Universidade Federal da Bahia
Escola Politécnica / Instituto de Matemática
Programa de Pós-Graduação em Mecatrônica

ULISSES TELEMACO NETO

ALGORITMO DE CONTROLE DE TOPOLOGIA PARA REDE DE SENSORES SEM FIO QUE CONSIDERA O EFEITO OVERHEARING

DISSERTAÇÃO DE MESTRADO

Salvador
2009

ULISSES TELEMACO NETO

ALGORITMO DE CONTROLE DE TOPOLOGIA PARA REDE DE SENSORES SEM FIO QUE CONSIDERA O EFEITO OVERHEARING

Dissertação apresentada ao Programa de Pós-Graduação em Mecatrônica da Escola Politécnica e do Instituto de Matemática da Universidade Federal da Bahia como requisito parcial para obtenção do grau de Mestre.

Orientador: *Prof. Dr. Flávio Morais de Assis Silva*

Salvador
2009

TERMO DE ACEITAÇÃO

ULISSES TELEMACO NETO

ALGORITMO DE CONTROLE DE TOPOLOGIA PARA REDE DE SENSORES SEM FIO QUE CONSIDERA O EFEITO OVERHEARING

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre no Programa de Pós-Graduação em Mecatrônica da Escola Politécnica e do Instituto de Matemática da Universidade Federal da Bahia, pela seguinte banca examinadora:

Prof. Dr. Flávio Morais de Assis Silva
Dr.-Ing, Technische Universität Berlin, Alemanha
Professor do Departamento de Ciência da Computação da UFBA

Prof. Dr. Amauri Oliveira
Doutor, Universidade Federal da Paraíba - UFPB
Professor do Departamento de Engenharia Elétrica da UFBA

Prof. Dr. Sérgio Gorender
Doutor, Universidade Federal de Pernambuco - UFPE
Professor do Departamento de Ciência da Computação da UFBA

Salvador, 9 de Novembro de 2009

Dedico essa dissertação de mestrado a todos os brasileiros que, apesar de terem a mesma capacidade que eu, o mesmo direito que eu, infelizmente não têm as mesmas oportunidades de estudar e contribuir com a comunidade acadêmica.

AGRADECIMENTOS

Ao Prof. Dr. Flávio Morais de Assis Silva, orientador desta dissertação, por todo empenho, sabedoria, compreensão e, acima de tudo, exigência. Gostaria de destacar a sua participação nas discussões, simulações e na elaboração do algoritmo proposto nesse trabalho. Seu envolvimento e suas idéias foram essencial para que concluíssemos este trabalho. Gostaria de agradecê-lo ainda por ter me mostrado o lado excitante da vida de um pesquisador. Os seus ensinamentos foram muito além dos temas tratados nesse estudo e modificou a forma como vejo a pesquisa e desenvolvimento de idéias.

À minha amada esposa Vanessa pelo amor, carinho e paciência que demonstrou durante os meses em que esse trabalho foi realizado. Agradeço também por demonstrar, a cada dia, admiração pelo meu trabalho e pelo meu desejo em crescer profissionalmente e academicamente.

Aos meus pais, Telemaco e Noilza, pelo amor incondicional, carinho, respeito, sabedoria e princípios usados para solidificar a base de nossa família e plantar, em cada um de seus filhos, o desejo pela educação. Agradeço ainda à minha amada mãe pela preocupação que demonstrou durante esse trabalho. Agradeço ao meu pai Telemaco por ter se tornado o maior referencial para um filho. O meu crescimento está diretamente associado aos exemplos e lições que ele deixou.

Às minhas queridas irmãs, Emmanuelle e Caroline, por acreditarem na minha capacidade e desejarem, junto comigo, a realização desse projeto.

Ao meu tio, Dr. Isaias Paiva, por acreditar no meu potencial e me incentivar a ir mais longe. Pela alegria demonstrada nas simples conquistas que faz com que os grandes desafios pareçam pequenos.

Aos demais familiares pela força e por acreditarem em mim as vezes mais que eu mesmo.

Ao meu amigo Walter Magioli por incentivar e torcer com as vitórias conquistadas nesse trabalho.

Ao amigo Prof. Galileu Batista pelas lições - de vida e acadêmicas. Apesar de não ter participado diretamente desse trabalho, o aprendizado que ele deixou foi importante para esse trabalho.

Aos demais professores do Curso de Mecatrônica da Universidade Federal da Bahia pelas aulas e pelos ensinamentos importantes para a base desse trabalho.

Aos funcionários da Escola Politécnica e do Instituto de Matemática da Universidade Federal da Bahia por serem prestativos, cordiais e pela competência e dedicação ao trabalho.

Aos colegas de curso que viveram comigo uma parte dessa caminhada. A convivência com cada um deles foi também um ingrediente que fez possível esse trabalho.

Aos demais amigos e amigas que, mesmo distantes, sempre estiveram presentes.

The formulation of a problem is often more essential than its solution, which may be merely a matter of mathematical or experimental skill.

—ALBERT EINSTEIN

PUBLICAÇÕES

O presente trabalho deu origem ao artigo *A Topology Control Algorithm for Wireless Sensor Networks that considers Overhearing* [1] publicado no *9th IEEE International Symposium on Autonomous Decentralized Systems (ISADS 2009)* em Atenas, Grécia, em Março de 2009.

RESUMO

Um dos maiores desafios relacionados a Rede de Sensores Sem Fio é desenvolver técnicas que otimizem o uso dos recursos dos dispositivos (em especial energia). O Controle de Topologia se destaca por ser uma das principais técnicas utilizadas para otimizar o uso de energia em uma Rede de Sensores Sem Fio. Apesar de ser um tema bastante estudado, pouca atenção foi dada aos efeitos do *overhearing* no Controle de Topologia, ou seja, o efeito do custo de recebimento dos nós que receberam uma mensagem que não era destinada a eles.

Esse trabalho apresenta um algoritmo para Controle de Topologia em Rede de Sensores Sem Fio local e distribuído, que tem como objetivo reduzir a potência de transmissão dos nós sensores para que eles utilizem os seus recursos de forma mais eficiente. A principal contribuição desse algoritmo em comparação aos trabalhos relacionados é que ele considera o custo de *overhearing* no cálculo no controle de topologia da rede. Adicionalmente, o algoritmo otimiza a rotina que reduz o grafo de conectividade da rede e propõe uma estratégia para eliminar as arestas denominadas *k-redundantes* (para $k \geq 2$) identificadas localmente.

Palavras-chave: Controle de Tolologia, *Overhearing*, Rede de Sensores Sem Fio, Arestas redundantes, Caminho de custo mínimo

ABSTRACT

One of the challenges involving Wireless Sensor Networks is developing techniques that can used to optimizing resources from the network. Topology control is one of the main techniques that can be used to decrease energy spenditure in Wireless Sensor Networks. Although it has been the subject of much research, less attention has been devoted to study the effects of *overhearing* on topology control, i.e., the effects of the cost implied by nodes hearing transmissions even if these transmissions were not intended to them.

In this text we describe a (distributed and localized) algorithm for topology control in wireless sensor networks. Our approach differs from previous work mainly in the sense that it takes the effects of overhearing into consideration and that it might eliminate more communication links from a given connectivity graph, and thus possibly assign lower transmission power to some nodes. This is done by eliminating so-called k-redundant edges ($k \geq 2$), instead of eliminating only two-redundant edges.

We present proofs of properties of the algorithm and simulation results.

Keywords: Topology Control, Overhearing, Wireless Sensor Network, Minimum-energy Property

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

Recentes avanços nas áreas de microprocessadores e redes sem fio levaram ao desenvolvimento das Redes de Sensores Sem Fio (RSSF). Essas são um tipo especial de rede (tipicamente) formada por centenas ou até mesmo milhares de dispositivos que atuam como sensores (de movimento, acústicos, de temperatura, de calor, sísmico, etc) e que trocam informação através de canais de comunicação sem fio.

As RSSF já são usadas em uma grande variedade de aplicações: na área de segurança do trabalho, de monitoramento ambiental, biomedicina, construção civil, automação industrial (para citar apenas algumas aplicações) [13–19]. Em muitos cenários, o posicionamento físico dos nós é feito de forma aleatória e pode ocupar uma grande região [20–22].

A infra-estrutura necessária para elaborar uma RSSF é relativamente simples. Como os nós se comunicam através de um canal sem fio, não é necessária uma infra-estrutura cabeada e o custo de uma RSSF é influenciado pelos baixos preços dos nós sensores, que podem chegar a custar menos de U$ 1,00 [20].

Uma característica derivada do baixo custo dos nós sensores é a sua limitação de *hardware*. Tipicamente esses dispositivos possuem uma fonte de energia limitada e não substituível. Assim uma das maiores restrições das RSSF é manter o consumo de energia baixo [20,23,24]. Esta restrição torna os problemas relacionados às Redes de Sensores Sem Fio (protocolos de roteamento, segurança, autenticação, etc) mais complexos. As técnicas tradicionais podem não ser viáveis por necessitarem de processamento intensivo ou de memória para armazenar tabelas de roteamento, o que implicaria um gasto inadequado de energia ou os dados não caberiam na memória. Assim, muitas das soluções para as RSSF são baseadas em um compromisso entre eficiência e consumo de energia.

Uma das estratégias utilizadas para economizar energia em uma RSSF é o Con-

trole de Topologia. Essa técnica consiste - simplificadamente - em reduzir a potência de transmissão de alguns nós preservando algumas propriedades da rede (conectividade, por exemplo) [25].

O Controle de Topologia pode ser implementado quando a RSSF é formada por nós sensores equipados com rádios transmissores que suportam o ajuste na potência de transmissão. Quanto maior a potência de transmissão, maior é a quantidade de energia consumida pelo nó sensor [8, 10, 11, 26, 27]. Portanto, reduzir a potência de transmissão contribui para economizar energia na rede.

Existem vários trabalhos que apresentam soluções para o Controle de Topologia em RSSF ([5, 6, 25, 28–32] são apenas alguns). Dentre esses trabalhos, destacam-se aqueles que propõem soluções distribuídas e localizadas [5, 6, 28, 29, 31, 32]. Um algoritmo de Controle de Topologia localizado é aquele em que cada nó decide como deve ajustar sua potência de transmissão a partir da análise das informações que ele conhece sobre a sua vizinhança. Soluções distribuídas e localizadas contribuem para aumentar a extensibilidade da rede (um requisito importante nesse tipo de rede [20]).

Apesar de os algoritmos de Controle de Topologia distribuídos serem discutidos em vários trabalhos, apenas alguns consideram o gasto de energia com o efeito *overhearing* ([5, 33, 34]). De acordo com o modelo de energia apresentado em [26], os nós consomem energia durante o recebimento de uma mensagem e a quantidade de energia consumida é um valor constante multiplicado pelo tempo em que o nó permaneceu com o receptor no estado de recebimento ativo. Nas transmissões realizadas através de *broadcast*, todos os nós que estão no raio de alcance do nó transmissor recebem a mensagem (com exceção daqueles que tenham os seus rádio receptores desligados). O efeito *overhearing* ocorre quando os nós que não têm interesse em uma mensagem a recebem. O gasto de energia com *overhearing*, mesmo não sendo considerado por vários trabalhos, tem impacto no consumo de energia total da rede [34].

O objetivo desse trabalho é apresentar um algoritmo de Controle de Topologia distribuído e localizado que considera o consumo de energia do *overhearing*. Esse estudo

apresenta uma contribuição para a área de pesquisa em RSSF por considerar um custo negligenciado por diversos trabalhos relacionados. Em [6] os autores até registram a importância do gasto de energia com o *overhearing*, mas não o consideram para calcular o custo de transmissão de uma mensagem na rede.

Será provado matematicamente e através de simulações que o *overhearing* pode influenciar no consumo de energia e, portanto, deve ser considerado. Foram realizadas simulações e comparações com vários algoritmos de Controle de Topologia (localizados e centralizados).

Os resultados obtidos pelo algoritmo proposto nesse estudo foram próximos aos obtidos pelos algoritmos que conheciam a topologia completa da rede. Ou seja, apenas com informações locais, o algoritmo proposto teve uma média de redução de arestas próxima a média alcançada pelos algoritmos que conheciam toda a rede.

O restante desse texto está organizado da seguinte forma: O capítulo 2 contextualiza o leitor sobre os aspectos relacionados a Redes de Sensores Sem Fio importantes no contexto desse trabalho. No capítulo 3 são apresentadas questões relacionadas ao problema de Controle de Topologia para RSSF e ao gasto de energia com o efeito *overhearing*. Ainda nesse capítulo, são apresentados os principais trabalhos relacionados a esse estudo. Uma proposta de um algoritmo para Controle de Topologia que considera o consumo de energia do efeito *overhearing* e reduz as arestas *k-redundantes* identificadas com informações locais da rede é apresentada capítulo 4. Os resultados das simulações realizadas, bem como uma discussão sobre esses resultados, são apresentados no final do capítulo 4. No capítulo 5, são apresentadas as conclusões desse trabalho de pesquisa bem como uma discussão sobre alguns possíveis trabalhos futuros.

CAPÍTULO 2

REDE DE SENSORES SEM FIO

Serão apresentados a seguir alguns dos principais aspectos relacionados a Redes de Sensores Sem Fio (RSSF). Existem diversos trabalhos (como, por exemplo, [20, 21]) cujo objetivo é apresentar um panorama detalhado sobre RSSF bem como dos estudos realizados sobre o tema. Esses trabalhos discutem, dentre outros aspectos, classificações e taxonomias para diferentes tipos de RSSF. Esse capítulo, no entanto, não tem este objetivo. Ao invés disso, serão apresentados os aspectos relacionados a RSSF importantes no contexto desse trabalho. Em termos gerais, os aspectos abordados serão aqueles relativos ao consumo de energia durante a operação dos nós.

Este capítulo está estruturado da seguinte maneira. Na seção 2.1 é apresentada a definição de RSSF e de alguns conceitos relacionados. A descrição dos componentes que compõem um nó sensor é feita na seção 2.2. Na seção 2.3 são apresentadas algumas características das RSSF. Na seção 2.4 são apresentadas três importantes plataformas de nós sensores. A seção 2.5 descreve algumas aplicações que utilizam as RSSF.

2.1 REDE DE SENSORES SEM FIO

Uma rede *ad hoc* é uma rede formada por nós que se comunicam entre si através de um canal sem fio sem a necessidade de um agente centralizador (um ponto de acesso por exemplo). Os nós da rede são normalmente distribuídos de forma aleatória e em alguns casos podem se locomover. A topologia desse tipo de rede é dita dinâmica, porque o conjunto de nós que compõe a rede não é previamente conhecido e pode variar durante o tempo de vida da rede (nós podem *entrar* e *sair* da rede de forma inesperada) [35–37].

Redes de Sensores Sem Fio (RSSF) são um tipo de rede *ad hoc* formada por dispositivos chamados nós sensores. As RSSF se diferem das redes *ad hoc* convencionais

em várias características. Dentre as mais importantes, destacam-se: normalmente são formadas por uma grande quantidade de nós (centenas ou até milhares); e os nós têm severas restrições de *hardware*, principalmente com relação à capacidade de energia. Essa limitação se deve ao fato de os nós de uma RSSF serem projetados para funcionar sem intervenção, utilizando baterias (normalmente não recarregáveis e não substituíveis) como fonte de energia.

O funcionamento de uma RSSF normalmente envolve o sensoriamento de eventos e o envio de informações ao longo da rede para uma unidade centralizadora de informações, chamada de estação base (EB).

2.2 NÓS SENSORES

Os nós sensores são dispositivos projetados para realizar basicamente três atividades: (a) identificar e converter informações analógicas em outra informação (tipicamente impulsos elétricos); (b) comunicar-se com outros nós para executar uma tarefa programada; e (c) processar informações (coletadas ou recebidas) [20].

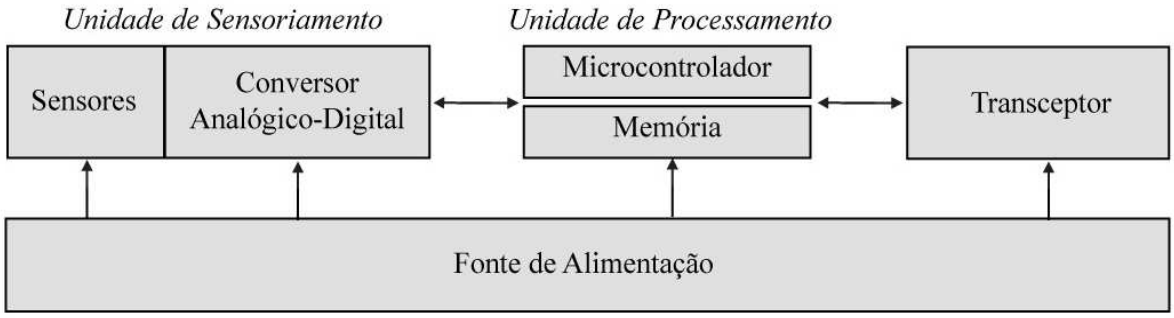
Esses dispositivos possuem severas restrições de *hardware* e são comumente compostos dos seguintes componentes: uma unidade de sensoriamento, uma unidade de processamento, um dispositivo de rádio transmissão e recepção (transceptor) e uma fonte de alimentação [20]. A estrutura básica de um nó sensor está ilustrada na Figura 2.1.

Os nós sensores podem ser equipados com módulos opcionais, como, por exemplo, módulo para locomoção, módulo de localização GPS, módulo para recarga de baterias, atuadores, dentre outros.

A unidade de sensoriamento é formada por dois tipos de componentes: um ou mais sensores e uma unidade de conversão de dados analógico/digital. Os sensores são dispositivos capazes de identificar fenômenos físicos e características do ambiente: temperatura, pressão, umidade, luminosidade e movimentação, para citar apenas alguns.

A unidade de processamento é composta por um microcontrolador e por uma unidade

Figura 2.1. Componentes de um nó sensor (baseada em [2])



de armazenamento de programas e dados. Os microcontroladores normalmente operam em baixa freqüência (1 a 8 MHz [7, 9, 12]) e possuem uma limitada capacidade de armazenamento (4 a 512 $Kbytes$ [7, 9, 12]).

O transceptor permite conectar o nó sensor a outros nós da rede através do envio e recebimento de mensagens através de um canal sem fio. Tipicamente os transceptores utilizam frequências que variam de 433 MHz a 2.4 GHz [8, 10, 11]. Os transceptores podem oferecer a possibilidade de ajuste na potência de transmissão. Normalmente esses dispositivos possuem um conjunto definido de níveis de potência - um conjunto discreto de valores de potência [8, 10, 11].

A fonte de alimentação de um nó sensor são, tipicamente, baterias não recarregáveis. A quantidade de energia disponível para um nó sensor é, como já expresso anteriormente, uma das principais restrições desses dispositivos.

As estações base (EB), por sua vez, possuem características distintas dos nós sensores. Como são unidades que centralizam e processam informações na rede, as estações base normalmente não têm as mesmas restrições de *hardware* dos nós sensores. Com relação a energia, por exemplo, podem ser alimentadas por baterias de possível reposição ou mesmo serem alimentadas com energia elétrica convencional. As EB podem ainda ser equipadas com unidades de processamento adequadas para um volume maior de informação e podem ter a sua localização predeterminada.

2.3 CARACTERÍSTICAS DE UMA RSSF

Uma RSSF é tipicamente composta por centenas/milhares de nós sensores e por uma estação base (EB). Embora não exista uma limitação quanto ao número de estações base, muitos trabalhos consideram a existência de apenas uma EB por RSSF [20].

Com relação à capacidade de movimentação, os nós podem ser classificados como estáticos ou dinâmicos. Os nós estáticos são aqueles que não têm a capacidade de se movimentar na rede. Já os nós dinâmicos têm a capacidade de se movimentarem.

Em uma rede sem fio, a conexão entre dois nós é determinada, dentre outras coisas, pela potência de transmissão dos nós [36]. Considere o par de nós u e v. Caso o nó v possa receber informações do nó u, então existe uma *conexão unidirecional* $u \rightarrow v$, caso contrário v não é alcançado por u. Quando existem duas *conexões unidirecionais* $u \rightarrow v$ e $v \rightarrow u$, a conexão entre u e v é dita *bidirecional*. Com base em uma representação de uma RSSF com grafos, uma rede é classificada como *conexa* ou *fortemente conexa* se houver um caminho na rede entre quaisquer pares de nós da rede e o grafo que modela a rede for, respectivamente, não direcionado ou direcionado.

Os nós sensores normalmente acumulam duas funções. Além de atuarem como nós sensores propriamente ditos, também atuam como roteadores de mensagens para outros nós da rede.

Os nós sensores podem se comunicar com a estação base de formas distintas. Nos modelos apresentados em [38, 39], os nós sensores se comunicam diretamente com a Estação Base. Outros modelos propõem que os nós se comuniquem com a estação base através de múltiplos passos [40–43].

2.4 PLATAFORMAS DE NÓS SENSORES

Existem diversas plataformas para RSSF. Um registro atualizado dos principais modelos de nós sensores para RSSF pode ser encontrado em [44–46]. Algumas dessas plata-

formas utilizam o mesmo modelo de transceptor. Como o custo associado a transmissão e recebimento de mensagens é um fato central neste trabalho, serão descritas a seguir algumas plataformas que utilizam transceptores de modelos distintos.

2.4.1 Plataformas Crossbow

A *Crossbow Technology* é uma importante fabricante de *kits* para RSSF. Os *kits* para RSSF são tipicamente compostos pelos seguintes tipos de dispositivos: (a) módulos de processamento e transceptor, (b) módulos de sensoriamento e (c) dispositivos de interface de rede e *gateway* [47]. Um nó sensor é formado pela combinação de um módulo de processamento e transceptor com um módulo de sensoriamento. A plataforma *Crossbow* possui uma grande variedade de sensores (de luminosidade, de temperatura, de pressão atmosférica, acelerômetro, acústico, dentre outros), que podem ser acoplados aos módulos de processamento e transceptor através de um conector de expansão.

O módulo de processamento e transceptor é formado pelos seguintes componentes: uma unidade de processamento, uma fonte de energia e um transceptor. A seguir serão apresentadas as especificidades de três plataformas de nós sensores da *Crossbow*: *Iris (XM2110)*, *Micaz (MPR2400)* e *Mica2 (MPR400)*. Estas plataformas representam, em ordem cronológica inversa, gerações de nós sensores produzidos pela empresa.

2.4.1.1 Plataforma Crossbow Iris XM2110

A unidade de processamento da plataforma *Crossbow Iris XM2110* (Figura 2.2) tem um processador de 8 *bits Atmel ATmega1281* [7] com um núcleo RISC que pode operar numa frequência de 1 a 8 MHz e com uma memória *SRAM* de 8 *Kbytes*. A unidade de armazenamento é composta de 128 *Kbytes* de memória para instruções, 512 *Kbytes* de memória para dados além de 4 *Kbytes* de memória *EEPROM*. O dispositivo é equipado com o transceptor *Atmel RF230* [8] - 2.4 GHz, compatível com o padrão IEEE 802.15.4 [48]. O alcance máximo do transceptor é 300 m e a taxa de transferência pode atingir

Figura 2.2. Sensor da Plataforma *Crossbow Iris XM2110* (fonte: [3])

até 250 *Kpbs*. O nó sensor é alimentado por duas baterias do tipo *AA*. Cada nó tem ainda a capacidade de atuar como roteador na rede.

2.4.1.2 Consumo de energia da plataforma Crossbow Iris XM2110

Os nós sensores consomem energia para manter o funcionamento dos seus componentes: unidade de processamento, unidade de sensoriamento e transceptor.

Os dados referentes ao consumo de energia do microcontrolador *Atmega 1281* [7] usado na plataforma *Iris XM2110* estão descritos na Tabela 2.1.

A Tabela 2.2 traz um sumário do consumo de energia do transceptor *Atmel RF230* operando a aproximadamente 3 V e com uma taxa de transferência de 250 *kbps* [8].

2.4.1.3 Plataforma Crossbow Micaz MPR2400

A unidade de processamento da plataforma *Crossbow Micaz MPR2400* (Figura 2.3) tem um processador de 8 *bits Atmel ATmega 128L* [9] com um núcleo RISC que pode

Tabela 2.1. Consumo de energia do microcontrolador *Atmega 1281* [7]

Modo de Funcionamento		Corrente Elétrica (mA)
Ativo	1 MHz - 2 V	0.5
	4 MHz - 3 V	3.2
	8 MHz - 5 V	10
Inativo	1 MHz - 2 V	0.14
	4 MHz - 3 V	0.7
	8 MHz - 5 V	2.7
Sleep		< 0.005

Tabela 2.2. Consumo de energia do transceptor *Atmel RF230* operando a aproximadamente 3 V e com uma taxa de transferência de 250 *kbps* [8]

Modo de Funcionamento	Corrente Elétrica (mA)	Consumo de Energia (nJ/bit)
Recepção (E_{Rx})	15.5	2.4
Transmissão (E_{Tx})		
-17 dBm	9.5	114.0
-3 dBm	12.5	150.0
1 dBm	14.5	174.0
3 dBm	16.5	198.0

Figura 2.3. Sensor da Plataforma *Crossbow Micaz MPR2400* (fonte: [3])

operar numa frequência de 1 a 8 MHz. A unidade de armazenamento é composta de memória para instruções de 128 $Kbytes$, 512 $Kbytes$ de memória para dados, além de 4 $Kbytes$ de memória *EEPROM*. O dispositivo é equipado com o transceptor *Chipcon CC2420* [10] - 2.4 GHz, compatível com o padrão IEEE 802.15.4 [48]. O alcance máximo do transceptor é aproximadamente 100 metros e a taxa de transferência pode alcançar até 250 $Kbps$. O nó sensor é alimentado por duas baterias do tipo AA.

2.4.1.4 Consumo de energia da plataforma Crossbow Micaz MPR2400

Os dados referentes ao consumo de energia do microcontrolador *Atmega 128L* [9] usado na plataforma *Micaz MPR2400* estão apresentados na Tabela 2.3.

A Tabela 2.4 traz um sumário do consumo de energia do transceptor *Chipcon CC2420* operando a aproximadamente 3 V e com uma taxa de transferência de 250 *kbps* [10].

2.4.1.5 Plataforma Crossbow Mica2 MPR400

A plataforma *Mica2 MR400* utiliza o microcontrolador *Atmel ATmega 128L* [9] (o mesmo microcontrolador da plataforma *Micaz MPR2400* descrito na seção 2.4.1.3). O

Tabela 2.3. Consumo de energia do microcontrolador *Atmega 128L* [9]

Modo de Funcionamento		Corrente Elétrica (mA)
Ativo	4 MHz - 3 V	3.8
	8 MHz - 5 V	14.4
Inativo	4 MHz - 3 V	1.4
	8 MHz - 5 V	7.5
Sleep		< 0.015

Tabela 2.4. Consumo de energia do transceptor *Chipcon CC2420* operando a aproximadamente 3 V e com uma taxa de transferência de 250 *kbps* [10]

Modo de Funcionamento	Corrente Elétrica (mA)	Consumo de Energia (nJ/bit)
Recepção (E_{Rx})	19.7	236.4
Transmissão (E_{Tx})		
-25 dBm	8.5	102.0
-15 dBm	9.9	118.8
-10 dBm	11	132.0
-5 dBm	14	168.0
0 dBm	17.4	208.8

Figu

transceptor é o modelo *Chipcon CC1000*, que pode funcionar numa frequência que varia de 868 a 916 MHz. O alcance máximo do transceptor é 152 m e a taxa de transferência pode atingir até 76 $Kpbs$. O nó sensor é alimentado por duas baterias do tipo AA.

2.4.1.6 Consumo de energia da plataforma Crossbow Mica2 MPR400

Os dados referentes ao consumo do microcontrolador *Atmega 128L* usado na plataforma *Mica2 MPR400* já foram descritos na seção anterior (Tabela 2.3).

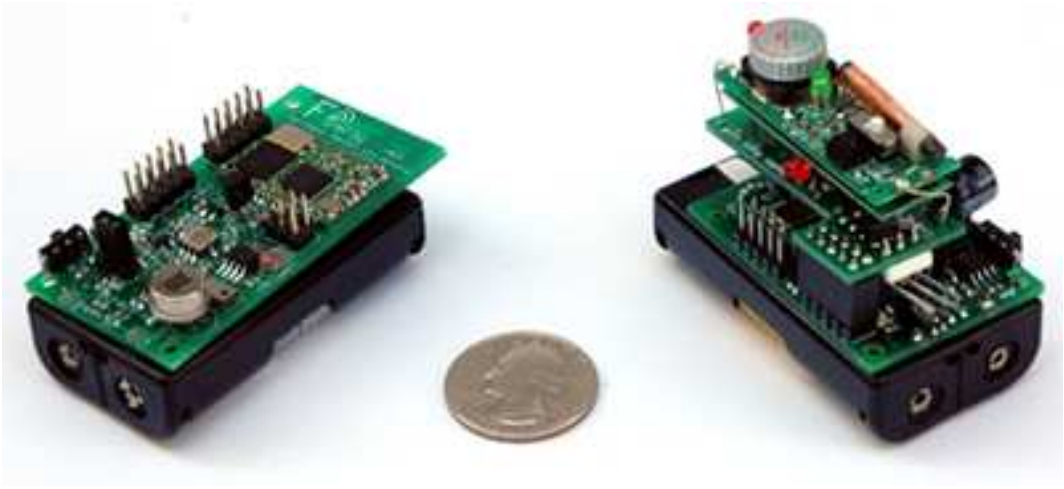
A Tabela 2.5 traz um sumário do consumo de energia do transceptor *Chipcon CC1000* operando a aproximadamente 3 V e com uma taxa de transferência de 76 *kbps* [11].

Pode-se observar nas Tabelas 2.2, 2.4 e 2.5 que a quantidade de energia consumida pelos dispositivos durante o recebimento de uma mensagem é significativa se comparada à quantidade de energia gasta com a transmissão. Isso reforça o argumento que define que o gasto de energia com o recebimento de uma mensagem deve ser considerado.

Tabela 2.5. Consumo de Energia do transceptor *Chipcon CC1000* operando a aproximadamente 3 V e com uma taxa de transferência de 76 *kbps* [11]

Modo de Funcionamento	Corrente Elétrica (mA)	Consumo de Energia (nJ/bit)
Recepção (E_{Rx})	7.4	292.1
Transmissão (E_{Tx})		
-20 dBm	5.3	209.2
-5 dBm	8.9	351.3
0 dBm	10.4	410.5
+5 dBm	14.8	584.2
+10 dBm	26.7	1053.9

Figura 2.5. Sensores *FireFly* (fonte: [4])



2.4.2 Plataforma FireFly

O *FireFly* [4, 13, 49], ilustrado na Figura 2.5, é um nó sensor de baixo custo desenvolvido pelo Departamento de Engenharia Elétrica e Computação da Universidade *Carnegie Mellon*. O dispositivo é equipado com o microcontrolador de 8 *bits Atmel Atmega32L*, que pode operar numa frequência de 1 a 8 MHz [12]. O *FireFly* utiliza o transceptor *Chipcon CC2420* [10] (o mesmo utilizado na plataforma *Crossbow Micaz MPR2400*). O dispositivo pode ser equipado com sensores de luminosidade, temperatura, áudio, acelerômetro (2 dimensões) e sensor de movimento.

Tabela 2.6. Consumo de energia do microcontrolador *Atmega 32L* [12]

Modo de Funcionamento		Corrente Elétrica (mA)
Ativo	1 MHz - 3 V	1.1
	4 MHz - 3 V	3.8
	8 MHz - 5 V	12
Inativo	1 MHz - 3 V	0.35
	4 MHz - 3 V	1.2
	8 MHz - 5 V	5.5
Sleep		0.010

O dispositivo possui 32 *Kbytes* de memória *ROM* e 2 *Kbytes* de memória *RAM*, além de um *slot* de expansão de memória (normalmente usado para armazenar dados coletados). A interface com um computador convencional é feita através de uma conexão do tipo USB.

2.4.2.1 Consumo da plataforma FireFly

Os dados referentes ao consumo de energia do microcontrolador *Atmega 32L* [12] usado na plataforma *FireFly* estão apresentados na Tabela 2.6.

O consumo de energia do transceptor *Chipcon CC2420* é apresentado na Tabela 2.4.

2.5 APLICAÇÕES DE RSSF

A integração entre microprocessadores e comunicação sem fio permite projetar e desenvolver aplicações para as mais variadas áreas: mecatrônica, indústria, medicina, agricultura, segurança, serviços, lazer, logística, dentre inúmeras outras [50].

A literatura é repleta de trabalhos que descrevem a utilização prática das RSSF. Em [13], os autores apresentam um estudo de caso que detalha a utilização de RSSF em uma aplicação de segurança do trabalho em minas de exploração de carvão. A rede é formada

por dois tipos de sensores: nós fixos instalados de forma aleatória por toda a extensão da mina e nós móveis que são transportados pelos operários mineradores. A função primária da rede é conhecer a localização física dos nós móveis (e consequentemente dos operários que estão carregando os sensores). Conhecer a localização física dos operários de uma mina é de fundamental importância em caso de um acidente na mina. As técnicas convencionais usadas para localizar os operários são bastante ineficientes: perfuração de dutos para utilização de câmeras e microfones, propagação de ondas, etc. Além de informar a sua localização na mina, os sensores móveis também têm a capacidade de enviar mensagens em difusão (*broadcast*) na rede. Assim, um operário pode alertar os outros sobre algum perigo ou pedir ajuda. Em condições normais, as informações coletadas na rede também são úteis, pois podem servir de insumo para análises técnicas e estratégicas: desempenho dos operários ou otimização de equipe de mineradores, por exemplo.

Na área da Mecatrônica, as RSSF podem ser usadas para automação industrial [14]. Tipicamente, a atividade industrial requer uma troca contínua de informações entre máquinas, computadores e operadores. A implantação de RSSF para atender a essa demanda pode ser uma opção viável.

Em [15] os autores apresentam um interessante estudo sobre a utilização das RSSF na área da Biomedicina. Nesse trabalho, um protótipo de retina artificial é proposto a partir de micro-sensores que se comunicam entre si através de um canal sem fio. A retina artificial simula o comportamento do órgão humano e pode ser usado para compensar deficiências visuais.

Outros trabalhos relatam a utilização das RSSF na área da Construção Civil. Em [16] os autores descrevem uma aplicação para o monitoramento de edificações prediais construída a partir da instalação de nós sensores (de temperatura, de movimento, de detecção de gás, etc) ao longo do prédio. As informações coletadas por esses nós sensores são enviadas ao longo da rede (através dos outros nós sensores) até uma estação de processamento, que analisa os dados recebidos.

Em [17] os autores apresentam uma aplicação com RSSF que permite monitorar o

índice de poluição do ar na cidade de Londres. A RSSF para essa aplicação é formada por nós móveis e estáticos. Os nós móveis são equipados com sensores capazes de medir a poluição do ar e fixados em automóveis públicos. Esses nós fazem o sensoriamento do índice de poluição do ar e enviam essas informações para uma estação base através de uma rede *ad hoc* secundária - formada por nós estáticos - instalada em formato de *grid* ao longo das ruas monitoradas. Essas informações são roteadas até a estação base que analisa os dados recebidos.

Em [18, 19] os autores descrevem a utilização de RSSF na área de controle ambiental. Uma aplicação que utiliza uma RSSF no monitoramento de florestas é decrita em [18]. Já [19] descreve a utilização de uma RSSF para captar informações sobre espécies de plantas em extinção. Essa aplicação monitora o ambiente onde as espécies das plantas se encontram através de nós equipados com sensores de temperatura, de medição de ventos, de umidade e de radiação solar. Os dados coletados são enviados através da rede até uma estação base.

No próximo capítulo serão apresentados os principais aspectos relacionados ao Controle de Topologia e ao efeito do *overhearing* no gasto de energia da rede.

CAPÍTULO 3

CONTROLE DE TOPOLOGIA

Esse capítulo discute o Controle de Topologia para Rede de Sensores Sem Fio. Serão apresentados os principais aspectos relacionados ao tema bem como uma discussão sobre os impactos do efeito *overhearing* no Controle de Topologia.

Como já foi mencionado na seção 2.2, uma das maiores restrições das RSSF é a limitação de energia dos nós sensores que formam a rede. A fonte de energia desses dispositivos normalmente são baterias e a reposição dessas pode ser inviável.

Uma alternativa para otimizar o uso desses recursos é reduzir a potência de transmissão dos nós preservando algumas propriedades da rede (conectividade, por exemplo). Essa técnica é conhecida como Controle de Topologia [25]. Encontrar a menor potência de transmissão para cada nó, preservando a rede conectada é um problema NP-Completo [30]. Assim, vários trabalhos propõem alternativas para o Controle de Topologia [5, 6, 28, 32, 51], que tentam, utilizando os recursos da RSSF de forma eficiente, ajustar a potência de transmissão dos nós da rede. Definir qual a melhor (menor) potência de transmissão dos nós sensores, utilizando de forma eficiente os recursos da rede, é, portanto, o desafio de um algoritmo de Controle de Topologia.

Esse trabalho está baseado numa técnica comumente utilizada por outros algoritmos de Controle de Topologia [5,6,28,32,51], em que a diminuição da potência está associada à redução do número de vizinhos alcançados diretamente por um nó. Ou seja, a eliminação de vizinhos pode possibilitar a redução da potência de transmissão de alguns nós.

O restante desse capítulo está dividido da seguinte forma: a seção 3.1 traz uma discussão mais abrangente sobre o Controle de Topologia em Rede de Sensores Sem Fio e introduz alguns conceitos e definições relacionados a esse tema: Grafo de Conectividade, Caminho de Custo Mínimo, etc. A seção 3.2 discute as principais técnicas utilizadas na

implementação de algoritmos de Controle de Topologia. A seção 3.3 apresenta o modelo de energia considerado nesse estudo bem como a importância de se considerar o custo com o recebimento de mensagens. A seção 3.4 introduz uma discussão sobre os efeitos do *overhearing* no Controle de Topologia. Em seguida, na seção 3.5, são apresentadas e discutidas algumas das principais soluções que influenciaram esse trabalho. A seção 3.6 apresenta outros trabalhos relacionados.

3.1 CONTROLE DE TOPOLOGIA

O algoritmo de Controle de Topologia apresentado nesse estudo se baseia na seguinte estratégia: cada nó determina a sua potência de transmissão a partir da eliminação de alguns nós vizinhos. Para entender melhor essa técnica, serão apresentados alguns conceitos introduzidos por [6, 32].

3.1.1 Grafo de Conectividade G

Segundo [6,32], uma RSSF pode ser representada através de um grafo de conectividade $G = (V, E)$ onde V é o conjunto de nós sensores da rede e E é o conjunto de arestas que representam as conexões (enlaces) entre os nós. Dois nós u e v estão ligados através de uma aresta $(u, v) \in E$ se for possível para u transmitir uma mensagem para v. Está sendo considerado que a relação de alcance entre todo par de vértices u e v é simétrica, ou seja, se u alcança v, então v alcança u ($(u, v) \in E \Leftrightarrow (v, u) \in E$).

O grafo de conectividade de potência máxima $G_{max} = (V, E_{max})$ representa o grafo quando todos os nós utilizam a potência máxima de transmissão, P_{max}. Utilizar a potência máxima de transmissão pode não ser adequado para a RSSF. Nesse sentido, um algoritmo de Controle de Topologia pode ser usado para determinar um subgrafo G' de G_{max} tal que:

1) O conjunto V é o mesmo em G_{max} e G';

2) Se existe um caminho entre u e v em G_{max}, há também um caminho entre u e v em G'; e

3) Os nós em G' (possivelmente) transmitem para os seus vizinhos usando uma potência menor do que a usada em G_{max}.

O algoritmo proposto nesse trabalho considera uma propriedade adicional:

4) G' deve ter a propriedade do custo mínimo (descrita a seguir).

3.1.2 Caminho de Custo Mínimo

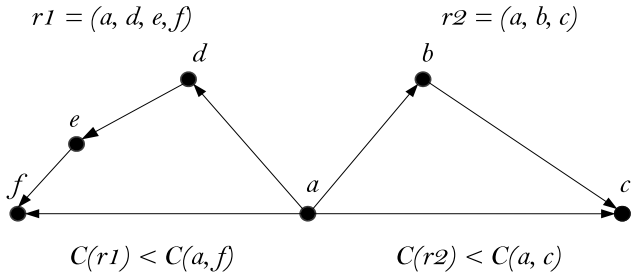
Um caminho $r = \langle u_0, .., u_k \rangle$ no grafo $G = (V, E)$ é uma lista ordenada de nós $(u_0, u_1, .., u_{k-1}, u_k)$ tal que $\forall i,\ 0 \leq i < k,\ (u_i, u_{i+1}) \in E$. O comprimento de r é definido por $|r| = k$.

O custo de uma aresta (u, v), $C(u, v)$, pode ser definido de várias formas. Em [6, 32], por exemplo, os autores consideram que o custo de uma aresta é proporcional à distância entre os nós elevada a um determinado valor. Em [5], por sua vez, os autores consideram um modelo de energia onde o custo $C(u, v)$ de uma aresta (u, v) está associado também ao número de nós alcançáveis por u quando este envia uma mensagem para v. O modelo de energia considerado nesse trabalho será apresentado na próxima seção. Por enquanto é necessário saber que o custo de um caminho r é a soma do custo de todas as arestas que compõem r:

$$C(r) = \sum_{i=0}^{k-1} C(u_i, u_{i+1}) \tag{3.1}$$

Um caminho $r_{min} = \langle u_0, .., u_k \rangle$ é um caminho de custo mínimo entre os nós u_0 e u_k se $C(r_{min}) \leq C(r')$ para todo caminho r' que conecta u_0 a u_k em G.

Figura 3.1. Exemplo de um grafo de conectividade com arestas redundantes



Um subgrafo $G_{min} = (V, E)$ de G_{max} tem a propriedade do custo mínimo se, para todo par de nós $u, v \in V$, existir um caminho $r_{min} = \langle u, .., v \rangle$ em G_{min} com mesmo custo de um caminho de custo mínimo entre u e v em G_{max}.

3.1.3 Arestas Redundantes

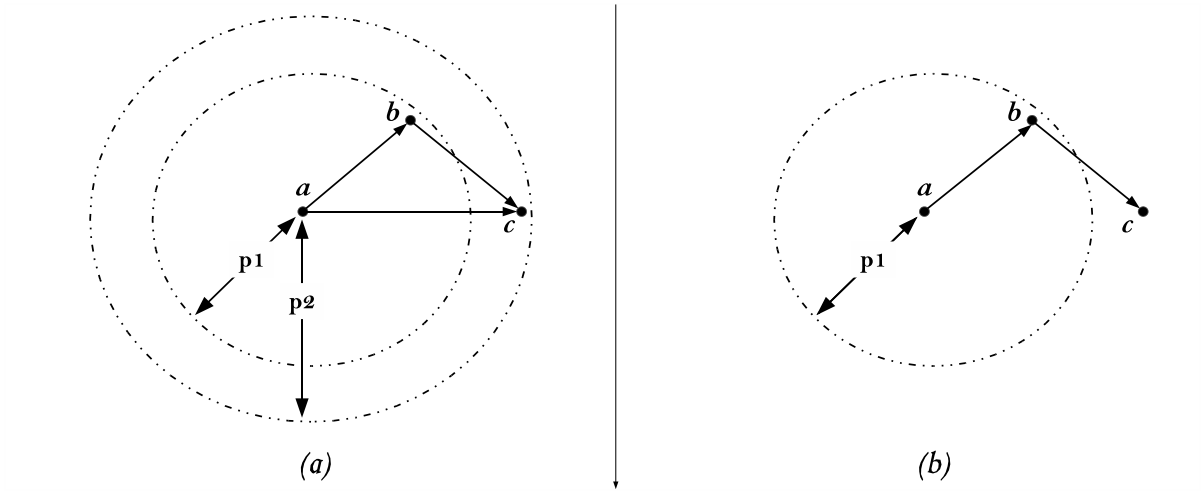
Transmitir uma mensagem através de um caminho r (para $|r| = k$ e $k \geq 2$) pode consumir menos energia do que transmitir através de um caminho r' (para $|r'| = 1$). Quando isso ocorre ($C(r) \leq C(r')$), r' é considerada uma aresta *redundante*. Dessa forma, uma aresta $(u, v) \in E$ é *k-redundante* se e somente se existir um caminho r em G tal que $|r| = k$, $k > 1$ e $C(r) \leq C(u, v)$. Um exemplo de grafo de conectividade com arestas redundantes pode ser visto na Figura 3.1.

No cenário apresentado na Figura 3.1, as arestas (a, f) e (a, c) são, respectivamente, *3-redundantes* e *2-redundantes*. Para o nó a, transmitir uma mensagem através dos caminhos $r1$ e $r2$ consome uma quantidade de energia menor do que transmitir diretamente para os nós f e c.

3.1.4 Ajuste de Potência no Controle de Topologia

Como já foi mencionado, a redução do número de arestas em G_{max} pode permitir que alguns nós ajustem (reduzam) a sua potência de transmissão. A Figura 3.2 ilustra um

Figura 3.2. Ajuste de potência de transmissão realizado pelo Controle de Topologia

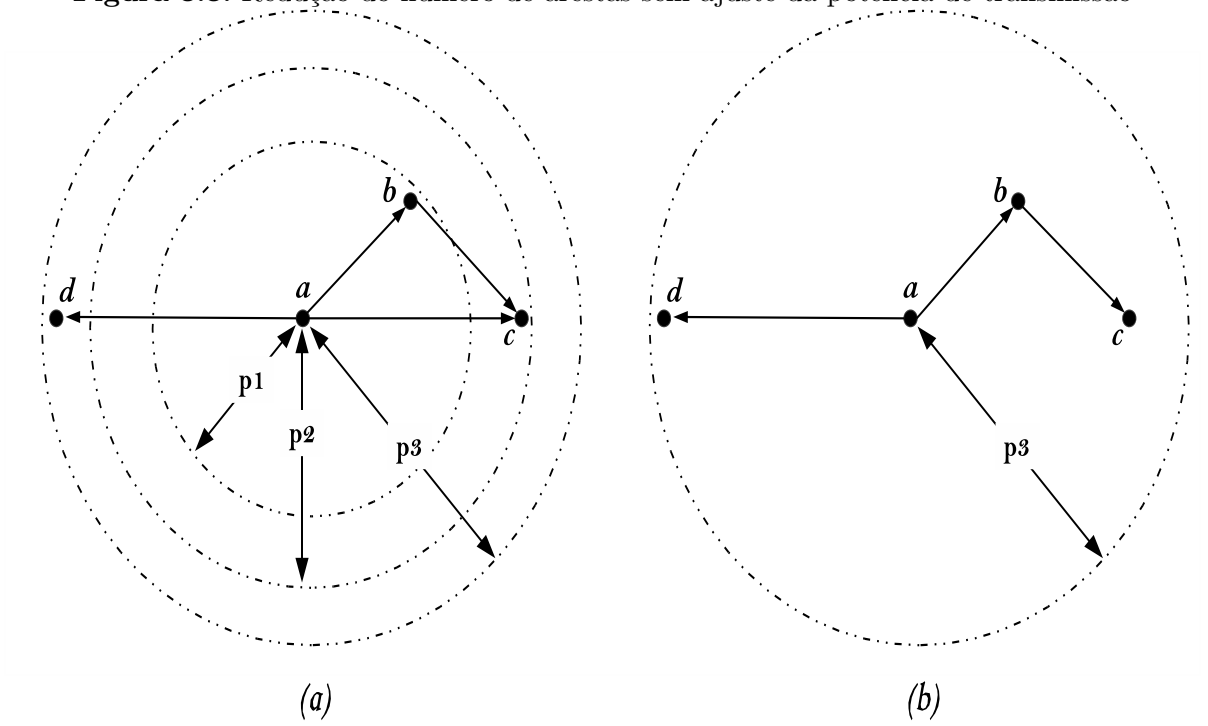


cenário onde a eliminação de uma aresta permite que o nó reduza a sua potência.

Considere os seguintes nós e arestas do grafo, como ilustrado na Figura 3.2.a. Nesse cenário, o nó a está configurado para alcançar os nós b e c da rede e sua potência de transmissão é p_2 (potência mínima necessária para alcançar todos os seus vizinhos). Considerando que o custo de transmissão $C(\langle a, b, c\rangle)$ é menor que o custo $C(\langle a, c\rangle)$, o algoritmo de Controle de Topologia atua no grafo original eliminando a aresta (a, c). A potência de transmissão do nó a pode ser reduzida para p_1 (a potência mínima necessária para atingir o nó b) conforme indicado na Figura 3.2.b.

Existem casos, no entanto, onde a eliminação de arestas do grafo de conectividade não garante a redução da potência de transmissão para alguns nós. Considere, por exemplo, a Figura 3.3.a (uma variação da topologia apresentada na Figura 3.2.a). No grafo de conectividade máxima da rede, o nó a se comunica diretamente com os nós b, c e d e sua potência de transmissão está ajustada para p_3 (a potência mínima necessária para alcançar todos os seus vizinhos). O algoritmo de Controle de Topologia pode atuar no grafo G_{max} e gerar um subgrafo G_{min} conforme ilustrado na Figura 3.3.b. A aresta (a, c) foi eliminada e o nó a passou a se comunicar diretamente apenas com os nós b e d. Nesse

Figura 3.3. Redução do número de arestas sem ajuste da potência de transmissão



caso, apesar de ter ocorrido uma redução no número de vizinhos, a potência do nó a continua sendo p_3 (a potência mínima necessária para alcançar seu vizinho mais distante - nesse caso o nó d).

Portanto, o Controle de Topologia permite a redução da potência de transmissão de um nó apenas quando os nós mais distantes de sua vizinhança são desconsiderados (removidos da sua lista de vizinhos).

3.1.5 Grafo de Conectividade G_{min}

Assim, G_{min} é um subgrafo de G_{max} que continua fortemente conexo e que contém - normalmente - menos arestas que G_{max}. Caso a propriedade de custo mínimo seja mantida no grafo G_{min}, então G_{min} pode ser usado no lugar de G_{max} para, por exemplo, encontrar o menor caminho entre qualquer par de nós da rede [32].

3.2 ESTRATÉGIAS DE CONTROLE DE TOPOLOGIA

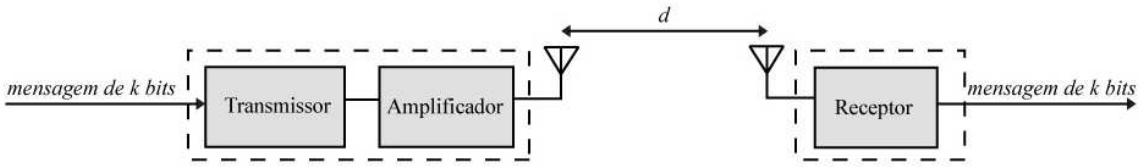
Existem diversas técnicas para encontrar o grafo G_{min}. A seguir serão descritas algumas dessas estratégias.

Soluções *Offline* e Adaptativas. Quando a posição dos nós é conhecida antecipadamente, o grafo G_{min} pode ser encontrado antes de os dispositivos serem dispostos no ambiente. No entanto, em cenários mais realistas, os nós são distribuídos de forma aleatória e as suas localizações só são conhecidas em tempo de execução. Em cenários mais específicos, a localização de um nó pode variar ao decorrer do tempo. Assim, a capacidade de ser adaptativo é um desafio adicional para um algoritmo de Controle de Topologia [25].

Soluções Centralizadas, Distribuídas e Localizadas. Uma solução adaptativa para encontrar G_{min} pode ser implementada através da centralização do cálculo das melhores rotas em um único nó (Estação Base, por exemplo). Uma possibilidade seria utilizar uma heurística em que cada nó da rede envia sua localização para a EB. A partir dessas informações (ou seja, a topologia total da rede), uma implementação do algoritmo Dijkstra [52], Bellman-Ford [53] ou Floyd-Warshall [54] pode ser usada para descobrir o grafo com as melhores rotas entre todo par de nós do grafo original. As soluções centralizadas, no entanto, mostram-se ineficientes por pelo menos duas razões: primeiro porque exigiria uma grande quantidade de mensagens trafegando na rede (todos os nós enviando e recebendo mensagens para a Estação Base). Outro problema - não menos importante - seria o tempo computacional necessário para calcular as melhores rotas em uma rede de larga escala (com centenas e até milhares de nós).

Uma estratégia mais extensível é usar um algoritmo distribuído e localizado de Controle de Topologia. Um algoritmo de Controle de Topologia é considerado localizado quando cada nó, utilizando apenas informações locais da rede (um subconjunto de G_{max}), consegue determinar a sua própria potência de transmissão. Vários trabalhos propõem soluções distribuídas e localizadas para o Controle de Topologia [5, 6, 28, 29, 31, 32].

Figura 3.4. Modelo de energia (baseada em [5])



3.3 MODELO DE ENERGIA

Será considerado nesse estudo o modelo de energia proposto em [38]. Este modelo considera que o dispositivo de transmissão e recebimento de mensagens (transceptor) é estruturado como ilustrado na Figura 3.4. Os nós consomem energia durante a transmissão e recebimento de mensagens. O custo com processamento é desconsiderado nesse modelo (conforme justificativa apresentada no final dessa seção).

Pode-se dividir o gasto de energia em duas categorias: (a) energia consumida pelos dispositivos Transmissor e Receptor, que depende apenas do tamanho (em *bits*) da mensagem que está sendo transmitida e (b) energia consumida pelo Amplificador do sinal, que depende também da potência de transmissão usada.

A partir de agora, será definida a notação usada no restante desse texto para representar o consumo dos nós da rede. A energia consumida na transmissão de uma mensagem de tamanho l bits de um nó p para um nó q é dada por $E_{Tx}(l, p, q)$:

$$E_{Tx}(l, p, q) = l.E_{Tx-elec} + l.\varepsilon_{amp}(p, q) \tag{3.2}$$

$$\varepsilon_{amp}(p, q) = \epsilon.d^{\alpha}, \tag{3.3}$$

Onde: $E_{Tx-elec}$ é a energia consumida pelo transmissor para realizar o processamento do sinal digital (codificação, modularização, filtragem, etc) antes de esse ser enviado

para o amplificador do dispositivo; ε_{amp} é a energia consumida pelo Amplificador; d é a distância entre os nós p e q; α é um coeficiente de perda do canal (*power loss exponent*) ($2 \leq \alpha < 4$); e ϵ é um parâmetro característico do transmissor e do canal [55].

O modelo de perda de sinal considerado é o *FSPL* (*free space propagation model*) [55]. Assim, as mensagens são enviadas através do caminho que representa uma reta entre o nó transmissor e o receptor.

A energia consumida pelo receptor para realizar o processamento do sinal digital, por *bit*, é dada por $E_{Rx}(l)$:

$$E_{Rx}(l) = l.E_{Rx-elec}, \tag{3.4}$$

Onde: $E_{Rx-elec}$ é a energia consumida pelo receptor para realizar o processamento do sinal digital (decodificação, modularização, filtragem, etc).

$E_{Tx-elec}$ e $E_{Rx-elec}$ são valores fixos e específicos de cada dispositivo.

A equação do custo de uma aresta (u, v), $C(u, v)$, apresentada inicialmente na Equação 3.1, pode ser redefinida da seguinte forma (considerando que apenas o nó v recebe a mensagem):

$$C(u, v) = E_{Tx-elec(u)} + \varepsilon_{amp(u,v)} + E_{Rx-elec(v)} \tag{3.5}$$

Nesse caso, $E_{Tx-elec(u)}$ e $\varepsilon_{amp(u,v)}$ representam a energia consumida pelo nó transmissor u enquanto $E_{Rx-elec(v)}$ é a energia gasta pelo nó receptor v. Na equação 3.5 está sendo considerado que apenas o nó v recebe a mensagem. O custo de recebimento dos outros nós alcançados por u quando esse transmite para v será considerado na seção 3.4.

Em [38] os autores consideram que: $E_{Tx-elec} = E_{Rx-elec} = 50nJ/bit$ e que $\varepsilon_{amp} = 100pJ/bit/d^2$. Neste trabalho, no entanto, serão considerados os valores descritos nas

tabelas 2.2, 2.4 e 2.5, que representam o consumo dos transceptores *Atmel RF230*, *Chipcon CC2420* e *Chipcon CC1000* (apresentados na seção 2.4). Esses valores indicam que os nós consomem, proporcionalmente, mais energia com recebimento do que o modelo considerado em [38].

Com relação ao gasto com processamento, [38] considera que esse custo é pouco significativo quando comparado ao consumo de energia com transmissão/recepção e não o considera no modelo de energia proposto.

Esses valores sugerem que o gasto com processamento tem impacto no Controle de Topologia e, portanto, deveria ser considerado. Esse trabalho, como já foi dito, utiliza o modelo de energia proposto em [38] e, portanto, não considera o gasto com processamento.

Para identificar os impactos do gasto com processamento no Controle de Topologia, seria necessária a utilização de um modelo de energia que considerasse o gasto com processamento durante a transmissão/recepção de uma mensagem.

A próxima seção discute como o efeito *overhearing* influencia no consumo de energia da aresta (u, v) e apresenta uma equação mais completa para $C(u, v)$.

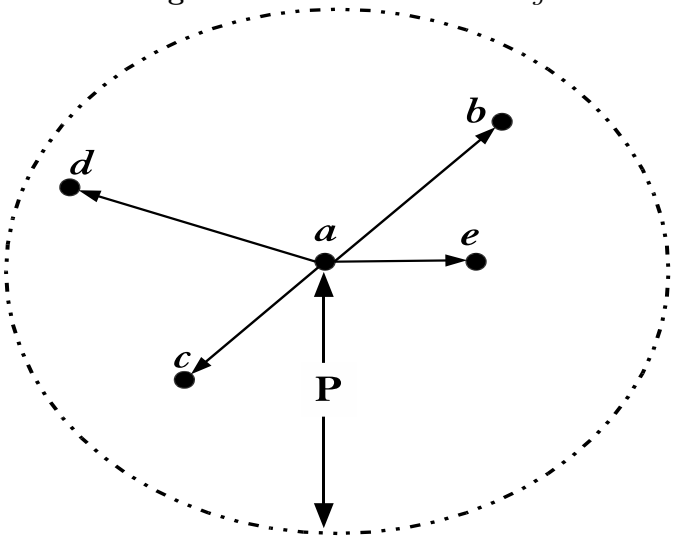
3.4 OVERHEARING

Em rede de comunicação de dados sem fio, *overhearing* consiste no recebimento desnecessário de uma mensagem. Ou seja, esse fato ocorre quando um nó recebe uma mensagem que não é destinada a ele.

A ocorrência de *overhearing* durante a transmissão de uma mensagem não compromete o funcionamento da rede. Isso porque normalmente os protocolos de roteamento implementam rotinas para descartar mensagens que não são destinadas àquele nó. No entanto, o simples fato de receber uma mensagem já consome uma quantidade de energia do nó que não deve ser desprezada.

Como já foi mencionado, no modelo de energia considerado, quando um nó p envia

Figura 3.5. Efeito *Overhearing*



uma mensagem para um nó q com potência P, todos os nós alcançáveis num raio circular com centro em p recebem a mensagem. Quando um nó recebe uma mensagem que não é destinada a ele, a mensagem é descartada. O custo de *overhearing* está relacionado à quantidade de energia gasta por todos os nós da rede que receberam uma mensagem que não estava destinada a eles. Considere a topologia representada na Figura 3.5. Quando o nó a envia uma mensagem para o nó e com a potência P, essa é recebida por todos os nós alcançáveis - nesse caso, os nós b, c, d e e. Esses três primeiros, ao receberem a mensagem e processá-la, verificam que não são o destinatário da mensagem e, portanto, descartam-na. Nessa transmissão houve, portanto, um custo com *overhearing* dos nós b, c e d.

O custo da aresta (a, e), quando a transmite usando a potência P, pode ser representado por:

$$C(a,e) = E_{Tx-elec(a)} + \varepsilon_{amp(a,e)} + E_{Rx-elec(b)} + E_{Rx-elec(c)} + E_{Rx-elec(d)} + E_{Rx-elec(e)} \quad (3.6)$$

Se o efeito *overhearing* for considerado, a equação 3.5 que descreve o custo de uma

aresta (u, v) pode ser reescrita da seguinte forma:

$$C(u,v) = E_{Tx-elec(u)} + \varepsilon_{amp(u)} + (OH(u,v)+1)E_{Rx-elec} \quad (3.7)$$

Nessa equação, $OH(u, v)$ é a quantidade de nós que receberam a mensagem mas não tinham interesse na mesma.

A seguir serão apresentados os trabalhos mais relacionados ao problema central desta dissertação.

3.5 TRABALHOS RELACIONADOS

Os algoritmos apresentados em [5, 6, 28, 32] são os mais relacionados ao algoritmo proposto nesse estudo. Trata-se de algoritmos de Controle de Topologia distribuídos e localizados capazes de reduzir o grafo de conectividade G_{max} a um subgrafo G_{min} que tem a propriedade do caminho de custo mínimo.

De forma genérica, esses algoritmos funcionam da seguinte forma: cada nó da rede executa um algoritmo, que pode ser dividido em duas fases: na primeira fase, o nó troca mensagens com os nós de sua vizinhança. Nessas mensagens, os nós informam as suas localizações físicas na rede. Ao final dessa fase, o nó tem uma visão parcial da rede. Na segunda fase, o nó determina/calcula a sua potência de transmissão baseado no grafo de conectividade (parcial e local) construído na primeira fase.

MECN [32] e *SMECN* [6] são algoritmos bastante similares. Os autores apresentam uma estratégia onde os nós tentam trocar informações utilizando uma potência preferencialmente menor que a potência máxima.

Inicialmente, o nó utiliza uma potência $P_{min} < P_{max}$. A potência é incrementada até que o nó conheça uma quantidade suficiente de nós. Essa estratégia tenta evitar que os

nós utilizem a potência máxima de transmissão P_{max} na primeira fase do algoritmo. Os algoritmos apresentam ainda uma estratégia para reduzir o grafo de conectividade G_{max} a partir da detecção e eliminação das arestas *2-redundantes* (ou seja, arestas *k-redundantes* para $k = 2$). O algoritmo *SMECN* é ilustrado na Figura 3.6.

Figura 3.6. Algoritmo *SMECN* executado pelo processo u [6].

Algorithm *SMECN*
01 $p = p_0$;
02 $A = 0$;
03 $NonNbrs = 0$;
04 $\eta = F(u, p_{max})$;
05 while $F(u, p) \not\supseteq \eta$ do
06 $p = Increase(p)$;
07 Envia uma mensagem $Hello$ para toda a rede (broadcast) com a potëncia p e os $Acks$ acumulados;
08 $M = \{v | Loc(v) \in F(u, pot), v \notin A, v \neq u\}$;
09 $A = A \bigcup M$;
10 for each $v \in M$ do
11 for each $w \in A$ do
12 if $Loc(v) \in R_{u \to w}$ then
13 $NonNbrs = NonNbrs \bigcup \{v\}$;
14 else if $Loc(w) \in R_{u \to v}$ then
15 $NonNbrs = NonNbrs \bigcup \{w\}$;
16 $\eta = \eta \cap \bigcap_{v \in M} (F(u, p_{max}) - R_{u \to v})$;
17 $N(u) = A - NonNbrs$;
18 $p(u) = min\{p : F(u, pot) \supseteq \eta\}$

Considere que o algoritmo *SMECN* está sendo executado pelo processo u. p representa a potência de u e é iniciada com um valor p_0 (linha 1, Figura 3.6). A representa o conjunto dos nós que o processo u identificou na sua vizinhança. O processo u inicia uma iteração (linha 5, Figura 3.6) para identificar os nós de sua vizinhança. Uma mensagem *broadcast* é enviada com potência p solicitando que os nós alcançados informem as suas localizações (linha 7, Figura 3.6). O valor de p é incrementado a cada iteração através da rotina *Increase* (linha 6, Figura 3.6). M representa o conjunto dos novos vizinhos alcançados a cada iteração. A cada iteração o algoritmo verifica se existe alguma aresta *2-redundante* no conjunto de arestas do grafo de conectividade $G = (A, E)$. Caso o algoritmo encontre

uma aresta (u, v) *2-redundante*, v é incluído no conjunto $NonNbrs$, que representa os nós que não fazem parte da vizinhança *otimizada* de u (linhas 10 a 15, Figura 3.6). Essa iteração (iniciada na linha 5, Figura 3.6) se repete até que um número suficiente de nós seja identificado ou que p atinja o valor p_{max}. O algoritmo calcula $N(u)$, que representa a vizinhança *otimizada* do nó u, a partir dos conjuntos A e $NonNbrs$ (linha 17, Figura 3.6). O valor $p(u)$, que representa o valor da potência mínima necessária para alcançar todos os nós em $N(u)$, é calculado ao final do algoritmo (linha 18, Figura 3.6).

A diferença básica entre os algoritmos *MECN* e *SMECN* é que esse último otimiza o cálculo das arestas redundantes proposto pelo primeiro. A estrutura desses algoritmos serviram de base para o algoritmo proposto nesse trabalho.

Vale destacar, no entanto, que os algoritmos *MECN* e *SMECN* não consideram o custo de energia com *overhearing*. Ao invés disso, o modelo de energia utilizado em [32] e [6] considera apenas o custo de transmissão (do nó emissor) e de recebimento (de um único nó receptor).

Os algoritmos propostos em [5, 28] utilizam outra abordagem para construir o grafo de conectividade da rede (primeira fase do algoritmo). Os nós trocam mensagens informando, além de sua localização, a localização de seus vizinhos. Isso permite que cada nó tenha uma visão (local) um pouco mais abrangente da rede.

O algoritmo proposto em [28] consegue eliminar as arestas *k-redundantes* do grafo G_{max} detectadas com informações locais e, adicionalmente, evita a convergência do fluxo de dados em determinados nós da rede. Com essa estratégia, o algoritmo trata também o problema da morte prematura de nós. O custo com *overhearing*, no entanto, não é considerado em [28].

Já o algoritmo proposto em [5] traz como contribuição o fato de utilizar um modelo de energia que considera o custo de energia com o *overhearing*. Esse algoritmo, no entanto, elimina apenas as arestas *2-redundantes* do grafo G_{max}.

O objetivo do algoritmo proposto nesse trabalho é encontrar, de forma distribuída e localizada, um subgrafo G_{min} de G_{max} considerando o custo de energia com o efeito

overhearing e que elimine todas as arestas *k-redundantes* identificadas com as informações locais de cada nó. G_{min} deve ser fortemente conexo (desde que G_{max} também o seja) e ter a propriedade do caminho mínimo. Apesar de esse estudo ter sido influenciado pelos trabalhos [5, 6, 28, 32], o algoritmo proposto se diferencia desses pelos seguintes aspectos: [6, 28, 32] não consideram o custo com *overhearing* e [5, 6, 32] eliminam apenas as arestas *2-redundantes*.

3.6 OUTROS TRABALHOS RELACIONADOS

Além dos trabalhos já citados [5,6,28,32], existem vários outros estudos que descrevem técnicas de Controle de Topologia para reduzir o consumo de energia em uma RSSF.

Em [56, 57] os autores apresentam algoritmos de Controle de Topologia que calculam de forma homogênea a potência de transmissão da rede. Nesse modelo, todos os nós são configurados com a mesma potência de transmissão p (*CTR - Critical Transmitting Range*). A desvantagem dessa estratégia é que ela induz a uma solução centralizada de Controle de Topologia. Já foram apresentadas na seção 3.2 algumas desvantagens das soluções centralizadas para as RSSF. Além disso, o ajuste homogêneo da potência de transmissão (a mesma potência p para todos os nós da rede) não é tão eficiente quanto o ajuste individual de cada nó [5].

Um algoritmo de Controle de Topologia que reduz o número de arestas no grafo de conectividade da rede usando uma heurística baseada na triangulação de Delaunay é descrito em [58]. A principal limitação dessa solução é que essa técnica não garante a preservação da conectividade da rede [58].

Em [59] os autores propõem uma solução para o Controle de Topologia baseada numa alteração no protocolo IEEE 802.11 [60] que reduz a interferência na rede. Os nós podem ajustar suas potências de transmissão a partir da análise das mensagens *RTS/CTS* que recebem da rede. O algoritmo apresentado em [61] segue uma abordagem semelhante e propõe a eliminação das arestas *2-redundantes* do grafo de conexão total da rede (G_{max})

a partir de uma modificação no mecanismo de detecção de colisão da camada MAC. Já os trabalhos [62,63] propõem técnicas para conservar energia na rede através do desligamento de alguns nós da rede.

Além do Controle de Topologia, existem outras estratégias para a utilização mais eficiente dos recursos de energia em uma RSSF.

Em [42] os autores propõem uma família de protocolos adaptativos (denominados protocolos *SPIN*) que utiliza a técnica de negociação para disseminar as informações de um nó sensor para os outros nós da rede. Nos protocolos *SPIN*, os nós utilizam descritores, chamados de meta-dados, para nomear seus dados. Quando um nó u tem uma mensagem m para ser enviada, ele envia uma mensagem contendo o meta-dado de m para a rede. Os nós interessados na mensagem m enviam uma mensagem de interesse para u. O nó u, então, envia m apenas para os nós interessados.

Uma técnica conhecida como *Difusão Direcionada* é proposta em [40]. Na *Difusão Direcionada* a transmissão é iniciada quando um nó (tipicamente a estação base) envia uma requisição de sensoriamento na rede. Os nós que podem atender a requisição respondem com uma mensagem contendo a informação solicitada. Nessa abordagem, nenhuma mensagem (exceto as mensagens de interesse) é enviada na rede sem que exista um nó interessado. O objetivo desta estratégia é evitar que mensagens sejam propagadas na rede sem que haja um nó interessado. Outras variações da *Difusão Direcionada* são propostas em [43, 64–67].

O algoritmo *LEACH* [38] propõe a divisão da rede em *clusters*. Os dados dentro de um *cluster* são agregados por um nó líder, o único nó do *cluster* que se comunica diretamente com a estação base. Periodicamente os *clusters* são reconfigurados e outros nós assumem o papel de líder. Essa técnica reduz a quantidade de nós que se comunicam com a estação base. Adicionalmente, o nó líder pode realizar a agregação das informações geradas em um *cluster* e reduzir a quantidade de mensagens enviadas à EB. Outros trabalhos [68–71] propõem algoritmos baseados no *LEACH*.

Em [39] os autores descrevem um algoritmo denominado *PEGASIS (Power-Efficient*

Gathering in Sensor Information Systems). Esse algoritmo propõe a seguinte estratégia: a topologia da rede é configurada numa espécie de corrente: os enlaces são unidirecionais e cada nó se comunica apenas com um vizinho. Apenas um nó da rede (denominado nó líder) se comunica com a estação base. As informações coletadas pelos nós sensores são enviadas até o nó líder. Esse, por sua vez, agrega os dados coletados e os envia, em uma única mensagem, para a estação base. A rede é reconfigurada num intervalo de tempo predefinido para evitar que um único nó assuma o papel de líder.

No próximo capítulo será apresentada uma solução de Controle de Topologia para RSSF que considera o consumo de energia do efeito *overhearing* e que reduz todas as arestas *k-redundantes* detectáveis localmente. O algoritmo é completamente distribuído e localizado e consegue calcular a potência de transmissão para cada nó a partir de informações locais sobre a topologia da rede (subconjunto de G_{max}). Os resultados das simulações realizadas que mostram a eficiência do algoritmo serão também apresentados no próximo capítulo.

CAPÍTULO 4

UMA PROPOSTA PARA CONTROLE DE TOPOLOGIA

É apresentado nesse capítulo um algoritmo distribuído e localizado para Controle de Topologia. O algoritmo proposto reduz o grafo de conectividade G_{max} a um subgrafo G_{min} eliminando as arestas redundantes (*k-redundantes*) encontradas localmente e garante a conectividade do grafo original e a propriedade do caminho mínimo. G_{min} é construído a partir do cálculo da *vizinhança reduzida* de cada nó. Achar esse conjunto (reduzido) de nós e permitir que os nós possam reduzir as potências de transmissão (considerando o efeito *overhearing*) é o objetivo do algoritmo proposto nesse capítulo.

4.1 MODELO DE REDE

O modelo de rede adotado nesse trabalho assume que os nós estão distribuídos de forma aleatória em uma área bidimensional. A localização dos nós é dada pelas suas coordenadas x, y e dois nós distintos possuem coordenadas distintas. Cada nó conhece a sua localização geográfica (essa informação pode ser obtida através de algum sistema de posicionamento, como o GPS, ou usando técnicas como a triangulação [72]).

Os canais de comunicação entre os nós são confiáveis, ou seja, as mensagens enviadas através de um canal certamente serão entregues não corrompidas e não duplicadas e cada mensagem é recebida uma única vez. Os processos não estão sujeitos a falhas. O raio de alcance de transmissão é circular (com o nó transmissor no centro do círculo) e a relação de alcance entre os nós é simétrica. Ou seja, para qualquer par de nós p e q, se p alcança q então q alcança p. Está sendo considerado também que todos os nós da rede possuem a mesma potência máxima de transmissão P_{max}. Cada transmissão pode ser feita utilizando uma potência no intervalo (fechado) entre P_{min} e P_{max}. O modelo de redes considera ainda que, quando uma mensagem é enviada por um nó, todos os nós que

estiverem no raio de alcance deste nó transmissor receberão (escutarão) a mensagem.

A RSSF possui uma única estação base estática (EB) que é alcançada por pelo menos um nó. Tanto a estação base quanto os nós da rede são estáticos. O grafo que representa a rede é um grafo direcionado fortemente conexo, ou seja, para qualquer par de nós p e q da rede, existe um caminho interligando esses dois nós.

Os nós no modelo de redes considerado acumulam duas funções. Além de atuarem como nós sensores propriamente ditos, também atuam como roteadores de mensagens para outros nós da rede. Como o grafo que modela a rede é fortemente conexo, todos os nós da rede alcançam a EB através de pelo menos um caminho. É possível existir múltiplos caminhos conectando um nó à EB (*multipath*).

4.2 DESCRIÇÃO DO ALGORITMO

O algoritmo apresentado nesse trabalho funciona, de maneira geral, da seguinte forma: cada nó executa o algoritmo, que, utilizando apenas informações locais da rede (um subgrafo de G_{max}), calcula um conjunto reduzido de vizinhos. A partir desse conjunto (tipicamente menor que o conjunto de vizinhos em G_{max}), o algoritmo ajusta a potência de transmissão do nó (de acordo com a estratégia discutida em 3.1.4).

Para cada nó da rede, existirá um processo associado que controlará o seu comportamento. A relação entre *nós* e *processos* é de um-para-um. Assim, em uma rede com n nós, existirão n processos: p_1, p_2, p_3,..., p_n.

Cada nó da rede executa o mesmo algoritmo, que está dividido em três fases. Na primeira fase todos os nós da rede enviam uma mensagem *broadcast* (difusão) para a rede, divulgando a sua localização - suas coordenadas x e y. Depois que todas as mensagens são recebidas e processadas, cada nó sabe quais são seus vizinhos (aqueles nós que o alcançam diretamente) bem como as suas localizações.

Na segunda fase, cada nó envia outra mensagem *broadcast* informando a localização dos nós que ele conhece (sua vizinhança). Depois de receber essas mensagens de seus

vizinhos diretos, cada nó constrói um grafo (direcionado) com uma visão parcial (local) de sua vizinhança (em dois saltos). O grafo construído por um processo p será chamado $G_p^c(V_p^c, E_p^c)$, em que: V_p^c é o conjunto dos nós conhecidos por p (seus vizinhos diretos e os vizinhos em dois saltos); e E_p^c é o conjunto de arestas que ligam os nós em V_p^c. Ou seja, para $r \in V_p^c$ e $s \in V_p^c$, $(r, s) \in E_p^c$ se e somente se s for vizinho direto de r e $r = p$ ou vizinho direto de p.

Na terceira fase, cada nó calcula qual a menor rota para cada um de seus vizinhos. Assim, dado um par de nós p, q onde $q \in V_p^c$, p calcula o menor caminho até q avaliando todas as arestas em E_p^c. Cada aresta desse conjunto tem um custo (peso) associado. O peso atribuído a cada aresta está relacionado à distância entre os nós que compõem a aresta e à quantidade de nós que recebem a mensagem. Portanto, o custo de envio de uma mensagem por uma aresta (r, s) é dado por:

$$Cost(r, s) = E_{Tx-elec} + \epsilon.d^{\alpha} + E_{Rx-elec}.h, \tag{4.1}$$

Os elementos $E_{Tx-elec}$, $E_{Rx-elec}$, ϵ e α já foram descritos na seção 3.3. d representa a distância entre os nós r e s. h é a quantidade de nós que recebem a mensagem, ou seja, a quantidade de vizinhos de p que estão a uma distância menor ou igual a d. Observe que o custo que foi associado à aresta (r, s) é a soma do custo de transmissão (que varia em função da distância entre os nós) e do custo de recebimento da mensagem (que varia em função da quantidade de vizinhos que recebem a mensagem).

Para um dado nó p, a sua *vizinhança reduzida* é formada pelo conjunto de nós Q de forma que para todo nó $q \in Q$, a aresta (p, q) pertence ao caminho de custo mínimo entre os nós p e q.

4.3 DESCRIÇÃO DETALHADA DO ALGORITMO

O algoritmo de Controle de Topologia localizado e distribuído (executado por cada processo) é apresentado na Figura 4.1. Esse algoritmo utiliza as seguintes variáveis:

Figura 4.1. Algoritmo *FindNeighbours* executado localmente por cada processo p para encontrar o seu conjunto reduzido de vizinhos.

```
Procedure FindNeighbours
01   RNbrs_p ← ∅   Pos_p ← ⟨p, ⟨x_p, y_p⟩⟩
02   broadcast ⟨Pos_p⟩                          {Phase 1}
03   wait until timeout(Δ)                      {Phase 2}
04   broadcast ⟨RNbrs_p, Pos_p⟩
05   wait until timeout(Δ)                      {Phase 3}
06   G_p^c ← buildConnectivityGraph(RNbrs_p)
07   G_p^min ← findMinimumCostPaths (p, G_p^c, Pos_p)
08   Nbrs_p ← {q : q ∈ V and (p, q) ∈ E_p^min}
09   return Nbrs_p

10   on receive ⟨Pos_q⟩ from q do
11      Pos_p ← Pos_p ∪ Pos_q
12      RNbrs_p ← RNbrs_p ∪ (p, q)

13   on receive ⟨RNbrs_q, Pos_q⟩ from q do
14      Pos_p ← Pos_p ∪ Pos_q
15      RNbrs_p ← RNbrs_p ∪ RNbrs_q
```

i) $RNbrs_p$: o conjunto de arestas (q, r) conhecidas por p, onde r representa todos os nós alcançados por q quando esse transmite usando sua potência máxima;

ii) Pos_p: conjunto de tuplas $\langle q, \langle x_q, y_q \rangle\rangle$, onde x_q e y_q representam as coordenadas Euclidianas x, y do processo q. Essa variável armazena a localização dos processos que p conhece;

iii) Δ: um período de tempo pré-definido. Assume-se que esse tempo seja suficiente para que o processo p receba todas as mensagens de seus vizinhos nas fases 1 e 2. (linhas 2 e 4 da Figura 4.1);

iv) $G_p^c(V_p^c, E_p^c)$: o gráfico de conectividade que o processo p consegue construir com as informações obtidas nas fases 1 e 2. V_p^c é o conjunto de nós conhecidos por p quando p e seus vizinhos imediatos transmitem usando a potência máxima. Esse conjunto inclui, além do processo p, os seus vizinhos e os vizinhos de seus vizinhos.

$$V_p^c = \{p\} \cup \{q : q \in V \wedge (((p,q) \in E_{max}) \vee (\exists r : r \in V \wedge (p,r) \in E_{max} \wedge (r,q) \in E_{max}))\}$$

e

$$E_p^c = \{(p,q) : q \in V_p^c \wedge (p,q) \in E_{max}\} \cup \{(r,s) : r \in V_p^c \wedge s \in V_p^c \wedge (p,r) \in E_{max} \wedge (r,s) \in E_{max}\}$$

v) $G_p^{min}(V_p^{min}, E_p^{min})$: representa o grafo que contém os caminhos de custo mínimo entre p e todos os seus vizinhos diretos. G_p^{min} é um subgrafo de G_p^c em que:

$$V_p^{min} = \{p\} \cup \{q : q \in V \wedge (p,q) \in E_{max}\}$$

e

$$E_p^{min} = \{(q,r) : q \in V_p^{min} \wedge r \in V_p^{min} \wedge (q,r) \in E_p^c \wedge (\exists s : s \in V_p^{min} \wedge (q,r) \text{ está em um caminho de menor custo entre os processos } p \text{ e } s)\}$$

vi) $Nbrs_p$: é o conjunto de nós que representa a vizinhança reduzida do nó p.

O algoritmo apresentado na Figura 4.1 tem o objetivo de calcular o conjunto reduzido de vizinhos de um nó.

Considere que o algoritmo está sendo executado pelo processo p. Quando o algoritmo inicia, o conjunto $RNbrs_p$ está vazio enquanto o Pos_p contém a posição (coordenadas x e y) de p (linha 1, Figura 4.1).

Na primeira fase do algoritmo, o processo p envia uma mensagem *broadcast* na rede contendo o seu identificador e suas coordenadas x e y (linha 2, Figura 4.1).

Na segunda fase (linhas 3 e 4, Figura 4.1) o processo p espera até que todos os seus vizinhos enviem os seus respectivos identificadores e suas localizações (linha 3, Figura 4.1). Assume-se que o período Δ é suficiente para o processo receber essa mensagem de todos os seus vizinhos. Ao receber uma mensagem $\langle q, (x_q, y_q)\rangle$ do nó q, o processo p inclui (p, q) no conjunto $RNbrs_p$ e as coordenadas do processo q em Pos_p (linhas 10 a 12, Figura 4.1). A aresta (p, q) indica que o processo q é um vizinho direto de p.

Após receber as mensagens com as localizações de todos os seus vizinhos diretos, o processo p envia uma mensagem *broadcast* com as informações de seus vizinhos diretos - conjuntos $RNbrs_p$ Pos_p (linha 4, Figura 4.1).

Na terceira fase (linhas 5 a 9, Figura 4.1), o processo p espera um período Δ até que todos os processos vizinhos enviem as informações de sua vizinhança (identificador e posições dos processos vizinhos) (linhas 5, Figura 4.1). Como expresso anteriormente, assume-se que o período Δ seja suficiente para que todos os vizinhos de p enviem uma mensagem com essas informações.

Ao receber uma mensagem $\langle RNbrs_q, Pos_q\rangle$, o processo p atualiza os conjuntos Pos_p e $RNbrs_p$ (linhas 13 a 15, Figura 4.1). O processo p inclui as informações sobre a vizinhança dos seus processos vizinhos nos conjuntos que representam a sua visão local da rede. Com esse procedimento, p pode ampliar a sua visão local da rede (caso o processo q conheça algum processo que não seja alcançado por p).

O procedimento *buildConnectivityGraph* utiliza o conjunto $RNbrs_p$ para construir uma representação do grafo de conectividade local do processo p, G_p^c - um subgrafo de G_{max} (linha 6, Figura 4.1).

A partir do grafo G_p^c e do conjunto Pos_p, o procedimento *findMinimumCostPath* calcula os caminhos de custo mínimo entre o processo p e todos os seus vizinhos (linha 7, Figura 4.1). O cálculo do caminho mínimo considera o custo com *overhearing*. Cada aresta (p, q) tem um custo, $C(p, q)$, associado - conforme a Equação 4.1.

G_p^{min} é o resultado do procedimento *findMinimumCostPath*, ou seja, é o grafo formado por todos os caminhos de custo mínimo entre p e seus vizinhos. O conjunto $Nbrs_p$

representa todos os processos (nós) que estão diretamente conectados a p em G_p^{min} (linha 8, Figura 4.1). Esse conjunto é o resultado do algoritmo (linha 9, Figura 4.1).

Considere que $G_p^R = (V_p^R, E_p^R)$ representa o grafo de conectividade entre o processo p e o seu conjunto reduzido de vizinhos. Ou seja, $V_p^R = \{p\} \cup Nbrs_p$ e $E_p^R = \{(p,q) : q \in Nbrs_p\}$.

$G_{min} = (V, E_{min})$ é o grafo de conectividade gerado a partir do grafo G_p^R de todos os processos $p \in V$. E_{min} é definido como:

$$E_{min} = \bigcup_{\forall p \in V} E_p^R \tag{4.2}$$

Assume-se, adicionalmente, o seguinte requisito na construção de G_{min}: se uma aresta (p,q) é um caminho de mínimo custo entre os processos p e q, esta aresta é escolhida como o caminho de mínimo custo mesmo que exista outro caminho r ($|r| > 1$) entre p e q de mesmo custo.

4.4 PROPRIEDADES DO ALGORITMO

Nessa seção serão apresentadas as provas das seguintes propriedades: (a) *propriedade da conectividade:* que determina que se G_{max} é um grafo fortemente conexo, então G_{min} é também fortemente conexo; e (b) *propriedade do caminho de custo mínimo*: que determina que as rotas de caminho mínimo em G_{max} continuam existindo em G_{min}. Essas propriedades são importantes para expressar a corretude do algoritmo proposto nesse trabalho. A primeira propriedade garante que o subgrafo gerado pelo algoritmo, G_{min}, mantém a conectividade do grafo, mantendo, portanto, caminhos entre qualquer par de vértices. Já a segunda propriedade garante que as rotas de menor custo em G_{max} foram mantidas em G_{min}.

Considere que cada aresta $(p,q) \in E_{max}$ tem um custo associado, chamado $Cost(p,q)$

(conforme definido na Equação 4.1).

Teorema 1: *Se $G_{max} = (V, E_{max})$ é fortemente conexo, então $G_{min} = (V, E_{min})$ também é fortemente conexo.*

Prova: Dado que G_{max} é fortemente conexo, existe um caminho entre todo par de vértices desse grafo. Considere dois vértices, p e q. G_{max} contém pelo menos um caminho de custo mínimo entre os vértices p e q. Seja um caminho de custo mínimo representado por $\langle r_1, r_2, ..., r_m \rangle$, onde $m \geq 2$, $r_1 = p$, $r_m = q$.

Considere o grafo de conectividade reduzido do processo r_i, $G^R_{r_i} = (V^R_{r_i}, E^R_{r_i})$ (apresentado na seção anterior). Primeiro será provado que $(r_i, r_{i+1}) \in E^R_{r_i}$, $\forall i : 1 \leq i < m$. Observe que todas as arestas em $E^c_{r_i}$ também pertencem a E_{max} (ou seja, $(e \in E^c_{r_i}) \Rightarrow (e \in E_{max})$) e o custo $Cost(p, q)$ que cada aresta terá será o mesmo que tem no grafo original, uma vez que o custo da aresta depende da distância entre os nós e do número de vizinhos no raio de alcance do nó transmissor. O número de vizinhos considerado localmente será o mesmo que o considerado no grafo original, uma vez que cada nó conhece seus vizinhos em dois saltos. Considerando que (r_i, r_{i+1}) é o caminho de custo mínimo em G_{max}, essa aresta é o caminho de menor custo entre r_i e r_{i+1}. Como $G^R_{r_i}$ é gerado por um algoritmo que calcula os caminhos de custo mínimo, (r_i, r_{i+1}) pertence a $E^R_{r_i}$.

Como todas as arestas pertencentes ao caminho de custo mínimo entre os nós p e q estão em E^R_s, para qualquer nó s, deve existir um caminho entre p e q em G_{min} (considerando que E_{min} é o conjunto resultante da união de todos os conjuntos E^R_s - veja a seção anterior). Como essa propriedade é válida para todo par de nós p e q, G_{min} é fortemente conexo.

Teorema 2: *Considere c como sendo o custo do caminho de custo mínimo entre o par de nós p e q em G_{max}. Existe um caminho em G_{min} entre p e q com custo c.*

Prova: Usando os mesmos argumentos da prova do Teorema 1, todas as arestas pertencentes ao caminho de custo mínimo entre os nós p e q em G_{max} estarão em E_{min}. Assim, o caminho de custo mínimo em G_{max} também existirá em G_{min}. Como o custo das arestas

em G_{max} e G_r^{min} é o mesmo, para todo nó r, então o custo do menor caminho em G_{min} é o mesmo, c.

4.5 SIMULAÇÕES

A simulação através de computador é uma das principais ferramentas usadas na pesquisa e desenvolvimento de novas idéias e algoritmos para rede de comunicação de dados. É uma técnica de custo baixo que permite construir ambientes virtuais que simulam o comportamento real de uma rede. Em se tratando de RSSF, o desafio de construir cenários reais torna-se ainda maior, dadas algumas características específicas desse tipo de rede (as RSSF podem ser formadas por centenas/milhares de nós distribuídos aleatoriamente em uma grande área geográfica - conforme discutido na seção 2.3). Para esses cenários, torna-se mais complexa a construção de um ambiente físico que represente diversos aspectos de uma RSSF. Assim, um simulador de redes pode ser usado para criar uma rede e controlar o seu funcionamento.

O comportamento do algoritmo proposto nesse trabalho foi avaliado com o uso do simulador de redes *Network Simulator 2 - ns-2* [73]. O *ns-2* é um dos mais populares simuladores para rede de comunicação de dados e oferece suporte a simulações de redes baseadas no protocolo TCP/IP. Dentre elas, suporte para simulação de redes com fio e sem fio (satélite, WiMAX, 802.11, etc), protocolos de roteamento (*unicast*, *multicast* ou *broadcast*), protocolos de transporte (TCP e UDP), protocolos de aplicação (FTP, TFTP, HTTP, DNS), geração aleatória ou programada de erros em nós, dentre outras.

4.5.1 Algoritmos e Dispositivos Avaliados

A escolha dos algoritmos usados nas simulações realizadas nesse trabalho foi feita com base em três critérios. O primeiro deles está relacionado ao efeito do *overhearing*: foram escolhidos algoritmos que consideram esse efeito e algoritmos que não o consideram. O segundo critério está relacionado ao tipo de aresta redundante que os algoritmos são capa-

zes de identificar: foram avaliados algoritmos que eliminam apenas arestas *2-redundantes* e algoritmos que eliminam arestas *k-redundantes* (para $k \geq 2$). O último critério está relacionado às informações que os nós conhecem da rede: foram utilizados algoritmos que utilizam informações locais da topologia da rede e algoritmos que conhecem toda a topologia rede.

Atendendo a esses critérios, foram escolhidos seis algoritmos de Controle de Topologia. São eles:

a) **L-2Red-NoOH**: algoritmo apresentado em [6], que utiliza informações locais da topologia da rede e elimina arestas *2-redundantes* sem considerar o custo com *overhearing*. Este algoritmo foi descrito na seção 3.5;

b) **L-2Red-OH**: uma variação do algoritmo proposto em [6], que utiliza informações locais da topologia da rede e elimina as arestas *2-redundantes* considerando o custo com *overhearing*;

c) **L-KRed-NoOH**: uma variação do algoritmo proposto nesse trabalho. Utiliza informações locais da topologia da rede e é capaz de eliminar as arestas *k-redundantes* mas sem considerar gasto de energia com *overhearing*;

d) **L-KRed-OH**: algoritmo que utiliza informações locais e elimina as arestas *k-redundantes* considerando o gasto de energia com *overhearing*. É o algoritmo proposto nesse trabalho;

e) **G-KRed-NoOH**: algoritmo que utiliza as informações globais da rede e elimina as arestas *k-redundantes* sem considerar o custo com *overhearing*;

e) **G-KRed-OH**: uma variação do algoritmo anterior que conhece toda a topologia da rede, elimina as arestas *k-redundantes* e considera o custo com *overhearing*.

Os algoritmos *L-2Red-NoOH* e *L-2Red-OH* utilizam uma estratégia distinta da apresentada em [6] (e reproduzida na Figura 3.6, na página 30) para ajustar a potência de

cada nó. Ao invés de determinar uma área denominada *relay region*, foi usada a estratégia apresentada na seção 3.1.4 onde a potência do nó é determinada a partir do seu conjunto reduzido de vizinhos (a potência mínima necessária para alcançar todos os seus vizinhos). Para implementar os algoritmos que não consideravam o efeito *overhearing* (*L-2Red-NoOH*, *L-KRed-NoOH* e *G-KRed-NoOH*), o procedimento que calcula o número de processos que recebem uma mensagem num determinado raio de alcance considera apenas o nó receptor.

Os *scripts* de simulação dos algoritmos *L-2Red-NoOH*, *L-2Red-OH*, *L-KRed-NoOH* e *L-KRed-OH* para o *ns-2* estão apresentados nos Apêndices A, B, C e D.

Os dois últimos algoritmos, *G-KRed-NoOH* e *G-KRed-OH*, são implementações do algoritmo de Floyd-Warshall [54] para encontrar o caminho de custo mínimo entre todos os nós da rede. Ao contrário dos algoritmos anteriores - localizados - onde os nós têm uma visão parcial da topologia da rede, o processo que executa os algoritmos *G-KRed-NoOH* e *G-KRed-OH* conhece a localização física de todos os processos da rede. O algoritmo *G-KRed-NoOH* não considera o custo com *overhearing*. Já o algoritmo *G-KRed-NoOH*, por sua vez, considera esse custo. Os *scripts* de simulação dos algoritmos *G-KRed-NoOH* e *G-KRed-NoOH* para o *ns-2* estão descritos, respectivamente, nos Apêndices E e F.

A topologia obtida a partir desses dois últimos algoritmos (que utilizam o conhecimento global da rede) resulta em um grafo que contém somente caminhos mínimos [54].

Os parâmetros de configuração usados nas simulações estão apresentados na Tabela 4.1.

O parâmetro *Channel Type* define que o canal de comunicação será sem fio. A potência máxima de transmissão é obtida através do acesso à interface definida pelo parâmetro *Network Interface Type*. O modelo de perda de sinal considerado, conforme justificado na seção 3.3, é o *FSPL* (*free space path loss*) [55]. O protocolo *TDMA* foi escolhido (parâmetro *MAC Type*) por eliminar a colisão de mensagens na camada de enlace. O parâmetro *Interface Queue Type* define o algoritmo *Droptail* para controle do *buffer* de dados da camada *MAC*. O *Droptail* armazena os pacotes na ordem em que eles chegam,

Tabela 4.1. Parâmetros de configuração do ns-2.

Channel Type	WirelessChannel
Network Interface Type	WirelessPhy
Radio Propagation Model	FreeSpaceModel
MAC Type	TDMA
Interface Queue Type	DropTail / PriQueue
Link Layer Type	LL
Max Packet in IFQ	50
Adhoc Routing Protocol	DumbAgent
Antenna Model	OmniAntenna
Energy Model	EnergyModel
Path Loss Exponent	2

e, assim que a rede permitir, envia-os nesta mesma ordem. Nenhuma decisão é feita sobre a prioridade dos pacotes, a ordem de chegada determina a ordem de envio. No caso da fila atingir seu limite, o algoritmo descarta os pacotes que chegarem a partir de então (*DropTail*). O tamanho do *buffer* é definido pelo parâmetro *Max Packet in IFQ*. O protocolo *DumbAgent* é usado como protocolo de roteamento entre nós (parâmetro *AdHoc Routing Protocol*. O *DumbAgent* pode ser usado em um cenário onde os nós de origem e destino conseguem se comunicar diretamente (são vizinhos diretos). Nesse caso, nenhum roteamento é necessário. Foi utilizada a antena do tipo *Omni Antenna* (parâmetro *Antenna Model*). Esse modelo de antena permite que uma mensagem seja enviada em todas as direções. Assim, quando um nó transmissor p envia uma mensagem, todos os nós alcançáveis num raio circular com centro em p recebem a mensagem. Os parâmetros *Energy Model* e *Path Loss Exponent* são usados para configurar o modelo de energia usado nas simulações. O primeiro parâmetro, *EnergyModel*, indica que o simulador deve considerar o gasto de energia na rede. Sem esse atributo, o simulador não registra o consumo de energia dos nós. O parâmetro *Path Loss Exponent* define o coeficiente de perda do canal. Esse parâmetro é usado para calcular o gasto de transmissão em função da distância (conforme descrito nas Equações 3.2 e 3.3).

4.5.2 Transceptores usados como base para as simulações

As simulações realizadas nesse trabalho foram baseadas no consumo dos transceptores apresentados na seção 2.4. Os valores retirados das especificações dos transceptores *Atmel RF230* [8], *Chipcon CC2420* [10] e *Chipcon CC1000* [11] e sumarizados nas tabelas 2.2, 2.4 e 2.5 foram usados como base para configurar os valores das variáveis $E_{Tx-elec}$, $E_{Rx-elec}$ e ϵ (usadas nas equações 3.3 e 3.7 para calcular o custo de transmissão atribuído a uma determinada aresta). A partir daqui, os termos *transRF230*, *transCC2420* e *transCC1000* serão usados para designar os valores de consumo baseados, respectivamente, nos transceptores *Atmel RF230*, *Chipcon CC2420* e *Chipcon CC1000*. Os parâmetros de configuração baseados nesses dispositivos são apresentados no Apêndice G.

4.5.3 Cenários e Resultados

Foram realizadas simulações com cinco diferentes topologias de rede: 45, 50, 55, 60 e 64 nós distribuídos aleatoriamente numa área bidimensional de 500x500 metros. A posição dos nós em cada cenário foi definida aleatoriamente.

Foram geradas aleatoriamente dez topologias de rede para cada um desses cenários. Para cada topologia, foram realizadas simulações com os seis algoritmos apresentados na seção 4.5.1 e com os três modelos de transceptores apresentados na seção 4.5.2.

Serão apresentados e discutidos a seguir os resultados obtidos nos cenários com menor densidade de nós (45 nós distribuídos em uma área de 500x500 metros) e com maior densidade de nós (64 nós distribuídos em uma área de 500x500 metros).

45 nós distribuídos em uma área de 500x500 metros. Os dados apresentados a seguir representam os resultados obtidos através das simulações com uma das dez topologias com 45 nós distribuídos em uma área de 500x500 metros.

O grafo de conexão máxima G_{max} para esse cenário tinha 300 arestas (Figura 4.2.a).

Os resultados obtidos pelos algoritmos que não consideraram o efeito *overhearing*

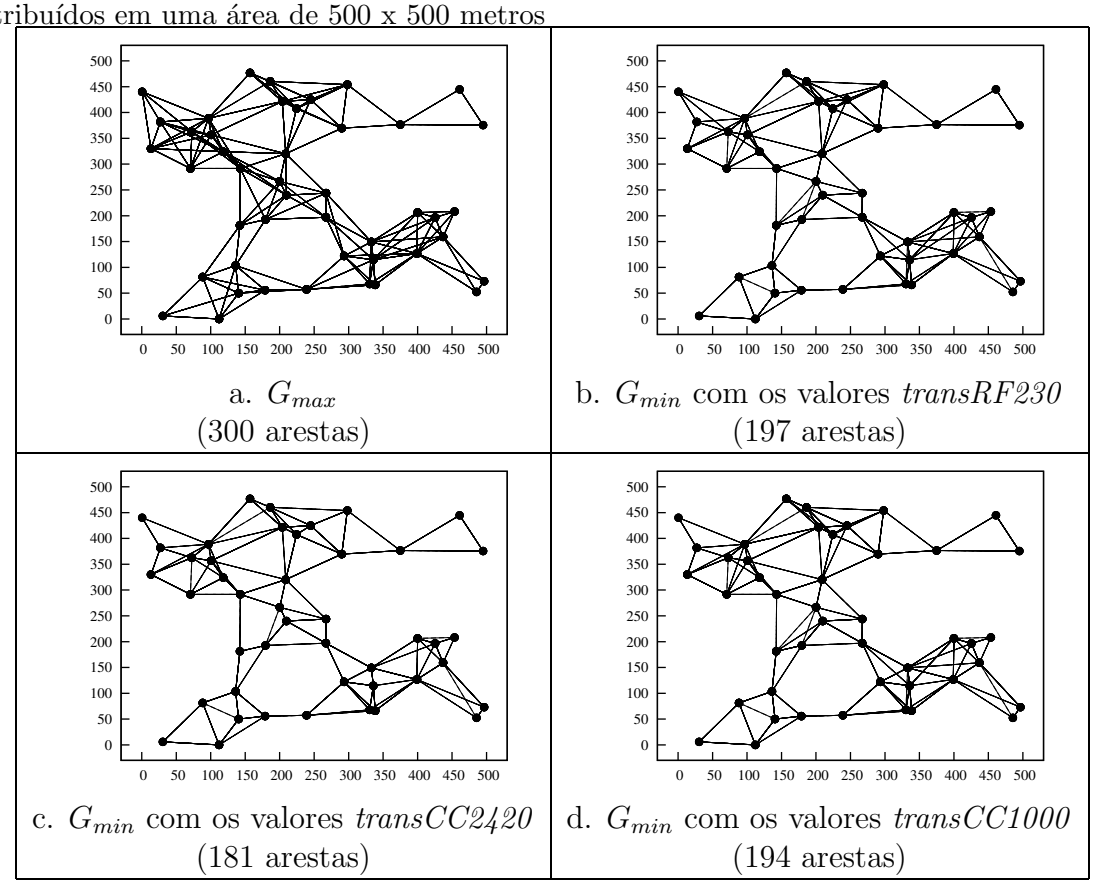
(*L-2Red-NoOH*, *L-KRed-NoOH* e *G-KRed-NoOH*) foram idênticos para todos os dispositivos considerados (*transRF230*, *transCC2420* e *transCC1000*). Os algoritmos não identificaram arestas redundantes e os grafos de conectividade permaneceram idênticos ao grafo de conexão máxima G_{max}. Não houve, consequentemente, redução da potência de transmissão de qualquer nó.

O algoritmo *L-2Red-OH*, por sua vez, conseguiu reduzir o número de arestas do grafo G_{max} para todos valores de consumo utilizados na simulação. Com os valores *transRF230*, o algoritmo reduziu o grafo G_{max} (Figura 4.2.a) para um subgrafo com 197 arestas, uma redução de 34.3% no número de arestas (Figura 4.2.b). Para os valores de *transCC2420*, o algoritmo reduziu o grafo G_{max} para um subgrafo com 181 arestas, uma redução de aproximadamente 40% do número de arestas (Figura 4.2.c). Para os valores de *transCC1000*, a redução foi de 35.3% e o subgrafo resultante tinha 194 arestas (Figura 4.2.d).

O algoritmo que elimina as arestas denominadas *k-redundantes*, *L-KRed-OH*, conseguiu reduzir um número maior de arestas para todos os dispositivos (quando comparado ao algoritmo *L-2Red-OH*). Para os valores de *transRF230*, o algoritmo reduziu o grafo de conectividade máxima G_{max} (Figura 4.3.a) em um subgrafo com 195 arestas (Figura 4.3.b). Esse número representa uma reduçao de 35.7% no número de arestas do grafo G_{max} (aproximadamente 1% a mais que o algoritmo *L-2Red-OH*). Para os valores de *transCC2420*, o subgrafo resultante teve um total de 179 arestas (Figura 4.3.c), uma redução de 40.3% no número de arestas quando comparado ao grafo G_{max} (aproximadamente 1.1% a mais que o algoritmo *L-2Red-OH*). Para os valores de *transCC1000*, o subgrafo resultante teve um total de 193 arestas. (Figura 4.3.d). Esse número representa uma redução de 40.3% no número de arestas (aproximadamente 0.5% a mais que o algoritmo *L-2Red-OH*).

O algoritmo que utiliza as informações globais da topologia da rede para reduzir as arestas denominadas *k-redundantes*, *G-KRed-OH*, gerou os mesmos resultados do algoritmo *L-KRed-OH* (Figura 4.3). Ou seja, para os valores *transRF230*, o subgrafo resultante tinha 195 arestas, para os valores *transCC2420* o subgrafo tinha 179 arestas e para

Figura 4.2. Grafo original G_{max} e subgrafos resultantes da simulação do algoritmo *L-2Red-OH* com os valores *transRF230*, *transCC2420* e *transCC1000* em um dos cenários com 45 nós distribuídos em uma área de 500 x 500 metros



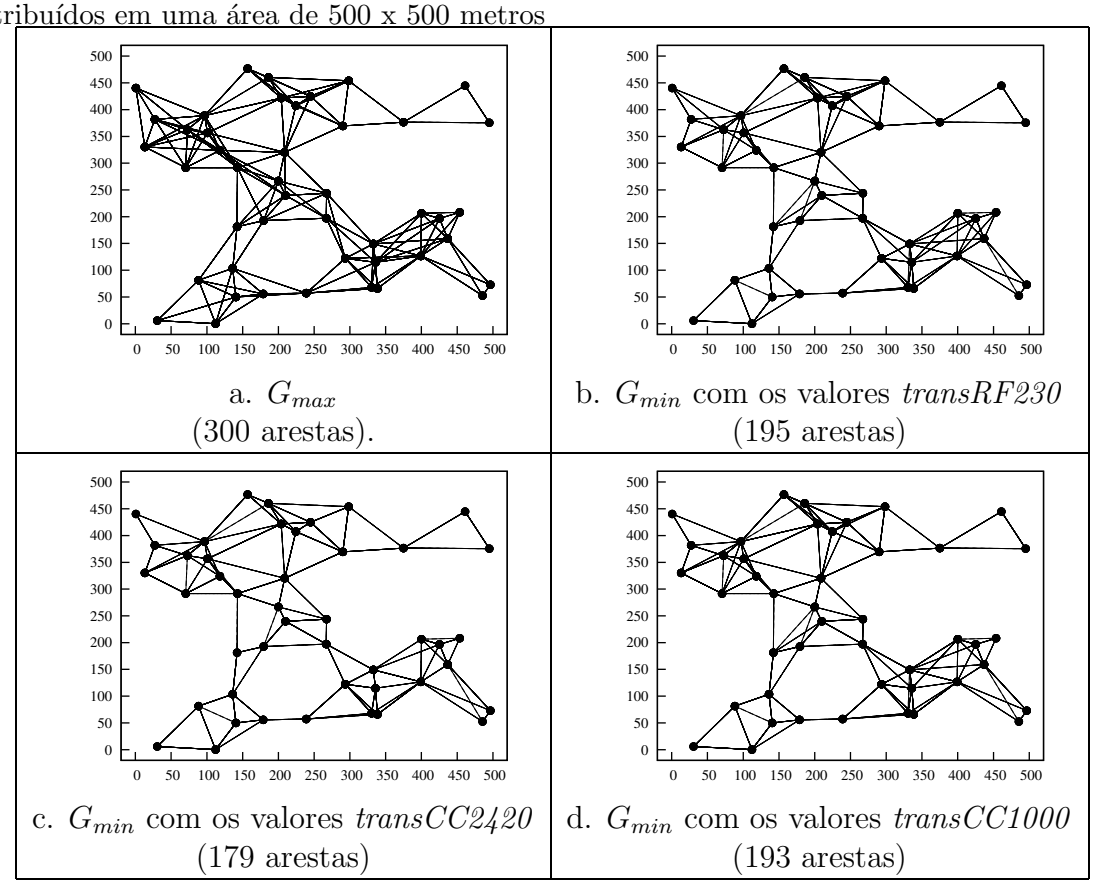
a. G_{max}
(300 arestas)

b. G_{min} com os valores *transRF230*
(197 arestas)

c. G_{min} com os valores *transCC2420*
(181 arestas)

d. G_{min} com os valores *transCC1000*
(194 arestas)

Figura 4.3. Grafo original G_{max} e subgrafos resultantes da simulação do algoritmo *L-KRed-OH* com os valores *transRF230*, *transCC2420* e *transCC1000* em um dos cenários com 45 nós distribuídos em uma área de 500 x 500 metros



a. G_{max} (300 arestas).

b. G_{min} com os valores *transRF230* (195 arestas)

c. G_{min} com os valores *transCC2420* (179 arestas)

d. G_{min} com os valores *transCC1000* (193 arestas)

os valores *transCC1000* o grafo resultante tinha um total de 193 arestas.

64 nós distribuídos em uma área de 500x500 metros. Serão apresentados a seguir os resultados obtidos com as simulações realizadas em uma das topologias com maior densidade: 64 nós distribuídos aleatoriamente em uma área de 500x500 metros.

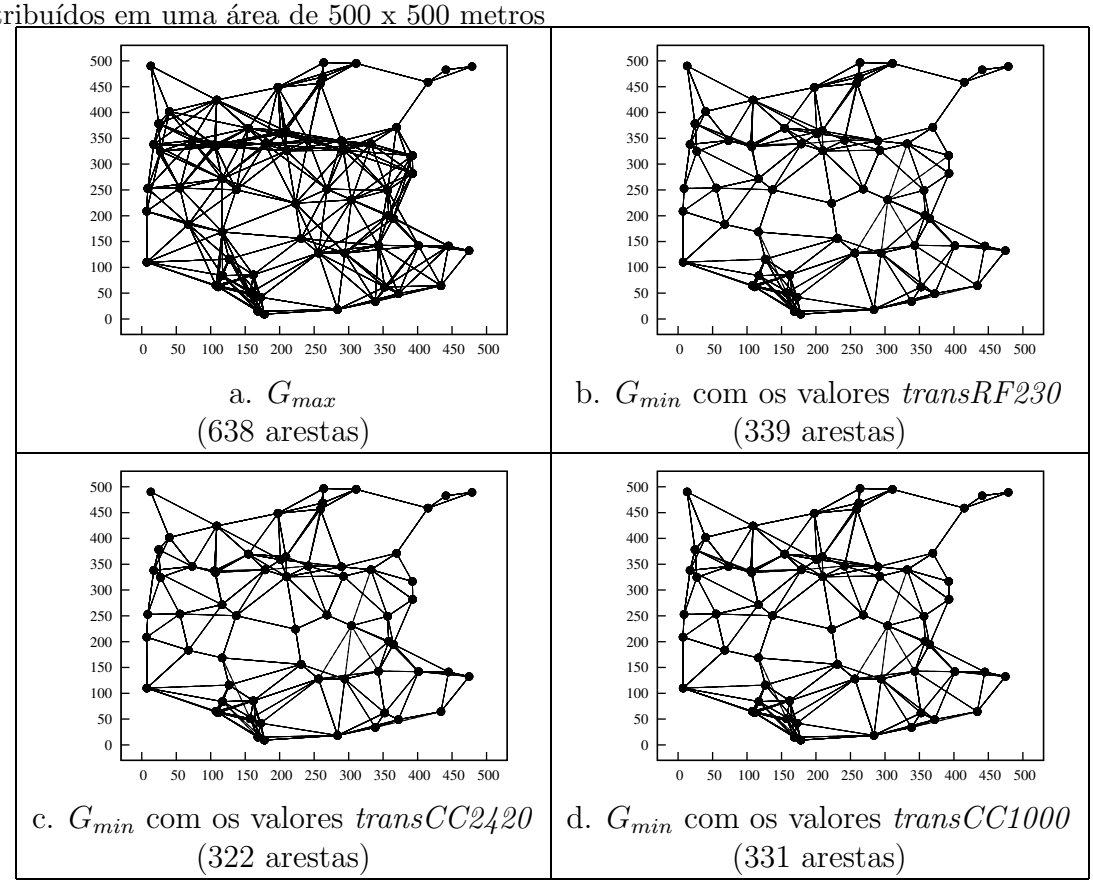
O grafo de conexão máxima G_{max} nesse cenário tem um total de 638 arestas (Figura 4.4.a).

Assim como no cenário anterior, os resultados obtidos pelos algoritmos que não consideraram o efeito *overhearing* (*L-2Red-NoOH*, *L-KRed-NoOH* e *G-KRed-NoOH*) foram idênticos para todos os valores de consumo considerados (*transRF230*, *transCC2420* e *transCC1000*). Em nenhum caso houve redução de arestas e os grafos de conectividade resultantes permaneceram idênticos ao grafo de conexão máxima G_{max} (com 638 arestas).

O algoritmo *L-2Red-OH*, por sua vez, conseguiu reduzir o número de arestas de G_{max} para todos os valores de consumo utilizados. Para os valores de *transRF230*, o algoritmo reduziu o grafo G_{max} (Figura 4.4.a) em um subgrafo com 339 arestas, uma redução de 46.9% no número de arestas (Figura 4.4.b). Para os valores de *transCC2420*, o algoritmo reduziu o grafo G_{max} em um subgrafo com 322 arestas, uma redução de aproximadamente 49.5% do número de arestas (Figura 4.4.c). Para os valores de *transCC1000* a redução foi de 48.1% e o subgrafo resultante teve 331 arestas (Figura 4.4.d).

Os resultados relacionados ao algoritmo *L-KRed-OH* seguiram a tendência do cenário anterior e o algoritmo conseguiu reduzir um número maior de arestas para todos os valores de consumo utilizados (quando comparado ao algoritmo *L-2Red-OH*). Para os valores de *transRF230*, o algoritmo reduziu o grafo G_{max} (Figura 4.5.a) em um subgrafo com 316 arestas (Figura 4.5.b), uma reduçao de 50.5% no número de arestas do grafo G_{max} (aproximadamente 6.8% a mais que o algoritmo *L-2Red-OH*). Para os valores de *transCC2420*, o subgrafo resultante teve um total de 308 arestas (Figura 4.5.c), uma redução de 51.7% no número de arestas do grafo G_{max} (aproximadamente 4.4% a mais que o algoritmo *L-2Red-OH*). Para os valores de *transCC1000*, o subgrafo resultante teve um total de 313 arestas (Figura 4.5.d). Esse número representa uma redução de 50.9%

Figura 4.4. Grafo original G_{max} e subgrafos resultantes da simulação do algoritmo *L-2Red-OH* com os valores *transRF230*, *transCC2420* e *transCC1000* em um dos cenários com 64 nós distribuídos em uma área de 500 x 500 metros



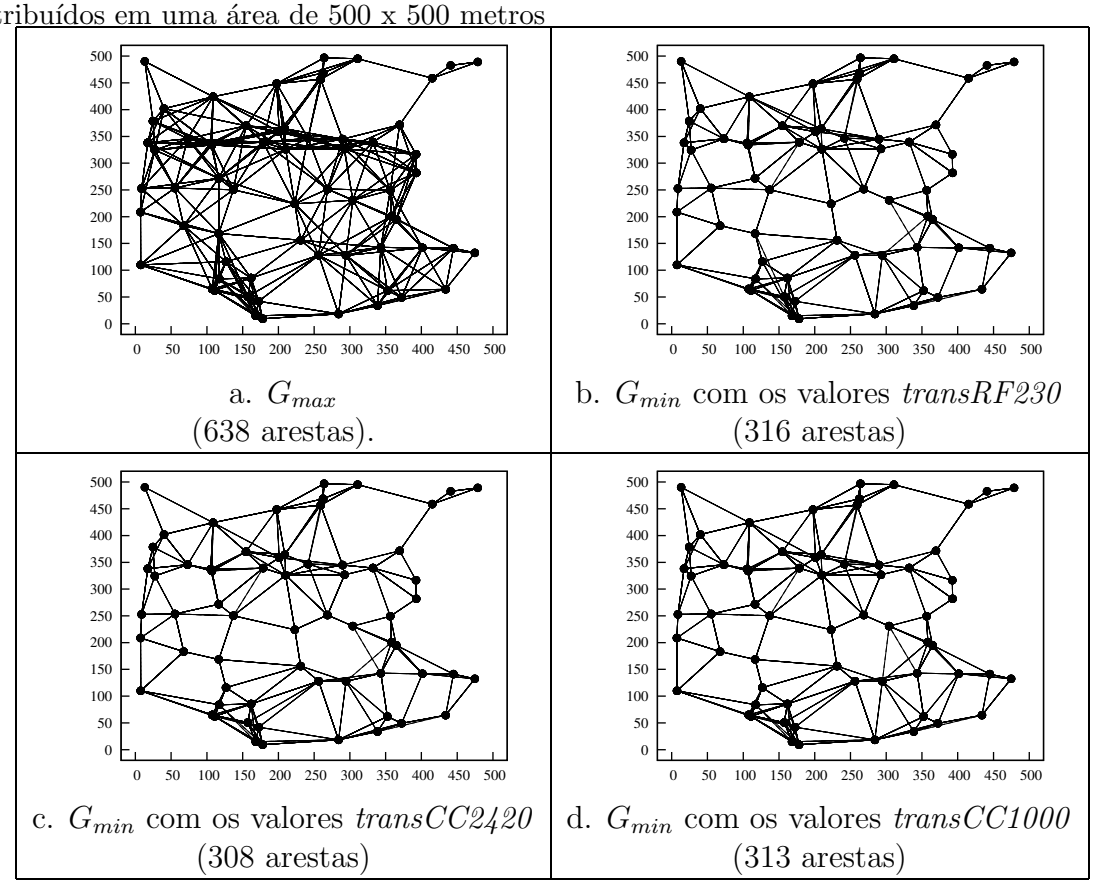
a. G_{max} (638 arestas)

b. G_{min} com os valores *transRF230* (339 arestas)

c. G_{min} com os valores *transCC2420* (322 arestas)

d. G_{min} com os valores *transCC1000* (331 arestas)

Figura 4.5. Grafo original G_{max} e subgrafos resultantes da simulação do algoritmo *L-KRed-OH* com os valores *transRF230*, *transCC2420* e *transCC1000* em um dos cenários com 64 nós distribuídos em uma área de 500 x 500 metros



a. G_{max} (638 arestas).

b. G_{min} com os valores *transRF230* (316 arestas)

c. G_{min} com os valores *transCC2420* (308 arestas)

d. G_{min} com os valores *transCC1000* (313 arestas)

no número de arestas quando comparado ao grafo G_{max} (aproximadamente 5.4% a mais que o algoritmo *L-2Red-OH*).

Assim como no cenário anterior, o algoritmo *G-KRed-OH* obteve os mesmos resultados do algoritmo *L-KRed-OH*. Ou seja, para os valores *transRF230*, o subgrafo resultante tinha 316 arestas, para os valores *transCC2420* o subgrafo resultante tinha 308 arestas e para os valores *transChipconCC100* o grafo resultante tinha um total de 313 arestas.

Sumário dos resultados obtidos. A seguir serão apresentados os resultados obtidos em todos os cenários. Os sumários dos resultados para os cenários com 45, 50, 55, 60 e 64 nós estão representados, respectivamente, nas Tabelas 4.2, 4.3, 4.4, 4.5 e 4.6.

Tabela 4.2. Média aritmética dos resultados obtidos para os 10 cenário com 45 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	316.4	0	316.4	0	316.4	0
L-2Red-OH	206.3	34.80	195.8	38.12	202.2	36.09
L-KRed-NoOH	316.4	0	316.4	0	316.4	0
L-KRed-OH	201.5	36.31	192.4	39.19	199.3	37.01
G-KRed-NoOH	316.4	0	316.4	0	316.4	0
G-KRed-OH	201.1	36.44	192.2	39.25	199.2	37.04

Tabela 4.3. Média aritmética dos resultados obtidos para os 10 cenário com 50 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	387.4	0	387.4	0	387.4	0
L-2Red-OH	244.5	36.89	234.8	39.39	239.3	38.23
L-KRed-NoOH	387.4	0	387.4	0	387.4	0
L-KRed-OH	239.8	38.10	231	40.37	236.6	38.63
G-KRed-NoOH	387.4	0	387.4	0	387.4	0
G-KRed-OH	239.8	38.10	231	40.37	236.6	38.63

Tabela 4.4. Média aritmética dos resultados obtidos para os 10 cenário com 55 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	468.6	0	468.6	0	468.6	0
L-2Red-OH	277.8	40.72	160.2	44.47	270.1	42.36
L-KRed-NoOH	468.6	0	468.6	0	468.6	0
L-KRed-OH	268.5	42.70	254.4	45.71	264.2	43.62
G-KRed-NoOH	468.6	0	468.6	0	468.6	0
G-KRed-OH	268.4	42.72	254.3	45.73	264.2	43.62

Tabela 4.5. Média aritmética dos resultados obtidos para os 10 cenário com 60 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	572.6	0	572.6	0	572.6	0
L-2Red-OH	314.3	45.11	301.5	47.35	307.7	46.26
L-KRed-NoOH	572.6	0	572.6	0	572.6	0
L-KRed-OH	305.5	46.65	294.9	48.5	302.4	47.19
G-KRed-NoOH	572.6	0	572.6	0	572.6	0
G-KRed-OH	305.4	46.66	204.9	48.5	302.3	47.21

Tabela 4.6. Média aritmética dos resultados obtidos para os 10 cenário com 64 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)	Número de Arestas	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	640.2	0	640.2	0	640.2	0
L-2Red-OH	338.8	47.08	319.1	50.16	329.3	48.56
L-KRed-NoOH	640.2	0	640.2	0	640.2	0
L-KRed-OH	323.3	49.5	308.3	51.84	318.2	50.30
G-KRed-NoOH	640.2	0	640.2	0	640.2	0
G-KRed-OH	322.9	49.56	307.9	51.91	318.1	50.31

4.5.4 Discussão

Serão discutidos a seguir os principais aspectos avaliados nesse trabalho: o impacto do efeito *overhearing* no Controle de Topologia, a comparação entre os algoritmos globais e locais, os reais benefícios dos algoritmos que eliminam as arestas *k-redundantes* em comparação com os algoritmos capazes de eliminar apenas as arestas ditas *2-redundantes*, as diferenças no comportamento dos algoritmos locais e dos algoritmos globais e a relação entre a densidade da rede e o Controle de Topologia.

4.5.4.1 Redução de aresta e ajuste da potência de transmissão

Analisar a redução do número de arestas de um grafo de conectividade de rede é uma forma eficiente de avaliar os benefícios de um algoritmo de topologia. Para certos algoritmos, como aqueles de cálculo de caminho de custo mínimo, por exemplo, a redução do número de arestas é algo bastante útil. No entanto, avaliar a quantidade de arestas eliminadas não é suficiente para assegurar a eficiência de um algoritmo de Controle de Topologia. Conforme discutido na seção 3.1.4, a eliminação de arestas redundantes em alguns casos não resulta na redução da potência do nó sensor.

Assim, o gasto médio de transmissão dos nós da rede foi utilizada como um item adicional na avaliação da eficiência dos algoritmos de Controle de Topologia. Esse valor foi calculado da seguinte forma: considere o grafo de conectividade $G = (V, E)$. O consumo médio de energia dos nós em G, denominado $AvEnergy(G)$, é a média aritmética do gasto de energia com transmissão de cada nó $u \in V$ quando u envia para o seu vizinho mais distante v, para $(u, v) \in E$.

Considere os dados apresentados na Tabela 4.7, que traz o consumo médio de energia dos nós nos subgrafos gerados pelos algoritmos de Controle de Topologia avaliados para os cenários com 45 nós distribuídos em uma área de 500x500 metros. A gasto de energia médio $AvEnergy(G_{max})$, para os valores *transRF230*, é 1378.1 nJ/bit. Como o grafo resultante dos algoritmos *L-2Red-NoOH*, *L-KRed-NoOH* e *G-KRed-NoOH* é idêntico a G_{max}, esse valor permaneceu inalterado (ou seja, 1378.1 nJ/bit).

Para os outros algoritmos, o gasto médio de energia $AvEnergy(G_{min})$ ficou menor. O algoritmo *L-2Red-OH*, por exemplo, reduziu o gasto médio de energia para 1056.4 nJ/bit (para os valores *transRF230*), uma redução de cerca de 23.34% quando comparado a $AvEnergy(G_{max})$. Já os algoritmos *L-KRed-OH* e *G-KRed-OH* reduziram, respectivamente, o gasto médio de energia para 1022.6 e 1019.7 nJ/bit (para os valores *transRF230*), uma redução de cerca de 25.80% e 26.01% quando comparado a $AvEnergy(G_{max})$.

As Tabelas 4.8, 4.9, 4.10 e 4.11 trazem os gastos médios de energia dos nós nos subgrafos gerados pelos algoritmos de Controle de Topologia ($AvEnergy(G_{min})$) para os outros cenários (50, 55, 60 e 64 nós distribuídos em uma área de 500x500 metros).

4.5.4.2 Comparação dos Resultados entre os algoritmos 2-redundantes e K-redundantes

Será apresentada a seguir uma análise dos resultados obtidos pelos algoritmos capazes de eliminar as arestas *k-redundantes* (para $k \geq 2$) e pelos algoritmos capazes de eliminar apenas as arestas *2-redundantes*

Pode-se observar, a partir dos dados apresentados, que o algoritmo *L-KRed-OH* teve

Tabela 4.7. Gasto médio de energia dos nós em G_{min} para os cenários com 45 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	1378.1	0	1923.1	0	2725.7	0
L-2Red-OH	1056.4	23.34	1416.7	26.33	2091.8	23.26
L-KRed-NoOH	1378.1	0	1923.1	0	2725.7	0
L-KRed-OH	1022.6	25.80	1379	28.29	2059.4	24.45
G-KRed-NoOH	1378.1	0	1923.1	0	2725.7	0
G-KRed-OH	1019.7	26.01	1377	28.40	2058.5	24.48

Tabela 4.8. Gasto médio de energia dos nós em G_{min} para os cenários com 50 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	1511.6	0	2094.2	0	2939.2	0
L-2Red-OH	1119.7	25.93	1494.9	28.62	2177.4	25.92
L-KRed-NoOH	1511.6	0	2094.2	0	2939.2	0
L-KRed-OH	1084.7	28.24	1457.4	30.41	2136.7	27.30
G-KRed-NoOH	1511.6	0	2094.2	0	2939.2	0
G-KRed-OH	1084.7	28.24	1457.4	30.41	2136.7	27.30

Tabela 4.9. Gasto médio de energia dos nós em G_{min} para os cenários com 55 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	1655.9	0	2281.9	0	3177.8	0
L-2Red-OH	1197.5	27.68	1552.4	31.97	2277.2	28.34
L-KRed-NoOH	1655.9	0	2281.9	0	3177.8	0
L-KRed-OH	1135.6	31.42	1494	34.53	2202.9	30.68
G-KRed-NoOH	1655.9	0	2281.9	0	3177.8	0
G-KRed-OH	1132.9	31.58	1490.4	34.69	2202.9	30.68

Tabela 4.10. Gasto médio de energia dos nós em G_{min} para os cenários com 60 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	1846.3	0	2524.5	0	3477.6	0
L-2Red-OH	1257.6	31.89	1663.6	34.10	2364.8	32.00
L-KRed-NoOH	1846.3	0	2524.5	0	3477.6	0
L-KRed-OH	1192	35.44	1591.1	36.97	2294.9	34.01
G-KRed-NoOH	1846.3	0	2524.5	0	3477.6	0
G-KRed-OH	1192	35.44	1591.1	36.97	2294.9	34.01

Tabela 4.11. Gasto médio de energia dos nós em G_{min} para os cenários com 64 nós distribuídos em uma área de 500x500 metros

	transRF230		*transCC2420*		*transCC1000*	
Algoritmo Simulado	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)	$AvEnergy$ (G_{min}) (nJ/bit)	Redução em relação ao G_{max} (em %)
L-2Red-NoOH	1932	0	2636.4	0	3620.1	0
L-2Red-OH	1298	32.82	1684	36.13	2422.8	33.07
L-KRed-NoOH	1932	0	2636.4	0	3620.1	0
L-KRed-OH	1217.3	36.99	1595	39.50	2304.2	36.35
G-KRed-NoOH	1932	0	2636.4	0	3620.1	0
G-KRed-OH	1215.6	37.08	1592.4	39.60	2304.2	36.35

um desempenho superior ao algoritmo *L-2Red-OH*. Com relação à quantidade de arestas eliminadas, o algoritmo *L-KRed-OH* eliminou, em média, 1.36% arestas a mais do que o algoritmo *L-2Red-OH*. Com relação ao gasto médio de energia, $AvEnergy(G_{min})$, o algoritmo *L-KRed-OH* também obteve resultados ligeiramente superiores, conseguindo reduzir o valor de $AvEnergy(G_{min})$, em média, 2.60% a mais que o algoritmo *L-2Red-OH*.

Esse resultado é importante pois permite avaliar os benefícios de um algoritmo que elimina as arestas *k-redundante* em relação a um algoritmo que elimina apenas as arestas *2-redundantes*.

4.5.4.3 Comparação dos Resultados entre os algoritmos Global e Local

O objetivo dessa seção é identificar como os algoritmos que utilizaram apenas informações locais da rede se comportaram em comparação aos algoritmos que conheciam a topologia completa da rede.

Os algoritmos *L-KRed-OH* e *G-KRed-OH* obtiveram resultados bastante similares. Em quase todos os cenários, os algoritmos eliminaram a mesma quantidade de arestas.

Uma das exceções ocorreu num dos cenários com 60 nós distribuídos em uma área de 500x500 metros. Nesse cenário o grafo G_{max} tinha 566 arestas. O algoritmo *L-KRed-OH* conseguiu reduzir G_{max} para subgrafos com 333 (para os valores *transRF230*) e 327 arestas (para os valores *transCC1000*). Já o algoritmo que conhecia a topologia de toda a rede, *G-KRed-OH*, conseguiu reduzir G_{max} para subgrafos com 332 (para os valores *transRF230*) e 326 arestas (para os valores *transCC1000*). Nesses dois casos, o algoritmo que conhecia a topologia completa da rede, *G-KRed-OH*, conseguiu reduzir uma aresta a mais. Com relação ao valor do gasto médio de energia ($AvEnergy(G_{min})$), no entanto, os algoritmos obtiveram os mesmos resultados para esse cenário. O gasto médio de energia dos nós em G_{max} é 1825 nJ/bit para os valores *transRF230* e 3429 nJ/bit para os valores *transCC1000*. Os algoritmos *L-KRed-OH* e *G-KRedOH* reduziram o valor de $AvEnergy(G_{min})$ para 1336 nJ/bit para os valores *transRF230* e 2555 nJ/bit para os valores *transCC1000*.

Esse resultado é importante, porque mostra que o algoritmo que conhecia apenas as informações locais da rede, ou seja, um subconjunto das informações globais, comportou-se de forma bastante próxima ou similar ao algoritmo que conhecia a topologia completa da rede.

4.5.4.4 Efeito do overhearing

Outro importante resultado observado está relacionado ao impacto do custo de *overhearing* no Controle de Topologia. Enquanto os algoritmos que consideraram o efeito *overhearing* (*L-2Red-OH*, *L-KRed-OH* e *G-KRed-OH*) conseguiram otimizar a topologia da rede, os algoritmos que não consideraram esse custo (*L-2Red-NoOH*, *L-KRed-NoOH* e *G-KRed-NoOH*) não conseguiram identificar (e consequentemente) reduzir nenhuma aresta redundante.

Os algoritmos *L-2Red-OH*, *L-KRed-OH* e *G-KRed-OH* conseguiram reduzir, em média, 43.29% das arestas de G_{max}. Com relação ao gasto médio de energia, esses algoritmos

reduziram o valor de $AvEnergy(G_{max})$ em média 31.18%.

4.5.4.5 Arestas redundantes para os algoritmos L-2Red-NoOH, L-KRed-NoOH e G-KRed-NoOH

Será discutida a seguir a relação que existe entre as arestas redundantes nos cenários onde o custo do *overhearing* é desconsiderado (*L-2Red-NoOH*, *L-KRed-NoOH* e *G-KRed-NoOH*) e os valores de consumo dos dispositivos avaliados (Tabelas 2.2, 2.4 e 2.5).

Conforme apresentado na seção 3.1, uma aresta (a, c) é dita redundante se existir um caminho $\langle a, b, c\rangle$ tal que:

$$Cost(a, c) > Cost(a, b) + Cost(b, c) \tag{4.3}$$

Considerando o modelo de energia apresentado na seção 3.3, que define que o custo de um caminho é a soma dos custos de transmissão e dos custos de recebimento (de todos os nós que recebem a mensagem), pode-se reescrever a Equação 4.3 acima da seguinte forma:

$$E_{Tx(a,c)} + E_{Rx(a,c)} > E_{Tx(a,b)} + E_{Rx(a,b)} + E_{Tx(b,c)} + E_{Rx(b,c)} \tag{4.4}$$

Onde: $E_{Tx(a,c)}$ e $E_{Rx(a,c)}$ são, respectivamente, os custos com transmissão e recebimento associados à aresta (a, c); $E_{Tx(a,b)}$ e $E_{Rx(a,b)}$ são, respectivamente, os custos com transmissão e recebimento associados à aresta (a, b); e $E_{Tx(b,c)}$ e $E_{Rx(b,c)}$ são, respectivamente, os custos com transmissão e recebimento associados à aresta (b, c).

Pode-se reescrever a Equação 4.4 da seguinte forma:

$$E_{Tx(a,c)} - E_{Tx(a,b)} - E_{Tx(b,c)} > E_{Rx(a,b)} + E_{Rx(b,c)} - E_{Rx(a,c)} \tag{4.5}$$

Se o fator *overhearing* for desconsiderando, os valores $E_{Rx(a,c)}$, $E_{Rx(a,b)}$ e $E_{Rx(b,b)}$ são constantes e iguais. Considera-se, portanto, $E_{Rx} = E_{Rx(a,c)} = E_{Rx(a,b)} = E_{Rx(b,c)}$.

A equação 4.5 pode ser reduzida para:

$$E_{Tx(a,c)} - E_{Tx(a,b)} - E_{Tx(b,c)} > E_{Rx} \tag{4.6}$$

Reescrevendo a Equação 4.6 de acordo com a Equação 3.7:

$$E_{Tx_{elec}} + \epsilon.d(a,c)^{\alpha} - (E_{Tx_{elec}} + \epsilon.d(a,b)^{\alpha}) - (E_{Tx_{elec}} + \epsilon.d(b,c)^{\alpha}) > E_{Rx} \tag{4.7}$$

Como $E_{Tx_{elec}}$ é constante, a Equação 4.7 pode ser reduzida para:

$$\epsilon.d(a,c)^{\alpha} - \epsilon.d(a,b)^{\alpha} - \epsilon.d(b,c)^{\alpha} > E_{Rx} + E_{Tx_{elec}} \tag{4.8}$$

$$d(a,c)^{\alpha} - d(a,b)^{\alpha} - d(b,c)^{\alpha} > (E_{Rx} + E_{Tx_{elec}})/\epsilon \tag{4.9}$$

Considerando $\alpha = 2$ (o valor considerado nas simulações):

Tabela 4.12. Valores específicos de cada dispositivo: alcance máximo de transmissão e custos de transmissão e recebimento

Modelo do Transceptor	d_{max} (m)	E_{Rx} (nJ/bit)	$E_{Tx_{elec}}$ (nJ/bit)	ϵ $(nJ/bit/m^2)$
Atmel RF230	300	186	48.0	0.0017
Chipcon CC2420	100	236.4	48.0	0.016
Chipcon CC1000	152	292.1	157.9	0.038

$$d(a,c)^2 - d(a,b)^2 - d(b,c)^2 > (E_{Rx} + E_{Tx_{elec}})/\epsilon \tag{4.10}$$

A expressão $d(a,c)^2 - d(a,b)^2 - d(b,c)^2$ atinge seu valor máximo quando $d(a,c) = d_{max}$, $d(a,b) = d(a,c)/2$ e $d(b,c) = d(a,c)/2$, onde d_{max} representa o alcance máximo do transceptor.

Considere os valores de consumo de energia (com transmissão e recepção) e do alcance máximo dos dispositivos *Atmel RF230*, *Chipcon CC2420* e *Chipcon CC1000* sumarizados na Tabela 4.12.

Para o dispositivo $AtmelRF230$, o valor máximo da expressão $d(a,c)^2 - d(a,b)^2 - d(b,c)^2$ é 11552.00 enquanto o valor (constante) da expressão $(E_{Rx}+E_{Tx_{elec}})/\epsilon$ é 11602.28. Portanto, considerando a Equação 4.10, não existem arestas redundantes em G_{max} para esse dispositivo quando o custo com *overhearing* é desconsiderado.

Os valores apresentados na Tabela 4.13 representam os valores máximos da expressão $d(a,c)^2 - d(a,b)^2 - d(b,c)^2$ e os valores da expressão $(E_{Rx} + E_{Tx_{elec}})/\epsilon$ para todos os dispositivos considerados. Pode-se concluir que, se o custo com *overhearing* for desconsiderado, não existem arestas redundantes no grafo de conectividade G_{max} para nenhum dos modelos considerados.

Tabela 4.13. Valores máximos da expressão $d(a,c)^2 - d(a,b)^2 - d(b,c)^2$ e da expressão $(E_{Rx} + E_{Tx_{elec}})/\epsilon$ para os dispositivos *Atmel RF230*, *Chipcon CC2420* e *Chipcon CC1000*.

Modelo do Transceptor	Valor máximo da expressão $d(a,c)^2 - d(a,b)^2 - d(b,c)^2$	Valor da expressão $(E_{Rx} + E_{Tx_{elec}})/\epsilon$
Atmel RF230	11552.00	11602.28
Chipcon CC2420	5000.00	17686.57
Chipcon CC1000	45000.00	140400.00

4.5.4.6 Densidade da Rede

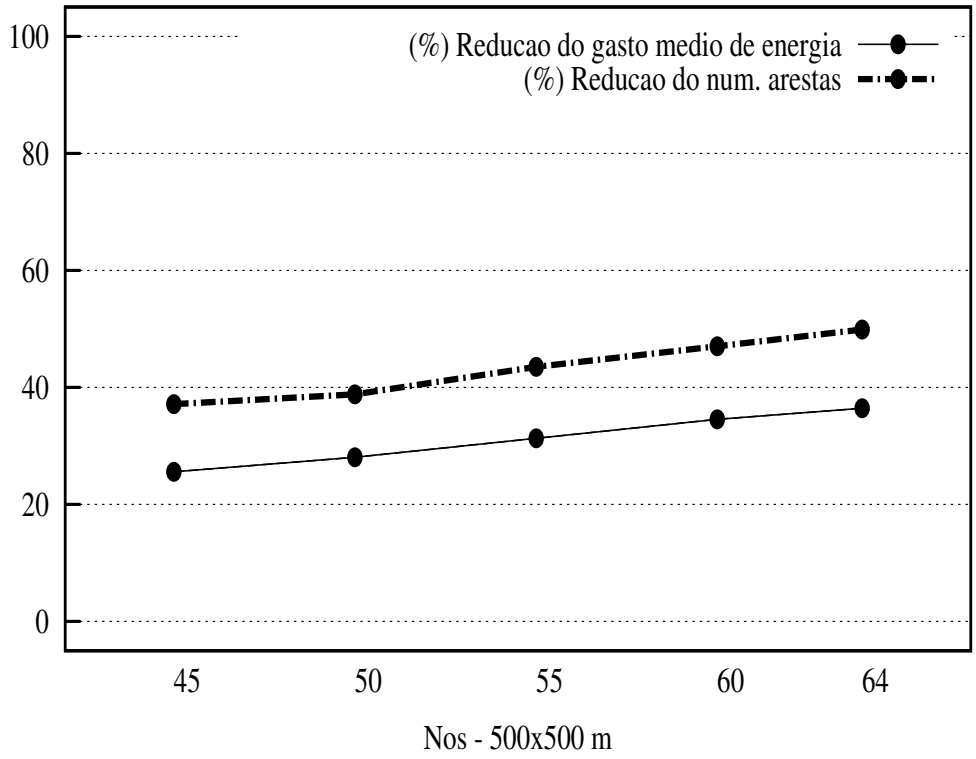
O comportamento dos algoritmos em função da densidade da rede (45, 50, 55, 60 e 64 nós distribuídos em uma área de 500x500) está sumarizado na Figura 4.6.

A linha tracejada representa a média aritmética da porcentagem das arestas reduzidas pelos algoritmos *L-2Red-OH*, *L-KRed-OH* e *G-KRed-OH* do grafo G_{max} para os dispositivos *Atmel RF230*, *Chipcon CC2420* e *Chipcon CC1000* (dados extraídos das Tabelas 4.2, 4.3, 4.4, 4.5 e 4.6).

Já a linha contínua representa a média aritmética da porcentagem das reduções do custo médio de transmissão, $AvEnergy(G_{max})$, obtidas pelos algoritmos *L-2Red-OH*, *L-KRed-OH* e *G-KRed-OH* para os dispositivos *Atmel RF230*, *Chipcon CC2420* e *Chipcon CC1000* (dados extraídos das Tabelas 4.7, 4.8, 4.9, 4.10 e 4.11).

De acordo com o gráfico apresentado na Figura 4.6, essas porcentagens (de arestas reduzidas e na redução do custo médio de transmissão) estão ligeiramente proporcionais à densidade da rede. Esse resultado está de acordo com a idéia apresentada em [33] que sugere que os impactos positivos da utilização de um algoritmo de Controle de Topologia são ainda mais evidentes em redes com alta densidade de nós.

Figura 4.6. Gráfico da média aritmética da redução (em %) do número de arestas e do gasto médio de energia para todas as simulações realizadas



CAPÍTULO 5

CONCLUSÕES E TRABALHOS FUTUROS

5.1 CONCLUSÕES

O Controle de Topologia é uma estratégia eficaz para a otimização do uso dos recursos de uma RSSF. Conforme discutido na seção 3.1.4, a redução no número de vizinhos de um nó pode permitir a redução da sua potência de transmissão. Um ganho adicional está relacionado ao tamanho das tabelas de roteamento mantidas por um nó. A redução do número de vizinhos torna o espaço necessário para manter essas informações menor. Outra vantagem associada ao Controle de Topologia é que as operações (computacionais) sobre um grafo G_{min} tendem a consumir uma quantidade de energia menor do que as operações sobre um grafo G_{max} - considerando que G_{min} tem menos arestas que G_{max}. De acordo com os valores relacionados ao consumo dos microcontroladores utilizados nos dispositivos apresentados na seção 2.4 (Tabelas 2.1, 2.3 e 2.6), o custo com processamento é significativo e uma otimização que permita reduzir a quantidade de informação a ser processada pode ser importante.

Esse trabalho apresentou um algoritmo para Controle de Topologia que leva em consideração o efeito *overhearing* e que elimina as arestas ditas *k-redundantes*, detectáveis localmente, do grafo de conectividade máxima G_{max}.

Foram realizadas simulações com várias topologias e cenários diferentes (45, 50, 55, 60 e 64 nós distribuídos em uma área de 500x500). Para cada um desses cenários, foram realizada simulações com valores de consumo baseados em três transceptores (*Atmel RF230* [8], *Chipcon CC2420* [10] e *Chipcon CC1000* [11]) usados por importantes plataformas de nós sensores.

As simulações foram feitas com os seguintes algoritmos de Controle de Topologia: *L-2Red-NoOH*, *L-2Red-OH*, *L-KRed-NoOH*, *L-KRed-OH*, *G-KRed-NoOH* e *G-KRed-OH*.

Tratam-se de variações de algoritmos de Controle de Topologia capazes de elimininar arestas *2-redundantes* ou *k-redundantes* (para $k \geq 2$), com ou sem considerar o efeito *overhearing* e executam local ou globalmente (a descrição mais detalhada de cada algoritmo foi feita na seção 4.5.1).

Um resultado importante das simulações foi que o algoritmo proposto nesse trabalho, *L-KRed-OH*, foi ligeiramente mais eficiente que o algoritmo que se propõe a eliminar apenas as arestas *2-redundantes*, *L-2Red-OH*. O número de arestas reduzidas por *L-KRed-OH* foi cerca de 2.87% maior que o número de arestas eliminadas por *L-2Red-OH*. Com relação ao custo médio de transmissão, $AvEnergy(G_{min})$, o algoritmo *L-KRed-OH* conseguiu reduzir o valor do custo médio de transmissão, $AvEnergy(G_{min})$, em média, 4.16% a mais que o algoritmo *L-2Red-OH*.

Os resultados mostraram ainda que o algoritmo proposto nesse trabalho e um algoritmo de Controle de Topologia global (que conhece toda a topologia da rede) se comportaram de maneira bastante similar. Em quase todas as simulações o número de arestas eliminadas foi o mesmo. Com relação ao gasto médio de energia, o comportamento dos algoritmos foi idêntico.

Outro resultado igualmente importante está relacionado ao efeito *overhearing*. Foi mostrado que não é possível reduzir arestas sem considerar *overhearing* para os transceptores considerados.

5.2 TRABALHOS FUTUROS

Nesse trabalho, considerou-se que o custo para transmitir através de arestas estava associado ao consumo global de energia. Não foram considerados os custos isolados de energia para cada nó. Em alguns cenários, o consumo de energia global pode levar à morte prematura de alguns nós. Assim, um possível trabalho futuro seria estudar o efeito do custo global em detrimento do custo isolado de cada nó na preservação da conectividade da rede.

Uma outra possível continuação desse trabalho seria utilizar/elaborar um modelo de energia que levasse em consideração o custo com processamento e analisar o impacto desse custo no Controle de Topologia.

APÊNDICE A

ALGORITMO L-2RED-NOOH PARA O NS-2

```
###############################################################################
#
#   L-2Red-NoOH Algorithm
#
# The  L-2Red-NoOH  Algorithm   is  an implementation of the SMECN  Algorithm.
# According to the original implementation, SMECN is a distributed (localized)
# algorithm  that  eliminates  the  2-redundant  edges  but does  not take the
# effects of  overhearing into consideration.  This implementation is  similar
# to     L-2Red-OH     Algorithm,    except   the   implementation   of    the
# 'neighboursInRange'  procedure.  We  will omit here the pieces of code that
# are  the  same  in  both  cases. For a complete  listing  of the L-2Red-NoOH
# Algorithm, see the procedures defined in L-2Red-OH Algorithm.
#
###############################################################################

###############################################################################
#
# Procedure to calculate number of neighbours in a range.
# As  we are not  considering the  overhearing effect,  this procedure always
# returns the value 1.
#
###############################################################################
Agent/MessagePassing/OMCT instproc neighboursInRange {nx ny n2x n2y Nxy} {
    return 1;
}

###############################################################################
#
#   End of the L-2Red-NoOH Algorithm
#
###############################################################################
```

APÊNDICE B

ALGORITMO L-2RED-OH PARA O NS-2

```
#============================================================================
#
#   L-2Red-OH Algorithm
#
# The  code  below  is  a variation of the SMECN. A distributed (localized)
# algorithm for topology control in wireless sensor networks that eliminates
# the  2-redundant  edges.  This  implementation  takes   the   effects  of
# overhearing into consideration.
#
#============================================================================


#============================================================================
#
#     Global Definitions
#
#============================================================================
set BROADCAST_ADDR -1

set OMCT_PORT 42

# Process tolology information (num_nodes, x,y nodes coordinates)
source "topoValues.tcl"
source "topo${num_nodes}Nodes${val(x)}x${val(y)}"

# Process input parameter (CC1000, CC2420 or RF230)
set arg1 [lindex $argv 0]

## Read radio configuration parameters:
## val(RadioModel), val(rxPower), val(txPowerFixo), val(txPowerCoeficienteE).
source "config${arg1}.tcl"


set MINTxPOWER 0.0045
set MAXTxPOWER [expr $MINTxPOWER * 4]
set ROUND [expr 0.02 * 4 * $num_nodes]

set XY          0
set RNEIGHBS    1


set val(chan)           Channel/WirelessChannel  ;#Channel Type
set val(prop)           Propagation/FreeSpace    ;# radio-propagation model
set val(netif)          Phy/WirelessPhy          ;# network interface type
set val(mac)            Mac/Tdma                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue  ;# interface queue type
set val(ll)             LL                       ;# link layer type
set val(ant)            Antenna/OmniAntenna      ;# antenna model
set val(ifqlen)         50                       ;# max packet in ifq
set val(rp)             DumbAgent
set val(em)             "EnergyModel"

#============================================================================
#
```

```
# Creating simulator and trace files and setting up topography object
#
# ============================================================================
set ns [new Simulator]

set f [open traceFile.tr w]
$ns trace-all $f
$ns use-newtrace

set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

# ============================================================================
#
# Create God node
#
# ============================================================================
create-god $num_nodes

# ============================================================================
#
# Node parameters
#
# ============================================================================
set chan_1_ [new $val(chan)]

$ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -topoInstance $topo \
                -agentTrace OFF \
                -routerTrace OFF \
                -macTrace OFF \
                -movementTrace OFF \
                -channel $chan_1_ \
                -energyModel $val(em) \
                -rxPower $val(rxp) \
                -txPower $val(txp) \
                -initialEnergy $val(inp)


# ============================================================================
#
# Agents Specification: Topology with Minimum Cost Property
#
# ============================================================================

Class Agent/MessagePassing/OMCT -superclass Agent/MessagePassing

# ============================================================================
#
# Procedure to initiate agent variables
#
# ============================================================================
Agent/MessagePassing/OMCT instproc initiateVars {} {
    $self instvar A_set Nbrs  node_ myindex myx myy A_XY_list XY_set
    global MAXTxPOWER
```

```
    set A_set {}      ; # Set of all neighbours (reachable with max power)
    set A_XY_list {}  ; # List of the coordinates of my neighbours
    set Nbrs {}       ; # Optimized set of neighbours

    set myindex [$node_ node-addr]
    set myx [$node_ set X_]
    set myy [$node_ set Y_]

    set XY_set($myindex) [list $myx $myy]

    set transPot $MAXTxPOWER
    set if_ [$node_ set netif_(0)]
    $if_ set Pt_ $transPot
}

#############################################################################
#
# Procedure to send a message with the node's position
#
#############################################################################
Agent/MessagePassing/OMCT instproc sendPosition {} {
    $self instvar myx myy
    global BROADCAST_ADDR OMCT_PORT XY

    set size 200
    set data "OURPROT:$XY:$myx:$myy"

    $self sendto $size $data $BROADCAST_ADDR $OMCT_PORT
}

#############################################################################
#
# Procedure to send a message with the neighbour's position
#
#############################################################################
Agent/MessagePassing/OMCT instproc sendReachableNeighbs {} {
    $self instvar A_set A_XY_list myindex
    global BROADCAST_ADDR OMCT_PORT RNEIGHBS

    set size 200
    set data "OURPROT:$RNEIGHBS:$A_set:$A_XY_list"

    $self sendto $size $data $BROADCAST_ADDR $OMCT_PORT
}

#############################################################################
#
# Procedure to process a received message
#
#############################################################################
Agent/MessagePassing/OMCT instproc recv {source sport size data} {
    $self instvar A_set XY_set nbs nbsxy_set A_XY_list optnbrs myindex
    global XY RNEIGHBS OPTNEIGHBS

    set typemsg [lindex [split $data ":"] 1]

    if {$typemsg == $XY} {
        lappend A_set $source
        set posx [lindex [split $data ":"] 2]
        set posy [lindex [split $data ":"] 3]
        set XY_set($source) [list $posx $posy]
        lappend A_XY_list $XY_set($source)
    } elseif {$typemsg == $RNEIGHBS} {
```

```
        set nbs($source) [lindex [split $data ":"] 2]
        set nbsxy_set($source) [lindex [split $data ":"] 3]
    }
}

##############################################################################
#
# Procedure to calculate number of neighbours in a range
#
##############################################################################
Agent/MessagePassing/OMCT instproc neighboursInRange {nx ny n2x n2y Nxy} {
    set ctr 0
    set dist [distance $nx $ny $n2x $n2y]
    for {set i 0} {$i < [llength $Nxy]} {incr i} {
        set dist2 [distance $nx $ny [lindex [lindex $Nxy $i] 0]
                                    [lindex [lindex $Nxy $i] 1]]
        if {$dist2 > 0.0} {
            if {$dist2 <= $dist} {
                incr ctr
            }
        }
    }
    return $ctr
}

##############################################################################
#
# Procedure to calculate the optimized set of neighbours
#
##############################################################################
Agent/MessagePassing/OMCT instproc calculateOptimizedNeighbs {} {
    $self instvar A_set A_XY_list XY_set Nbrs myindex myx myy nbs nbsxy_set
    global PATHLOSS_EXPONENT val

    set NonNbrs {}

    for {set i 0} {$i < [llength $A_set]} {incr i} {
        for {set j 0} {$j < [llength $A_set]} {incr j} {
            if {[$self inRelayRegion [lindex $A_set $i] [lindex $A_set $j]] == 1} {
                if {[lsearch $NonNbrs [lindex $A_set $i]] == -1} {
                    lappend NonNbrs [lindex $A_set $i]
                }
            }
        }
    }

    # Calculate optimized neighbours set
    set Nbrs {}
    for {set i 0} {$i < [llength $A_set]} {incr i} {
        if {[lsearch $NonNbrs [lindex $A_set $i]] == -1} {
            lappend Nbrs [lindex $A_set $i]
        }
    }
}

##############################################################################
#
# Procedure to calculate if a node is in another node's relay region.
# It returns 1 if the node 'node' is in the relay region of the node 'relay'.
# Otherwise, it returns 0.
#
##############################################################################
Agent/MessagePassing/OMCT instproc inRelayRegion {node relay} {
```

```
   $self instvar XY_set A_XY_list node_ myindex myx myy A_set nbs nbsxy_set
   global PATHLOSS_EXPONENT val

    if { [lsearch $nbs($relay) $node] == -1 } {
        return 0
    }

    set nodex [lindex $XY_set($node) 0]
    set nodey [lindex $XY_set($node) 1]

    set relayx [lindex $XY_set($relay) 0]
    set relayy [lindex $XY_set($relay) 1]

    # cost myself -> node:
    set numberOfNbMyselfToNode [$self neighboursInRange
                                $myx $myy $nodex $nodey $A_XY_list]
    set costMyselfNode [cost [distance $myx $myy $nodex $nodey]
                                $numberOfNbMyselfToNode]


    # cost myself -> relay:
    set numberOfNbMyselfRelay [$self neighboursInRange $myx $myy $relayx
                                $relayy $A_XY_list]
    set costMyselfRelay [cost [distance $myx $myy $relayx $relayy]
                                $numberOfNbMyselfRelay]


    # cost relay -> node:
    set numberOfNbRelayToNode [$self neighboursInRange $relayx $relayy
                                $nodex $nodey $nbsxy_set($relay)]
    set costRelayNode [cost [distance $relayx $relayy $nodex $nodey]
                                $numberOfNbRelayToNode]

    # compare costs:
    if {$costMyselfNode <= [expr $costMyselfRelay + $costRelayNode]} {
        return 0
    } else {
        return 1
    }
}

#############################################################################
#
# Procedure that triggers the algorithm
# At the end of this procedure, each node has a set with its
# optimized neighbours
#
#############################################################################
Agent/MessagePassing/OMCT instproc findOptimizedNeighbourhood {} {
    global ns ROUND

    set inst1 $ROUND
    set inst2 [expr $ROUND * 2]
    set inst3 [expr $ROUND * 3]

    $ns at $inst1 "$self sendPosition"              ;   # Start Phase 1
    $ns at $inst2 "$self sendReachableNeighbs"      ;   # Start Phase 2
    $ns at $inst3 "$self calculateOptimizedNeighbs" ;   # Start Phase 3
}

#############################################################################
#
#   Defining Topology
```

```
#   Creating nodes
#
#############################################################################
for {set i 0} {$i < $num_nodes} {incr i} {
    set n($i) [$ns node]
}

#############################################################################
#
#   Attaching agents to nodes
#   Attaching a new Agent/MessagePassing/OMCT to each node on port
#   $OMCT_PORT
#
#############################################################################
for {set i 0} {$i < $num_nodes} {incr i} {
    set omct($i) [new Agent/MessagePassing/OMCT]
    $n($i) attach  $omct($i) $OMCT_PORT
    $omct($i) initiateVars
}

#############################################################################
#
#   Setting up starting events
#
#############################################################################
for {set i 0} {$i < $num_nodes} {incr i} {
    $ns at 0 "$omct($i) findOptimizedNeighbourhood"
}

$ns at 2000 "finish"

#############################################################################
#
# Global procedures
#
#############################################################################

#############################################################################
#
# Procedure to calculate distance between two nodes
#
#############################################################################
proc distance {x1 y1 x2 y2} {
    return [expr sqrt(pow($x1 - $x2,2) + pow($y1 - $y2,2))]
}

#############################################################################
#
# Procedure to calculate edge cost
#
#############################################################################
proc cost {distance numVizinhos} {
    global val PATHLOSS_EXPONENT
    return [expr $val(txPowerFixo) +
            ( pow($distance ,$PATHLOSS_EXPONENT)
                * $val(txPowerCoeficienteE) ) +
            ($numVizinhos * $val(rxPower)) ]
}

#############################################################################
#
# Procedure to plot the optimized topology graph
#
```

```
#############################################################################
proc plot-optimized-neighb-graph {} {
    global num_nodes n omct val logFile PATHLOSS_EXPONENT

    set ng [open otimizedGraph.xg w]

    for {set i 0} {$i < $num_nodes} {incr i} {
        for {set j 0} {$j < [llength [$omct($i) set Nbrs]]} {incr j} {
            if {$j == 0} {
                puts $ng "\n[$n($i) set X_] [$n($i) set Y_]"
                puts $ng "[$n([lindex [$omct($i) set Nbrs] $j]) set X_]
                          [$n([lindex [$omct($i) set Nbrs] $j]) set Y_]"
            } else {
                puts $ng "[$n($i) set X_] [$n($i) set Y_]"
                puts $ng "[$n([lindex [$omct($i) set Nbrs] $j]) set X_]
                          [$n([lindex [$omct($i) set Nbrs] $j]) set Y_]"
            }
        }
    }
    close $ng
}

#############################################################################
#
# Procedure to plot the original topology graph
#
#############################################################################
proc plot-maxpower-graph {} {
    global num_nodes n omct val logFile PATHLOSS_EXPONENT

    set mpg [open originalGraph.xg w]

    for {set i 0} {$i < $num_nodes} {incr i} {
        for {set j 0} {$j < [llength [$omct($i) set A_set]]} {incr j} {
            if {$j == 0} {
                puts $mpg "\n[$n($i) set X_] [$n($i) set Y_]"
                puts $mpg "[$n([lindex [$omct($i) set A_set] $j]) set X_]
                           [$n([lindex [$omct($i) set A_set] $j]) set Y_]"
            } else {
                puts $mpg "[$n($i) set X_] [$n($i) set Y_]"
                puts $mpg "[$n([lindex [$omct($i) set A_set] $j]) set X_]
                           [$n([lindex [$omct($i) set A_set] $j]) set Y_]"
            }
        }
    }
    close $mpg
}

#############################################################################
#
#   Finishing the algorithm
#
#############################################################################
proc finish {} {
    global ns f nf
    close $f
    plot-optimized-neighb-graph
    plot-maxpower-graph
    exit 0
}

#############################################################################
#
```

```
#   Running ns
#
################################################################################
$ns run

################################################################################
#
#   End of the L-2Red-OH Algorithm
#
################################################################################
```

APÊNDICE C

ALGORITMO L-KRED-NOOH PARA O NS-2

```
#############################################################################
#
#   L-KRed-NoOH Algorithm
#
# This algorithm is a varitation of the L-KRed-OH algorithm. This algorithm does not
# consider   the   overhering   effect. This implementation  is  similar to
# L-KRed-OH Algorithm, except for the implementation of the 'neighboursInRange'
# procedure. We will omit here the pieces of code that are the same in both
# cases.
# For a complete listing  of  the L-KRed-NoOH  Algorithm,  see  the
# procedures defined in L-KRed-OH Algorithm.
#
#############################################################################

#############################################################################
#
# Procedure to calculate number of neighbours in a range.
# As we are not considering the overhearing effect, this procedure always
# returns the value 1.
#
#############################################################################
Agent/MessagePassing/OMCT instproc neighboursInRange {nx ny n2x n2y Nxy} {
    return 1;
}

#############################################################################
#
#   End of the L-KRed-NoOH Algorithm
#
#############################################################################
```

APÊNDICE D

ALGORITMO L-KRED-OH PARA O NS-2

```
#============================================================================
#
#   L-KRed-OH Algorithm
#
# The code below describes a distributed (localized) algorithm  for topology
# control in wireless  sensor networks.  Our approach takes  the effects of
# overhearing  into    consideration   and  that  it may eliminate more
# communication links from  a given  connectivity graph, and  thus possibly
# assign  lower   transmission   power  to  some  nodes. This  is  done  by
# eliminating  so-called  k-redundant  edges,  instead  of eliminating only
# two-redundant edges.
#
#============================================================================


#============================================================================
#
#     Global Definitions
#
#============================================================================
set BROADCAST_ADDR -1

set OMCT_PORT 42

# Process tolology information (num_nodes, x,y nodes coordinates)
source "topoValues.tcl"
source "topo${num_nodes}Nodes${val(x)}x${val(y)}"

# Process input parameter (CC1000, CC2420 or RF230)
set arg1 [lindex $argv 0]

## Read radio configuration parameters:
## val(RadioModel), val(rxPower), val(txPowerFixo), val(txPowerCoeficienteE).
source "config${arg1}.tcl"


set MINTxPOWER 0.0045
set MAXTxPOWER [expr $MINTxPOWER * 4]
set ROUND [expr 0.02 * 4 * $num_nodes]

set XY          0
set RNEIGHBS    1


set val(chan)           Channel/WirelessChannel  ;#Channel Type
set val(prop)           Propagation/FreeSpace    ;# radio-propagation model
set val(netif)          Phy/WirelessPhy          ;# network interface type
set val(mac)            Mac/Tdma                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue  ;# interface queue type
set val(ll)             LL                       ;# link layer type
set val(ant)            Antenna/OmniAntenna      ;# antenna model
set val(ifqlen)         50                       ;# max packet in ifq
set val(rp)             DumbAgent
set val(em)             "EnergyModel"
```

```
#############################################################################
#
# Creating simulator and trace files and setting up topography object
#
#############################################################################
set ns [new Simulator]

set f [open traceFile.tr w]
$ns trace-all $f
$ns use-newtrace

set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

#############################################################################
#
# Create God node
#
#############################################################################
create-god $num_nodes

#############################################################################
#
# Node parameters
#
#############################################################################
set chan_1_ [new $val(chan)]

$ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -topoInstance $topo \
                -agentTrace OFF \
                -routerTrace OFF \
                -macTrace OFF \
                -movementTrace OFF \
                -channel $chan_1_ \
                -energyModel $val(em) \
                -rxPower $val(rxp) \
                -txPower $val(txp) \
                -initialEnergy $val(inp)


#############################################################################
#
# Agents Specification: Topology with Minimum Cost Property
#
#############################################################################

Class Agent/MessagePassing/OMCT -superclass Agent/MessagePassing

#############################################################################
#
# Procedure to initiate agent variables
#
#############################################################################
Agent/MessagePassing/OMCT instproc initiateVars {} {
```

```
    $self instvar A_set Nbrs  node_ myindex myx myy A_XY_list XY_set
    global MAXTxPOWER

    set A_set {}      ; # Set of all neighbours (reachable with max power)
    set A_XY_list {}  ; # List of the coordinates of my neighbours
    set Nbrs {}       ; # Optimized set of neighbours

    set myindex [$node_ node-addr]
    set myx [$node_ set X_]
    set myy [$node_ set Y_]

    set XY_set($myindex) [list $myx $myy]

    set transPot $MAXTxPOWER
    set if_ [$node_ set netif_(0)]
    $if_ set Pt_ $transPot
}

###############################################################################
#
# Procedure to send a message with the node's position
#
###############################################################################
Agent/MessagePassing/OMCT instproc sendPosition {} {
    $self instvar myx myy
    global BROADCAST_ADDR OMCT_PORT XY

    set size 200
    set data "OURPROT:$XY:$myx:$myy"

    $self sendto $size $data $BROADCAST_ADDR $OMCT_PORT
}

###############################################################################
#
# Procedure to send a message with the neighbour's position
#
###############################################################################
Agent/MessagePassing/OMCT instproc sendReachableNeighbs {} {
    $self instvar A_set A_XY_list myindex
    global BROADCAST_ADDR OMCT_PORT RNEIGHBS

    set size 200
    set data "OURPROT:$RNEIGHBS:$A_set:$A_XY_list"

    $self sendto $size $data $BROADCAST_ADDR $OMCT_PORT
}

###############################################################################
#
# Procedure to process a received message
#
###############################################################################
Agent/MessagePassing/OMCT instproc recv {source sport size data} {
    $self instvar A_set XY_set nbs nbsxy_set A_XY_list optnbrs myindex
    global XY RNEIGHBS OPTNEIGHBS

    set typemsg [lindex [split $data ":"] 1]

    if {$typemsg == $XY} {
        lappend A_set $source
        set posx [lindex [split $data ":"] 2]
        set posy [lindex [split $data ":"] 3]
```

```
        set XY_set($source) [list $posx $posy]
        lappend A_XY_list $XY_set($source)
    } elseif {$typemsg == $RNEIGHBS} {
        set nbs($source) [lindex [split $data ":"] 2]
        set nbsxy_set($source) [lindex [split $data ":"] 3]
    }
}

#############################################################################
#
# Procedure to calculate number of neighbours in a range
#
#############################################################################
Agent/MessagePassing/OMCT instproc neighboursInRange {nx ny n2x n2y Nxy} {
    set ctr 0
    set dist [distance $nx $ny $n2x $n2y]
    for {set i 0} {$i < [llength $Nxy]} {incr i} {
        set dist2 [distance $nx $ny [lindex [lindex $Nxy $i] 0]
                                    [lindex [lindex $Nxy $i] 1]]
        if {$dist2 > 0.0} {
            if {$dist2 <= $dist} {
                incr ctr
            }
        }
    }
    return $ctr
}

#############################################################################
#
# Procedure to calculate the optimized set of neighbours
#
#############################################################################
Agent/MessagePassing/OMCT instproc calculateOptimizedNeighbs {} {
    $self instvar A_set A_XY_list XY_set Nbrs myindex myx myy nbs nbsxy_set
    global PATHLOSS_EXPONENT valZ

    # Create cost graph based on local information:
    set nbsl($myindex) $A_set
    set nbsc($myindex) {}
    for {set i 0} {$i < [llength $A_set]} {incr i} {
        set ni [lindex $A_set $i]
        set nbx [lindex $XY_set($ni) 0]
        set nby [lindex $XY_set($ni) 1]
        set nc [expr $val(txPowerFixo) +
                (pow([distance $myx $myy $nbx $nby],$PATHLOSS_EXPONENT) *
                $val(txPowerCoeficienteE))]
        set nnr [$self neighboursInRange $myx $myy $nbx $nby $A_XY_list]
        lappend nbsc($myindex) [list $nc $nnr]
    }

    for {set i 0} {$i < [llength $A_set]} {incr i} {
        set niindex [lindex $A_set $i]
        set nbsl($niindex) {}
        set nbsl($niindex) $nbs($niindex)
        set nix [lindex $XY_set($niindex) 0]
        set niy [lindex $XY_set($niindex) 1]
        for {set j 0} {$j < [llength $nbsl($niindex)]} {incr j} {
            set njindex [lindex $nbsl($niindex) $j]
            set njx [lindex [lindex $nbsxy_set($niindex) $j] 0]
            set njy [lindex [lindex $nbsxy_set($niindex) $j] 1]
            set nc [expr $val(txPowerFixo) +
                (pow([distance $nix $niy $njx $njy],$PATHLOSS_EXPONENT) *
```

```
                $val(txPowerCoeficienteE))]
            set nnr [$self neighboursInRange $nix $niy $njx $njy
                                             $nbsxy_set($niindex)]
            lappend nbsc($niindex) [list $nc $nnr]
        }
    }

    # Calculate shortest cost paths
    for {set i 0} {$i < [llength $A_set]} {incr i} {
        set indexVizinho [lindex $A_set $i]
        set cost($indexVizinho) -1
        set fwrdn($indexVizinho) -1

        for {set i2 0} {$i2 < [llength $nbs($indexVizinho)]} {incr i2} {
            set indexVizinhoDoVizinho [lindex $nbs($indexVizinho) $i2]
            set cost($indexVizinhoDoVizinho) -1
            set fwrdn($indexVizinhoDoVizinho) -1
        }
    }

    set cost($myindex) 0
    set fwrdn($myindex) -1
    set lvisited {}
    lappend lvisited $myindex
    set i 0

    while {$i < [llength $lvisited]} {
        set nt [lindex $lvisited $i]
        for {set j 0} {$j < [llength $nbsl($nt)]} {incr j} {
            set nbi [lindex $nbsl($nt) $j]
            if {[lsearch $A_set $nbi] != -1} {
                set txcost [lindex [lindex $nbsc($nt) $j] 0]
                set rxcost [expr ([lindex [lindex $nbsc($nt) $j] 1]) *
                            $val(rxPower)]
                if {($cost($nbi) == -1) ||
                    ($cost($nbi) > [expr $cost($nt) + $txcost+$rxcost])} {
                    set cost($nbi) [expr $cost($nt)+$txcost+$rxcost]
                    lappend lvisited $nbi
                    set fwrdn($nbi) $nt
                }
            }
        }
        incr i
    }

    # Set the optimized neighbours set
    set Nbrs {}
    for {set i 0} {$i < [llength $A_set]} {incr i} {
        set ni [lindex $A_set $i]
        if {$fwrdn($ni) == $myindex} {
            lappend Nbrs $ni
        }
    }
}


##############################################################################
#
# Procedure that trigger the algorithm
# At the end of this procedure, each nodes has a set with its
# optimized neighbours
#
##############################################################################
```

```
Agent/MessagePassing/OMCT instproc findOptimizedNeighbourhood {} {
    global ns ROUND

    set inst1 $ROUND
    set inst2 [expr $ROUND * 2]
    set inst3 [expr $ROUND * 3]

    $ns at $inst1 "$self sendPosition"              ;   # Start Phase 1
    $ns at $inst2 "$self sendReachableNeighbs"      ;   # Start Phase 2
    $ns at $inst3 "$self calculateOptimizedNeighbs" ;   # Start Phase 3
}

#############################################################################
#
#   Defining Topology
#   Creating nodes
#
#############################################################################
for {set i 0} {$i < $num_nodes} {incr i} {
    set n($i) [$ns node]
}

#############################################################################
#
#   Attaching agents to nodes
#   Attaching a new Agent/MessagePassing/OMCT to each node on port
#   $OMCT_PORT
#
#############################################################################
for {set i 0} {$i < $num_nodes} {incr i} {
    set omct($i) [new Agent/MessagePassing/OMCT]
    $n($i) attach  $omct($i) $OMCT_PORT
    $omct($i) initiateVars
}

#############################################################################
#
#   Setting up starting events
#
#############################################################################
for {set i 0} {$i < $num_nodes} {incr i} {
    $ns at 0 "$omct($i) findOptimizedNeighbourhood"
}

$ns at 2000 "finish"

#############################################################################
#
# Global procedures
#
#############################################################################

#############################################################################
#
# Procedure to calculate distance between two nodes
#
#############################################################################
proc distance {x1 y1 x2 y2} {
    return [expr sqrt(pow($x1 - $x2,2) + pow($y1 - $y2,2))]
}

#############################################################################
#
```

```
# Procedure to calculate edge cost
#
#============================================================================
proc cost {distance numVizinhos} {
    global val PATHLOSS_EXPONENT
    return [expr $val(txPowerFixo) +
            ( pow($distance ,$PATHLOSS_EXPONENT)
                * $val(txPowerCoeficienteE) ) +
            ($numVizinhos * $val(rxPower)) ]
}

#============================================================================
#
# Procedure to plot the optimized topology graph
#
#============================================================================
proc plot-optimized-neighb-graph {} {
    global num_nodes n omct val logFile PATHLOSS_EXPONENT

    set ng [open otimizedGraph.xg w]

    for {set i 0} {$i < $num_nodes} {incr i} {
        for {set j 0} {$j < [llength [$omct($i) set Nbrs]]} {incr j} {
            if {$j == 0} {
                puts $ng "\n[$n($i) set X_] [$n($i) set Y_]"
                puts $ng "[$n([lindex [$omct($i) set Nbrs] $j]) set X_]
                          [$n([lindex [$omct($i) set Nbrs] $j]) set Y_]"
            } else {
                puts $ng "[$n($i) set X_] [$n($i) set Y_]"
                puts $ng "[$n([lindex [$omct($i) set Nbrs] $j]) set X_]
                          [$n([lindex [$omct($i) set Nbrs] $j]) set Y_]"
            }
        }
    }
    close $ng
}

#============================================================================
#
# Procedure to plot the original topology graph
#
#============================================================================
proc plot-maxpower-graph {} {
    global num_nodes n omct val logFile PATHLOSS_EXPONENT

    set mpg [open originalGraph.xg w]

    for {set i 0} {$i < $num_nodes} {incr i} {
        for {set j 0} {$j < [llength [$omct($i) set A_set]]} {incr j} {
            if {$j == 0} {
                puts $mpg "\n[$n($i) set X_] [$n($i) set Y_]"
                puts $mpg "[$n([lindex [$omct($i) set A_set] $j]) set X_]
                           [$n([lindex [$omct($i) set A_set] $j]) set Y_]"
            } else {
                puts $mpg "[$n($i) set X_] [$n($i) set Y_]"
                puts $mpg "[$n([lindex [$omct($i) set A_set] $j]) set X_]
                           [$n([lindex [$omct($i) set A_set] $j]) set Y_]"
            }
        }
    }
    close $mpg
}
```

```
#############################################################################
#
#   Finishing the algorithm
#
#############################################################################
proc finish {} {
    global ns f nf
    close $f
    plot-optimized-neighb-graph
    plot-maxpower-graph
    exit 0
}

#############################################################################
#
#   Running ns
#
#############################################################################
$ns run

#############################################################################
#
#   End of the L-KRed-OH Algorithm
#
#############################################################################
```

APÊNDICE E

ALGORITMO G-KRED-NOOH PARA O NS-2

```
#############################################################################
#
#   G-KRed-NoOH Algorithm
#
# This algorithm is a variation of the G-KRed-OH  algorithm. This algorithm does not
# consider   the   overhering    effect.  This  implementation  is  similar  to
# G-KRed-OH Algorithm, except for the implementation of the 'neighboursInRange'
# procedure. We will omit here the pieces of code that are the  same  in  both
# cases.  For  a  complete  listing  of  the  G-KRed-NoOH  Algorithm,  see  the
# procedures defined in G-KRed-OH Algorithm.
#
#############################################################################

#############################################################################
#
# Procedure to calculate number of neighbours in a range.
# As we are not considering the overhearing effect, this procedure always
# returns the value 1.
#
#############################################################################
Agent/MessagePassing/OMCT instproc neighboursInRange {nx ny n2x n2y Nxy} {
    return 1;
}

#############################################################################
#
#   End of the G-KRed-NoOH Algorithm
#
#############################################################################
```

APÊNDICE F

ALGORITMO G-KRED-OH PARA O NS-2

```
#============================================================================
#
#   G-KRed-OH Algorithm
#
# This algorithm is a variation of the L-KRed-OH algorithm. The G-KRed-OH is
# a distributed algorithm for topology control that eliminates the so-called
# k-redundant  edges and take the effects of overhearing into consideration.
# Unlike the algorithms L-KRed-OH, L-KRed-NotOH, L-2Red-OH and L-2Red-NotOH,
# in this implementation, the nodes have the complete information about  the
# network topology.
#
#============================================================================

#============================================================================
#
#     Global Definitions
#
#============================================================================
set BROADCAST_ADDR -1

set OMCT_PORT 42

# Process topology information (num_nodes, x,y nodes coordinates)
source "topoValues.tcl"
source "topo${num_nodes}Nodes${val(x)}x${val(y)}"

# Process input parameter (CC1000, CC2420 or RF230)
set arg1 [lindex $argv 0]

## Read radio configuration parameters:
## val(RadioModel), val(rxPower), val(txPowerFixo), val(txPowerCoeficienteE).
source "config${arg1}.tcl"


set MINTxPOWER 0.0045
set MAXTxPOWER [expr $MINTxPOWER * 4]
set ROUND [expr 0.02 * 4 * $num_nodes]

set XY          0
set RNEIGHBS    1


set val(chan)           Channel/WirelessChannel  ;#Channel Type
set val(prop)           Propagation/FreeSpace    ;# radio-propagation model
set val(netif)          Phy/WirelessPhy          ;# network interface type
set val(mac)            Mac/Tdma                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue  ;# interface queue type
set val(ll)             LL                       ;# link layer type
set val(ant)            Antenna/OmniAntenna      ;# antenna model
set val(ifqlen)         50                       ;# max packet in ifq
set val(rp)             DumbAgent
set val(em)             "EnergyModel"

#============================================================================
```

```
#
# Creating simulator and trace files and setting up topography object
#
#############################################################################
set ns [new Simulator]

set f [open traceFile.tr w]
$ns trace-all $f
$ns use-newtrace

set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)

#############################################################################
#
# Create God node
#
#############################################################################
create-god $num_nodes

#############################################################################
#
# Node parameters
#
#############################################################################
set chan_1_ [new $val(chan)]

$ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -topoInstance $topo \
                -agentTrace OFF \
                -routerTrace OFF \
                -macTrace OFF \
                -movementTrace OFF \
                -channel $chan_1_ \
                -energyModel $val(em) \
                -rxPower $val(rxp) \
                -txPower $val(txp) \
                -initialEnergy $val(inp)


#############################################################################
#
# Agents Specification: Topology with Minimum Cost Property
#
#############################################################################

Class Agent/MessagePassing/OMCT -superclass Agent/MessagePassing

#############################################################################
#
# Procedure to initiate agent variables
#
#############################################################################
Agent/MessagePassing/OMCT instproc initiateVars {} {
    $self instvar A_set Nbrs  node_ myindex myx myy A_XY_list XY_set
    global MAXTxPOWER
```

```
    set A_set {}      ; # Set of all neighbours (reachable with max power)
    set A_XY_list {}  ; # List of the coordinates of my neighbours
    set Nbrs {}       ; # Optimized set of neighbours

    set myindex [$node_ node-addr]
    set myx [$node_ set X_]
    set myy [$node_ set Y_]

    set XY_set($myindex) [list $myx $myy]

    set transPot $MAXTxPOWER
    set if_ [$node_ set netif_(0)]
    $if_ set Pt_ $transPot
}

#############################################################################
#
# Procedure to send a message with the node's position
#
#############################################################################
Agent/MessagePassing/OMCT instproc sendPosition {} {
    $self instvar myx myy
    global BROADCAST_ADDR OMCT_PORT XY

    set size 200
    set data "OURPROT:$XY:$myx:$myy"

    $self sendto $size $data $BROADCAST_ADDR $OMCT_PORT
}

#############################################################################
#
# Procedure to send a message with the neighbour's position
#
#############################################################################
Agent/MessagePassing/OMCT instproc sendReachableNeighbs {} {
    $self instvar A_set A_XY_list myindex
    global BROADCAST_ADDR OMCT_PORT RNEIGHBS

    set size 200
    set data "OURPROT:$RNEIGHBS:$A_set:$A_XY_list"

    $self sendto $size $data $BROADCAST_ADDR $OMCT_PORT
}

#############################################################################
#
# Procedure to process a received message
#
#############################################################################
Agent/MessagePassing/OMCT instproc recv {source sport size data} {
    $self instvar A_set XY_set nbs nbsxy_set A_XY_list optnbrs myindex
    global XY RNEIGHBS OPTNEIGHBS

    set typemsg [lindex [split $data ":"] 1]

    if {$typemsg == $XY} {
        lappend A_set $source
        set posx [lindex [split $data ":"] 2]
        set posy [lindex [split $data ":"] 3]
        set XY_set($source) [list $posx $posy]
        lappend A_XY_list $XY_set($source)
```

```
    } elseif {$typemsg == $RNEIGHBS} {
        set nbs($source) [lindex [split $data ":"] 2]
        set nbsxy_set($source) [lindex [split $data ":"] 3]
    }
}

#############################################################################
#
# Procedure to calculate number of neighbours in a range
#
#############################################################################
Agent/MessagePassing/OMCT instproc neighboursInRange {nx ny n2x n2y Nxy} {
    set ctr 0
    set dist [distance $nx $ny $n2x $n2y]
    for {set i 0} {$i < [llength $Nxy]} {incr i} {
        set dist2 [distance $nx $ny [lindex [lindex $Nxy $i] 0]
                                    [lindex [lindex $Nxy $i] 1]]
        if {$dist2 > 0.0} {
            if {$dist2 <= $dist} {
                incr ctr
            }
        }
    }
    return $ctr
}

#############################################################################
#
# Procedure to calculate the optimized set of neighbours
#
#############################################################################
Agent/MessagePassing/OMCT instproc calculateOptimizedNeighbs {} {
    $self instvar A_set A_XY_list XY_set Nbrs myindex myx myy nbs nbsxy_set
    global PATHLOSS_EXPONENT val n omct fwrdn vizinhos num_nodes

    for {set i 0} {$i < $num_nodes} {incr i} {
        set cost($i) -1
        set fwrdn($i) -1
    }

    set cost($myindex) 0
    set fwrdn($myindex) -1
    set lvisited {}
    lappend lvisited $myindex
    set i 0
    while {$i < [llength $lvisited]} {
        set nt [lindex $lvisited $i]
        for {set j 0} {$j < [llength $vizinhos($nt)]} {incr j} {
            set nbi [lindex $vizinhos($nt) $j]
            set ntx [$n($nt) set X_]
            set nty [$n($nt) set Y_]
            set nbix [$n($nbi) set X_]
            set nbiy [$n($nbi) set Y_]
            set txcost [cost [distance $ntx $nty $nbix $nbiy]]
            set numberOfNb [$self numberNeighboursInRange $ntx $nty $nbix
                           $nbiy $vizinhos($nt)]
            set rxcost [expr ($numberOfNb) * $val(rxPower)]
            if {($cost($nbi) == -1) || ($cost($nbi) > [expr $cost($nt) +
                                                  $txcost+$rxcost])} {
                set cost($nbi) [expr $cost($nt)+$txcost+$rxcost]
                lappend lvisited $nbi
                set fwrdn($nbi) $nt
            }
```

```
        }
    incr i
    }

    # Set the optimized neighbours set
    set Nbrs {}
    for {set i 0} {$i < [llength $A_set]} {incr i} {
        set ni [lindex $A_set $i]
        if {$fwrdn($ni) == $myindex} {
            lappend Nbrs $ni
        }
    }

}

##########################################################################
#
# Procedure to calculate if a node is in another node's relay region.
# It returns 1 if the node 'node' is in the relay region of the node 'relay'.
# Otherwise, it returns 0.
#
##########################################################################
Agent/MessagePassing/OMCT instproc inRelayRegion {node relay} {
   $self instvar XY_set A_XY_list node_ myindex myx myy A_set nbs nbsxy_set
   global PATHLOSS_EXPONENT val

    if { [lsearch $nbs($relay) $node] == -1 } {
        return 0
    }

    set nodex [lindex $XY_set($node) 0]
    set nodey [lindex $XY_set($node) 1]

    set relayx [lindex $XY_set($relay) 0]
    set relayy [lindex $XY_set($relay) 1]

    # cost myself -> node:
    set numberOfNbMyselfToNode [$self neighboursInRange
                                $myx $myy $nodex $nodey $A_XY_list]
    set costMyselfNode [cost [distance $myx $myy $nodex $nodey]
                                $numberOfNbMyselfToNode]


    # cost myself -> relay:
    set numberOfNbMyselfRelay [$self neighboursInRange $myx $myy $relayx
                                $relayy $A_XY_list]
    set costMyselfRelay [cost [distance $myx $myy $relayx $relayy]
                                $numberOfNbMyselfRelay]


    # cost relay -> node:
    set numberOfNbRelayToNode [$self neighboursInRange $relayx $relayy
                                $nodex $nodey $nbsxy_set($relay)]
    set costRelayNode [cost [distance $relayx $relayy $nodex $nodey]
                                $numberOfNbRelayToNode]

    # compare costs:
    if {$costMyselfNode <= [expr $costMyselfRelay + $costRelayNode]} {
        return 0
    } else {
        return 1
    }
}
```

```
#############################################################################
#
# Procedure that triggers the algorithm
# At the end of this procedure, each nodes has a set with its
# optimized neighbours
#
#############################################################################
Agent/MessagePassing/OMCT instproc findOptimizedNeighbourhood {} {
    global ns ROUND

    set inst1 $ROUND
    set inst2 [expr $ROUND * 2]
    set inst3 [expr $ROUND * 3]

    $ns at $inst1 "$self sendPosition"              ;   # Start Phase 1
    $ns at $inst2 "$self sendReachableNeighbs"      ;   # Start Phase 2
    $ns at $inst3 "$self calculateOptimizedNeighbs" ;   # Start Phase 3
}

#############################################################################
#
#   Defining Topology
#   Creating nodes
#
#############################################################################
for {set i 0} {$i < $num_nodes} {incr i} {
    set n($i) [$ns node]
}

#############################################################################
#
#   Attaching agents to nodes
#   Attaching a new Agent/MessagePassing/OMCT to each node on port
#   $OMCT_PORT
#
#############################################################################
for {set i 0} {$i < $num_nodes} {incr i} {
    set omct($i) [new Agent/MessagePassing/OMCT]
    $n($i) attach  $omct($i) $OMCT_PORT
    $omct($i) initiateVars
}

#############################################################################
#
#   Setting up starting events
#
#############################################################################
for {set i 0} {$i < $num_nodes} {incr i} {
    $ns at 0 "$omct($i) findOptimizedNeighbourhood"
}

$ns at 2000 "finish"

#############################################################################
#
# Global procedures
#
#############################################################################

#############################################################################
#
# Procedure to calculate distance between two nodes
```

```
#
#############################################################################
proc distance {x1 y1 x2 y2} {
    return [expr sqrt(pow($x1 - $x2,2) + pow($y1 - $y2,2))]
}

#############################################################################
#
# Procedure to calculate edge cost
#
#############################################################################
proc cost {distance numVizinhos} {
    global val PATHLOSS_EXPONENT
    return [expr $val(txPowerFixo) +
            ( pow($distance ,$PATHLOSS_EXPONENT)
                * $val(txPowerCoeficienteE) ) +
            ($numVizinhos * $val(rxPower)) ]
}

#############################################################################
#
# Procedure to plot the optimized topology graph
#
#############################################################################
proc plot-optimized-neighb-graph {} {
    global num_nodes n omct val logFile PATHLOSS_EXPONENT

    set ng [open otimizedGraph.xg w]

    for {set i 0} {$i < $num_nodes} {incr i} {
        for {set j 0} {$j < [llength [$omct($i) set Nbrs]]} {incr j} {
            if {$j == 0} {
                puts $ng "\n[$n($i) set X_] [$n($i) set Y_]"
                puts $ng "[$n([lindex [$omct($i) set Nbrs] $j]) set X_]
                          [$n([lindex [$omct($i) set Nbrs] $j]) set Y_]"
            } else {
                puts $ng "[$n($i) set X_] [$n($i) set Y_]"
                puts $ng "[$n([lindex [$omct($i) set Nbrs] $j]) set X_]
                          [$n([lindex [$omct($i) set Nbrs] $j]) set Y_]"
            }
        }
    }
    close $ng
}

#############################################################################
#
# Procedure to plot the original topology graph
#
#############################################################################
proc plot-maxpower-graph {} {
    global num_nodes n omct val logFile PATHLOSS_EXPONENT

    set mpg [open originalGraph.xg w]

    for {set i 0} {$i < $num_nodes} {incr i} {
        for {set j 0} {$j < [llength [$omct($i) set A_set]]} {incr j} {
            if {$j == 0} {
                puts $mpg "\n[$n($i) set X_] [$n($i) set Y_]"
                puts $mpg "[$n([lindex [$omct($i) set A_set] $j]) set X_]
                           [$n([lindex [$omct($i) set A_set] $j]) set Y_]"
            } else {
                puts $mpg "[$n($i) set X_] [$n($i) set Y_]"
```

```
                puts $mpg "[$n([lindex [$omct($i) set A_set] $j]) set X_]
                           [$n([lindex [$omct($i) set A_set] $j]) set Y_]"
            }
        }
    }
    close $mpg
}

#############################################################################
#
#   Finishing the algorithm
#
#############################################################################
proc finish {} {
    global ns f nf
    close $f
    plot-optimized-neighb-graph
    plot-maxpower-graph
    exit 0
}

#############################################################################
#
#   Running ns
#
#############################################################################
$ns run

#############################################################################
#
#   End of the G-KRed-OH Algorithm
#
#############################################################################
```

APÊNDICE G

CONFIGURAÇÃO DOS TRANSCEPTORES PARA O NS-2

G.1 CONFIGURAÇÃO DO DISPOSITIVO ATMEL RF230

```
#############################################################
# Configuration for Atmel RF230 device

set val(radioModel)             RF230
set val(rxPower)                186.0           ;%nJ/bit
set val(txPowerFixo)            48.0            ;%nJ/bit
set val(txPowerCoeficienteE)    0.0016666667    ;%nJ/bit/m2

# End of the configuration for Atmel RF230 device
#############################################################
```

G.2 CONFIGURAÇÃO DO DISPOSITIVO CHIPCON CC2420

```
#############################################################
# Configuration for Chipcon CC2420 device

set val(radioModel)             CC2420
set val(rxPower)                236.4           ;%nJ/bit
set val(txPowerFixo)            48.0            ;%nJ/bit
set val(txPowerCoeficienteE)    0.0160800000    ;%nJ/bit/m2

# End of the configuration for Chipcon CC2420 device
#############################################################
```

G.3 CONFIGURAÇÃO DO DISPOSITIVO CHIPCON CC1000

```
#############################################################
# Configuration for Chipcon CC1000 device

set val(radioModel)             CC1000
set val(rxPower)                292.1           ;%nJ/bit
set val(txPowerFixo)            157.9           ;%nJ/bit
set val(txPowerCoeficienteE)    0.0387854917    ;%nJ/bit/m2

# End of the configuration for Chipcon CC1000 device
#############################################################
```

REFERÊNCIAS BIBLIOGRÁFICAS

[1] F. Assis and U. Telemaco, "A Topology Control Algorithm for Wireless Sensor Networks that considers Overhearing," in *Proceedings of the IEEE 9th International Symposium on Autonomous Decentralized Systems (ISADS)*, 2009.

[2] A. Rowe, R. Mangharam, and R. Rajkumar, "FireFly: A Time Synchronized Real-Time Sensor Networking Platform," *Wireless Ad Hoc Networking: Personal-Area, Local-Area, and the Sensory-Area Networks*.

[3] W. Jeong and S. Y. Nof, "Performance evaluation of wireless sensor network protocols for industrial applications," *Journal of Intelligent Manufacturing*, vol. 19(3), pp. 335–345, 2008.

[4] L. Schwiebert, S. K. S. Gupta, and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," in *Proceeding of MOBICOM*, 2001, pp. 151–165, http://doi.acm.org/10.1145/381677.381692.

[5] V. Mehta and M. E. Zarki, "A bluetooth based sensor network for civil infrastructure health monitoring," *Wireless Networks*, vol. 10, no. 4, 2004.

[6] Y. Ma, M. Richards, M. Ghanem, Y. Guo, and J. Hassard, "Air pollution monitoring and mining based on sensor grid in london," *Molecular Diversity Preservation International*, 2008.

[7] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA-02)*. New York: ACM Press, 2002, pp. 88–97.

[8] E. S. Biagioni and K. W. Bridges, "The application of remote sensor technology to assist the recovery of rare and endangered species," *The International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 315–324, 2002.

[9] I. F. Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[10] G. Pottie and W. Kaiser, "Wireless Integrated Network Sensors," *Communication of ACM*, pp. 43(5):51–8.

[11] M. M. A. Howard and G. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in

Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02), 2002.

[12] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J.-C. Chen, "A survey of energy efficient network protocols for wireless networks," *Wireless Networks*, vol. 7, pp. 343–358, 2001.

[13] D. D. Wentzloff, B. H. Calhoun, R. Min, A. Wang, N. Ickes, and A. Chandrakasan, "Design considerations for next generation wireless power-aware microsensor nodes," in *VLSI Design*, 2004, pp. 361–.

[14] P. Santi, "Topology Control in Wireless Ad Hoc and Sensor Networks," *ACM Computing Surveys*, vol. 37, no. 2, pp. 164–194, 2005.

[15] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[16] V. Shnayder, M. Hempstead, B. rong Chen, G. W. Allen, and M. Welsh, "Simulating the power consumption of large-scale sensor network applications," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys 2004*, J. A. Stankovic, A. Arora, and R. Govindan, Eds. Baltimore, MD, USA: ACM, 2004, pp. 188–200, http://doi.acm.org/10.1145/1031495.1031518.

[17] Texas Instruments, "Chipcon CC1000 Data Sheet," 2009, http://focus.ti.com.

[18] ——, "Chipcon CC2420 Data Sheet," 2009, http://focus.ti.com.

[19] Atmel Corporation, "Atmel RF230 Data Sheet," 2009, http://www.atmel.com.

[20] S. Hong, Y.-J. Choi, and S.-J. Kim, "An Energy Efficient Topology Control Protocol in Wireless Sensor Networks," in *Proceedings of the International Conference on Advanced Communication Technology - ICACT2007*, 2007.

[21] A. Rahman and P. Gburzinski, "On Constructing Minimum-Energy Path-Preserving Graphs for Ad-Hoc Wireless Networks," in *Proceedings of the IEEE International Conference on Communications - ICC*, South Korea, 2005.

[22] L. Li and J. Y. Halpern, "A Minimum-Energy Path-Preserving Topology-Control Algorithm," *IEEE Transactions on Wireless Communications*, vol. 3, no. 3, pp. 910–921, 2004.

[23] B. H. Liu, Y. Gao, C. T. Chou, and S. Jha, "An Energy Efficient Select Optimal Neighbor Protocol for Wireless Ad Hoc Networks," Network Research Laboratory, School of Computer Science and Engineering, The University of New South Wales, Sydney, Australia, Tech. Rep. UNSW-CSE-TR-0431, 2004.

[24] X.Cheng, B.Narahari, R.Simha, M.X.Cheng, and D.Liu, "Strong Minimum Energy Topology in Wireless Sensor Networks: NP-Completeness and Heuristics," *IEEE Transactions on Mobile Computing*, vol. 2, no. 3, pp. 248–256, 2003.

[25] X.-Y.Li and P.-J.Wan, "Constructing Minimum Energy Mobile Wireless Networks," in *Proceedings of the MobiHOC 2001*, California, USA, 2001, pp. 283–286.

[26] V. Rodoplu and T. H. Meng, "Minimum Energy Mobile Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, 1999.

[27] P. Basu and J. Redi, "Effect of Overhearing Transmissions on Energy Efficiency in Dense Sensor Networks," in *Proceedings of IPSN'04*, Berkeley, California, USA, 2004.

[28] S. Gobriel, R. Melhem, and D. Mossé, "A Unified Interference/Collision Model for Optimal MAC Transmission Power in Adhoc Networks," *International Journal of Wireless and Mobile Computing*, vol. 1, no. 3/4, pp. 179–190, 2006.

[29] C. K. Toh, *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, P. Hall, Ed. Prentice Hall, 2003.

[30] S. Kai and H. Chiu, "Survey of mobile wireless ad-hoc networks," 2005.

[31] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," in *Proceeding of IEEE Personal Communications*, vol. 6, no. 2, 1999, pp. 46–55.

[32] University of Kassel, "The Distributed System research group website," 2009, http://www.vs.uni-kassel.de/systems/index.php/Sensor_network.

[33] Atmel Corporation, "Atmel ATmega 1281 Data Sheet," 2009, http://www.atmel.com.

[34] ——, "Atmel ATmega 128L Data Sheet," 2009, http://www.atmel.com.

[35] A. Corporation, "Atmel ATmega 32L Data Sheet," 2009, http://www.atmel.com.

[36] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Eficient Communication Protocol for Wireless Microsensor Networks," in *33rd Hawaii International Conference on System Sciences*, 2000, pp. 3005–3014.

[37] S. Lindsey and C. S. Raghavendra, "PEGASIS: Power-efficient gathering in sensor information systems," in *Proceedings of the IEEE International Conference on Communications - ICC 2001*, 2001.

[38] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, 2003.

[39] Y. Yu, R. Govindan, and D. Estrin, "Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks," UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, Tech. Rep., 2001.

[40] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, "Negotiation-based protocols for disseminating information in wireless sensor networks," *Wireless Networks*, vol. 8, no. 2-3, pp. 169–185, 2002.

[41] D. Braginsky and D. Estrin, "Rumor routing algorithm for sensor networks," in *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA-02)*. New York: ACM Press, 2002, pp. 22–31.

[42] ETH Zurich - Swiss Federal Institute of Technology Zurich, "The sensor network museum," 2009, http://www.snm.ethz.ch/Main/HomePage.

[43] T. Bokareva, "Mini hardware survey," 2009, http://www.cse.unsw.edu.au/~sensar/hardware/hardware_survey.html.

[44] G. Yang, "BSN Nodes," 2009, http://ubimon.doc.ic.ac.uk/bsn/index.php?m=206.

[45] Crossbow, "Product reference guide," http://www.xbow.com.

[46] IEEE Standard Association, "IEEE 802.15.4 - Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)," 2006, http://grouper.ieee.org/groups/802/15/.

[47] Crossbow, "Mote Kit Data Sheet," 2007, http://www.xbow.com.

[48] Carnegie Mellon University - Dept. of Electrical and Computer Engineering, "FireFly Plataform - Real-Time Wireless Sensor Network Platform," http://www.ece.cmu.edu/firefly.

[49] R. Mangharam, A. Rowe, and R. Rajkumar, "FireFly: a cross-layer platform for real-time embedded wireless networks," *Real-Time Systems*, vol. 37, no. 3, pp. 183–231, 2007, http://dx.doi.org/10.1007/s11241-007-9028-z.

[50] P. Davis, P. Smith, E. Campbell, J. Lin, K. Gross, G. Bath, Y. Low, M. Lau, Y. Degani, J. Gregus, R. Frye, and K. Tai, "Silicon-on-silicon integration of a GSM transceiver with VCO resonator," in *IEEE Institute Solid-State Circuits Conference*, 1998, pp. 248–249.

[51] R. R. Regina and R. Rosales-hain, "Topology control of multihop wireless networks using transmit power adjustment," 2000.

[52] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[53] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

[54] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the Association for Computing Machinery*, vol. 5, p. 345, 1962.

[55] T. S. Rappaport, *Wireless Communications Principles and Practice*. Upper Saddle River: Prentice Hall, 1996.

[56] P. Santi, "The Critical Transmitting Range for Connectivity in Mobile Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 4, pp. 310–317, 2005.

[57] M. Sánchez, P. Manzoni, and Z. J. Haas, "Determination of critical transmission range in ad-hoc networks," in *Proceeding of Multiaccess, mobility and teletraffic in wireless communications*, 2000.

[58] L. Hu, "Topology control for multihop packet radio networks," *IEEE Transactions on Communications*, vol. 41, pp. 1474–1481, 1993.

[59] J. P. Monks, V. Bharghavan, and W. mei W. Hwu, "A power controlled multiple access protocol for wireless packet networks," in *Infocom*, 2001, pp. 219–228.

[60] IEEE Standard Association, "IEEE 802.11 wireless local area networks," 2005, http://grouper.ieee.org/groups/802/11/.

[61] A. Rahman and P. Gburzynski, "MAC-assisted topology control for ad hoc wireless networks," *Int. J. Communication Systems*, vol. 19, no. 9, pp. 955–976, 2006, http://dx.doi.org/10.1002/dac.776.

[62] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," in *ACM Wireless Networks Journal*, 2001, pp. 85–96.

[63] Y. Xu, J. S. Heidemann, and D. Estrin, "Geography-informed energy conservation for ad hoc routin," in *MOBICOM*, 2001, pp. 70–84, http://doi.acm.org/10.1145/381677.381685.

[64] J. Faruque and A. Helmy, "Gradient-based routing in sensor networks," *Mobile Computing and Communications Review*, vol. 7, no. 4, pp. 50–52, 2003.

[65] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," *The International Journal of High Performance Computing Applications*, vol. 16, no. 3, pp. 293–313, 2002.

[66] L. Lin, N. B. Shroff, and R. Srikant, "Energy-Aware Routing in Sensor Networks: a Large System Approach," *Ad Hoc Networks*, vol. 5, pp. 818–831, 2007.

[67] R. C. Shah and J. M. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Orlando, FL, 2002.

[68] M. Handy, M. Haase, and D. Timmermann, "Low energy adaptive clustering hierarchy with deterministic cluster-head selection," in *Proceeding of the IEEE Mobile and Wireless Communications Networks - MWCN*. IEEE, 2002, pp. 368–372.

[69] A. Manjeshwar and D. P. Agrawal, "TEEN: ARouting protocol for enhanced efficiency in wireless sensor networks," in *IPDPS*. IEEE Computer Society, 2001, p. 189.

[70] A. Manjeshwar and D. Agrawal, "APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks," in *Proceedings of the 16th International Parallel & Distributing Processing Symposium - IPDPS-02*, B. Werner, Ed., 2002, pp. 195–202.

[71] E. H. B. Maia, D. Câmara, and A. A. F. Loureiro, "Ica: Um novo algoritmo de roteamento para redes de sensores," *XXII Simpósio Brasileiro de Redes de Computadores*, 2004.

[72] C. Savarese and J. M. Rabaey, "Locationing in distributed ad-hoc wireless sensor networks," in *in ICASSP*, 2001, pp. 2037–2040.

[73] Information Sciences Institute of the University of Southern California, "The Network Simulator: NS-2: notes and documentation," 2009, http://nsnam.isi.edu/nsnam/.