponto dentro deste setor possui cobertura 1 - e qualquer ponto fora possui cobertura 0. A figura 2.2(a) exibe este modelo.

Discos Booleanos: Modelo mais utilizado na literatura. A cada sensor é atribuído um raio de sensoreamento R_s. Se a distância euclidiana entre um ponto e a localização do sensor

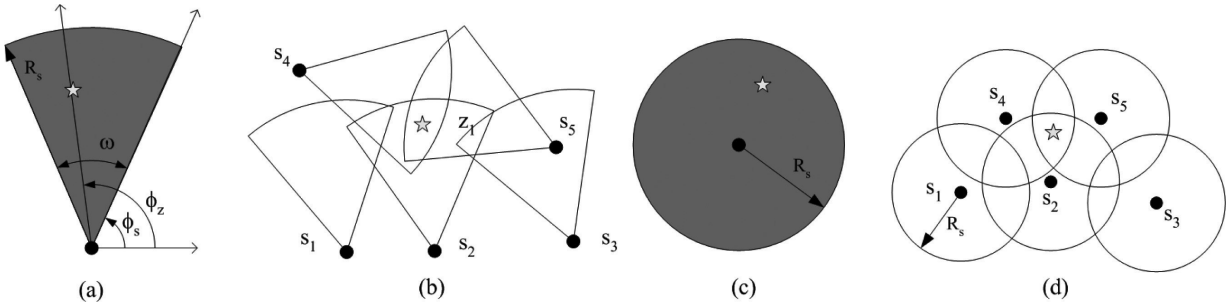

Figura 2.2: Modelos de Cobertura. Em (a) temos o modelo de Setores Booleanos. Em (b) temos a 3-cobertura de um ponto utilizando o modelo de Setores Booleanos. Em (c) temos o modelo de Discos Booleanos e em (d) temos um ponto com 3-cobertura pelo modelo de discos booleanos (WANG, 2011).

é menor que R_s, o ponto tem cobertura 1, caso contrário a cobertura é 0. Temos um exemplo na figura 2.2(c).

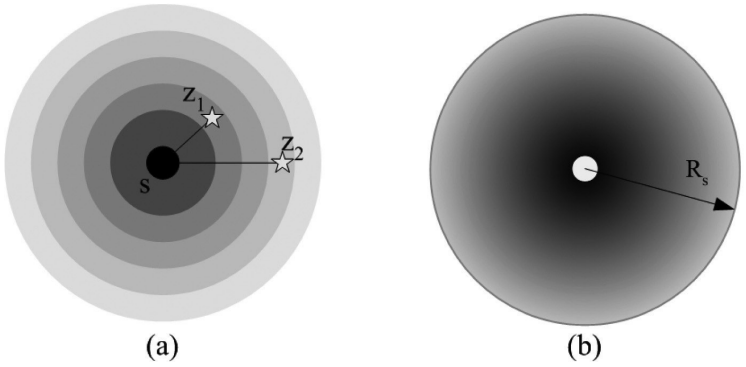

Figura 2.3: Modelos de Cobertura por Discos Atenuados(a) e Discos Atenuados Truncados(b). (WANG, 2011)

Discos Atenuados e Discos Atenuados Truncados: Alguns autores argumentam que, conforme um ponto se distancia do sensor, a qualidade dos dados sensoreados diminui. Neste modelo, um expoente de atenuação é utilizado para mensurar o valor da cobertura de determinado ponto a uma determinada distância do sensor como, por exemplo: $f(d(s,z)) = \frac{C}{d^{\alpha}(s,z)}$ (WANG, 2011) Onde α é o expoente de atenuação e C é uma constante. Utilizando este tipo de fórmula, para grandes distâncias o valor da cobertura fica muito pequeno, por conta disto, o modelo dos Discos Atenuados Truncados atua pondo um limite no raio sob o qual a equação de atenuação atuará, ou seja, após um determinado valor, a cobertura é considerada 0. A figura 2.3 mostra os modelos de Discos Atenuados e Discos Atenuados Truncados.

Modelos de Detecção e Modelos por Estimativa: Em modelos de detecção o valor da cobertura de um determinado ponto pode ser atribuído à probabilidade de ele detectar um evento ocorrido naquele ponto. Entre outros fatores, a probabilidade vai depender da distância do ponto ao(s) sensor(es). O modelo por estimativa tenta estimar o sinal obtido pelo sensor, baseado em dados como a distância e a atenuação de propagação do sinal.

2.2.2 Questões Principais no Desenvolvimento de Problemas de Cobertura

A seguir mostramos as principais questões, segundo (WANG, 2011), no desenvolvimento de soluções para os problemas de cobertura que devem ser consideradas.

Tipos de Coberturas em RSSF

O tipo de cobertura refere-se sobre o que, exatamente, está sendo monitorado por uma RSSF. Existem basicamente 3 tipos de cobertura, que são descritos abaixo e mostrados na figura 2.4.

Cobertura por pontos: os alvos a serem monitorados são modelados como pontos discretos dentro de uma área monitorada e o objetivo geral é cobrir todos os pontos.

Cobertura por área: Toda uma área será monitorada, todos os pontos são tratados igualmente, o objetivo geral é cobrir toda a área.

Cobertura de barreira: Constitui uma barreira para detecção de intrusão ou a descoberta de caminhos que penetram uma determinada área monitorada.O objetivo geral é cobrir uma linha, ou barreira, que não deve ser ultrapassada.

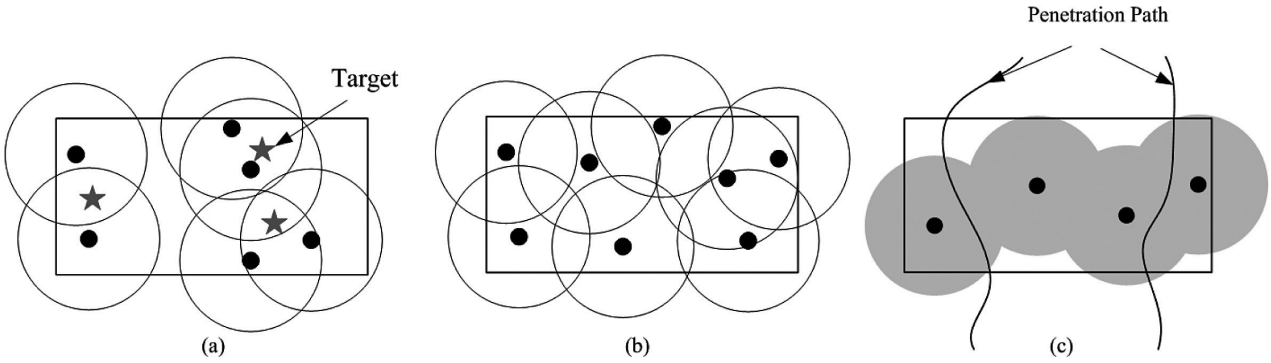

Figura 2.4: Tipos de cobertura. (a) cobertura por pontos (b) cobertura por área (c) cobertura de barreira. (WANG, 2011)

Método de Implantação

O método de implantação consiste em como a rede é implantada na área a ser monitorada. Geralmente, uma RSSF pode ser construída depositando-se os nós na área a ser monitorada de forma determinística - em locais pré-definidos, ou de forma aleatória. A maneira determinística é mais utilizada em redes pequenas sob um ambiente mais controlável. Quando a área a ser monitorada é suficientemente grande ou o ambiente de monitoramento é hostil, a implantação aleatória provavelmente é a única escolha. O modelo de implantação aleatória mais utilizada é o uniforme - onde cada nó tem a mesma probabilidade de estar em qualquer local da área a ser monitorada.

Grau de Cobertura

O grau de cobertura define como um ponto está coberto. O grau de cobertura indica quantos sensores estão cobrindo um determinado ponto. Quanto maior o grau de cobertura de uma rede, mais robusta é a mesma, pois um grau de cobertura de um determinado ponto possuir o valor k, significa que aquele ponto é tolerante à falha de k-1 sensores. O grau de cobertura é considerado como um dos requerimentos de aplicação para cobertura - ou seja, uma aplicação, como monitoramento ambiental, por exemplo, pode requerer um determinado grau de cobertura para funcionar corretamente, como toda a área monitorada sendo coberta por pelo menos 1 sensor.

Taxa de Cobertura

A taxa de cobertura é um valor que indica quanto da área monitorada - ou quantos pontos dela, está sendo coberto pelos sensores. Também é um dos requerimentos de aplicação para cobertura. Quando a aplicação requer uma taxa de cobertura de 100%, por exemplo, todos os pontos da área monitorada devem estar cobertos, neste caso, dizemos que é requerida uma cobertura completa.

Escalonamento de Atividade

É o escalonamento dos nós entre ativos e inativos. Se um ponto é coberto por mais de um sensor, o escalonamento decide quais sensores ficam ligados - e por quanto tempo, para cobrir aquele ponto, cumprindo os requerimentos de cobertura da aplicação.

Conectividade da Rede

A conectividade da rede consiste em garantir que cada nó da rede consiga enviar seus dados até o destino. Uma rede conectada garante que qualquer nó da rede consiga transmitir dados a outros nós e à estação base.

Cobertura e Roteamento

O gerenciamento de cobertura é normalmente implementando junto à camada rede, embora também possa ser feito na camada de acesso ao meio (MAC). Ao ser implementado na camada de rede, a técnica utilizada para a gerência da cobertura deve integrar-se bem ao algoritmo de roteamento utilizado. A próxima seção trás uma visão geral sobre algoritmos de roteamento em RSSF.

2.3 Algoritmos de Roteamento em Redes de Sensores Sem Fio

O roteamento das informações em uma RSSF pode ser bem desafiante. Em geral, a RSSF é composta de centenas ou milhares de sensores, onde não é possível manter um sistema de endereçamento global ou um agente responsável pela manutenção de uma tabela de rotas ou endereços. Desta forma, os nós precisam atuar colaborativamente, de maneira distribuída, visando conseguir entregar os dados sensoreados à Estação Base. Este problema tem sido estudado já há bastante tempo, e existem várias soluções que aplicam-se de maneira diferente para os diversos tipos de aplicação - pois dada a diversidade destas, não existe um melhor algoritmo - que funcione bem para todos os cenários. Em geral, existem determinados tipos de algoritmos que funcionam bem para determinados tipos de aplicação.

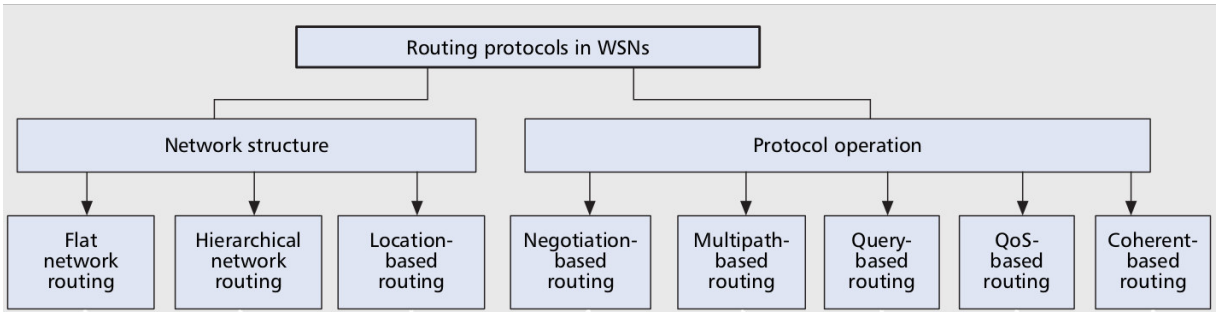

Figura 2.5: Classificação de protocolos de roteamento para RSSF. (AL-KARAKI; KAMAL, 2004)

Na figura 2.5, temos uma classificação de algoritmos de roteamento de acordo com a estrutura da rede e operação do protocolo, segundo (AL-KARAKI; KAMAL, 2004). Os protocolos para RSSF podem ser divididos, de acordo com a estrutura da rede, em três tipos:

Roteamento Plano: todos os nós tem tipicamente as mesmas regras e funcionalidades. Todos os nós podem participar do caminho de roteamento até a Estação Base.

Roteamento Hierárquico: os nós são divididos em agrupamentos - *clusters*, e recebem diferentes papéis na rede. Em cada *cluster* há a figura do *Cluster Head*, um nó central que recebe as informações dos nós de seu grupo e as encaminha para a estação base. Aqui, somente os *Cluster Heads* participam do caminho até a estação base, os nós comuns só se preocupam em entregar os dados aos seus respectivos *Cluster Heads*.

Roteamento Baseado em Localização: aqui as posições dos nós são conhecidas e aproveitadas para a decisão do roteamento dos dados entre os nós da rede.

Ainda segundo (AL-KARAKI; KAMAL, 2004), os algoritmos de roteamento para RSSF também podem ser divididos de acordo com o tipo de operação do protocolo em 5 principais tipos:

Baseado em Negociação : Utilizam descrições de alto nível para os dados, desta forma, podem eliminar redundâncias nos dados transmitidos através de negociação. A principal

motivação é evitar o envio de informações redundantes ou duplicadas para o próximo nó ou para a Estação Base, através da troca de mensagens de negociação.

Baseado em Múltiplas Rotas : Utilizam mais de um caminho da fonte ao destino, de forma a aumentar a performance da rede.

Baseado em Consulta : Os nós não enviam dados de maneira contínua - apenas em resposta a consultas de outros nós.

Baseado em QoS : nesse tipo de algoritmo deve haver um equilíbrio entre consumo de energia e a qualidade dos dados - satisfazendo métricas definidas de QoS.

Baseado em Coerência : No processamento de dados não-coerente, os nós enviam todos os dados crus para os agregadores, que realizam um processamento sobre os mesmos e decidem se os dados deve ser encaminhados à estação base. Já no processamento coerente, os nós realizam um mínimo de processamento antes de enviar os dados aos agregadores, como remoção de duplicados ou acréscimo de *timestamps*.

Um protocolo de roteamento é considerado adaptativo se certos parâmetros podem ser controlados de maneira a adaptar-se às condições atuais da rede ou aos níveis de energia disponíveis. Além das classificações acima, um protocolo ainda pode ser dividido entre pró-ativo, reativo ou híbrido, dependendo de como um nó fonte encontra uma rota para a EB. Nos protocolos pró-ativos, todas as rotas são computadas antes de serem necessárias, enquanto no reativo, as rotas são calculadas sob demanda. Nos híbridos, uma combinação destas duas ideias é utilizada. Quando os nós são estáticos, é preferível utilizar o roteamento baseado em tabelas ao invés de protocolos reativos. Uma grande quantidade de energia é usada na descoberta de rotas e configuração de protocolos reativos. Uma outra categoria de protocolos, são os chamados cooperativos, onde os nós enviam dados para um nó central onde esses dados são agregados e podem estar sujeitos a um processamento futuro, assim reduzindo os custos de roteamento em termos de utilização de energia (AL-KARAKI; KAMAL, 2004).

A seguir, temos os principais desafios no projeto de algoritmos de roteamento em RSSF(RAGHUNANDAN; LAKSHMI, 2011; AL-KARAKI; KAMAL, 2004):

Distribuição dos nós: A distribuição dos nós em RSSF depende da aplicação e afeta diretamente a performance do protocolo de roteamento. Os nós podem ser distribuídos de forma determinística, onde os sensores são posicionados em locais específicos e os dados são roteados por caminhos pré-determinados, ou podem ser distribuídos de forma aleatória, criando uma infraestrutura *ad-hoc* (como por exemplo, nós sendo jogados de um avião sobre uma floresta).

Consumo de energia sem perder a precisão: Os sensores são completamente dependentes da bateria, logo o processamento e transmissão de informações precisam ser utilizados de maneira a conservar energia ao máximo.

Modelo de envio de dados: O levantamento e envio de dados é dependente da aplicação e também depende da urgência no recebimento dos dados. O envio pode ser contínuo, baseado em eventos, baseado em solicitação ou um híbrido entre estes. O protocolo de roteamento é altamente influenciado pelo modelo de envio de dados em relação ao consumo de energia e roteamento estável.

Tolerância a falhas: Sensores podem falhar - por conta do esgotamento de energia, quebra, ou falhas causadas por algum agente externo. O protocolo de roteamento deve prever a queda de nós, o que faz com que sejam necessários o estabelecimento de novos caminhos de roteamento dinamicamente - ajustando a potência de transmissão ou requerendo o envio de sinalizações. São necessários vários níveis de tolerância a falhas em um protocolo para que tudo funcione bem - mesmo nos piores cenários possíveis.

Dinâmica da Rede: Em muitos estudos, considera-se que os nós são estáticos, ou seja, fixos. Em alguns casos porém, tanto os nós como a estação base, podem movimentar-se. Enviar e receber dados destes nós moveis pode ser bem desafiante - devido às restrições de potência de transmissão ou de energia, entre outros.

Conectividade: Uma alta densidade nos nós impede que que eles fiquem completamente isolados um do outro. Portanto, espera-se que em uma RSSF, os nós sejam altamente conectados. Isto, no entanto não previne que a topologia da rede venha a sofrer alterações ou a possibilidade do tamanho da rede ir diminuindo devido à falhas em nós. Adicionalmente, a conectividade depende da distribuição - possivelmente aleatória, dos nós.

Mídia de Transmissão: Em uma RSSF os nós são ligados através de redes sem fio. Logo, os problemas tradicionais associados com um canal de redes sem fio (alcance, taxa de erros), também podem afetar o funcionamento das RSSF. Geralmente RSSF requerem uma baixa largura de banda, da ordem de 1-100kb/s.

Qualidade de Serviço: Em algumas aplicações, os dados precisam ser entregues até um certo período depois de terem sido coletados, caso contrário, tornam-se desnecessários. Logo, uma latência limitada é uma outra condição para aplicações com restrição de tempo. Em alguns casos, conforme a energia vai se esgotando, a rede pode requerer uma diminuição na qualidade dos dados, de maneira a reduzir o consumo de energia e manter um maior tempo de vida. Neste caso, protocolos de roteamento dependentes do nível de energia são requeridos.

Ambiente de Operação: As RSSF podem ser montadas nos mais diversos locais, como dentro de uma casa, no fundo do oceano, em um ambiente contaminado quimicamente etc.

Custos de Produção: Como uma RSSF, em geral, consiste em uma grande quantidade de sensores, o preço de um sensor individual deve ser baixo, para que o custo total de uma RSSF justifique sua utilização.

Heterogeneidade dos nós/links: Em vários estudos, os nós de uma RSSF são considerados homogêneos. Entretanto, dependendo da aplicação, nós podem ter diferentes capacidades ou desempenhar diferentes papéis. Essa heterogeneidade traz complicações aos

algoritmos de roteamento. Imagine um ambiente onde é necessário um misto de sensores para detectar umidade, temperatura e pressão, por exemplo, cada tipo de sensor pode ter uma taxa de leitura e envio de dados diferente. Eles podem inclusive, seguir múltiplos modelos de envio de dados (um baseado em eventos, outro baseado em solicitação).

Escalabilidade: O número de sensores em uma área a ser monitorada pode ser de centenas a milhares de nós. O esquema de roteamento deve ser capaz de lhe dar com este grande número de sensores.

Cobertura: cada sensor de uma RSSF obtém uma certa visão do ambiente. A visão deste sensor é limitada tanto em alcance como em precisão. Logo a cobertura de uma área por sensores também é um fator importante no design de uma RSSF.

Agregação de dados: Em alguns casos, os nós sensores podem gerar uma grande quantidade de informações redundantes. Logo, pacotes com dados similares de múltiplos nós podem ser agregados para reduzir o número de transmissões. A agregação de dados é a combinação de dados de diferentes fontes, de acordo com certa função de agregação (supressão de dados duplicados, mínimo, máximo e média etc.).

2.4 Trabalhos Relacionados

Em (WANG et al., 2003) foi proposto o *Coverage Configuration Protocol* (CCP), um algoritmo que garante o grau de cobertura requerido pela aplicação - desde que o raio de comunicação seja maior ou igual a duas vezes o raio de sensoreamento ($R_c \geq 2R_s$). O algoritmo é completamente descentralizado, o que garante uma boa escalabilidade. No CCP, cada nó executa um algoritmo de elegibilidade, onde o mesmo determina se é necessário estar no estado ativo. Dado um determinado grau de cobertura - requerido pela aplicação, K_s, um nó v é inelegível se cada ponto em sua área de cobertura já está K_s-coberto por outros nós ativos na rede. O algoritmo de elegibilidade requer que os nós conheçam sua localização. Baseado em sua elegibilidade, um nó pode estar nos estados ATIVO, DORMINDO e ESCUTANDO. No estado DORMINDO, o no é desativado, para conservar energia. No estado ATIVO o nó está realizando as tarefas de sensoreamento normalmente e eventualmente um nó ATIVO entra no estado ESCUTANDO para coletar mensagens *HELLO* de seus vizinhos e recalcular sua elegibilidade. No início da rede todos os nós estão ativos. Se um ponto possui uma cobertura maior que a requerida, os nós redundantes mudarão para o estado DORMINDO executando o algoritmo de elegibilidade - e descobrindo-se inelegíveis. Os nós podem entrar no estado DORMINDO até que a cobertura requerida seja atingida, quando mais nenhum nó poderá passar a este estado. Os resultados mostraram a eficiência, a capacidade de auto-configuração e a conectividade do algoritmo.

Em (MENG et al., 2010) é apresentado o *Energy-Efficient and Coverage-Specific Node Scheduling for Wireless Sensor Networks* (ECNS). O ECNS utiliza um algoritmo de escalonamento distribuído visando controlar a densidade de nós ativos - sem necessitar da localização dos nós. Cada sensor percebe o número de vizinhos ativos e calcula a seu grau de redundância esperado - de acordo com um estudo analítico apresentado no artigo. Após uma

comparação para verificar a redundância na vizinhança, um nó pode ser desativado - para economizar energia. Os resultados demonstram um significante aumento no tempo de vida da rede e o alcance da cobertura requerida, além de uniformizar o consumo de energia entre os nós com uma baixa sobrecarga.

Em (MISRA; KUMAR; OBAIDAT, 2011) um algoritmo de cobertura para uma rede hierárquica é proposto. Neste trabalho, assume-se que os *Cluster Heads* são nós com maior capacidade de processamento, energia e comunicação. Os nós associados à um *Cluster Head* são divididos em conjuntos e só um conjunto fica ativo ao mesmo tempo. É o *Cluster Head* que calcula as distâncias dos nós e atribui a que grupo cada nó irá pertencer - para isto a informação de localização dos nós é necessária. Estes conjuntos são escolhidos de forma que a ativação de todos os nós de cada um deles é suficiente para manter a cobertura da área monitorada. Os resultados mostram uma economia no consumo de recursos, embora tenha um impacto na qualidade da cobertura.

(OLIVEIRA, 2011) desenvolveu uma solução de escalonamento de atividades de nós baseada no algoritmo de roteamento desenvolvido por (RIBEIRO; CASTRO, 2010) - Bio4Sel. Bio4Sel é um algoritmo de roteamento plano para redes de sensores sem fio homogêneas baseado em Otimização por Colônias de Formigas (*Ant Colony Optimization*). Este tipo de algoritmo baseia-se em como colônias de formigas obtém rotas eficientes para fontes de comida. Na figura 2.6, temos um exemplo de como formigas lidam com este problema. Ao passar por um caminho, as formigas vão depositando feromônios no mesmo. No menor caminho, as formigas conseguem ir e voltar mais rapidamente, tornando maior a quantidade de feromônio depositada naquele caminho. A próxima formiga eventualmente escolhe o caminho com mais feromônio, assim, após um certo período de tempo, o melhor caminho terá uma probabilidade muito maior de ser escolhido, devido à quantidade de feromônio depositada no mesmo.

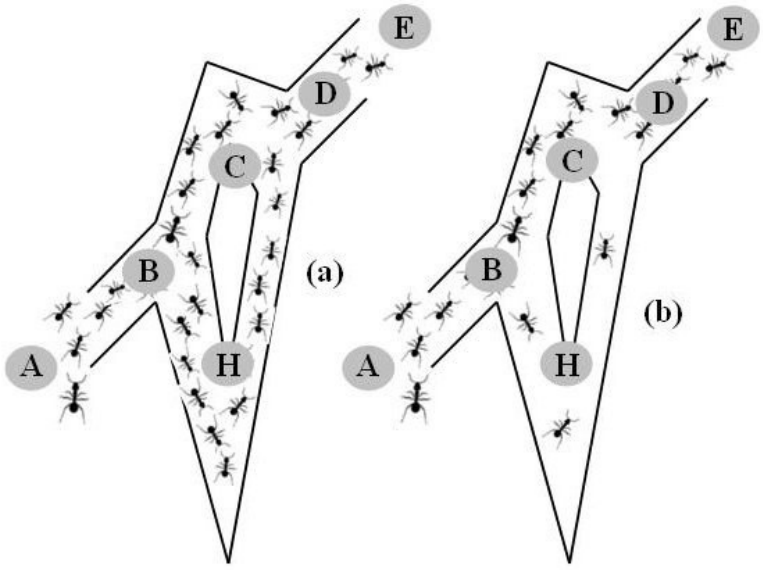

Figura 2.6: Após um certo período de tempo, formigas encontram o menor caminho de sua colônia - do ponto *A*, até uma fonte de comida - no ponto *E*. (DOMINGUEZ; CRUZ-CORTÉS, 2011)

Os resultados apontam que Bio4Sel garante um bom tempo de vida da rede. Para inicialização da rede, pacotes *formigas* são enviados, para detectar caminhos e o estado energético da rede. Junto com o envio dos dados através da rede, *formigas* continuam a ser enviadas *de carona* nos pacotes de dados, visando manter o mapa energético da rede sempre atualizado. Como BiO4SeL é focado em manter o tempo de vida da rede, um caminho muito longo pode ser escolhido. Os resultados mostraram grande escalabilidade do algoritmo e melhor resultados em cenários onde poucos nós produzem dados. O trabalho de (OLIVEIRA, 2011) visava melhorar ainda mais a eficiência energética de Bio4Sel, através de técnicas de gerência autonômica da cobertura de rede através do escalonamento de atividade dos nós. O principal diferencial do trabalho é que, ao invés de preocupar-se em desativar a maior quantidade de nós possível, a solução proposta preocupa-se em poupar nós bastante utilizados - forçando a participação de outros nós no roteamento e distribuindo melhor o consumo energético entre os nós. São propostas duas versões da solução, uma na qual são desativados os nós com pouca energia e que são muito utilizados no roteamento de pacote de dados e a outra estende a primeira levando em conta no processo de escalonamento dos nós, aqueles que não estão sendo utilizados no roteamento dos dados. A ideia da solução é atribuir 3 estados aos nós: *acordado*, *dormindo* e *indeciso*. Após uma certa quantidade de pacotes enviados, um nó ativo passa ao estado de *indeciso*, de onde pode voltar ao estado de *acordado* ou passar ao estado *dormindo* caso satisfaça certas condições: possuir mais de 10 vizinhos e possuir ao menos um nó que possa substituí-lo - um nó pode substituí-lo se satisfizer algumas condições especificadas no trabalho.

Em (LEE; LEE; LEE, 2013) os autores propõe o algoritmo *JENGA-INSPIRED OPTIMIZATION ALGORITHM (JOA)*. *JOA* é baseado em um jogo conhecido como *JENGA*, um jogo de tabuleiro onde os jogadores se alternam removendo blocos de uma torre de blocos, e os recolocando no alto da mesma torre, aumentando o tamanho e tornando-a mais instável, quando a torre cai, o jogo acaba. O algoritmo proposto é divido em um estágio de preparação e inicialização, seguido pelo estágio de execução propriamente dito. *JOA* baseia-se em um modelo de cobertura por pontos, onde cada ponto a ser coberto é chamado de *PoI - Point of Interest* e supõe que os sensores conhecem sua localização, além de dividir o tempo de execução em *time slots* - intervalos de tempo. No primeiro estágio a localização dos sensores e de cada *PoI* são armazenadas em suas respectivas matrizes, onde também é salvo o nível de energia dos sensores. Ainda nesta fase, a quantidade de *PoIs* que pode ser alcançada por cada sensor é armazenada em um vetor. Em seguida, ajustam-se os parâmetros - o algoritmo tem como parâmetros a quantidade de jogadores e número de rodadas - N_T, por exemplo. Após estes passos, o jogo começa, iniciando-se um *time slot*. Dentro de um *time slot*, cada jogador irá *jogar* N_T vezes. Em cada jogada, um jogador remove sensores do jogo - um por vez, de maneira aleatória. Remover um sensor, significa que aquele sensor será desativado. A cada retirada, é verificado se os *PoIs* continuam cobertos, caso contrário, o jogador passa a vez ao próximo. Após o jogador concluir sua jogada, os pontos dos jogadores são atualizados e o custo daquela cobertura - os sensores que restaram no jogo daquele jogador, é calculado, sendo melhor quanto menos energia for gasta para cobrir todos os *PoIs*. Após os N_T turnos - ou seja, ao final de um *time slot*, o algoritmo pega a melhor cobertura resultante do jogo. E assim sucede-se, sendo que a quantidade de energia dos sensores é atualizada ao fim de cada rodada. *JOA* utiliza um modelo de detecção probabilístico. Um exemplo do resultado de uma rodada pode ser visto na figura 2.7. Os resultados

mostraram que *JOA* conseguiu prolongar o tempo de vida da rede, em comparação com outros algoritmos de otimização.

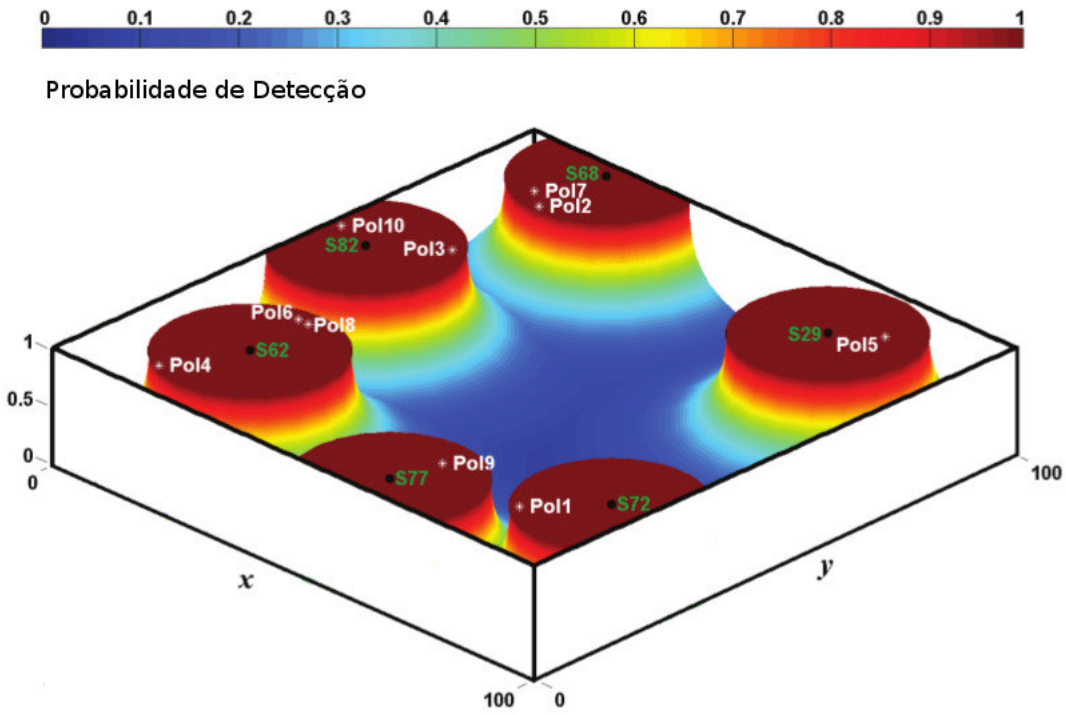

Figura 2.7: Exemplo de cobertura de PoIs em (LEE; LEE; LEE, 2013) em um cenário com 100 sensores e 10 PoIs.

Em (GUPTA; RAO; VENKATESH, 2013) os autores descrevem um algoritmo para o escalonamento de nós para sensores heterogêneos. É proposto um modelo de cobertura probabilístico, que é utilizado pelo protocolo de escalonamento. Inicialmente a Estação Base calcula uma tabela de redundância de cada tipo de sensor - utilizando o modelo probabilístico proposto, que é posteriormente enviada aos nós, ou a tabela pode ser armazenada em cada nó, sendo calculada antes da deposição da RSSF. O protocolo de escalonamento é dividido em rodadas e cada rodada possui duas fases: decisão e sensoreamento. Cada nó também possui um estado, que pode ser: ativo - o nó está sensoreando e comunicando-se com os vizinhos, aguardando - quando um nó é considerado redundante e irá decidir se permanecerá ativo ou irá ser desativado, e o estado desativado - quando um nó é desativado para economizar energia. Durante o processo de decisão, cada sensor utiliza uma tabela de vizinhos e, de acordo com o grau de cobertura requerido, decide se é ou não redundante. Durante a fase de sensoreamento, os nós que possuem redundância podem ser desativados até o final daquela rodada. Os sensores redundantes decidem quais ficarão ativos e quais serão desativados naquela rodada. No início de cada rodada, cada nó envia uma mensagem *Hello* contendo seu *ID*, tipo e estado atual. Cada nó monta, então, a tabela de vizinhos. A tabela 2.1 mostra um exemplo de uma tabela de vizinhos. As informações da tabela de vizinhos são atualizadas periodicamente através de mensagens *Hello*.

Utilizando as informações da tabela de vizinhos, cada nó pode então verificar na

ID	Tipo	Estado
1	Tipo 1	Ativo
2	Tipo 2	Aguardando
3	Tipo 2	Aguardando
4	Tipo 1	Ativo
5	Tipo 1	Ativo

Tabela 2.1: Tabela de vizinhos (GUPTA; RAO; VENKATESH, 2013)

tabela de redundância - dados a quantidade e o tipo de vizinhos, e descobrir se é ou não considerado um nó redundante. Se um nó for considerado redundante, ele altera seu estado para aguardando, caso contrário permanece no estado ativo. Cada nó redundante, então, aguarda um tempo aleatório escutando os vizinhos. O tempo aleatório é para garantir que mais de um nó não tente dormir ao mesmo tempo, o que poderia causar uma falha na cobertura naquele local. Se ao final deste tempo, o nó ainda tiver número suficiente de nós ativos para garantir sua redundância, ele então envia uma mensagem *Sleep* para uma quantidade de vizinhos - suficiente para garantir sua redundância, e passa para o estado inativo. Se um sensor receber uma mensagem de *Sleep*, ele passa imediatamente para o estado de ativo, uma vez que ele está cobrindo uma área de um nó que ficará inativo. Os resultados demonstraram que o algoritmo proposto conseguiu manter a taxa de cobertura requerida durante todo o tempo, além de prolongar o tempo de vida da rede por utilizar menos nós para esta tarefa. Também demonstrou-se que a quantidade de sensores necessários para manter o grau de cobertura requerida foi baixo.

3 SOLUÇÃO PROPOSTA

3.1 Escalonamento por Inversão de Arestas

O algoritmo de Escalonamento por Inversão de Arestas, proposto por (BARBOSA; GAFNI, 1989), consiste em um mecanismo de controle de concorrência de acesso a recursos compartilhados em sistemas distribuídos. Vejamos inicialmente as definições de *Sistemas Distribuídos* e de *recurso*, segundo (COULOURIS; DOLLIMORE; KINDBERG, 2007):

> Definimos um Sistema Distribuído como sendo aquele no qual os componentes de hardware e software, localizados em computadores interligados em rede, se comunicam e coordenam suas ações apenas enviando mensagens entre si.
>
> O termo "recurso"é bastante abstrato, mas caracteriza bem o conjunto de coisas que podem ser compartilhadas de maneira útil em um sistema de computadores interligados em rede.

Chamaremos aqui de *processo*, um sistema formado por *hardware* e *software* em um Sistema Distribuído. Consideraremos a situação em que um processo deve possuir acesso único a um recurso compartilhado - que pode ser um arquivo ou uma variável em memória, por exemplo, por um determinado período de tempo, passando a vez a outro processo ao final deste período. Eventualmente, todos os processos devem conseguir acessar o recurso – característica conhecida como ausência de *starvation*, e nenhum processo deve ficar esperando por um recurso bloqueado por outro processo que não será liberado - característica conhecida como ausência de *deadlock*. Na figura 3.1 temos um exemplo de *deadlock*.

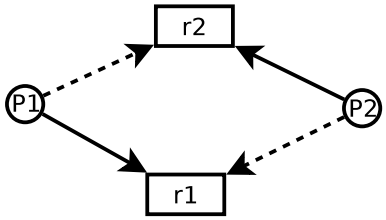

Figura 3.1: Exemplo de *Deadlock*. As setas normais representam a posse de um recursos, enquanto as setas tracejadas representam um recurso necessário que está sendo aguardado. O Processo P1 detém o recurso r1 e está bloqueado aguardando pelo recurso r2. Por sua vez, o processo P2 detém o recurso r2 e está bloqueado aguardando pelo recurso r1. Nenhum dos dois processos irá executar.

Uma das clássicas abordagens utilizadas para ilustrar problemas em concorrência é a discussão do problema do Jantar dos Filósofos, formulado por Edsjer Dijkstra. Este problema segue descrito abaixo e é ilustrado na figura 3.2.

> Cinco filósofos sentam em silêncio em uma mesa redonda, cada um com um prato de espaguete. Um garfo é colocado entre cada par de filósofos vizinhos (uma formulação alternativa utiliza arroz e pauzinhos, ao invés de espaguete e garfos).

Cada filósofo pode alternar entre pensar e comer. Entretanto, um filósofo só pode comer o espaguete quando ele possuir ambos os garfos - direito e esquerdo. Cada garfo só pode estar em poder de um filósofo, por isso um filósofo só pode usar um garfo se ele não estiver sendo utilizado por outro filósofo. Depois de terminar de comer, um filósofo deve por de volta o garfo na mesa, assim outro filósofo poderá utilizá-lo. Um filósofo pode pegar um garfo em sua mão direita ou esquerda assim que estiverem disponíveis, mas não pode começar a comer enquanto não tiver os dois garfos.

A comida não é limitada pela quantidade de espaguete: assuma uma quantidade infinita de espaguete.

O problema é como criar um *comportamento de disciplina* (um algoritmo concorrente), tal que cada filósofo não morrerá de fome - ou seja, ele pode sempre alterar entre comer e pensar e assumindo que nenhum filósofo pode saber quando os outros irão pensar ou comer.

(WIKIPEDIA, 2013)

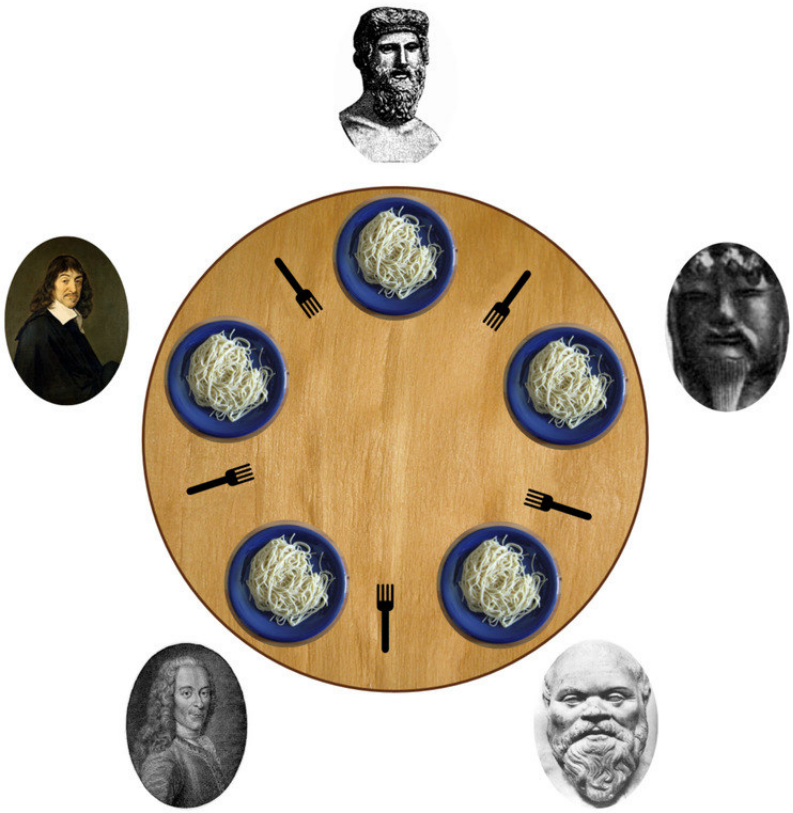

Figura 3.2: Problema do Jantar dos Filósofos. (WIKIPEDIA, 2013)

A solução de (BARBOSA; GAFNI, 1989), propõe modelar os processos que compartilham recursos como os nós de um grafo direcionado G. Os processos que compartilham recursos possuem uma aresta entre si. O algoritmo pode ser descrito como:

> Dado um grafo orientado G, inicialmente oriente G por uma orientação acíclica – que pode ser obtida facilmente se os nós possuem identificadores únicos, basta que,

> em cada nó, as arestas apontem para os nós que possuem identificadores de valor maior que seu próprio identificador - nós que são sorvedouros nesta primeira orientação tem o direito de operar inicialmente. Após a execução, apenas os nós que eram sorvedouros invertem suas arestas, assim, os nós que passarem a ser sorvedouros nesta nova orientação – que também é acíclica, ganharão o direito de operar. As inversões sucedem-se de maneira que todos os nós terão sua chance de obter o recurso compartilhado. (BARBOSA; GAFNI, 1989).

A figura 3.3 mostra um exemplo da execução do algoritmo de Escalonamento por Inversão de Arestas.

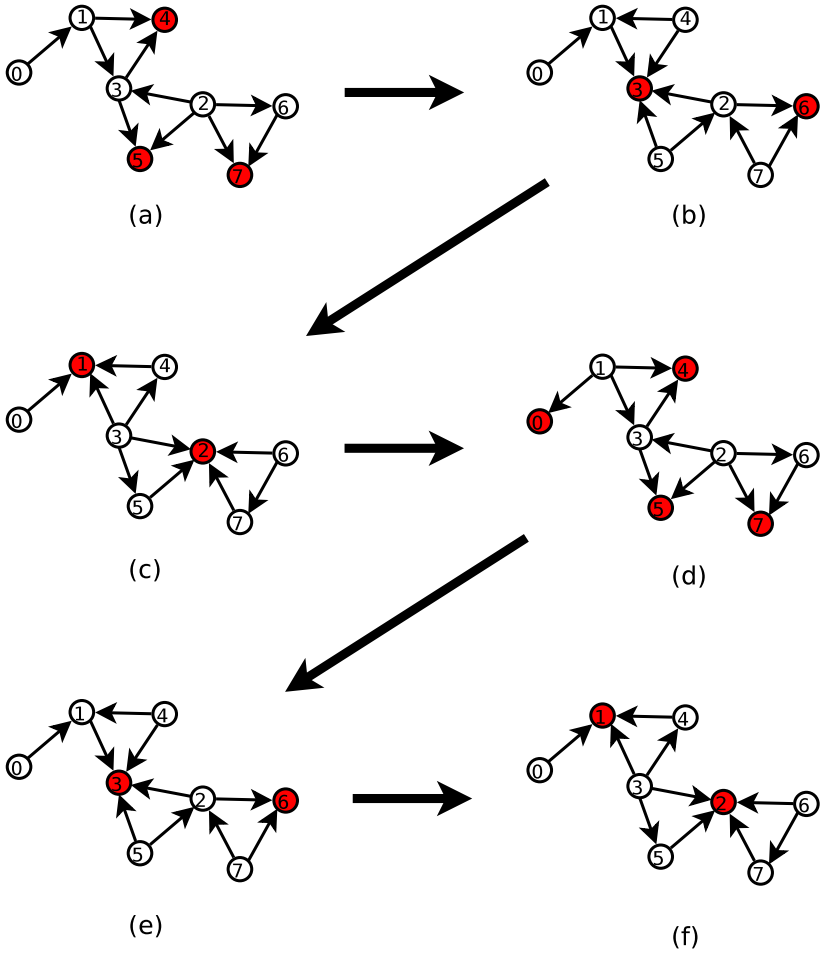

Figura 3.3: Exemplo do algoritmo Escalonamento por Inversão de Arestas. Em (a), temos a orientação inicial - cada nó aponta suas arestas para os nós que possuem maior ID. São escalonados inicialmente para executar os nós 4, 5 e 7. Em (b), os nós escalonados na rodada passada invertem suas arestas, liberando para utilizar o recurso compartilhado os nós 3 e 6. A sequência segue em (c) e em (d) temos a primeira repetição do ciclo - note que com exceção da aresta entre os nós 1 e 0, todas as outras arestas estão no mesmo estado que na posição inicial. A partir deste momento, fecha-se um ciclo, e as próximas iterações resultam em estados consecutivos iguais a estados anteriores, como podemos ver em (e) - que possui o mesmo estado de (b), e (f), que possui o mesmo estado de (c).

3.2 ESquema de COBErtura Distribuído para escalonamentO do rádiO de sensores em uma RSSF - ESCOBEDOO

3.2.0.1 Motivação

A principal motivação deste trabalho foi a utilização do algoritmo Escalonamento por Inversão de Arestas (EIA) para escalonar o rádio de nós em uma RSSF. Manter o rádio ativo pode consumir mais energia que uma transmissão - caso do rádio CC2420, da *Texas Instruments*, bastante utilizado em RSSF (TEXAS, 2013). A ideia, então, era aproveitar a simplicidade do algoritmo EIA e aplicá-lo ao domínio de RSSF para o escalonamento do rádio de um sensor em uma RSSF. O fato de desligar o rádio entre as iterações do EIA traz um enorme desafio, pois os nós precisam manter algum tipo de sincronismo, uma vez que, ao enviar uma mensagem a um determinado nó - e o algoritmo utiliza-se de mensagens para a coordenação de sua execução, é preciso que aquele nó esteja com o rádio ligado naquele momento, ou não receberá a mensagem, causando um operação incorreta.

3.2.1 Inicialização da Rede

Em grande parte da literatura, não se mostra nenhuma preocupação com a descoberta inicial de vizinhos, necessária por boa parte dos algoritmos propostos. Nesta descoberta inicial, em geral os nós enviam uma mensagem de *broadcast* - chamada aqui de mensagem *Hello*, e escutam a rede para obter as mensagens *Hello* de seus vizinhos. Dependendo da aplicação executada - e de como os nós são depositados na área, geralmente os nós iniciam-se quase no mesmo momento, logo esta estratégia de envio de mensagens *Hello* por todos os nós pode causar congestionamento ou perda de dados nesta fase inicial, fazendo com que um nó possa não ter informação sobre um vizinho - que nunca seria identificado, por exemplo. Grande parte das simulações nas soluções propostas na literatura consideram um ambiente ideal, sem colisões, ou simplesmente delegam este papel completamente ao Protocolo de Acesso ao Meio (MAC) utilizado. Devido às restrições energéticas, quanto mais simples forem os protocolos, melhor o consumo energético, além do fato de que as reduções de colisões economizam energia por evitar retransmissões. Logo, caso se utilize um protocolo MAC mais elaborado, o consumo de energia por conta da maior complexidade deste MAC poderá ser maior - embora isto dependa também de vários outros fatores, ou ainda, o MAC pode ser complicado demais para rodar em sensores com capacidade muito limitada de processamento. Na solução aqui proposta, nós optamos por utilizar uma solução probabilística de (PAILLARD et al., 2004), onde, a cada iteração do algoritmo, um nó tem uma determinada probabilidade de enviar uma mensagem *Hello* - desta forma, reduz-se a probabilidade de ocorrerem colisões na inicialização da rede, tornando o protocolo mais apto a atuar em aplicações reais. A seguir temos o algoritmo de inicialização descrito em (PAILLARD et al., 2004) em pseudo-código.

Algoritmo 1: ExchangeID (PAILLARD et al., 2004)

```
1 begin
2 |  for i ← 1 to C(l)log(n)^2 do
3 |  |  With probability 1/log(n), each node u sends a message containing its
  |  |  own identity;
4 |  endfor
5 end
```

$C(l)$ é uma constante e n é a quantidade de nós da rede. O algoritmo garante que todos os nós enviam pelo menos uma mensagem de *Hello* quando o número de nós tende ao infinito. O algoritmo 2 é uma versão modificada de 1, onde foi feita uma pequena alteração para garantir que cada nó realmente envie uma mensagem de *Hello*, uma vez que, embora proposto para atuar em redes com uma grande quantidade de nós, é possível executá-lo em uma rede com uma quantidade de nós que não seja suficiente para garantir que todos enviem suas mensagens *Hello*. A alteração é simplesmente o acréscimo de uma verificação durante o laço principal - que marca em uma variável, se um nó enviou uma mensagem *Hello*. Uma outra pequena modificação foi feita logo após o término do algoritmo, que verifica a variável citada anteriormente, a fim de descobrir se o nó enviou sua mensagem durante o laço principal e, caso isto não tenha ocorrido, o envio da mensagem *Hello* é executado neste momento. Como boa parte dos nós envia suas mensagens durante o laço, a probabilidade de ocorrer uma colisão no final é bem pequena: o algoritmo 1 garante que quando o número de nós tende a infinito, todos os nós enviam pelo menos uma mensagem, logo, restarão poucos - ou nenhum, nós para enviar seu ID ao final - a prova formal pode ser encontrada em (PAILLARD et al., 2004). Para validar a alteração do algoritmo foram executadas simulações usando o valor da constante $C(l) = 1$ e $C(l) = numNodes$ para 200 e 300 nós. Cada simulação foi repetida 30 vezes. Com $C(l) = numNodes$ os resultados confirmam a suposição do algoritmo 1 e nenhum nó precisou enviar a mensagem *Hello* ao final. Os resultados para $C(l) = 1$ - que reduz significativamente o tamanho do laço do algoritmo, estão a seguir.

	200 nós	300 nós
Percentagem de envios de *Hello* durante o laço	97%	98%

Algoritmo 2: ExchangeIDMod

```
1  boolean packetSent = false;
2  begin
3      for i ← 1 to log(n)^2 do
4          if not packetSent then
5              With probability 1/log(n), send a message containing its own
               identity;
6              if message was sent then
7                  packetSent = true;
8              endif
9          endif
10     endfor
11     if not packetSent then
12         send a message containing its own identity;
13     endif
14 end
```

3.2.2 Estimativa de distância entre nós

Através desta mensagem *Hello* inicial, é feito uma estimativa da distância entre cada nó e seus vizinhos. Esta distância irá definir quando há sobreposição nas áreas sensoreadas entre dois nós - a quantidade de sobreposição aceitável para que um nó seja considerado vizinho dependente da aplicação e do tipo de sensor utilizado. É também através dessa distância que calculamos a potência de transmissão necessária para a comunicação entre vizinhos - uma vez que estes nós estão bem próximos, pode-se utilizar uma potência de transmissão menor neste tipo de comunicação, economizando energia. A estimativa de distância foi calculada utilizando o RSSI (*Received Signal Strength Inication*) do pacote recebido. O uso do RSSI como mecanismo de localização não é muito confiável (HEURTEFEUX; VALOIS, 2012; BENKIC et al., 2008). Porém, para pequenas distâncias - foco da utilização neste trabalho, é a forma mais simples de se obter uma estimativa, considerando-se que os nós não possuem um dispositivo específico de localização e a deposição dos nós é feita de forma aleatória. A fórmula 3.1 (HEURTEFEUX; VALOIS, 2012) foi utilizada para calcular a estimativa de distância.

$$log_{10}(Distancia) = \frac{P_r(D) - P_{r1}}{K} \tag{3.1}$$

$P_r(D)$ é a *força* do sinal recebido à distância D - aqui é onde é utilizado o RSSI. P_{r1} é a potência do sinal recebido a $1m$ de distância e K é o expoente de perda de sinal (*Path Loss Exponent*). K e P_{r1} são geralmente obtidos empiricamente. Para melhorar a precisão da medida, além de utilizar os dados do RSSI da mensagem *Hello*, também é analisado o RSSI da mensagem imediatamente seguinte.

3.2.3 Alteração do Escalonamento por Inversão de Arestas

Além da alteração na inicialização da rede, foi verificado um caso particular onde se pode melhorar a cobertura, fazendo-se uma pequena alteração no algoritmo de Escalonamento por Inversão de Arestas. Este caso particular ocorre quando um nó possui somente um vizinho, e este vizinho, por sua vez, possui mais de um vizinho. A figura 3.4 mostra um exemplo. Neste caso, o nó que tem somente um vizinho, poderia continuar ativo após terminar seu tempo ativo, uma vez que seu vizinho irá receber a aresta, mas não terá ainda as arestas de outros vizinhos, ou seja, o nó sendo desativado irá ocasionar o surgimento de um *buraco* na rede que poderia ser evitado, caso a aplicação necessite da maior cobertura possível.

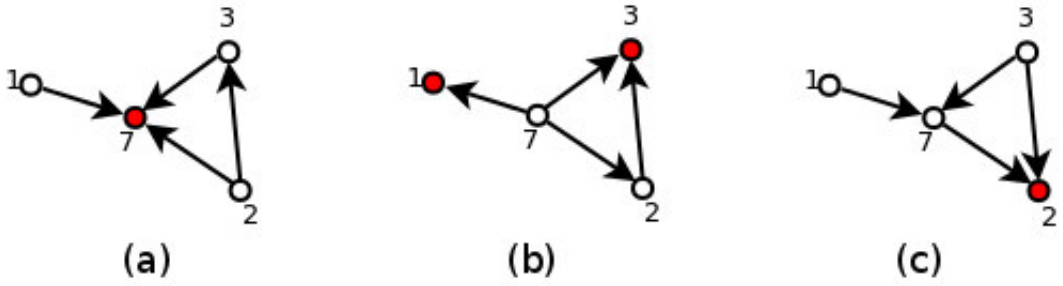

Figura 3.4: Caso dos *Nós Sozinhos*. Em (a), o nó 7 tem todas suas arestas apontadas para si, tornado-se ativo. Em (b) o nó 7 inverte as arestas, tornando ativos os nós 1 e 3. Em (c) os nós 1 e 3 invertem as arestas, ativando o nó 2, porém o nó 1 poderia continuar ativo, pois não há nenhum vizinho seu ativo - o nó 7 só se tornará ativo ao receber a aresta de 2, na próxima rodada. Esta situação pode causar um *buraco* na rede, pois a área monitorada em comum pelos nós 1 e 7 não terá nenhum nó ativo.

Para identificar estes casos, após a mensagem de *Hello*, um nó envia mais uma mensagem contendo sua lista de vizinhos. Esta mensagem também tem a função de melhorar a estimativa de distância baseada no RSSI, conforme citado na seção 3.2.2. Desta forma, um nó tem a informação de seus vizinhos, e de quais são os vizinhos destes, possuindo um mapa da rede a até 1 salto de distância. Assim ele pode identificar se é um nó sozinho (possui somente um vizinho - e este possui mais de um vizinho), ou se possui vizinhos sozinhos (tem mais de um vizinho e um - ou mais, deles não tem outros vizinhos).

Para melhorar a cobertura quando ocorre este caso, um nó sozinho não irá inverter suas arestas ao término da rodada. Ele continuará ativo até que o nó para o qual possui a aresta a tome dele. Desta forma, se um nó possui um vizinho nesta condição, assim que ele obtiver todas as arestas, com exceção da aresta do vizinho sozinho, ele enviará uma mensagem a este tomando a aresta e passando ao estado ativo. Assim, um nó sozinho nunca envia uma mensagem de inversão de arestas, apenas as recebe ou tem sua aresta tomada.

3.2.4 Sincronismo entre os nós

Como a solução proposta irá desativar o rádio em alguns nós por um certo período de tempo, foi necessário pensar em uma forma de sincronismo, de forma a evitar que um

determinado nó tente enviar uma mensagem para outro nó que esteja com o rádio desligado, ocasionando o não recebimento da mesma. Este problema foi resolvido através da utilização de temporizadores (*timers*). Quando um nó ativo inverte todas as suas arestas, antes de passar para o estado inativo, é iniciado um temporizador que dura o valor de uma rodada. Quando o temporizador expira, o nó começa a receber mensagens de outros nós. Após ele receber todas as mensagens daquela rodada, um novo temporizador é ativado, mas desta vez pelo tempo do tamanho de uma rodada menos o tempo que já passou desde o recebimento da primeira mensagem recebida nesta rodada - ou seja, o nó acordará ao mesmo tempo em que o primeiro nó que encerrou a rodada anterior acordar, impedindo que se perca qualquer mensagem que possa vir de um nó que está em um tempo diferente. Caso, por algum motivo, ocorra um atraso no envio de uma mensagem por algum nó, o atraso será propagado para todos os nós - exceto os que estão isolados, e todos estarão novamente em sincronia. A figura 3.5 ilustra esta situação.

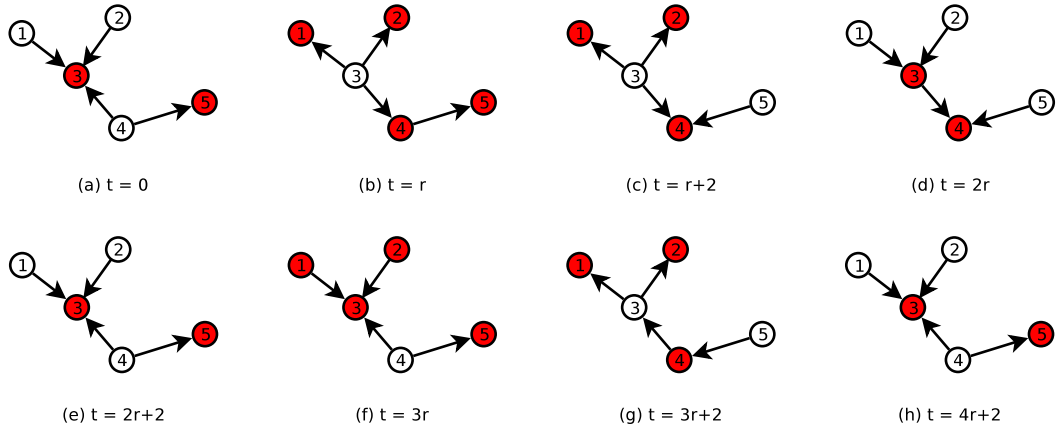

Figura 3.5: Sincronismo e a propagação de atrasos.

Na figura 3.5, *r* é o tempo de duração de uma rodada do algoritmo. Em (a) temos uma disposição em um determinado instante de tempo, por simplificação supomos que t=0. Estão ativos os nós 3 e 5. Em (b), ao finalizar a rodada, o nó 3 inverte sua aresta. Porém, por algum motivo, o nó 5 atrasa sua inversão. Neste caso, os nós 1 e 2 ficarão ativos durante o tempo de uma rodada e o nó 4 continua esperando a aresta de 5. Em (c) com 2 unidades de tempo de atraso, o nó 5 inverte sua aresta. Agora o tempo do nó 4 e do nó 5 ficam defasados em 2 unidades de tempo do resto dos nós. Em (d) os nós 1 e 2, que ainda estão com seus tempos sincronizados pelo tempo inicial invertem suas arestas. O nó 3 continua ativo aguardando a aresta de 4. Em (e) em seu novo tempo, o nó 4 inverte a aresta para 3. Agora 3 também estará com a defasagem de duas unidades de tempo em relação ao tempo inicial. Em (f) os nós 1 e 2 acordam para receber a inversão de 3, ainda no tempo inicial, porém o nó 3 só irá inverter suas arestas no tempo 3t+2, sincronizando finalmente os tempos dos nós 1 e 2. Todos os nós agora estão novamente sincronizados, como podemos ver em (g) e (h).

3.2.5 Energia Resultante e Inversão de Arestas

Assim como em outros trabalhos (OLIVEIRA, 2011), iremos utilizar o nível energético atual do nó e compará-lo com seus vizinhos, porém ao invés de determinar o tempo

que o nó permanecerá dormindo, utilizaremos esta estratégia para decidir se um nó irá ou não inverter suas arestas em uma determinada rodada. Este tipo de estratégia ajuda a balancear o consumo de energia dos nós, prolongando o tempo até que o primeiro nó esgote seus recursos energéticos. Embora pareca simples, a utilização desta estratégia também traz desafios, como o problema de descoberta do nível energético dos nós vizinhos. A solução utilizada aqui, realiza o envio de uma mensagem para os vizinhos - utilizando uma baixa potência, sempre que um nó decidir que não irá inverter suas arestas devido ao nível energético. Em todas as mensagens do algoritmo enviadas, um nó sempre envia - em um campo da mensagem, seu nível energético, de forma a manter este valor atualizado em todos os vizinhos. A fórmula 3.2 foi utilizada para decidir se um nó não irá inverter arestas por ter mais energia que os vizinhos.

$$T = max(Energia_{vizinhos}) \tag{3.2}$$

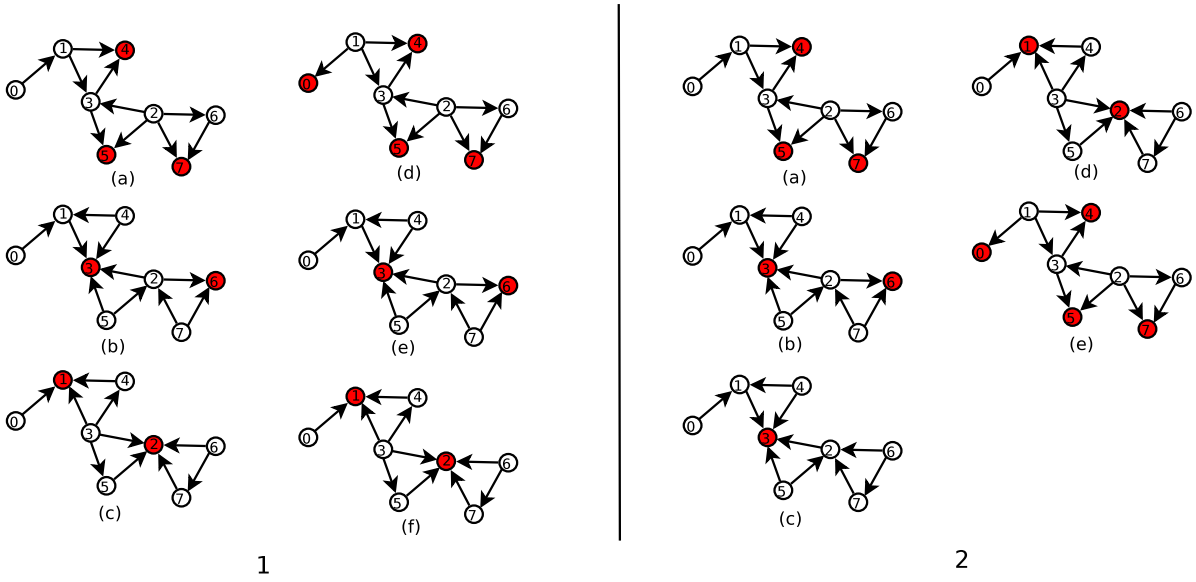

Figura 3.6: A utilização do nível de energia como estratégia de decisão antes da inversão de arestas pode afetar a cobertura da rede. Em 1 temos a inversão de arestas pura. A cada iteração, os nós ativos invertem suas arestas. Note que em $1-(d)$ ocorre a repetição do ciclo, que continua a ocorrer continuamente a cada 3 iterações. Já em 2 temos a inversão de arestas levando em conta que os nós podem não inverter suas arestas devido à comparação do nível de energia com os vizinhos. Em $2-(c)$ o nó 3 decidiu não inverter sua aresta por estar com mais energia que os vizinhos. Assim, durante aquela rodada ele foi o único nó a ficar ativo. A cobertura durante a iteração $2-(c)$ fica reduzida por conta da quebra do ciclo de inversões.

Se o nível de energia após o término da rodada que o nó ficou ativo, for superior ao valor de T, ou seja, ele é o nó que possui mais energia, dentre seus vizinhos, ele não irá inverter suas arestas, enviando uma mensagem aos vizinhos com esta informação. Ao receber uma mensagem desse tipo, um nó verifica se está esperando uma aresta daquele nó, caso afirmativo, ele deixa de esperar aquela aresta - e passa ao estado inativo, enviando uma mensagem aos vizinhos, ou continua esperando, caso ainda hajam arestas para receber naquela rodada. Ao

levar em conta a energia durante a inversão, sempre que um nó decidisse não inverter suas arestas, haveria um impacto na cobertura, uma vez que o nó irá quebrar o ciclo de inversão de arestas - a figura 3.6 mostra este caso em especial. Para resolver este problema, sempre que um nó percebe que todos os seus vizinhos irão dormir, ele passa para o estado ativo, de forma a evitar uma redução na cobertura naquela área. No caso da figura 3.6, em $2-(c)$ o nó 7 passaria ao estado ativo, ao receber a aresta de 6 e a informação que 5 e 2 continuariam dormindo.

3.2.6 Lidando com a *morte* dos Nós

Para que o algoritmo continue funcionando após a morte de um nó, foi implementada uma técnica utilizando temporizadores - *timers*. Cada vez que um nó acorda, ele deve receber mensagens de seus vizinhos, invertendo as arestas ou informando se vão ser desativados ou continuar ativos. A cada mensagem que um nó recebe, ele atualiza em sua lista de vizinhos o tempo em que recebeu a última mensagem daquele nó. Assim, quando um nó acorda, um temporizador é disparado e aguarda por um tempo configurado - o tempo de uma rodada. Se este temporizador expirar, a lista de vizinhos será varrida e caso a última comunicação de algum deles tenha sido há mais tempo que o tempo de uma rodada, aquele nó é considerado *morto* e retirado da lista de vizinhos.

3.2.7 O Algoritmo EsCobeDoo

A solução proposta foi chamada de ESquema de COBErtura Distribuído para escalonamento do rádio de sensores em RSSF. A seguir temos o pseudo-código da solução proposta. Alguns detalhes foram omitidos por questão de espaço, e podem ser consultados diretamente na implementação.

Algoritmo 3: EsCobeDoo

```
   Envia a mensagem inicial
1  ExchangeIDMod();
2  sleep(1s) ;                                   aguarda mensagens
3  enviaVizinhos() ;                  Envia minha lista de vizinhos
4  sleep(1s) ;                                   aguarda mensagens
5  ajustaArestas() ;          Ajusta arestas para os nós de maior ID
6  verificaArestas() ;               Seta estado em ativo ou inativo
7  if estadoAtivo then
8  |  iniciaTimer(FimDeRodada : TempoDaRodada);
9  else
10 |  DesligaoRadio;
   |  Seta timer (seção 3.2.4)
11 |  iniciaTimer(FimTempoInativo : (TempoRestanteDaRodada));
12 endif
```

O algoritmo 3, é o algoritmo principal da solução. Os nós enviam seus IDs, e aguardam para receber os ID's de outros nós - preenchendo a lista de vizinhos. Em seguida

enviam esta lista de vizinhos e aguardam para receber as listas dos outros nós. Logo após, o nó faz o ajuste inicial das arestas, apontando-as para os nós de maior ID. A partir daí ele verifica se possui todas as arestas, ficando ativo até o fim da rodada, ou se deve passar ao estado inativo, caso não possua todas as arestas.

Algoritmo 4: temporizadorExpirado

```
1  begin
2      switch TemporizadorExpirado do
3          case FimDeRodada
4              if nao tenho mais energia que meus vizinhos then // seção 3.2.5
5                  enviaMsgInversaoDeArestas() ;          Inverte Arestas
6                  Desliga o Radio;
7                  iniciaTimer(FimTempoInativo : (TempoDaRodada));
8              else
                       envia mensagem informando que não vai inverter
9                  enviaMsgNaoInverte();
                   iniciaTimer FimDeRodada com tempo : TempoDaRodada;
10             endif
11         endsw
12         case FimTempoInativo
               Ativa o radio novamente e aguarda novas mensagens
13             LigaoRadio;
               Ativa o timer de verificação de vizinhos
14             iniciaTimer VerificaNos com tempo : TempoDaRodada;
15         endsw
16         case VerificaNos
               Algum vizinho não enviou mensagens durante uma
               rodada verifico qual e removo
17             procuraVizinhoDesativado
18         endsw
19     endsw
20 end
```

O algoritmo 4 é executado quando um dos temporizadores expira. Ao terminar uma rodada, o nó verifica o nível de energia em relação aos vizinhos e inverte as arestas e passa aos estado inativo, ou envia uma mensagem informando que vai continuar no estado ativo. Quando um nó termina uma rodada em que estava inativo, ele liga o rádio e aguarda mensagens dos vizinhos, iniciando também o temporizador que irá expirar apenas, caso faltem mensagens de algum nó ao fim da rodada - o que irá causar a remoção deste nó, considerando-o *morto*.

O algoritmo 5 atua quando um nó recebe uma mensagem, verificando o tipo de mensagem e executando os passos a tomar ao receber cada tipo de mensagem. A função verificaEstado() - linhas 24, 33 e 46, apenas verifica se, após as mensagens, o nó ficará ativo ou inativo e executa os passos correspondentes. Esta função está descrita no algoritmo 6.

Algoritmo 5: AvaliaMensagensRecebidas

```
1  Para cada mensagem m_i recebida :
2  switch m_i → tipoMensagem do
3      case HELLO
4          calcula distancia e adiciona na lista de vizinhos;
5      endsw
6      case VIZINHOS
7          adiciona os vizinhos na lista de vizinhos do no' que enviou;
8      endsw
9      case INVERSÃO
10         Inverte a aresta recebida;
11         if ainda faltam arestas then
12             if arestas restantes sao de no's sozinhos then
13                 enviaMsgTomaAresta(); estadoAtivo = true;
14             else
15                 if arestas restantes EstaoEm(pulaRodada) then
16                     estadoAtivo = false;
17                 else
18                     estadoAtivo = true ;          continuo esperando
19                 endif
20             endif
21         else
22             estadoAtivo = true;
23         endif
24         verificaEstado();
25     endsw
26     case NAOINVERTE
27         pulaRodada.inclui(msg_i → address);
28         if arestas restantes EstaoEm(pulaRodada) then
29             estadoAtivo = false;
30         else
31             estadoAtivo = true ;              continuo esperando
32         endif
33         verificaEstado();
34     endsw
35     case TOMAARESTA
36         inverteArestaRecebida; enviaMsgDormir(); Desliga o Radio;
37         iniciaTimer(FimTempoInativo : (TempoRestanteDaRodada));
38     endsw
39     case DORMIR
40         pulaRodada.inclui(msg_i → address);
41         if arestas restantes EstaoEm(pulaRodada) then
42             estadoAtivo = false;
43         else
44             estadoAtivo = true ;              continuo esperando
45         endif
46         verificaEstado();
47     endsw
48 endsw
```

Algoritmo 6: verificaEstado

```
1 if estadoAtivo then
2 |   iniciaTimer(FimDeRodada : TempoDaRodada)
3 |   pulaRodada = vazio
4 else
5 |   enviaMsgDormir() ;                    Avisa que vai dormir
6 |   Desliga o Radio;
7 |   iniciaTimer(FimTempoInativo : (TempoRestanteDaRodada))
8 endif
```

4 ANÁLISE DE RESULTADOS

4.1 Definições

Nesta seção são apresentadas todas os valores de variáveis e considerações do processo de simulação para a validação da solução.

4.1.1 Distância Considerada para Existência de Arestas entre Nós

A área de sobreposição de sensoreamento de dois sensores - a área que é monitorada por dois sensores ao mesmo tempo, depende somente da distância euclidiana entre os mesmos, se os raios forem iguais e constantes (MISRA; KUMAR; OBAIDAT, 2011). O cálculo das distâncias obtido experimentalmente em simulações, mostrou inconsistência a partir de uma distância considerada a partir de 8 metros - em condições reais o RSSI pode variar bastante, não sendo recomendado para estimativa de distâncias muito grandes (HEURTEFEUX; VALOIS, 2012; BENKIC et al., 2008) e o simulador foi projetado para tentar ser o mais realista possível, apresentando grande variação na medida do RSSI. Desta forma, a distância máxima entre dois nós para definir uma área mínima aceitável de sobreposição de sensoreamento, ou seja, a distância máxima para que seja criada uma aresta entre dois nós próximos foi definida com o valor de 6 metros. Ao definir o raio de sensoreamento de um sensor como, no máximo, a metade do alcance de comunicação, ao garantir a cobertura nesta situação, também estará garantida a conectividade (WANG et al., 2003). Desta forma, o raio de sensoreamento de cada sensor considerado neste trabalho foi de 12 metros, o que corresponde à metade da distância de comunicação considerada entre os nós - distância obtida experimentalmente no simulador - repetindo 30 vezes o experimento, a -3dbm, que resultou em uma média de aproximadamente 24,83 metros, sendo truncado para o valor de 24 metros, uma vez que nenhuma das medidas foi inferior a este valor. Dado este raio e a distância máxima definida, temos uma taxa de sobreposição mínima - de sensores considerados redundantes, de aproximadamente 62%, como mostra a figura 4.1.

4.1.2 Simulador

As simulações foram executadas no simulador Castalia(CASTALIA, 2013), versão 3.2. A escolha do simulador baseou-se, principalmente, nos seguintes fatos: é um simulador específico para RSSF, possui código fonte livre e, por fim, é amplamente utilizado pela comunidade acadêmica (PEDIADITAKIS; TSELISHCHEV; BOULIS, 2010). O Castalia foi desenvolvido sobre o *framework* de simulação de redes Omnet++ (VARGA, 2001; VARGA; HORNIG, 2008).

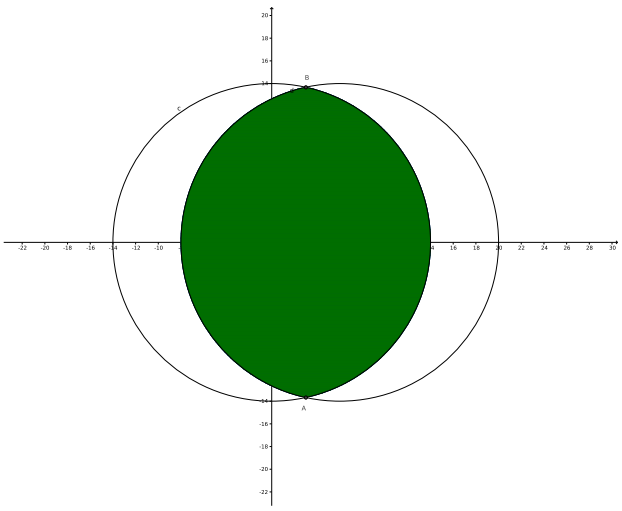

Figura 4.1: Área mínima de interseção considerada neste trabalho.

4.1.2.1 Rádio

O rádio utilizado no simulador foi o modelo CC2420 da empresa *Texas Instruments*. O modelo de consumo energético deste rádio no simulador foi desenvolvido baseado inteiramente no documento de especificações do rádio *(datasheet)* fornecido pelo fabricante (TEXAS, 2013). Neste modelo, há algumas diferenças entre muitos dos modelos encontrados na literatura, em especial o fato do consumo de energia pelo rádio durante o estado *RX* ser maior que o consumo no estado *TX* - diferença ainda maior quando se utiliza potências menores de transmissão. A tabela 4.1 mostra o consumo (em Joules) do rádio enquanto transmite. O arquivo de configuração do rádio CC2420 no simulador define o consumo do rádio no estado *RX* como sendo $62mW$, enquanto no estado *SLEEP*, o consumo definido é de $1.4mW$.

Tx_dBm	0	-1	-3	-5	-7	-10	-15	-25
Tx_mW	57,42	55,18	50,69	46,2	42,24	36,3	32,67	29,04

Tabela 4.1: Consumo de transmissão do rádio -em mW, para cada potência selecionada, utilizado pelo simulador para o rádio CC2420.

4.1.2.2 Quantidade de Energia

A quantidade de energia inicial em cada nó foi definida como 90 Joules. Este valor corresponde a, aproximadamente, 1% da energia disponível em uma pilha AA. Esta quantidade reduzida de energia inicial foi escolhida para reduzir o tempo de simulação necessário para que observássemos os efeitos causados por nós esgotando sua energia. Nos *Cluster Heads* a energia foi definida como o dobro deste valor - 180 Joules.

4.1.3 Arquitetura de Rede

A arquitetura de rede escolhida para a realização dos testes foi um modelo hierárquico, considerando *Cluster Heads* com uma maior capacidade energética. Esta arquitetura foi escolhida pela maior facilidade de implementação, além de ser possível separar o impacto do roteamento e do algoritmo de cobertura aqui proposto no consumo energético. Os *Cluster Heads* foram dispostos em forma de grade, em um total de 16 nós, dispostos em 4x4 na área da simulação. A área total da simulação foi definida como 100mx100m.

Os nós são distribuídos aleatoriamente - de maneira uniforme, e foram executadas simulações com diferentes quantidades de nós, de forma a verificar o impacto da densidade de sensores nos resultados - as simulações foram executadas com 100, 150, 200 e 250 nós.

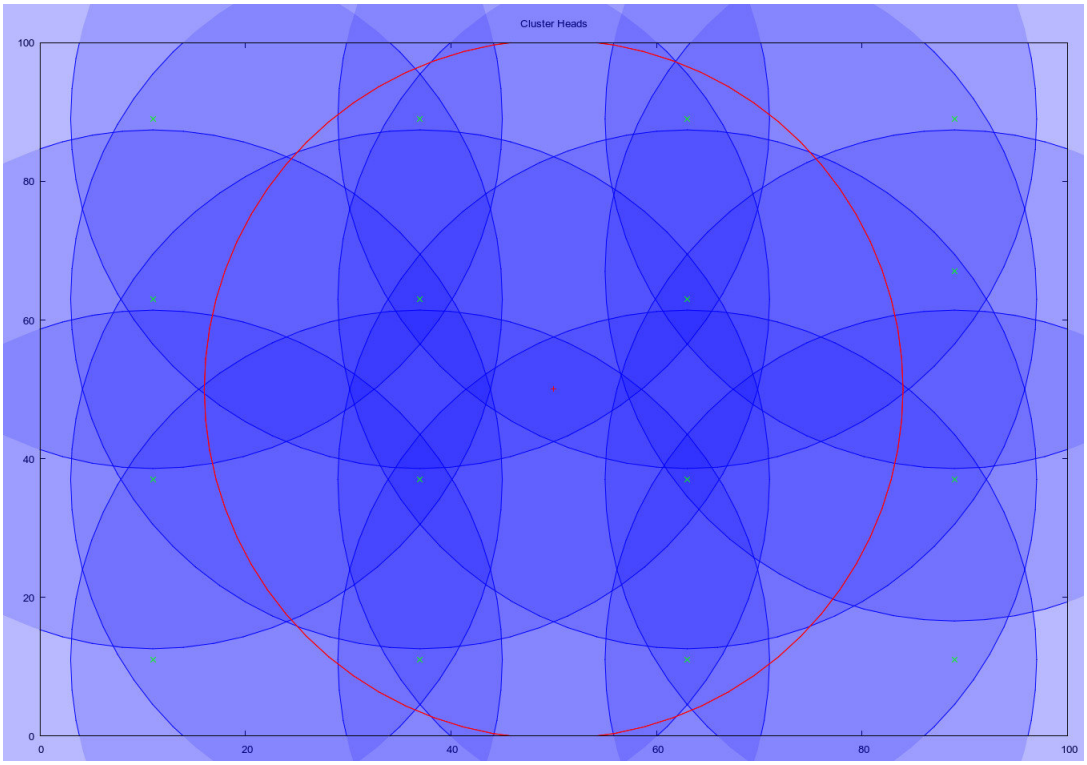

Figura 4.2: Disposição e alcance dos *Cluster Heads*.

4.1.4 Roteamento e Aplicação

De forma a validar a execução do algoritmo de escalonamento em uma simulação completa, o EsCobeDoo foi integrado a um algoritmo de roteamento estático, seguindo pelo menor caminho - em saltos, até a Estação Base - localizada no centro, o que nos permite saltar a fase de criação de rotas. Embora seja uma arquitetura bastante robusta, a utilização real de uma disposição deste tipo pode não estar disponível na maioria das aplicações, especialmente onde todos os nós são dispostos de forma aleatória. A figura 4.2 mostra a localização e o alcance dos *Cluster Heads*. Os pontos marcados com 'x' são os *Cluster Heads*, o ponto em cruz no centro é a Estação Base. Os círculos azuis são o alcance máximo dos *Cluster Heads* e o círculo vermelho o alcance da Estação Base.

Foi utilizada a aplicação *ThroughputTest* - disponibilizada pelo simulador. É uma aplicação simples, que gera pacotes a uma taxa constante - no caso da simulação foi utilizada a taxa de geração de 1 pacote por segundo em cada um dos nós.

4.1.5 Protocolo de Acesso ao Meio (MAC)

O MAC utilizado nas simulações foi o *Carrier sense multiple access with collision avoidance* (CSMA/CA) - implementação disponível no simulador.

4.1.6 Modelo de cobertura

O modelo de cobertura utilizado foi o modelo de discos booleanos, onde dada localização de um nó e um raio de sensoreamento r, um ponto x está coberto se pelo menos um sensor está a uma distância no máximo r do ponto x. Este modelo foi escolhido por ser amplamente utilizado na literatura e facilitar o cálculo da cobertura dos nós ativos.

4.1.7 Hipóteses

Como não foi implementado nenhum tipo de confiabilidade, durante as simulações utilizamos um modelo de rádio ideal, desconsiderando colisões ou qualquer variabilidade. Porém pacotes ainda podem ser perdidos caso um nó receba uma transmissão e ainda não esteja com o rádio no modo de recepção - após uma transmissão leva um pequeno tempo até que o nó faça a troca para o estado de recepção, por isto a estratégia utilizada na inicialização dos nós (3.2.1) é importante. Todas as simulações citadas foram repetidas 30 vezes - com exceção das citando explicitamente o contrário. O tempo máximo de simulação foi de 500s.

4.1.8 Casos Executados

Para comparar os resultados, foram criados os seguintes casos:

Roteamento Hierárquico - Todos os nós enviam para seus *Cluster Heads* todo o tempo da simulação. É o roteamento utilizado na solução proposta, porém sem nenhum controle de escalonamento de nós.

EsCobeDoo - Algoritmo proposto nesta dissertação.

RadioAtivo - É o algoritmo desta dissertação, porém sem a desativação do rádio dos nós. Desta forma, a economia de energia gerada nesta configuração é apenas referente ao não envio de dados quando um nó está em seu estado de inativo - ou a desativação apenas da interface de sensoreamento, o que não é suportado no simulador escolhido. Esta solução foi escolhida para mostrar como se comportaria a solução utilizando o modelo bastante difundido na literatura de que a transmissão consome uma quantidade expressivamente maior de energia do que a recepção.

ECNS - Energy-Efficient and Coverage-specific Node Scheduling for Wireless Sensor Networks ((MENG et al., 2010)). Este algoritmo foi escolhido por também ser um algoritmo distribuído e não depender da localização física dos sensores, tal qual a solução aqui proposta. Embora não dependa da localização dos sensores, em citeMENG10 não ficou claro se os autores implementaram o método de cálculo de distâncias descrito em (**??**) ou apenas apresentaram-no como uma solução passível de ser utilizada. Como o foco deste trabalho não é o algoritmo de localização, e o método utilizado aqui não foi efetivo para distâncias a partir de 8 metros - e o tamanho da área de sensoreamento foi definido como 12 metros, na implementação do ECNS foi necessário utilizar a localização física dos sensores, obtida diretamente do simulador.

4.2 Resultados

4.2.1 Definindo o tempo entre as inversões de arestas

Para definir o tempo que cada sensor permanece ativo até que inverta suas arestas, foram feitas simulações com diferentes intervalos de tempo entre as inversões. O intervalo inicial escolhido foi de $5s$, que corresponde a $\frac{1}{100}$ do tempo total simulação, em seguida simulações de $5s$ em $5s$, até o valor de $75s$, que corresponde a 15% do tempo máximo de simulação - este limite superior foi escolhido devido a permitir somente pouco mais de 6 iterações do algoritmo durante o tempo máximo de simulação. Os resultados estão na tabela 4.2.

Tempo	Média	Desvio Padrão
5	66,767	1,288
10	67,616	1,402
15	67,862	1,418
20	68,130	1,442
25	68,296	1,469
30	68,320	1,477
35	68,399	1,484
40	68,438	1,467
45	68,556	1,573
50	68,571	1,535
55	68,592	1,542
60	68,605	1,518
65	68,630	1,578
70	68,645	1,731
75	68,690	1,637

Tabela 4.2: Resultado da simulação inicial para analisar o tempo de duração de cada rodada do algoritmo. Foram utilizados 150 nós. A coluna Tempo mostra o tamanho de cada rodada, a coluna Média é a média da quantidade de energia residual - em Joules, dos nós ao fim da simulação (500s). E na última coluna temos o desvio padrão da média do nível de energia dos nós.

Analisando os dados da tabela 4.2, podemos observar que, embora a utilização de intervalos maiores entre as inversões nos mostre um valor maior na média da energia residual, estes valores devem-se ao fato de muitos nós sequer terem sido escalonados, portanto passaram quase toda a simulação em estado inativo, ou terem sido escalonados por poucas vezes. Podemos observar também, que a utilização de tempos entre inversões menores equilibram melhor o consumo de energia, uma vez que houveram várias iterações do algoritmo mas, em compensação, há um maior consumo energético. O menor intervalo entre inversões foi justamente o que mostrou uma menor quantidade de energia média residual, embora tenha sido o que melhor balanceou o consumo energético, por isto escolhemos este valor para a execução das simulações. Vale notar que, em um ambiente real, este valor de intervalo não seria o ideal, visto que a duração esperada da rede seria bem maior que o tempo de simulação aqui utilizado.

4.2.2 Cobertura da Rede

Como o algoritmo irá desativar nós, a cobertura da rede será reduzida em comparação com a não desativação de nenhum nó. Logo, este é um importante indicador para verificar a viabilidade da utilização da estratégia proposta. Calcular a área coberta por sensores em uma RSSF não é um trabalho fácil (AZIZ; AZIZ; ISMAIL, 2009; HUANG; TSENG, 2003), a obtenção destes resultados demandou bastante tempo durante a conclusão deste trabalho. A solução utilizada foi enviar para o arquivo de saída a localização e o estado do nó ao final de cada rodada do algoritmo, o que tornou-se complicado devido à quantidade de estados em que os nós poderiam estar em cada momento - o estado final do nó em uma determinada rodada, ativo ou inativo, só deveria ser enviado para o arquivo de saída quando ele concluísse a mesma, e ainda era necessário uma forma de padronizar esta saída, uma vez que cada nó termina sua rodada em tempos um pouco diferentes - o que dificulta a ação de escrever um *parser*. O caso dos nós sozinhos (seção 3.2.3), precisou de atenção especial, pois estes não marcam o fim de suas rodadas, uma vez que os mesmos ficam ativos até que suas arestas sejam tomadas. Após concluir o código que enviava para o arquivo de saída os estados dos nós a cada rodada, foi necessário o desenvolvimento de um *script* para separar as repetições e analisar uma a uma e rodada a rodada, quais nós estavam ativos e calcular a área coberta pelos sensores ativos naquele instante. Para este cálculo foi adaptado um *script*, escrito em linguagem *python*, obtido em (ROSETA, 2013) e transcrito no apêndice 1.

Na figura 4.3 temos o impacto da cobertura, em relação à medida obtida pela manutenção de todos os nós ativos.

Como podemos ver na figura 4.3, quanto mais densa for a rede, menor é o impacto na cobertura, chegando a bem próximo de 100% a partir de 200 nós. O algoritmo ECNS necessita receber como entrada a taxa de cobertura requerida - pois o mesmo garante ao menos a taxa de cobertura fornecida como entrada. Nas simulações do ECNS a seguir, foram utilizados como taxa de cobertura requerida os valores encontrados nesta simulação. A seguir temos os valores da porcentagem de cobertura requerida utilizada como entrada no ECNS para as respectivas quantidades de nós utilizados em cada situação.

100 Nós: 90%

150 Nós: 98%

200 Nós: 99,75%

250 Nós: 99,9%

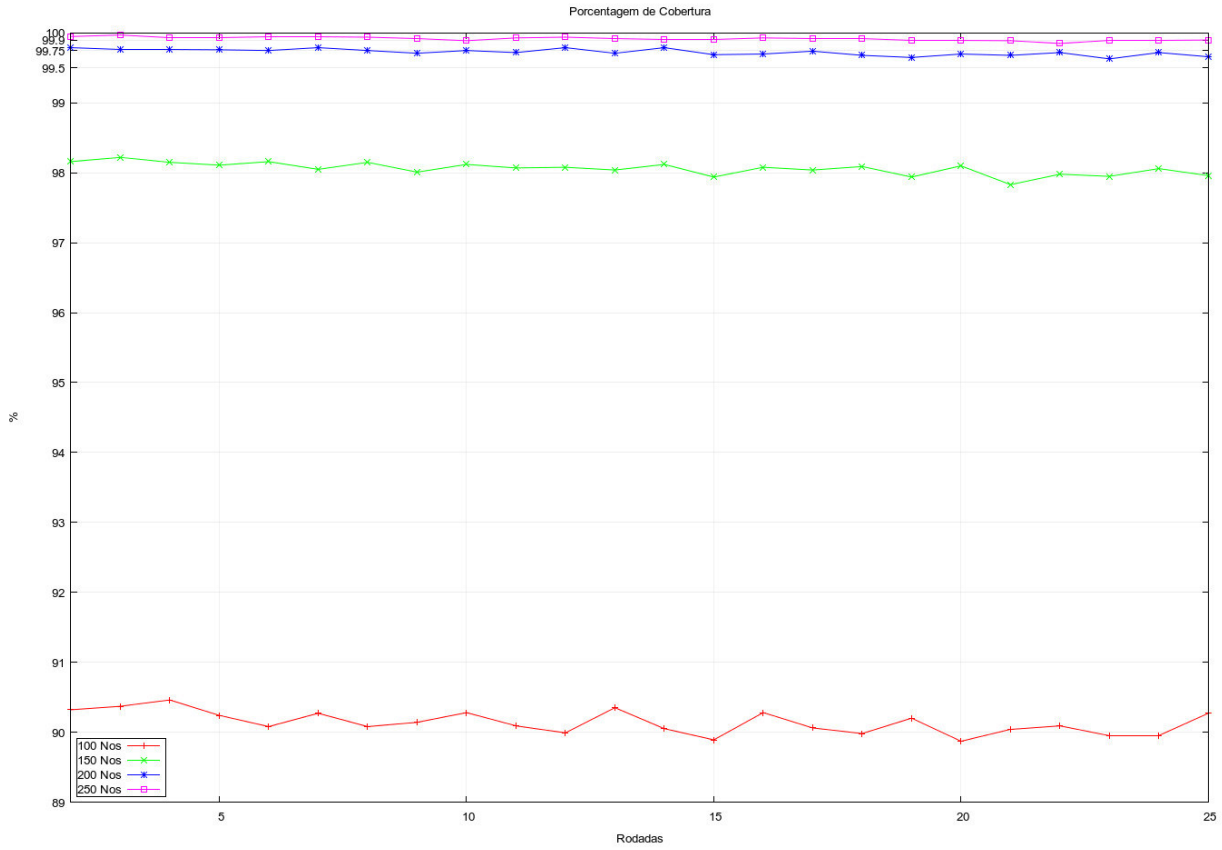

Figura 4.3: Porcentagem da cobertura em relação à manutenção de todos os nós ativos (25 rodadas)

4.2.3 Consumo de energia

Foram executadas simulações para verificar o consumo energético ao logo do tempo. As figuras 4.4, 4.5, 4.6 e 4.7 mostram os gráficos da energia residual média dos nós com a distribuição de 100, 150, 200 e 250 nós respectivamente.

Na figura 4.4, na simulação com 100 nós, obtivemos ao final da simulação, os valores de energia residual média de $64,741J$, $56,075J$, $56,095J$ e $56,004J$ para as configurações EsCobeDOO, ECNS, RadioAtivo e Hierárquico respectivamente. O algoritmo EsCobeDoo consumiu aproximadanente 25% menos energia que ECNS, Hierárquico e Rádio Ativo, que ficaram bem próximos. Na figura 4.5, com 150 nós, os valores da energia média residual foram $66,504J$, $56,004J$, $56,129J$ e $56,003J$, na mesma ordem anterior. O consumo médio ao final foi de cerca de 30% menor que os algoritmos de comparação, que continuaram praticamente iguais. Na figura 4.6, com 200 nós, os valores obtidos foram: $67,494J$,$56,133J$, $56,159$ e $56,003J$, e o consumo médio foi aproximadamente 33% inferior. E, por fim, com a utilização de 250 nós (4.7), obtivemos os valores de energia residual média de: $67,236J$, $61,37J$, $56,176$ e $56,004J$. A economia neste cenário foi de cerca de 35% sobre as configurações Hierárquico e Rádio Ativo e 20% sobre o algoritmo ECNS. Nas configurações RádioAtivo e Hierárquico, a energia residual média não foi afetada pela adição de mais nós na rede, uma vez que todos os nós permanecem ligados todo o tempo, o consumo médio não sofrerá alterações. Já a adição de mais nós no algoritmo EsCobeDOO acarreta um aumento na energia residual média, uma vez que mais nós poderão ser desativados. O Algoritmo ECNS começou a economizar energia somente a partir de 250 nós, pois neste algoritmo, para que um nó seja desativado, ele deve possuir uma quantidade relativamente grande de vizinhos - apenas a título de exemplo: para garantir uma cobertura de 95,78% são necessários 11 vizinhos ativos no ECNS.

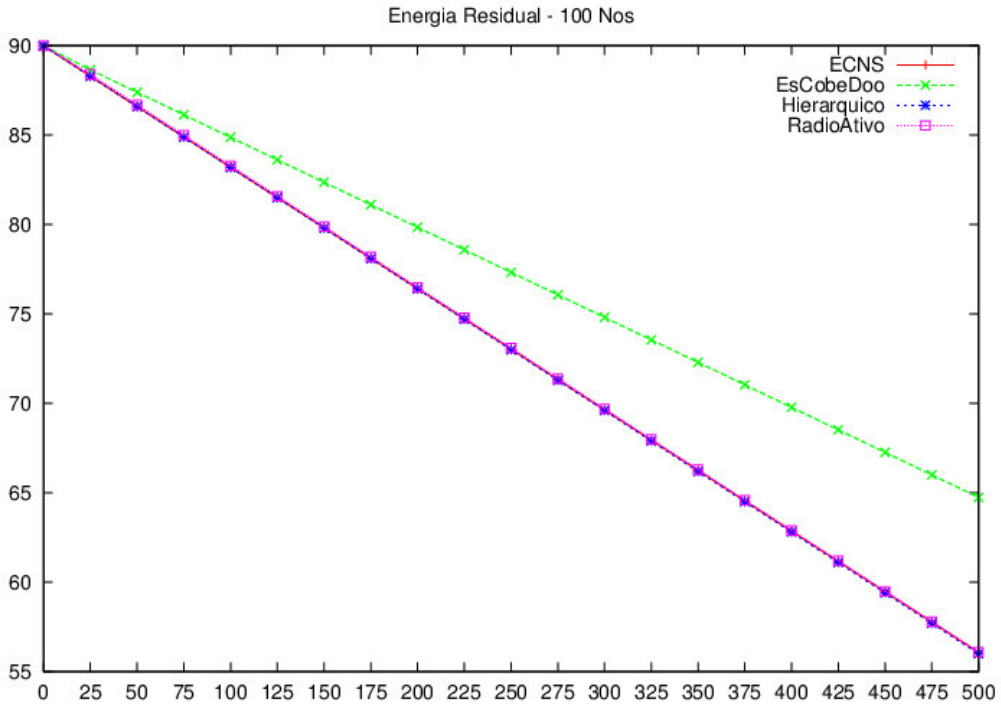

Figura 4.4: Energia residual média por tempo de simulação para 100 nós por .

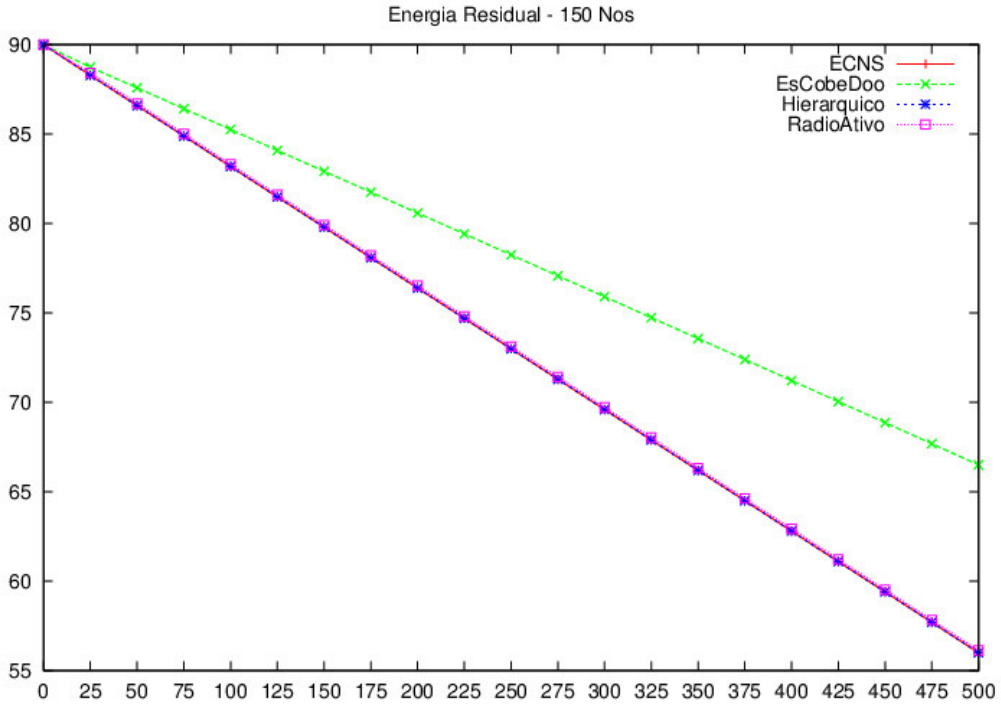

Figura 4.5: Energia residual média por tempo de simulação para 150 nós.

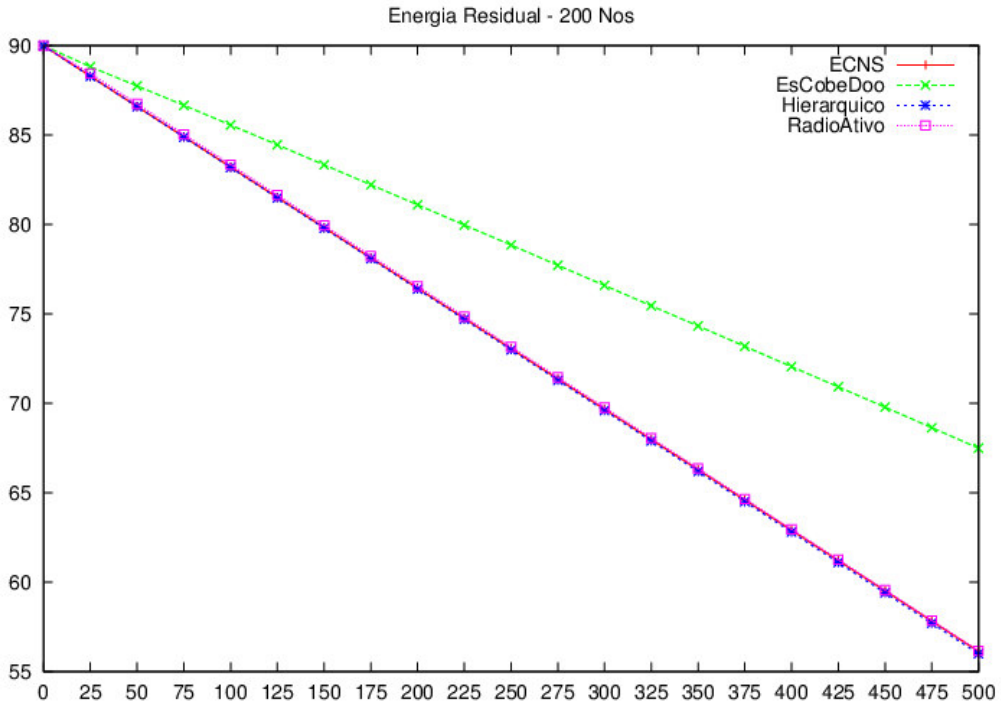

Figura 4.6: Energia residual média por tempo de simulação para 200 nós.

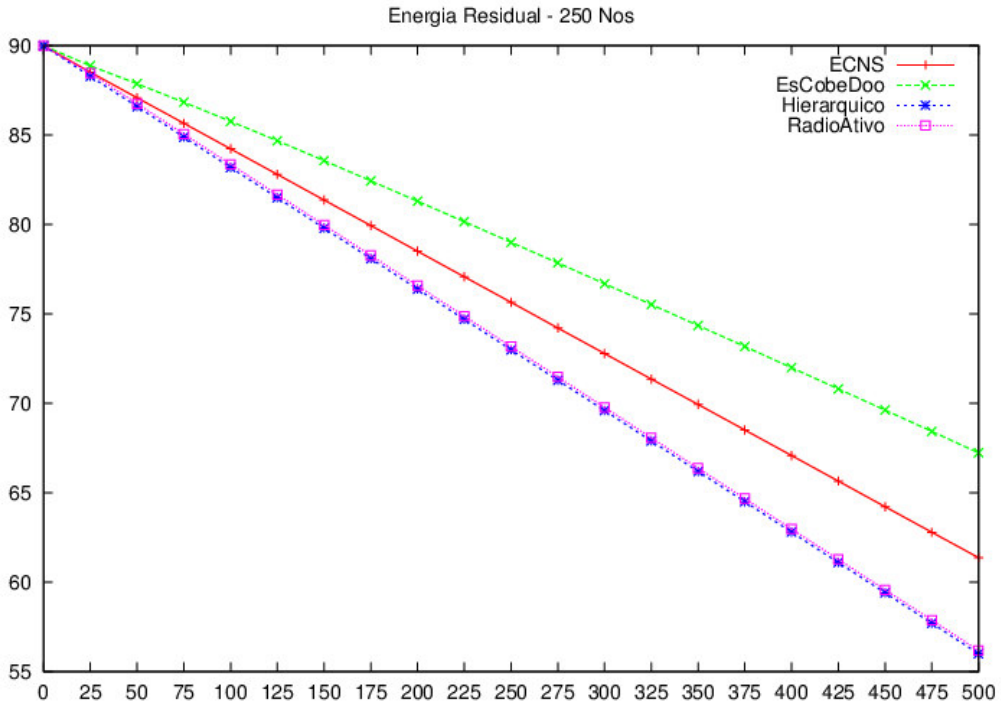

Figura 4.7: Energia residual média por tempo de simulação para 250 nós.

4.2.4 Distribuição da energia

Nesta seção mostramos como ficou a distribuição energética ao longo do espaço ao final da simulação. Para esta simulação, é mostrada o resultado com apenas uma repetição. Os

eixos x e y são o espaço físico da simulação (área de sensoreamento), enquanto o eixo z mostra o nível de energia residual dos nós.

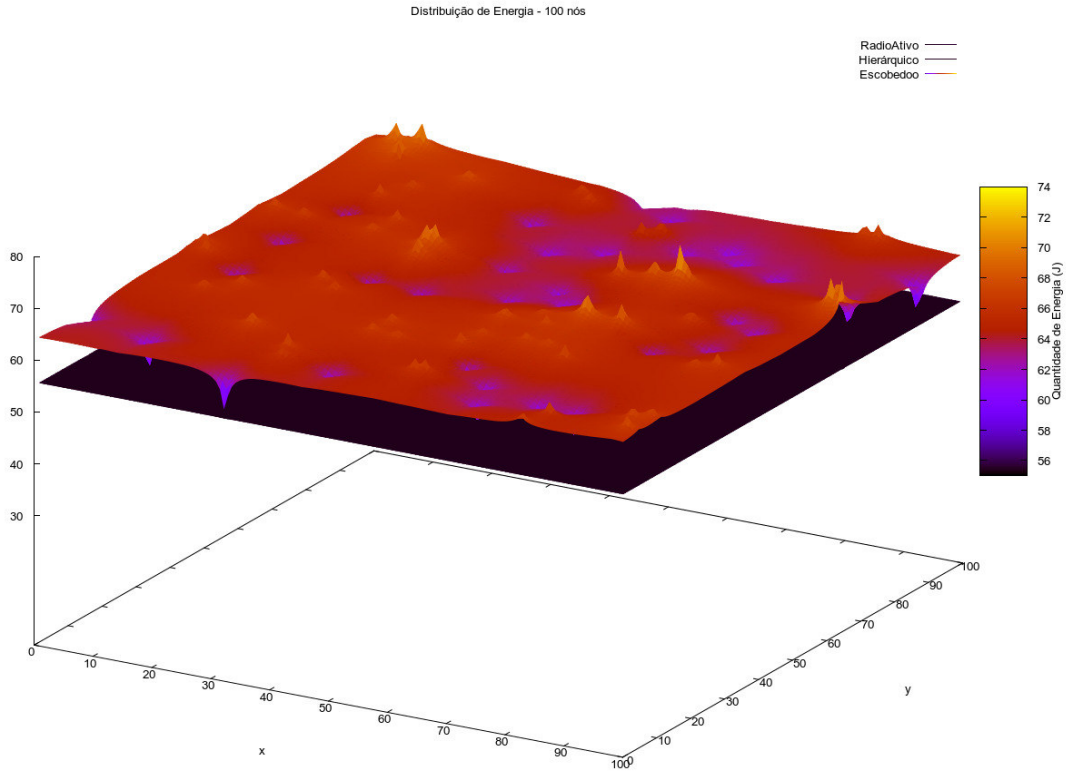

Figura 4.8: Distribuição de energia ao final da simulação com 100 nós. Os picos correspondem a nós com mais energia, as depressões, os nós com menos energia.

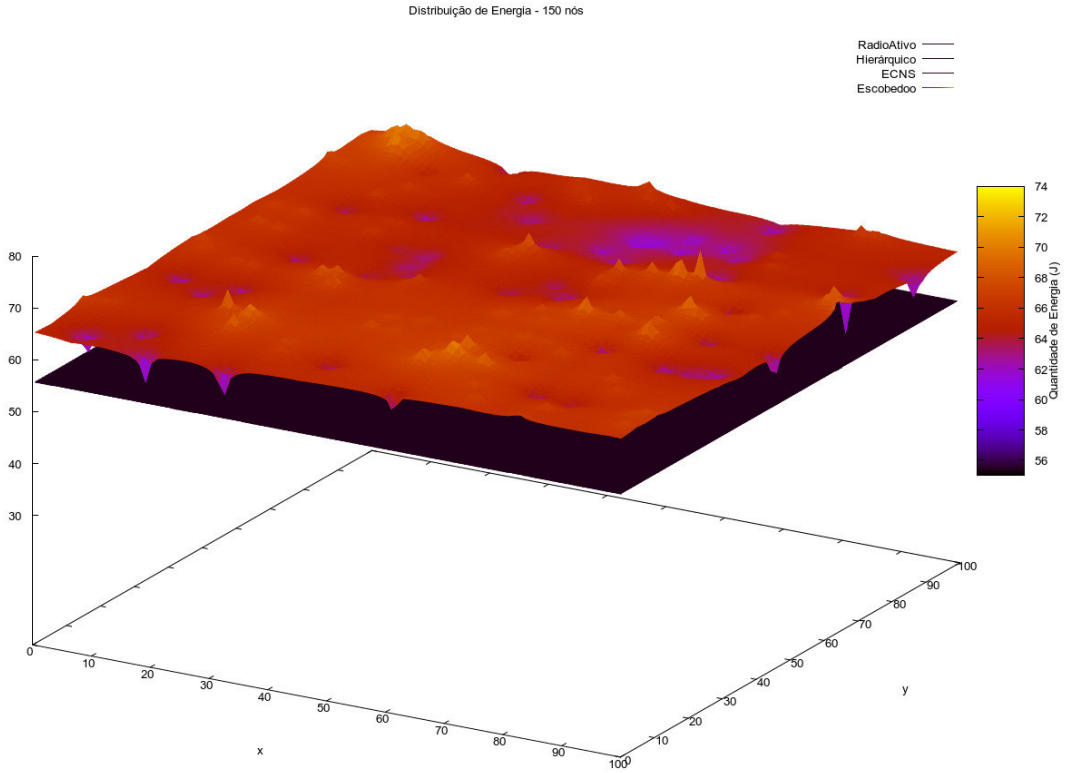

Figura 4.9: Distribuição de energia ao final da simulação com 150 nós.

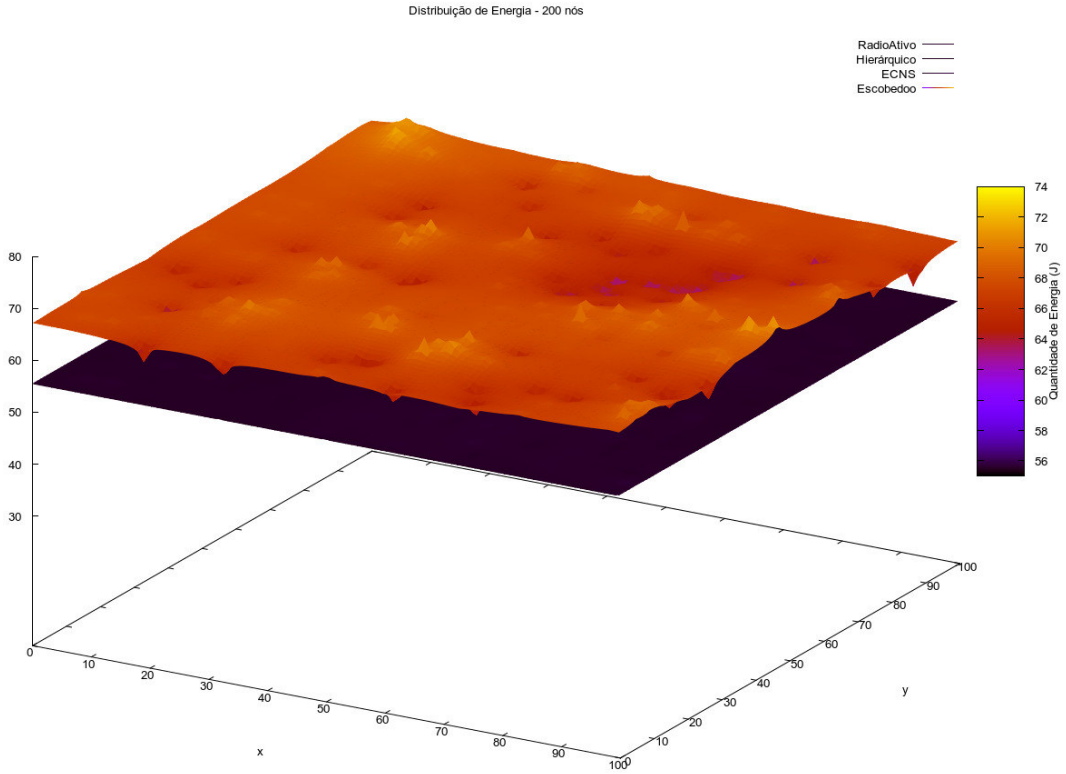

Figura 4.10: Distribuição de energia ao final da simulação com 200 nós.

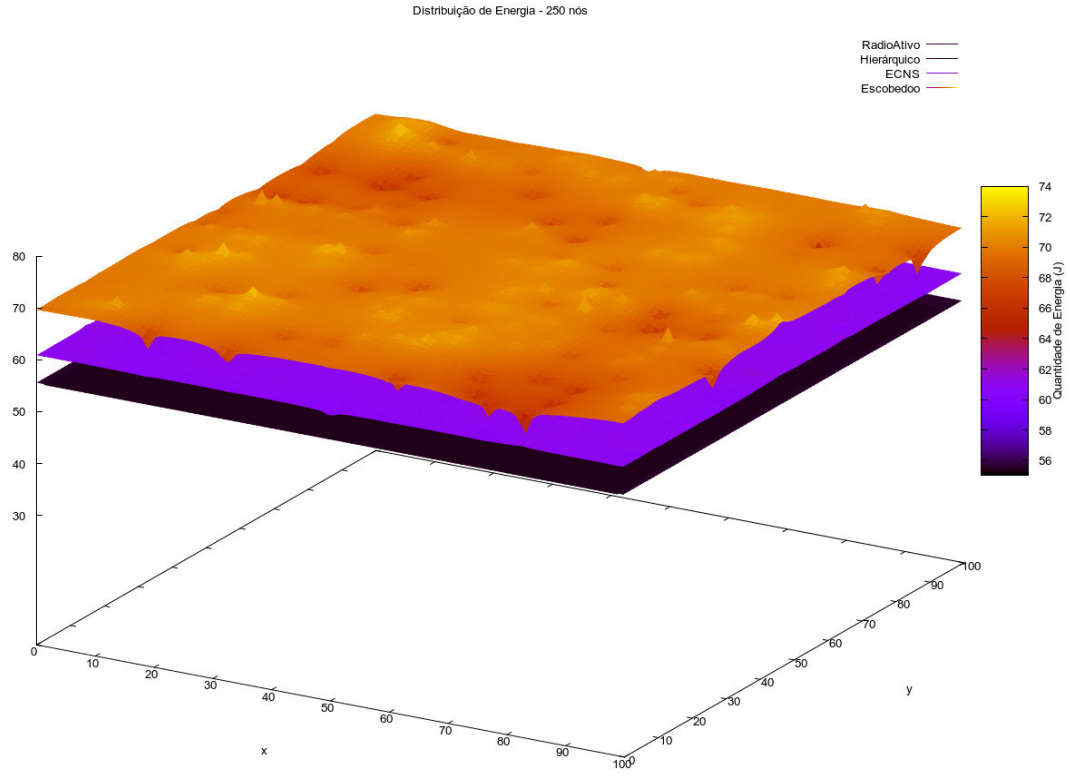

Figura 4.11: Distribuição de energia ao final da simulação com 250 nós.

De acordo com os resultados, no algoritmo EsCobeDoo o consumo energético não foi completamente uniforme, uma vez que, nós com muitos vizinhos ficam mais tempo no estado inativo, enquanto nós com menos vizinhos permanecem ativos por mais tempo - especialmente na utilização de uma quantidade menor de nós (figura 4.8), uma vez que, com uma baixa densidade de sensores os nós tendem a ter menos vizinhos e passarem mais tempo no estado ativo. Assim, podemos ver na figura áreas com um menor nível de energia (áreas em roxo), em quantidade bem maior que nas simulações com densidades superiores. Nestas, a distribuição de energia foi melhor, além da quantidade de energia residual ser superior - os resultados tendem mais para a cor amarelo claro, indicando níveis energéticos mais altos. A distribuição energética com a utilização do algoritmo hierárquico permanece praticamente constante entre todos os nós, uma vez que todos estão ativos transmitindo ao mesmo tempo - o esperado é que o consumo decaia igualmente entre todos. Desta forma, não chega a ser visível nesta escala as diferenças energéticas entre os nós, logo a imagem exibe um plano quase perfeito - motivo pelo qual foi exibida abaixo do resultado do algoritmo EsCobeDOO. O mesmo vale para os resultados da configuração RadioAtivo, que ficou tão próximo da configuração Hierárquica, que sequer é possível diferenciar seu plano nas figuras. O algoritmo ECNS só aparece nestes resultados quando utilizados 250 nós e sua distribuição também foi quase plana, mostrando que o mesmo não teve muitos nós em estado inativo - o que explica sua menor eficiência energética nas simulações aqui apresentadas.

4.2.5 Tempo de vida da rede

O tempo de vida da rede pode ser definido como o tempo de duração de uma RSSF. O que define a duração depende essencialmente da aplicação que estiver sendo executada sobre aquela RSSF. Para algumas aplicações a duração da rede pode ser representada pelo tempo até a falha do primeiro nó, em outras o tempo até que quebra da conectividade da rede - até que algum nó ainda ativo não consiga mais enviar seus dados. O tempo de vida pode ainda ser definido como o tempo até que uma porcentagem dos nós tornem-se inativos ou ainda até que todos os nós esgotem sua energia. Utilizamos neste trabalho a definição do tempo até a morte do primeiro nó, devido à indisponibilidade de outras métricas através do simulador. O tempo de vida da rede foi obtido através da estimativa de consumo obtida após os 500s de simulação. A fórmula 4.1 foi utilizada para estimar o tempo de duração da bateria em cada nó. Como a quantidade de energia dos nós foi reduzida a somente 90 Joules (seção 4.1.2.2), o tempo de duração estimado está descrito em minutos.

$$T_i = \frac{E_{I(i)} \times SimTime}{(E_{I(i)} - E_{F(i)}) \times 60} \tag{4.1}$$

T é a duração estimada do nó n_i. $E_{I(i)}$ é a energia inicial do nó n_i. $E_{F(i)}$ $SimTime$ é o tempo total da simulação - em segundos. A fórmula 4.1, bem como a função que estima o tempo de vida da rede, foram adaptadas da versão beta do simulador Castalia (versão 3.3beta) e implementadas na versão corrente (3.2). A figura 4.12, mostra o tempo médio até a morte do primeiro nó, nas simulações com 100, 150, 200 e 250 nós respectivamente.

Tanto no algoritmo Hierárquico, como na configuração RadioAtivo, a morte do primeiro nó ocorreu, em média, no tempo de $24,48$ minutos, com qualquer quantidade de nós, o que é esperado uma vez que o consumo decai de maneira uniforme quando todos os nós estão ativos executando as mesmas tarefas ao mesmo tempo. No algoritmo EsCobeDOO, o tempo médio da morte do primeiro nó com a utilização de 100 nós foi de $34,56$ minutos - tempo aproximadamente 41% superior às outras configurações. Com a utilização de 150 nós, o tempo foi de 36 minutos - 47% superior. Com 200 nós o tempo médio medido foi de $37,44$, mesmo tempo obtido com 250 nós, valor aproximadamente 53% superior às configurações Rádio Ativo e Hierárquico e 36% superior ao ECNS ($27,36$).

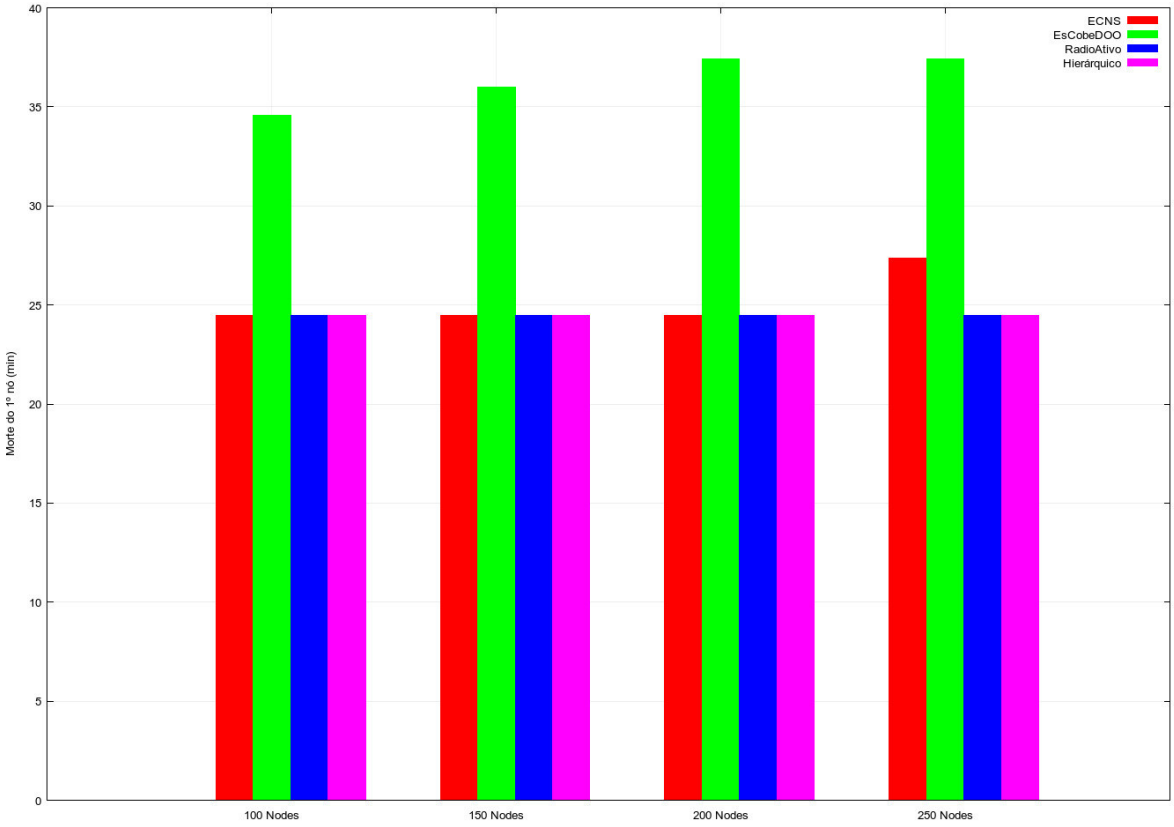

Figura 4.12: Tempo médio até a morte do primeiro nó (em minutos).

5 CONCLUSÃO E TRABALHOS FUTUROS

5.1 Conclusão

Como demonstrado nos resultados, o algoritmo EsCobeDOO propicia uma boa economia de energia nos nós de uma RSSF desativando, em turnos, nós redundantes, além de balancear o consumo de energia em redes densas.

Como a taxa de sobreposição mínima escolhida é relativamente alta, a solução aqui proposta funcionará melhor em RSSF densas, uma vez que em redes com baixa densidade de nós teríamos somente em uma pequena parte da rede uma taxa de sobreposição alta o suficiente para a desativação de nós. Porém, o impacto na cobertura obtido foi bastante reduzido, podendo ser utilizado em aplicações que requerem uma boa taxa de cobertura. O algoritmo ECNS, utilizado para comparação, provavelmente requer uma densidade de nós ainda maior que a solução aqui apresentada, uma vez que seus resultados começaram a aparecer somente com a maior densidade simulada aqui, portando o algoritmo EsCobeDOO o superou nas condições apresentadas neste trabalho e não foram executadas simulações com densidades superiores devido à obtenção de 99,9% de cobertura com a utilização de 250 nós. Este trabalho traz também como diferencial, a utilização de um algoritmo determinístico na inicialização da rede - que alivia o trabalho do MAC e ajuda a evitar colisões e perda de pacotes durante a transição de estados nos sensores.

Este trabalho pode ser facilmente modificado para utilização em redes com sensores heterogêneos - com diferentes raios de sensoreamento e comunicação, alterando-se apenas a mensagem de *Hello* - para que ela informe, por exemplo, o tipo daquele sensor, e alterando o cálculo da verificação da distância entre um nó e um vizinho, levando em conta qual a distância necessária para haver sobreposição com aquele tipo de sensor. Poucos trabalhos levam em conta a utilização de sensores heterogêneos (GUPTA; RAO; VENKATESH, 2013) (o ECNS também esta apto à utilização com sensores heterogêneos, porém requer de que a variação do raio de sensoreamento nos sensores siga a distribuição normal).

A distância entre os nós para definir a existência de uma aresta tem fundamental importância na obtenção dos resultados. Maiores distâncias possibilitam uma maior economia, uma vez que haverão mais nós monitorando uma mesma área quando a área de monitoramento considerada de cada sensor cresce. Em aplicações reais este valor vai depender do tipo de sensor utilizado e dos requisitos da aplicação para a cobertura.

5.2 Trabalhos Futuros

Como trabalhos futuros poderia-se integrar a proposta atual a um algoritmo de roteamento - ou mesmo o desenvolvimento de um algoritmo que aproveite as características desta solução. Ou ainda, a proposição de uma solução multi-camadas, incluindo um protocolo de acesso ao meio (MAC), o que poderia aproveitar ainda mais o potencial aqui demonstrado. É necessário ainda, pensar na confiabilidade no envio de mensagens - ou uma estratégia que lide

com a perca de mensagens do protocolo aqui proposto. Há ainda a necessidade de um melhor mecanismo para medir a distância entre os nós, uma vez que a solução aqui utilizada funciona bem somente para pequenas distâncias - embora uma opção viável seria calcular a distância através de várias medidas do RSSI (com medidas de RSSI suficientes, a distância pode ser bem mais confiável (BENKIC et al., 2008)), verificando a viabilidade do custo deste maior envio de mensagens. Uma outra proposição seria a utilização de tempos entre as inversões dinâmicos, uma vez que a solução tem capacidade de propagação deste tipo de alteração a todos os nós - poderia-se utilizar intervalos maiores no início da execução e menores no final em uma aplicação que requira que todos os nós permaneçam ativos o maior tempo possível, por exemplo, ou utilizar um tempo menor ao ser disparado um determinado evento, em redes orientadas a eventos.

REFERÊNCIAS BIBLIOGRÁFICAS

AL-KARAKI, J.; KAMAL, A. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, v. 11, n. 6, p. 6–28, 2004. ISSN 1536-1284.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Comput. Netw.*, Elsevier North-Holland, Inc., New York, NY, USA, v. 54, n. 15, p. 2787–2805, out. 2010. ISSN 1389-1286. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2010.05.010>.

AZIZ, N. A. A.; AZIZ, K. A.; ISMAIL, W. Z. W. Coverage strategies for wireless sensor networks. World Academy of Science, Engineering and Technology, v. 3, n. 2, p. 134 – 140, 2009. ISSN 1307-6892. Disponível em: <http://waset.org/Publications?p=26>.

BARBOSA, V. C.; GAFNI, E. Concurrency in heavily loaded neighborhood-constrained systems. *ACM Trans. Program. Lang. Syst.*, ACM, New York, NY, USA, v. 11, n. 4, p. 562–584, out. 1989. ISSN 0164-0925. Disponível em: <http://doi.acm.org/10.1145/69558.69560>.

BENKIC, K. et al. Using rssi value for distance estimation in wireless sensor networks based on zigbee. In: *Systems, Signals and Image Processing, 2008. IWSSIP 2008. 15th International Conference on*. [S.l.: s.n.], 2008. p. 303–306.

CASTALIA. *Castalia Wireless Network Simulator*: Site. 2013. Disponível em: <http://castalia.research.nicta.com.au>. Acesso em: 16 ago. 2013.

CHEN, J.; ZHANG, L.; KUO, Y. Coverage-enhancing algorithm based on overlap-sense ratio in wireless multimedia sensor networks. *Sensors Journal, IEEE*, v. 13, n. 6, p. 2077–2083, 2013. ISSN 1530-437X.

CISCO. *Cisco Visualization | The Internet of Things*: Site. 2013. Disponível em: <http://share.cisco.com/a-internet-e-as-coisas.html>. Acesso em: 13 ago. 2013.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. *Sistemas Distribuídos: Conceitos e Projeto*. 4. ed. [S.l.]: BOOKMAN COMPANHIA ED, 2007. ISBN 9788560031498.

DOMINGUEZ, C.; CRUZ-CORTÉS, N. Energy-efficient and location-aware ant colony based routing algorithms for wireless sensor networks. In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2011. (GECCO '11), p. 117–124. ISBN 978-1-4503-0557-0. Disponível em: <http://doi.acm.org/10.1145/2001576.2001593>.

GUPTA, H.; RAO, S.; VENKATESH, T. Sleep scheduling for partial coverage in heterogeneous wireless sensor networks. In: *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*. [S.l.: s.n.], 2013. p. 1–10.

HEURTEFEUX, K.; VALOIS, F. Is rssi a good choice for localization in wireless sensor network? In: *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on*. [S.l.: s.n.], 2012. p. 732–739. ISSN 1550-445X.

HUANG, C.-F.; TSENG, Y.-C. The coverage problem in a wireless sensor network. In: *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*. New York, NY, USA: ACM, 2003. (WSNA '03), p. 115–121. ISBN 1-58113-764-8. Disponível em: <http://doi.acm.org/10.1145/941350.941367>.

LEE, J.-W.; LEE, J.-Y.; LEE, J.-J. Jenga-inspired optimization algorithm for energy-efficient coverage of unstructured wsns. *Wireless Communications Letters, IEEE*, v. 2, n. 1, p. 34–37, 2013. ISSN 2162-2337.

MENG, F. et al. Energy-efficient and coverage-specific node scheduling for wireless sensor networks. In: *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*. New York, NY, USA: ACM, 2010. (MSWIM '10), p. 368–375. ISBN 978-1-4503-0274-6. Disponível em: <http://doi.acm.org/10.1145/1868521.1868582>.

MISRA, S.; KUMAR, M. P.; OBAIDAT, M. S. Connectivity preserving localized coverage algorithm for area monitoring using wireless sensor networks. *Comput. Commun.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 34, n. 12, p. 1484–1496, ago. 2011. ISSN 0140-3664. Disponível em: <http://dx.doi.org/10.1016/j.comcom.2010.03.002>.

OLIVEIRA, C. H. S. *Gerenciamento Autonômico de Energia em Redes de Sensores Sem Fio Através do Escalonamento de Atividade dos Nós*. Dissertação (Mestrado) — Mestrado e Doutorado em Ciência da Computação, Centro de Ciências, UFC, 2011.

PAILLARD, G. et al. Assigning codes in a random wireless network. In: SOUZA, J. N.; DINI, P.; LORENZ, P. (Ed.). *Telecommunications and Networking - ICT 2004*. [S.l.]: Springer Berlin Heidelberg, 2004, (Lecture Notes in Computer Science, v. 3124). p. 348–353. ISBN 978-3-540-22571-3.

PEDIADITAKIS, D.; TSELISHCHEV, Y.; BOULIS, A. Performance and scalability evaluation of the castalia wireless sensor network simulator. In: *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010. (SIMUTools '10), p. 53:1–53:6. ISBN 978-963-9799-87-5. Disponível em: <http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8727>.

RAGHUNANDAN, G. H.; LAKSHMI, B. N. A comparative analysis of routing techniques for wireless sensor networks. In: *Innovations in Emerging Technology (NCOIET), 2011 National Conference on*. [S.l.: s.n.], 2011. p. 17–22.

RIBEIRO, L.; CASTRO, M. de. Bio4sel: A bio-inspired routing algorithm for sensor network lifetime optimization. In: *Telecommunications (ICT), 2010 IEEE 17th International Conference on*. [S.l.: s.n.], 2010. p. 728–734.

ROSETA. *Total circles area - Roseta Code*: Site. 2013. Disponível em: <http://rosettacode.org/wiki/Total_circles_area>. Acesso em: 13 ago. 2013.

SHARMILA, D.; SUJITHA, R.; RAJKUMAR, G. On improving the lifetime of wireless sensor networks using virtual scheduling backbone replacement. In: *Information Communication Technologies (ICT), 2013 IEEE Conference on*. [S.l.: s.n.], 2013. p. 249–252.

TEXAS. *CC2420(ACTIVE) Single-Chip 2.4 GHz IEEE 802.15.4 Compliant and ZigBee^{TM} Ready RF Transceiver*: Site. 2013. Disponível em: <http://www.ti.com/product/cc2420>. Acesso em: 14 ago. 2013.

VARGA, A. The omnet++ discrete event simulation system. *Proceedings of the European Simulation Multiconference (ESM'2001)*, June 2001.

VARGA, A.; HORNIG, R. An overview of the omnet++ simulation environment. In: *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. (Simutools '08), p. 60:1–60:10. ISBN 978-963-9799-20-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1416222.1416290>.

WANG, B. Coverage problems in sensor networks: A survey. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 43, n. 4, p. 32:1–32:53, out. 2011. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/1978802.1978811>.

WANG, L.; XIAO, Y. A survey of energy-efficient scheduling mechanisms in sensor networks. *Mob. Netw. Appl.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 11, n. 5, p. 723–740, out. 2006. ISSN 1383-469X. Disponível em: <http://dx.doi.org/10.1007/s11036-006-7798-5>.

WANG, X. et al. Integrated coverage and connectivity configuration in wireless sensor networks. In: *Proceedings of the 1st international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2003. (SenSys '03), p. 28–39. ISBN 1-58113-707-9. Disponível em: <http://doi.acm.org/10.1145/958491.958496>.

WIKIPEDIA. *Dining Philosophers Problem*: Site. 2013. Disponível em: <http://en.wikipedia.com/wiki/Dining_philosophers_problem>. Acesso em: 16 ago. 2013.

APÊNDICE A – CÁLCULO DA ÁREA TOTAL DE VÁRIOS CÍRCULOS

Código utilizado para o cálculo da área total de cobertura de círculos em um plano - a área coberta por um ou mais círculo só deve ser contada uma vez. O código está em linguagem python e foi adaptado do obtido de (ROSETA, 2013).

```
1
2  from collections import namedtuple
3
4  Circle = namedtuple("Circle", "x y r")
5
6  circles = [
7  Cicle(0.0, 2.0, 2.0),
8  Cicle(5.8, 1.1, 1.5),
9  Cicle(3.0, 2.2, 2.3),
10 Cicle(0.8, 2.9, 1.9),
11 Cicle(1.9, 4.1, 3.2)
12 ]
13
14 def main():
15
16     # area de sensoreamento:
17     x_min = 0.0
18     x_max = 100.0
19     y_min = 0.0
20     y_max = 100.0
21
22     box_side = 500
23
24     dx = (x_max - x_min) / box_side
25     dy = (y_max - y_min) / box_side
26
27     count = 0
28
29     for r in xrange(box_side):
30         y = y_min + r * dy
31         for c in xrange(box_side):
32             x = x_min + c * dx
33             if any((x-circle.x)**2 + (y-circle.y)**2 <= (circle.r ** 2)
34                    for circle in circles):
35                 count += 1
36
37     print "Approximated area:", count * dx * dy
38
39 main()
```