ALGORITMOS PARA RASTREAMENTO DE ALVOS EM ÁREAS QUANTIZADAS COM REDES DE SENSORES SEM FIO

ÉFREN LOPES DE SOUZA

ALGORITMOS PARA RASTREAMENTO DE ALVOS EM ÁREAS QUANTIZADAS COM REDES DE SENSORES SEM FIO

Tese apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Doutor em Informática.

Orientador: Eduardo Freire Nakamura

Manaus

Março de 2014

Lopes de Souza, Éfren

D1234p Algoritmos para Rastreamento de Alvos em Áreas Quantizadas com Redes de Sensores Sem Fio / Éfren Lopes de Souza. — Manaus, 2014
xix, 185 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal do Amazonas
Orientador: Eduardo Freire Nakamura

1. Rastreamento de Alvos. 2. Algoritmos Distribuídos. 3. Agrupamento. 4. Previsão. 5. Estrutura de Faces. 6. Algoritmos de Localização. I. Título.

CDU 519.6*82.10

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

Se a imagem da folha de aprovação precisar ser ajustada, use:
`approval=[`*ajuste*`][`*escala*`]{`*nome do arquivo*`}`
onde *ajuste* é uma distância para deslocar a imagem para baixo
e *escala* é um fator de escala para a imagem. Por exemplo:
`approval=[-2cm][0.9]{`*nome do arquivo*`}`
desloca a imagem 2cm para cima e a escala em 90%.

Agradecimentos

Agradeço aos meus pais, irmãos, amigos e professores.

Este trabalho contou com o apoio da Superintendência da Zona Franca de Manaus (SUFRAMA) e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES).

Resumo

Rastreamento de alvos em Redes de Sensores Sem Fio (RSSFs) é um tipo de aplicação em que os nós cooperam para estimar a posição de um ou mais objetos de interesse. Nesse contexto, este trabalho possui quatro contribuições. A primeira contribuição é um levantamento bibliográfico do estado-da-arte, em que identificamos três diferentes formulações de rastreamento e as classificamos de acordo com suas características. Além disso, dividimos o processo de rastreamento em componentes para facilitar o entendimento geral. A segunda contribuição é a elaboração e avaliação do algoritmo PRATIQUE para rastrear animais em florestas. Nesse caso, os nós são organizados em grade para viabilizar a utilização dos nós sensores nesse tipo de área, de forma que cada célula da grade é uma região que pode ser ocupada pelo alvo. O algoritmo estima a célula em que o alvo está, e usa previsão e um esquema híbrido de agrupamento para reduzir o custo de comunicação e garantir a precisão do rastreamento. Os resultados das simulações mostram que os erros de previsão são de aproximadamente uma célula. A terceira contribuição é o algoritmo TATI, esse algoritmo guia um objeto que visa alcançar o alvo. A rede é estruturada em faces para facilitar a cooperação entre os nós e reduzir o caminho entre o objeto guiado e o alvo. Os resultados mostram que o consumo de energia é reduzido em 15% e o objeto guiado fica cerca de 10 m mais próximo do alvo, se comparado com a abordagem relacionada. A quarta contribuição é um esquema para executar as tarefas de localização e rastreamento simultaneamente para reduzir os erros dos algoritmos de localização baseados em alcance. As mensagens enviadas para rastrear o alvo são aproveitadas para filtrar os ruídos presentes nas estimativas de distância, reduzindo o erro de localização enquanto o rastreamento ocorre. Os resultados mostram que os erros de localização podem ser reduzidos em até 70%.

Palavras-chave: Rastreamento de Alvos, Algoritmos Distribuídos, Agrupamento, Previsão, Estrutura de Faces, Algoritmos de Localização.

Abstract

Target tracking in Wireless Sensor Networks (WSNs) is an application in which the nodes cooperate to estimate the position of one or more objects of interest. In this context, the contributions of this work are fourfold. First, a survey the state-of-the-art about target tracking algorithms, in which we identified three formulations of tracking problem, and we classified them according to their characteristics. Furthermore, we divided the target tracking process in components to make the general understanding easier. Second, we propose and evaluate the PRATIQUE algorithm for tracking animals in forests. In this case, the nodes are organized into a grid to make feasible the use of sensor nodes in this kind of area in such a way that each cell of the grid is a region that can be occupied by the target. The algorithm estimates the cell where the target is, and uses predictions and hybrid clustering to reduce the communication cost and ensure the tracking accuracy. The results of the simulations show that prediction errors are approximately one cell. The third contribution is the TATI algorithm, this algorithm guides a tracker to approach the target. The sensor network is organized into faces to make the cooperation among the nodes easier, and reduce the path between the tracker and the target. The results show that energy consumption is reduced by 15%, and the tracker stays about 10 m closer to the target, compared to the baseline. The fourth contribution is a scheme for performing localization and tracking tasks simultaneously in such a way that errors of range-based localization algorithms are reduced. This algorithm takes advantage of the messages sent to track the target to filter the noise in the distance estimation, reducing localization errors while tracking. The results show that the localization errors can be reduced by up to 70%.

Keywords: Target Tracking, Distributed Algorithms, Clustering, Prediction, Face Structure, Localization Algorithms.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Siglas

ABF *Alpha-Beta Filter* . .3, 9–11, 14, 27, 30–32, 47, 88, 97, 103, 110, 113, 142, 152, 153, 159, 160

ADCT *Adaptive Dynamic Cluster-based Tracking* 46, 48, 49

ASIR *Auxiliary Sampling Importance Resampling* 34

CB *Cluster Border* 88–91, 93

CH *Cluster Head* . 23–25, 37, 40, 49, 50, 52, 56–58, 88–95, 97–99, 103, 107, 161, 162

CM *Cluster Member* 23, 49, 88, 91

CODA *Continuos Object Detection and tracking Algorithm* 46, 48, 50, 52, 53, 55, 56

CRW *Correlated Random Walk* 64, 68, 72, 99, 129, 146

DAB *Drain-And-Balance* 45, 47–51, 53–55

DAT *Deviation Avoidance Tree* 45, 47–51

DCTC *Dynamic Convoy Tree-Based Collaboration* 46–48, 52

DELTA *Distributed Event Localization and TrAcking* 5, 46, 48, 49

DKF *Distributed Kalman Filter* 29

DMMT *Distributed Multi-sensor Multi-target Tracking* 46, 48, 49, 51

DOT *Dynamic Object Tracking* . 10, 46, 48, 50, 51, 113, 125, 128, 130–133, 135, 161

DPE *Directed Position Estimation*11, 42, 139, 145, 147, 149, 151, 152, 155, 157, 159, 161

EKF *Extended Kalman Filter* 28–30, 34

FAR *Face-Aware Routing* 59

FOTP *Face-based Object Tracking Protocol* 47, 48, 50

GAF *Geography-informed energy conservation for Ad Hoc routing* 44

GG *Gabriel Graph* 25, 50, 116, 118

GPS *Global Positioning System* 41, 136, 137, 139

HCMTT *Hybrid Clustering for Multi-Target Tracking* 46, 48, 50, 51

HCTT *Hybrid Cluster-based Target Tracking* 46, 48, 50

KF *Kalman Filter*3, 8–11, 14, 27–31, 38, 47, 88, 97, 103, 104, 110, 142, 144, 152–154, 159, 160

LEACH *Low Energy Adaptive Clustering Hierarchy* 24

LW *Lévy Walk* 64, 65, 67, 68

MANET *Mobile Ad hoc NETwork* 61, 63, 67

OCO *Optimized Communication and Organization* 3, 46–48, 51

OGDC *Optimal Geographical Density Control* 44

PDF *Probability Density Function* 32

PEGASIS *Power-Efficient GAthering in Sensor Information System* 24

PES *Prediction-based Energy Saving* 46, 48, 49, 51

PET *Prediction-based Energy-efficient Target tracking protocol* . . 5, 46, 48, 50–53, 58–60, 113

PF *Particle Filter* . 3, 8–11, 14, 27, 33, 47, 88, 97, 103, 110, 142, 152–154, 159–161

POOT *Prediction-based Optimistic Object Tracking Scheme* . 5, 46, 48, 50, 51, 113

PRATIQUE *Prediction-based clusteRing Algorithm for TrackIng targets in QUantized arEas for wireless sensor networks* 9, 12, 86–88, 97, 110, 160

PRECO *PREdictive Continuous Object tracking* 46, 48, 49, 52

RARE *Reduced Area REporting* 46, 48, 49

RARE-Area *Reduced Area REporting* 4

RARE-Node *Reduction of Active node REdundancy* 4

RD *Random Direction* 62–64

RFID *Radio Frequency Identification* 19

RNG *Relative Neighborhood Graph* 25, 50, 116, 118

RPE *Recursive Position Estimation* . 11, 42, 139, 145, 147, 148, 151, 152, 154, 155, 157, 159, 161

RPF *Regularized Particle Filter* 34

RPGM *Reference Point Group Mobility* 66

RSSF *Rede de Sensores Sem Fio* 1

RSSI *Received Signal Strength Indicator* 137, 139, 145

RW *Random Walk* 62–64, 67, 68

RWP *Random WayPoint* 62, 63, 65

RWPB *Random Waypoint on the Border* 62

SB *Static-cluster Boundary-sensor* 56–58

SCH *Super Cluster Head* 95, 97, 98

SI *Static-cluster Inner-sensor* 56–58

SIR *Sampling Importance Resampling* 34

SIS *Sequential Importance Sampling* 34

SMC *Sequential Monte Carlo* 35

SR *Smooth Random* 64, 65

STUN *Scalable Tracking Using Networked Sensors* 3, 53

TATI *Tracking Algorithm for Target Interception in face-structured sensor networks* 10, 12, 113, 117, 124, 125, 128, 130–133, 135, 161

TG-COD *Two-tier Grid based Continuous Object Detection and tracking* 46, 48, 50, 52

Sumário

Capítulo 1

Introdução

Uma Rede de Sensores Sem Fio (RSSF) [Akyildiz et al., 2002] é um tipo especial de rede *ad-hoc* composta por dispositivos com recursos limitados, chamados nós sensores. Esses sensores são capazes de monitorar um ambiente, coletar dados, realizar processamento localmente e disseminar dados. O maior objetivo dessas redes é monitorar o ambiente para detectar e avaliar eventos de interesse. Dentre as aplicações das redes de sensores, rastreamento de alvos é uma importante aplicação [Tsai et al., 2007; Wang et al., 2010b; Sharma et al., 2011a; Hsu et al., 2012; Campos et al., 2012]. Em poucas palavras, rastreamento consiste em detectar um alvo (como animais, pessoas e veículos) para calcular sua posição atual e, em alguns casos, estimar sua posição futura [Nakamura et al., 2007].

Rastreamento de alvos é uma aplicação não trivial de redes de sensores sem fio [Li & Zhou, 2010]. O propósito das aplicações de rastreamento é coletar dados sobre um ou mais alvos (objetos móveis de interesse) e então estimar suas posições no campo de sensores. Em algumas situações, prever a próxima posição do alvo também é interessante [Sharma et al., 2011b; Kumar & Sivalingam, 2012]. A previsão pode ser usada pelo usuário da aplicação, por exemplo, para interceptar um animal que se dirige para uma área de risco, como estradas em uma reserva ecológica. A própria aplicação também pode fazer uso da previsão para ativar somente os sensores próximos do alvo, enquanto os nós mais distantes ficam desativados para economizar energia. Dessa forma o gasto de energia é minimizado, aumentando o tempo de vida da rede [Xu et al., 2004a; Bhuiyan et al., 2010; Deldar & Yaghmaee, 2011].

Rastreamento de alvos são aplicados em diferentes áreas, como vigilância, monitoramento ambiental, monitoramento de tráfego e detecção de intrusos [Semertzidis et al., 2010; Komagal et al., 2012; Kozma et al., 2012]. Os algoritmos de rastreamento usualmente buscam manter a precisão do rastreamento e aumentar o tempo de vida da rede

[Hajiaghajani et al., 2012; Hsu et al., 2012]. Devido às interferências nas leituras dos sensores e às restrições de energia e comunicação [Nakamura et al., 2007] é desafiador desenvolver uma técnica eficiente e precisa.

Neste trabalho tratamos sobre rastreamento de alvos em redes de sensores. Inicialmente discutimos e classificamos os trabalhos existentes na literatura. Depois propomos e avaliamos abordagens de rastreamento para dois tipos de formulação do problema de rastreamento. Além disso, integramos localização e rastreamento visando melhorar o desempenho de ambos.

1.1 Motivação

Para auxiliar na pesquisa ecológica, rastreamento de alvos com redes de sensores pode ser usado para estudar o comportamento animal, relacionando sua mobilidade com seus hábitos, como alimentação e descanso. Este trabalho foi elaborado para o rastreamento do sauim-de-coleira que está incluído nas listas nacionais e internacionais de espécies ameaçadas como Criticamente em Perigo [Machado et al., 2005], por isso a rede de sensores deve ser implantada para funcionar dentro da floresta densa. Nesse caso, conhecer a região em que o alvo se encontra é o suficiente, a posição exata é dispensável.

A dificuldade de se implantar essas redes em florestas é devido aos problemas de comunicação e sensoriamento, que faz com que os nós coletem dados pouco precisos sobre o evento ou até mesmo os perca. Por outro lado, as reservas ecológicas de florestas tropicais comumente possuem vias de acesso usadas por pesquisadores para estudar a fauna e a flora da região, mas que também podem ser aproveitadas para viabilizar a comunicação entre os nós dentro da floresta densa [Figueiredo et al., 2009].

Essas vias de acesso são, geralmente, feitas em forma de grade, favorecendo que a topologia da rede seja organizada da mesma forma. A organização em grade pode ser usada para discretizar o campo de sensores para usar as células como regiões de ocupação. Apesar da maioria dos algoritmos de rastreamento buscar calcular a posição exata do alvo, uma região deve ser considerada devido aos erros de localização. Quando os nós são distribuídos aleatoriamente pelo campo de sensores, um algoritmo de localização deve ser executado antes do rastreamento, fornecendo a cada nó a sua localização [Boukerche et al., 2007; Oliveira et al., 2009]. A localização gerada por esses algoritmos possuem erros que são refletidos no rastreamento. Em Souza et al. [2013] mostramos que os filtros usados para o rastreamento não são capazes de filtrar os erros de localização, sendo necessário reduzir os erros das estimativas de distância para melhorar o resultado do rastreamento. Dessa forma, os algoritmos de rastreamento despendem um

grande esforço para calcular a posição exata do alvo, mas uma área ainda precisa ser considerada, por isso discretizar o campo de sensores é uma alternativa mais simples e com resultados similares.

Além disso, um fator importante a ser considerado é a coleta dos sensores no fim da vida da rede. Pois esses são compostos por baterias e outros materiais prejudiciais ao meio ambiente, demandando uma coleta posterior de dispositivos inoperantes. A distribuição estratégica dos nós facilita essa tarefa.

Em algumas situações é interessante interceptar alguns animais que precisam de cuidados ou evitar que eles entrem em alguma área de risco. Nesse caso, a rede pode auxiliar nessa tarefa, rastreando o alvo e guiando o objeto interessado em interceptar o alvo.

1.2 Contexto

Na literatura existem vários trabalhos que atacam o problema de rastreamento de alvos usando diferentes abordagens em suas soluções. Esses trabalhos focam principalmente na forma como os nós cooperam para coletar e disseminar os dados sobre o alvo. Para isso, a estrutura lógica da rede pode ser organizada em forma árvore [Lin et al., 2006; Yen et al., 2010], agrupamentos [Xu & Lee, 2007; Mansouri et al., 2011] ou faces [Huang et al., 2005; Tsai et al., 2007; Ji et al., 2009; Bhuiyan et al., 2010; Hsu et al., 2012]. Independente da estrutura utilizada, esses algoritmos podem ser combinados com previsão para reduzir o consumo de energia da rede [Naderan et al., 2009]. Filtros Bayesianos, como *Kalman Filter* (KF), *Particle Filter* (PF) e *Alpha-Beta Filter* (ABF) são frequentemente utilizados para realizar previsões e reduzir os ruídos gerados durante as medições [Nakamura et al., 2007; Xingbo et al., 2011; Li & Wang, 2012; Sharma et al., 2011a].

Nos algoritmos de rastreamento baseados em árvore, os nós sensores são organizados em uma estrutura de árvore ou representados como um grafo, em que os vértices representam os nós sensores e as arestas representam as ligações entre os nós que podem se comunicar diretamente. Essa estrutura facilita a fusão de dados e reduz o custo de comunicação e o consumo de energia [Yen et al., 2010]. O *Optimized Communication and Organization* (OCO) é apresentado em Tran & Yang [2006], esse algoritmo proporciona auto-organização e roteamento para rastreamento de alvos em redes de sensores. Ele coleta a posição de todos os nós da rede para criar um mapa geográfico da rede e remover os nós redundantes. [Kung & Vlah, 2003] propõem o *Scalable Tracking Using Networked Sensors* (STUN), é um algoritmo que aplica árvores hierárquicas

para conectar os sensores, de forma que o ponto de consulta é a raiz, os sensores são as folhas e o restante dos nós são chamados de nós intermediários. A rede é considerada como um banco de dados distribuído, já que os nós registram dados sobre o alvo para reduzir o custo das consultas. Um evento é gerado sempre que um alvo entra ou sai do raio de detecção de um sensor. A informação sobre a presença do alvo é enviada pelos sensores ao nó *sink*, sendo que os nós intermediários também armazenam essa informação juntamente com seus descendentes. Dessa forma, os nós folhas armazenam a informação da presença somente dos alvos que eles detectaram, enquanto a raiz contém a informação de presença de todos os alvos.

Nos algoritmos baseados em agrupamento os nós são divididos em grupos, cada grupo tem seu líder e os membros. O líder é responsável por receber os dados gerenciar as operações do seu agrupamento [Alaybeyoglu et al., 2009]. Os algoritmos de rastreamento baseados em agrupamentos podem ser divididos em dois grupos: estáticos [Olule et al., 2007; Xu & Lee, 2007] e dinâmicos [Chen et al., 2004; Yang et al., 2007; Suganya, 2008]. No agrupamento estático, a estrutura do agrupamento, ou seja, quais dos nós são membros ou líderes, é definida assim que a rede começa a operar. Esse método é simples e reduz o consumo de energia. Entretanto pode trazer alguns problemas quando os nós falham ou até mesmo gerar redundâncias quando o alvo é detectado por nós de diferentes agrupamentos. Já no agrupamento dinâmico, a estrutura do agrupamento é formada de acordo com a detecção do alvo. Nesse caso, o nó líder e os membros são definidos sempre que um novo agrupamento é formado. Isso facilita no processo de tolerância a falhas, uma vez que os nós inoperantes ficam fora do agrupamento. Entretanto exige um custo de comunicação extra sempre que um novo agrupamento dinâmico é formado. Chen et al. [2004] propõem um algoritmo descentralizado baseado em agrupamento dinâmico para rastreamento de alvos. Os agrupamentos são formandos usando diagrama de Voronoi Koucheryavy & Salim [2009] e somente o líder do agrupamento é ativado quando a potência do sinal acústico, emitida pelo alvo, excede um limiar predeterminado. O líder convida os nós da vizinhança a se juntarem ao agrupamento. Baseado na distância estimada entre os nós do agrupamento e o alvo, o líder calcula a posição do alvo e envia o resultado para o *sink*. Olule et al. [2007] propõem dois algoritmos para melhorar a eficiência energética em aplicações de rastreamento usando agrupamentos estáticos: o algoritmo *Reduced Area REporting* (RARE-Area) e o algoritmo *Reduction of Active node REdundancy* (RARE-Node). O primeiro algoritmo reduz o número de nós que participam do rastreamento inibindo a ação de alguns nós de acordo com a qualidade dos dados fornecidos por eles. Já o segundo reduz a quantidade de informações enviadas ao *sink* identificando os nós que possuem área de sensoriamento sobrepostas. Walchli et al. [2007] apresenta um

algoritmo para rastreamento de pessoas, chamado *Distributed Event Localization and TrAcking* (DELTA). Esse algoritmo detecta e rastreia o alvo com base em medições de luz e a na posição dos nós sensores, para isso considera que o alvo se movimenta em velocidade constante e que o mesmo está equipado com uma fonte de luz (uma lanterna, por exemplo). Ele mantém agrupamentos construídos dinamicamente ao redor do alvo assim que esse aparece. Um algoritmo de eleição baseado na medição de luz é feito para escolher o líder do agrupamento. O líder é responsável por coletar e processar os dados e enviar o resultado para o *sink*.

Nos algoritmos baseados em faces, os nós são divididos em grupos de nós organizados em um modelo anel, chamados faces. As faces são construídas de forma que evite os casos em que a comunicações de dois pares de nós se cruzem [Bhuiyan et al., 2010; Hsu et al., 2012]. O *Prediction-based Energy-efficient Target tracking protocol* (PET), proposto por Bhuiyan et al. [2010], e o *Prediction-based Optimistic Object Tracking Scheme* (POOT), proposto por Hsu et al. [2012], usam estrutura de faces para analisar analisa o caminho percorrido pelo alvo e utiliza o padrão dessa movimentação para economizar energia e aumentar o tempo de vida da rede. Esses algoritmos mantém a maioria dos nós inativos, ativando somente os sensores necessários de acordo com a previsão.

Localização é um requisito para aplicações de rastreamento. Dessa forma, algoritmos de localização precisam ser executados antes do rastreamento ser iniciado. Os algoritmos de localização podem ser classificados como não baseados em alcance (*range-free*) [Niculescu & Nath, 2003; Gui et al., 2012] e baseados em alcance (*range-based*) [Albowicz et al., 2001; Oliveira et al., 2009], esses se diferem nos dados usados para calcular as posições dos nós. Os algoritmos não baseados em alcance assumem que os nós são distribuídos uniformemente pelo campo de sensores, de forma que a quantidade de saltos entre dois nós represente a distância entre eles. Já os algoritmos baseados em alcance assumem que os nós são equipados com algum dispositivo que possibilita medir a distância entre os nós [Oliveira et al., 2009].

Nas soluções não baseadas em alcance, Niculescu & Nath [2003] e Chen et al. [2008] apresentam uma soluções de localização em que todos os nós obtêm a quantidade de saltos até os nós *beacons*. Com isso, esse estima a distância média de um salto para calcular a localização dos nós. Fang et al. [2007] apresenta um algoritmo de localização não baseado em distância que dispensa a utilização de nós *beacons*. Esse considera que os nós são implantados em grupos, sendo que os nós de um mesmo grupo são implantados ao redor de uma posição conhecida e com uma probabilidade de distribuição Gaussiana. Com isso, esse calcula a localização dos nós usando estimação estatística, observando os grupos dos vizinhos de cada nó.

Nas soluções baseadas em alcance, Albowicz et al. [2001] propõem uma solução em que os nós *beacons* divulgam suas localizações para ajudar outros nós a se localizarem. Os nós que se localizam se tornam referências para outros nós. Com base nessa solução, Oliveira et al. [2009] propõem um algoritmo que usa estruturas de *beacons* para fazer a recursão iniciar de um único ponto e seguir uma direção conhecida. Com isso, um nó pode estimar sua localização usando apenas duas referências.

Alguns trabalhos encontrados na literatura avaliam o impacto dos erros de localização nos algoritmos geográficos. Oliveira et al. [2009] avalia o efeito dos erros de localização nos algoritmos de roteamento geográfico. Souza et al. [2009] e Campos et al. [2012] avaliam o desempenho de aplicações de rastreamento na presença de erros de localização.

Alguns trabalhos reduzem o erro dos algoritmos de localização. Taylor et al. [2006] e Teng et al. [2012] reduzem o erro de localização dos nós enquanto a aplicação de rastreamento é executada. Esse considera que os nós são pré-localizados usando um algoritmo de localização não baseado em alcance e sem a utilização de nós *beacons*, de forma similar ao algoritmo de localização de Fang et al. [2007]. Dessa forma, esse reduz o erro de localização observando as inconsistências geométricas entre as localizações dos nós e do alvo durante o rastreamento. Souza et al. [2013] reduzem o erro de localização dos nós de algoritmos baseados em alcance. Para isso, esse realiza múltiplas estimativas de distância durante a localização para filtrar os ruídos e obter uma estimativa de distância mais precisa antes do rastreamento ser iniciado. Nesse caso, a aplicação deve definir o *trade-off* entre custo e precisão.

1.3 Objetivos

O objetivo geral deste trabalho é propor e demonstrar soluções para rastreamento de alvos com redes de sensores em áreas discretizadas, visando maximizar a precisão e minimizar o custo de comunicação.

A seguir apresentamos os objetivos específicos separados por capítulo.

Capítulo 2: (i) fornecer uma visão ampla sobre rastreamento de alvos em redes de sensores; (ii) identificar as características comuns entre as soluções de rastreamento e utilizá-las para classificar essas soluções; (iii) identificar e discutir sobre os componentes que formam um algoritmo de rastreamento (e.g., detecção do alvo, cálculo de posição, previsão); e apresentar e discutir sobre as métricas avaliadas (e.g., precisão, atraso, mensagens) e requisitos necessários (e.g., localização, roteamento e sincronização) nas aplicações de rastreamento.

Capítulo 3: (i) demonstrar a viabilidade da utilização das células como referência de posição do alvo; (ii) demonstrar a qualidade das estimativas de posição utilizando um esquema de votação de nós sensores; (iii) e identificar os métodos de previsão mais adequados para o contexto de rastreamento abordado.

Capítulo 4: (i) demonstrar a viabilidade do rastreamento discretizado em células de forma distribuída; (ii) demostrar um esquema híbrido de agrupamento para reduzir a quantidade de mensagens enviadas e preservar a qualidade do rastreamento; (iii) e identificar os métodos de previsões mais adequados para o contexto de rastreamento abordado.

Capítulo 5: (i) minimizar o conjunto de nós ativos com base no deslocamento do alvo registrado pela rede; (ii) e reduzir a rota do objeto guiado utilizando o raio de comunicação nominal dos nós.

Capítulo 6: (i) aproveitar as mensagens enviadas pela aplicação de localização para reduzir os erros de localização; (ii) e com essas mesmas mensagens localizar os nós que permaneceram livres após o processo de localização.

1.4 Contribuições

Neste trabalho demonstramos e avaliamos soluções para dois diferentes tipos de formulações do problema de rastreamento. Além disso, apresentamos uma abordagem que integra localização e rastreamento. A seguir listamos e descrevemos as contribuições deste trabalho.

Levantamento bibliográfico do estado-da-arte sobre rastreamento de alvos em RSSFs. Nessa parte apresentamos e classificamos os trabalhos sobre rastreamento de alvos existentes na literatura. Analisando esses trabalhos, identificamos que o problema de rastreamento de alvos em redes de sensores pode ser formulado de formas diferentes, cada uma delas com seus próprios requisitos, exigindo abordagens específicas.

Além disso, identificamos processos comuns entre os algoritmos de rastreamento e os dividimos em dois tipos de componentes: os básicos e os adicionais. O primeiro tipo inclui componentes responsáveis por encontrar o alvo, já o segundo tipo inclui componentes responsáveis por atividades secundárias, mas que são importantes para o desempenho do algoritmo. Cada um desses componentes representa um tema de pesquisa, sendo assim apresentamos as principais técnicas utilizadas em cada um deles.

A rede de sensores deve satisfazer alguns requisitos para que o rastreamento de alvos seja possível. Um desses requisitos, por exemplo, é a utilização de um algoritmo de localização que possibilite que os nós sensores conheçam as suas localização. Essa informação é fundamental para que os nós sejam capazes de calcular a posição do alvo. Dessa forma, conceituamos e discutimos alguns desses requisitos. As métricas usualmente avaliadas em rastreamento de alvos também são descritas aqui.

Prova-de-conceito da utilização de regiões de ocupação como posições do alvo. Demostramos a utilização de um campo de sensores discretizado em células para rastreamento de alvos. Os nós sensores são dispostos em grade, onde as células formadas são as regiões de ocupação do alvo. A ideia é mostrar que essas células podem ser usadas como referências para determinar de forma simples a posição do alvo e prever a sua próxima posição.

A rede é organizada em grade para viabilizar a comunicação entre os nós sensores dentro da floresta densa, utilizando as vias de acesso existentes nas reservas florestais [Figueiredo et al., 2009], uma vez que essa abordagem foi desenvolvida no contexto dos projetos SAUIM (CNPq 55.4087/2006-5) e RastroAM (CNPq 47.4194/2007-8) para o rastreamento do sauim-de-coleira (primata ameaçado de extinção e presente apenas no Amazonas). Nesse caso, é importante determinar o padrão de movimentação desses animais, correlacionando, por exemplo, com seus hábitos alimentares e comportamento social. A posição exata do alvo é dispensável, sendo importante o conhecimento da região e não a coordenada exata do animal.

Até então, o aspecto distribuído foi desconsiderado, servindo apenas como prova-de-conceito do rastreamento em uma área discretizada. A função dos nós é detectar o alvo e encaminhar os dados até o *sink*, que por sua vez é responsável por toda a fusão de dados. O *sink* realiza um processo de votação para determinar a célula em que o alvo está. Cada nó que detecta o alvo indica que o alvo pode estar em uma das quatro células ao seu redor. A célula que possuir mais votos é a posição do alvo. Criamos dois métodos de votação: a votação simples, em que todos os nós possuem votos de mesmo peso; e a votação ponderada, em que os nós mais próximos do alvo possuem votos de maior peso. As previsões são feitas usando os métodos KF e PF.

Nas simulações avaliamos diferentes cenários de alcance de comunicação e precisão dos sensores. Os resultados mostram que em todos os cenários a votação ponderada possui melhor desempenho para encontrar a célula do alvo, apresentando erros de aproximadamente 0,15 células.

PRATIQUE – Algoritmo baseado em agrupamento e previsão para rastreamento de alvos em áreas quantizadas. Propomos e demonstramos o *Prediction-based clusteRing Algorithm for TrackIng targets in QUantized arEas for wireless sensor networks* (PRATIQUE) para prover o aspecto distribuído ao rastreamento quantizado. O objetivo é realizar o rastreamento do alvo de forma distribuída usando como referência as células formadas pela rede organizada em grade. Dessa forma, a tarefa de determinar a célula onde o alvo está e prever a célula para onde ele está indo fica a cargo dos nós que detectam o alvo.

Para isso, esse algoritmo possui um esquema híbrido de agrupamento: estático e dinâmico. No agrupamento estático a rede é dividida em setores logo depois que a rede é implantada. Os nós que estão localizados dentro de um mesmo setor fazem parte do mesmo agrupamento, sendo que um deles é eleito como líder. O agrupamento dinâmico é formado somente por líderes de agrupamento e depende da cobertura do evento. O agrupamento híbrido visa aproveitar as vantagens desses dois tipos de agrupamento. Dessa forma, evita a dispersão de dados, causados pelos agrupamentos estáticos, e a comunicação excessiva, necessária para a criação de agrupamentos dinâmicos.

No PRATIQUE, quando um evento ocorre, todos os nós geram localmente uma posição prévia para o alvo, essa é imprecisa pois os dados coletados por um único nó são insuficientes para determinar com exatidão a posição do alvo. Esses nós encaminham suas posições para o líder do seu agrupamento através de múltiplos saltos. A cada salto é feita a fusão dos dados de posição. Até esse momento somente o agrupamento estático é utilizado, seu objetivo é melhorar a precisão da posição e reduzir o custo de comunicação. No entanto, um evento pode ocorrer entre dois ou mais agrupamentos, nesse caso cada líder terá dados de posição parciais. Para obter uma resultado mais preciso é necessário que a fusão de todos os dados seja feita em um único nó. Para resolver esse problema um agrupamento dinâmico é criado, sendo esse formado somente pelos líderes envolvidos no evento. Um deles é escolhido para receber todos os dados sobre o evento, calcular a posição do alvo e prever sua próxima posição.

A previsão é feita com KF [Wang et al., 2012], PF [Li & Xu, 2010] ou ABF [Sharma et al., 2011b]. A cada previsão, o estado do filtro é modificado apenas no nó que realizou essa tarefa. Dessa forma, para dar continuidade ao rastreamento, o estado do filtro precisa ser atualizado no próximo nó que realizará a previsão. A forma mais simples e segura é atualizar os líderes de todos os agrupamentos estáticos. Entretanto, essa solução exige um custo de comunicação alto. Por isso, o PRATIQUE utiliza a última previsão para atualizar somente os líderes dos agrupamentos que estão próximos da posição prevista.

Nas simulações avaliamos diferentes cenários de raio de alcance do alvo, impreci-

são dos sensores, quantidade de agrupamentos e número de alvos rastreados. Comparamos a abordagem de agrupamento híbrido com os agrupamentos estáticos e dinâmicos puros. Os resultados mostram que os agrupamentos híbrido e dinâmico levam a cálculos de posição mais precisos, entretanto o custo para a formação do agrupamento híbrido é mais baixo. Também comparamos a precisão das previsões dos métodos KF, PF e ABF nesse contexto de rastreamento, sendo que o KF é o que apresenta melhor desempenho.

TATI – Algoritmo para rastreamento de alvos com objeto guiado. Propomos e avaliamos o *Tracking Algorithm for Target Interception in face-structured sensor networks* (TATI) em um contexto de rastreamento que possui a presença de um objeto guiado que visa alcançar o alvo. O objeto guiado não pode detectar o alvo por si mesmo, por outro lado, ele pode se comunicar com a rede e possui recursos de mobilidade e orientação. Dessa forma, a rede deve ajudar o objeto guiado a se aproximar do alvo, gerando uma rota uma rota entre o objeto guiado e o alvo. O objeto guiado segue a rota que é informada pela rede.

Além de ajudar o alvo, o algoritmo deve reduzir a quantidade de nós em atividade para aumentar o tempo de vida da rede. As abordagens para esse tipo de rastreamento usualmente estruturam a rede em faces para facilitar a seleção do grupo adequado de nós que deve rastrear o alvo. Essa estrutura é criada com técnicas de planarização, essas organizam a rede como um grafo planar, i.e., não deve haver arestas que se cruzem, dividindo o plano em regiões denominadas faces.

O TATI reduz a quantidade de nós ativos para economizar energia. Para isso, as informações geográficas das faces, a posição atual do alvo e a sua velocidade máxima registrada durante o rastreamento são usados como parâmetros para selecionar os nós que devem ser ativados. Somente são ativados os nós das faces que o alvo pode alcançar até a próxima amostragem, independente da direção que ele seguir.

O TATI também reduz a rota entre o objeto guiado e o alvo, dessa forma o alvo pode ser alcançado mais rapidamente. Para isso, esse algoritmo usa a posição atual do objeto guiado e o raio de comunicação nominal dos nós para calcular pontos de consulta, que é a posição onde o objeto guiado consulta a rede para obter atualizações sobre a rota a ser seguida. O ponto de consulta é o caminho mais curto entre o objeto guiado e o nó que possui a informação sobre a rota.

Nas simulações avaliamos diferentes cenários de velocidade, escala, e alcance de comunicação, comparando o desempenho do TATI com o algoritmo *Dynamic Object Tracking* (DOT) [Tsai et al., 2007]. Os resultados mostram que o TATI consume 15% menos energia da rede e o objeto guiado fica cerca de 10m mais próximo do alvo.

Localização e rastreamento simultâneos para reduzir os erros de localização. Propomos e avaliamos um esquema de localização e rastreamento simultâneos em que as mensagens utilizadas durante o rastreamento são aproveitadas para melhorar a localização estimada dos nós. Com isso, o desempenho do rastreamento é melhorado ao longo do tempo e sem custo de comunicação adicional.

As aplicações para rastreamento de alvos em redes de sensores usam as localizações dos nós sensores para calcular a posição do alvo [Tsai et al., 2007; Hsu et al., 2012; Hajiaghajani et al., 2012]. Por outro lado, os nós sensores nem sempre conhecem as suas localizações precisamente, pois essas são estimadas com algoritmos de localização [Oliveira et al., 2009; Souza et al., 2013].

Os algoritmos de localização podem ser classificados como não baseados em alcance (*range-free*) [Niculescu & Nath, 2003; Gui et al., 2012] e baseados em alcance (*range-based*) [Albowicz et al., 2001; Oliveira et al., 2009], esses se diferem nos dados usados para calcular as posições dos nós. Os algoritmos baseados em alcance usam a distância estimada entre os nós, essas distâncias são as responsáveis pelos erros dessa classe de algoritmos de localização, pois possuem ruídos que afetam o cálculo de localização. O erro dos algoritmos de localização prejudicam o desempenho do rastreamento.

Nossa abordagem foca na redução de erros dos algoritmos de localização baseados em alcance. Os nós são pré-localizados com *Recursive Position Estimation* (RPE) ou *Directed Position Estimation* (DPE), dois algoritmos de localização recursivos e baseados em alcance. Nesse processo, cada nó registra as suas referências de localização e a distância estimada para cada uma delas. Depois disso, a rede inicia o rastreamento do alvo.

Os nós que detectam o alvo trocam mensagens para compartilhar a suas observações sobre o evento. Antes de calcular a posição do alvo, os nós aproveitam essas mensagens para filtrar as distâncias estimadas das suas referências e melhorar as suas localizações. A ideia é coletar várias mensagens de uma mesma referência para filtrar os ruídos presentes nas distâncias estimadas entre os nós. Com isso, a localização dos nós e, consequentemente, o rastreamento ficam mais precisos. O processo de filtragem e a aplicação de rastreamento são executados simultaneamente e não exigem custo de comunicação adicional.

Nas simulações verificamos a redução do erro de localização ao longo do tempo em cenários com diferentes escalas e qualidades de estimativas de distância. Comparando com a qualidade da pré-localização, dependendo do cenário, os erros de localização são reduzidos em aproximadamente 55% com o RPE e em aproximadamente 80% com o DPE. Também comparamos o desempenho das técnicas de previsão KF, PF e ABF. O glsPF se destaca nas previsões que são alimentadas com medições com mais ruídos.

1.5 Organização do Documento

A estrutura deste trabalho está organizada como segue. No capítulo 2 apresentamos alguns conceitos e trabalhos relacionados acerca de rastreamento de alvos em redes de sensores sem fio. Além disso, classificamos esses trabalhos de acordo com a sua estrutura e outras características comuns em rastreamento.

No capítulo 3 demonstramos a prova-de-conceito do rastreamento quantizado. O objetivo desse capítulo é mostrar que essas células podem ser usadas como referências para determinar de forma simples a posição atual do alvo e estimar sua posição futura. No capítulo 4 descrevemos o funcionamento do algoritmo PRATIQUE, que usa agrupamentos e previsões para aplicar o rastreamento quantizado de forma distribuída.

No capítulo 5 propomos e avaliamos o TATI. Esse algoritmo é aplicado para ajudar um objeto guiado a se aproximar do alvo rastreado ao mesmo tempo que reduz a quantidade de nós em atividade. No capítulo 6 demostramos um esquema de localização e rastreamento simultâneos que aproveita as mensagens de rastreamento para filtrar os ruídos das estimativas de distância e reduzir os erros de localização. Por fim, o capítulo 7 apresenta as considerações finais e as publicações geradas.

Capítulo 2

Fundamentos Teóricos e Trabalhos Relacionados

Antes de avançarmos para a proposta deste trabalhos, apresentamos alguns conceitos e trabalhos relacionados acerca de rastreamento de alvos em redes de sensores sem fio. Em poucas palavras, rastreamento de alvos consiste em detectar um alvo (como animais, pessoas e veículos) para calcular sua posição atual e, em alguns casos, estimar sua posição futura [Nakamura et al., 2007].

Rastreamento de alvos é uma importante e não trivial aplicação de redes de sensores sem fio [Li & Zhou, 2010]. O propósito das aplicações de rastreamento é coletar dados sobre um ou mais alvos (objetos móveis de interesse) e então estimar suas posições no campo de sensores [Nakamura et al., 2007]. Rastreamento de alvos são aplicados em diferentes áreas, como vigilância, monitoramento ambiental, monitoramento de tráfego e detecção de intrusos [Semertzidis et al., 2010; Komagal et al., 2012; Kozma et al., 2012]. Muitas abordagens de rastreamento de alvos vem sendo investigadas considerando métricas como precisão, atraso, custo de comunicação e consumo de energia. Assim, esses algoritmos buscam manter a precisão do rastreamento e reportar os resultados rapidamente enquanto reduzem o custo de comunicação e o consumo de energia [Souza et al., 2011a; Hajiaghajani et al., 2012; Hsu et al., 2012].

A formulação geral do problema de rastreamento consiste de um grande número de nós sensores estrategicamente implantados [Souza et al., 2011a] ou distribuídos aleatoriamente [Shi & Tan, 2009] pelo campo de sensores. Os nós podem, dentro de um raio de alcance, detectar a presença ou ausência de um alvo pela amostragem de sinais coletados (e.g. luz, som, imagem ou vídeo) [Gustafsson & Gunnarsson, 2007]. Essas medidas são usadas para determinar a trajetória do alvo, que posteriormente são reportados ao nó *sink* [Zhu et al., 2011]. Em simulações, geralmente é associado ao

alvo um modelo de mobilidade [Bettstetter, 2001a; Sarma et al., 2010; Vasanthi et al., 2011] que mais se aproxime do tipo de movimentação do alvo que se deseja rastrear [Yen et al., 2010; Chen et al., 2011a].

Adicionalmente, as aplicações de rastreamento também podem realizar previsões que representam a posição do alvo em um tempo futuro [Nakamura et al., 2007; Sharma et al., 2011b; Kumar & Sivalingam, 2012]. A previsão de posição é uma informação valiosa que é útil tanto para o usuário da aplicação como para a aplicação em si. O usuário da aplicação pode, por exemplo, usar essa informação para interceptar um animal que se dirige para uma área de risco, como estradas dentro de uma reserva ecológica. Já a aplicação pode usar a previsão para ativar os sensores localizados na região em que o alvo está se movendo, enquanto o restante dos nós sensores permanece inativo. Somente uma pequena quantidade de nós sensores precisa estar ativado durante cara período, dessa forma o gasto de energia é minimizado, aumentando o tempo de vida da rede [Xu et al., 2004a; Bhuiyan et al., 2010; Deldar & Yaghmaee, 2011]. Por outro lado, se um grupo de sensores for erroneamente ativado, o alvo será perdido. Nesse caso, um mecanismo de recuperação deve ser disparado [Khare & Sivalingam, 2011]. Filtros Bayesianos, como KF, PF e ABF são frequentemente utilizados para realizar previsões e reduzir os ruídos gerados durante as medições [Xingbo et al., 2011; Li & Wang, 2012; Sharma et al., 2011a].

Neste capítulo, mostramos o estado-da-arte sobre rastreamento de alvos em redes de sensores e organizamos uma classificação dos algoritmos abordados. Além disso, listamos algumas métricas que usualmente são levadas em consideração em aplicações de rastreamento e também alguns pré-requisitos que devem ser satisfeitos para que a rede seja capaz de executar o rastreamento. Para facilitar o entendimento de todo o processo de rastreamento, nós os dividimos em duas categorias de componentes: os componentes básicos e os componentes adicionais.

Uma versão deste capítulo está atualmente em avaliação no ACM Computing Surveys.

2.1 Fundamentos

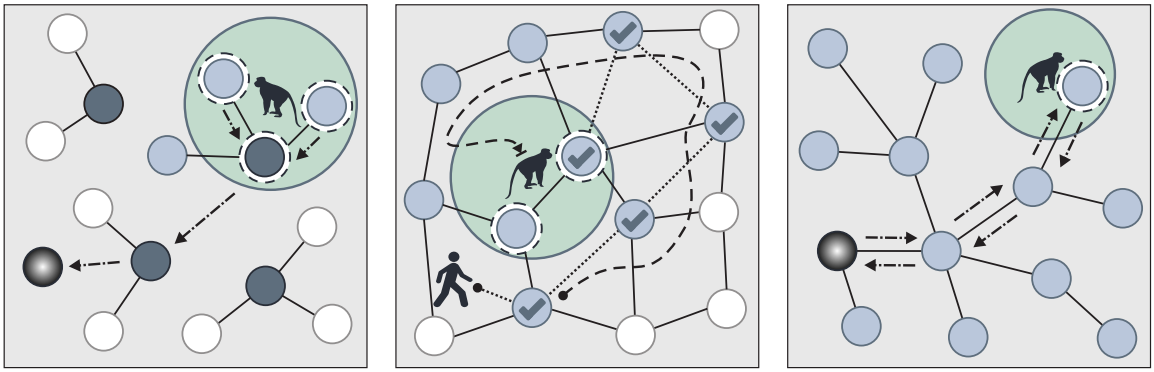
O termo rastreamento de alvos (ou rastreamento de objetos) é usado para descrever diferentes formulações do problema de rastreamento. Essas formulações mudam o funcionamento dos algoritmos e os elementos envolvidos. Assim, nesta seção, mostramos as diferentes formulações do problema de rastreamento os elementos envolvidos em cada um deles.

2.1.1 Terminologia

A terminologia usada para os elementos envolvidos no rastreamento, assim como os próprios elementos, podem mudar dependendo das características do algoritmo de rastreamento, como a formulação do problema(seção 2.1.2), a estrutura usada (seção 2.2.2) e o modo de operação dos nós sensores (seção 2.2.5). A tabela 2.1 descreve os principais elementos dos algoritmos de rastreamento.

2.1.2 Formulações do Problema de Rastreamento

O problema de rastreamento pode ser formulado de diferentes formas. Apesar do principal propósito do rastreamento ser o mesmo, a maneira como o problema é formulado pode mudar significativamente os algoritmos que são aplicados para resolvê-lo. Nós identificamos três tipos de formulações do problema de rastreamento: clássica, navegação guiada e baseada em consulta.



(a) Clássica: Os nós calculam a posição do alvo e periodicamente enviam o resultado para o *sink*. Normalmente a estrutura de agrupamentos é usada nesse caso.

(b) Navegação guiada: Alguns nós são marcados como guia para formar uma rota até o alvo. O objeto guiado deve seguir essa trilha para interceptar o alvo. Normalmente a estrutura de faces e usada nesse caso.

(c) Baseada em consulta: Os nós registram informações sobre a presença do alvo para prover uma consulta de baixo custo. O resultado do rastreamento é enviado ao *sink* somente quando ele solicita (consulta). Normalmente estrutura de árvore é usada nesse caso.

Figura 2.1. Formulações do problema de rastreamento.

A formulação clássica (figura 2.1(a)) é a maneira mais genérica de se realizar o rastreamento. Os nós colaboram entre si para fazer a fusão dos dados e reportar a posição do alvo para o nó *sink* periodicamente. Uma alta taxa de reportes é usada quando a aplicação requer atualizações frequentes sobre a movimentação do alvo. Por outro lado, reduzir a taxa de reportes diminui a quantidade de transmissões, aumentando o tempo de vida da rede [Xu et al., 2004b; Hajiaghajani et al., 2012].

Tabela 2.1. Elementos dos algoritmos de rastreamento de alvos.

Gráfico	Elemento	Outros Termos	Descrição
	Alvo	Objeto móvel	É o elemento que a aplicação visa rastrear, usualmente uma pessoa, animal ou veículo. Dependendo do propósito do rastreamento, mais de um alvo pode ser rastreado simulataneamente.
	Nó ativo	Nó acordado	Um nó ativo é capaz de sensoriar o alvo e se comunicar com outros nós ativos dentro de um certo raio.
	Nó detectando	-	Trata-se de um estado do nó ativo manifestado quando o alvo entra no seu raio de sensoriamento. Nesse estado, o nó coleta dados sobre o alvo.
	Nó inativo	Nó adormecido	É o nó que está com o rádio e/ou sensores desligados, dessa forma ele não pode se comunicar com outros nós ou detectar o alvo. Os nós ficam nesse estado para economizar energia. Dependendo da referência, pode haver diferentes nomenclaturas para identificar as combinações do modo de operação do rádio e sensor.
	Líder de agrupamento	Líder de grupo, nó mestre	Em algoritmos de rastreamento baseados em agrupamento, o líder é responsável por coletar os dados do seu agrupamento, fazer a fusão desses dados e construir uma única mensagem que será reportada ao *sink* (*data report*).
	Nó guia	Âncora	Na formulação de navegação guiada, o nó guia é usado para fornecer a rota que o objeto guiado deve seguir. Eles são marcados de acordo com a trajetória do alvo para formar um caminho até o alvo.
	Objeto guiado	Interceptador, usuário móvel, veículo guiado	O objeto guiado pode ser uma pessoa, um robô, ou um veículo que recebe da rede a posição do alvo. Seu objetivo é alcançar o alvo no menor tempo possível.
	Nó sorvedouro	Estação base, ponto de consulta	É um nó com alto poder de processamento e armazenamento. Ele coleta e armazena os dados da rede e faz a fusão desses dados para obter um resultado mais preciso. Ele também faz o intermédio entre a rede e o usuário da aplicação.

Na navegação guiada (figura 2.1(b)), um objeto guiado é adicionado ao problema. A função do objeto guiado é interceptar o alvo. O objeto guiado pode ser uma pessoa ou um veículo que é informado pela rede sobre a posição do alvo. Dessa forma, a rede forma um caminho que deve ser seguido pelo objeto guiado. O objetivo nesse tipo de rastreamento é interceptar o alvo o quanto antes [Tsai et al., 2007; Bhuiyan et al., 2010; Hsu et al., 2012].

Na formulação baseada em consulta (figura 2.1(c)), os resultados do rastreamento são enviados para o *sink* (nesse caso mais comumente chamado de ponto de consulta) somente quando ele faz a consulta. A rede atua como uma bando de dados distribuído, em que os nós registram informações sobre a presença do alvo para possibilitar uma consulta de baixo custo. Nesse caso, esse banco de dados distribuído deve ser atualizado de forma que minimize o custo da consulta. Também é importante que o custo dessa atualização seja levado em consideração, para manter o equilíbrio entre atualização e consulta [Kung & Vlah, 2003; Lin et al., 2006; Liu et al., 2008].

2.2 Componentes dos Algoritmos de Rastreamento

Nós podemos identificar três componentes básicos (figura 2.2) nos algoritmos de rastreamento de alvos: detecção, cooperação e cálculo de posição. Com os componentes básicos é possível obter a posição atual do alvo, funcionalidade mínima para uma aplicação de rastreamento. O alvo emite sinais que podem ser lidos pelos nós sensores, a interpretação desses sinais caracteriza a detecção do alvo. Então, os nós sensores compartilham informações para calcular a posição do alvo.

O algoritmo de rastreamento pode ter três componentes adicionais se ele prevê a futura posição do alvo e usa essa informação para economizar a energia da rede (figura 2.2): previsão, ativação e recuperação. Para conservar energia usando a previsão é necessário manter ativado somente os nós que irão detectar o alvo em um tempo particular. A escolha de quais nós deverão ser ativados e desativados é feita pelo mecanismo de ativação. Se essa escolha for errada o alvo é perdido, então para reiniciar o rastreamento é necessário disparar o mecanismo de recuperação.

2.2.1 Detecção

A detecção de eventos é um elemento essencial para várias aplicações em redes de sensores. O evento é detectado quando as leituras dos sensores correspondem às condições que descrevem o evento. Em rastreamento de alvos, um evento ocorre quando o alvo

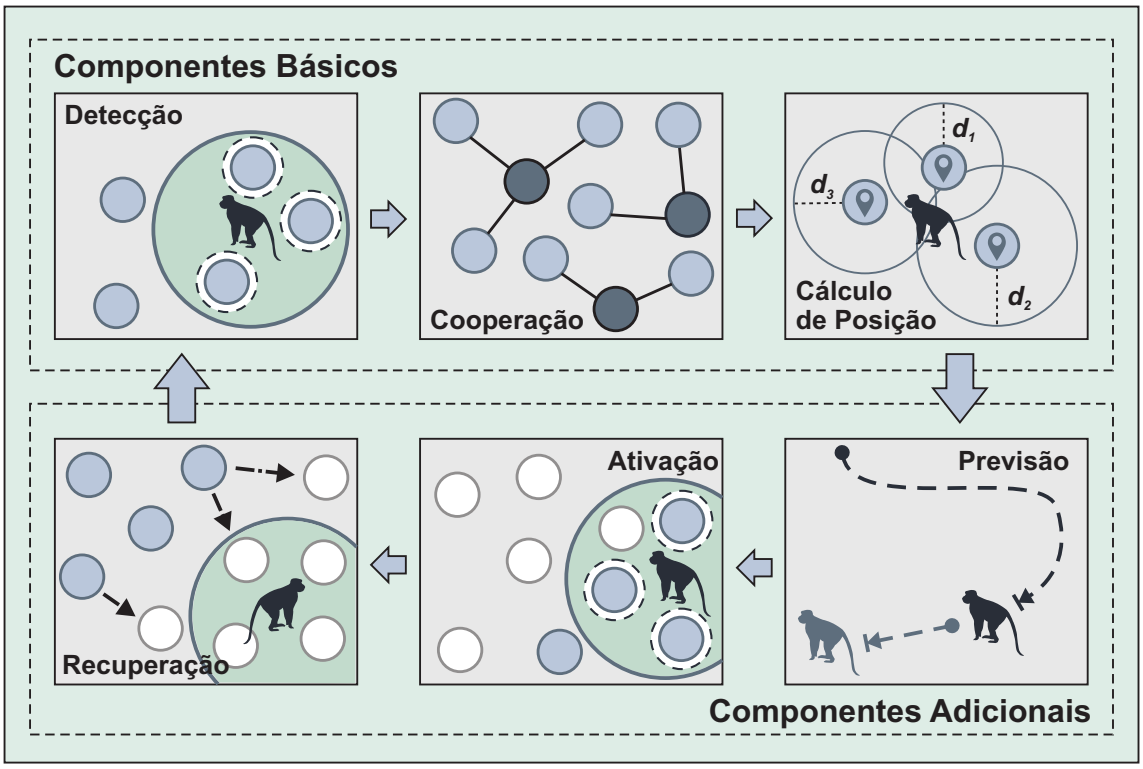


Figura 2.2. Componentes dos algoritmos de rastreamento. Detecção: os nós sensores coletam e analizam os sinais emitidos pelo alvo para determinar sua presença; Cooperação: os nós divulgam seus dados de forma coordenada para realizar fusão de dados e reduzir a quantidade de mensagens; Cálculo de Posição: os dados coletados pelos nós são usados para determinar a posição do alvo; Previsão: a futura posição do alvo é prevista baseado no histórico de movimentação do alvo; Ativação: para economizar energia, um pequeno conjunto de nós permanece ativado em um dado momento para detectar o alvo; Recuperação: se a previsão falhar, o conjunto de sensores ativos pode não detectar o alvo, causando a perda do mesmo. Para reencontrar o alvo, os nós ativos iniciam uma busca na rede.

se aproxima de um ou mais nós que podem detectá-lo. Em outras palavras, o evento a ser detectado é a presença do alvo [Vairo et al., 2010; Xue et al., 2011].

Na prática, o processo de detecção é realizado analisando amostras dos sinais emitidos pelo alvo que são coletados pelos nós sensores. Vários tipos de amostras podem ser usados nesse processo, o tipo de sinal a ser coletado depende do tipo de alvo que deve ser rastreado [Gustafsson & Gunnarsson, 2007; Jin et al., 2012].

Por outro lado, alguns modelos de detecção são usados para validar algoritmos. Esses modelos tornam os ambientes simulados mais próximos da realidade. Dessa forma, o comportamento dos algoritmos pode ser examinado em condições adversas que ocorrem na prática [Nakamura & Souza, 2010].

2.2.1.1 Sensores e Amostras

Para o alvo ser rastreado por uma rede de sensores, ele precisa ser detectado pelos nós sensores. Os nós podem ser equipados com vários tipos de sensores. Algumas das amostras que podem ser usadas para detecção de alvos são: acústica, sísmica, magnética, rádio frequência, *Radio Frequency Identification* (RFID), luz, radar, imagem e vídeo.

Sensores acústicos podem ser usados para distinguir objetos pelas diferentes assinaturas de sons. Chen et al. [2004] rastreiam alvos usando sinais acústicos. Os nós sensores captam o nível de energia dos sinais acústicos para analisar e classificar o som. Um nó sensor acusa a presença do alvo quando a força do sinal excede um limiar definido. Isbitiren & Akan [2011] propõem usar redes de sensores subaquáticas acústicas como uma alternativa aos tradicionais arranjos de sonar. Essa técnica é baseada nos ecos vindos dos alvos depois que pulsos acústicos são emitidos pelos nós.

Sensores sísmicos podem distinguir alvos com diferentes pesos e velocidades. Jin et al. [2012] fazem uso de sensores sísmicos e infravermelho passivo para detecção e classificação de humanos, veículos leves e animais domésticos. Wahlström et al. [2010] e Wahlström et al. [2011] usam magnetômetros para detectar alvos metálicos (veículos). Estes trabalhos apresentam um modelo de sensor com magnetômetros de três eixos que depende do modelo físico do alvo e da sua posição em relação ao sensor.

Pereira et al. [2008] e Oka & Lampe [2010] integram RFID com redes de sensores para rastrear alvos. Cada alvo carrega um RFID ativo que possibilita a identificação de cada um deles. Walchli et al. [2007] mostram um algoritmo para rastreamento de pessoas que usa sensores de luminosidade. Para isso, o alvo deve estar equipado com alguma fonte de luz, como uma lanterna. A intensidade das medidas de luz são usadas nos cálculos de posição e na escolha dos nós líderes.

Chiani et al. [2009] mostram uma rede de sensores com radar para detectar alvos com base no impulso de rádio de banda ultralarga. A rede é formada por um nó transmissor e vários nós receptores. A cada varredura, o nós receptores formam uma imagem do ambiente e a enviam para o nó *sink*, onde a decisão sobre a posição do alvo é tomada. Kozma et al. [2012] descrevem a integração de radares de baixa potência com sensores acústicos e sísmicos para detectar, identificar e rastrear veículos em uma área extensa e em condições de ruído.

Lee & Choi [2011] apresentam um algoritmo de rastreamento para alvos que mudam de forma em imagens de vídeo. O processo de detecção extrai os objetos em movimento por segmentação entre os quados. Komagal et al. [2012] propõem técnicas para detecção de alvos em imagens de vídeo. Essas técnicas possibilitam ignorar o

plano de fundo da imagem para identificar os elementos no primeiro plano, que no caso são os alvos.

2.2.1.2 Modelos de Detecção de Eventos

As abordagens baseadas em eventos em redes de sensores usualmente são avaliadas considerando um modelo de detecção de eventos. A eficiência da detecção depende de vários fatores, como condições do ambiente, confiabilidade do sensor e características do evento [Nakamura & Souza, 2010]. Assim, modelos de detecção de eventos visam incluir as interferências desses fatores nas avaliações dos algoritmos [Pujolle et al., 2009].

O mais simples desses modelos é o modelo de detecção binário [Wang et al., 2010a]. Nesse modelo, para um dado evento e, cada sensor s, cuja distância d é menor que a um raio de detecção R, asseguradamente detecta o evento. Então a probabilidade do evento ser detectado é definido como

$$P(s,e) = \begin{cases} 1, & \text{se } d \leq R \\ 0, & \text{caso contrário} \end{cases}. \tag{2.1}$$

O modelo de detecção binário geralmente é usado para avaliar os algoritmos em condições ideais. Wang et al. [2010a] e He et al. [2010] apresentam algoritmos de rastreamento usando detecção binária em redes de sensores.

A probabilidade de um evento ser detectado depende da distância entre o sensor e o evento, além das condições físicas do ambiente. Dessa forma, o modelo de detecção probabilístico inclui parâmetros que representam a tecnologia do sensor e fatores ambientais (α e β). Além disso, ele considera que a probabilidade de se detectar um evento é inversamente proporcional a uma função linear [Lu & Suda, 2003] ou a uma função exponencial Dhillon et al. [2002] com a distância que separa o sensor do evento. Essas funções são, respectivamente, definidas como

$$P(s,e) = (1 + \alpha d)^{-\beta} \tag{2.2}$$

e

$$P(s,e) = \varepsilon^{-\alpha d}. \tag{2.3}$$

Huang et al. [2011] detectam múltiplos alvos usando múltiplos sensores cooperativamente, em que é dado a cada alvo um limiar para a probabilidade de detecção.

Zhang et al. [2006] estendem os modelos probabilísticos adicionando a influencia do tempo de duração do evento t em aplicações de rastreamento. A ideia é aumentar a

probabilidade da detecção quando o evento permanece por mais tempo em um mesmo ponto. Esse modelo é definido como

$$P(s,e,t) = \begin{cases} P(s,e), & \text{se } 0 < t \leq T \\ 1 - (1 - p(s,e)^{\lfloor t/T \rfloor}, & \text{se } t > T \end{cases}, \tag{2.4}$$

em que T é o período de tempo em que o algoritmo de sensoriamento é executado.

Um modelo de detecção híbrido que une os modelos binário e probabilístico é apresentado por Zou & Chakrabarty [2004]. O modelo híbrido é baseado em um limiar r e considera estas três diferentes situações

$$P(s,e) = \begin{cases} 1, & \text{se } d < r \\ 0, & \text{se } d > R \\ \alpha e^{-\beta d}, & \text{se } r \leq d \leq R \end{cases}. \tag{2.5}$$

Nakamura & Souza [2010] propõem um modelo flexível de detecção de eventos que possibilita modelar vários cenários de detecção usados em redes de sensores, como raio fixo e modelo exponencial. Para isso, é necessário somente ajustar quatro parâmetros. O modelo flexível é definido como

$$P(s,e) = \alpha\gamma^{(\beta d)^{\theta}}, \tag{2.6}$$

em que $0 < \alpha \leq 1$ é o parâmetro de precisão, $\gamma > 1$ e $\beta > 0$ são, respectivamente, os parâmetros de espeço vertical e horizontal, e $\theta > 0$ é o parâmetro de inclinação. As avaliações experimentais desse modelo são realizadas para aplicações de rastreamento.

2.2.2 Cooperação

A cooperação entre os nós da rede pode ser considerado o mais importante componente das aplicações de rastreamento. Nesse componente é definido como ocorre a comunicação entre os nós quando um evento é detectado [Chen et al., 2004; Liu et al., 2008; Wang et al., 2010b]. Esse processo é importante para reduzir a quantidade de mensagens e conservar energia, pois facilita a fusão de dados. A maioria das abordagens de rastreamento visam melhorar esse componente [Nakamura et al., 2007; Yeong-Sung et al., 2010; Bhuiyan et al., 2010; Deldar & Yaghmaee, 2011].

A metodologia de rastreamento mais simples fazem uso da abordagem centralizada, em que os nós sensores que detectam o alvo enviam seus dados para o nó *sink*. Nesse caso, o nó *sink* é responsável pela fusão de todos os dados enviados pela

rede para calcular a posição do alvo. A medida que o número de nós sensores cresce, mais mensagens são enviadas para o *sink*, aumentando o custo de comunicação. Além disso, muitas mensagens são perdidas por causa tráfego excessivo [Souza et al., 2009; Singh et al., 2011].

Por essas razões, os nós sensores cooperam entre si para enviar ao *sink* somente o resultado do rastreamento. Para facilitar o processamento de dados colaborativo em aplicações de rastreamento, a estrutura lógica da rede geralmente é organizada como árvore [Lin et al., 2006; Yen et al., 2010], agrupamento [Xu & Lee, 2007; Mansouri et al., 2011] ou face [Huang et al., 2005; Tsai et al., 2007; Ji et al., 2009; Bhuiyan et al., 2010; Hsu et al., 2012].

2.2.2.1 Cooperação Baseada em Árvore

Na cooperação baseada em árvore, a estrutura lógica da rede é organizada como uma árvore ou grafo, em que os vértices representam os nós sensores e as arestas representam a ligação entre os nos que podem se comunicar diretamente (figura 2.1(c)).

Uma abordagem desse tipo de cooperação é usar o nó *sink* como raiz. Dessa formar, o *sink* inicia um *flooding* para encontrar o caminho mais curto entre ele e cada um dos nós. A rede pode ser reconfigurada caso ocorram alterações na organização física da rede, e.g. nós que ficaram sem bateria ou mudaram de posição [Tran & Yang, 2006; Souza et al., 2011a].

Outra abordagem é a árvore em movimento dinâmico que acompanha o alvo. Quando o alvo é detectado pela primeira vez, os nós sensores que o detectaram constroem a árvore inicial. O nó mais próximo do alvo usualmente é o nó raiz. Os nós sensores são adicionados à árvore (expansão) a medida que o alvo se aproxima deles. Os sensores também podem ser removidos da árvore (poda) quando o alvo se distancia deles. O nó raiz também precisa ser alterado de acordo com a trajetória do alvo para evitar sobrecarga de comunicação. Para isso, um simples critério de seleção é escolher como raiz o nó mais próximo do centro geográfico da árvore, reduzindo a quantidade de comunicações durante a coleta de dados. Para enviar dados sobre o evento ao *sink*, os nós folhas enviam seus dados para os seus respectivos nós pais. Os nós intermediários fazem a fusão dos seus dados com os dados recebidos de seus filhos para gerar um novo resultado mais preciso e enviar para seus pais. Esse processo é repetido até que o nó raiz receba os dados de todos os seus descendentes e gere o resultado final que é enviado ao *sink*. A cooperação baseada em árvore reduz a transmissão de dados redundantes e tráfego da rede, aumentando o tempo de vida da rede [Zhang & Cao, 2004a,b].

A rede também pode ser organizada como um banco de dados distribuído

(*message-pruning tree*), uma vez que cada nó sensor registra quando o alvo entra ou sai de seu raio de detecção. A hierarquia é usada para conectar os sensores, o nó *sink* (ponto de consulta) é usado como raiz. Nessa abordagem, o processo de rastreamento possui duas operações: atualização e consulta. As atualizações da posição do alvo são realizadas quando o alvo se move do raio de detecção de um sensor para outro. Uma consulta pode ser feita sempre que for preciso conhecer a posição do alvo. O objetivo é manter os nós sensores sempre informados sobre a posição do alvo e possibilitar uma consulta eficiente. A consulta é iniciada do topo da hierarquia, seguindo um único caminho em direção ao nó que relatou sobre o alvo requerido. Sem a atualização dos nós, as consultas precisariam percorrer toda a rede. Essas árvores devem ser construídas para reduzir o custo de comunicação da rede, para isso devem levar em consideração os custos de atualização e consulta [Kung & Vlah, 2003; Lin & Tseng, 2004; Lin et al., 2006; Liu et al., 2008; Liu, 2010].

2.2.2.2 Cooperação Baseada em Agrupamentos

Em aplicações de rastremento, a cooperação baseada em agrupamento é usada para facilitar o processamento de dados colaborativo e gerenciar os recursos escassos das redes de sensores. Agrupamentos são úteis principalmente quando a aplicação requer escalabilidade para centenas ou milhares de nós, pois dá suporte para fusão de dados e reduz interferência durante a comunicação.

Os nós sensores são organizados em agrupamentos, cada agrupamento consiste de um nó líder (*Cluster Head* (CH)) e vários nós membros (*Cluster Member* (CM)). Um CH é responsável por coletar dados de seus CMs, então faz a fusão dos dados e envia o resultado para o nó *sink* (figura 2.1(a)). Estruturas baseadas em agrupamentos são divididos em três abordagens: estática [Olule et al., 2007; Xu & Lee, 2007; Deldar & Yaghmaee, 2011; Bhatti et al., 2011], dinâmica [Ji et al., 2004; Chen et al., 2004; Jin et al., 2006; Yang et al., 2007; Lee et al., 2007; Suganya, 2008; Jian et al., 2011] e híbrida (estática/dinâmica) [Chang et al., 2008; Wang et al., 2010b; Hajiaghajani et al., 2012].

Nos agrupamentos estáticos, os agrupamentos são criados durante a implantação da rede, i.e. antes do rastreamento ser iniciado. As propriedades de cada agrupamento, como o tamanho do mesmo e seus membros, são fixas. Entretanto, os nós podem revezar na função de CH periodicamente para distribuir o gasto de energia de maneira homogênea. A escolha do CH afeta a eficiência energética da rede, então várias estratégias quem consideram a energia residual e a localização dos nós são estudadas [Deosarkar et al., 2008; Koucheryavy & Salim, 2009]. Alguns algoritmos de rastrea-

mento consideram que os agrupamentos são criados usando técnicas de roteamento hierárquico [Chang et al., 2008], como *Low Energy Adaptive Clustering Hierarchy* (LEACH) [Yektaparast et al., 2012] e *Power-Efficient GAthering in Sensor Information System* (PEGASIS) [Chen et al., 2011b]. O agrupamento estático tem a vantagem da simplicidade e de reduzir o custo de comunicação, mas tem baixa tolerância à falhas e os nós sensores de agrupamentos diferentes não podem compartilhar informações entre si, causando o chamado problema de fronteira [Wang et al., 2010b] – quando o alvo está localizado na fronteira entre agrupamentos.

Nas abordagens dinâmicas, os agrupamentos são formados a medida que o alvo se move pelo campo de sensores. Um nó não fica restrito a um único agrupamento, ele pode ser membro de diferentes agrupamentos. A formação de um agrupamento é iniciada pelo presença do alvo, os nós que detectam o alvo trocam mensagens entre eles para decidir quem deve ser o CH [Chen et al., 2004]. Agrupamentos dinâmicos melhoram o resultado do rastreamento porque toda a fusão de dados é realizada em um único CH, diferente das abordagens estáticas que pode ocorrer em mais de um. Nesse caso, a colaboração entre os nós não é completa, pois os dados coletados são divididos entre CHs de agrupamentos diferentes, aumentando a incerteza da localização do alvo [Wang et al., 2010b]. Apesar das vantagens dos agrupamentos dinâmicos, os requisitos para eleger o CH aumenta o consumo de energia, uma vez que essa escolha deve ser feita várias vezes durante o rastreamento [Alaybeyoglu et al., 2009].

A abordagem de agrupamento híbrido alterna entre agrupamentos estáticos e dinâmicos durante o rastreamento. O objetivo é unir a economia de energia dos agrupamentos estáticos com a flexibilidade e qualidade dos agrupamentos dinâmicos. Assim, os agrupamentos estáticos são criados durante a implantação da rede. Se um evento ocorre dentro dos limites de um dado agrupamento, o próprio agrupamento realiza a coleta e a fusão dos dados. Por outro lado, se o evento cobre dois ou mais agrupamentos, esses são unidos para formar um agrupamento dinâmico. Criar um agrupamento dinâmico a partir de agrupamentos estáticos requer menos trocas de mensagens quando comparado com a formação de um agrupamento dinâmico puro [Chang et al., 2008; Wang et al., 2010b; Hajiaghajani et al., 2012].

2.2.2.3 Cooperação Baseada em Faces

Na cooperação baseada em faces, os sensores da rede são divididos em várias faces. Uma face consiste de um grupo de nós organizados em um modelo anel (figura 2.1(b)). Um nó não precisa conhecer todos os sensores e as suas localizações, ele somente precisa saber as informações dos nós que estão em suas faces adjacentes [Bhuiyan et al., 2010].

A construção de uma estrutura de faces tem duas partes: gerar o grafo planarizado; e identificar as faces na rede [Hsu et al., 2012]. A primeira parte visa evitar os casos em que a comunicações de dois pares de nós se cruzem. Deste modo, para quaisquer dois nós que podem comunicar directamente, é necessário verificar se existe um outro nó intermediário, de tal forma que a distância de comunicação entre os dois nós pode ser dividida em duas distâncias mais curtas. Geralmente *Relative Neighborhood Graph* (RNG) ou *Gabriel Graph* (GG) [Maignan & Gruau, 2011; Tsai et al., 2007] são usados para essa tarefa. Na segunda parte, cada nó precisa conhecer os outros nós da mesma face, a regra da mão direita Huang et al. [2005] é usada para isso. Essa regra basicamente cria um fluxo de mensagens em cada face. Como uma mensagem atravessa uma face descoberta, as coordenadas dos nós que tenha atravessado são adicionadas à mensagem. Depois que a mensagem de descoberta termina de percorrer a face, todas as localizações dos nós da face são coletadas e a mensagem percorre a face novamente para informar a todos os nós as informações coletadas da face.

A cooperação baseada em agrupamentos, geralmente, tem um bom desempenho e é amplamente utilizada em redes de sensores. No entanto a sobreposição entre dois agrupamentos e a seleção de CHs precisa de comunicação redundante que pode causar resultar em um alto custo de comunicação [Lee et al., 2007]. A cooperação baseada em face pode resolver o problema dos agrupamentos, pois ele não tem problema de dados duplicados [Ji et al., 2009]. Além disso, ela é adequada para redes de qualquer densidade por ser capaz de resolver os problemas de buracos e obstáculos [Tsai et al., 2007].

2.2.3 Cálculo de Posição

O componente de cálculo de posição encontra a posição do alvo usando a localização dos nós que o detectaram e a distância desse nós para o alvo. Os algoritmos de rastreamento consideram que os nós conhecem sua localização a priori ou que eles a calculam usando algum algoritmo de localização [Boukerche et al., 2007; Albowicz et al., 2001; Oliveira et al., 2009]. As mesmas técnicas usadas para calcular a localização dos nós podem ser usadas para calcular a posição do alvo. Esse componente tem dois passos: estimativa de distância; e estimativa de posição.

2.2.3.1 Estimativa de Distância

A estimativa de distância é feita de acordo com as amostras de sinal emitidas pelo alvo e coletadas pelos nós sensores durante a detecção do alvo. A potência do sinal emitido pelo alvo diminui com a distância entre o alvo e o nó receptor. Em outras palavras,

quanto maior for a distância entre o nó e o alvo, menor a intensidade do sinal, então esse sinal pode ser usado para estimar a distância entre o alvo e o nó.

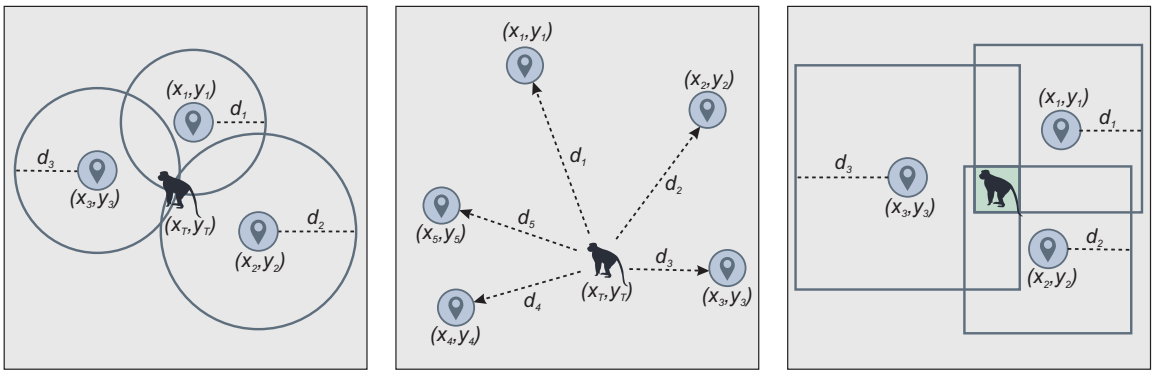
Chen et al. [2004] estima a distância entre o nó e o alvo a partir de sinais acústicos. Um nó detecta o alvo quando a potência de som recebida ultrapassa um determinado limiar, então a potência desse sinal é usada para estimar a distância entre eles. Teoricamente, a potência do sinal é inversamente proporcional ao quadrado da distância, e um modelo conhecido de propagação do sinal pode ser usado para converter a potência do sinal em distância [Alawi, 2011].

2.2.3.2 Estimativa de Posição

Quando os nós que detectam o alvo têm informações sobre sua localização e a sua distância para o alvo, eles podem calcular a posição do alvo com trilateração, multilateração ou *bounding box* [Boukerche et al., 2007; Oliveira et al., 2009].

A trilateração calcula a posição do alvo pela interseção de três círculos, como mostra a figura 2.3(a). Os círculos formados pela posição e distância de cada nó pode ser representada por $d_i^2 = (x_T - x_i)^2 + (y_T - y_i)^2$, em que (x_T, y_T) é a posição do alvo, (x_i, y_i) é a localização do i-ésimo nó, e d_i é a distância do i-ésimo nó para o alvo. Nesse caso, existe três equações com duas incógnitas. Por outro lado, quando mais de três nós detectam o alvo, a multilateração pode ser usada (figura 2.3(b)). Nesse caso, um sistema sobredeterminado de equações deve ser resolvido.

O *bouding box* usa quadrados ao invés de círculos, como na figura 2.3(c). Cada nó i tem um quadrado com seus centros na posição (x_i, y_i), com lados de tamanho $2d_i$, e com coordenadas $(x_i - d_i, y_i - d_i)$ e $(x_i + d_i, y_i + d_i)$. As interseções dos quadrados podem ser facilmente calculadas sem a necessidade de operações de ponto flutuante. A posição final do alvo é o centro de todas as interseções.



(a) Trilateração (b) Multilateração (c) Bounding box

Figura 2.3. Técnicas para estimar a posição do alvo.

2.2.4 Previsão

O componente de previsão dá a futura posição do alvo [Nakamura et al., 2007; Guo et al., 2006; Kumar & Sivalingam, 2012]. Essa informação pode ser usada para conservar energia, pois é possível ativar os nós em regiões próximas à posição do alvo [Xu et al., 2004b; Bhuiyan et al., 2010; Hong et al., 2010; Deldar & Yaghmaee, 2011].

Para isso, métodos de estimação são usados para prever a posição do alvo a partir de um conjunto de medidas obtidas dos sensores [Nakamura et al., 2007]. Alguns métodos de estimação geralmente usados para rastreamento de alvos em redes de sensores são KF [Xu et al., 2012], ABF [Sharma et al., 2011b] e PF [Gustafsson, 2010; Jiang & Ravindran, 2011].

2.2.4.1 Filtro de Kalman

O KF é o método de fusão de dados mais popular usado para fundir um baixo nível de dados redundantes [Nakamura et al., 2007]. Se um modelo linear pode ser descrito como um sistema e o erro pode ser modelado como um ruído com distribuição de probabilidade Gaussiana, o filtro de Kalman gera recursivamente estimativas estatisticamente ótimas.

Esse método, como mostrado na figura 2.4(a), faz com que em cada incremento de tempo discreto, um operador linear seja aplicado ao estado atual para gerar um novo estado, com algum ruído agregado a ele e, opcionalmente, alguma informação dos controles no sistema, se eles são conhecidos. Então, outro operador linear agregado a mais ruído gera a saída.

O filtro de Kalman estima o estado x de um tempo discreto k baseado no modelo de espaço de estados governado pela equação do processo

$$x_{k+1} = Ax_k + Bu_k + w_k, \tag{2.7}$$

com medições y representadas por

$$y_k = Cx_k + v_k, \tag{2.8}$$

em que A é a matriz de transição de estados, B é a matriz de controle de entrada que é aplicada ao vetor de controle u, C é a matriz de medições; w é o ruído do processo e v é o ruído da medição, modelados por variáveis aleatórias que obedecem as leis Gaussianas de média zero com matrizes de covariância Q e R respectivamente.

Baseado nas medições y e nos parâmetros do sistema, a estimativa de x, repre-

sentada por $\hat{x}$ é obtida por

$$\hat{x}_{k+1} = (A\hat{x}_k + Bu_k) + K_k(y_k - C\hat{x}_k), \tag{2.9}$$

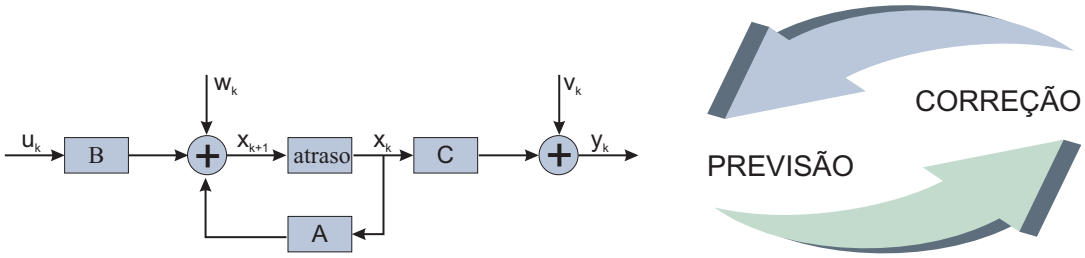
em que K é o ganho de Kalman determinado por

$$K_k = P_k C^T (CP_k C^T + R)^{-1}, \tag{2.10}$$

enquanto P é a matriz de covariância da previsão que é determinado por

$$P_{k+1} = A(I - K_k C)P_k A^T + Q. \tag{2.11}$$

A figura 2.4(b) mostra que o filtro de Kalman possui duas fases: previsão e correção. A fase de previsão é responsável por projetar o estado atual, obtendo o estado no próximo tempo, sendo formado pelas Equações 2.7 e 2.8. A fase de correção é responsável por incorporar a nova medição ao estado atual para obter uma estimativa mais precisa considerando todo o histórico das medições. Essa fase é formada pelas Equações 2.9, 2.10 e 2.11 [Nakamura et al., 2007].



(a) Diagrama de bloco do filtro de Kalman. (b) Fases do filtro de Kalman.

Figura 2.4. Representação do filtro de Kalman e suas fases (figura adaptada de Nakamura et al. [2007]).

O filtro de Kalman permite a elaboração de um algoritmo computacional capaz de estimar os valores ótimos do vetor de estado. Dessa forma, é possível gerar uma sequência de valores de estado por unidade de tempo, prevendo estados futuros com a utilização de estados atuais, permitindo a criação de sistemas com atualizações em tempo real.

Como grande parte dos problemas não podem ser modelados como funções lineares, surgiram algoritmos baseados na formulação original do KF para tratar esses problemas. As principais variações do KF para problemas Gaussianos não lineares são o *Extended Kalman Filter* (EKF) [Zhao et al., 2011; Aydogmus & Talu, 2012] e

o *Unscented Kalman Filter* (UKF) [Julier & Uhlmann, 2004; Gustafsson & Hendeby, 2012]. O EKF é a alternativa mais popular para problemas não lineares. Esse método utiliza o modelo linearizado do processo utilizando séries de Taylor, mas devido a isso é um estimador sub-ótimo sem garantia de convergência. Já o UKF realiza estimações em sistemas dotados de não-linearidade sem a necessidade de linearizá-los, pois usa o princípio de que um conjunto de pontos de amostragem discreta pode ser usado para parametrizar a média e a covariância. A qualidade das suas estimativas chega próxima do KF padrão para sistemas lineares. LaViola [2003], Orderud [2005], and Gustafsson & Hendeby [2012] comparam o desempenho do EKF e do UKF em aplicações de rastreamento.

Existem vários trabalhos que visam comparar as variações do KF. Wan & Merwe [2000] comparam o EKF e o UKF nas estimações de estados e nas estimações duplas (estimativa de estado e estimativa de parâmetro). Para isso, é verificado o desempenho desses métodos para estimar a série temporal de *Mackey-Glass*. O UKF apresenta um melhor desempenho nessa avaliação. LaViola [2003] propõe um estudo empírico que compara o desempenho do EKF e do UKF no rastreamento de movimento humano para aplicações de realidade virtual na presença de dados ruidosos. Os seus resultados mostram que os desempenhos dessas técnicas são equivalentes para o cenário adotado, mas conclui que o EKF é a melhor alternativa, devido ao custo computacional do UKF. Orderud [2005] compara o desempenho das estimativas do EKF com as estimativas do UKF. Nessa comparação ele utiliza dois modelos de rastreamento com medições não-lineares: um modelo de rastreamento por radar e um modelo de rastreamento por triangulação. Em ambos os casos o UKF obtém melhores resultados.

Lin et al. [2009b], Olfati-Saber & Jalalkamali [2012] e Hsu et al. [2012] usam o KF para prever a posição do alvo em aplicações de rastreamento em redes de sensores. Para ser aplicado em redes de sensores, o KF pode ser descentralizado ou distribuído. Na versão descentralizada, a estimativa de estados usa um conjunto de filtros locais, em que cada nó se comunica com todos os outros para sincronizar os filtros, tonando-a pouco eficiente devido ao excesso de comunicação. Rao et al. [1993] aplicam um KF descentralizado para rastreamento e vigilância em redes de sensores. Já na versão distribuída, os nós precisam se comunicar apenas com seus vizinhos, essa estratégia é mais viável para as redes de sensores por ser mais escalável. Olfati-Saber [2007] e Olfati-Saber & Jalalkamali [2011] propõem o *Distributed Kalman Filter* (DKF) para redes de sensores, em que um KF centralizado é decomposto em múltiplos micro-KFs. Ele mostra que essa abordagem distribuída alcança resultados tão bons quanto um filtro de Kalman centralizado. Para reduzir o custo de comunicação necessário para manter o KF sincronizado.

Lin et al. [2009b] desenvolveram um KF comprimido, substituindo a matriz de covariância por uma matriz diagonal. Seus resultados mostram que é possível melhorar a eficiência da rede em termos de energia e manter a precisão do algoritmo de rastreamento.

Wang et al. [2012] propõem um modelo de ruído melhorado aplicado na distância obtida pelos sensores. Esses usam uma estimativa por máxima verossimilhança para obter uma pré-localização e remover a não linearidade antes de aplicar o KF padrão. Os nós sensores fornecem somente a distância entre eles e o alvo, entretanto o estado do alvo é composto por posição e velocidade, por isso as medidas dos sensores são não lineares no estado do alvo. Nessa situação o EKF geralmente é utilizado, mas seus resultados insatisfatórios serviram de motivação para esse trabalho.

2.2.4.2 Filtro Alfa-Beta

O ABF [Sharma et al., 2011b] é um estimador simplificado bastante similar ao KF, mas possui a vantagem de não exigir um modelo de sistema detalhado. Esse filtro presume que um sistema está adequadamente aproximado por um modelo de dois estados internos, em que o primeiro estado é obtido através da integração do valor do segundo estado ao longo do tempo. O princípio básico do ABF é ajustar os valores dos coeficientes $0 < \alpha < 1$ e $0 < \beta < 1$ (que dão nome ao filtro) para corrigir a previsão com base no valor medido.

O ABF é uma escolha simples e eficiente para aplicações de rastreamento de alvos em redes de sensores, uma vez que sua baixa complexidade computacional justifica seu uso em dispositivos com recursos limitados, como os nós sensores.

Esse filtro aproxima o movimento do alvo usando como estados internos as variáveis $\hat{x}$ (posição) e $\hat{v}$ (velocidade). Os valores de medida do sistema correspondem às observações de localização x no k-ésimo instante, mais o ruído da medida. Supondo que a velocidade se mantém aproximadamente constante entre as medidas ao longo do intervalo de tempo ΔT, o ABF estima a posição pela equação

$$\hat{x}_k = \hat{x}_{k-1} + \Delta T \hat{v}_{k-1}. \tag{2.12}$$

Como assumimos que $\hat{v}$ é constante, o valor projetado na próxima amostra é igual ao valor atual, então

$$\hat{v}_k = \hat{v}_{k-1}, \tag{2.13}$$

se alguma informação adicional que muda o estado de $\hat{v}$ durante cada intervalo de tempo é conhecida, a equação 2.13 pode incluí-la.

A medida pode se afastar da previsão porque os efeitos do ruído não estão incluídos no modelo simplificado. Esse erro de previsão $\hat{r}$, também chamado de residual, pode ser calculado como

$$\hat{r}_k = x_k - \hat{x}_k. \tag{2.14}$$

As constantes α e β são usadas juntamente com $\hat{r}$ para corrigir, respectivamente, o valor da posição estimada e da velocidade estimada fazendo

$$\hat{x}_k = \hat{x}_k + \alpha\hat{r}_k \tag{2.15}$$

e

$$\hat{v}_k = \hat{v}_k + \left(\frac{\beta}{\Delta T}\right)\hat{r}_k. \tag{2.16}$$

Essas equações formam um relacionamento recursivo, igualmente ao KF (figura 2.4(b)), pois quando ocorre um incremento no tempo, ou seja, quando uma nova medida de posição é feita, o processo se repete. As Equações 2.12 e 2.13 são usadas na fase de previsão para prever a posição e a velocidade. Já as Equações 2.14, 2.15 e 2.16 são usadas na fase de correção.

Os valores de α e β geralmente são ajustados empiricamente baseado no modelo de mobilidade do alvo. Um bom ajuste dessas variáveis é que define se as estimativas são mais precisas que as medições diretas. Esses coeficientes substituem os coeficientes otimizados do KF.

Em redes de sensores, o ABF é bastante utilizado em aplicações de rastreamento de alvos. Wang et al. [2007a] apresentam uma abordagem de rastreamento de nós móveis em redes de sensores que formula a estimativa de localização como um problema de quadrados mínimos ponderado, que é resolvido de maneira descentralizada. No nó *sink*, essa estimativa é melhorada usando um ABF ou KF. Quando comparados, o ABF alcança resultados próximos aos do KF, apesar da baixa complexidade computacional. Além disso, ele também uma configuração simplificada do KF para ajustar os parâmetros do ABF, nesse caso os resultados das estimativas são equivalentes.

Sharma et al. [2011a] combinam arquitetura hierárquica com o ABF para avaliar o desempenho de rastreamento de alvos em redes de sensores. Eles comparam o resultado do rastreamento desse filtro com o resultado obtido com KF e concluem que, apesar da simplicidade, o ABF, com seus parâmetros corretamente ajustados, pode alcançar resultados tão bons quanto o KF.

Sharma et al. [2011b] utilizam uma abordagem de rastreamento usando ABF para guiar um veículo. Nesse tipo de aplicação, o veículo guiado é usado para interceptar o alvo, usando previsões para que a interceptação ocorra no menor tempo possível.

Dessa forma, eles utilizam as previsões do ABF para orientar o veículo guiado.

2.2.4.3 Filtro de Partículas

Os filtros de Partículas [Gustafsson, 2010] são recursivas implementações dos métodos sequenciais de Monte Carlo. Apesar do filtro de Kalman ser a abordagem clássica para estimação de estados, o filtro de Partículas representa uma alternativa para problemas não-lineares com ruídos não-Gaussianos [Doucet et al., 2001].

Ao contrário dos problemas lineares e Gaussianos, os cálculos da distribuição posterior são extremamente complexos. Para superar essa dificuldade o filtro de Partículas adota uma abordagem denominada amostragem por importância. O objetivo é estimar a densidade de probabilidade posterior. A ideia central do filtro de Partículas é representar tais densidades por um conjunto de partículas.

Esse método tenta construir uma *Probability Density Function* (PDF) baseado em um grande número de amostras aleatórias, denominadas partículas. Essas partículas são propagadas ao longo do tempo, combinando sequencialmente os passos de amostragem e reamostragem. A cada passo no tempo, a reamostragem é usada para descartar algumas partículas, aumentando a relevância das regiões de maior probabilidade. Cada partícula tem um peso associado que indica a sua qualidade. Então, a estimativa é o resultado da soma dos pesos de todas as partículas.

De acordo com Arulampalam et al. [2002], o filtro de Partículas genérico pode ser representado pelo algoritmo 2.1, em que $\{x^i_{0:k}, w^i_k\}^{N_s}_{i=1}$ denota as medidas aleatórias que caracterizam o PDF posterior $p(x_{0:k}|z_{i:k})$, sendo $\{x^i_{0:k}, i = 0, ..., N_s\}$ um conjunto de partículas com pesos associados $\{w^i_k, i = 1, ..., N_s\}$ e $x_{0:k} = \{x_j, j = 0, ..., k\}$ o conjunto de todos os estados para o tempo k.

Nesse algoritmo, as partículas são distribuídas de acordo com o modelo aplicado de forma a estimar o próximo estado, usando a distribuição PDF. Para garantir a qualidade das estimações seguintes, todas as partículas têm seu peso reavaliado baseado na medição atual. Então, para evitar o problema de degeneração (em que as partículas passam a ter pesos insignificantes após algumas iterações) é realizada a reamostragem que elimina as partículas com menores pesos. Entretanto, a reamostragem exige grande esforço computacional para atualizar todas as partículas, por isso o tamanho amostral efetivo ($\hat{Neff}$) é usado para indicar quando a degeneração justifica uma reamostragem, no caso quando ele for menor que um limiar N_T.

A reamostragem pode ser representada como no algoritmo 2.2, em que $\bigcup[0, N_s^{-1}]$ é a distribuição uniforme no intervalo $[0, N_s^{-1}]$, sendo que para cada partícula x^{j*}_k reamostrada, esse algoritmo armazena os índices dos seus ancestrais, que são denotados

Result: $[\{x_k^i, w_k^i\}_{i=1}^{N_s}] = PF[\{x_{k-1}^i, w_{k-1}^i\}_{i=1}^{N_s}, z_k]$
for $i = 1 : N_s$ **do**
  Produzir $x_k^i \sim q(x_k|x_{k-1}^i, z_k)$
  Atribuir o peso da partícula, $w_k^i \propto w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_k)}$
end
Calcular o peso total: $t = \sum_{i=1}^{N_s}(w_k^i)$
for $i = i : N_s$ **do** Normalizar: $w_k^i = t^{-1}w_k^i$
Calcular $\hat{Neff} = \frac{1}{\sum_{i=1}^{N_s}(w_k^i)^2}$
if $\hat{Neff} < N_T$ **then**
  Reamostrar: $[\{x_k^i, w_k^i, -\}_{i=1}^{N_s}] = REAMOSTRAGEM[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$
end

Algoritmo 2.1: Filtro de Partículas genérico.

por i^j, e CDF representa a função distribuição acumulada. As partículas de maior peso (maior importância) são selecionadas. De acordo com o seu peso é realizada uma nova amostragem da distribuição anterior. Assim, as partículas de maior importância dão origem a um maior número de partículas, já as partículas de menor importância desaparecem e não originam descendentes.

Result: $[\{x_k^{j*}, w_k^j, i^j\}_{j=1}^{N_s}] = REAMOSTRAGEM[\{x_k^i, w_k^i\}_{i=1}^{N_s}]$
Inicializar o CDF: $c_1 = 0$
for $i = 2 : N_s$ **do** Construir CDF: $c_i = c_{i-1} + w_k^i$
Começar na parte inferior da CDF: $i = 1$
Produzir um ponto de partida: $u_1 \sim \bigcup[0, N_s^{-1}]$
for $j = 1 : N_s$ **do**
  Mover ao longo da CDF: $u_j = u_1 + N_s^{-1}(j-1)$
  while $u_j > c_i$ **do** $i = i + 1$
  Atribuir amostra: $x_k^{j*} = x_k^i$
  Atribuir peso: $w_k^j = N_s^{-1}$
  Atribuir ancestral: $i^j = i$
end

Algoritmo 2.2: Reamostragem do Filtro de Partículas.

Arulampalam et al. [2002] descrevem algoritmos Bayesianos ótimos e sub-ótimos, como os filtros de Kalman, Kalman estendido, Partículas e outras variações. Nakamura et al. [2007] apresentam alguns métodos de estimação, entre eles o filtro de Partículas. Além disso, esse trabalho mostra aplicações do filtro de Partículas em redes de sensores.

Devido às suas diversas aplicações, o PF sofreu algumas adptações. A base para

vários filtros de Partículas é o *Sequential Importance Sampling* (SIS), sendo esse abordado por Doucet et al. [2000], Doucet et al. [2003] e Cappe et al. [2007]. Gordon et al. [1993] propõem o *Sampling Importance Resampling* (SIR). Esse método tem a vantagem de avaliar facilmente o peso das partículas e a importância das amostras, entretanto perde rapidamente a diversidade de partículas por fazer a reamostragem a cada passo. Para melhorar o SIR, Pitt & Shephard [1999] propõem o *Auxiliary Sampling Importance Resampling* (ASIR). Esse algoritmo gera pontos da amostra anterior baseado nos pontos estimados, que condicionados às medições ficam mais próximas do estado real, entretanto o mesmo não tem bom desempenho quando o ruído do processo é alto. Já Musso et al. [2001] propõem o *Regularized Particle Filter* (RPF) que modifica o processo de reamostragem do SIR para resolver o problema de diversidade de partículas, comum devido ao processo de reamostragem.

Existem trabalhos que comparam o filtro de Partículas com outros métodos de estimação. Yuen & MacDonald [2002] criam múltiplos filtros de Partículas para estimar simultaneamente as posições de robôs e de suas referências. Em suas avaliações, eles mostram que sua técnica obtém melhores resultados que o EKF. Arulampalam et al. [2002] comparam o filtro de Partículas com duas variações do EKF. As avaliações são realizadas no problema de rastreamento com manobras. A superioridade do filtro de Partículas fica evidente, principalmente quando as condições de ruídos são altas.

Visando a aplicação do filtro de Partículas em redes de sensores, alguns trabalhos propõem abordagens distribuídas desse método. O trabalho de Rosencrantz et al. [2003] desenvolve um filtro de Partículas distribuído para fusão de dados descentralizada aplicada em rastreamento. A ideia é determinar quais medições devem ser compartilhadas. Para isso cada nó sensor mantém uma instância local de filtro de Partículas. O esquema trabalha usando um sistema consulta-resposta, ou seja, cada nó consulta seus vizinhos para usar as medições. A consulta é composta de um pequeno conjunto de partículas que contêm o estado completo da trajetória. Baseado nesse conjunto de partículas, o nó consultado determina e transmite apenas as informações relevantes. Coates [2004] propõe realizar a estimativa do estado atual de múltiplos nós sensores de forma distribuída. Para isso descreve duas metodologias: a primeira abordagem é baseada em uma modelagem paramétrica, que associa uma probabilidade às partículas e as usa como dados de treinamento. Em seguida, o modelo de parâmetros é trocado entre os nós sensores em vez de informações sobre as partículas. Mas, para isso exige uma comunicação substancial. Já a segunda abordagem usa uma codificação adaptativa. Essa visa treinar a previsão em cada passo de tempo baseado em um filtro de Partículas comum mantido em todos os nós. Esse processo exige um custo computacional alto.

Sheng et al. [2005] propõem dois filtros de Partículas distribuídos para rastrear múltiplos alvos em redes de sensores sem fio. No primeiro algoritmo, resultados parciais são atualizados sequencialmente em cada clique de sensores, sendo baseados em resultados parciais encaminhados de cliques vizinhos e em observações locais. Já no segundo algoritmo, todos os cliques calculam paralelamente estimativas parciais baseados somente em observações locais, e encaminha suas estimativas para um nó *sink* onde a fusão de dados é realizada para obter o resultado final. Jiang & Ravindran [2011] propõem um filtro de partículas distribuídos para rastreamento de alvos, em que as partículas são mantidas em diferentes nós sensores e propagadas de acordo com a trajetória do alvo. Durante a propagação, o algoritmo inclui um processo de fusão de dados para minimizar o custo de comunicação.

Rastreamento de alvos é o principal problema em que o filtro de Partículas é utilizado em redes de sensores. Aslam et al. [2003] propõem um algoritmo de rastreamento baseado no filtro de Partículas. Esse explora as propriedades geométricas de uma rede composta de sensores usando um modelo de detecção binário, um *bit* representa se um alvo está se aproximando ou se afastando do sensor. Guo & Wang [2004] propõem uma nova solução chamada *Sequential Monte Carlo* (SMC) para rastreamento de alvos que faz uso do filtro de Partículas para fusão de dados e de uma representação reduzida da distribuição posterior para diminuir a quantidade de dados transmitidos entre os nós sensores. Visando o rastreamento de múltiplos alvos.

Vercauteren et al. [2005] propõem uma solução colaborativa baseada na metodologia SMC para rastrear conjuntamente vários alvos e classificá-los de acordo com o seu padrão de movimento. Hu & Evans [2004] usam o filtro de Partículas para obter a localização dos nós de uma rede composta por nós móveis. Entretanto, a solução trabalha como uma solução de rastreamento aplicada para todos os nós. Esse trabalho ainda mostra que a mobilidade pode melhorar a precisão e reduzir o custo da localização. Li & Xu [2010] propõem um filtro de partículas otimizado para rastreamento em que a trajetória do alvo possui manobras bruscas. Para isso, eles usam uma faixa de direção do movimento do alvo para diminuir a quantidade de partículas que desviam da trajetória, reduzindo a degradação de partículas.

2.2.5 Ativação

O componente de ativação é responsável por ativar e desativar o rádio dos nós sensores para conservar energia [Zahedi et al., 2010; Deldar & Yaghmaee, 2011; Hsu et al., 2012]. O nó está em modo ativo quando o rádio está ligado e inativo caso contrário. Economia de energia é o fator mais importante em redes de sensores, uma vez que os

nós sensores são suportados por baterias, que são inviáveis de serem recarregadas ou substituídas [Yan & Wang, 2011].

O radio dos nós tem quatro modos de operação: transmitindo, recebendo, ouvindo (o rádio está ligado, mas não está transmitindo nem recebendo) e dormindo (o rádio está desligado) [Schurgers, 2008]. Quando o rádio está ligado, o consumo de energia não varia significativamente. Então o único caminho para reduzir o consumo de energia é desligar o rádio [Miller & Vaidya, 2004].

Em aplicações de rastreamento, os nós gastam grande parte do tempo ativados sem efetivamente participar da comunicação. Nesse caso, o rádio pode ser desligado para economizar energia. Estratégias cooperativas usam previsão para escolher um conjunto de nós para rastrear o alvo, enquanto os outros nós ficam inativos [Yan & Wang, 2011]. Dessa forma, o mecanismo de ativação tem a função de fazer com que o conjunto de nós atualmente ativo ative um conjunto de nós adjacentes antes do alvo alcance essa área [Jin et al., 2006].

Duas estratégias são usadas para gerenciar a atividade do rádio: *duty cycle* [Yan & Wang, 2011; Zahedi et al., 2010]; e *paging channel* [Xu et al., 2004a; Bhuiyan et al., 2010]. O *duty cycle* define janelas de tempo em que os nós permanecem ativos. Os nós sensores alternam entre os estados ativo e inativo baseados na atividade da rede. Um longo *duty cycle* fazem os nós terem mais tempo para transmitir dados, já o baixo faz os sensores economizarem mais bateria [Hsu & Wu, 2008]. O *paging channel* permite que os nós ativos acordem os nós que estão dormindo. Para isso, os nós são equipados com dois rádios – o rádio principal para transmissão de dados, e o rádio de paginação para acordar os nós que estão com o rádio principal desativado. Esses rádios operam em frequências diferentes para evitar interferências [Schurgers et al., 2002].

Hsu & Wu [2008] propõem um *duty cycle* ajustado para conservar a energia em nós com o tráfego de dados baixo e reduzir a latência em nós com o tráfego de dados pesados, em que cada nó sensor têm diferentes tempos de ativação e desativação. Yan & Wang [2011] usam *duty cycle* e agrupamentos dinâmicos para estender o tempo de vida da rede de sensores em aplicações de rastreamento. Zahedi et al. [2010] analisam o impacto de diferentes mecanismos de *duty cycle* em rastreamento, variando os parâmetros de *duty cycle* para avaliar sua influência na precisão de latência.

Xu et al. [2004a], Lin et al. [2009a], Wang et al. [2010a], Hong et al. [2010], Bhuiyan et al. [2010], Deldar & Yaghmaee [2011] e Hsu et al. [2012] usam *paging channel* para ativar o rádio dos nós com uma mensagem de ativação. Essas abordagens preveem a futura posição do alvo e a usam para decidir quais nós precisam ser ativados para continuar o rastreamento.

2.2.6 Recuperação

O componente de recuperação visa encontrar um alvo perdido, uma vez que falhas podem tornar o alvo temporariamente indetectável. O desafio é encontrar o alvo de forma precisa e rápida usando a menor quantidade possível de mensagens [Khare & Sivalingam, 2011].

Como dito anteriormente, a previsão é usada para escolher o conjunto de nós que permanece ativo e rastreia o alvo, enquanto o restante permanece inativo para economizar energia e aumentar o tempo de vida da rede. Entretanto, não existe garantia que o conjunto correto de sensores será ativado, causando a perda do alvo [Khare & Sivalingam, 2011; Gupta et al., 2012]. Nesse cenário, um mecanismo de recuperação necessário já que não é possível evitar a perda do alvo [Hsu et al., 2012].

A qualidade da previsão é responsável pela perda do alvo. Existem muitas razões que afetam a qualidade da previsão para causar a perda do alvo. Uma dessas razões é o erro de localização dos nós, uma vez que essa localização é usada para calcular e prever a posição do alvo [Souza et al., 2009; Campos et al., 2012]. Outra razão é a falha dos nós, devido à falha de *hardware*, falha na comunicação ou falta de energia, que faz com que não exista dados suficientes para calcular a posição do alvo com precisão. Mesmo se a rede estiver operando em condições ideais, uma mudança brusca na trajetória do alvo pode ocasionar uma previsão pouco precisa [Gupta et al., 2012].

O mecanismo de recuperação precisa precisa ativar um conjunto de nós maior que o conjunto convencional (sensores ativados com base na previsão) para encontrar o alvo. Uma solução simples e segura é ativar todos os nós da rede, mas isso tem um custo muito alto. Portanto, os mecanismos de recuperação são elaborados para acordar a quantidade mínima de nós suficiente para encontrar o alvo.

Yang & Sikdar [2003] e Xu et al. [2004b] ativam alguns nós ao redor da área estimada na previsão para encontrar o alvo perdido. Se o alvo não é encontrado, um *flooding* é iniciado e todos os nós são ativados. Evidentemente, esse mecanismo de recuperação causa grande consumo de energia devido ao aumento de nós ativos. Lalooses et al. [2007] propõem um mecanismo de recuperação baseado em lugares populares para aplicações de rastreamento de animais. Se o alvo é perdido, ele checa a presença do animal em lugares que ele costuma frequentar para conseguir água, se abrigar ou descansar.

Khare & Sivalingam [2011] propõem um mecanismo de recuperação para encontrar alvos perdidos em redes organizadas em agrupamentos. Esse mecanismo é dividido em quatro fases. A primeira fase é responsável por definir se o alvo foi perdido. Se o CH não detecta o alvo no tempo estipulado, ele conclui que o alvo foi perdido. A

segunda fase existe para evitar que a recuperação seja iniciada erroneamente. Dois ou mais agrupamentos podem estar ativados, então eles consultam uns aos outros para garantir que algum deles detectou o alvo, caso contrário a próxima fase é disparada. Na terceira fase, o agrupamento que detectou a perda do alvo ativa os seus agrupamentos vizinhos. Caso o alvo ainda não seja encontrado, esses outros agrupamentos também ativam seus vizinhos. Esse processo se repete até que o alvo seja encontrado. Na última fase, depois que o alvo é encontrado, os agrupamentos que estão participando do rastreamento permanecem ativos enquanto o restante é desativado. Gupta et al. [2012] propõem um mecanismo de recuperação similar ao proposto por Khare & Sivalingam [2011], mas usam as previsões do KF para escolher os agrupamentos que devem ser ativados.

Ji et al. [2009] e Hsu et al. [2012] propõem um mecanismo de recuperação para redes organizadas em faces. Em Ji et al. [2009], quando o alvo é perdido, o último nó que detectou o evento acorda todas as suas faces adjacentes. Se ainda assim o alvo não é encontrado, uma mensagem é enviada ao *sink* e o rastreamento é reiniciado. Hsu et al. [2012] propõem dois mecanismos de recuperação que podem ser adotados de acordo com o os requisitos da aplicação. O primeiro é projetado para encontrar o alvo de forma rápida. Ele ativa todos os nós cujas distâncias para a posição do alvo são menores que a maior distância obtida usando a velocidade máxima do movimento. A segunda abordagem usa a menor quantidade de comunicações para encontrar o alvo usando a direção do movimento. Ele busca sequencialmente pelo alvo de acordo com a probabilidade de ele estar em determinada face.

Alguns algoritmos de rastreamento perdem o alvo, mas não implementam nenhum mecanismo de recuperação. Em vez disso, usam alternativas simples para evitar que o alvo seja perdido. Kung & Vlah [2003] e Liu [2010] constroem árvores de alta densidade para reduzir a probabilidade do alvo ser perdido. Já Tsai et al. [2007] também diminui essa probabilidade aumentando a frequência com que os nós são ativados.

2.3 Métricas

As métricas consideras para analisar algoritmos de rastreamento de alvos dependem dos objetivos da aplicação. Métricas como precisão, atraso, custo de comunicação e consumo de energia são fundamentais em algoritmos de rastreamento.

2.3.1 Precisão

A precisão dos algoritmos de rastreamento mede o quanto a posição estimada se diferen da posição real do alvo [Campos et al., 2012; Souza et al., 2013]. Essa métrica também é usada para as previsões de posição do alvo. Nesse caso, a posição prevista é comparada com a posição do alvo no tempo futuro referente à previsão [Bhuiyan et al., 2010; Sharma et al., 2011a; Hajiaghajani et al., 2012].

A precisão da previsão de posição pode ter relação direta com a energia consumida pela rede durante o rastreamento, pois o mecanismo de ativação usa as previsões para escolher os sensores que serão ativados para rastrear o alvo. Uma previsão imprecisa pode ativar o conjunto de sensores errados e perder o alvo. Dessa forma, será necessário o uso de mais mensagens para recuperar o alvo [Xu et al., 2004b; Bhuiyan et al., 2010; Hsu et al., 2012].

2.3.2 Atraso

O atraso é uma métrica importante para aplicações de rastreamento que precisão conhecer a trajetória em tempo real. Ela representa o tempo que se leva para estimar a posição do alvo depois que o mesmo é detectado. Além disso, o tempo necessário para enviar o resultado até o nó *sink* também pode ser adicionado.

Existem vários fatores que podem causar atrasos no rastreamento. Para evitar colisão de pacotes na rede, os nós usualmente aguardam um pequeno tempo aleatório antes de enviar uma mensagem. Em cada salto, esses atrasos são acumulados, representando um atraso significativo até alcançar o *sink* [Sharma et al., 2011b]. Obstáculos ou buracos na rede causados por nós sem energia ou terrenos acidentados aumentam o atraso porque aumenta a quantidade de saltos necessária para alcançar o *sink* [Olule et al., 2007].

Em algoritmos de rastreamento baseados em consulta [Kung & Vlah, 2003; Lin et al., 2006; Liu et al., 2008; Yeong-Sung et al., 2010], o atraso é definido com o tempo necessário para o resultado sobre a presença do alvo alcançar o *sink* a partir do sensor que está detectando o alvo. Nessa formulação, o pior caso de atraso ocorre em árvores degeneradas, para reduzir o atraso é preciso manter a árvore balanceada.

Quando o alvo é perdido, ele deve ser reencontrado o quando antes. O atraso, nesse caso, pode ser definido como o tempo necessário para achar o alvo depois que o mecanismo de recuperação é disparado [Hsu et al., 2012].

Para criar ou atualizar um agrupamento dinâmico, os nós precisam esperar um tempo aleatório para obter informações dos outros nós para eleger o líder do agrupamento [Jin et al., 2006; Lee et al., 2007; Hamouda & Phillips, 2011; Park et al., 2010;

Hajiaghajani et al., 2012]. Em outro caso, quando o alvo se move para o limite entre agrupamentos, os nós que detectam o alvo relatam seus CH independentemente. Os CHs divulgam as informações coletadas de seu agrupamento com cada um dos outros CHs para realizar a estimativa de posição. Todo esse processo leva tempo que deve ser contabilizado como atraso [Wang et al., 2010b].

2.3.3 Custo de Comunicação

O custo de comunicação a quantidade de mensagens geradas pela rede durante a execução da aplicação. Algoritmos de rastreamento visam manter a qualidade do rastreamento e reduzir o custo de comunicação.

Vários fatores podem gerar sobrecarga de comunicação, como a estrutura da rede, precisão dos dados, frequência de reporte de resultado e a quantidade de alvos. Para evitar que todos os sensores que detectam o alvo enviem seus dados para o *sink*, os algoritmos de rastreamento organizam a estrutura lógica da rede (como uma árvore [Yeong-Sung et al., 2010], agrupamentos [Hajiaghajani et al., 2012] ou faces [Hsu et al., 2012]) para possibilitar a fusão dos dados e reduzir o custo de comunicação.

Uma alta precisão dos dados requer mais dados coletados, gerando mais comunicação. Se a aplicação precisar de atualizações frequentes sobre a situação do alvo, ela pode ser configurada com uma alta taxa de reporte de resultados. Por outro lado, uma baixa taxa de reporte reduz o número de comunicações. Nesses casos, o equilíbrio entre a qualidade do rastreamento e o custo de comunicação deve ser avaliado [Xu et al., 2004b].

A quantidade de alvos que estão sendo rastreamento aumenta significativamente o tráfego da rede. Assim, para reduzir o custo quando mais alvos estão em uma mesma área, os sensores podem unir os dados sobre os alvos em uma mesma mensagem [Hamouda & Phillips, 2011; Hajiaghajani et al., 2012].

2.3.4 Consumo de Energia

O consumo de energia é um fator chave para qualquer aplicação em redes de sensores, uma vez que os nós sensores tem restrição de energia. Essa métrica representa a quantidade de energia usada pela rede para executar o rastreamento [Deldar & Yaghmaee, 2011; Hamouda & Phillips, 2011].

Um nó é tipicamente equipado com múltiplos sensores, bateria, unidade de processamento e um módulo de comunicação para transmitir e receber mensagens. Quando a energia do nó termina, ele se torna inutilizável. A substituição da bateria é inviável

dependendo do local onde a rede foi implantada. Para aumentar o tempo de vida da rede é necessário conservar energia enquanto a aplicação é executada [Yan & Wang, 2011].

O módulo de comunicação é responsável pela maioria da energia consumida por um nó. Portanto, os algoritmos de rastreamento usam fusão de dados para reduzir a quantidade de mensagens transmitidas [Nakamura et al., 2007]. Esses algoritmos também usam mecanismos de ativação combinados com previsão para ativar somente o conjunto de sensores necessário para rastrear o alvo, enquanto o restante fica inativo para economizar energia [Zahedi et al., 2010; Bhuiyan et al., 2010]. Por outro lado, os erros de previsão podem causar a perda do alvo e para encontrá-lo é necessário ativar mais nós e usar mensagens extras, aumentando o consumo de energia. Dessa forma, a precisão das previsões são importantes para aumentar o tempo de vida da rede [Khare & Sivalingam, 2011; Hsu et al., 2012; Gupta et al., 2012].

2.4 Requisitos das Aplicações de Rastreamento

Redes de sensores são usadas para monitorar uma área de interesse. Essas redes tem muitas aplicabilidades que formam o objetivo principal dessas redes. Assim, aplicação de rastreamento é um exemplo dessas aplicações.

Para que a aplicação de rastreamento seja executada de forma satisfatória, alguns objetivos secundários, ou requisitos, devem ser satisfeitos. Entres outros requisitos, a rede deve: estabelecer um sistema de localização; possibilitar o roteamento dos dados coletados; controlar a densidade da rede; sincronizar o relógio dos nós sensores; e prover segurança para a comunicação. A figura 2.5 ilustra esses requisitos.

Para focar somente no rastreamento, muitos trabalhos assumem que esses requisitos já estão satisfeitos. Entretanto, é importante avaliar a integração desses requisitos com o rastreamento, pois eles podem gerar erros que afetam o resultado final do rastreamento [Souza et al., 2009; Campos et al., 2012].

2.4.1 Sistema de Localização

Um sistema de localização possibilita que os nós passem a conhecer suas posições no campo de sensores. Em muitas soluções de localização existe uma fração dos nós que conhecem a sua localização a priori (usando *Global Positioning System* (GPS) [Goswami, 2013] ou posicionamento manual), na área de localização esses nós são chamados de

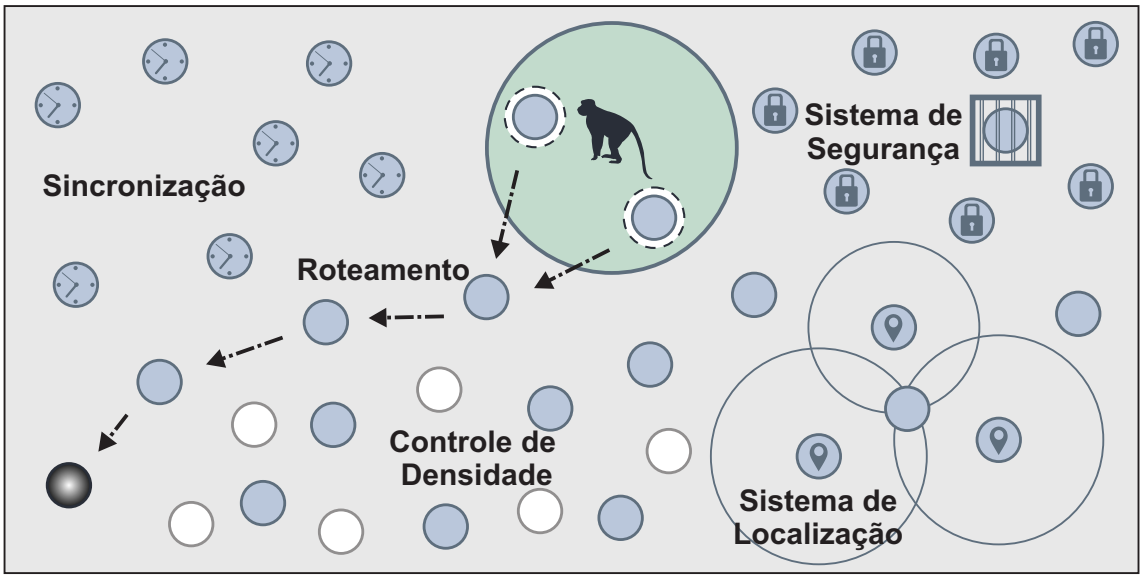


Figura 2.5. Requisitos das aplicações de rastreamento em redes de sensores. Sistem de Localização: provê a localização dos nós por compartilhar a posição de uma fração dos nós que conhecem sua posição a priori; Roteamento: define rotas apropriadas para enviar os dados até o *sink*; Controle de Densidade: em uma rede de alta densidade, mantém ativado somente a quantidade necessária de nós para efetuar o rastreamento; Sincronização: fornece uma escala de tempo comum para o relógio local dos nós, uma vez que as condições do ambiente causam variações nesses relógios; Sistema de Segurança: verifica o conteúdo das mensagens e isola nós maliciosos para garantir integridade e confiabilidade à aplicação.

beacons[1]. Esses algoritmos compartilham a localização dos *beacons* para estimar a localização dos outros nós da rede, mas essa estimativa não é perfeita [Boukerche et al., 2007].

Dois importante sistemas de localização são o RPE [Albowicz et al., 2001] e o DPE [Oliveira et al., 2009]. O RPE é a solução iterativa pioneira. Nesse algoritmo, os nós *beacons* divulgam suas localizações para ajudar outros nós a se localizarem. Os nós que se localizam se tornam referências e também podem ajudar os nós restantes a se localizarem, dessa forma o número de referências aumenta iterativamente. O DPE é uma solução baseada no RPE, mas reduz os erros e o custo. Esse algoritmo usa uma estrutura de *beacons* para fazer a recursão iniciar de um único ponto e seguir uma direção conhecida. Com isso, um nó pode estimar sua localização usando apenas duas referências.

Aplicações de rastreamento dependem da localização dos nós, pois usam essas localizações como referências para calcular a posição do alvo. Entretanto, os erros de

[1]O termo *beacon* é utilizado tanto em localização como em rastreamento. Em localização, o *beacon* é o nó que está previamente localizado e inicia o processo de localização. Em rastreamento com objeto guiado, o *beacon* é o nó que orienta o objeto guiado até o alvo.

localização gerados pelos sistemas de localização tem influência negativa no desempenho de rastreamento. Souza et al. [2009], Souza et al. [2011a] e Campos et al. [2012] avaliam o impacto de algoritmos de localização reais nos algoritmos de rastreamento com os filtros de Kalman e Partículas.

2.4.2 Roteamento

Em redes de sensores, algoritmos de rastreamento definem caminhos que ligam os nós sensores com o nó *sink* através de múltiplos saltos, usando o menor número possível de comunicações. Assim, esses caminhos são usados para enviar o resultado da aplicação para o *sink* [Al-Karaki & Kamal, 2004]. Em aplicações de rastreamento, o roteamento é fundamental, pois seleciona um caminho apropriado para transmitir a posição do alvo em direção ao sink [Sharma et al., 2011b; Hsu et al., 2012].

Baseado na estrutura da rede, os algoritmos de roteamento são classificados em três categorias: plano [Intanagonwiwat et al., 2003], hierárquico [Yektaparast et al., 2012] e geográfico [Zhou et al., 2010]. No roteamento plano, todos os nós têm a mesma informação sobre o estado da rede e o mesmo papel na coleta de dados. No roteamento hierárquico, os nós com mais energia podem ser usados para processar e enviar os dados, enquanto os nós com menos energia são usados somente para coletar dados. Esse roteamento é uma forma eficiente de reduzir o consumo de energia, pois realiza fusão de dados para reduzir a quantidade de mensagens enviadas até o nó *sink*. Por fim, no roteamento geográfico, a localização dos nós são exploradas na criação das rotas. Nesse caso, as mensagens são encaminhadas de maneira gulosa, i.e. um nó encaminha a mensagem para o seu vizinho que está mais próximo do destino final.

2.4.3 Controle de Densidade

Controle de densidade consiste de desligar a maior quantidade possível de sensores pelo maior tempo possível e manter as funcionalidades da rede. A densidade da rede é controlada para aumentar o seu tempo de vida, mantendo ativado somente a quantidade de sensores necessária para a aplicação. Se todos os nós de uma rede de alta densidade estão em atividade simultaneamente, os dados coletados são altamente correlacionados e redundantes, consumindo energia excessiva sem necessidade [Jia et al., 2012].

Shang & Shi [2005] propõem três algoritmos de controle de densidade considerando o equilíbrio entre energia e cobertura. Esses algoritmos maximizam a cobertura, evitando sobreposição das áreas monitoradas tomando como base a localização, distância e ajuste do raio de sensoriamento. Kapnadak & Coyle [2011] mostram um algoritmo

ótimo para controle de densidade para detecção de alvos no campo de sensores.

Campos et al. [2012] avaliam como diferentes combinação de algoritmos de controle de densidade e sistemas de localização afetam as aplicações de rastreamento. Para isso, os algoritmos de controle de densidade *Optimal Geographical Density Control* (OGDC), o *Geography-informed energy conservation for Ad Hoc routing* (GAF) e o A3 são adaptados para forma mais usual do problema de cobertura-k (cada ponto no campo de sensores está coberto por pelo menos k sensores).

2.4.4 Sincronização

A sincronização fornece uma escala de tempo comum aos relógios locais dos nós sensores da rede [Lasassmeh & Conrad, 2010]. O relógio de um nó sensor possui um oscilados e um contador. Baseado na frequência angulas do oscilador, o contador aumenta seu valor para representar o tempo local. Em condições ideais, essa frequência angular é constante, mas devido às variações físicas (e.g. temperatura e pressão), ela muda e causa distorções no tempo marcado no relógio do nó. Em outras palavras, na prática, cada nó da rede marca um tempo diferente [Yik-Chung et al., 2011].

Em redes de sensores, a sincronização do tempo é necessária para coordenar tarefas que exigem cooperação entre os nós. Fusão de dados [Nakamura et al., 2007] é um exemplo de tarefa em que a sincronização é importante, pois possibilita agrupar os dados referentes ao mesmo evento. Em aplicações de rastreamento, os nós divulgam suas localizações e tempo para identificar e estimar a posição e velocidade do alvo detectado [Merhi et al., 2009; Dixiao et al., 2010; Bhuiyan et al., 2010]. Assim, se os nós sensores não estão sincronizados, o rastreamento é afetado. Entretanto, existem algumas abordagens que não requerem que os nós estejam sincronizados para executar o rastreamento [Wang et al., 2010a; Lee et al., 2011]. A sincronização do tempo também é usada para economizar a energia dos nós [Hsu & Wu, 2008; Zahedi et al., 2010], uma vez que os nós podem ser desativados em um tempo determinado e ativados quando necessário.

Zhou et al. [2012] melhoram a precisão e o consumo de energia em rastreamento de múltiplos alvos sincronizando os sensores envolvidos no rastreamento. Pashazadeh & Sharifi [2008] avaliam o impacto dos erros de sincronização em aplicações rastreamento. Wang et al. [2010a] e Lee et al. [2011] propõem uma abordagem de rastreamento que opera independente do tempo marcado pelos nós.

2.4.5 Sistema de Segurança

O sistema de segurança visa prover confidencialidade, integridade e autenticidade de todas as mensagens. Em redes de sensores, cada nó sensor deve ser capaz de verificar a integridade de todas as mensagens destinadas a ele, bem como a identidade do remetente. Intrusos não deve ser capazes de ler o conteúdo das mensagens. Garantir a segurança e privacidade para os nós sensores é um desafio, pois é necessário executar várias operações para criptografia, que são difíceis para os nós sensores, devido as suas limitações da computação, comunicação, memória e energia [Du & Chen, 2008].

Segurança é fundamental para rastreamento aplicado nas áreas militares e de vigilância. A localização e a sincronização dos nós são importantes para identificar o alvo e para calcular a sua posição, mas são vulneráveis a vários tipos de ataque. Por exemplo, antes do processo de localização, um nó malicioso pode comprometer um nó *beacon* para divulgar uma localização falsa [Iqbal & Murshed, 2010]. Em outro caso, durante a sincronização, um nó malicioso pode modificar as mensagens de tempo para prejudicar ou interromper o processo de sincronização [Liu et al., 2012].

Para defender as aplicações de rastreamento de ataques no sistema de localização, Chang et al. [2007] usam um algoritmo de rotulagem para detectar nós maliciosos e remover seus dados dos cálculos de rastreamento. Já Mansouri & Khoukhi [2011] propõem uma abordagem para estimar a posição do alvo detectando os nós maliciosos e selecionando um grupo apropriado de nós que participara da coleta de dados.

2.5 Classificação dos Algoritmos de Rastreamento

Nós encontramos características comuns entre alguns algoritmos de rastreamento e os classificamos de acordo com elas. A estrutura da rede, usada na cooperação entre os nós sensores, é uma característica chave encontrada em todos os algoritmos.

Outra importante característica é como o problema de rastreamento é formulado, por exemplo um veículo guiado pode ser adicionado no contexto para interceptar o alvo, tal detalhe altera significativamente o funcionamento do algoritmo. Além disso, características relacionadas com o alvo rastreado, como o tipo e a quantidade de alvos rastreados, também são consideradas. Abaixo descrevemos brevemente os algoritmos de rastreamento usados na classificação.

O *Drain-And-Balance* (DAB) [Kung & Vlah, 2003] e *Deviation Avoidance Tree* (DAT) [Lin et al., 2006] constroem árvores com poda por mensagens (*message-pruning trees*) para atualizar a rede para atualizar a rede sobre a presença de múltiplos alvos e para possibilitar uma pesquisa de baixo custo. Liu et al. [2008] propõem uma nova

estrutura para fusão de dados que cria atalhos em árvores com poda por mensagem. Yeong-Sung et al. [2010] aborda essa questão como um problema de otimização para construir uma árvore com custos mínimos de atualização e consulta.

O DOT [Tsai et al., 2007], PET [Bhuiyan et al., 2010], e POOT [Hsu et al., 2012] fornecem a posição do alvo para o objeto guiado, então ele pode se mover em direção ao alvo. Sharma et al. [2011b] resolve o mesmo problema considerando que o nó *sink* se comunica diretamente com o objeto guiado usando um rádio de alta potência. O *Dynamic Convoy Tree-Based Collaboration* (DCTC) [Zhang & Cao, 2004a] encontra uma árvore de alta cobertura com baixo consumo de energia usando um esquema baseado em previsão para expandir e podar a árvore. No OCO [Tran & Yang, 2006], o *sink* coleta a posição de todos os nós da rede e aplica técnicas de processamento de imagem para desativar nós redundantes e encontrar o menor caminho entre cada nó e o *sink*.

O *Continuos Object Detection and tracking Algorithm* (CODA) [Chang et al., 2008], *Two-tier Grid based Continuous Object Detection and tracking* (TG-COD) [Park et al., 2010] e *PREdictive Continuous Object tracking* (PRECO) [Hong et al., 2010] determinam o conjunto de nós cuja localização represente os limites de um objeto contínuo.

O DELTA [Walchli et al., 2007] e o algoritmo proposto por Jin et al. [2006] formam agrupamentos dinamicamente de acordo com o caminho percorrido pelo alvo. No *Hybrid Cluster-based Target Tracking* (HCTT) [Wang et al., 2010b] e *Hybrid Clustering for Multi-Target Tracking* (HCMTT) [Hajiaghajani et al., 2012] a tarefa de rastreamento alterna entre agrupamentos estáticos e dinâmicos. O agrupamento dinâmico é usado quando o alvo se aproxima dos limites entre agrupamentos, uma vez que a colaboração não é completa porque os sensores pertencem à agrupamentos diferentes.

Lee et al. [2007] propõem um algoritmo com agrupamento dinâmico que minimiza a área sobreposta entre dois agrupamentos para reduzir dados duplicados. O *Reduced Area REporting* (RARE) [Olule et al., 2007] e o *Adaptive Dynamic Cluster-based Tracking* (ADCT) [Yang et al., 2007] escolhe alguns nós que participam do rastreamento para minimizar a sobrecarga de comunicação e manter a qualidade do rastreamento. Eles garantem que os nós com dados redundantes não participem do rastreamento. No *Distributed Multi-sensor Multi-target Tracking* (DMMT) [Hamouda & Phillips, 2011], os nós sensores que detectam mais do que um alvo ao mesmo tempo determinam o nós preferenciais baseado na potência do sinal.

O *Prediction-based Energy Saving* (PES) Xu et al. [2004b] usa os componentes de previsão, ativação e recuperação para ativar somente os nós sensores necessário para o rastreamento. Ele avalia a influência da frequência de amostragem na precisão

das previsões e consumo de energia. O *Face-based Object Tracking Protocol* (FOTP) [Ji et al., 2009] combina um algoritmo de hexágono com estrutura de faces. Dentre todos os nós que detectam o alvo, a previsão do alvo é prevista pelo nós mais próximo dele. Sharma et al. [2011a] combinam estrutura baseada em agrupamentos e ABF para avaliar o seu desempenho em rastreamento de alvos em redes de sensores. Kumar & Sivalingam [2012] evitam a comunicação excessiva devido à divulgação da mobilidade do alvo e os vários parâmetros de filtros complexos, tais como KF e PF. Para isso, eles usam nós direcionais com feixe eletrônico para o rastreamento.

A tabela 2.2 mostra os algoritmos estudados e resume a classificação. Além disso, ela mostra os componentes adicionais que são aplicados em cada algoritmo. Abaixo, nós exploramos com mais detalhes a classificação dos algoritmos apresentados.

2.5.1 Estrutura da Rede

A característica chave dos algoritmos de rastreamento é a estrutura usada para organizar a rede e alcançar a cooperação entre os nós. Assim, classificamos os algoritmos de acordo com a sua estrutura lógica.

2.5.1.1 Estrutura de Árvore

O OCO usa uma árvore simples, o *sink* inicia um *flooding* para que os nós da rede encontrem o menor caminho até ele. Quando ocorre alguma mudança na topologia da rede, a árvore pode ser reconfigurada.

O DCTC é uma estrutura de árvore dinâmica para beneficiar a colaborações entre os múltiplos sensores ao redor do alvo. Quando o alvo é detectado pela primeira vez, a árvore inicial é construída. O nó mais próximo do alvo geralmente é a raiz. Alguns nós sensores são adicionados à árvore a medida que o alvo se aproxima deles. Alguns nós também são removidos da árvore quando o alvo se afasta deles.

No DAB, DAT e DAT+Shortcut, a rede de sensores é considerada um banco de dados distribuído, pois usa hierarquia para gravar dados sobre a presença do alvo. A árvore de poda por mensagem é construída para conectar todos os nós sensores e reduzir a quantidade de transmissões redundantes. Nesse caso, o rastreamento funciona com atualização e consulta. Quando o alvo vai do raio de sensoriamento de um nó para outro, o nó do qual o alvo se afastou registra sua ausência, já o nó que está atualmente detectando o alvo registra sua presença. Quando a consulta por um alvo é solicitada, ela percorre um caminho específico da árvore para achar o nó mais próximo do alvo. Assim, o resultado do rastreamento só é enviado ao *sink* sob demanda. Essas

Tabela 2.2. Resumo da classificação dos algoritmos de rastreamento.

Estrutura		Método	Comp. Adicionais			Alvo		Formulação
			Prev.	Ativ.	Recup.	Multi	Tipo	
Árvore		DAB – Kung & Vlah [2003]	não	não	não	sim	simples	consulta
		DCTC – Zhang & Cao [2004a]	sim	sim	não	não	simples	clássica
		DAT – Lin et al. [2006]	não	não	não	sim	simples	consulta
		OCO – Tran & Yang [2006]	não	sim	sim	sim	simples	clássica
		DAT+Shortcut – Liu et al. [2008]	não	não	não	sim	simples	consulta
		Yeong-Sung et al. [2010]	não	não	não	sim	simples	consulta
Agrupamento	Estático	PES – Xu et al. [2004b]	sim	sim	sim	sim	simples	clássica
		RARE – Olule et al. [2007]	não	não	não	não	simples	clássica
		Sharma et al. [2011a]	sim	sim	não	não	simples	clássica
		Sharma et al. [2011b]	sim	sim	não	não	simples	clássica
		Kumar & Sivalingam [2012]	sim	sim	não	não	simples	guiada
	Dinâmico	Jin et al. [2006]	sim	sim	sim	não	simples	clássica
		Lee et al. [2007]	sim	sim	não	não	simples	clássica
		ADCT – Yang et al. [2007]	sim	sim	sim	não	simples	clássica
		DELTA – Walchli et al. [2007]	não	não	não	não	simples	clássica
		PRECO – Hong et al. [2010]	sim	sim	não	não	contínuo	clássica
		DMMT – Hamouda & Phillips [2011]	sim	sim	sim	sim	simples	clássica
	Híbrido	CODA – Chang et al. [2008]	não	não	não	sim	contínuo	clássica
		HCTT – Wang et al. [2010b]	sim	sim	sim	não	simples	clássica
		TG-COD – Park et al. [2010]	sim	não	não	não	contínuo	clássica
		HCMTT – Hajiaghajani et al. [2012]	não	sim	não	sim	simples	clássica
Face		DOT – Tsai et al. [2007]	não	sim	não	não	simples	guiada
		FOTP – Ji et al. [2009]	sim	sim	sim	não	simples	clássica
		PET – Bhuiyan et al. [2010]	sim	sim	sim	não	simples	guiada
		POOT – Hsu et al. [2012]	sim	sim	sim	não	simples	guiada

abordagens visam equilibrar os custos de atualização e consulta para reduzir o custo de comunicação geral.

No DAB a hierarquia da árvore é baseada nos padrões de movimento do alvo, como a frequência de movimentos sobre uma determinada região. Nessa árvore, as arestas podem corresponder a múltiplos saltos entre nós, logo não reflete a estrutura física da rede. Já a árvore construída pelo DAT considera a estrutura física da rede e os custos de atualização e consulta para construir a árvore. Liu et al. [2008] adiciona atalhos na árvore construída pelo DAT, dessa forma reduz os custos de consulta. Yeong-Sung et al. [2010] desenvolvem uma heurística para construir a árvore para rastreamento com um custo de comunicação mínimo.

2.5.1.2 Estrutura de Agrupamentos

Vários trabalhos de rastreamento forma agrupamentos estáticos no momento em que a rede é implantada. Nas avaliações do PES, a estrutura da rede está fora do escopo. Dessa forma, esse assume que um nó é uma representação lógica de um conjunto de nós sensores que rastreiam o alvo, i.e. cada nó opera como um CH de um agrupamento. O RARE aumenta o tempo de vida da rede reduzindo o número de nós que participam do rastreamento. Em suas avaliações, agrupamentos estáticos são criados com CHs fixos para mensurar a economia de energia dos mesmos. A abordagem proposta por Kumar & Sivalingam [2012] usa uma estrutura de rede baseada em agrupamentos estáticos em que os nós são associados estatisticamente associados a um CH. Os trabalhos de Sharma et al. [2011a] e Sharma et al. [2011b] formam agrupamentos estáticos em que os CHs formam uma árvore que tem o nó *sink* como raiz.

Agrupamentos dinâmicos são formados de acordo com a presença do alvo. No ADCT, os nós que detectam o alvo e enviam para seus nós vizinhos uma mensagem contendo as distâncias dos nós para o alvo. Um nó se torna candidato a CH se ele não receber mensagens com uma distância menor que a sua. Esses candidatos trocam mensagens entre si para definir quem é o líder. Os CMs mudam quando o alvo se move e a previsão é usada para escolher o novo CH. O DELTA mantém agrupamentos formados dinamicamente ao redor do alvo assim que ele é detectado. Um algoritmo de eleição que se baseia nas medidas determina quem deve ser o CH. Nos agrupamentos do DMMT, um nó atua como o nó principal e é responsável por refazer o agrupamento se necessário. O PRECO constrói agrupamentos dinâmicos selecionando os nós próximos dos limites de um alvo contínuo. Lee et al. [2007] propõem um agrupamento dinâmico para reduzir a transmissão de dados redundantes causados pela sobreposição de áreas entre agrupamentos usando os valores de energia e distância. Jin et al. [2006] propõem

um mecanismo de agrupamento dinâmico para rastreamento, formando agrupamentos de acordo com a rota do alvo.

Agrupamentos híbridos são formados usando a vantagem dos agrupamentos estáticos e dinâmicos. O CODA divide a rede em agrupamentos estáticos. Os nós que detectam um alvo contínuo transmitem uma mensagem de detecção para seus CHs. Com essa informação, os CHs executam uma função para estimar os sensores que detectam a parte mais externa do objeto contínuo. Os sensores de borda de diferentes agrupamentos estáticos são unidos, formando o agrupamento dinâmico que calcula e envia o resultado do rastreamento para o nó *sink*. O TG-COD constrói agrupamentos estáticos assim que a rede é implantada, dividindo a rede em células de alta granularidade. O agrupamento dinâmico é construído somente quando um alvo contínuo aparece. Nesse caso, uma grade de baixa granularidade é construído sobre a grade já construída. No HCTT e HCMTT os nós que estão nos limites dos agrupamentos estáticos contribuem para a formação do agrupamento dinâmico. A tarefa de rastreamento alterna entre agrupamentos estáticos e dinâmicos quando o alvo alcança os limites de agrupamentos estáticos.

2.5.1.3 Estrutura de Faces

Os algoritmos DOT, FOTP, PET e POOT usam estrutura de face para distinguir diferentes áreas nas redes de sensores e reduzir a quantidade de faces ativas para reduzir a comunicação e o consumo de energia.

Os algoritmos DOT e FOTP usam grafos planares gerados por GG para criar suas estruturas lógicas. Já os algoritmos PET e POOT usam RNG para criar suas faces. Nesse caso, os nós divulgam suas localizações e determinam seus vizinhos de face. Cada nó precisa saber somente as informações dos nós que estão nas mesmas faces que ele.

Os algoritmos DOT e FOTP mantêm ativados os nós da face em que o alvo se move, fazendo uma trilha da trajetória do alvo. O FOTP combina estrutura de faces com algoritmos de hexágono e previsões para minimizar a quantidade de nós que atendem o rastreamento. O POOT mostra algoritmo eficiente de recuperação de alvos em estruturas de faces para reduzir o tempo e a quantidade de comunicação para completar essa tarefa.

2.5.2 Formulação do Problema

Os algoritmos DAB, DAT, DAT+Shortcut e o trabalho proposto por Yeong-Sung et al. [2010] são usados na formulação de rastreamento baseada em consulta. Eles constroem

e atualizam a estrutura da árvore com base na trajetória do alvo. Quando a consulta para um alvo específico é feita pelo *sink*, a consulta segue um caminho específico na árvore para encontrar o nó mais próximo do alvo. Esse caminho é traçado durante a a fase de atualização da árvore.

Os algoritmos DOT, PET e POOT são propostos para a formulação de navegação guiada. Esses algoritmos marcam alguns nós que detectaram o alvo, formando um caminho que deverá ser seguido pelo objeto guiado. No algoritmo proposto por Sharma et al. [2011b], o nó *sink* comunica a posição atualizada do alvo diretamente ao objeto guiado usando um rádio de alta potência.

Com exceção dos estudos citados acima, os algoritmos usados são classificados na formulação clássica do problema de rastreamento, em que os nós que detectam o alvo coletam dados sobre o mesmo por algum tempo e depois relatam o resultado do rastreamento ao nó *sink*.

2.5.3 Número de Alvos

Vários algoritmos em redes de sensores fazem rastreamento de um único alvo [Bhuiyan et al., 2010; Zhang & Cao, 2004a]. Entretanto, rastreamento de múltiplos alvos representam grande importância devidos suas aplicações reais. Rastrear múltiplos alvos consome mais energia, pois aumenta o número de nós ativos para rastreamento e por aumentar a quantidade de resultados que devem ser relatados ao *sink* [Hajiaghajani et al., 2012; Xu et al., 2004b].

O HCMTT reduz as mensagens redundantes causados por alvos próximos um do outro fazendo fusão de dados. Dessa forma, os nós divulgam seus dados coletados para um único nó que gera o resultado final. O algoritmo DMMT seleciona o alvo preferencial com base na potência do sinal recebida pelo nó. A seleção é feita quando o nó sensor detecta mais que um alvo. O nó pode detectar vários alvos, mas o rastreamento é feito de forma mais precisa para os alvos mais próximos. O nó decide localmente seu alvo preferencial, deixando os alvos restantes para os outros sensores.

No PES é assumido que os nós podem identificar todos os alvos usando uma tabela de códigos, assim cada alvo pode ser rastreado individualmente. O OCO assume que existe dois tipos de nós: O primeiro tipo pode distinguir os alvos, portanto pode rastrear os alvos precisamente e só precisa ativar seus nós vizinhos quando o alvo está deixando seu raio de alcance; Já o segundo tipo somente sabe que está detectando um alvo, por isso precisa ativar periodicamente seus vizinhos, pois um ou mais alvos podem deixar seu raio de alcance.

Os algoritmos DAB, DAT e DAT+Shortcut empregam estrutura hierárquica para

rastrear uma grande quantidade de alvos. Os nós registram informações sobre a presença de cada alvo, criando rotas na árvore específicas para cada um deles. Dessa forma, o *sink* pode fazer consultas sobre qualquer alvo. Nas avaliações do CODA, dois objetos contínuos são rastreados. O rastreamento de múltiplos objetos contínuos é relativamente mais simples, pois se os alvos se aproximarem, eles se tornam um só.

Nos algoritmos PET e DCTC é assumido que somente um alvo está sendo avaliado durante o rastreamento. Os restante dos algoritmos não avaliam e nem deixam claro se existe suporte para o rastreamento de múltiplos alvos.

2.5.4 Tipo de Alvo

Na literatura sobre rastreamento de alvos existe dois tipos de alvo: simples e contínuo. Muitos estudos abordam o rastreamento de alvos simples, como animais, pessoas e veículos. Por outro lado, em alguns casos, é preciso rastrear um fenômeno, como fogo, gases tóxicos e difusão de material bioquímico. Esses fenômenos são chamados de alvos contínuos porque são continuamente distribuídos por uma região e geralmente cobrem uma grande área. Além disso, eles podem mudar de forma, aumentar de tamanho ou se dividir em objetos menores [Chang et al., 2008; Hong et al., 2010; Park et al., 2010].

O CODA usa agrupamento híbrido para detectar e rastrear alvos contínuos. Para isso, a rede é dividida em agrupamentos estáticos, sendo que o alvo contínuo pode se espalhar por mais de um desse agrupamentos. Os nós que detectam o alvo transmitem a informação de detecção para seus respectivos CHs, que por sua vez executam uma função para estimar o conjunto de sensores que formam os limites do alvo contínuo em cada agrupamento estático. Depois esses sensores são unidos para formar um agrupamento dinâmico que representa os limites de todo o alvo contínuo.

O PRECO prolonga o tempo de vida da rede enquanto rastreia alvos contínuos. Trabalhos anteriores, como o CODA, consideram que todos os nós estão ativos, mesmo quando o alvo está distante dos mesmos. Por essa razão, para otimizar a eficiência energética, um componente de ativação é usado para ativar somente os nós sensores próximos do alvo. O TG-COD usa dois níveis de estrutura de grade para reduzir o custo de organização dos agrupamentos e dar forma mais detalhada do alvo contínuo.

O restante dos algoritmos usados na classificação consideram o rastreamento de alvos simples em suas avaliações.

2.6 Detalhamento

Nesta seção apresentamos uma descrição mais detalhada sobre o funcionamento de alguns algoritmos de rastreamento. Os algoritmos escolhidos para isso foram o STUN, o CODA e o PET, pois cada um deles representa uma das estruturas explicadas anteriormente (árvore, agrupamento e faces, respectivamente).

2.6.1 STUN

O STUN [Kung & Vlah, 2003] é um algoritmo que aplica uma árvore hierárquica para conectar os sensores, de forma que o *sink* é a raiz, os sensores são as folhas e o restante dos nós são chamados de nós intermediários. Essa estrutura é construída por outro algoritmo denominado DAB, que utiliza o padrão de mobilidade do alvo para cumprir seu objetivo.

No STUN, um evento é gerado sempre que um alvo entra ou sai do raio de detecção de um sensor. A informação sobre a presença do alvo é enviada pelos sensores ao nó *sink*, sendo que os nós intermediários também armazenam essa informação juntamente com seus descendentes. Dessa forma, os nós folhas armazenam a informação da presença somente dos alvos que eles detectaram, enquanto a raiz contém a informação de presença de todos os alvos. Entretanto, para que o nó raiz fique atualizado, não é necessário que a informação de todos os nós folhas cheguem à raiz. Nesse ponto, entra o mecanismo de poda para eliminar as transmissões redundantes.

No mecanismo de poda, um nó intermediário somente encaminha a mensagem de detecção do alvo se houver modificação nessa informação. Caso contrário, a mensagem é interrompida sem qualquer modificação nos seus ancestrais. Essa poda é o recurso chave para reduzir o custo com comunicação.

A figura 2.6 mostra um exemplo do mecanismo de poda. O alvo é detectado pelos sensores A, B, C e D, então cada um desses envia uma mensagem sobre a chegada do alvo. O sensor A atualiza a informação de presença em todos os seus ancestrais, pois até então nenhuma informação havia sido registrada. Depois disso, todas as mensagens dos outros sensores não atualizam o nó 3 e não são passadas adiante, pois esse nó é ancestral de todos os sensores e já foi atualizado pelo sensor A. De forma similar, o sensor C atualiza a informação de presença do nó 2, que por sua vez encaminha para o nó 3, entretanto a informação desse último permanece inalterada e poda a transmissão. As mensagens dos nós 2 e 4 não atualizam nenhum nó e são logo podadas.

A árvore do STUN é gerada das folhas para a raiz usando o algoritmo DAB. Esse algoritmos liga os sensores em uma sub-árvore balanceada, a efetividade desse método

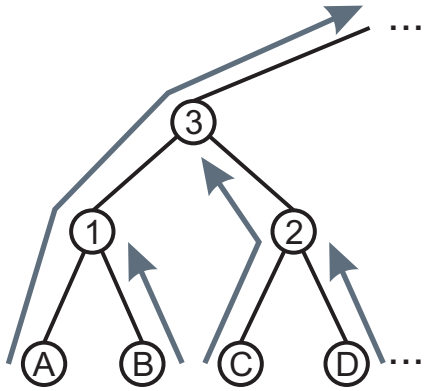


Figura 2.6. Mecanismo de poda do STUN (figura adaptada de Kung & Vlah [2003]).

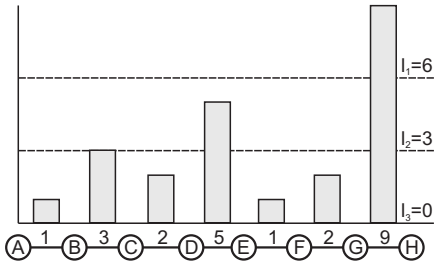
depende de escolher bem os nós que serão unidos. Para isso, o algoritmo assume que o alvo sempre é detectado por um par de sensores adjacentes, os sensores são ditos adjacentes se o alvo passa do raio de detecção de um sensor para outro sem passar por um terceiro.

Dessa forma, é possível representar o padrão de mobilidade do alvo associando um peso a cada par de sensores adjacentes, ou seja, o par de nós que mais detecta o alvo tem maior peso. Esses pesos são representados por um grafo ponderado, chamado grafo de sensores. A figura 2.7(a) mostra um grafo de sensores onde o par de sensores $G - H$ é o que mais detectou o evento (nove vezes), logo esse par é o que possui maior peso. Os pesos são divididos em uma sequência decrescente de acordo com um grupo de limiares, sendo que o valor do último limiar sempre é zero, para garantir que todos os sensores serão inseridos na árvore. No exemplo dessa figura tem três limiares: $l_1 = 6$, $l_2 = 3$ e $l_3 = 0$. Esses pesos e limiares são usados como referências para definir quais nós serão unidos.

No DAB, a árvore é inicialmente vazia. Para cada limiar, o algoritmo faz uma iteração. Cada iteração é dividida em duas fases: drenagem e escoamento. Na drenagem, os nós que têm pelo menos uma aresta incidente com o peso maior ou igual ao limiar atual são adicionados à árvore. A fase de escoamento liga pares de árvores adjacentes, duas árvores são adjacentes se alguma de suas folhas são adjacentes no grafo de sensores. Dessa forma, a raiz das duas árvores adjacentes são ligadas à um novo nó intermediário, a árvore resultante dessa ligação terá um número menor de sensores do que os outros possíveis pares de árvores adjacentes.

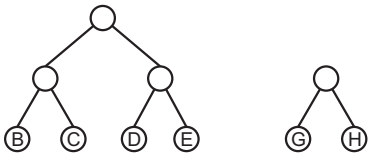
A figura 2.7 mostra um exemplo do processo de construção da árvore usando DAB. Na figura 2.7(a) são representados os nós sensores e os pesos atribuídos a cada par de sensores adjacentes (grafo de sensores). As barras verticais indicam o limiar

que é alcançado por esses pares de sensores. A figura 2.7(b) mostra o resultado da primeira iteração do DAB, nesse ponto o único par de sensores que alcança o limiar $l1$ é $G - H$, então esses nós são unidos por um novo nó intermediário. O resultado da segunda iteração é apresentado na figura 2.7(c), em que os pares de sensores $B - C$ e $D - E$ alcançam o limiar $l2$, então cada par é unido por novos nós intermediários. Além disso, como as duas árvores geradas são adjacentes, as suas raízes são ligadas por mais um nó intermediário. Por fim, a figura 2.7(d) mostra a árvore resultante após a execução da última iteração, em que todos os sensores são incluídos.

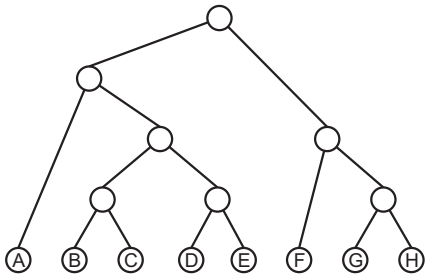


(a) Grafo de sensores e os pesos de cada par de sensores adjacentes.

(b) Árvore gerada após a primeira iteração.



(c) Árvore gerada após a segunda iteração.



(d) Árvore gerada após a terceira iteração

Figura 2.7. Exemplo da construção de uma árvore usando DAB (figura adaptada de Kung & Vlah [2003]).

A quantidade de limiares usados afetam na qualidade da árvore gerada pelo DAB. Se um único limiar for usado, não haverá diferença entre os nós com altas e baixas taxas de detecção (peso). Entretanto, se muitos limiares fores usados, haverá uma quantidade limitada de sensores para serem ligados, dificultando a obtenção de uma árvore balanceada.

2.6.2 CODA

O CODA [Chang et al., 2008] é um algoritmo desenvolvido para rastrear alvos contínuos (como fogo e material bioquímico). Esse tipo de alvo se difere dos alvos tradi-

cionais, pois ele é continuamente espalhado em uma região e geralmente cobre uma área grande, quando comparado com um alvo tradicional. Além disso, ele tende a aumentar de tamanho, mudar de forma e até mesmo se dividir em dois ou mais. Dessa forma, o mecanismo do CODA possibilita que os nós sensores detectem o movimento do contorno do alvo contínuo para determinar a área em que ele está espalhado.

O mecanismo do CODA é uma otimização do trabalho de Ji et al. [2004] que se baseia em um esquema de agrupamento híbrido (estático/dinâmico). A rede é dividida em agrupamentos estáticos, em que os nós que detectam o alvo contínuo transmitem a informação de detecção para o líder do seu grupo. Com essa informação, o líder executa uma função para estimar o contorno de sensores que detectam o alvo dentro do agrupamento estático. Depois os contornos de sensores se unem para formar um agrupamento dinâmico que envia a informação do contorno do alvo para o *sink*. O CODA pode ser dividido em quatro fases: configuração, detecção, reconhecimento e rastreamento.

Na fase de configuração, no estágio de implantação da rede, o CODA divide os nós sensores em agrupamentos estáticos, usando qualquer técnica de agrupamento estático já existente, como nos trabalhos de Handy et al. [2002], Lindsey & Raghavendra [2002] e Manjeshwar & Agrawal [2001]. Dentro de cada agrupamento estático, os nós podem ser classificados como líder do agrupamento (CH), sensor de borda (*Static-cluster Boundary-sensor* (SB)) ou sensor interno (*Static-cluster Inner-sensor* (SI)). A figura 2.8 mostra uma rede dividida em agrupamentos estáticos e seus nós sensores de acordo com a classificação.

Os CHs são escolhidos de acordo com a técnica de agrupamento estático utilizada, a função desses nós é coletar os dados do agrupamento, fazer a fusão e enviar o resultado ao nó *sink*. O CODA considera que todos os nós conhecem a sua localização, dessa forma, logo depois que os agrupamentos estáticos são formados, os nós do agrupamento informam suas posições ao CH. Além disso, ele utiliza as posições para determinar quais nós são SBs, resolvendo o problema do invólucro convexo [Nakagawa et al., 2009], e os informa dessa classificação. Os SBs são os nós que estão posicionados nos limites dos agrupamentos, a função desses nós é identificar por quais agrupamentos o alvo contínuo está espalhado. Os nós restantes são os SIs, a função dessa classe de nós é sensoriar o alvo e enviar os dados ao seu CH.

A segunda fase consiste em detectar os contornos de sensores em cada agrupamento alcançado pelo alvo contínuo. Os SIs que detectam o alvo enviam para o seu CH uma mensagem de sensoriamento contendo somente o seu identificador. Enquanto isso, os SBs que detectam o alvo se comunicam com seus vizinhos SBs de outro agrupamento para saber se o alvo também o cobre. Em seguida os SBs enviam para o seu

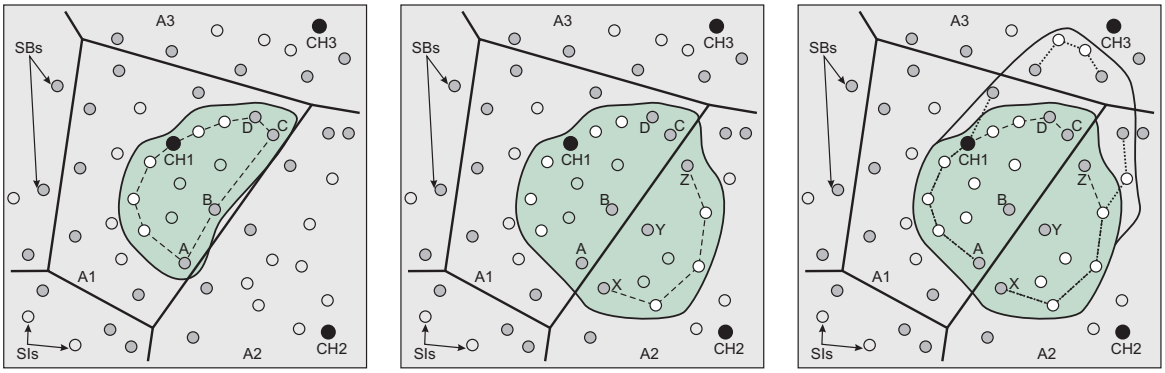
CH uma mensagem de notificação contendo o seu identificador e um outro valor que indica todos os agrupamentos alcançados pelo alvo. Dessa forma, os CHs envolvidos conhecem os nós do seu agrupamento que detectam o alvo e os agrupamentos por onde o alvo está espalhado. Cada CH calcula o contorno dos sensores que pertencem ao seu agrupamento da mesma forma que determina os SBs na fase anterior, entretanto utiliza apenas as posições dos nós que detectam o alvo, além de eliminar nós redundantes. A eliminação de nós redundantes se dá quando o alvo se espalha por mais de um agrupamento, então o CH de cada agrupamento calcula a distância entre cada par de SBs que detectam o alvo e mantém somente os nós do par que possui a maior distância.

A figura 2.8(a) mostra um exemplo em que o alvo contínuo (linha curva sólida) cobre apenas o agrupamento $A1$. Nesse caso, os SIs que detectam o alvo enviam mensagens de sensoriamento ao $CH1$, enquanto os SBs A, B, C e D consultam seus vizinhos SBs nos agrupamentos $A2$ e $A3$ para depois enviar uma mensagem de notificação ao $CH1$. Assim, o $CH1$ verifica que o alvo cobre apenas o seu próprio agrupamento, por isso usa a localização de todos os nós que detectaram o alvo para estimar o seu contorno (linha pontilhada).

Já na figura 2.8(b), o alvo cobre os agrupamentos $A1$ e $A2$. Nesse caso os SIs dos dois agrupamentos enviam mensagens de detecção para seus respectivos CHs. Os SBs A, B e C consultam, respectivamente, os seus vizinhos SBs X, Y e Z do agrupamento $A2$, sendo que esses últimos também consultam os primeiros. O SB D também consulta o seu vizinho do agrupamento $A3$. Quando as mensagens de notificação são entregues, o $CH1$ e o $CH2$ verificam que o alvo está espalhado pelos agrupamentos $A1$ e $A2$. Cada um desses CHs estimam somente a parte do contorno do alvo que está no seu agrupamento, entretanto precisam eliminar os nós redundantes. Para isso, o $CH1$ calcula a distância entre cada par dos seus nós SBs, ou seja, os pares $A - B$, $B - C$ e $A - C$, e mantém somente os nós que pertencem ao par com maior distância, ou seja, os nós A e C. Esse mesmo processo é realizado pelo $CH2$, que elimina o nó Y.

Na fase de reconhecimento, depois que cada agrupamento gera uma parte do contorno do alvo, cada CH organiza os sensores que formam esse contorno como um novo agrupamento. Esse novo agrupamento é dito dinâmico porque a sua formação depende do alvo contínuo, além disso esse agrupamento pode ser ajustado de acordo com o movimento desse alvo. Os CHs compactam suas informações de contorno e as enviam ao *sink*, que as compila para determinar o contorno completo do alvo.

Por fim, a fase de rastreamento atualiza os membros do agrupamento dinâmico no caso de mudanças no contorno do alvo. Uma vez que o alvo foi reconhecido, o algoritmo precisa apenas acompanhar suas mudanças. A atualização ocorre se o contorno do alvo alcança um ou mais novos nós ou então deixa de ser detectado por um ou mais dos



(a) Alvo contínuo espalhado pela área de um único agrupamento.

(b) Alvo contínuo espalhado pela área de dois agrupamentos.

(c) Atualização de contorno do alvo.

Figura 2.8. Exemplos de detecção e atualização de contorno do alvo contínuo com CODA (figura adaptada de Chang et al. [2008]).

nós atuais. No primeiro caso, quando outros nós SBs ou SIs começam a detectar o alvo, eles enviam as mensagens de sensoriamento e notificação para o seus respectivos CHs, igualmente à fase de detecção (segunda fase). Os CHs mantêm as informações de detecção, por isso basta adicionar as informações dos novos nós e redefinir o contorno do alvo. Já no segundo caso, quando um nó deixa de detectar o alvo, ele envia uma mensagem para informar seu CH, que por sua vez o remove do grupo e redefine o contorno do alvo.

A figura 2.8(c) mostra uma atualização nos sensores de contorno quando o alvo contínuo se expande e alcança novos nós. No estado anterior do alvo (área cinza envolvida por linha curva sólida), ele está espalhado nos agrupamentos $A1$ e $A2$, portanto alguns nós desses agrupamentos são os sensores de contorno (nós conectados por linha tracejada). Entretanto, no estado atual (áreas cinza e branca envolvidas por linha curva sólida), o alvo se expande e alcança o agrupamento $A3$. Os nós que detectam o alvo pela primeira vez (nós localizados na área branca envolvida por linha curva sólida) enviam as mensagens de detecção e notificação para seus CHs, que por sua vez os inclui como sensores de contorno e redefine o contorno do alvo (nós conectados por linha pontilhada).

2.6.3 PET

O PET [Bhuiyan et al., 2010] analisa o caminho percorrido pelo alvo e utiliza o padrão dessa movimentação para economizar energia e aumentar o tempo de vida da rede. O alvo é rastreado em regiões conhecidas como faces, obtidas por técnicas de planariza-

ção [Cong & Jie, 2009]. Esse algoritmo mantém a maioria dos nós inativos, ativando somente os sensores necessários de acordo com a previsão. É assumido que todos os nós conhecem sua localização e a localização dos seus vizinhos e que os relógios dos mesmos estão sincronizados.

Esse algoritmo é aplicado em um contexto de rastreamento em que um usuário móvel (*tracker*) tenta alcançar o alvo consultando a rede para obter a localização do mesmo. Dessa forma, além dos nós sensores e do alvo, existe ainda a figura do usuário móvel e do nó guia (*beacon*) para interagir com a rede. Um nó guia é um nó comum escolhido pela rede, trata-se do nó mais próximo do alvo no momento da detecção. O objetivo desse nó é informar ao usuário móvel a posição do alvo e trocar informações com os outros nós.

Para isso, a rede é modelada com base no conceito de *Face-Aware Routing* (FAR) [Huang et al., 2005], sendo que a mesma é planarizada como um *Unit Disk Graph* (UDG) [Cong & Jie, 2009], formando as faces. Cada nó conhece todos os nós que estão nas suas faces e as suas localizações. Entretanto só há comunicação entre os vizinhos da face em que o alvo está presente. Por exemplo, a figura 2.9 mostra o exemplo de uma rede organizada em faces, sendo que o nó sensor $s1$ é o nó guia. Esse nó está associado às faces $f1$, $f2$ e $f5$, então ele armazena as informações sobre todos os nós dessas faces, ou seja, $s1$, $s2$, $s3$, $s4$, $s5$, $s6$ e $s7$. Porém, $s1$ possui apenas três vizinhos imediatos, que são $s2$, $s4$ e $s7$. Além disso, o PET usa somente os vizinhos presentes na face em que o alvo está, esses são chamados de vizinhos imediatos de face. Considerando que o alvo está na face $f2$, o nó sensor $s1$ precisa se comunicar apenas com $s4$ e $s7$. Desconsiderando os nós restantes, o PET reduz o custo de energia.

Com o objetivo de economizar energia, os nós podem operar em três estados: inativo (*inactive*), acordado (*awake*) e ativo (*active*). No estado inativo, os nós desabilitam os canais de comunicação e sensoriamento. Eles só passam para o estado acordado se receberem mensagens para isso. No estado acordado, um nó periodicamente ativa o canal de comunicação para checar possíveis mensagens de aproximação do alvo. Se essas mensagens não chegarem, ele volta para o estado inativo, caso contrário passa para ativo. No estado ativo, o nó participa do processo de rastreamento e aguarda por mensagens dos seus vizinhos ou do usuário. Ele passa para o estado acordado depois que a sua tarefa é realizada ou para inativo, caso receba uma mensagem para isso.

O PET estima a posição e a direção do alvo usando previsão linear, que usa a posição atual e a anterior para obter a próxima. Dado que t_i é o tempo atual, que (x_i, y_i) é a posição atual do alvo e (x_{i-1}, y_{i-1}) é a posição anterior, a velocidade do alvo

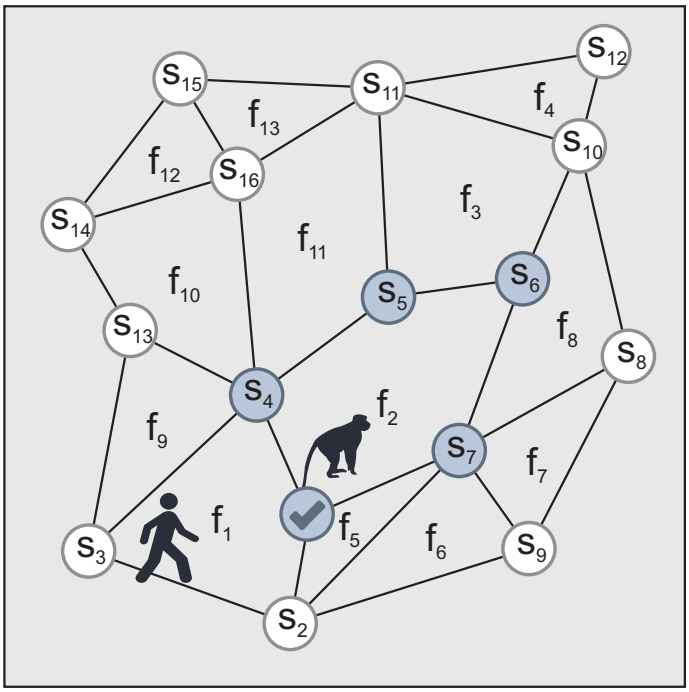


Figura 2.9. Rede organizada em faces (figura adaptada de Bhuiyan et al. [2010]).

é estimada como

$$v = \frac{\sqrt{(x_i, x_{i-1})^2 + (y_i, y_{i-1})^2}}{t_i - t_{i-1}}, \tag{2.17}$$

já a direção é obtida por

$$\Theta = \cos^{-1} \frac{x_i - x_{i-1}}{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}. \tag{2.18}$$

Sendo assim, a previsão do alvo é estimada como

$$\begin{cases} x_{i+1} = x_i + vt_i \cos \Theta \\ y_{i+1} = y_i + vt_i \sin \Theta. \end{cases} \tag{2.19}$$

A informação de previsão é usada para avisar antecipadamente os nós próximos dessa posição sobre a aproximação do alvo. Assim, esses nós ficam prontos para a tarefa de sensoriamento.

Todo o processo de rastreamento do PET pode ser dividido em quatro fases: inicialização, detecção do alvo, estimativa de localização e direção e movimento do usuário. Na fase de inicialização todos os nós estão acordados. O usuário inicia um *flooding* na rede, ativando todos os nós, para solicitar a localização do alvo. Essa mensagem é enviada quando os nós ativam o canal de comunicação. Na fase de detecção

do alvo, os nós que estão detectando o alvo cooperam para determinar qual deles está mais próximo do alvo. Esse nó é ativado, escolhido como guia e, com ajuda dos seus vizinhos imediatos de face, calcula a posição do alvo. Na terceira fase, o nó guia prevê a localização e a direção do alvo, de acordo com as Equações 2.17, 2.18 e 2.19, descritas anteriormente. Ele verifica qual é o nó mais próximo da posição prevista e o informa da aproximação do alvo. Na quarta fase, o usuário móvel segue em direção ao nó guia para consultar a próxima direção que deve seguir. Se o guia tem essa informação, o usuário móvel segue seu caminha e o algoritmo volta para a segunda fase, caso contrário, o alvo foi perdido e o algoritmo volta para a primeira fase.

2.7 Mobilidade

A mobilidade (i.e. deslocamento espacial) de um objeto móvel pode ser adquirida registrando a movimentação do objeto de interesse em cenários reais ou então utilizando os modelos de mobilidade. A primeira opção é difícil de ser obtida, muitas vezes até inviável, por isso os modelos de mobilidade são amplamente explorados. Esses modelos descrevem a posição, velocidade, direção e outros estados dinâmicos de objetos em movimento, descrevendo um padrão da trajetória [Bai & Helmy, 2004].

Inicialmente, o principal objetivo dos modelos de mobilidade era caracterizar o movimento de partículas em sistemas físicos [Einstein, 1956]. Posteriormente, a aplicação desses modelos se estenderam para outras áreas, como em *Mobile Ad hoc NETworks* (MANETs), na computação, em que esses modelos são muito importantes na simulação e validação de protocolos [Broch et al., 1998], e na biologia, para a modelagem de movimentos de animais, células e outros organismos [Codling et al., 2008].

Dependendo do problema abordado, deve ser adotado um modelo de mobilidade mais adequado, ou seja, mais próximos dos observados em situações reais, caso contrário as observações e conclusões dos estudos podem ser equívocas. Por isso, existem diversos trabalhos que propõem modelos de mobilidade, novos ou adaptados, para um problema específico, buscando encontrar características para criar uma mobilidade mais realística [Gloss et al., 2005; Zheng et al., 2004; Jardosh et al., 2003].

Nesta seção descreveremos alguns dos modelos de mobilidade mais conhecidos, classificando-os de acordo com Bai & Helmy [2004]. Como o foco deste trabalho é o rastreamento de animais, também trataremos nesta seção alguns trabalhos que simulam trajetórias de animais usando modelos de mobilidade.

2.7.1 Modelos de Mobilidade

Bai & Helmy [2004] classificaram os modelos de mobilidade de acordo com as características específicas de cada tipo de modelo. Dessa forma, esses modelos podem ser classificados pela aleatoriedade, dependência temporal, dependência espacial e restrições geográficas.

2.7.1.1 Modelos de Mobilidade Aleatórios

Nos modelos de mobilidade aleatórios, o objeto se move aleatoriamente e sem restrições, isto é, a velocidade e direção são escolhidas de forma aleatória e independente de outros objetos. Esses modelos são mais simples e os mais comuns na literatura. O *Random WayPoint* (RWP) [Broch et al., 1998] e suas duas variações, o *Random Walk* (RW) [Pearson, 1905] e o *Random Direction* (RD) [Nain et al., 2005], são modelos de mobilidade aleatórios frequentemente utilizados.

Random Waypoint Um objeto móvel que segue o modelo RWP escolhe aleatoriamente um ponto de destino. Ele se move até esse destino com uma velocidade constante, essa velocidade também é escolhida aleatoriamente. Quando chega ao destino, o objeto permanece parado por um determinado tempo. Depois disso, escolhe outro destino e velocidade para recomeçar o processo.

Nesse modelo existem basicamente três parâmetros: a velocidade mínima, a velocidade máxima e o tempo de espera. As velocidades mínima e máxima determinam um intervalo $[v_{min}, v_{max}]$ em que a velocidade, que é escolhida aleatoriamente, está uniformemente distribuída. O tempo de espera determina o tempo em que o objeto fica parado antes de seguir para o próximo destino, caso o tempo de espera seja zero, há uma mobilidade contínua. Variando esses parâmetros podem ser obtidos cenários de pouca mobilidade (velocidade máxima baixa e tempo de espera longo) até cenários de intensa mobilidade (velocidade máxima alta e tempo de espera curto).

O RWP pode ser adaptado de forma que os pontos de destino escolhidos pelo objeto móvel sejam uniformemente distribuídos em um determinado domínio, delimitando a área de locomoção. Quando o objeto chega ao limite da área podem ser realizadas três ações: eliminar o objeto móvel atual e criar um novo em outro ponto; fazer o objeto móvel refletir no limite; ou transportá-lo para o lado oposto da área. Essa adaptação é chamada de *Random Waypoint on the Border* (RWPB) [Bettstetter & Wagner, 2002]. Em simulações de rede, geralmente, a mobilidade dos nós é delimitada, por isso o RWPB é bastante aplicado nesses estudos.

Em MANET, o RWP é muito utilizado na validação de protocolos. Entretanto, Yoon et al. [2003] mostram que nesse modelo a velocidade média dos nós cai continuamente. Em seus experimentos eles mostram que pequenas mudanças nos limites de velocidade dos nós produzem grandes mudanças nas métricas de controle, atraso e número de pacotes perdidos em protocolos de roteamento. Outro problema do RWP, apresentado por Bettstetter [2001a], é a ocorrência de uma distribuição não uniforme dos nós, em que os nós se concentram mais no centro da área a medida que o tempo de simulação aumenta.

Apesar de ser muito conhecido e utilizado, o RWP gera paradas, acelerações e manobras súbitas [Campos et al., 2004]. Por isso, não é uma boa escolha na maioria dos casos, pois é pouco provável que em uma situação real a mobilidade de um objeto seja similar a desse modelo.

Random Walk O RW, também conhecido como *Brownian motion*, é um tipo específico de RWP em que o objeto móvel modifica a sua direção e velocidade a cada intervalo de tempo discreto, entretanto não há tempo de espera. Em sua forma tradicional, o RW não possui correlação, isto é, a direção escolhida pelo objeto móvel é totalmente independente das direções escolhidas anteriormente.

Nesse modelo existem dois parâmetros para serem ajustados: a velocidade mínima e a velocidade máxima. Essas velocidades determinam um intervalo $[v_{min}, v_{max}]$ em que a velocidade, que é escolhida aleatoriamente, está uniformemente distribuída. A direção do objeto é escolhida aleatoriamente e uniformemente no intervalo $(0, 2\pi]$.

O RW, assim como o RWP, são modelos de mobilidade que não registram o histórico da mobilidade, sendo assim, as informações de estados anteriores não são utilizados na escolha do próximo estado, isso ocasiona manobras bruscas e aceleração repentina.

Random Direction O RD foi criado para evitar o problema de distribuição não uniforme do RWP. Nesse modelo, o objeto móvel em vez de selecionar um ponto de destino seleciona uma direção qualquer e se move até ela por um determinado tempo. Caso a área seja delimitada, o objeto é refletido quando alcança o limite.

Esse tipo de mobilidade consiste de duas fases: movimento e espera. Na primeira fase o objeto escolhe a direção, uniformemente distribuída no intervalo $(0, 2\pi]$, para onde se move por um certo tempo. Na próxima fase ele aguarda um tempo antes de começar outra movimentação. Nas fases de movimento seguintes a direção é distribuída uniformemente no intervalo $[0, \pi]$.

No RD devem ser ajustados três parâmetros: velocidade, tempo de movimentação e tempo de espera. A velocidade é escolhida aleatoriamente no início da fase de movimentação e determina a distância que o objeto percorrerá durante o tempo de movimentação. Depois disso, ele fica parado de acordo com o intervalo determinado no parâmetro tempo de espera antes de reiniciar o ciclo.

2.7.1.2 Modelos de Mobilidade com Dependência Temporal

Nos modelos de mobilidade aleatórios, os componentes de movimento (como posição e velocidade) são calculados independentemente dos estados do objeto em tempos anteriores. Nos modelos com dependência temporal, o estado atual do objeto móvel depende dos estados anteriores, por isso os componentes de movimento são correlacionados.

A natureza sem memória dos modelos aleatórios os tornam incapazes de considerar os estados anteriores. Sendo assim, surgiram vários novos modelos de mobilidade com dependência temporal. Dois exemplos comuns desse tipo de mobilidade são *Correlated Random Walk* (CRW) [Gillis, 1955], *Smooth Random* (SR) [Bettstetter, 2001b] e *Lévy Walk* (LW) [Rhee et al., 2008].

Correlated Random Walk O CRW é um modelo de mobilidade usado como ferramenta da física estatística para obter um modelo aproximado do comportamento animal [Wu et al., 2000]. Esse modelo é um RW que tem a tendência de manter sua direção, ou seja, existe correção entre os passos sucessivos da trajetória. Para isso, inclui o conceito de memória direcional, que determina o grau de relação em uma caminhada aleatória.

Para configurar esse modelo existem basicamente dois parâmetros: o tamanho do deslocamento e o grau do correlação (α). O tamanho do deslocamento pode ser a distância entre cada deslocamento (movimento uniforme) ou a distância média. O parâmetro α pode variar entre 0 e 1 ($0 \leq \alpha \leq 1$). Variando o parâmetro α, os seguintes cenários de mobilidade podem ser obtidos:

- Cenário aleatório: Nesse caso $\alpha = 0$, significa que o modelo estar sem memória, ou seja, a velocidade e direção atuais do objeto independem do da velocidade e direção anteriores. Esse cenário se torna equivalente ao modelo RW caso não haja tempo de espera.

- Cenário determinístico: Ocorre quando $\alpha = 1$, a mobilidade passa a ser totalmente determinística, de forma que a velocidade e direção atuais são iguais as do tempo anterior.

- Cenário intermediário: Se $0 < \alpha < 1$, a velocidade e direção atuais dependem das anteriores, entretanto ainda há um certo grau de aleatoriedade. Quanto mais o valor de α aproxima-se de 1, mais determinístico é o modelo. Já quanto mais próximo de 0, o modelo se torna mais aleatório.

Smooth Random O modelo de mobilidade SR possui dependência temporal de velocidade e direção, pois as alterações dessas variáveis são incrementais e suaves. Isso evita manobras bruscas e mudanças súbitas de velocidade, que geram mobilidades muito diferentes dos cenários reais, como ocorre com o RWP. A inspiração desse método veio de observações em mobilidades reais, em que foi concluído que, na maioria das vezes, o objeto se move usando um conjunto fixo de velocidades.

Nesse modelo, o objeto móvel possui um conjunto de velocidades preferenciais $[v_1, v_2, ..., v_n]$. As velocidades preferenciais têm alta probabilidade de serem escolhidas, enquanto o restante está uniformemente no intervalo $[v_{min}, v_{max}]$. A frequência com que a velocidade muda segue uma distribuição de Poisson. Já a direção do movimento é distribuída uniformemente no intervalo $[0, 2\pi]$ e sua frequência de mudança segue uma distribuição exponencial.

Lévy Walk No modelo LW existe a definição de tempo de deslocamento, em que o deslocamento é a distância que o objeto móvel percorre sem fazer pausa ou mudar a direção do movimento. A principal característica desse modelo é a presença de grandes saltos alternados com um conjunto de pequenos deslocamentos.

Cada passo do LW é representado por quatro variáveis: tamanho do deslocamento, direção, tempo de deslocamento e tempo de espera. O modelo seleciona o tamanho e tempo de deslocamento a partir de duas distribuições de Lévy. Este método é complexo, pois possui muitos parâmetros que podem ser usados para ajustar o padrão de movimentação.

2.7.1.3 Modelos de Mobilidade com Dependência Espacial

Modelos de mobilidade com dependência espacial tratam da movimentação de múltiplos objetos móveis e a relação entre eles. Dependendo do cenários, a trajetória de um objeto móvel pode influenciar a trajetória de outros objetos próximos.

A dependência espacial permite considerar alguns cenários de mobilidade específicos, como a mobilidade de veículos em uma estrada, em que um veículo não deve ultrapassar a velocidade do veículo a sua frente. Um exemplo comum de mo-

delo de mobilidade com dependência espacial é o *Reference Point Group Mobility* (RPGM) [Hong et al., 1999].

Reference Point Group Mobility No modelo RPGM, cada grupo tem um ponto, que é um centro lógico ou um objeto é o líder do grupo. A movimentação do líder ou do centro lógico determina o comportamento de movimentação de todo o grupo. Exemplos de mobilidades reais semelhantes a esse modelo podem ser de um grupo de soldados durante uma operação militar ou então de equipes de resgate trabalhando cooperativamente.

A movimentação do líder pode ser feita de forma aleatória ou ser moldado como um caminho pré-definido. Essa movimentação não define apenas a trajetória do líder, mas também de todos os membros do grupo. Cada membro desse grupo é associado com um ponto de referência que segue a movimentação do grupo.

2.7.1.4 Modelos de Mobilidade com Restrições Geográficas

Nos modelos de mobilidade com restrições geográficas, as condições geográficas da área onde o objeto móvel se encontra influenciam na escolha de sua trajetória, diferente dos modelos aleatórios que se movem livremente, desconsiderando as condições da área.

Em muitos cenários reais existem algum tipo de obstáculo que impede a livre movimentação dos objetos móveis. Dois exemplos de mobilidade que consideram restrições geográficas são *Pathway Mobility* [Tian et al., 2002] e *Obstacle Mobility* [Jardosh et al., 2003].

Pathway Mobility Um tipo de restrição geográfica comum em cenários reais e fazer o modelo de mobilidade seguir por caminhos de um mapa pré-definido. Esse tipo de mobilidade representa a mobilidade de veículos que se movem por estradas ou a movimentação de pedestres em uma área urbana.

Os mapas podem ser modelados como um grafo gerado aleatoriamente ou baseado em alguma mapa real. Os vértices do grafo representam os possíveis destinos, um prédio por exemplo, que o objeto móvel deseja alcançar. Já as arestas representam os caminhos entre esses destinos, como estradas e ruas. Inicialmente, os objetos móveis são posicionados nos vértices do grafo, então escolhem aleatoriamente um destino. Depois o objeto se move pelo menor caminho até o destino, e ao chegar pausa por um tempo e escolhe um novo destino.

Nesse tipo de mobilidade ainda há um certo grau de aleatoriedade, pois os objetos móveis escolhem aleatoriamente seus destinos. Entretanto, é mais complexo dos que os

modelos puramente aleatórios que se movem livremente e não por caminhos definidos.

Obstacle Mobility Além dos caminhos, obstáculos também influenciam na mobilidade dos objetos, pois para evitar os obstáculos o objeto precisa mudar sua trajetória. No caso de MANET, além da trajetória, os obstáculos também interferem na comunicação entre os nós.

Johansson et al. [1999] avaliam a influencia de obstáculos nos modelos de mobilidade. Nessa avaliação, a quantidade de objetos móveis e a velocidade dos mesmos são variadas para considerar diferentes situações de cenários reais. Nesses cenários, os obstáculos são distribuídos aleatoriamente e têm a forma retangular. O objeto móvel escolhe uma trajetória adequada para evitar os obstáculos. O ponto fraco desse trabalho é que quando há um obstáculo entre dois objetos não há comunicação entre eles. Já Jardosh et al. [2003] também verificam a influencia dos obstáculos na mobilidade, mas inclui uma análise de propagação de rádio.

2.7.2 Mobilidade de Animais

Modelos de mobilidade aparecem na ecologia para analisar dados de movimento animal [Codling et al., 2008]. O modelo RW pode ser usado no estudo da mobilidade de muitos organismos biológicos [Levin, 1986]. Entretanto, para produzir o movimento animal de maneira mais realística é preciso utilizar modelos com memória direcional, usados nos modelos com dependência espacial [Wu et al., 2000]. Isso se dá pela simetria bilateral e a polarização céfalo-caudal que levam os animais a ter uma tendência de ir para a frente [Bovet & Benhamou, 1988].

Dessa forma, Marsh & Jones [1988] investigam modelos de mobilidade para movimentos de animais matematicamente e por simulações. Eles classificam esses modelos de duas formas: uma em que o tamanho do deslocamento e a direção são independentes e outra em que essas variáveis são independentes. Cada classe pode ser orientada a modelos, em que a direção é relacionada com um ponto específico, ou não orientada a modelos, em que a direção é relacionada apenas com o estado anterior. Seus resultados mostram que cada classe pode representar diferentes cenários de mobilidade animal.

Um modelo de mobilidade muito utilizado para estudar o comportamento de animais é o LW. Fernandez et al. [2004] estudaram o padrão de mobilidade de macacos-aranha em seu ambiente natural e notaram invariâncias espaciais e temporais características do LW. Viswanathan et al. [1996] também encontrou um padrão semelhante no voo de gaivotas. Um estudo mais recente realizado por Humphries et al. [2010] mostra que muitos predadores marinhos, como tubarões e peixes de bico, adotam estratégias

de buscas em que a mobilidade pode ser modelada como LW quando a presa é escassa. Em alguns outros casos essa estratégia alterna entre RW e LW. De acordo com Rhee et al. [2008], o LW tem similaridades estatísticas com o padrão de movimentação humano em ambientes externos.

Como muitos animais tendem a se locomover avançando de forma persistente, o CRW tem sido bastante utilizado para modelar diversos contextos de mobilidade animal [Wu et al., 2000]. Kareiva & Shigesada [1983] variam as distribuições de probabilidade do tamanho de deslocamento e ângulos de manobras do CRW para caracterizar a mobilidade de animais, principalmente insetos e pássaros. Os resultados desse trabalho mostram que em alguns casos mais complexos não é possível obter uma boa representação, sendo sugerido, para isso, utilizar processos de Markov de alta ordem.

2.8 Conclusões Parciais

Neste capítulo apresentamos os fundamentos necessários para responder algumas questões sobre os algoritmos de rastreamento de alvos, como (1) O que é rastreamento de alvos? (2) O que buscam as abordagens elaboradas para resolver o problema de rastreamento? (3) O que as redes de sensores precisam para executar um rastreamento? Resumidamente, as respostas são:

1. É uma aplicação de redes de sensores que coleta dados sobre um ou mais alvos para estimar suas posições no campo de sensores.

2. As abordagens de rastreamento visam manter a precisão do rastreamento e o tempo em que os resultados são relatados enquanto reduzem os custos com comunicação e o consumo de energia geral da rede. Uma vez que a energia é um recurso escasso em redes de sensores.

3. Para executar o rastreamento, a rede deve satisfazer alguns requisitos, e.g. estabelecer um sistema de localização, controlar a densidade da rede e sincronizar os relógios dos nós sensores.

Neste capítulo, nós dividimos as soluções de rastreamento em três componentes básicos – fornecem os mecanismos para estimar a posição do alvo – e três componentes adicionais – fornecem mecanismos para prever a posição do alvo e usar essa informação para economizar energia. Cada um desses componentes pode ser desenvolvido usando diferentes técnicas que influenciam na qualidade e eficiência finais do algoritmo de rastreamento.

Além disso, descrevemos e classificamos alguns estudos relacionados com rastreamento de alvos em redes de sensores. Com essa classificação, verificamos que na maioria das abordagens de rastreamento, os nós cooperam usando estruturas de agrupamentos, pois essa facilita a fusão de dados e pode ser ajustada dinamicamente, eficiente para as situações que exigem maior flexibilidade. Por outro lado, as estruturas de árvore são alternativas mais interessantes para algoritmos de rastreamento baseados em consulta. Já nos algoritmos para navegação guiada, as estruturas de face são mais usadas. Assim, a estrutura mais apropriada depende de como o problema é formulado.

Independente da estrutura usada, a previsão associada com mecanismos de ativação tem um importante papel para aumentar o tempo de vida da rede. Pois a previsão possibilita que somente um pequeno conjunto de sensores permaneça ativo para rastrear o alvos.

No próximo capítulo exploramos a prova-de-conceita de rastreamento de alvos com área discretizada, o rastreamento é formulado de maneira clássica, como discutido na seção 2.1.2 e aplica os métodos de previsão discutidos nas seções 2.2.4.1 e 2.2.4.3. Nesse mesma linha, no capítulo 4 aplicamos o rastreamento com área discretizada de maneira distribuída usando a estrutura apresentada na seção 2.2.2.2 e os métodos de previsão da seção 2.2.4. Já no capítulo 5, elaboramos um algoritmo rastreamento com a presença de um objeto guiado, conforme a formulação da seção 2.1.2. Para isso, a rede é organizada de acordo com a estrutura discutida na seção 2.2.2.3.

Capítulo 3

Algoritmo Quantizado de Rastreamento em Redes de Sensores Sem Fio

Nas abordagens de rastreamento tradicionais, os nós sensores são distribuídos aleatoriamente pelo campo de sensores, um algoritmo de localização é executado antes do rastreamento, para que os nós da rede passem a conhecer aonde estão localizados. As posições geradas por esses algoritmos possuem um erro que reflete no cálculo de posição do alvo durante o rastreamento. Além disso, o erro do rastreamento também é somado ao erro de localização [Oliveira et al., 2009; Souza et al., 2013]. Sendo assim, um grande esforço é despendido para calcular a posição exata do alvo, mas uma região deve ser considerada devido ao erro acumulado.

Neste capítulo, mostramos e avaliamos a prova-de-conceito de uma abordagem quantizada de rastreamento, em que a rede é organizada em grade, onde as células são as regiões de ocupação do alvo (animal). O objetivo é mostrar que essas células podem ser usadas como referências para determinar de forma simples a posição atual do alvo e estimar sua posição futura. Assim, desconsideramos o aspecto distribuído, e toda a fusão de dados é centralizada no nó *sink*. A função dos nós sensores é apenas detectar o alvo e encaminhar os dados até o *sink*.

Esse algoritmo foi proposto no contexto dos projetos SAUIM (CNPq 55.4087/2006-5) e RastroAM (CNPq 47.4194/2007-8) para o rastreamento do sauim-de-coleira. Trata-se de um primata territorialista e presente apenas no Amazonas que está incluído nas listas nacionais e internacionais classificado como Criticamente em Perigo [Machado et al., 2005]. O objetivo de rastrear esse animal é determinar seu padrão de movimento, correlacionando com seu hábitos alimentares e comportamento social.

Nesse caso, a posição exata do alvo é dispensável, sendo importante o conhecimento da região em que ele se encontra no ambiente monitorado.

Geralmente as técnicas de rastreamento consideram que os nós são distribuídos aleatoriamente [Chang et al., 2008; Bhuiyan et al., 2010; Hsu et al., 2012], principalmente quando o ambiente monitorado é extenso. Entretanto, como nossa técnica é destinada a ser aplicada na reservas, dispor os nós em grade é mais adequado pelas seguintes razões:

- As reservas ecológicas de florestas tropicais comumente possuem vias de acesso usadas por ecologistas para estudar a fauna e a flora da região. Essas vias geralmente são feitas em forma de grade, dessa forma podem ser usadas para viabilizar a comunicação entre os nós dentro da floresta densa [Figueiredo et al., 2009].

- Estimativas de distância com base na potência do sinal são inviáveis. Geralmente, a posição do alvo é calculada com trilateração, sendo que a distância entre os nós é estimada com base na potência do sinal recebido. Entretanto, em um ambiente florestal, a variação da potência recebida é alta, inviabilizando a relação entre potência recebida e distância [Boukerche et al., 2007; Oliveira et al., 2009].

- No fim da vida útil da rede, os nós sensores precisam ser coletados. Pois esses são compostos por baterias e outros materiais prejudiciais ao meio ambiente, demandando uma coleta posterior de dispositivos inoperantes. A distribuição estratégica dos nós facilita essa tarefa.

As soluções e resultados deste capítulo estão parcialmente publicadas no Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos [Souza et al., 2011b] e no *Local Computer Networks* [Souza et al., 2011a]. Além disso, uma versão estendida está em desenvolvimento para publicação no *Journal of Distributed Sensor Networks*.

No restante deste capítulo, detalhamos o funcionamento dessa abordagem de rastreamento e mostramos os resultados das avaliações experimentais.

3.1 O Alvo

Muitos trabalhos sobre rastreamento de alvos abstraem a detecção do alvo considerando um modelo de detecção binário [Aslam et al., 2003; Wang et al., 2010a; He et al., 2010]. Nesse modelo de detecção, cada nó sensor detecta o alvo que está dentro da sua área de sensoriamento, gerando um *bit* de informação sobre o alvo: 1 para presença e 0 para ausência.

Por outro lado, há trabalhos que abordam essa questão de forma mais detalhada, verificando as mudanças no ambiente provocadas pelo alvo. O tipo de sinal a ser sensoriado depende do alvo que se deseja rastrear [Chen et al., 2004]. O trabalho de Chen et al. [2004], por exemplo, rastreia veículos com base no som, já Jin et al. [2012] classifica e rastreia animais, pessoas e veículos com base nas vibrações criadas pelos alvo, usando um sensor sísmico.

Neste trabalho, consideramos que o alvo possui um colar transmissor. Em intervalos de tempo definidos, esse colar emite sinais que são detectados pelos nós sensores que estão dentro do raio de alcance da transmissão, caracterizando os eventos a serem monitorados. Cada sinal transmitido é uma mensagem que contém o identificador (ID) do alvo e a sequência da transmissão. Com esses dados, podemos distinguir um alvo de outro e estabelecer a ordem dos eventos sem a necessidade de sincronizar os nós.

A mobilidade do alvo é dada como uma CRW [Wu et al., 2000], pois é uma ferramenta da física estatística para obter uma representação aproximada do comportamento animal.

3.2 Algoritmo Quantizado de Rastreamento

No modelo de rastreamento proposto, a topologia da rede é disposta em forma de grade com I linhas e J colunas. As interseções dessas linhas e colunas representam N nós sensores afastados entre si a uma distância d. Cada nó $n_{i,j}$, com $0 \leq i < I-1$ e $0 \leq j < J-1$, conhece a linha e coluna a que pertence. Somente com esses índices é possível determinar a célula em que o alvo está.

Diferente das abordagens tradicionais, que utilizam coordenadas absolutas para identificar a posição do alvo, nossa abordagem considera a posição do alvo como sendo um célula $c_{x,y}$, com $0 \leq x \leq I-1$ e $0 \leq y \leq J-1$. Assim, existem $I \times J$ posições para o alvo. A figura 3.1 ilustra a disposição dos nós, as células formadas e o raio de alcance do alvo (evento).

Dessa forma, o alvo divulga sua presença em intervalos de tempo definidos (ex. através de uma coleira com um transmissor). Esse evento é detectado pelos nós da grade que encaminham esses dados para o *sink*, onde a fusão de dados é feita para identificar a célula onde o alvo se encontra e a célula em que ele estará na próxima detecção. Os intervalos de tempo para a divulgação devem ser ajustados com base na velocidade do alvo e tamanho das células.

Idealmente, o alcance dos nós deve ser de pelo menos igual a distância entre os nós da grade (d), para permitir a comunicação entre todos os nós. Entretanto,

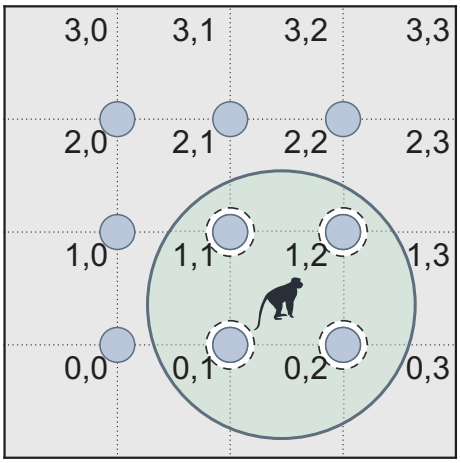


Figura 3.1. Rede organizada em grade.

em cenários reais esse alcance deve ser maior para compensar as interferências do ambiente ou alguma imprecisão da distribuição dos nós na grade. A distância entre os nós determina a área das células. Então, dependendo do porte do alvo que se pretende rastrear é possível ajustar a distância d e o alcance dos nós para obter células de tamanho adequado.

Para estimar a célula em que o alvo se encontra (tempo t) é realizado um processo de votação, que elege a célula mais votada como posição do alvo. Já a previsão da próxima posição do alvo (tempo $t+1$) é realizada com filtros *Bayesianos* alimentados com histórico de mobilidade do alvo.

3.2.1 Estimando a Posição do Alvo

Estimar a posição do alvo consiste em definir em que célula ele se encontra no tempo atual t. Para isso, são utilizadas apenas as coordenadas (i, j) contidas nos nós, fazendo uma votação entre os nós que detectam o alvo no tempo t. Esse processo de votação pode ser simples ou ponderado, como mostramos a seguir.

3.2.1.1 Votação Simples

Na votação simples, cada nó atribui um voto para suas células candidatas. As células candidatas de um nó $n_{i,j}$ são as células ao seu redor. A posição do alvo é definida como a célula mais votada.

Como prova-de-conceito, os processos de estimativa e previsão de posição são centralizados no nó *sink*. Os nós sensores que detectam o alvo no tempo t encaminham para o *sink* uma mensagem sinalizando que detectaram o alvo. Essa mensagem contém

a posição (i, j) do nó que a originou. Baseado nessa informação, o *sink* determina as células candidatas e elege a mais votada. A figura 3.2 exemplifica o processo de votação simples quando quatro nós detectam o evento. Nesse caso, existem nove células candidatas, a célula $c_{1,1}$ recebe voto de todos os nós, portanto é a mais votada e escolhida como a posição do alvo.

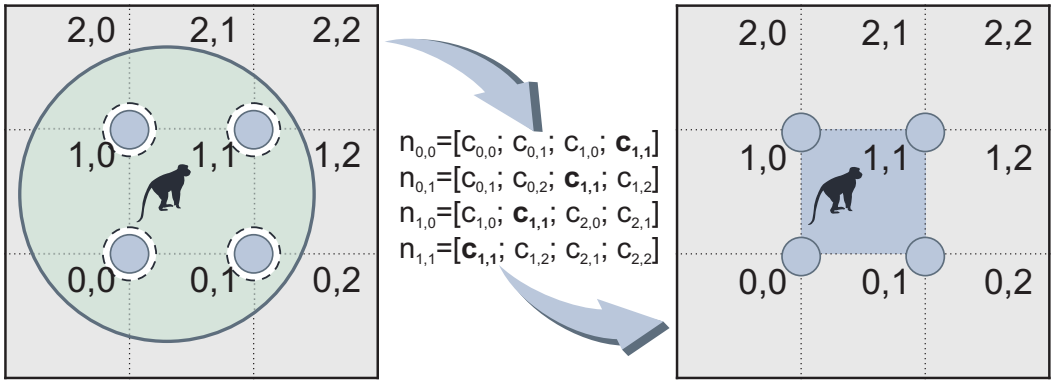


Figura 3.2. Cálculo de posição simples.

Empates podem ocorrer no processo de votação de um evento. Um possível caso de empate ocorre quando poucos nós (um ou dois) detectam o evento. Empates também ocorrem com frequência quando o alcance o alvo, ao divulgar sua posição, é maior do que distância entre os nós. Nos casos de empate, a célula eleita é a mais central dentre as candidatas, sendo calculada como

$$c(x, y) = c\left(\frac{\sum_{k=1}^{m} x_k}{m}, \frac{\sum_{k=1}^{m} y_k}{m}\right), \tag{3.1}$$

onde m é a quantidade de células candidatas e (x_k, y_k) é a coordenada da k-ésima célula candidata.

3.2.1.2 Votação Ponderada

A votação ponderada utiliza estimativas de distância (obtida através da potência do sinal) para classificar os nós sensores com maior poder de decisão. Os nós com menor distância estimada, provavelmente os mais próximos do evento, têm maior peso, de forma que o peso do voto de um nó $n_{i,j}$ é dado por

$$w_{i,j} = e^{-d_{i,j}}, \tag{3.2}$$

em que $d_{i,j}$ é a distância estimada entre o nó $n_{i,j}$ e o alvo. Dessa forma, os nós mais próximos têm voto de maior peso, mesmo que a diferença de distância seja pequena.

Em aspectos gerais a votação ponderada é bastante similar a votação simples. Entretanto, calcular a posição do alvo classificando os nós por peso é mais preciso quando se utiliza raios de detecção maiores dos que a distância entre os nós. Quanto maior é o alcance do alvo maior é a quantidade de referências que confirmam que a posição está correta. Por outro lado, esse método está sujeito a imprecisão das estimativas de distância que pode resultar em uma distribuição imprecisa de pesos.

3.2.2 Prevendo a Posição do Alvo

Prever a posição do alvo consiste em determinar a posição (célula) em que o alvo estará no tempo $t + 1$. Os parâmetros usados para se realizar essa previsão são as posições do alvo até o tempo t. A figura 3.3 mostra um exemplo em que o algoritmo de rastreamento no tempo t prevê a célula $c_{1,5}$ como a posição do alvo no tempo $t + 1$.

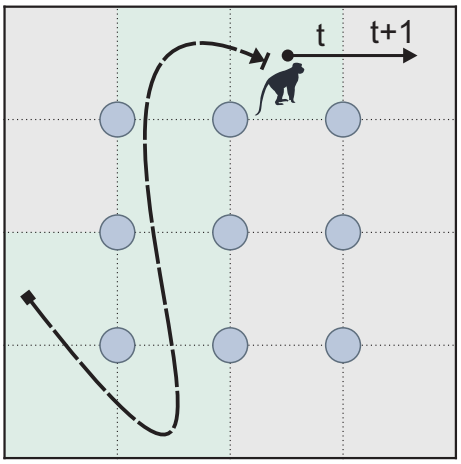


Figura 3.3. Previsão de posição do alvo.

Neste trabalho, utilizamos os filtros de Kalman e Partículas para realizar as previsões. O valor obtido por esses métodos na fase de previsão no tempo t é definido como a provável posição do alvo no tempo $t + 1$. Esses filtros são alimentados na fase de correção pelas posições estimadas com a votação simples ou ponderada.

Os filtros *Bayesianos* são métodos normalmente usados para trabalhar com dados contínuos, mas no caso deste trabalho, em que as coordenadas são representadas por valores inteiros e discretos, os filtros podem ser adaptados para executar apenas operações com valores inteiros (menor custo).

3.3 Avaliação

Nesta seção descrevemos a metodologia de avaliação e os resultados obtidos.

3.3.1 Metodologia

As avaliações são realizadas por meio de simulações com Sinalgo [Group, 2008]. Trata-se de um simulador para verificar algoritmos distribuídos, dessa forma abstrai as diferentes camadas de rede. Esse simulador é uma boa escolha para nossos experimentos, uma vez que estamos interessados apenas em avaliar a qualidade do rastreamento quantizado, dispensando outros aspectos específicos de redes.

As estimativas e previsões são centralizadas no nó *sink*, portanto cabe aos nós sensores somente enviar a ele os dados coletados. A configuração padrão da rede é composta de 2401 nós sensores distribuídos em grade em um campo de sensores de $500\times500\,\mathrm{m}^2$, além do *sink* e do alvo que são posicionados aleatoriamente. Dessa forma, os nós ficam a 10 m de distância uns dos outros, resultando em 50×50 células de $10\,\mathrm{m}^2$ cada.

Os alcances dos nós sensores e do *sink* são de 10 m. Em um cenário ideal isso possibilita que todos os nós se comuniquem através de múltiplos saltos. O roteamento dos dados é feito com uma árvore de roteamento em que o *sink* é a raiz. Já o alcance do alvo é de 30 m, permitindo que mais de três nós detectem o evento. A tabela 3.1 resume os parâmetros de simulação usados.

Tabela 3.1. Parâmetros de simulação usados nas avaliações do rastreamento quantizado.

Parâmetro	Valor
Campo de sensores	$500\times500\,\mathrm{m}^2$
Número de nós	2401
Alcance dos nós	10 m
Alcance do alvo	30 m

O modelo de mobilidade seguido pelo nó móvel é a Caminhada Aleatória Correlacionada com grau de correlação de 0,99 e velocidade de $1\,\mathrm{m/s}$. Agendamos 300 eventos (alvo divulgando sua presença) em intervalos 10 s. Nessa configuração, a trajetória do alvo cobre todo o campo de sensores, garantindo que o alvo divulgue sua posição quando mudar de célula. Para detectar o alvo, usamos o modelo de detecção binário (seção 2.2.1.2).

O *sink* calcula a posição do alvo por votação simples (ver seção 3.2.1.1) ou ponderada (ver seção 3.2.1.2). Já a previsão é feita usando os Filtros de Kalman ou

Partículas. O filtro de Kalman tem seu sistema de equações lineares configurado como

$$\begin{cases} x_{k+1} = \begin{bmatrix} px_{k+1} \\ py_{k+1} \\ vx_{k+1} \\ vy_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} px_k \\ py_k \\ vx_k \\ vy_k \end{bmatrix} + w_k \\ y_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} px_k \\ py_k \\ vx_k \\ vy_k \end{bmatrix} + z_k, \end{cases} \tag{3.3}$$

em que x representa o estado de um tempo discreto k formado pela posição (px, py) e velocidade (vx, vy); y é o valor de medição; w e v são os ruídos do processo e da medição, respectivamente.

O filtro de Partículas foi fixado em 1000 partículas, valor obtido em avaliações anteriores que mostraram que mais de 1000 partículas não reduz significativamente o erro do rastreamento. O filtro de Partículas usado nos experimentos é representado pelo algoritmo 3.1.

Para efeito de ilustração, o algoritmo apresentado considera apenas uma dimensão, sendo x a posição, v a velocidade e w o peso de cada uma das N partículas de um tempo discreto k; y é o valor de medição passado como entrada. Primeiramente, o algoritmo distribui as partículas aleatoriamente (linha 1). A propagação das partículas e o cálculo de suas importâncias consideram a distância da posição de cada partícula para a posição da medição (linhas 3–8). O processo de normalização (linha 9), prepara o peso das partículas para o processo de reamostragem (linhas 12–17). Finalmente é calculada a previsão da posição (linha 18).

Modelamos o consumo de energia de acordo com Yupho & Kabara [2007]. A tabela 3.2 mostra os parâmetros usados nas simulações. Consideramos que cada mensagem tem um tamanho 64 bytes.

Para calcular o erro das estimativas em células, considerando que c_{x_1,y_1} é a célula em que o alvo está e que c_{x_2,y_2} é a célula estimada, o erro é dado por

$$\epsilon\left(c_{x_1,y_1}, c_{x_2,y_2}\right) = max\left(|x_1 - x_2|, |y_1 - y_2|\right). \tag{3.4}$$

Os resultados dos experimentos são obtidos da média de 50 execuções diferentes. As barras de erro representam um intervalo de confiança de 99%.

input : medição y_k
output: previsão x_{k+1}
for $i = 1 : N$ **do** $x_0^i \leftarrow random()$;
$totalWeight \leftarrow 0$;
for $i = 1 : N$ **do**
    $x_k^i \leftarrow x_{k-1}^i + v_{k-1}^i + gaussian()$;
    $v_k^i \leftarrow v_{k-1}^i + gaussian() * 0.05$;
    $w_k^i \leftarrow \frac{1}{distance(y_k, x_k^i)}$;
    $totalWeight \leftarrow totalWeight + w_k^i$;
end
for $i = 1 : N$ **do** $w_k^i \leftarrow \frac{w_k^i}{totalWeight}$;
$slice_k^0 \leftarrow w_k^0$;
for $i = 2 : N$ **do** $slice_k^i \leftarrow slice_k^{i-1} + w_k^i$
for $i = 1 : N$ **do**
    $c \leftarrow random()$;
    $j \leftarrow 0$;
    while $j < N - 1$ *and* $slice_k^j < c$ **do** $j \leftarrow j + 1$;
    $resampling_k^i \leftarrow particle_k^j$;
end
for $i = 1 : N$ **do** $x_{k+1} \leftarrow x_{k+1} + x_k^i \times w_k^i$;
return x_{k+1};

Algoritmo 3.1: Filtro de Partículas para o rastreamento.

Tabela 3.2. Parâmetros de energia usados nas avaliações do rastreamento quantizado.

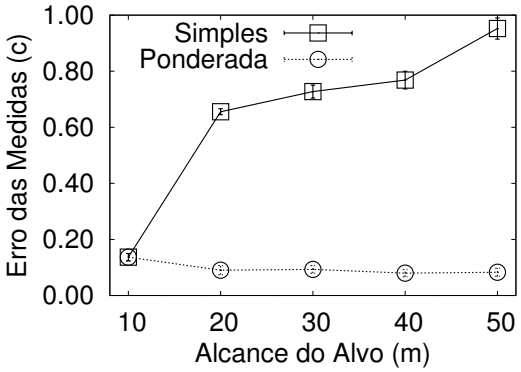
Parâmetro	Valor
Energia inicial	50 joules
Transmissão	342×10^{-7} joules/byte
Escutando	1888×10^{-8} joules/s

3.3.2 Resultados das Simulações

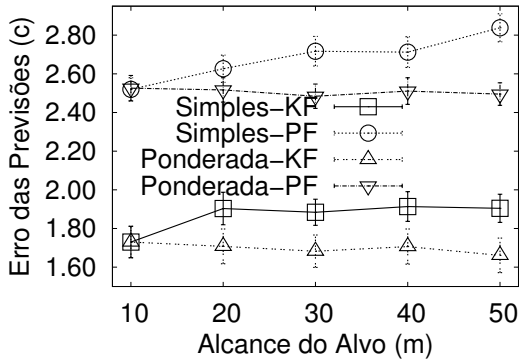
3.3.2.1 Alcance do Alvo

O alcance do alvo determina a quantidade de nós que podem detectar o evento. O ideal seria que um nó $n_{i,j}$ detectasse o evento sempre que o alvo estivesse em alguma das células ao seu redor. Entretanto, considerando que o evento é detectado com o modelo de raio fixo, o alvo deve ter um alcance próximo da distância entre os nós da grade para que, na maioria dos casos, o evento seja detectado por pelo menos três nós. Se menos de três nós detectam o alvo é impossível determinar com exatidão a célula em que o alvo está, mas é possível obter uma célula próxima.

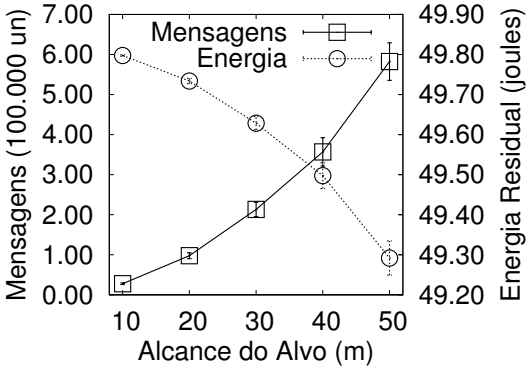
Desse modo, para avaliarmos como a quantidade de nós que participam da votação influenciam na precisão do rastreamento, variamos o alcance do alvo de 10 m a 50 m. O modelo de detecção de eventos usado é o raio fixo, então é possível executar o rastreamento satisfatoriamente com essa variação. Os resultados são mostrados na figura 3.4.



(a) Medidas.



(b) Previsões.



(c) Mensagens e Energia.

Figura 3.4. Influência do alcance do alvo.

Notamos na figura 3.4(a) que no cálculo de posição com alcance de 10 m, as votações simples e ponderada obtêm o mesmo resultado, pois o peso do voto não interfere no resultado com a baixa quantidade de nós que participam da votação. Quando o alcance do alvo aumenta há uma melhora sutil na votação ponderada, que só erra quando o alvo está nas bordas do campo de sensores. Já a votação simples é prejudicada com esse aumento, pois ocorrem muitos empates.

A figura 3.4(b) mostra as previsões combinando as técnicas de votação simples e ponderada com os filtros de Kalman e Partículas. Verificamos que o filtro tem mais influência na previsão do que o tipo de votação, embora as previsões reflitam os erros das medidas. Por ser a solução ótima, o filtro de Kalman alcança resultados mais

precisos que o filtro de Partículas.

Cada nó que detecta o evento gera uma mensagem que é encaminhada para o *sink*, onde é realizada a votação e a previsão. Por isso, quando o alcance do alvo aumenta mais mensagens são geradas na rede, incrementando o consumo de energia dos nós, como mostra a figura 3.4(c).

3.3.2.2 Detecção de Eventos

A eficiência de detecção depende de diversos fatores, como condições do ambiente e características do evento. Na avaliação anterior consideramos que o evento sempre é detectado pelos nós dentro do raio de alcance do alvo. Essa suposição é interessante para verificar o cenário ideal, mas é pouco realista.

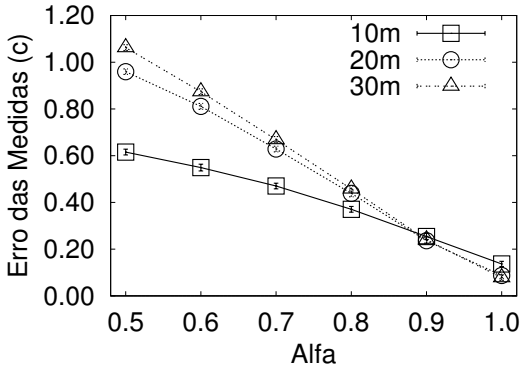
Por isso, nesta seção usamos o modelo de detecção de eventos flexível para verificar como nossa abordagem reage a falhas de detecção de eventos. Configuramos $\gamma = 2$, $\theta = 1000$ e variamos o parâmetro de precisão α de 0,5 a 1,0. Para que o alvo tenha alcance de 10 m, 20 m ou 30 m, usamos o valor de β como $\frac{1}{10}$, $\frac{1}{20}$ e $\frac{1}{30}$, respectivamente. Os resultados são mostrados na figura 3.5.

A figura 3.5(a) mostra que quanto menor o raio de alcance do alvo, menor é o erro das medidas[1], principalmente quando o valor de α é baixo (sensor impreciso). Entretanto, é errado concluirmos que configurar o alvo com um alcance pequeno é vantajoso, pois o erro das medidas se torna menor porque muitos eventos não são detectados, como mostra a figura 3.5(d). Nessa figura, 100% dos eventos são garantidamente detectados, independente do valor de α quando o alcance do alvo é de 30 m, para valores menores, eventos são perdidos.

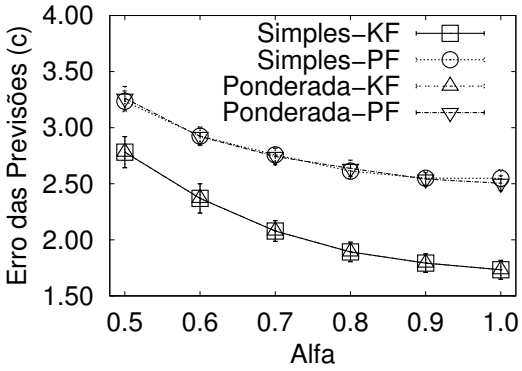
As figuras 3.5(b) e 3.5(c) mostram que o rastreamento é mais preciso a medida que a detecção de eventos é melhor, independente do alcance do alvo. Entretanto, com alcance de 30 m, o rastreamento é menos prejudicado, pois a divulgação do evento alcança mais nós, logo há uma probabilidade maior do evento ser capturado. Isso não ocorre quando o alcance é de 10 m, pois os eventos são perdidos com mais facilidade. Ainda podemos notar nessas figuras que, embora a votação ponderada e a votação simples tenham o mesmo resultado quando o alcance do alvo é de 10 m, a votação ponderada tem vantagem sobre a simples nos demais casos. Além disso, o KF tem vantagem sobre o PF, logo a combinação que tem melhor resultado é a votação ponderada com KF.

Tanto o alcance do alvo quanto a precisão da detecção de eventos têm influência

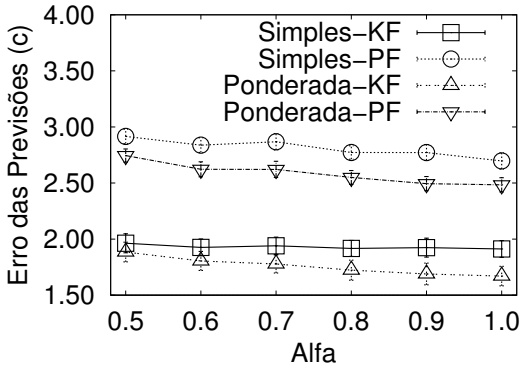
[1]Mostramos apenas os resultados da votação ponderada para simplificar o gráfico, pois ela apresentou melhores resultados que a votação simples.



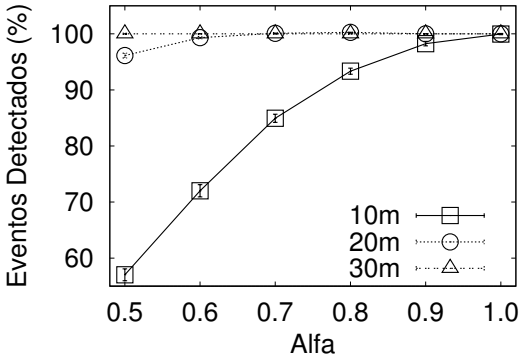
(a) Medidas obtidas com votação ponderada.



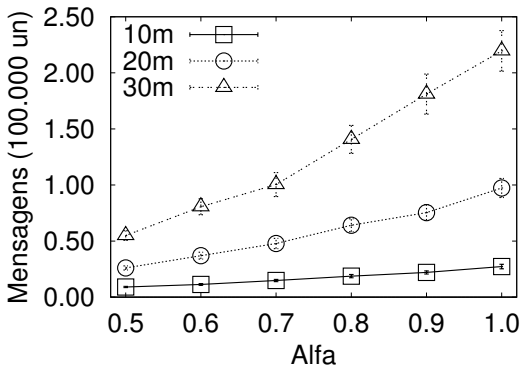
(b) Previsões com alcance do alvo de 10m.



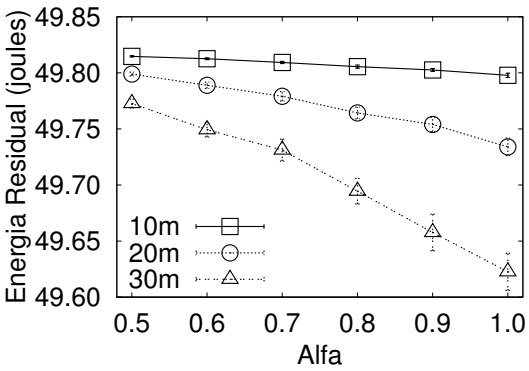
(c) Previsões com alcance do alvo de 30m.



(d) Percentagem de eventos detectados.



(e) Mensagens.



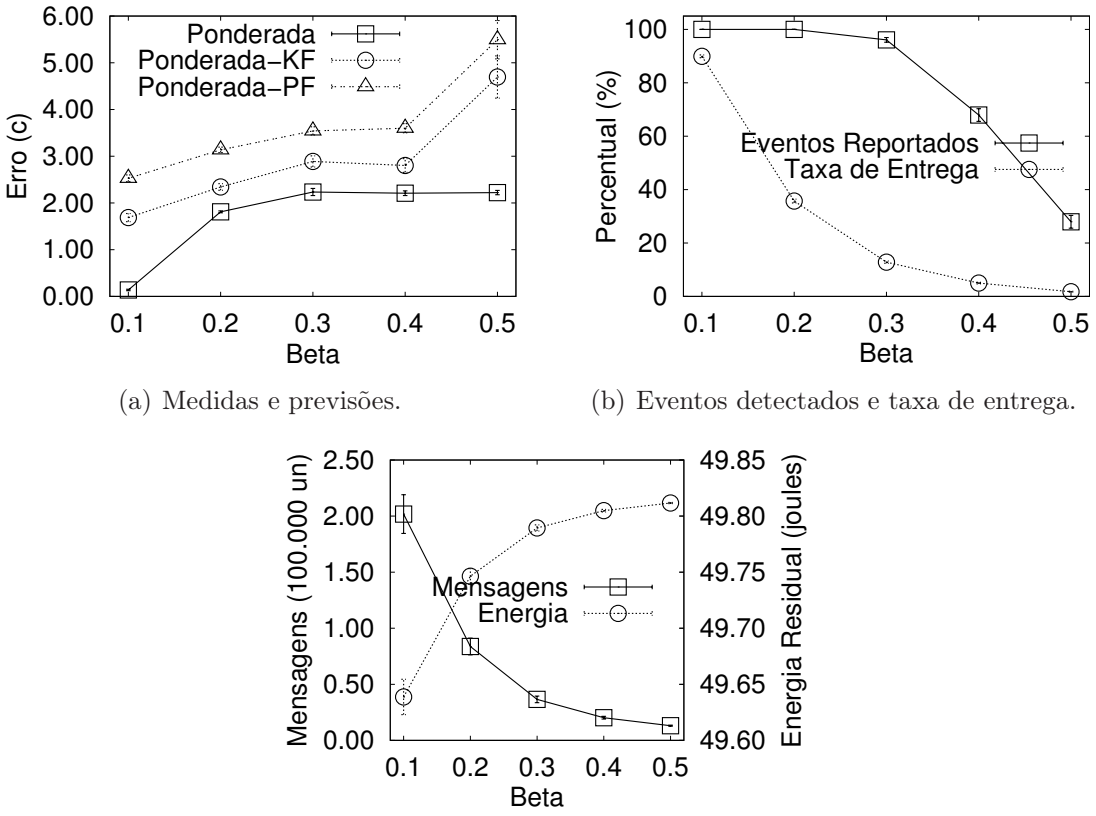
(f) Energia.

Figura 3.5. Influência das falhas de detecção de eventos.

na quantidade de mensagens enviadas pela rede e consequentemente no consumo de energia da mesma (figuras 3.5(e) e 3.5(f)). Quanto maior for o raio de alcance do alvo, maior é a quantidade de nós que detectam o evento, logo mais mensagens precisam ser enviadas, aumentado o consumo de energia. Além disso, a quantidade de mensagens e o consumo de energia diminuem quando a imprecisão da detecção de eventos aumenta, pois os nós que falham em detectar o evento não enviam mensagens.

3.3.2.3 Interferências

Nesta seção avaliamos como nosso modelo reage às interferências provocadas pelo ambiente. Para isso, usamos a relação sinal/ruído mais interferência (SINR – *Signal to Interference plus Noise Ratio*) [Gupta & Kumar, 2000]. Esse modelo assume que a intensidade de um sinal diminui exponencialmente com a distância. Três parâmetros precisam ser configurados nesse modelo: τ, parâmetro de perda pela distância; η, limiar usado para definir se o pacote chegou ou não; e ν, ruído do ambiente. Neste experimento fixamos $\tau = 2$, $\nu = 0$ e variamos η de 0 a 0,5. Os resultados são mostrados na figura 3.6.



(a) Medidas e previsões.

(b) Eventos detectados e taxa de entrega.

(c) Mensagens e energia.

Figura 3.6. Influência da interferência do ambiente.

A figura 3.6(a) mostra que o incremento de η afeta o desempenho da votação, mas o erro estabiliza quando $\eta \geq 0,3$. Isso ocorre porque, nesses casos, a quantidade de dados que chegam ao *sink* são insuficientes para gerar uma votação consistente, pois a taxa de entrega de dados é inferior a 20% (figura 3.6(b)).

O erro das medidas, gerado pela baixa taxa de entrega de dados, também interfere nas previsões do rastreamento, como mostra a figura 3.6(a). Entretanto, o aumento

do erro das previsões fica mais evidente na diferença entre 0,4 e 0,5 de η. Isso ocorre porque com $\eta = 0,5$ a quantidade de eventos detectados é inferior a 30% (figura 3.6(b)), tornando o trabalho dos filtros Bayesianos mais difícil, pois se formam lacunas entre os valores passados como medidas. O excesso dessas lacunas faz com que os filtros fiquem constantemente instáveis.

Por fim, a quantidade de mensagens e, consequentemente, a energia de toda a rede também são influenciadas pelas interferências do ambiente. Quanto maior for a interferência do ambiente menor é a quantidade de mensagens que são enviadas, como mostra a figura 3.6(c), pois alguns dos dados que deveriam chegar até o *sink* se perdem ao longo do processo de roteamento. Por esse mesmo motivo, a rede acaba usando menos energia.

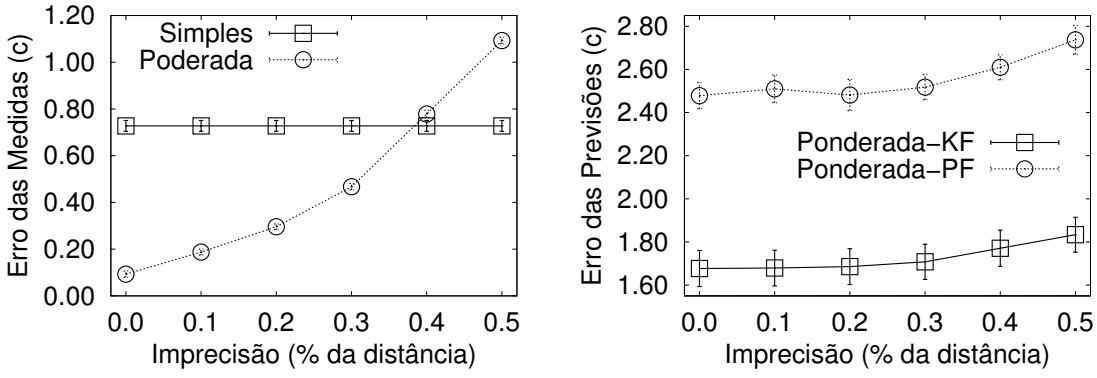
3.3.2.4 Imprecisão das Estimativas de Distância

A distância estimada pelos nós sensores não é perfeita. Os erros dessas estimativas podem ser altos dependendo do ambiente monitorado, afetando o desempenho do rastreamento. Em geral, esses erros dependem da distância e podem ser modelados como uma variável Gaussiana de média zero, em que o desvio padrão é um percentual da distância atual [Oliveira et al., 2009].

Na nossa abordagem, a votação ponderada depende das estimativas de distância. Desse modo, para avaliarmos diferentes situações, variamos o desvio padrão de 0% a 50% da distância. O desvio padrão de 0% resulta em estimativas de distância perfeitas, 10% da distância corresponde às estimativas obtidas por técnicas baseadas no tempo como Tempo de Chegada do Sinal (TOA – *Time Of Arrival*), que tem erros menores que 1 m. As demais situações representam as estimativas obtidas por métodos mais imprecisos como Intensidade do Sinal Recebido (RSSI – *Received Signal Strength Indicator*). Os resultados são mostrados na figura 3.7.

Notamos na figura 3.7(a) que a votação ponderada erra mesmo quando as estimativas de distância são perfeitas. Essas falhas ocorrem somente se o alvo estiver nos limites do campo de sensores, pois nessa situação o alvo não fica cercado pelos nós da grade. O aumento da imprecisão das estimativas de distância afeta diretamente o desempenho da votação ponderada, já que os pesos de votação acabam sendo distribuídos de forma incorreta.

As estimativas também são afetadas com esse aumento no erro das medidas, mas em menor escala (figura 3.7(b)). A diferença do erro das medidas entre as imprecisões de 0% e 50% é de aproximadamente 1 célula, mas essa mesma diferença nas estimativas é de aproximadamente 0,15 células. Isso acontece porque os filtros reduzem a influência



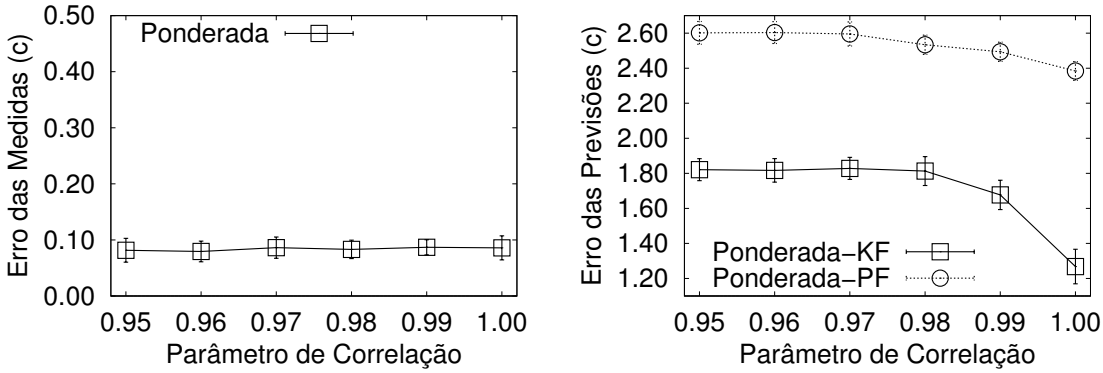
(a) Medidas. (b) Estimativas.

Figura 3.7. Influência da imprecisão das estimativas de distância.

dos erros das medidas.

3.3.2.5 Correlação da Mobilidade do Alvo

A maneira como o alvo se move é uma característica importante que deve ser considerada pelas aplicações de rastreamento, pois a correlação da mobilidade influencia no desempenho dos estimadores. Nesta avaliação variamos o parâmetro de correlação (ω) de 0,95 a 1,00. Os resultados são mostrados na figura 3.8.



(a) Medidas. (b) Estimativas.

Figura 3.8. Influência da correlação de mobilidade do alvo.

A figura 3.8(a) mostra que a votação não é influenciada pela correlação da mobilidade, pois para esse processo não importa como o alvo se move, mas sim onde ele está no momento em que divulga sua posição. Já a figura 3.8(b) mostra que quanto mais correlacionada for a mobilidade menor será o erro das previsões. Isso ocorre porque uma correlação baixa faz o alvo realizar mais manobras, levando o filtro a errar até estabilizar novamente. Quando a mobilidade é 100% correlacionada, os filtros erram

apenas quando o alvo chega no limite do campo de sensores, sendo obrigado a manobrar para ficar dentro desse campo.

3.4 Conclusões Parciais

Neste capítulo propomos e avaliamos uma técnica quantizada de rastreamento em redes de sensores com uma estrutura em grade. Para calcular a posição do alvo são usadas duas técnicas de votação: a simples, em que todos os nós têm o mesmo peso; e a ponderada, em que os nós mais próximos do alvo têm maior peso. As previsões são realizadas com o filtro de Kalman ou Partículas, alimentados com as coordenadas das células obtidas pela votação.

A votação simples é uma boa opção quando o raio de alcance do alvo é igual a distância entre os nós da grade. Entretanto quando consideramos cenários em que ocorrem falhas de comunicação, a votação ponderada obtém melhores resultados, mesmo nas situações em que a imprecisão das estimativas de distância são altas. Além disso, o filtro de Kalman mostra-se mais eficiente que o filtro de Partículas independente do cenário adotado. São necessários aproximadamente quatro nós envolvidos na votação para determinar com exatidão a célula em que o alvo está, portanto é importante que o alcance do alvo seja configurado para cobrir mais nós, afim de evitar que alguns eventos sejam perdidos.

Capítulo 4

PRATIQUE: Um Algoritmo Hierárquico para Rastreamento de Alvos em Áreas Quantizadas para Redes de Sensores

No capítulo anterior, propomos e avaliamos a utilização das células formadas por uma rede de sensores organizada em grade como as posições do alvo. Entretanto, essa avaliação foi feita de forma centralizada, em que todos os dados coletados pela rede são encaminhados para o nó *sink*, que por sua vez realiza a fusão de dados para gerar o resultado do rastreamento.

Neste capítulo, mantemos essa abordagem de discretizar o campo de sensores, por outro lado considerando mais o aspecto distribuído das redes de sensores. Para isso, desenvolvemos o *Prediction-based clusteRing Algorithm for TrackIng targets in QUantized arEas for wireless sensor networks* (PRATIQUE). Trata-se de um algoritmo de rastreamento para rede de sensores baseado em agrupamento e previsão. Portanto, esse busca determinar a posição atual de um dado alvo e também estimar sua posição futura. A posição do alvo é dada como uma região do ambiente monitorado. Para isso, a rede é organizada em grade, onde as células formadas nesse arranjo são as regiões de ocupação do alvo.

Estruturas de agrupamento (seção 2.2.2.2) são frequentemente usadas em redes de sensores para unir os dados coletados em um nó próximo do evento para reduzir o custo de comunicação. Esse nó é responsável pela fusão desses dados e por notificar o resultado obtido. As estruturas de agrupamento mais simples, chamadas de agrupamentos estáticos, dividem o campo de sensores em regiões antes da aplicação

ser executada. Os nós que estão localizados dentro da mesma região fazem parte do mesmo agrupamento e têm o mesmo líder. O ponto fraco dos agrupamentos estáticos é o problema da fronteira [Wang et al., 2010b], quando o alvo está posicionado na fronteira entre agrupamentos, ele pode ser detectado por mais de um agrupamento. Dessa forma, os dados coletados são divididos, afetando a qualidade do resultado e aumentando notificações redundantes. Os agrupamentos dinâmicos são alternativas mais robustas que evitam esse problema, entretanto possuem um custo maior para formar o agrupamento e escolher um novo líder de acordo com a mobilidade do alvo.

Para evitar a dispersão de dados, causados pelos agrupamentos estáticos, e a comunicação excessiva, necessária para a criação de agrupamentos dinâmicos, o PRATIQUE usa um esquema híbrido de agrupamento que aproveita as vantagens desses dois tipos de agrupamento. Nesse esquema, a rede é dividida em setores logo depois que a rede é implantada. Os nós que estão localizados dentro de um mesmo setor fazem parte do mesmo agrupamento estático, sendo que um deles é eleito como líder. O agrupamento dinâmico é formado somente por líderes de agrupamento e só é criado quando o evento cobre mais de um agrupamento estático, sua função é unir em um único nó os dados dispersos em diferentes líderes de agrupamento.

Quando um evento ocorre, cada nó que detecta o evento gera uma posição prévia para o alvo, essa é imprecisa pois baseia-se na observação de apenas um nó isolado. Esses encaminham suas posições para o líder do seu agrupamento através de múltiplos saltos. A cada salto é feita a fusão dos dados de posição. Até esse momento, somente o agrupamento estático foi utilizado. Seu objetivo é melhorar a precisão da posição e reduzir o custo de comunicação. No entanto, um evento pode ocorrer entre dois ou mais agrupamentos, nesse caso cada líder terá apenas dados de posição parciais. Para obter um resultado mais preciso, é necessário que a fusão de todos os dados seja feita em um único nó. Para resolver esse problema é criado um agrupamento dinâmico formado somente pelos líderes envolvidos no evento. Um deles é escolhido para receber todos os dados sobre o evento, calcular a posição do alvo e prever sua próxima posição.

Para rastrear múltiplos alvos, cada mensagem possui um vetor indexado pelos eventos. Assim, se mais de evento é detectado pelo mesmo nó em um curto espaço de tempo, os eventos podem ser identificados e tratados isoladamente. Além disso, apenas uma mensagem pode ser usada para vários eventos.

No PRATIQUE, a previsão, além de ser um resultado, é usada pela rede para atualizar o estado do filtro responsável pelo rastreamento. A cada previsão, o estado do filtro é modificado apenas no nó que realizou essa ação. Dessa forma, o estado do filtro precisa ser atualizado no próximo nó que realizará a previsão. A forma mais simples e segura é atualizar os líderes de todos os agrupamentos. Entretanto, essa solução exige

um custo de comunicação alto, por isso usamos a previsão para atualizar os líderes dos agrupamentos próximos da posição prevista. As previsões são feitas com *Kalman Filter* (KF) [Wang et al., 2012], *Particle Filter* (PF) [Li & Xu, 2010] ou *Alpha-Beta Filter* (ABF) [Sharma et al., 2011a].

As soluções e resultados deste capítulo estão parcialmente publicados no Simpósio Brasileiro de Computação Ubíqua e Pervasiva [Souza et al., 2013]. Uma versão estendida está em avaliação no *Wireless Networks*.

No restante deste capítulo descrevemos com mais detalhes o funcionamento do PRATIQUE e avaliamos o seu desempenho em diferentes cenários.

4.1 Fases do PRATIQUE

Esta seção descreve o funcionamento do PRATIQUE. Esse algoritmo pode ser dividido em sete fases: configuração, anúncio de borda, encaminhamento de dados, formação de agrupamento dinâmico, posicionamento/previsão, sincronização de filtro e notificação de resultado. A fase de configuração é executada uma única vez durante a implantação da rede. As outras fases ocorrem em sequência após o alvo divulgar sua presença. A seguir explicamos o funcionamento de cada uma dessas fases.

4.1.1 Configuração

Os nós são divididos em agrupamentos estáticos assim que a rede é implantada, antes que o alvo comece a divulgar sua posição e o rastreamento seja iniciado. Para isso, o nó *sink* inicia um *flooding* na rede com uma mensagem de configuração. O nó que recebe essa mensagem se configura e repassa a mensagem.

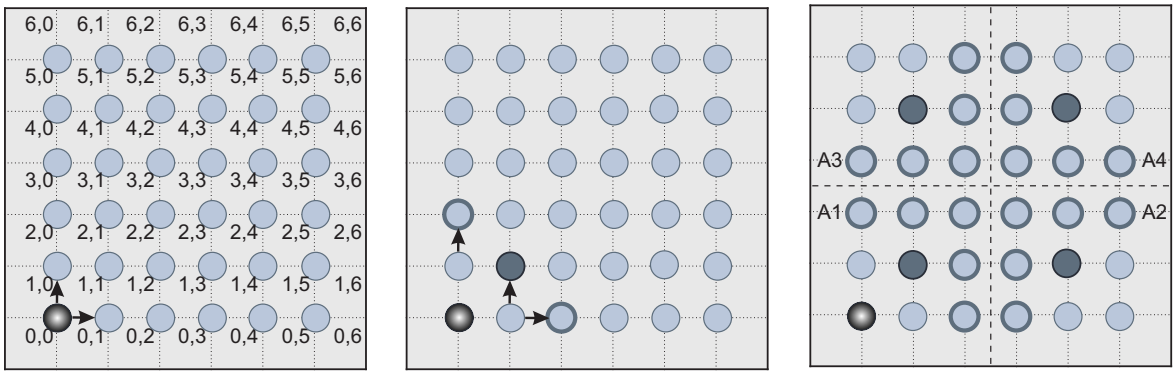
Em um agrupamento, um nó pode ser classificado como Membro (*Cluster Member* (CM)), Borda (*Cluster Border* (CB)) ou Líder (*Cluster Head* (CH)). Um CM é qualquer nó que faz parte de um agrupamento, sua função é detectar o alvo e enviar essa informação até seu CH. Um CB é o nó que está localizado na fronteira entre agrupamentos, sua função é avisar ao CH se existe outro agrupamento participando de um mesmo evento. Por fim, um CH é o nó que recebe e faz a fusão de dados de todos os nós do seu agrupamento, por enquanto consideramos que o CH é o nó mais próximo do centro do agrupamento. Vale ressaltar que os CBs e CHs também são CMs. Maiores detalhes das funções de cada classe de nós são mostrados nas próximas fases.

A figura 4.1 mostra como funciona o processo de configuração. Os nós são arranjados em forma de grade e cada um deles já conhece a priori o seu índice nesse arranjo. As linhas pontilhadas dividem as células que irão representar as posições do alvo e as

linhas tracejadas dividem os nós de acordo com o agrupamento estático ($A1$, $A2$, $A3$ e $A4$) a que pertencem. Os índices da figura 4.1(a) servem para os nós e para as células, o nó que possui o mesmo índice de uma célula é chamado de nó referência da célula. O *sink* é um dos nós da grade, no caso do exemplo é o nó $n_{0,0}$, mas poderia ser qualquer um dos outros nós. O *sink* inicia a configuração da rede enviando uma mensagem para seus vizinhos.

A mensagem de configuração contém o índice (i, j) do *sink*, as dimensões dos agrupamentos, as dimensões das células e o raio de alcance nominal dos nós. Esses dados aliados com o índice do próprio nó são suficientes para que ele determine uma rota para o *sink*, ajuste a potência de transmissão, calcule o seu agrupamento e se classifique dentro do mesmo. No caso da figura 4.1(b), o nó $n_{1,1}$ está na posição para ser um CH, então ele já se classifica como tal antes de repassar a mensagem.

A mensagem de configuração é repassada até que toda a rede seja configurada. A figura 4.1(c) mostra o estado da rede depois que a fase de configuração é concluída. As linhas tracejadas dividem os agrupamentos estáticos.



(a) O *sink* inicia um *flooding*.

(b) Os nós que recebem a mensagem se classificam e passam a mensagem adiante.

(c) A rede está configurada. Os nós determinaram seus agrupamentos e sua classificação.

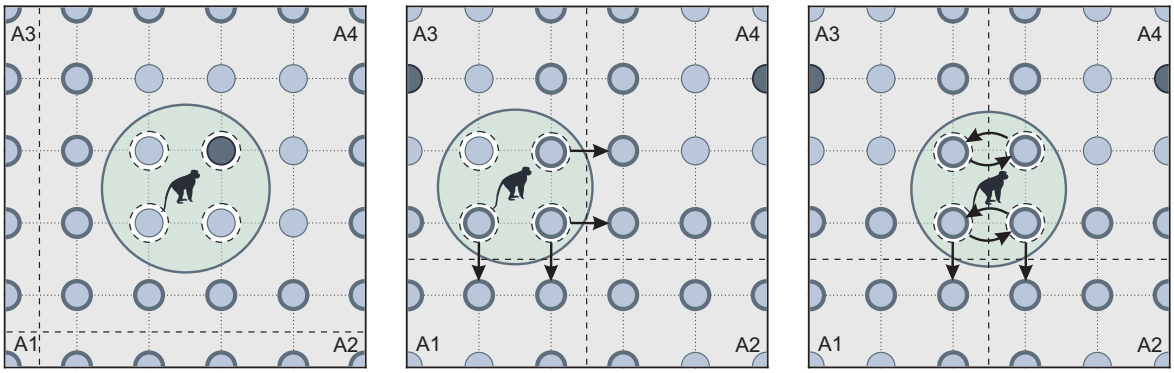
Figura 4.1. Configuração da rede antes do rastreamento iniciar.

4.1.2 Anúncio de Borda

Quando ocorre um evento, os CBs são os primeiros a executar suas tarefas. O objetivo dos CBs é fazer a comunicação entre agrupamentos. Se um evento se estende por mais de um agrupamento, ele obrigatoriamente é detectado pelos CBs. Essa comunicação é importante quando o evento cobre mais de um agrupamento, pois assim todos os CHs que estão participando do evento passam a conhecer todos os outros agrupamen-

tos envolvidos no mesmo evento. O anúncio de borda é usado para criar o chamado agrupamento dinâmico (seção 4.1.4) e unir todos os dados coletados em um único nó.

Dependendo da posição e da abrangência do evento, o anúncio de borda pode não ocorrer ou ocorrer de forma incompleta. A figura 4.2 mostra três possibilidades de como o anúncio de borda pode ocorrer. No primeiro caso (figura 4.2(a)), o anúncio de borda não existe, pois o evento ocorre inteiramente dentro do agrupamento $A4$, sem alcançar os CBs. No segundo caso (figura 4.2(b)), o evento continua contido em um único agrupamento, porém alcança os CBs. Os CBs que detectam esse evento fazem um *broadcast* para anunciar aos CBs do agrupamento vizinho que o evento ocorreu no seu agrupamento. Por outro lado, como o evento cobriu apenas o agrupamento $A3$, os CBs dos agrupamentos vizinhos $A1$ e $A4$ ignoram a mensagem. No último caso (figura 4.2(c)), o evento cobre os agrupamentos $A3$ e $A4$, os CBs que detectam o evento divulgam a mensagem de borda que é recebida somente pelos CBs de agrupamentos diferentes que também detectaram o evento. Os CBs dos agrupamentos $A1$ e $A2$ recebem a mensagem, mas não detectam o evento, por isso ignoram a mensagem. Com isso, ocorre uma troca de mensagens somente entre agrupamentos que detectaram o evento.



(a) O evento cobre um único agrupamento. Não há anúncio de borda.

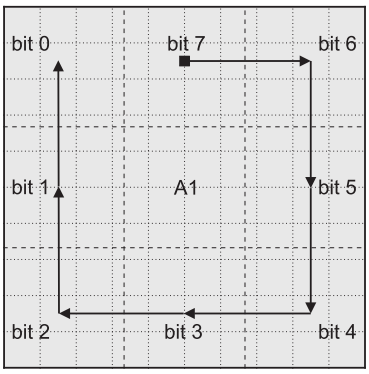
(b) O evento cobre um agrupamento, mas no limite do mesmo. O anúncio de borda ocorre, mas sem efeito, pois os nós de borda que não detectam o evento ignoram a mensagem.

(c) O evento cobre dois agrupamentos. Existe troca de mensagens entre os nós de borda de agrupamentos diferentes que detectaram o evento.

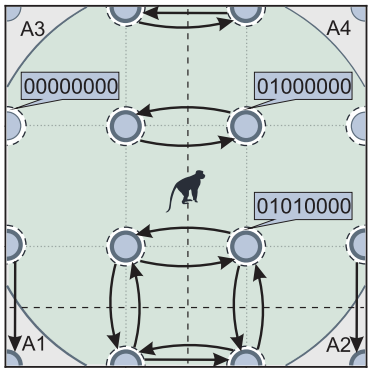
Figura 4.2. Três possibilidades de anúncio de borda.

Cada nó que detecta o evento gera um byte de agrupamentos participantes. Esse byte diz ao CH do agrupamento que está participando do evento se há outros agrupamentos participando do mesmo evento e quais são eles. Para isso, cada bit representa um agrupamento ao redor do agrupamento em questão. A figura 4.3(a) mostra nove agrupamentos que compõem uma rede. Observando o agrupamento central $A1$, cada

agrupamento tem no máximo mais oito agrupamentos ao seu redor. Esses são identificados pelos bits, iniciando do agrupamento vizinho superior, seguindo o sentido horário até o último agrupamento no canto superior esquerdo.



(a) Sequência dos agrupamentos representados por cada *bit* no *byte* de agrupamentos participantes.



(b) O valor do *byte* de agrupamentos participantes depende da troca de informação entre os CBs.

Figura 4.3. Organização do *byte* de agrupamentos participantes.

O valor 1 em um desses bits indica que o agrupamento representado por aquela posição também está participando do evento e o valor 0 indica o contrário. Inicialmente todos os bits possuem valor 0, mas eles podem mudar para 1 quando há troca de mensagens entre os CB de agrupamentos diferentes. A figura 4.3(b) mostra o byte de agrupamentos participantes de três nós na situação em que o evento ocorre no limite dos agrupamentos $A1$, $A2$, $A3$ e $A4$. O byte do agrupamento $A3$ pertence a um CM que detectou o evento, todos os bits têm valor 0, pois ele não recebe o anúncio de borda, por outro lado ele pode mudar com a fusão de dados que ocorre quando os dados sobre o evento forem encaminhados ao CH, mas isso ocorre na próxima fase. O byte do agrupamento $A4$ mais acima pertence a um nó que recebeu uma mensagem de borda de um nó do agrupamento $A3$, então o valor do sétimo bit muda para 1. Ainda no agrupamento $A4$, o byte mais abaixo pertence a um nó que recebem mensagens de borda de nós dos agrupamentos $A2$ e $A3$, logo muda para 1 o valor do quinto e do sétimo bit.

Depois que o anúncio de borda termina, o byte de agrupamentos participantes de cada nó é enviado ao seu CH na mesma mensagem usada para enviar os dados coletados sobre o evento. A cada salto, o nó que recebe essa mensagem realiza a fusão dos seus dados com os dados recebidos. A fusão feita com o byte de agrupamentos participantes é o ou bit-a-bit. Assim, no final o CH terá um único byte que lhe informa todos os agrupamentos que estão participando do evento.

4.1.3 Encaminhamento de dados

Assim que os nós detectam um evento, eles geram uma informação de posição prematura, já que um nó isolado só pode afirmar que o alvo está em uma das quatro células ao seu redor (células candidatas do nó). Os índices dessas células são usados para gerar essa posição prematura. A medida que os dados vão sendo encaminhados ao CH, a posição vai sendo refinada, pois nesse processo ocorre a fusão de todos os dados coletados pelos nós.

Os nós podem atribuir pesos as suas posições prematuras, dessa forma se um nó tem uma posição mais relevante, ela terá uma influência maior na posição final. Esses pesos podem ser constantes (valor 1, por exemplo) ou baseados na potência do sinal recebida do alvo. No primeiro caso, todas as posições têm a mesma relevância, então o resultado da fusão de dados sempre será a célula mais central dentre todas as células candidatas. No segundo caso, a potência do sinal indica os nós que estão mais próximos ao evento, então as células candidatas desses nós ficam com um peso maior. Por isso, o resultado da fusão é a célula que se encontra na região de maior importância.

Os dados coletados devem ser enviados ao CH, que é responsável pela fusão de todos os dados do agrupamento. Para evitar redundância de dados, reduzir o tráfego da rede e evitar colisões, esse envio é feito com base no número de saltos do nó até o CH. O agrupamento é dividido em camadas de transmissão, onde cada camada é formada pelos nós que têm a mesma distância em saltos. As camadas mais distantes sempre enviam os dados antes das camadas mais próximas. A cada salto, o nó que recebe a mensagem realiza a fusão dos seus dados e repassa o resultado. A fusão dos dados de posição é realizada usando o algoritmo 4.1.

```
1 global x_c, y_c, w_c;
  input: x_i, y_i, w_i
2 begin
3     w_s ← w_c + w_i;
4     x_c ← x_c × w_c/w_s + x_i × w_i/w_s;
5     y_c ← y_c × w_c/w_s + y_i × w_i/w_s;
6     w_c ← w_s;
7 end
```

Algoritmo 4.1: Fusão dos dados de posição.

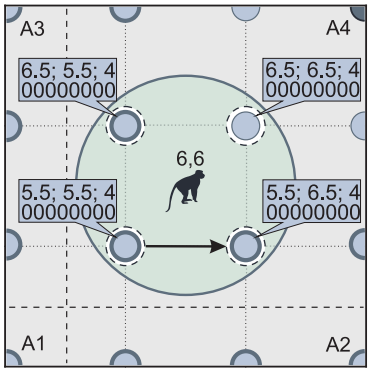
A linha 1 desse algoritmo mostra a declaração das variáveis que representam a posição prematura do alvo. A entrada passada para esse algoritmo também é uma posição e um peso (linha 2). Essa entrada pode ser os índices das células candidatas do nó ou então índices que são recebidos de outros nós, que passam pelo processo de fusão

a cada salto até chegar ao CH. As linhas 3–7 mostram como os dados da posição do alvo são atualizados. A posição com maior peso tem mais influência no resultado final. Uma mesma célula pode ser candidata de vários nós, aumentando sua importância. Em um mesmo nó, esse algoritmo pode ser executado várias vezes para atualizar os dados de um mesmo evento.

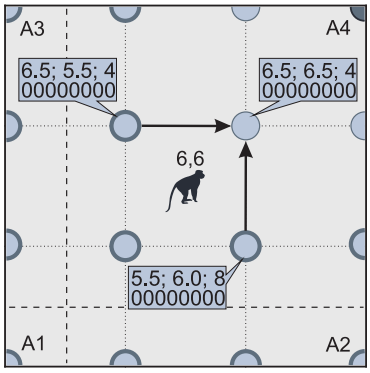
A figura 4.4 mostra o encaminhamento dos dados até o CH quando o evento cobre apenas um agrupamento. O alvo está localizado na célula $c_{6,6}$ e o evento alcança apenas os CBs do agrupamento em questão, por isso, após o anúncio de borda, todos os bits dos bytes de agrupamentos participantes dos nós têm valor 0. Além desse byte, os nós que detectam o evento também possuem a posição do alvo e o peso dessa informação. Na figura 4.4(a), cada nó que detecta o evento usa o algoritmo 4.1 para calcular a posição prematura do alvo, passando como entrada as quatro células ao seu redor. Todos os nós fazem cálculos imprecisos sobre a posição do alvo, pois até o momento fazem isso isoladamente. Para esse exemplo, consideramos que os nós atribuem peso contante 1 para cada célula, portanto o valor do peso no primeiro momento é quatro. Depois disso, o nó mais distante do CH (quatro saltos de distância) passa seus dados para um outro nó mais próximo do CH. Na figura 4.4(b), esse nó, por sua vez, faz a fusão dos seus dados com os dados recebidos, usando o algoritmo 4.1, e posteriormente encaminha esses dados para o próximo nó mais próximo do CH, juntamente com o outro nó que possui a mesma distância em saltos. Na figura 4.4(b), o nó recebe e faz a fusão de todos os dados do agrupamento, portanto já estima a posição do alvo com precisão e encaminha os dados até o CH.

A figura 4.5 mostra o encaminhamento dos dados quando o evento ocorre entre quatro agrupamentos, mas é possível visualizar o processo somente nos agrupamentos $A3$ e $A4$. Os bytes de agrupamentos participantes têm valor diferente de nulo, pois há troca efetiva de mensagens durante o anúncio de borda. A figura 4.5(a) mostra os nós mais distantes dos CHs de seus respectivos agrupamentos passando seus dados para os nós mais próximos. Na figura 4.5(b), os nós que recebem esses dados fazem a fusão dos dados de posição usando o algoritmo 4.1 e a fusão dos bytes de agrupamentos participantes usando ou bit-a-bit. Nesse ponto já ocorreu a fusão de todos os dados em cada agrupamento, então os dados continuam sendo encaminhados até os respectivos CHs (figura 4.5(c)).

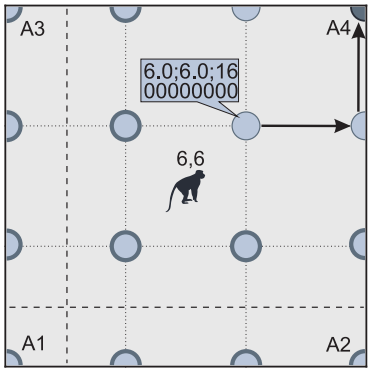
No exemplo da figura 4.5, os dados sobre um mesmo evento foram divididos entre quatro agrupamentos. Dessa forma, há quatro CHs com dados parciais sobre o evento que notificariam quatro resultados diferentes e imprecisos. Para evitar esse problema, um agrupamento dinâmico é formado para unir esses dados.



(a) Quatro nós do agrupamento $A4$ detectam o alvo e cada um, isoladamente, gera dados sobre a posição do alvo. O nó mais distante do CH é o primeiro a passar seus dados para um nó mais próximo.

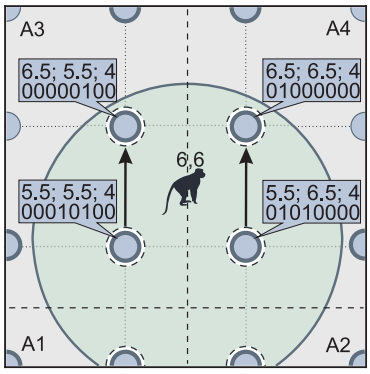


(b) O nó recebe os dados e faz a fusão deles com os dados que ele já possui. Depois disso passa seus dados para o seu vizinho mais próximo do CH. Nesse mesmo tempo, o outro nó que possui a mesma distância em saltos também passa os dados.

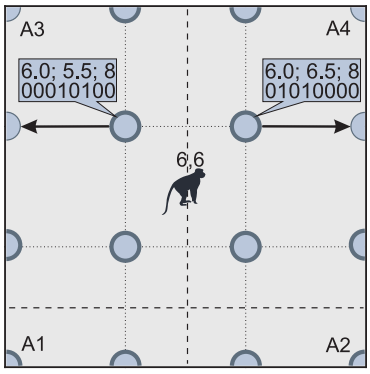


(c) Um nó recebe e faz a fusão de todos os dados do agrupamento. Seus dados são repassados até alcançar o CH.

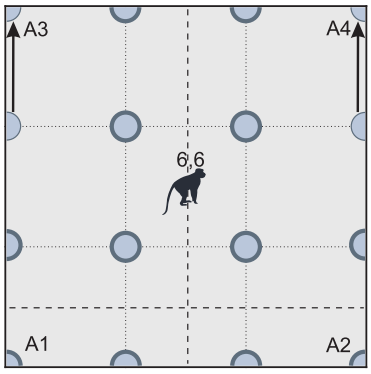
Figura 4.4. Encaminhamento dos dados quando o evento cobre apenas um agrupamento.



(a) O evento cobre quatro agrupamentos. Cada nó gera seus dados sobre o evento. Os nós mais distantes dos CHs em seus respectivos agrupamentos passam seus dados para um nó mais próximo.



(b) Os nós que recebem os dados fazem a fusão dos seus dados com os dados recebidos.



(c) Os dados de cada agrupamento são enviados para seus respectivos CHs.

Figura 4.5. Encaminhamento dos dados quando o evento cobre mais de um agrupamento.

4.1.4 Formação de Agrupamento Dinâmico

No fim da fase anterior, o CH recebe todos os dados do seu agrupamento, mas um mesmo evento pode ocorrer entre dois ou mais agrupamentos. Nesse caso, temos dife-

rentes CHs com dados parciais sobre o evento. Para unir os dados em um único nó, criamos um novo agrupamento, esse é dito dinâmico porque sua formação depende da posição e a abrangência do evento. O agrupamento dinâmico é formado somente pelos CHs dos agrupamentos que o evento cobriu, dentre eles um é eleito como *Super Cluster Head* (SCH) – líder do agrupamento dinâmico. Os outros CHs enviam seus dados de posição para o SCH.

O byte de agrupamentos participantes, gerado na fase de anúncio de borda, é usado na criação do agrupamento dinâmico. A distância em saltos do CH até o sink é o critério usado para escolher o SCH, dessa forma se torna SCH o CH mais próximo ao sink dentre todos os CHs envolvidos no evento. Após coletar os dados do seu agrupamento, cada CH envolvido no evento analisa seu byte para determinar localmente se ele é o SCH do agrupamento dinâmico. Caso o evento tenha cobrido somente um agrupamento, o CH desse agrupamento é automaticamente eleito como SCH. Depois disso, o SCH aguarda a mensagem dos outros nós do agrupamento dinâmico e une todos os dados coletados sobre o evento usando o algoritmo 4.1.

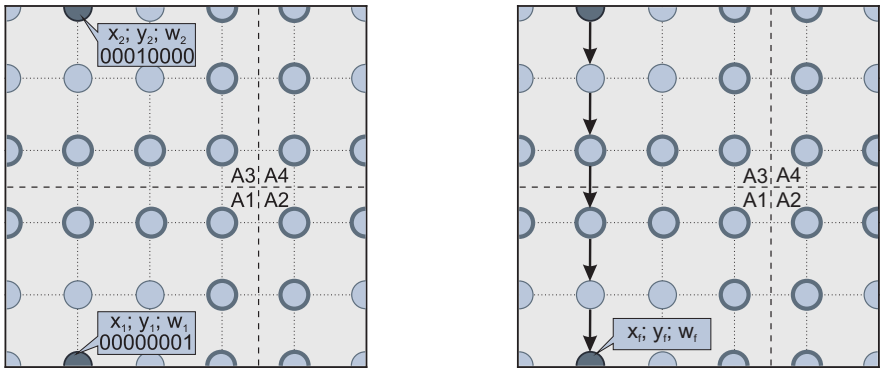
Na figura 4.6 o evento ocorreu entre os agrupamentos $A1$ e $A3$. Dessa forma, cada CH possui uma posição diferente do alvo que foi gerada a partir dos dados coletados pelos nós do seu agrupamento. Na figura 4.6(a), o CH do agrupamento $A1$ possui um byte que aponta que o agrupamento $A3$ também está participando do mesmo evento, logo ele verifica quem está mais próximo do sink, ele próprio ou o CH do agrupamento $A3$. Considerando que o nó $n_{0,0}$ é o sink, ele se intitula o SCH do agrupamento dinâmico, pois está mais próximo do sink. Já o CH do agrupamento $A3$ possui um byte que aponta que o agrupamento $A1$ também está participando do evento. Ao verificar quem está mais próximo do sink, ele nota que o CH do agrupamento $A1$ está mais próximo, então o considera como SCH. Depois disso (figura 4.6(b)), o SCH aguarda pelos dados de posição do alvo que serão enviados pelos outros membros do agrupamento dinâmico. No final desse processo, ele terá coletado todos os dados sobre o evento e faz a fusão de todos eles usando o algoritmo 4.1.

Caso o evento ocorresse dentre de um único agrupamento, o byte do CH desse agrupamento seria nulo, logo ele notaria que é o único agrupamento participando do evento e se intitularia SCH.

Desse ponto em diante, o SCH fica encarregado pelas próximas fases do algoritmo.

4.1.5 Posicionamento/Previsão

O SCH coleta todos os dados sobre o evento. Como a fusão desses dados é feita iterativamente à medida que esse são encaminhados, o SCH já possui a melhor estimativa



(a) Os CHs dos agrupamentos $A1$ e $A3$ coletaram os dados dos seus respectivos agrupamentos. Ambos calculam a posição do alvo de forma imprecisa, pois possuem dados parciais sobre o evento.

(b) Com base no *byte* de agrupamentos participantes o CH do agrupamento $A1$ é escolhido como SCH. Dessa forma, o CH do agrupamento $A3$ envia seus dados para o SCH, que faz a fusão final dos dados.

Figura 4.6. Formação de agrupamento dinâmico e fusão de todos os dados sobre um evento.

de posição do alvo no final da coleta. Ele utiliza a estimativa de posição mais recente juntamente com o histórico de estimativas realizados anteriormente pela aplicação para prever a próxima posição do alvo. A figura 4.7 mostra que no tempo t, a aplicação estima a posição atual do alvo e prevê a posição do alvo no tempo $t + 1$.

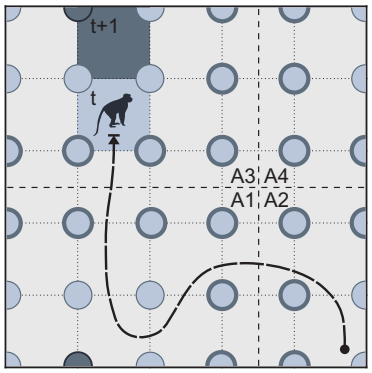


Figura 4.7. Previsão de posição com base no histórico de mobilidade do alvo.

A previsão da próxima posição do alvo é uma informação valiosa. Ela é útil tanto para o usuário da aplicação quanto para a aplicação em si. O usuário da aplicação pode, por exemplo, usar essa informação para interceptar o animal rastreado que se dirige para uma área de risco, como estradas dentro de uma reserva ecológica. Para a aplicação, essa informação pode ser usada na preparação dos nós próximos da região prevista para o próximo evento.

As previsões são calculadas com o KF, PF ou ABF. Vimos anteriormente que que esses filtros possuem duas fases: previsão e correção. Na fase de previsão o filtro utiliza os dados existentes até o momento atual para fazer uma previsão. Já na fase de correção, o filtro é atualizado, i.e. novos dados são incluídos para dar suporte às previsões posteriores. Esse ciclo é mantido a cada nova previsão. Dessa forma, a posição estimada do alvo é usada como medida na etapa de correção do filtro, que por sua vez retorna a previsão da próxima posição do alvo.

Quando um SCH prevê a posição de um alvo pela primeira vez, ele cria um filtro específico para esse alvo, sendo que o mesmo filtro é usado para os eventos posteriores do alvo em questão. Dessa forma, múltiplos alvos podem ser rastreados isoladamente.

4.1.6 Sincronização de Filtro

A cada previsão o filtro muda seu estado. Garantir que a versão mais atual do filtro seja usada na previsão é fundamental para rastrear o alvo com sucesso e de forma distribuída. O PRATIQUE usa a última previsão para atualizar o filtro com baixo custo de comunicação, chamamos esse processo de **sincronização de filtro**.

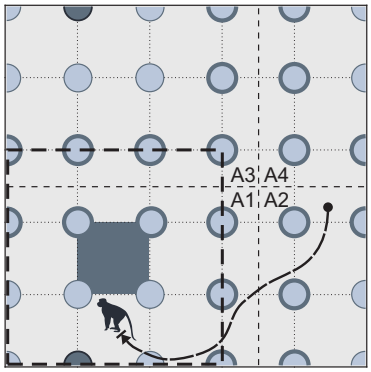
Quando um alvo é detectado pela primeira vez, uma instância de filtro é criada especificamente para fazer previsões sobre esse alvo. Esse filtro precisa ser passado para outros nós para manter a continuidade do rastreamento, pois as previsões são realizadas por um SCH e a cada novo evento um nó diferente pode ser eleito como tal. Caso o filtro não seja sincronizado, uma versão ultrapassada do filtro pode ser usada, prejudicando a precisão do rastreamento.

Uma solução simples de sincronização de filtro é enviar o filtro para todos os CHs da rede depois de cada previsão. Por outro lado, essa solução exige um custo de comunicação alto. Por isso, o PRATIQUE usa a previsão para atualizar apenas um pequeno conjunto de CHs. Esse conjunto é formado pelos CHs dos agrupamentos em que o alvo provavelmente estará quando gerar o próximo evento.

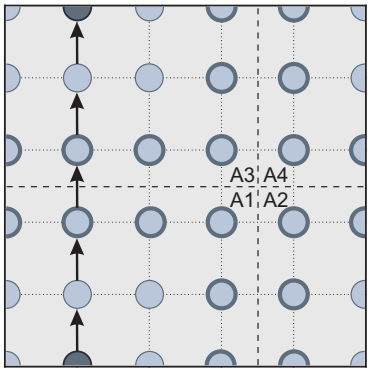
Quando a previsão é feita, o SCH verifica a que agrupamento pertence o nó que tem o mesmo índice da célula prevista. Se o agrupamento for diferente do dele, o SCH envia para o CH desse agrupamento o estado atual do filtro. Entretanto, como as previsões nem sempre são precisas, a sincronização pode não ocorrer corretamente. Por isso uma margem de erro $\mu \geq 0$ é considerada para incluir outras células próximas da célula prevista na verificação dos agrupamentos que devem ser atualizados. Quanto maior o valor de μ, mais células são usadas nessa verificação.

A figura 4.8 mostra um exemplo de sincronização quando o valor de $\mu = 1$. A célula prevista juntamente com as oito células ao redor dela são incluídas na verificação

da sincronização. Considerando que o SCH é o CH do agrupamento $A1$, o SCH verifica que existe um nó do agrupamento $A3$ cujo índice é o mesmo de uma das células usadas na verificação (figura 4.8(a)). Assim, ele decide que deve atualizar o filtro no CH do agrupamento $A1$ (figura 4.8(b)).



(a) Com a margem de erro, as células ao redor da célula prevista também são consideradas na verificação de sincronização de filtro.



(b) O SCH detecta que o CH do agrupamento $A3$ deve receber o filtro atualizado, então encaminha uma mensagem para isso.

Figura 4.8. Sincronização de filtro.

Nesse mesmo cenário, se $\mu = 0$, somente a célula prevista seria utilizada na verificação de atualização. Como a célula que possui o mesmo índice da célula prevista está no mesmo agrupamento do SCH, esse não enviaria nenhuma mensagem de atualização de filtro.

4.1.7 Notificação de Resultado

Em paralelo com a atualização de filtro, o SCH que estima e prevê a posição de um alvo notifica esses resultados para o nó sink.

4.2 Avaliação

Nesta seção descrevemos a metodologia de avaliação e os resultados obtidos.

4.2.1 Metodologia

As avaliações são realizadas por meio de simulações com ns-2. A configuração padrão da rede é composta de 2401 nós sensores monitorando uma região de $500 \times 500\,\mathrm{m}^2$, os nós ficam a 10 m de distância uns dos outros, resultando em 50×50 células de $10 \times 10\,\mathrm{m}^2$

cada. A região monitorada é dividida em 4×4 agrupamentos estáticos. O nó $n_{0,0}$ é o nó sink.

Os raios de alcance dos nós da rede são de 16 m, isso possibilita que todos os nós se comuniquem através de múltiplos saltos. Já o alcance do alvo é de 20 m. O modelo de mobilidade seguido pelo alvo é o CRW com grau de correlação de 0,80 e velocidade de 1 $^{m}/_{s}$. Agendamos 500 eventos em intervalos 10 s. A tabela 4.1 resume os parâmetros de simulação usados.

Tabela 4.1. Parâmetros de simulação usados nas avaliações do PRATIQUE.

Parâmetro	Valor
Campo de sensores	$500 \times 500\,\mathrm{m}^2$
Número de nós	2401
Agrupamentos	4×4
Alcance dos nós	16 m
Alcance do alvo	20 m

No cálculo de posição, usamos dois tipos de medidas para o peso: todos os nós têm peso 1 (simples); os nós mais próximos da origem do evento têm maior peso (ponderado), nesse caso a potência do sinal é usada como peso.

Nas previsões usamos os filtros de Kalman, Partículas ou Alfa-Beta combinados com os cálculos de posição simples e ponderado. O Filtro de Kalman tem seu sistema de equações lineares configurado como a equação 3.3. O Filtro de Partículas utiliza 1000 partículas e é representado pelo algoritmo 3.1. No Filtro Alfa-Beta $\alpha = \beta = 0,99$.

Comparamos o desempenho do agrupamento híbrido com as abordagens de agrupamentos estáticos e dinâmicos. Na abordagem estática, os agrupamentos são divididos na fase de configuração da rede. A abordagem dinâmica usada é similar a proposta por Yang et al. [2007], em que o CH é um dos nós que detecta o evento e é escolhido em duas etapas. A primeira etapa reduz a quantidade de candidatos a CH, para isso cada nó que detecta o evento divulga por *broadcast* uma mensagem de eleição contendo a sua distância para o alvo e seu id. Se um nó não recebe uma dessas mensagens com uma distância menor que a sua, ele se candidata a CH. Na segunda fase, cada candidato faz um *flooding* da mensagem de eleição para alcançar os outros candidatos. O candidato com a menor distância é eleito CH. Para ser reconhecido, o CH divulga sua condição para os outros nós que detectaram o evento.

Consideramos seis métricas: eventos notificados, atraso de notificação, mensagens enviadas, erro das medidas/previsões e energia residual. A taxa dos eventos notificados é o percentual de resultados que chegam ao nó *sink* em relação ao total de eventos ocorridos. O atraso de notificação é o tempo que leva até o nó *sink* ser notificado

sobre a ocorrência de um evento depois que esse ocorre. As mensagens enviadas é a quantidade de mensagens usadas em todo o processo de rastreamento.

O erro das medidas (ou erro das estimativas) é referente ao cálculo de posição (simples ou ponderado). Esse é calculado da mesma forma que o erro das previsão e é dado em células ou em metros. Sendo c_{x_1,y_1} a célula em que o alvo está e c_{x_2,y_2} a célula estimada pela aplicação, o erro em células é dado por

$$\epsilon\left(c_{x_1,y_1}, c_{x_2,y_2}\right) = max\left(|x_1 - x_2|, |y_1 - y_2|\right). \tag{4.1}$$

Já o erro em metros é dado como a distância Gaussiana entre a posição do alvo e o centro da célula estimada pela aplicação.

Modelamos o consumo de energia de acordo com Basagni et al. [2008]. A energia residual é a energia contida nos nós no final da simulação. A tabela 4.2 mostra os parâmetros de energia usados. Cada mensagem tem um tamanho 32 bytes.

Tabela 4.2. Parâmetros de energia usados no PRATIQUE.

Parâmetro	Valor
Energia inicial	100 joules
Transmissão	$1{,}48\times10^{-2}$ watts
Recepção	$1{,}25\times10^{-2}$ watts
Escutando	$1{,}60\times10^{-5}$ watts

Os resultados dos experimentos são obtidos da média de 35 execuções diferentes. As barras de erro representam um intervalo de confiança de 99%.

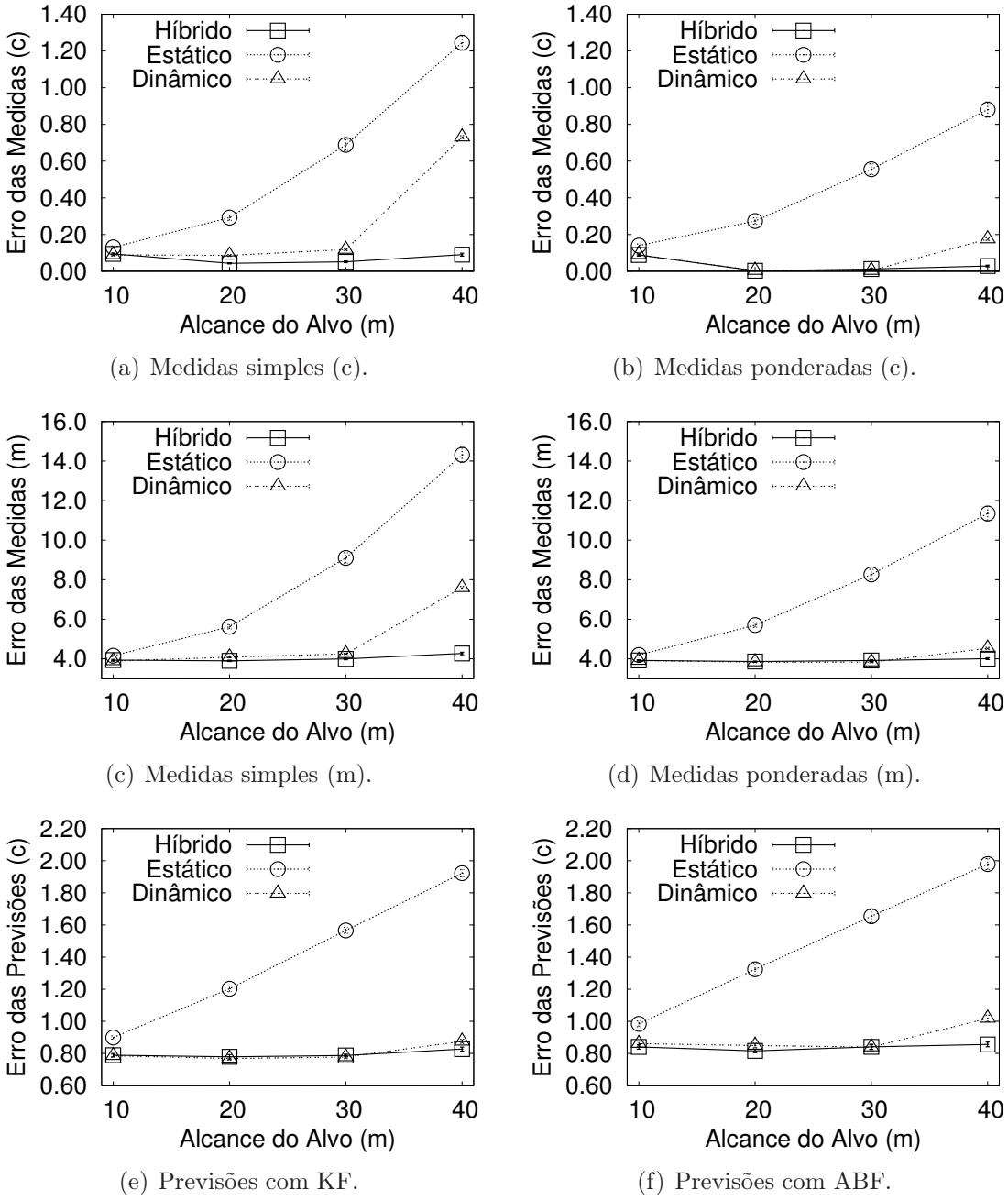
4.2.2 Resultados das Simulações

4.2.2.1 Alcance do Alvo

O alcance do alvo determina a quantidade de nós que o detectam. O ideal é que cada evento seja detectado por pelo menos três nós de uma célula. Desse modo, variamos o alcance do alvo de 10 m a 40 m (figura 4.9).

As figuras 4.9(a) e 4.9(b) mostram, respectivamente, o erro dos cálculos de posição simples e ponderado em células. O cálculo de posição ponderado reduz o erro do cálculo de posição em relação ao cálculo simples, pois os nós mais próximos do alvo têm maior influência na estimativa, então a estimativa pode ser precisa mesmo com a perda de alguns dos dados coletados.

Essas figuras ainda mostram que o agrupamento híbrido apresenta vantagem em relação aos agrupamentos estático e dinâmico. O agrupamento híbrido apresenta



(a) Medidas simples (c).

(b) Medidas ponderadas (c).

(c) Medidas simples (m).

(d) Medidas ponderadas (m).

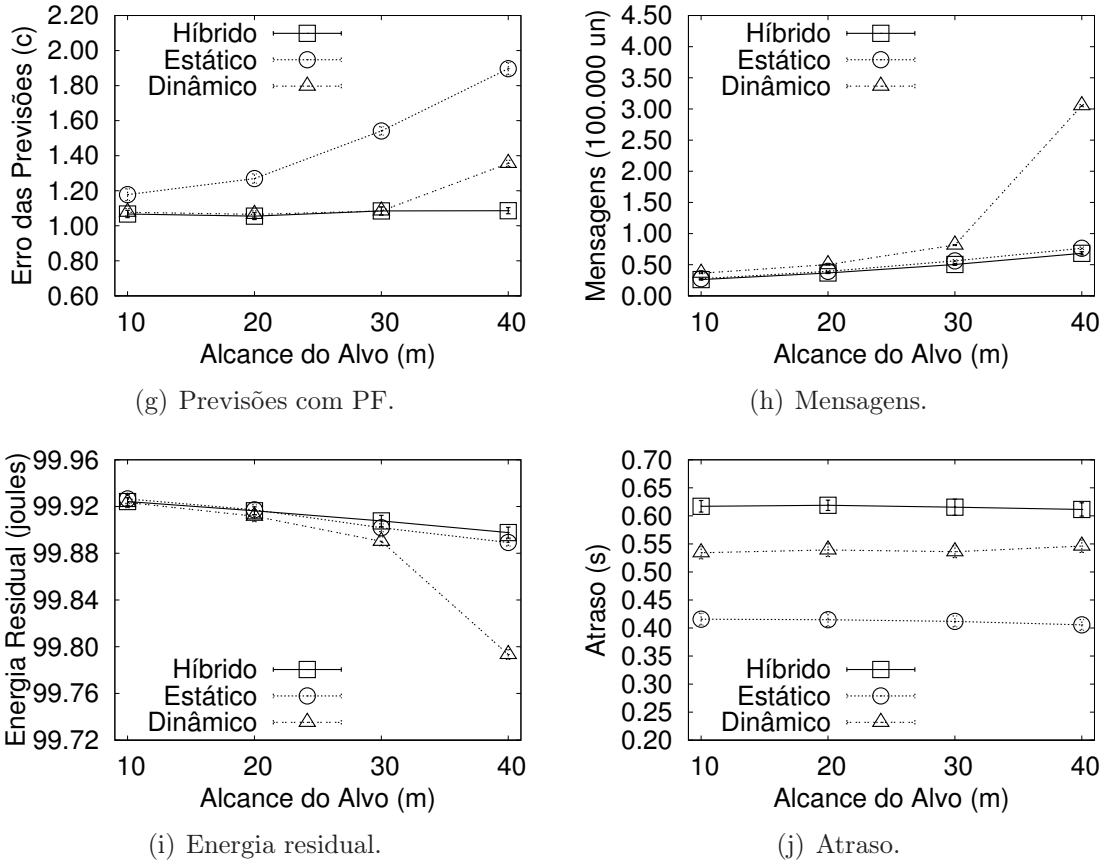
(e) Previsões com KF.

(f) Previsões com ABF.

Figura 4.9. Impacto do alcance do alvo (parte 1).

melhores resultados quando o raio de alcance do alvo é de 20 m ou 30 m, pois com 10 m poucos nós detectam o evento, já com 40 m o excesso de tráfego faz com que alguns dos dados coletados sejam perdidos, prejudicando a precisão da estimativa.

Com o agrupamento estático, o erro das estimativas cresce com o raio de alcance do alvo. Isso ocorre porque o evento alcança mais de um agrupamento estático a cada evento, dividindo os dados sobre o evento entre os agrupamentos, gerando resultados



(g) Previsões com PF.

(h) Mensagens.

(i) Energia residual.

(j) Atraso.

Figura 4.9. Impacto do alcance do alvo (parte 2).

redundantes e imprecisos. Quando o alcance do alvo é de 10 m, as estimativas com agrupamento estático são próximas as feitas com agrupamento híbrido, pois com esse alcance os dados quase sempre se concentram no mesmo agrupamento.

O agrupamento dinâmico também é afetado pelo crescimento do alcance do alvo, embora que em uma escala menor que o agrupamento estático. O crescimento do raio de alcance do alvo aumenta a quantidade de nós que participam da formação do agrupamento dinâmico. Dessa forma, o tráfego gerado nesse processo tem como consequência a perda de alguns dados coletados, prejudicando a precisão da estimativa.

As figuras 4.9(c) e 4.9(d) também mostram o erro das estimativas, mas em metros, apenas para dar uma melhor noção do erro. Nesse caso, o erro também vai depender do tamanho das células, quanto maior é a célula maior é o erro. Como as células usadas nas avaliações têm $10 \times 10\,\mathrm{m}^2$, mesmo que o algoritmo acerte a célula em que o alvo está, o erro da estimativa usando agrupamento híbrido fica em média 5 m, uma vez que a distância entre o alvo e o centro da célula estimada é usada como erro.

As figuras 4.9(e), 4.9(f) e 4.9(g) mostram, respectivamente os resultados das

previsões com KF, ABF e PF usando medidas ponderadas. A princípio, independente do filtro utilizado, as previsões refletem o comportamento das posições estimadas, i.e. quanto maior o erro das medidas mais as previsões são afetadas. Dentre os filtros utilizados nas previsões, o KF é o que retorna resultados mais precisos. O problema do rastreamento é modelado como um sistema linear, portanto o KF retorna a solução estatisticamente ótima. Apesar de ser uma versão simplificada do KF, o ABF retorna resultados similares aos dos KF, apesar do ABF ser inferior (0,06 células em média). O PF retorna resultados menos precisos que os filtros anteriores na maioria dos casos, entretanto apresenta resultados similares quando o erro das medidas é alto, chegando a superar o ABF quando o alcance do alvo é de 40 m com agrupamento estático[1].

Cada nó que detecta o evento gera uma mensagem, por isso o aumento do alcance do alvo incrementa o consumo de energia, como mostra a figura 4.9(h). Utilizar o agrupamento dinâmico consome mais energia da rede porque a formação desse agrupamento exige troca excessiva de mensagens a cada novo evento detectado. O agrupamento estático supera o híbrido na quantidade de mensagens quando o raio de alcance do alvo aumenta. Isso se deve aos resultados redundantes que são notificados para o nó *sink*, uma vez que usando o agrupamento estático, os dados podem ser divididos por mais de um agrupamento, o que gera resultados redundantes e imprecisos. A figura 4.9(i) mostra a energia residual dos nós da rede, a energia está relacionada com a quantidade de mensagens enviadas, portanto quanto mais mensagens são enviadas menos energia sobra na rede.

A figura 4.9(j) mostra o tempo que o evento leva para ser notificado depois de sua ocorrência. O alcance do alvo não interfere no atraso, pois o CH aguarda um tempo fixo pelos dados do agrupamento antes de notificar o resultado. O agrupamento estático é o mais rápido, pois os agrupamentos já foram previamente criados, portanto o atraso ocorre somente para coletar os dados e enviar o resultado para o nó *sink*. Os agrupamentos dinâmico e híbrido são formados a cada evento. O agrupamento híbrido é o que tem maior atraso, pois esse agrupamento é formado por nós (CHs de agrupamentos diferentes) que estão distantes entre si, isso exige um tempo maior para uni-los. Já o agrupamento dinâmico é formado por nós mais próximos, por isso se forma mais rápido que o híbrido.

[1]Nos próximos experimentos utilizaremos somente o cálculo de posição ponderado combinado com as previsões do KF, pois essa combinação é a que apresenta melhores resultados.

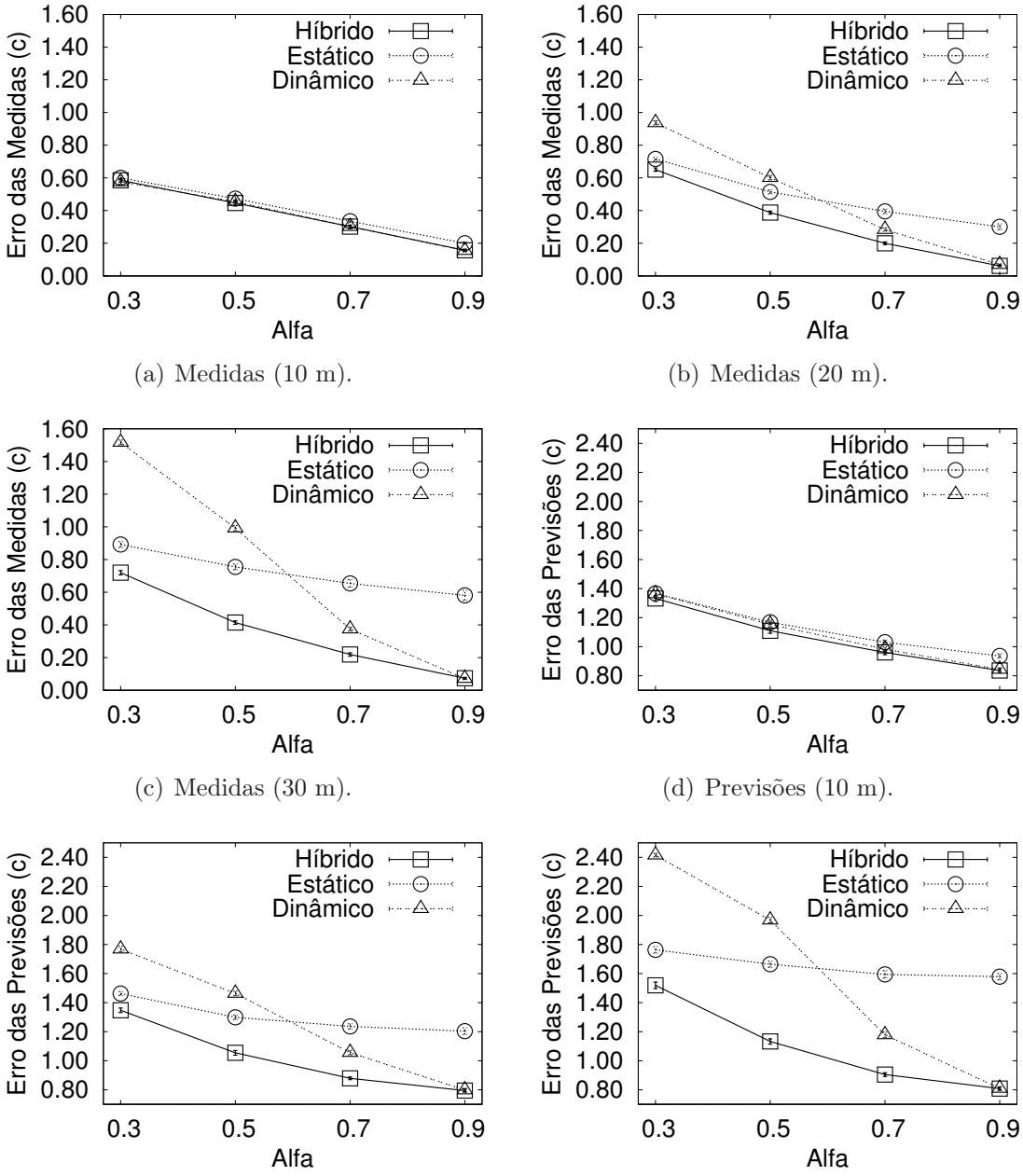
4.2.2.2 Detecção de Eventos

De acordo com a avaliação anterior, um alvo com alcance de 10 m (distância entre os nós grade) parece ser a melhor escolha, mas isso pode prejudicar o rastreamento, uma vez que os poucos nós que detectam o evento podem falhar, afetando a taxa de eventos notificados. Usamos o modelo flexível de detecção de eventos (seção 2.2.1.2) [Nakamura & Souza, 2010] para simular um cenário mais realístico. Configuramos $\gamma = 2$, $\theta = 1000$ e variamos o parâmetro de precisão α de 0,3 a 0,9. Avaliamos apenas o cálculo de posição ponderado combinando com KF. Os resultados são mostrados na figura 4.10.

As figuras 4.10(a), 4.10(b) e 4.10(c) mostram os erros das estimativas de posição do alvo quando esse possui 10 m, 20 m e 30 m, respectivamente. Quanto menor o valor de α maior é o erro das medidas, pois menos nós detectam o evento. Além disso, aumentar o raio de alcance do alvo também aumenta o erro das medidas, já que a distribuição dos nós que coletam dados sobre o evento se torna irregular devido aos nós que falharam em detectar o alvo, isso dificulta achar a célula mais central. O resultado das previsões reflete a qualidade das medidas usadas, como mostram as figuras 4.10(d), 4.10(e) e 4.10(f).

Quando o alcance do alvo é de 10 m, todos os tipos de agrupamentos geram resultados próximos, mas que se afastam quando o alcance aumenta para 20 m ou 30 m. Nesses casos, o erro do agrupamento estático aumenta pela divisão dos dados sobre o evento em diferentes agrupamentos. Já no agrupamento dinâmico, as falhas dos nós formam grupos de nós isolados, por isso mais de um agrupamento dinâmico é formado para um mesmo evento. Assim, de forma similar ao agrupamento estático, o agrupamento dinâmico acaba dividindo os dados sobre o evento, prejudicando a qualidade do estimativa de posição. O agrupamento híbrido, mesmo com as falhas dos nós, consegue unir os dados coletados em um único nó, por isso alcança melhores resultados.

A quantidade de mensagens usada pela rede é apresentada pelas figuras 4.11(a), 4.11(b) e 4.10(c). Quanto maior é a quantidade de nós que detectam o evento, mais mensagens são geradas para estimar e prever sua posição. Portanto, aumentar o alcance do alvo ou a precisão de detecção dos nós aumenta a quantidade de mensagens usada pela aplicação. Por outro lado, quando o alcance do alvo é de 20 m ou 30 m, o agrupamento dinâmico gera mais mensagens quando os sensores são mais imprecisos ($\alpha = 0,3$ ou $\alpha = 0,5$). Isso ocorre porque as falhas dos nós geram grupos isolados de nós que detectam o evento, então mais mensagens são usadas para criar mais agrupamentos e para notificar resultados redundantes.



(a) Medidas (10 m).

(b) Medidas (20 m).

(c) Medidas (30 m).

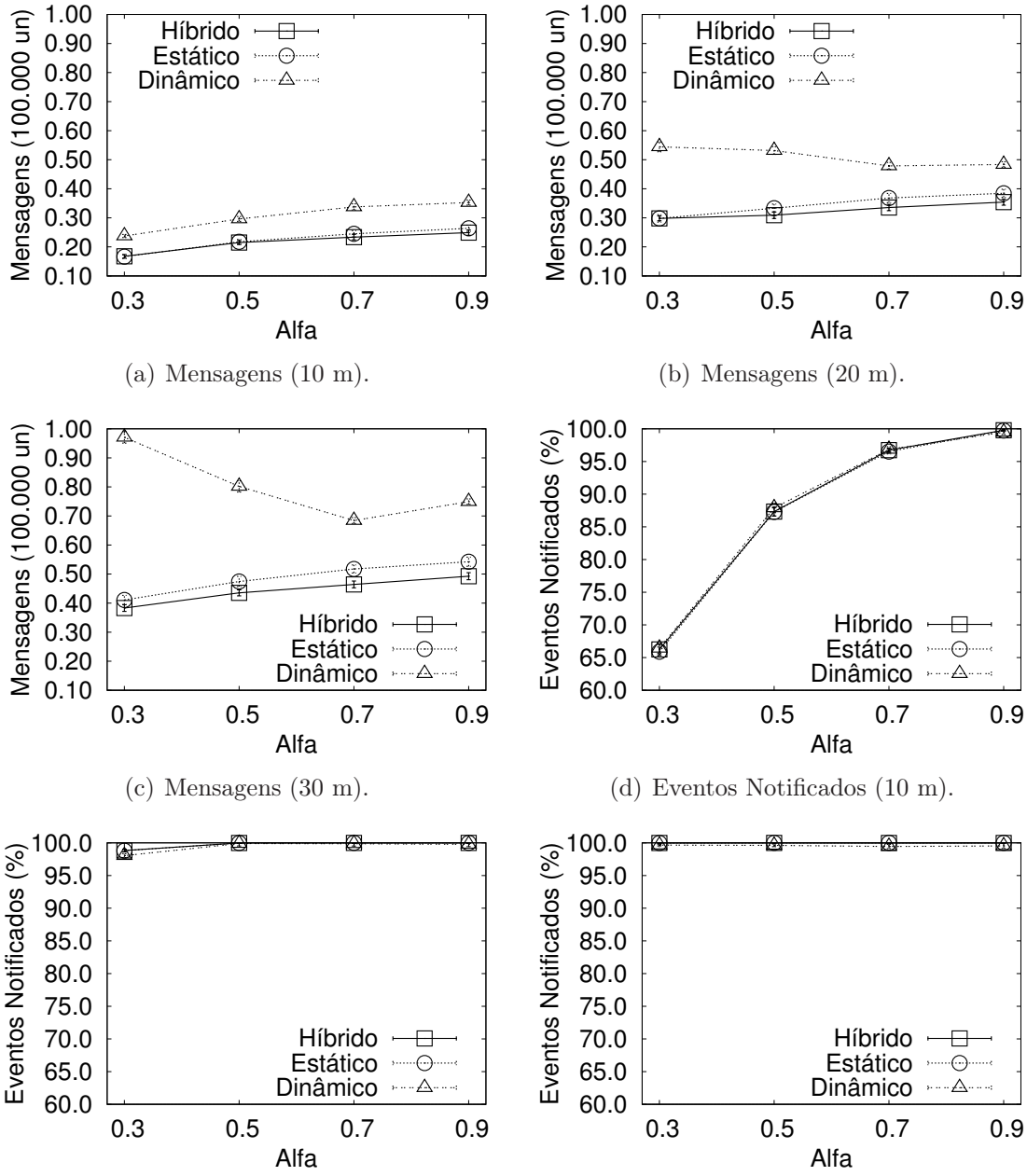
(d) Previsões (10 m).

(e) Previsões (20 m).

(f) Previsões (30 m).

Figura 4.10. Impacto da detecção de eventos (parte 1).

Apesar dos bons resultados obtidos com 10 m não é vantajoso utilizá-lo, o erro das medidas se torna menor porque muitos eventos não são notificados, já que os nós falham em detectá-los, como mostra a figura 4.10(d). Já quando o alvo tem alcance de 30 m, 100% dos eventos são notificados independente da precisão dos sensores, entretanto muitas mensagens são geradas e a distribuição irregular dos nós que detectam o evento afetam o cálculo de posição. Dessa forma, configurar o alcance do alvo para 20 m



(a) Mensagens (10 m).

(b) Mensagens (20 m).

(c) Mensagens (30 m).

(d) Eventos Notificados (10 m).

(e) Eventos Notificados (20 m).

(f) Eventos Notificados (30 m).

Figura 4.10. Impacto da detecção de eventos (parte 2).

(duas vezes a distância entre os nós da grade) garante que 100% dos eventos sejam notificados na maioria dos casos, conservando a qualidade das estimativas e usando menos mensagens.

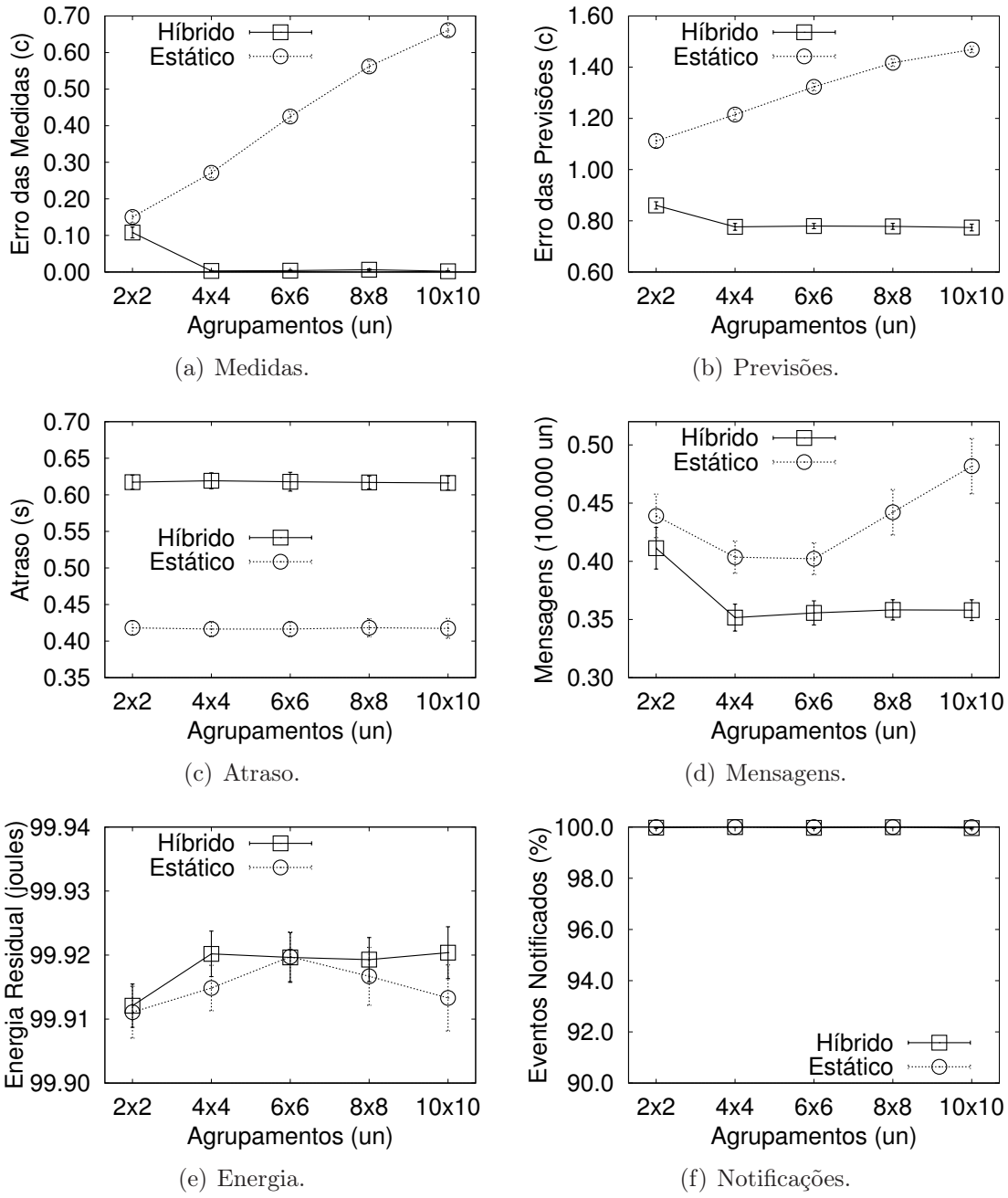
4.2.2.3 Agrupamentos

Os agrupamentos têm a função de reduzir a carga de comunicação da rede, pois cada agrupamento reúne seus dados coletados em um único nó (CH), que é responsável pela fusão desses dados e por notificar o resultado ao nó *sink*. Para avaliarmos o impacto da quantidade de agrupamentos no desempenho do rastreamento, dividimos o campo de sensores em 2×2 até 10×10 agrupamentos. Apenas os agrupamentos do tipo estático e híbrido podem ser avaliados nesse experimento. Os resultados são mostrados na figura 4.11.

A figura 4.11(a) mostra que o erro das medidas gerado pelo agrupamento estático aumenta com a quantidade de agrupamentos. Pois com isso, o evento alcança mais agrupamentos e os dados ficam dispersos em mais agrupamentos, gerando resultados redundantes e imprecisos. Já os resultados das medidas com agrupamento híbrido ficam estáveis independente da quantidade de agrupamentos, exceto quando o campo de sensores possui 2×2 agrupamentos. Essa quantidade faz com que os agrupamentos sejam formados por mais nós, logo a quantidade de saltos para alcançar o CH aumenta. O CH aguarda um tempo fixo para receber os dados do seu agrupamento, se o agrupamento é grande, os nós levam mais tempo para enviar seus dados ao CH. Dessa forma, o tempo que o CH aguarda pelos dados pode expirar antes que ele tenha coletado todos os dados, prejudicando a qualidade da estimativa. A qualidade das previsões reflete a qualidade das estimativas, como mostra a figura 4.11(b).

A figura 4.11(d) mostra que entre 4×4 e 6×6 é a quantidade de agrupamentos mais indicada para o cenário adotado. Usar 2×2 agrupamentos prejudica tanto o agrupamento estático quanto o híbrido, pois aumenta a quantidade de mensagens para encaminhar os dados coletados até o CH. No agrupamento estático, mais de 6×6 agrupamentos faz com que os dados coletados se dividam por mais agrupamentos, isso gera resultados redundantes que são notificados ao nó sink, exigindo mais mensagens. No caso do agrupamento híbrido, usar uma quantidade baixa de agrupamentos exige mais mensagens para encaminhar os dados até o CH, mas economiza em mensagens para formar o agrupamento dinâmico. Por outro lado, usar uma quantidade alta de agrupamentos exige mais mensagens para formar o agrupamento dinâmico, mas economiza no momento de encaminhar os dados ao CH. Essa compensação faz o custo de comunicação com agrupamento dinâmico permanecer em equilíbrio a partir de 4×4 agrupamentos. Economizar mensagens significa que os nós economizam mais energia, portanto a energia residual da rede reflete o custo de comunicação, como mostra a figura 4.11(e).

As figuras 4.11(c) e 4.11(f) mostram, respectivamente, que a quantidade de agru-



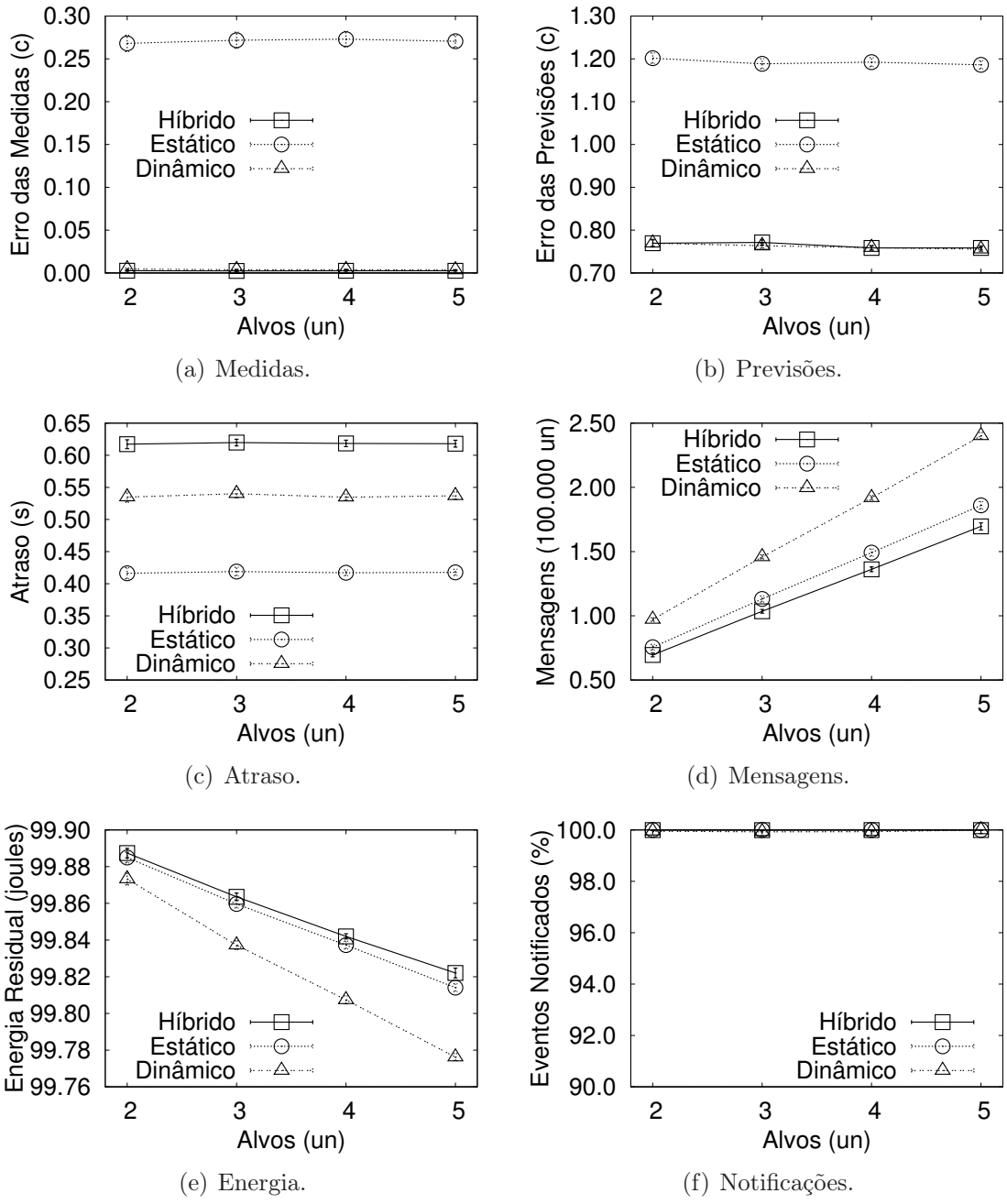
(a) Medidas. (b) Previsões.

(c) Atraso. (d) Mensagens.

(e) Energia. (f) Notificações.

Figura 4.11. Impacto da quantidade de agrupamentos.

pamentos não influenciam no atraso nem na taxa de notificação dos resultados. O agrupamento híbrido gera mais atraso que o estático, pois precisa de um tempo adicional para unir os dados de diferentes agrupamentos estáticos. Em ambos os tipos de agrupamento, aproximadamente 100% dos resultados são notificados.



(a) Medidas. (b) Previsões.

(c) Atraso. (d) Mensagens.

(e) Energia. (f) Notificações.

Figura 4.12. Impacto da quantidade de alvos.

4.2.2.4 Múltiplos Alvos

Aumentar a quantidade de alvos representa uma sobrecarga na rede, pois mais eventos ocorrem e mais mensagens são necessárias para tratá-los. Para avaliarmos isso, variamos a quantidade de alvos de 2 a 5 alvos. Os resultados são mostrados na figura 4.12.

Como as amostras de cada alvo são tratadas isoladamente, aumentar a quantidade de alvos não interfere no cálculo de posição e previsão, como mostram as figuras

4.12(a) e 4.12(b)). Os agrupamentos híbrido e dinâmico são mais precisos, pois centralizam todos os dados coletados em um único nó antes de calcular as estimativas e as previsões. O agrupamento estático, por outro lado, perde a precisão por deixar os dados dispersos em diferentes agrupamentos. Mais alvos geram mais eventos a serem tratados. Assim, as mensagens aumentam em aproximadamente 30% para cada alvo acrescentado, como mostra a figura 4.12(d). A energia residual diminui na mesma proporção (figura 4.12(e)). O atraso (figura 4.12(c)) e a taxa de eventos notificados (figura 4.12(f)) também não são afetados pelo aumento da quantidade de alvos.

4.3 Conclusões Parciais

Neste capítulo propomos e avaliamos o PRATIQUE. Esse algoritmo usa dois níveis de agrupamento (estático e dinâmico) para reduzir o custo de comunicação e garantir a qualidade da previsão, pois faz a fusão de todos os dados do evento em um único nó, mesmo que o evento alcance mais de um agrupamento. Além disso, utiliza previsão para atualizar o estado do filtro apenas nos nós que tem maior probabilidade de detectar o próximo evento.

Comparamos a técnica proposta de agrupamento híbrido com os agrupamentos puramente estáticos ou dinâmicos. O agrupamento híbrido exige um tempo maior para ser formado, entretanto garante resultados tão precisos quanto agrupamento dinâmico, mas utilizando menos mensagens. Além disso, o processo de formação do agrupamento híbrido gera menos colisões quando muitos nós estão envolvidos em um mesmo evento e é menos prejudicado pelos nós que falham em detectar o evento. O agrupamento estático gera resultados imprecisos, pois não garante que todos os dados serão reunidos em um único nó antes de estimar a posição do alvo.

O cálculo de posição é mais preciso se os nós mais próximos do evento possuem maior peso (cálculo de posição ponderado). Nesse caso, os dados gerados pelos nós mais próximos do alvo têm maior relevância no cálculo de posição. Ajustar o alcance do alvo para aproximadamente o dobro da distância entre os nós mantém a qualidade do rastreamento e da taxa de notificação.

Foram avaliados três filtros diferentes para prever a posição do alvo. O KF é mais eficiente que o ABF, mas o PF os supera quando o erro das medidas é alto (superior a 0,4 células).

O KF é o mais preciso, pois retorna a solução estatisticamente ótima. Apesar de ser uma versão simplificada do KF, o ABF gera resultados quase tão bons quanto seu antecessor. O PF é o que gera maior erro, entretanto se aproxima dos outros filtros

quando o erro das medidas é alto (superior a uma célula).

A quantidade de agrupamentos estáticos deve ser ajustado de acordo com as dimensões do campo de sensores para melhor reduzir o custo de comunicação. Pois poucos agrupamentos aumentam a quantidade de mensagens para entregar os dados ao CH. Por outro lado, muitos agrupamentos precisam de mais mensagens para configurar os agrupamentos dinâmicos e sincronizar os filtros.

Capítulo 5

TATI: Um Algoritmo de Rastreamento para Interceptação de Alvo em Redes de Sensores com Estrutura de Faces

Nas aplicações de rastreamento tradicionais, a rede precisa reportar as informações de rastreamento para o nó *sink*, que tem posição estática. Dessa forma, os algoritmos de rastreamento usam fusão de dados para reduzir a quantidade de dados transmitidos e, de acordo com a posição do alvo rastreado, usam rotas mais convenientes para reduzir os saltos de comunicação ou distribuir melhor a carga de comunicação entre os nós [Xu et al., 2004b; Liu et al., 2008; Hajiaghajani et al., 2012].

Por outro lado, o rastreamento pode ser formulado considerando um *sink* móvel, também chamado de usuário móvel ou objeto guiado. Nessa formulação, o objeto guiado deve se aproximar do alvo. Para isso, a rede detecta o alvo e permanece seguindo seu rastro para criar uma rota a ser seguida pelo objeto guiado.

Aplicações de rastreamento em redes de sensores precisam ser precisos em calcular a posição do alvo e ao mesmo tempo economizar energia por causa das limitações dos nós sensores. Os nós da rede precisam trabalhar de forma cooperada para rastrear o alvo. Um grupo de nós é selecionado para rastrear o alvo a cada amostragem, reduzindo a quantidade de nós em atividade para economizar energia.

O método trivial de rastrear o alvo em rede de sensores com objeto guiado é baseado em *floodings* sucessivos. O objeto guiado inicia um *flooding* na rede solicitando a posição do alvo. Ele recebe a posição do alvo e se dirige até ela. Chegando ao destino, ele faz um novo *flooding* para saber seu próximo destino. Esse processo tem um custo

muito alto, pois toda a rede é consultada a cada amostragem.

Métodos mais elaborados usam somente parte dos nós para rastrear o alvo, esse grupo de nós muda de acordo com a trajetória do alvo, evitando que toda a rede seja consultada. Esses métodos geralmente organizam a rede em faces [Karp & Kung, 2000; Huang et al., 2005] para facilitar a escolha do grupo de nós que deve ser ativado a cada amostragem. O *Dynamic Object Tracking* (DOT) [Tsai et al., 2007] determina quem é o nó mais próximo do alvo, que por sua vez ativa todas as suas faces adjacentes, enquanto os nós das demais faces da rede permanecem desativados para economizar energia. Assim, o DOT garante que na maioria dos casos haverá nós ativos para rastrear o alvo independente da direção que ele siga.

Alguns métodos usam previsão para reduzir a quantidade de faces ativas. O PET [Bhuiyan et al., 2010] e o POOT [Hsu et al., 2012] usam modelos de mobilidade linear para prever e ativar a face para onde o alvo está se dirigindo. Já Sharma et al. [2011b] utilizam ABF em suas previsões. Apesar de ajudar a rede a reduzir o consumo de energia, métodos baseados em previsão consideram que a mobilidade do alvo possui alguma correlação que a torne o resultado da previsão relevante.

Neste trabalho propomos e avaliamos o *Tracking Algorithm for Target Interception in face-structured sensor networks* (TATI). Trata-se de um algoritmo de rastreamento para redes de sensores com objeto guiado baseado no DOT. Esse algoritmo reduz a quantidade de faces ativas e o percurso do objeto guiado, dessa forma a rede economiza mais energia e o alvo é interceptado mais rápido. Para isso, a rede é organizada em faces, a posição atual do alvo e a velocidade máxima registrada durante o rastreamento são usados como parâmetros para definir as faces que devem ser ativadas, isso reduz o número de faces ativas e garante que há nós para detectar o alvo indepentente da direção que ele seguir. Além disso, o TATI calcula pontos de consulta com base no alcance de comunicação para reduzir o percurso do objeto guiado.

As soluções e resultados deste capítulo estão parcialmente publicados no Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos [Souza et al., 2014b] e em avaliação no *Local Computer Networks*. Uma versão estendida está em desenvolvimento para publicação no *Elsevier Ad Hoc Networks*. Além disso, um *survey* sobre algoritmos de rastreamento de alvos em redes estruturadas em faces está em avaliação no *IEEE Wireless Communications*.

No restante deste capítulo descrevemos com mais detalhes o funcionamento do TATI e avaliamos o seu desempenho em diferentes cenários.

5.1 Considerações e Definições

No rastreamento tratado neste trabalho, a rede de sensores deve fornecer ao objeto guiado a posição do alvo que está sendo rastreado, ajudando o mesmo a alcançar o alvo. O algoritmo aplicado nessa situação deve gerar uma rota que facilite a interceptação do alvo, além disso deve reduzir a quantidade de nós que efetivamente rastreiam o alvo para economizar energia.

O alvo (pessoa, animal ou veículo) se move aleatoriamente pelo campo de sensores, ele pode ser detectado pelos nós sensores que usam trilateração [Boukerche et al., 2007; Oliveira et al., 2009] para calcular sua posição e manter o rastreamento, por isso é necessário que pelo menos três nós detectem o alvo.

O objeto guiado (pessoa, robô ou veículo) não pode detectar o alvo, ele usa a rede para decidir a direção a ser seguida para alcançar o alvo. Esse pode se comunicar com os nós da rede e possui recursos de auto-orientação e mobilidade.

Os nós da rede estão sincronizados no tempo e conhecem sua localização e seu raio de comunicação nominal. Todos os nós possuem os mesmos alcances de comunicação e sensoriamento. Para economizar energia, os rádios dos nós sensores devem ser desligados, pois o rádio ligado, mesmo que ocioso, é responsável por maior parte do consumo de energia [Yan & Wang, 2011].

Dessa forma, os nós podem operar em cinco estados: *sleep*, *awake*, *active*, *near* e *beacon*. Esses estados definem o nível de consumo de energia do nó e a sua função no rastreamento. A tabela 5.1 descreve e representa graficamente os elementos envolvidos no algoritmo de rastreamento apresentado neste capítulo.

No estado *sleep* o nó economiza energia, pois permanece com o rádio desligado e não pode se comunicar com outros nós nem detectar o alvo, portanto não pode ajudar no rastreamento. Periodicamente, o nó *sleep* passa para *awake*, nesse estado ele pode receber uma mensagem para se tornar *active* e efetivamente participar do rastreamento. Essa mensagem pode ser originada por um nó *active* ou pelo objeto guiado. Se dentro de um período definido o nó *awake* não se tornar *active*, ele volta ao estado *sleep*. O nó em estado *active* é responsável por detectar o alvo, depois de executar sua tarefa, esse nó volta a alternar entre os estados *sleep* e *awake*. O nó *active* mais próximo do alvo se torna *near*, nesse estado ele é responsável por receber os dados coletados pelos outros nós e calcular a posição do alvo. O nó *near* se torna *beacon* caso o alvo mude de face. Os nós *beacons* são pontos de consulta para o objeto guiado, eles formam o caminho que o objeto guiado deve seguir para encontrar o alvo. O nó *beacon* volta para o estado *sleep* depois que o objeto guiado o consulta. A figura 5.1 mostra o diagrama de estados dos nós.

Tabela 5.1. Elementos envolvidos no rastreamento.

Gráfico	Elemento	Descrição
	Alvo	Objeto móvel que é rastreado pela rede.
	Objeto guiado	Elemento móvel que objetiva alcançar o alvo. Ele se comunica com a rede para obter informações sobre o alvo.
	Nó *sleep*	Nó em estado de economia de energia, não participa do rastreamento.
	Nó *active*	Nó que está efetivamente rastreando o alvo.
	Nó *beacon*	Nós que marcam a trajetória do alvo e indicam o caminho que o objeto guiado deve seguir.
	Nó *near*	É o nó *active* que está mais próximo do alvo.
	Nó detectando	É um nó que está detectando o alvo.

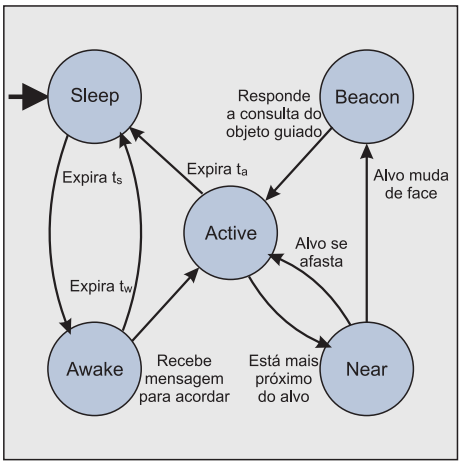
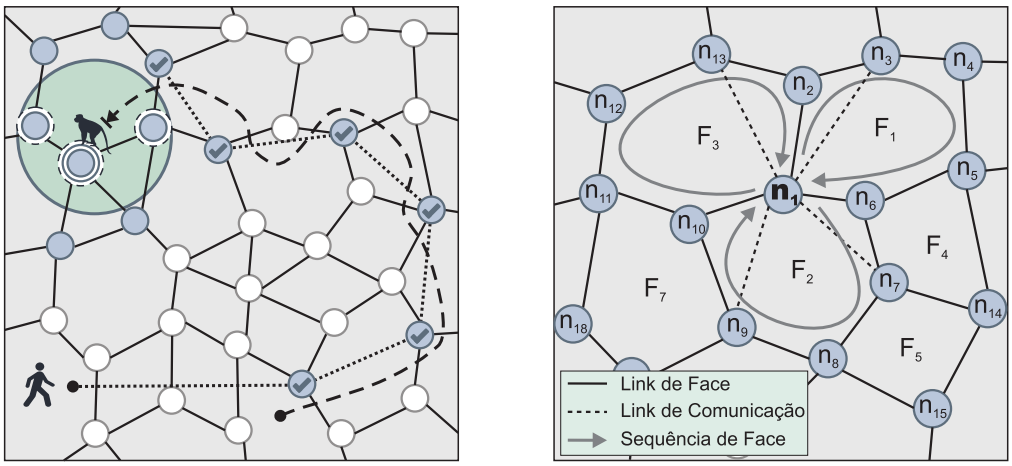


Figura 5.1. Diagrama de estados dos nós.

A redução da quantidade de nós em estado *active* e a redução da quantidade de mensagens enviadas são os fatores que mais contribuem para a economia de energia da rede. Entretanto, os nós a serem ativados devem ser escolhidos com cuidado para evitar a perda do alvo. O alvo é perdido quando vai para uma área que não possui sensores ativados. Nesse caso, o rastreamento é reiniciado e todos os nós são acionados para encontrar o alvo novamente.

A rede é organizada em uma estrutura de faces para melhor classificar os nós que devem ser ativados para o rastreamento. A figura 5.2(a) dá uma visão geral do

funcionamento do rastreamento, apenas os nós mais próximos do alvo ficam ativados, enquanto o restante mantém os rádios desligados para economizar energia. Além disso, cada face por onde o alvo passou possui um nó *beacon* que registra o caminho que deve ser seguido pelo objeto guiado.



(a) Visão geral de funcionamento do rastreamento.

(b) Visão geral da estrutura de faces.

Figura 5.2. Visão geral do algoritmo de rastreamento e da estrutura de faces.

As faces são criadas na fase de configuração da rede usando as técnicas de planarização GG e RNG [Maignan & Gruau, 2011; Tsai et al., 2007]. A planarização provê aos nós informações mais detalhadas sobre os seus vizinhos, dessa forma os nós podem cooperar de forma mais eficiente. A rede plana geralmente é utilizada para resolver o problema de buracos durante o roteamento [Huang et al., 2005]. Para o rastreamento de alvos, as faces são usadas para escolher o grupo de nós que efetivamente rastreia o alvo [Tsai et al., 2007; Hsu et al., 2012].

A figura 5.2(b) mostra um exemplo de rede planarizada e será usada para ilustrar as definições relacionadas com a estrutura de faces. Do ponto de vista físico, em uma rede planar, uma face é a área envolvida por um conjunto de nós e suas ligações. A face F_1, por exemplo, é a área envolvida pelos nós n_1, n_2, n_3, n_4, n_5 e n_6. O nó n_1 possui três faces adjacentes, pois está associado às faces F_1, F_2 e F_3.

Na estrutura de faces existe duas classes de vizinhos: vizinhos de rede e vizinhos de face. Os vizinhos de rede de um nó são todos os nós dentro do seu raio de comunicação. O nó n_1 pode se comunicar diretamente com os nós n_2, n_3, n_6, n_7, n_9, n_{10} e n_{13}, logo esses são os seus vizinhos de rede. Os vizinhos de face de um nó são os nós que formam as suas faces adjacentes. O nó n_1 possui doze vizinhos de face, sendo que n_6, n_7, n_8, n_9 e n_{10} são seus vizinhos de face da face F_2. Sempre que um nó precisar divulgar alguma

informação em uma face, a mensagem é enviada primeiramente para o vizinho de face imediato, sendo repassada para os outros vizinhos de face até completar o ciclo. Por exemplo, se o nó n_1 precisa divulgar uma mensagem na face F_3, a mensagem segue a sequência da face em questão, passando pelos nós n_{10}, n_{11}, n_{12}, n_{13} e n_2.

5.2 Descrição do TATI

O TATI é aplicado para ajudar o objeto guiado a perseguir o alvo. Para isso, a rede é estruturada em faces para distinguir diferentes áreas do campo de sensores e ajudar a escolher de forma mais adequada o grupo de nós que deve ser ativado para rastrear o alvo. Esse algoritmo reduz o consumo de energia reduzindo as faces ativas e reduz o caminho percorrido pelo objeto guiado usando a informação do raio de comunicação nominal.

O rastreamento é iniciado quando o objeto guiado faz um *flooding* na rede para descobrir o alvo. Os nós que detectam o alvo, calculam a posição do mesmo e o nó mais próximo dessa posição é marcado como *beacon* da face em que o alvo se encontra. O objeto guiado se aproxima desse *beacon* para consultar o seu próximo destino. Para manter o rastreamento, um nó *beacon* é criado para cada face por onde o alvo passou. Dessa forma, o rastro do alvo é uma sequência de nós *beacons* que são consultados um após o outro pelo objeto guiado.

Para economizar energia durante esse processo, o TATI registra a maior distância entre dois cálculos de posição do alvo, i.e. o deslocamento máximo do alvo entre duas amostragens, esse deslocamento é divulgado para os nós *active*. Com isso, os nós podem determinar as faces em que o alvo pode estar na próxima amostragem, somente os nós dessas faces são ativados. Essa estratégia reduz a quantidade de faces ativas e garante que o alvo não será perdido na maioria dos casos.

O TATI reduz a distância percorrida pelo objeto guiado. O objeto guiado não precisa chegar até a posição exata do *beacon* para consultar seu próximo destino, basta chegar até a área dentro do radio de comunicação. Assim, o objeto guiado segue o caminho mais curto até o ponto de consulta.

No restante desta seção descrevemos o funcionamento o TATI com maiores detalhes.

5.2.1 Estrutura da Rede

O TATI funciona em uma rede de sensores com estrutura de faces, então essa estrutura é criada na fase de configuração da rede. Para criar as faces, a rede precisa ser organizada

como um grafo planar.

Um grafo é dito ser planar quando existe alguma forma de se dispor seus vértices em um plano de tal modo que nenhum par de arestas se cruze, dividindo o plano em regiões denominadas faces.

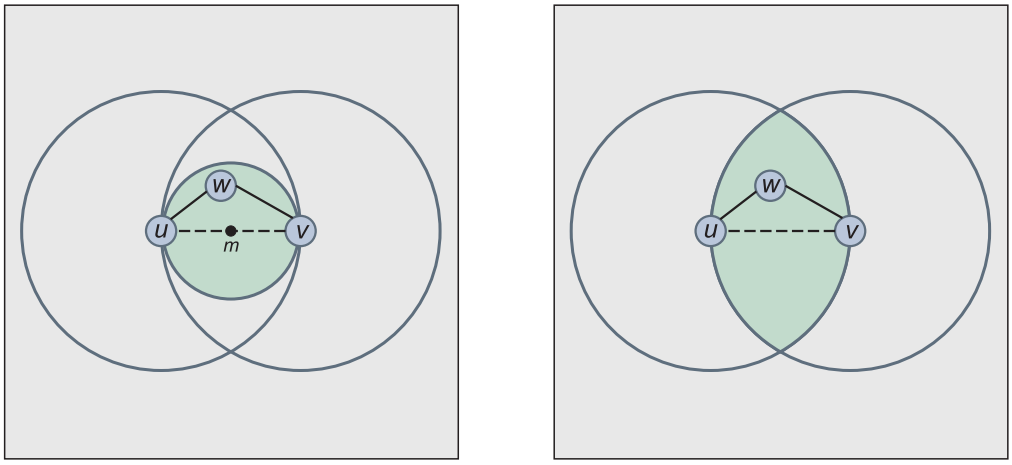
A construção distribuída da estrutura de faces é dividida em duas fases: planarização de rede e formação de faces. Após esse processo, cada nó conhece os seus vizinhos de face, com isso ele conhece a área da face e a sequência de saltos entre nós para percorrer uma face específica.

5.2.1.1 Planarização da Rede

Para dar início a planarização da rede, cada nó envia uma mensagem por *broadcast* para informar o seu ID e localização, depois disso cada nó possui uma lista de seus vizinhos imediatos. Somente com essa informação de localização, os nós eliminam as arestas que se cruzam de forma que a rede represente um grafo planar. Os algoritmos mais utilizados para formar grafos planares são GG e RNG.

A rede de sensores pode ser vista como um grafo $G = (V, E)$, onde V é o conjunto de nós e E é o conjunto de arestas que ligam um par de nós que pode se comunicar diretamente. Em um grafo GG, dado que m é o ponto central entre vértices u e v, existe uma aresta (u, v) entre esses dois vértices, se o vértice w for mais distante de m do que o vértice u. A figura 5.3(a) mostra graficamente como uma aresta é removida durante a construção do grafo GG, a região sombreada é um círculo de centro m e de diâmetro $d(u, v)$, a aresta (u, v) é removida porque w está dentro desse círculo. O algoritmo 5.1 mostra como cada nó u, partindo da sua lista completa de vizinhos N, remove as arestas que não pertencem ao grafo GG [Karp & Kung, 2000].

Em um grafo RNG, uma aresta (u, v) existe entre os vértices u e v, se a distância entre eles, $d(u, v)$, é menor ou igual as distância desses vértices para o vértice w. A figura 5.3(b) mostra graficamente como uma aresta é removida durante a construção do grafo RNG, a região sombreada é a interseção formada pelos círculos de centro u e v com raio $d(u, v)$, a aresta (u, v) é removida porque w está na região de interseção. O algoritmo 5.2 mostra como cada nó u, partindo da sua lista completa de vizinhos N, remove as arestas que não pertencem ao grafo RNG [Karp & Kung, 2000].



(a) GG – A aresta (u, v) é removida porque w está dentro do círculo de diâmetro $d(u, v)$.

(b) RNG – A aresta (u, v) é removida porque w está na região de interseção.

Figura 5.3. Representação gráfica dos algoritmos de planarização.

```
1  foreach v in N do
2      foreach w in N do
3          if v = w then
4              continue;
5          else if
           d(m, w) > d(u, m)
           then
6              m ⇐ midpoint(uv);
7              remove edge (u, v);
8              break;
9          end
10     end
11 end
```

Algoritmo 5.1: Removendo as arestas que não pertencem ao grafo GG.

```
1  foreach v in N do
2      foreach w in N do
3          if v = w then
4              continue;
5          else if d(u, v) >
           max(d(u, w), d(v, w))
           then
6              remove edge (u, v);
7              break;
8          end
9      end
10 end
```

Algoritmo 5.2: Removendo as arestas que não pertencem ao grafo RNG.

5.2.1.2 Formação de Faces

Após a planarização, os nós ainda não conhecem as suas faces, cada nó conhece apenas os seus vizinhos de face imediatos, resultantes das arestas que não foram removidas durante a planarização. Para cada vizinho de face imediato existe uma face a ser

descoberta.

Para descobrir uma face, um dos nós da face deve gerar uma mensagem que percorre toda a face, passando por todos os nós que a compõe até completar o ciclo. Enquanto a mensagem atravessa a face, as coordenadas dos nós por onde a mensagem passou são adicionadas a mensagem. Depois que a mensagem atravessa a face, o nó que originou a mensagem passa a conhecer as coordenadas de todos os nós da face, com isso ele forma a face. Posteriormente, esse mesmo nó gera uma outra mensagem que percorre a face novamente para informar aos nós restantes a face descoberta [Huang et al., 2005]. A figura 5.4 representa uma rede durante a formação das faces. A face F_1 ainda está em formação, enquanto as outras faces já foram formadas.

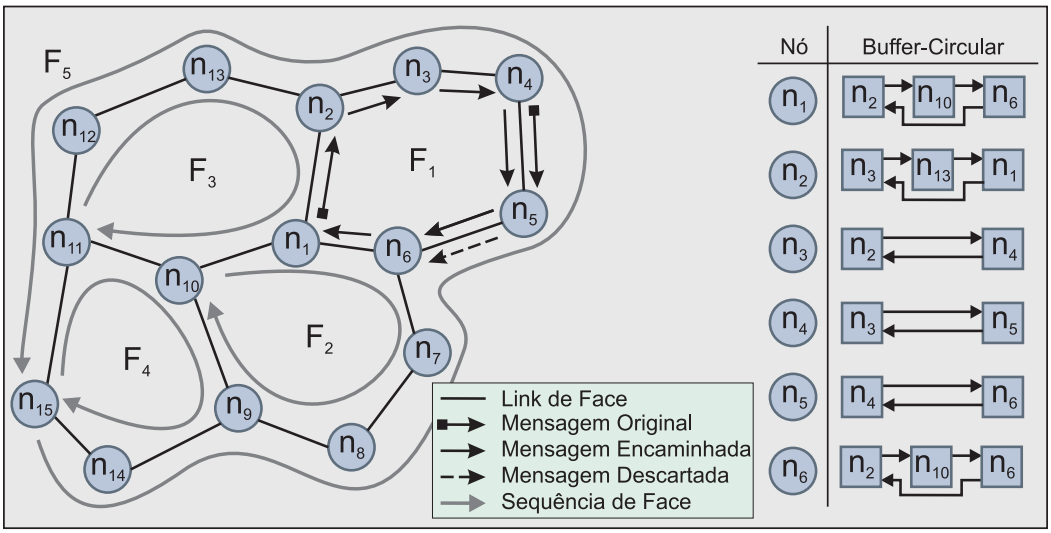


Figura 5.4. Formação de faces usando *buffer*-circular ordenado.

Para fazer a mensagem percorrer a face corretamente, cada nó cria um *buffer*-circular que armazena os vizinhos de faces imediatos ordenados no sentido anti-horário. Um nó faz essa ordenação com base nos ângulos entre ele e seus vizinhos. Dessa forma, quando uma mensagem chega de um nó, ela será encaminhada para o próximo nó do *buffer*-circular. Como a mensagem possui a lista ordenada dos nós por onde ela passou, essa informação é usada para definir de onde a mensagem veio e para onde ela deve ir para percorrer uma face no sentido horário. A figura 5.4 mostra o *buffer*-circular de cada nó da face F_1, o nó n_1 origina uma mensagem que é enviada para seu vizinho de face imediato n_2, o nó n_2 conhece a origem da mensagem consultando a própria mensagem e descobre para onde ela deve ser encaminha consultando o *buffer*-circular. Nesse caso, como a origem é n_1, o destino é n_3, pois n_3 vem depois de n_1 na sequência do *buffer*-circular. Seguindo esse critério, as mensagens continuam a ser encaminhadas até

chegar novamente em n_1, que é o nó que descobre a face e depois informa a descoberta da mesma para os outros nós que a compõe.

Um problema durante a formação das faces é fazer com que apenas uma mensagem percorra cada face para reduzir o custo de comunicação. Uma estratégia é criar uma regra de prioridade com base na posição: o nó mais a esquerda da face tem prioridade, se ocorrer empate, o nó mais acima tem prioridade, persistindo o empate, o nó com menor ID tem prioridade. Dessa forma, os nós iniciam aleatoriamente o envio de mensagens para descobrir as faces, o nó que receber uma mensagem a passa adiante somente se o nó que a originou tem maior prioridade. Além disso, se esse nó ainda não originou a sua própria mensagem, ela não será mais enviada, pois nesse caso já existe outro nó com maior prioridade na face. No caso da figura 5.4, consideramos que n_1 e n_4 originam simultaneamente mensagens para descobrir a mesma face. O nó n_4 envia a mensagem para n_5, que tem menor prioridade e portanto a encaminha para n_6. Por outro lado, n_6 descarta a mensagem, pois tem maior prioridade que n_4. O nó n_1 possui a maior prioridade da face, portanto todos os nós encaminham a mensagem originada por ele e não mais originam suas próprias mensagem.

No processo de formação das faces, uma face composta pelos nós de borda da rede, chamada de face externa, é criada juntamente com as faces internas. A princípio não há como distinguir entre esses tipos de face durante a sua formação. Uma solução simples é atribuir um limite de saltos para a quantidade de nós que formam a face, se esse limite for ultrapassado, a face é ignorada. Entretanto, em algumas situação, essa solução pode errar, fazendo com que algumas faces internas não sejam formadas. Na figura 5.4, a face F_5 é a face externa, ela é formada quando n_{15} origina uma mensagem para n_{14}, pois seguindo a sequência do *buffer*-circular, a mensagem volta para n_{15} no sentido anti-horário.

5.2.2 Encontrando o Alvo

A posição do alvo é desconhecida antes do rastreamento começar. Sendo assim, no período em que os nós estão *awake*, o objeto guiado inicia um *flooding* na rede para solicitar que o alvo seja encontrado. Os nós são sincronizados, portanto têm tempos definidos para alternar entre os estados *sleep* e *awake*.

O objeto guiado faz o *flooding* com a mensagem *TargetRequest*, nesse processo é criada uma rota temporária entre os nós e o objeto guiado. O nó sensor que recebe essa mensagem a passa adiante e verifica se o alvo está dentro do seu raio de sensoriamento. Os nós que detectam o alvo trocam entre si a mensagem *TargetDetected*, contendo a distância estimada entre o nó e o alvo. Como os nós conhecem a localização de todos

os seus vizinhos de face, os nós podem calcular a posição do alvo com multilateração e determinar qual deles está mais próximo do alvo.

O nó mais próximo vai para o estado *near*, ativando todos os nós das suas faces adjacentes. Depois ele se torna *beacon* da face em que o alvo está e envia para o objeto guiado a localização do alvo pela mensagem *TargetRequestReply*. Se ocorrer algum imprevisto e o objeto guiado não receber a resposta em um tempo predefinido, um novo *flooding* é feito no próximo período de *awake*.

Depois que o alvo é encontrado, o rastreamento é mantido pelos nós *active*.

5.2.3 Rastreando o Alvo

Uma maneira simples de rastrear o alvo seria fazer com que o objeto guiado use o *flooding* continuamente para obter a posição do alvo enquanto se aproxima do mesmo. Entretanto, esse processo possui um custo muito alto. Por isso, o *flooding* deve ser usado apenas para encontrar o alvo (como descrevemos anteriormente).

Uma vez que o alvo foi encontrado, os nós próximos do alvo devem manter o rastreamento, assim apenas um pequeno grupo de nós efetivamente participa do rastreamento a cada amostragem. Esse grupo de nós é ajustado dinamicamente de acordo com a estrutura de faces e com a trajetória do alvo.

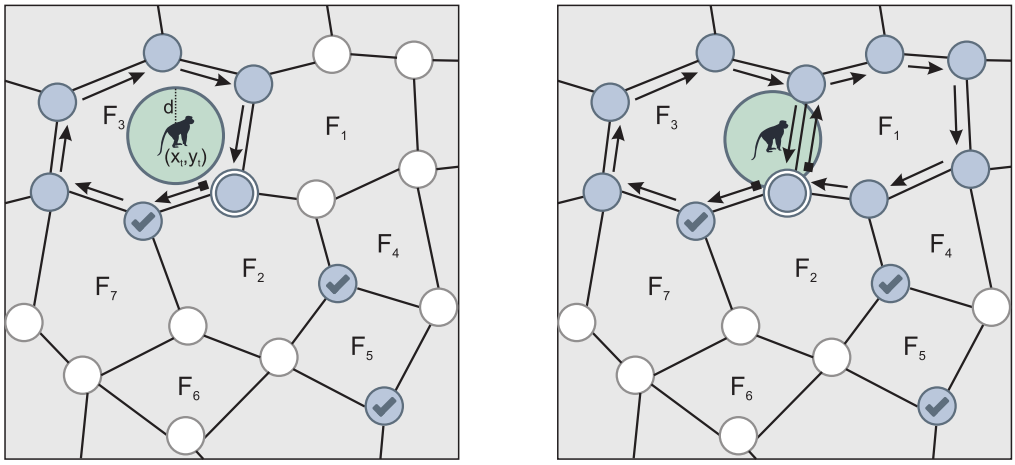
Quando o alvo é encontrado, a face em que ele está tem um nó *beacon*, esse mesmo nó também é *near* e todos os nós das suas faces adjacentes estão no estado *active*. Todas as faces adjacentes são ativadas nesse primeiro momento porque ainda não há informações suficientes sobre o alvo para escolher de forma mais inteligente as faces que devem ser ativadas. Isso garante que o alvo não seja perdido independente da direção que ele seguir.

Periodicamente o nó *near* envia uma mensagem *Wakeup* para os nós das faces que ele pretende ativar. Essa mensagem segue a sequência da face até alcançar todos os nós da face em questão. A mensagem possui todas as informações necessárias para o rastreamento, como o ID do último *beacon*, o ID do nó *near* atual, a posição do alvo, a face em que o alvo está, a distância máxima percorrida pelo alvo entre duas amostragens e a quantidade de faces por onde o alvo já passou.

Os nós que recebem a mensagem de *Wakeup* mudam seus estados para *active*, armazenam todas as informações de rastreamento contidas na mensagem e iniciam a detecção colaborativa do alvo. Os nós que detectam o alvo trocam entre si a mensagem *TargetDetected* para calcular a posição do alvo e determinar qual deles é o nó *near*. Dessa forma, o nó *near* pode mudar enquanto o alvo se move pela face, essa mudança é um dos fatores que determina as faces que são ativadas.

Somente o nó *near* envia mensagens de *Wakeup*. Ele escolhe quais de suas faces adjacentes devem ser ativadas com base na distância máxima d percorrida pelo alvo entre duas amostragens. Para isso, verifica se há interseção entre cada uma de suas faces adjacentes e o círculo de centro (x_t, y_t) e raio d, onde (x_t, y_t) é a posição atual do alvo. Se houver interseção, a face é ativada, então o nó *near* envia uma mensagem de *Wakeup* que passa por todos os nós da face.

Independente da direção que o alvo escolhe seguir é muito provável que na próxima amostragem ele ainda esteja na área do círculo, pois o histórico da sua trajetória mostra o quanto ele costuma se deslocar entre as amostragens. Isso possibilita que a quantidade de faces ativas seja reduzida de forma segura. A energia da rede é economizada, pois uma quantidade menor de nós permanece com o rádio ligado e menos mensagens de *Wakeup* são usadas. Na figura 5.5, o círculo representa a área em que o alvo pode estar na próxima amostragem, sendo assim as faces que são ativadas possuem alguma interseção da sua área com esse círculo.



(a) Apenas a face F_3 é ativada. (b) As faces F_1 e F_3 são ativadas.

Figura 5.5. Ativação das faces durante o rastreamento. O alvo pode estar em qualquer parte do círculo na próxima amostragem, por isso as faces que fazem interseção com o círculo são ativadas.

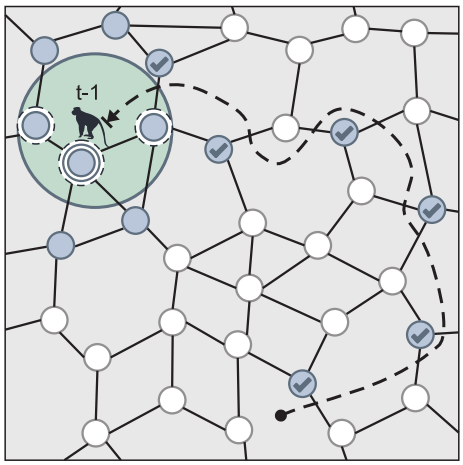
Um nó *near* se torna *beacon* quando ele percebe que o alvo mudou de face. Cada face por onde o alvo passou possui um nó *beacon* que registra a trajetória do alvo. No caso da figura 5.5, quando o alvo sair da face F_3 para entrar em F_1, o nó *near* se tornará *beacon* da face F_1.

5.2.4 Perda e Recuperação do Alvo

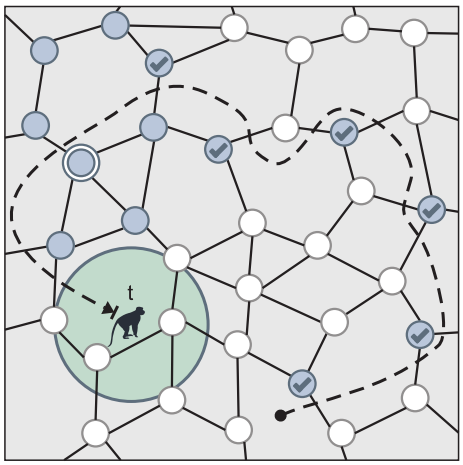
O alvo é perdido quando entra em uma área que não possui nós em estado *active* para detectá-lo. Isso ocorre quando o grupo de nós ativos não consegue acompanhar o alvo, seja porque a velocidade do alvo é alta, porque a área das faces é pequena ou por falha de comunicação.

Os nós periodicamente alternam entre os estados *sleep* e *awake* para checar se precisam ser ativados para detectar o alvo (frequência de *awaking*). Se essa frequência é alta, a perda do alvo é evitada, mas aumenta o consumo de energia da rede. A frequência pode ser ajustada para evitar que o alvo seja perdido e para economizar energia.

A figura 5.6 mostra uma situação em que o rastreamento corre bem até o tempo $t-1$. Entretanto, por uma aumento brusco de velocidade, o alvo se distancia do grupo de nós ativos, impossibilitando que ele possa ser detectado no tempo t. Dessa forma, o alvo é perdido.



(a) A rede rastreia o alvo até o tempo $t-1$.



(b) O alvo se desloca de forma que no tempo t sai do alcance de todos os nós ativos.

Figura 5.6. Alvo sendo perdido pela rede no intervalo entre duas amostragens.

Quando a rede detecta que o alvo foi perdido, ela pode executar um algoritmo de recuperação. Esse usa as últimas informações sobre o alvo para encontrá-lo novamente sem precisar consultar toda a rede [Khare & Sivalingam, 2011; Hsu et al., 2012].

Algoritmos de recuperação podem ser adicionados ao TATI, mas como mecanismos de recuperação não são o foco no momento, quando o alvo é perdido, o rastreamento é reiniciado usando *flooding*, como mostramos anteriormente.

5.2.5 Rota do Objeto Guiado

Os nós *beacons* registram a trajetória do alvo, cada face por onde o alvo passou possui um *beacon*. Esses nós podem ser vistos como uma lista duplamente encadeada, pois cada nó conhece os *beacons* imediatamente anterior e posterior a ele. Um *beacon* é o primeiro *beacon* se não tem referência para um *beacon* anterior, e é o último *beacon* se não tiver referência para um *beacon* posterior.

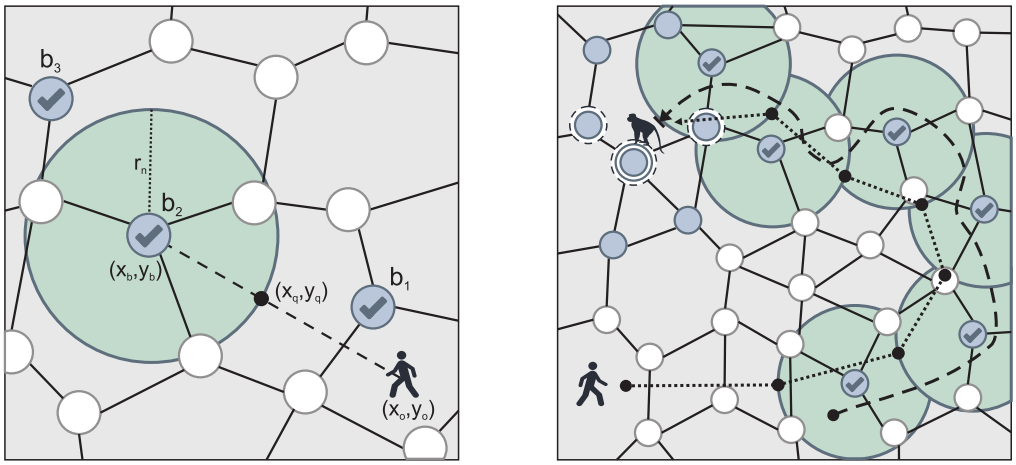
Para seguir o alvo após ele ser encontrado, o objeto guiado se aproxima do primeiro *beacon* para consultar o caminho que deve seguir. A consulta é feita enviando uma mensagem *QueryNext* para o *beacon*, que por sua vez responde com uma mensagem *QueryNextReply*. Se esse *beacon* não é o último, ele envia na resposta o próximo ponto de consulta, caso contrário ele envia a posição do alvo. Então o objeto guiado segue em direção a posição informada, após chegar ao seu destino ele inicia outra consulta.

Depois do nó *beacon* informar o próximo ponto de consulta, ele volta para o estado *awake* ou *sleep*. Mas antes disso, ele envia para o próximo *beacon* a mensagem *InfoFirst*. Essa mensagem serve para informar o próximo *beacon* que agora ele é o primeiro *beacon*.

Usualmente a posição do *beacon* é usada como ponto de consulta. Entretanto, o TATI usa o raio de comunicação nominal r_n, a posição do objeto guiado (x_o, y_o) e a posição do próximo *beacon* (x_b, y_b) para calcular um ponto de consulta (x_q, y_q) que reduz o percurso do objeto guiado. Dessa forma, (x_q, y_q) é a interseção entre reta que liga os pontos (x_o, y_o) e (x_b, y_b) e o círculo de centro (x_b, y_b) e raio r_n (figura 5.7(a)). Se o objeto guiado já está localizado dentro do raio de comunicação do *beacon* a ser consultado, a sua própria posição é usada como ponto de consulta. O valor de r_n é menor que o raio de comunicação real, essa diferença é uma margem de erro para evitar possíveis falhas de comunicação.

A figura 5.7(a) mostra parte de uma rede por onde o alvo passou. O nó b_1 é o *beacon* que informa ao objeto guiado a posição para consultar b_2. Em outras palavras, o ponto de consulta é a menor distância entre o objeto guiado e a área de comunicação do próximo *beacon* a ser consultado. A figura 5.7(b) mostra a rota do objeto guiado seguindo os pontos de consulta para se aproximar do alvo. O percurso gerado é mais curto do que se o objeto guiado tivesse que chegar até o *beacon* para fazer a consulta.

O TATI cria atalhos e remove círculos no caminho formado pelos *beacons* da mesma forma que o DOT, porém descreveremos esses processos a seguir para explicar o funcionamento completo do rastreamento.



(a) Calculando o ponto de consulta.

(b) Objeto guiado se aproxima do alvo usando os pontos de consulta.

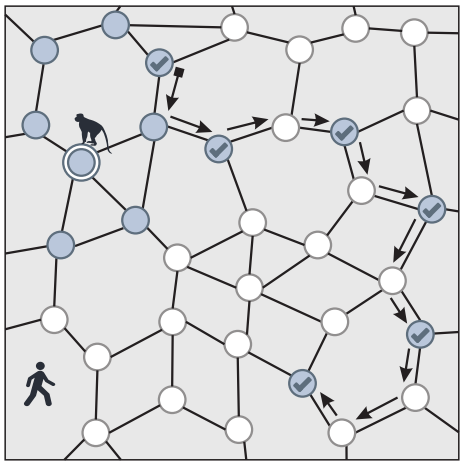
Figura 5.7. Pontos de consulta calculados pela rede para reduzir o percurso do objeto guiado.

5.2.5.1 Atalho

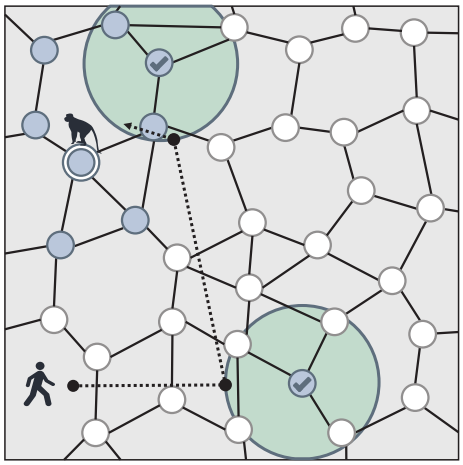
Em alguns cenários do rastreamento, principalmente quando o objeto guiado é mais lento que o alvo, a rede acumula uma sequência com muitos *beacons* para representar a trajetória do alvo. Dessa forma, o percurso do objeto guiado pode ser reduzido criando um caminho direto do primeiro para o último *beacon*.

Enquanto o alvo é rastreado, os nós ativos possuem uma variável que é incrementada sempre que o alvo muda de face, então a sequência de *beacons* é ajustada quando o alvo mudar de face k vezes. Para criar esse atalho, o nó que se torna *beacon* na *k-ésima* mudança de face envia para o seu *beacon* anterior a mensagem *TrackShortcut*. O nó que recebe essa mensagem a encaminha para o seu *beacon* anterior e depois deixa de ser *beacon*. Esse processo é repetido até que a mensagem chegue ao primeiro *beacon*. Com isso, o nó que originou a mensagem de atalho passa a ser o próximo *beacon* do primeiro *beacon*.

A figura 5.8 mostra o atalho sendo criado depois que alvo passou por seis faces. O nó que se torna *beacon* no momento em que o alvo cruza a sexta face inicia o processo de atalho. Ele envia a mensagem de atalho, passando por todos os *beacons* até chegar no primeiro (figura 5.8(a)). Todos os *beacons*, com exceção do primeiro e do último, deixam de ser *beacons*. Depois disso, o primeiro *beacon* conhece o ponto de consulta do último beacon, tornando o caminho do objeto guiado mais curto (figura 5.8(b)).



(a) O processo de atalho é iniciado pelo último *beacon* e passa por todos os *beacons* até chegar no primeiro.



(b) Todos os beacons intermediários deixam de ser *beacons*. O primeiro beacon cria uma rota direta para o último beacon.

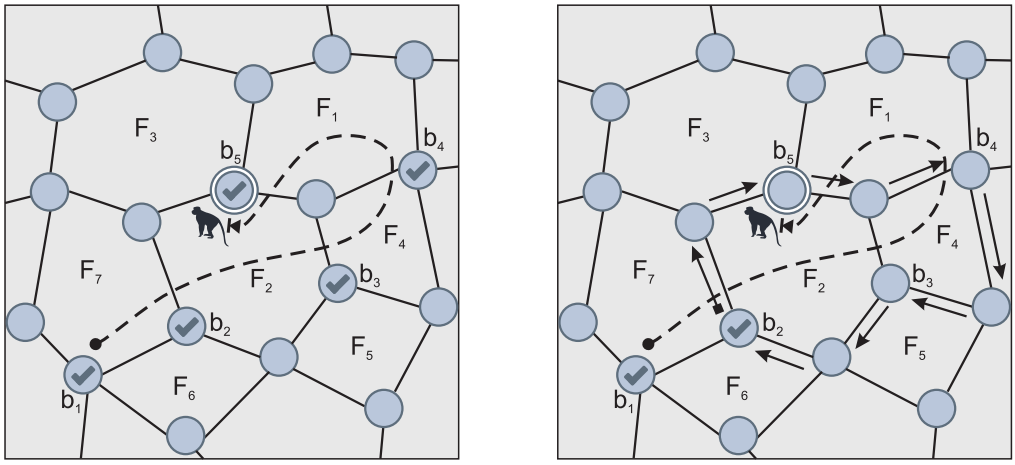
Figura 5.8. Atalho criado no caminho de *beacons* depois que o alvo passa por $k = 6$ faces.

5.2.5.2 Removendo Laços da Rota

A trajetória do alvo pode criar um caminho circular de *beacons*. Esse laço circular deve ser removido para facilitar que o alvo seja alcançado pelo objeto guiado.

O nó *beacon* b_1 descobre que o caminho de *beacons* formou um laço quando um novo *beacon* b_2 é criado para a mesma face. Isso acontece quando mensagens de *Wakeup* são enviadas para ativar as faces. Para remover esse laço, b_1 envia uma mensagem *RemoveLoop* para b_2, que por sua vez envia essa mensagem para o seu *beacon* anterior. Essa mensagem vai sendo passada para os *beacons* anteriores até voltar para b_1, que se torna o último *beacon*, enquanto os outros nós deixam de ser *beacon*.

A figura 5.9 mostra uma rota formada pela sequência de *beacons* b_1, b_2, b_3, b_4 e b_5. Essa rota forma um laço porque passa pelas faces F_7, F_2, F_4, F_1 e F_2. Os nós b_2 e b_5 são ambos *beacons* da face F_2. Assim que b_5 se torna *beacon*, o nó b_2 detecta que um laço foi criado e inicia a remoção do mesmo enviando uma mensagem para b_5. Essa mensagem segue o laço, passando por todos os *beacons* até voltar para b_2. No fim, todos os *beacons* posteriores a b_2 deixam de ser *beacons*, b_2 volta a ser o último *beacon* e b_5 o nó *near*.



(a) (b)

Figura 5.9. Removendo o laço de *beacons* formados pela trajetória do alvo.

5.3 Avaliações

Nesta seção descrevemos a metodologia de avaliação e os resultados obtidos. Comparamos o desempenho do algoritmo TATI com o DOT.

5.3.1 Metolodologia

As avaliações são realizadas por meio de simulações com ns-2. A configuração padrão da rede é composta de 81 nós sensores monitorando uma região de $200 \times 200\,\mathrm{m}^2$. Cada simulação tem duração de 2000 s, sendo que os primeiros 200 s são reservados para a configuração da rede.

Os raios de alcance dos nós da rede são de 40 m, isso possibilita que todos os nós se comuniquem através de múltiplos saltos e que as faces sejam criadas apropriadamente durante a configuração da rede. Já o raio de sensoriamento é de 30 m para garantir que existe pelos menos três nós cobrindo qualquer ponto do campo de sensores. A tabela 5.2 resume os parâmetros de simulação usados.

Tabela 5.2. Parâmetros de simulação usados nas avaliações do TATI.

Parâmetro	Valor
Campo de sensores	$200 \times 200\,\mathrm{m}^2$
Número de nós	81
Alcance dos nós	40 m
Alcance do alvo	30 m

A posição inicial do alvo é aleatória. A velocidade do alvo é de $10\,\mathrm{m/s}$ e o seu

modelo de mobilidade é o CRW com grau de correlação de 0,70, com isso a trajetória do alvo se espalha por todo o campo de sensores. O objeto guiado inicia o rastreamento no centro do campo de sensores e também tem velocidade de $10\,^{m}/_{s}$.

Cada mensagem enviada pelos nós tem tamanho 32 bytes. Modelamos o consumo de energia de acordo com os diferentes modos de operação do rádio CC2420 [Suh et al., 2006; Wang et al., 2007b]. A tabela 5.3 mostra os parâmetros de energia usados nas simulações.

Tabela 5.3. Parâmetros de energia usados nas simulações do TATI.

Parâmetro	Valor
Energia inicial	18720 joules
Sensoriamento	2×10^{-5} watts
Transmissão	3132×10^{-5} watts
Recepção	3546×10^{-5} watts
Escutando	$76{,}68\times10^{-5}$ watts
Dormindo	$3{,}6\times10^{-5}$ watts
Transição de estado	5×10^{-5} watts

Consideramos três métricas: distância, mensagens e energia. A primeira métrica é a distância euclidiana entre o alvo e o objeto guiado no decorrer do rastreamento. A segunda métrica é a quantidade de mensagens enviadas durante o rastreamento, todas as mensagens são contabilizadas, inclusive as mensagens usadas para configurar a rede. A última métrica é a energia consumida pelos nós até o final da simulação.

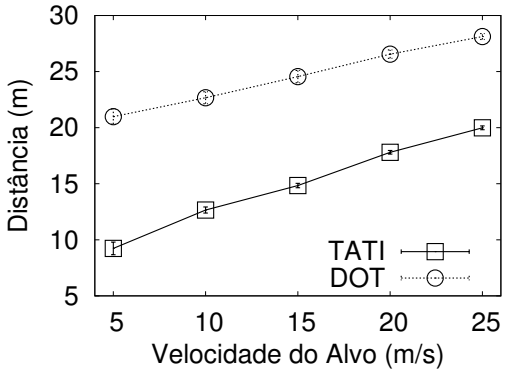
Os resultados dos experimentos são obtidos da média de 35 execuções diferentes. As barras de erro representam um intervalo de confiança de 99%.

5.3.2 Resultados das Simulações

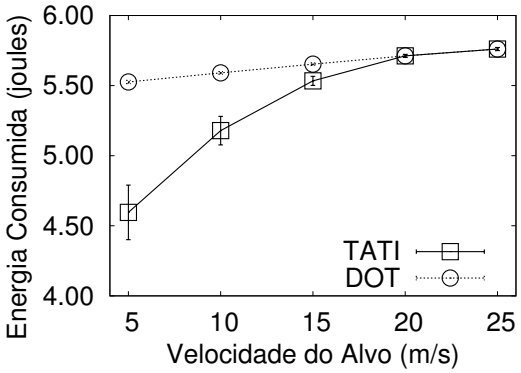
5.3.2.1 Velocidade do Alvo

A velocidade do alvo é o principal fator que deve ser avaliado no rastreamento com objeto guiado. Se a velocidade do alvo for alta, as faces podem não ser ativadas a tempo, fazendo com que o alvo seja perdido. Além disso, o atraso para detectar o alvo e calcular sua posição mantém o objeto guiado afastado do alvo. Dessa forma, Variamos a velocidade do alvo de $5\,^{m}/_{s}$ a $25\,^{m}/_{s}$, a velocidade do objeto guiado sempre é igual a velocidade do alvo (figura 5.10).

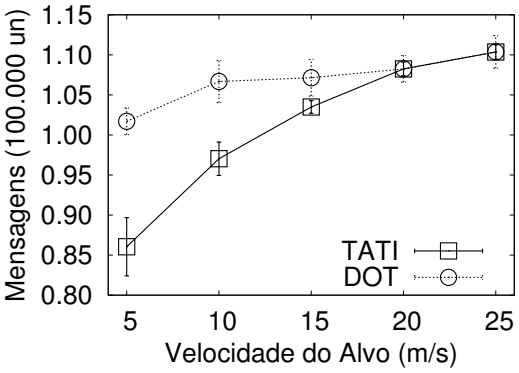
A figura 5.10(a) mostra a distância entre o alvo e o objeto guiado quando a velocidade do alvo aumenta. A distância aumenta com a velocidade porque há um atraso para detectar o alvo, calcular sua posição e informar o resultado ao objeto



(a) Distância.



(b) Energia consumida.



(c) Mensagens enviadas.

Figura 5.10. Avaliações variando a velocidade do alvo.

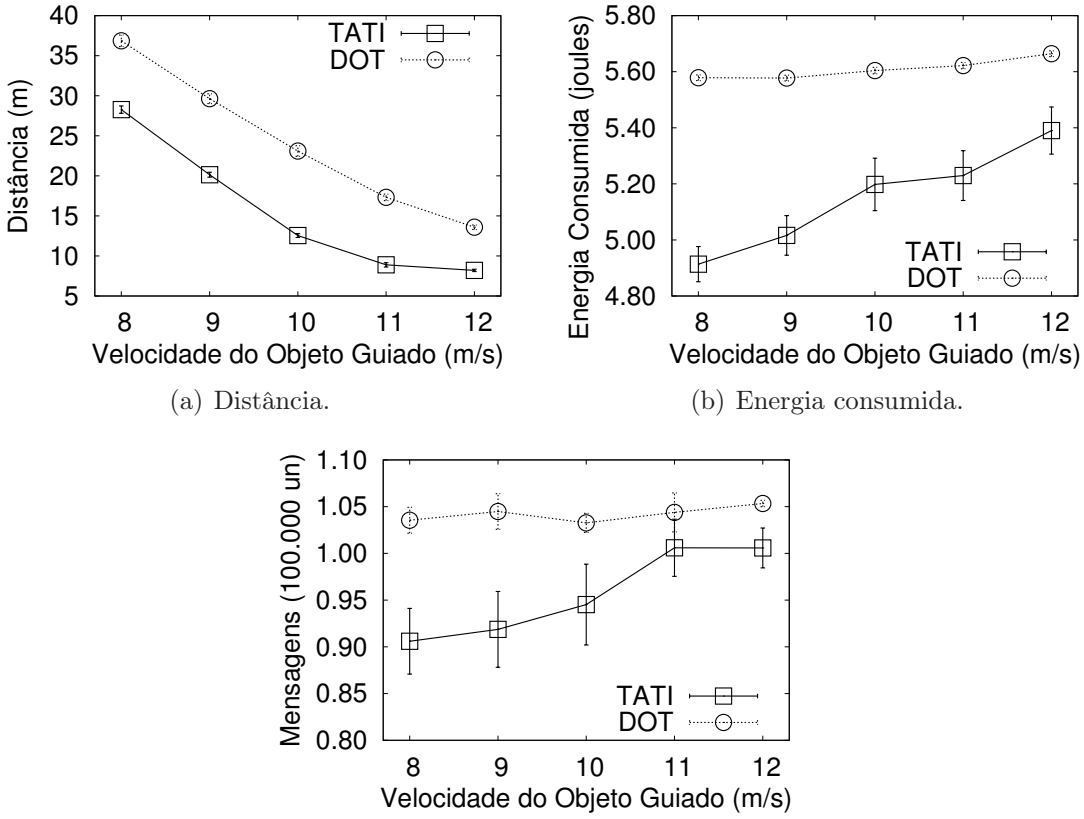
guiado. Enquanto isso, o alvo continua a se mover e quanto mais rápido é o alvo mais ele se distancia. O TATI consegue reduzir essa distância porque os pontos de consulta minimizam a rota que objeto guiado usa para consultar os *beacons*, enquanto o DOT faz o objeto guiado se dirigir até a posição do *beacon*.

A figura 5.10(b) mostra que o TATI consome menos energia que o DOT quando a velocidade do alvo é inferior a $20\,^{m}/_{s}$. Isso ocorre porque o deslocamento do alvo entre duas amostragens é pequeno o suficiente para proporcionar que o TATI ative poucas faces. Nos casos em que a velocidade é superior a $15\,^{m}/_{s}$, o consumo dos métodos avaliados é aproximado, pois o deslocamento do alvo faz com que o TATI ative todas as faces adjacentes ao nó *near*, da mesma forma que o DOT.

Uma face é ativada usando mensagens que percorrem todos os nós que a compõe. Como o TATI ativa menos faces, menos mensagens são enviadas durante o rastreamento, como mostra a figura 5.10(c). Quando a velocidade é superior a $15\,^{m}/_{s}$, a quantidade de mensagens aumenta nos dois métodos. Isso acontece porque o alvo é perdido mais vezes, sendo assim um *flooding* é feito na rede para recuperar o alvo.

5.3.2.2 Velocidade do Objeto Guiado

Nesta avaliação verificamos o impacto que a diferença de velocidade entre alvo e o objeto guiado causa no rastreamento. A velocidade do alvo é mantida em 10 m/s, enquanto a velocidade do objeto guiado varia de 8 m/s a 12 m/s (figura 5.11), para considerar os casos em que o objeto guiado é mais lento e mais rápido que o alvo.



(a) Distância.

(b) Energia consumida.

(c) Mensagens enviadas.

Figura 5.11. Avaliações variando a diferença de velocidade entre o objeto guiado e o alvo.

Na figura 5.11(a), o TATI mantém o objeto guiado mais próximo do alvo, pois os pontos de consulta minimizam a rota do objeto guiado. Quando o objeto guiado é mais lento que o alvo, ele tem dificuldade de se aproximar, mesmo que a rota seja minimizada, pois o alvo se move constantemente e se mantém afastado devido a velocidade superior.

Mesmo quando o objeto guiado é mais rápido que o alvo, ele ainda fica afastado do alvo por alguns metros (aproximadamente 10 m com TATI e 15 m com o DOT). Isso acontece porque o objeto guiado alcança o último *beacon*. Nesse caso ele se aproxima do alvo enquanto estiver dentro do raio de alcance do *beacon*. Depois disso deve esperar até que outro *beacon* seja criado, nesse tempo o alvo se afasta.

Na figura 5.11(b), o TATI consome menos energia que o DOT porque usa menos faces ativas durante o rastreamento. Ambos os métodos consomem mais energia quando a velocidade do objeto guiado aumenta, principalmente com o TATI. A razão disso é que o objeto guiado, quando está próximo do alvo, consulta frequentemente o último *beacon* para continuar a aproximação. Como o DOT tem mais dificuldade de fazer o objeto guiado alcançar o último *beacon*, não há muita diferença do consumo de energia quando a velocidade do objeto guiado é maior que a velocidade do alvo.

O TATI usa menos mensagens que o DOT porque envia menos mensagens para ativar as faces, como mostra a figura 5.11(c). O TATI aumenta a quantidade de mensagens quando a velocidade do objeto guiado é superior a velocidade do alvo ($11\,^{m}/_{s}$ e $12\,^{m}/_{s}$), pois consulta frequentemente o último *beacon* para manter a aproximação.

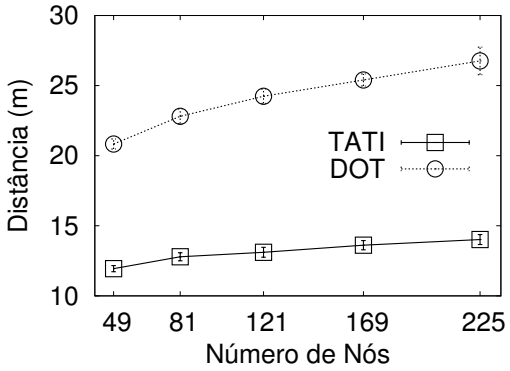
5.3.2.3 Escalabilidade

A escalabilidade é avaliada variando o tamanho do campo de sensores de $150\times150\,m^2$ a $350\times350\,m^2$ com uma densidade constante de $0{,}002\,^{nós}/_{m^2}$. Dessa forma, variamos a quantidade de nós de 49 até 225 nós. Os resultados da avaliação de escalabilidade são mostrados na figura 5.12.

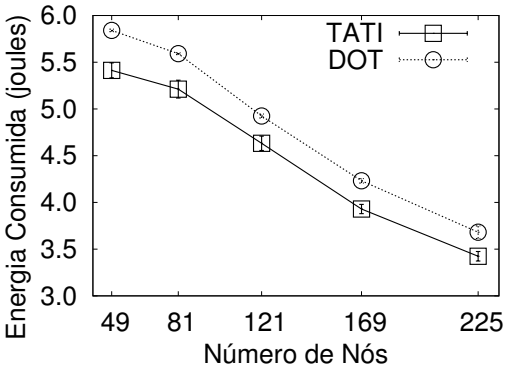
Na figura 5.12(a) a distância entre o alvo e o objeto guiado aumenta com a escala da rede. Quanto maior é o campo de sensores maior é a distância entre o alvo e o objeto guiado no início do rastreamento. Dessa forma o objeto guiado leva mais tempo para alcançar o alvo, o que aumenta a distância média. O TATI mantém o objeto guiado mais próximo do alvo porque minimiza a rota usando os pontos de consulta.

A figura 5.12(b) mostra que a energia consumida pela rede diminui com o aumento da escala. Isso acontece porque a quantidade de nós aumenta com a escala, porém a quantidade de nós que efetivamente rastreia o alvo permanece a mesma. Em outras palavras, aproximadamente a mesma quantidade de energia é consumida durante o rastreamento, mas ela é distribuída entre uma quantidade maior de nós, logo o consumo por nó é menor.

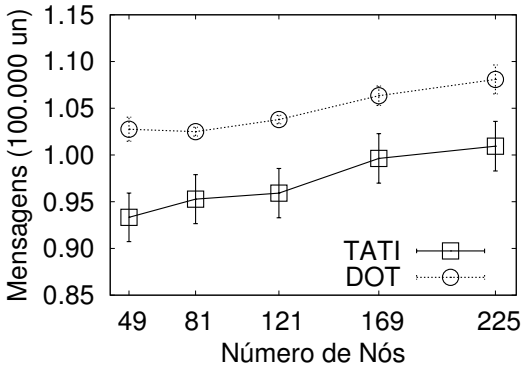
A figura 5.12(c) mostra que a quantidade de mensagens usadas no rastreamento aumenta com a escala da rede. Isso acontece porque mais mensagens são usadas no *flooding* para encontrar o alvo no início do rastreamento ou para recuperá-lo caso ele seja perdido. Além disso, mais mensagens são necessárias para gerar as faces na fase de configuração da rede.



(a) Distância.



(b) Energia consumida.



(c) Mensagens enviadas.

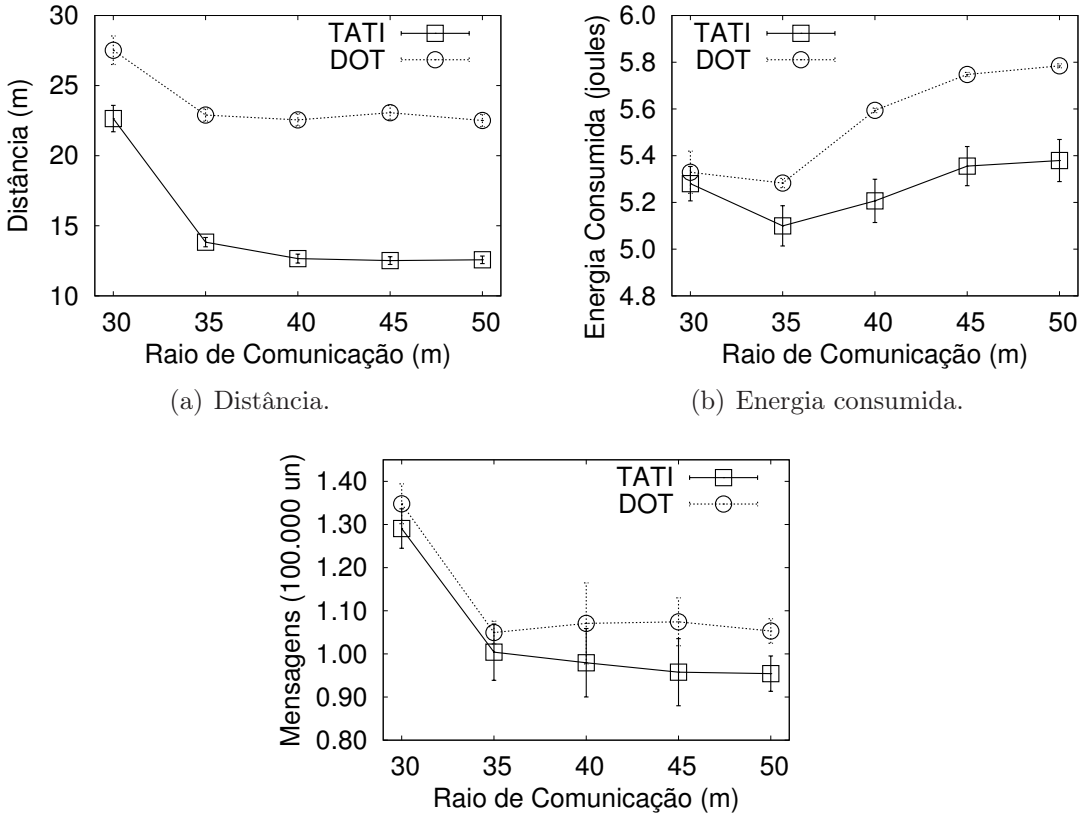
Figura 5.12. Avaliação de escalabilidade.

5.3.2.4 Raio de Comunicação

O raio de comunicação dos nós modifica a quantidade de vizinhos dos nós e consequentemente a formação das faces. Nesta avaliação variamos o raio de comunicação dos nós de 30 m a 50 m. Os resultados são apresentados na figura 5.13.

A figura 5.13(a) mostra que a distância entre o alvo e o objeto guiado é maior quando o raio de comunicação dos nós é de 30 m, essa distância diminui e estabiliza nos demais casos. No TATI e no DOT, os nós que detectam o alvo divulgam essa informação por *broadcast*, logo a fusão de dados em um nó ocorre somente com os dados de seus vizinhos. O ideal é que o raio de comunicação dos nós seja pelos menos duas vezes o raio de sensoriamento para que um nó reúna os dados de todos os nós que detectaram o alvo. Dessa forma, se o raio de comunicação é de 30 m, os nós muitas vezes não possuem dados suficientes para calcular a posição do alvo, então ele é frequentemente perdido pela rede. O tempo necessário para detectar que o alvo foi perdido e recuperá-lo é suficiente para que ele se afaste do objeto guiado.

Nos casos em que o raio de comunicação é superior a 30 m, os nós reúnem da-



(a) Distância.

(b) Energia consumida.

(c) Mensagens enviadas.

Figura 5.13. Avaliações aumentando o raio de comunicação dos nós.

dos suficientes para calcular a posição do alvo, por isso o objeto guiado consegue se aproximar do alvo de forma mais efetiva.

Na figura 5.13(b) o consumo de energia aumenta juntamente com o raio de comunicação. Isso acontece porque mais nós irão receber uma mesma mensagem, mesmo que ela seja descartada depois. Uma exceção ocorre quando o raio de comunicação é de 30 m, pois supera o consumo de quando o raio de comunicação é de 35 m. Isso ocorre porque o alvo é perdido com mais frequência, fazendo com que toda a rede seja consultada para recuperar o alvo. Por essa mesma razão, mais mensagens são enviadas quando o raio de comunicação é de 30 m, pois o *flooding* é usado mais vezes para recuperar o alvo, como mostra a figura 5.13(c).

Nos demais casos da figura 5.13(c), a quantidade de mensagens utilizadas é aproximada, pois as mensagens percorrem a sequência das faces, então aproximadamente a mesma quantidade de mensagens é enviada independente do raio de comunicação. Entretanto, a variação é reduzida quando o raio de comunicação é grande (50 m), pois as faces geradas ficam com tamanhos aproximados.

5.4 Conclusões Parciais

Neste capítulo propomos e avaliamos o TATI, usando o DOT como *baseline*. O TATI é um algoritmo distribuído para rastreamento de alvo em redes de sensores, criando uma rota que é informado ao objeto guiado que objetiva interceptar o alvo.

Esse algoritmo reduz o consumo de energia por ativar menos faces, assim economiza em mensagens de ativação e menos nós permanecem ativos para detectar o alvo. Além disso, reduz a rota percorrida pelo objeto guiado usando pontos de consulta. Esse pontos são a menor distância entre a posição do objeto guiado e a área de comunicação dos *beacons*.

O TATI registra o deslocamento máximo do alvo entre duas amostragens para escolher as faces que devem ser ativadas. Portanto, o TATI ativa menos faces que o DOT quando a velocidade do alvo impossibilita que ele se desloque para determinadas faces até a próxima amostragem. Entretanto o consumo de energia desses métodos é aproximado quando a velocidade do alvo é alta ($20\,^{\mathrm{m}}/_{\mathrm{s}}$ ou mais).

No geral, o TATI mantém o objeto guiado mais próximo do alvo, pois usa a informação do raio de comunicação nominal dos nós para reduzir o percurso do objeto guiado. Ele envia o objeto guiado para o ponto mais próximo em que é possível consultar o *beacon* da vez. Além disso, evita que o objeto guiado se desloque desnecessariamente se ele já está na área de comunicação do *beacon*.

Capítulo 6

Reduzindo o Erro de Localização Provido por Sistemas de Localização Basedos em Distância Durante a Aplicação do Rastreamento

As aplicações para rastreamento de alvos em redes de sensores usam as localizações dos nós sensores para calcular a posição do alvo [Tsai et al., 2007; Hsu et al., 2012; Hajiaghajani et al., 2012]. Por outro lado, os nós sensores nem sempre conhecem a sua localização precisamente [Oliveira et al., 2009; Souza et al., 2013].

A localização dos nós é importante, por exemplo, para identificar e correlacionar os dados coletados, gerenciar e consultar nós localizados em uma determinada região, verificar a densidade e cobertura dos nós, criar rotas entre os nós, e gerar mapas de energia [Boukerche et al., 2007]. Sendo assim, a localização dos nós também é requerida por várias outros tipos de algoritmos de redes de sensores, e.g. roteamento [Huang et al., 2005; Tan & Kermarrec, 2011] e controle de densidade [Zhang & Hou, 2005; Campos et al., 2012]. Esses algoritmos são chamados de algoritmos geográficos [Oliveira et al., 2009].

O GPS [Goswami, 2013] pode ser usado para obter a localização dos nós, mas equipar todos os nós com um receptor de GPS representa um alto custo que torna sua utilização inviável em redes de sensores [Fang et al., 2007]. Além disso, As redes de sensores podem ser implantadas em grande escala e em locais de difícil acesso,

impossibilitando que a localização dos nós seja pré-determinada [Akyildiz et al., 2002]. Esses problemas criaram a necessidade de se desenvolver algoritmos que utilizem a capacidade de cooperação dos nós para obter a suas localizações de forma distribuída. Esses algoritmos são chamados de algoritmos de localização [Boukerche et al., 2007].

Os algoritmos de localização geralmente usam um conjunto de nós que possuem a capacidade de auto-localização, chamados nós *beacons*, utilizando GPS ou configuração manual. Os outros nós da rede calculam as suas localizações com base nos dados de localização fornecidos pelos nós *beacons* [Sheu et al., 2008; Oliveira et al., 2009; Yifeng & Lamont, 2011].

Os algoritmos de localização podem ser classificados como não baseados em alcance (*range-free*) [Niculescu & Nath, 2003; Gui et al., 2012] e baseados em alcance (*range-based*) [Albowicz et al., 2001; Oliveira et al., 2009], esses se diferem nos dados usados para calcular as posições dos nós. Os algoritmos não baseados em alcance assumem que os nós são distribuídos uniformemente pelo campo de sensores, de forma que a quantidade de saltos entre dois nós represente a distância entre eles. Já os algoritmos baseados em alcance assumem que os nós são equipados com algum dispositivo que possibilita medir a distância entre os nós. O *Received Signal Strength Indicator* (RSSI) é a técnica mais comum para estimar a distância entre os nós, pois não precisa que o nó sensor seja equipado com hardware adicional [Oliveira et al., 2009].

A posição estimada por esses algoritmos possuem erros que podem prejudicar o funcionamento dos algoritmos geográficos [Huang et al., 2009, 2013]. Os algoritmos não baseados em alcance, por exemplo, têm seu desempenho prejudicado por irregularidades na topologia da rede e no raio de comunicação dos nós [Sheu et al., 2008]. O desempenho dos algoritmos baseados em alcance são prejudicados por ruídos presentes nas distâncias estimadas [Huang et al., 2009; Yifeng & Lamont, 2011].

Niculescu & Nath [2003] apresentam a solução pioneira para localização não baseada em distância. Nesse algoritmo, todos os nós da rede obtêm a quantidade de saltos até os nós *beacons*. Com isso, esse estima a distância média de um salto para calcular a localização dos nós com trilateração. Esse algoritmo é melhorado por Chen et al. [2008], esses usam um esquema de correção de erro diferencial para reduzir o erro acumulado durante os múltiplos saltos. Fang et al. [2007] apresentam um algoritmo de localização não baseado em distância que dispensa a utilização de nós *beacons*. Esse considera que os nós são implantados em grupos, sendo que os nós de um mesmo grupo são implantados ao redor de uma posição conhecida e com uma probabilidade de distribuição Gaussiana. Com isso, esse calcula a localização dos nós usando estimação estatística, observando os grupos dos vizinhos de cada nó.

Albowicz et al. [2001] propõem a solução pioneira para localização baseada em

alcance. Nesse algoritmo, os nós *beacons* divulgam suas localizações para ajudar outros nós a se localizarem. Os nós que se localizam se tornam referências para outros nós. Com base nessa solução, Oliveira et al. [2009] propõem um algoritmo que usa estruturas de *beacons* para fazer a recursão iniciar de um único ponto e seguir uma direção conhecida. Com isso, um nó pode estimar sua localização usando apenas duas referências.

Alguns trabalhos encontrados na literatura avaliam o impacto dos erros de localização nos algoritmos geográficos. Oliveira et al. [2009] avalia o efeito dos erros de localização nos algoritmos de roteamento geográfico. Souza et al. [2009] e Campos et al. [2012] avaliam o desempenho de aplicações de rastreamento na presença de erros de localização. Os erros de localização avaliados nesses trabalhos foram gerados pela execução de algoritmos de localização baseados em alcance [Albowicz et al., 2001; Oliveira et al., 2009].

Alguns trabalhos reduzem o erro dos algoritmos de localização. Taylor et al. [2006] e Teng et al. [2012] reduzem o erro de localização dos nós enquanto a aplicação de rastreamento é executada. Esse considera que os nós são pré-localizados usando um algoritmo de localização não baseado em alcance e sem a utilização de nós *beacons*, de forma similar ao algoritmo de localização de Fang et al. [2007]. Dessa forma, esse reduz o erro de localização observando as inconsistências geométricas entre as localizações dos nós e do alvo durante o rastreamento. Souza et al. [2013] reduzem o erro de localização dos nós de algoritmos baseados em alcance. Para isso, esse realiza múltiplas estimativas de distância durante a localização para filtrar os ruídos e obter uma estimativa de distância mais precisa antes do rastreamento ser iniciado. Nesse caso, a aplicação deve definir o *trade-off* entre custo e precisão.

A abordagem apresentada neste capítulo é uma extensão do trabalho de Souza et al. [2013]. O objetivo é filtrar os ruídos presentes nas distâncias estimadas usadas por algoritmos de localização baseados em alcance. Com isso, a localização dos nós e, consequentemente, o rastreamento ficam mais precisos. O processo de filtragem e a aplicação de rastreamento são executados simultaneamente e não exigem custo de comunicação adicional.

Antes do rastreamento iniciar, os nós se localizam usando um algoritmo de localização baseado em alcance, e guardam os dados sobre as suas referências. Durante o rastreamento, os nós que detectam o alvo trocam mensagens para compartilhar a suas observações sobre o evento. Essa cooperação é necessária para calcular a posição do alvo. Em nossa abordagem, antes de calcular a posição do alvo, os nós aproveitam essas mensagens para filtrar as distâncias estimadas das suas referências e melhorar as suas localizações. A posição do alvo é calculada usando as novas posições dos nós como

referências. Dessa forma, os erros de localização dos nós são reduzidos por onde o alvo passa, e o rastreamento melhora ao longo do tempo.

As soluções e resultados deste capítulo estão parcialmente publicados no Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos [Souza et al., 2014a] e em avaliação no *Local Computer Networks*. Uma uma versão estendida está em desenvolvimento para publicação no *Journal of the Brazilian Computer Society*.

No restante deste capítulo, detalhamos o funcionamento da abordagem de localização e rastreamento simultâneos e apresentamos a metodologia experimental e os resultados das avaliações.

6.1 Algoritmos de Localização

Localização consiste em determinar a localização física (e.g., latitude, longitude e altitude) dos nós sensores da rede, sendo essencial que este processo seja desempenhado pelos próprios nós (auto-localização). Neste trabalho, usamos algoritmos de localização baseados em alcance para pré-localizar os nós. As localizações dos nós serão corrigidas durante o rastreamento.

Nesse tipo de localização, os nós podem ser classificados como: (1) beacons – nós que conhecem precisamente e antecipadamente as suas localizações, pois são equipados com GPS ou configurados manualmente; (2) estabelecidos – nós que calculam as suas posições durante a execução do algoritmo de localização; (3) livres – nós que não conhecem a sua localização.

Um nó *beacon* ou estabelecido se torna referência quando divulga a sua localização para ajudar os nós livres a se localizarem. Um nó livre se localiza usando a posição dos nós referências e as distâncias entre ele e cada uma das suas referências.

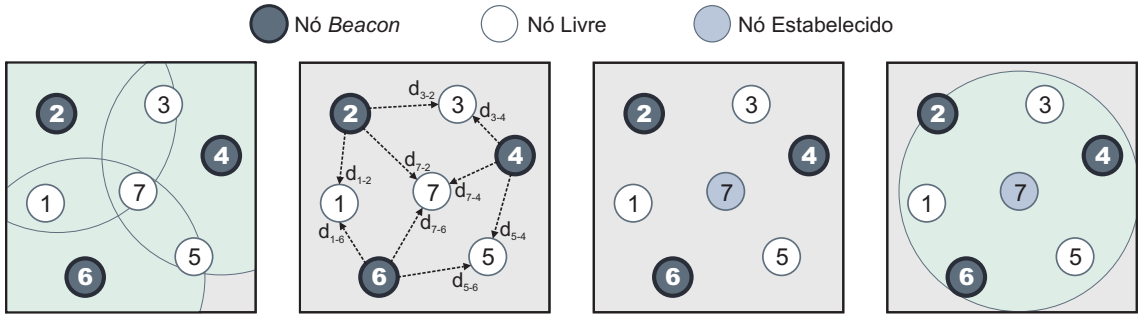
Neste trabalho, usamos os algoritmos RPE [Albowicz et al., 2001] e o DPE [Oliveira et al., 2009] para estimar a posição dos nós. Esses algoritmos são baseados em alcance, por isso, o erro resultante dessas técnicas são consequência da imprecisão das distâncias estimadas entre os nós, geralmente obtidas por RSSI. A seguir descrevemos o funcionamento desses algoritmos.

6.1.1 Estimativa de Localização Recursiva

O RPE é um algoritmo de localização que utiliza um conjunto de nós *beacons* espalhados pelo campo campo de sensores. Os nós *beacons* divulgam as suas localizações para ajudar os outros nós a se localizarem. Um nó livre calcula a sua localização quando obtém informações de pelo menos três referências. Depois disso, esse nó se torna

estabelecido e divulga a sua localização para servir de referência para outros nós. Dessa forma, o número de referências aumenta iterativamente, fazendo com que a maioria dos nós tenham as suas localizações estimadas.

Esse algoritmo pode ser dividido em quatro fases, como mostra a figura 6.1. Na primeira fase, os nós *beacons* iniciam a recursão enviando mensagens que contêm as suas localizações. Na segunda fase, o nó livre que recebe uma mensagem de localização estima a distância entre ele e a origem da mensagem (nó referência). Os nós que já estão localizados ignoram essas mensagens. Na terceira fase, os nós livres com três referências ou mais calculam suas localizações usando trilateração ou multilateração, então se tornam nós estabelecidos. Na última fase, os nós estabelecidos divulgam suas localizações para auxiliar os nós livres a serem localizados.



(a) Fase 1 (b) Fase 2 (c) Fase 3 (d) Fase 4

Figura 6.1. Fases da localização recursiva.

No caso ilustrado na figura 6.1, depois que os nós *beacons* divulgam suas localizações, apenas o nó 7 consegue as três referências necessárias para calcular sua localicação. Os nós livres restantes possuem apenas duas referências cada, por isso permanecem aguardando por mais referências. O nó 7 calcula sua localização e a divulga. Com isso, os nós livres completam as suas referências, agora eles podem ser localizados e podem ajudar outros nós livres.

Os nós estabelecidos também são usados como referências, isso é vantajoso porque mesmo os nós distantes dos *beacons* podem ser localizados. Por outro lado, a localização dos nós estabelecidos possuem erros que são propagados e acumulados, por isso os nós mais distantes dos *beacons* possuem maiores erros de localização.

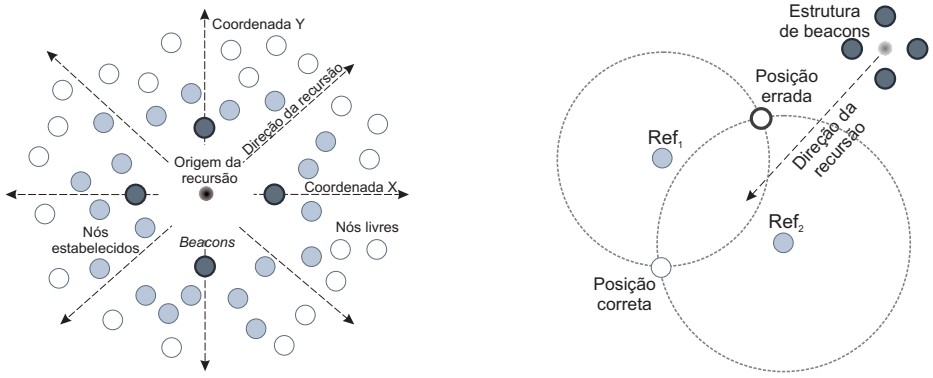
Por fim, o RPE é um algoritmo que trabalha de forma distribuída usando múltiplos saltos. Ele não necessita de infra-estrutura, por isso é adequado para ambientes externos.

6.1.2 Estimativa de Localização Direcionada

O DPE é um algoritmo similar ao RPE. A ideia do DPE é fazer a recursão do sistema iniciar de um único ponto, fazendo ela seguir uma direção conhecida. Isso possibilita que um nó estime sua localização usando somente duas referências e trabalhar em redes de baixa densidade. Além disso, o caminho controlado da recursão leva a estimativas de localização com erros menores.

Para garantir que a recursão seja iniciada de um único ponto, o algoritmo faz uso de uma estrutura de *beacons* como mostra da figura 6.2(a). Essa estrutura possui, geralmente, quatro *beacons* que ficam a uma distância conhecida da origem da recursão. Esses *beacons* iniciam a execução do algoritmo divulgando suas localizações aos seus vizinhos.

Com o uso de somente duas referências, as localizações dos nós são calculadas fazendo a interseção de dois círculos, obtidos pelas localizações e distâncias das referências. Logo um par de possíveis soluções resultam do sistema: a localização correta e a incorreta (ver figura 6.2(b)). Uma vez que a direção da recursão é mantida, a localização correta do nó livre é o ponto mais distante da origem da recursão. Quando o nó livre possui mais de duas referências, ele escolhe as que estão mais distantes entre si e mais próximas da origem da recursão, essa escolha proporciona cálculos de posição mais consistentes.



(a) Recursão direcionada. (b) Uso de duas referências.

Figura 6.2. Localização direcionada.

Da mesma forma que o RPE, o DPE também propaga os erros de localização. Então os nós mais distantes da estrutura de *beacons* têm maior probabilidade de terem erros de localização maiores. Para reduzir a propagação de erros, mais estruturas de *beacons* podem ser acrescentadas a rede, isso reduz o número de saltos e, consequentemente, melhora o desempenho final do algoritmo.

6.2 Considerações de Definições

Os nós sensores são distribuídos aleatoriamente pelo campo de sensores. Todos os nós possuem os mesmos raios de comunicação e sensoriamento. Os nós sensores estão sincronizados no tempo. Alguns nós são *beacons*, esses são necessários para iniciar o processo de localização.

Um algoritmo de localização baseado em alcance é executado para localizar os nós. Durante a localização, os nós estimam as distâncias entre eles e suas referências. Essas distâncias estimadas possuem ruídos que são responsáveis pelo erro de localização que se acumula a cada iteração do algoritmo de localização.

A rede de sensores é aplicada para rastrear um alvo (pessoa, animal ou veículo) específico que se move aleatoriamente pelo campo de sensores. Os nós periodicamente consultam seus sensores para verificar se podem detectar o alvo. O modelo de detecção binário é usado para detectar o alvo, i.e., um nó sempre detecta o alvo se ele está dentro do seu raio de sensoriamento.

Para reduzir o tráfego de comunicação, os nós que detectam o alvo formam um agrupamento. Consideramos que o raio de comunicação é duas vezes o raio de sensoriamento, com isso apenas um agrupamento é formado em cada amostragem e todos os membros do agrupamentos podem se comunicar com apenas um salto.

Os agrupamentos são formados dinamicamente de acordo com a movimentação do alvo. Cada nó que detecta o alvo estima a distância entre ele e o alvo, depois disso envia uma mensagem contendo a sua localização e a distância estimada para o alvo. Essa distância estimada também está sujeita à ruídos que interferem no cálculo de posição do alvo. Com isso, cada nó passa a conhecer a localização de todos os membros do agrupamento e a que distância cada um deles está do alvo. Todos os nós calculam a posição do alvo com multilateração [Oliveira et al., 2009] e verificam qual nó está mais próximo do alvo. Somente o nó mais próximo do alvo reporta a posição do alvo para o nó *sink*.

O nó sink recebe as posições do alvo gerados pela rede e faz previsões com esses dados usando KF, PF e ABF. As previsões também são afetadas pelos erros de localização. O Filtro de Kalman tem seu sistema de equações lineares configurado como a equação 3.3. O Filtro de Partículas utiliza 1000 partículas e é representado pelo algoritmo 3.1. No Filtro Alfa-Beta $\alpha = \beta = 0,99$.

6.3 Abordagem para Melhorar a Localização Estimada

Neste trabalho, aproveitamos as mensagens enviadas durante o rastreamento para melhorar a localização dos nós e, consequentemente, melhorar o resultado do rastreamento. A localização dos nós é refinada ao longo do tempo, à medida que o alvo é rastreado, sem utilizar comunicação adicional. Os nós que não puderam ser localizados durante a execução do algoritmo de localização também podem ser localizados durante o rastreamento.

Nos algoritmos de localização baseados em distância, os ruídos presentes nas distâncias estimadas entre os nós são responsáveis pelos erros de localização. Dessa forma, neste trabalho, refinamos a localização dos nós filtrando esses ruídos para melhorar a distância estimada entre os nós. Souza et al. [2009] também filtra os ruídos das distâncias estimadas entre os nós para reduzir o erro de localização. Entretanto o custo de localização aumenta, pois precisa que cada nó envie múltiplas mensagens durante a execução do algoritmo de localização.

A figura 6.3 mostra que cada nó sensor possui dois módulos: o módulo de localização e o módulo de rastreamento. O módulo de localização possui as rotinas que são executadas pelos nós durante a localização. Esse módulo armazena a localização do nó e os dados sobre as referências usadas para calcular essa localização. O módulo de rastreamento possui as rotinas necessárias para calcular a posição do alvo. O módulo de rastreamento precisa da localização do nó para calcular a posição do alvo, para obter essa informação, ele deve consultar o módulo de localização.

Cada nó que detecta o alvo inicia um *timer* e divulga uma mensagem do tipo *TargetDetection*, essa contém a sua localização e a distância entre ele e o alvo. Enquanto o *timer* não expirar, o nó fica esperando por mensagens de outros nós que também detectaram o alvo. Essa troca de informações é necessária para calcular a posição do alvo. Nossa abordagem tira proveito dessa troca de mensagens para filtrar os ruídos presentes nas distâncias estimadas entre os nós.

Quando o nó recebe uma mensagem *TargetDetection*, ele adiciona os dados dessa mensagem como referência de posição do alvo. Depois disso, o nó verifica no seu módulo de localização se o nó que originou essa mensagem é uma referência de localização. Caso positivo, a distância estimada dessa referência é filtrada e atualizada na lista de referências.

Para filtrar os ruídos das distâncias estimadas das referências, cada referência de

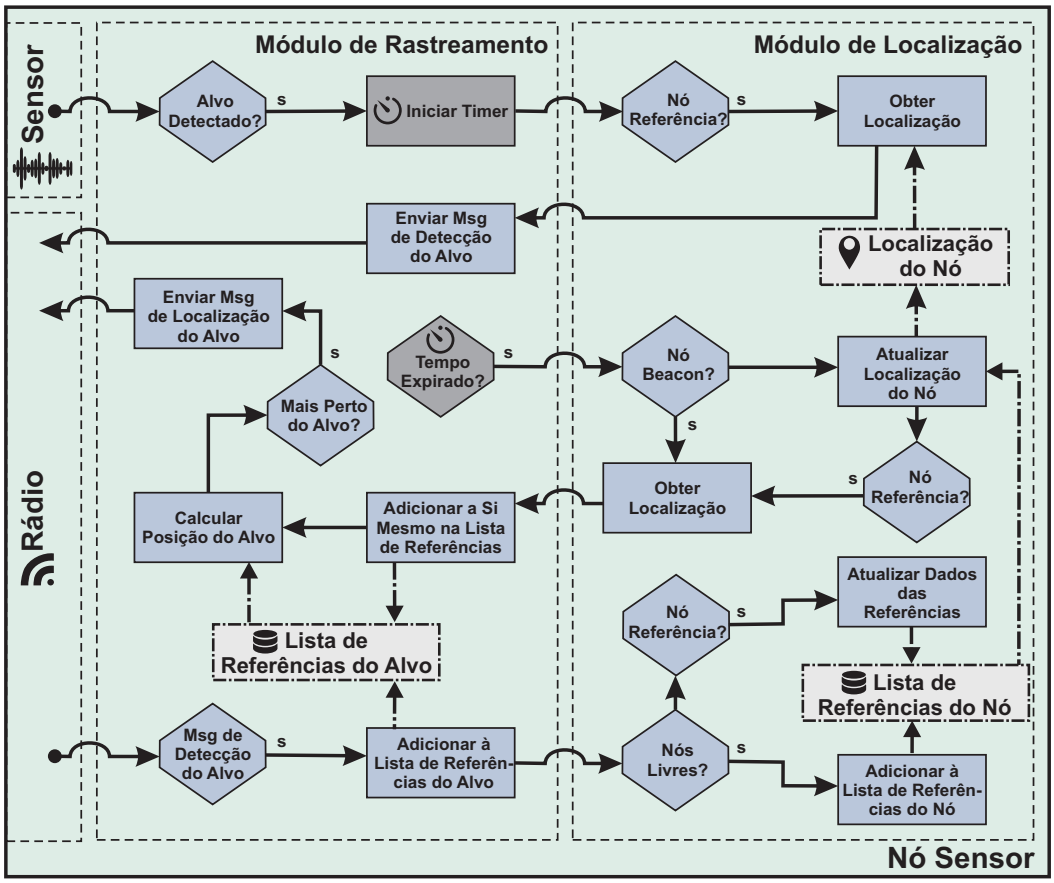


Figura 6.3. Comunicação entre os módulos de rastreamento e localização.

localização de um nó possui uma instância de KF configurada como

$$\begin{cases} x_{k+1} = d_{k+1} = \left[1\right] \times \left[d_k\right] + w_k \\ y_k = \left[1\right] \times \left[d_k\right] + v_k, \end{cases} \tag{6.1}$$

em que $x = d$ representa o estado, distância nesse caso, de um evento k; y é o valor de medição; w e v são os ruídos do processo e da medição, respectivamente.

Sempre que um nó estabelecido recebe uma mensagem de uma referência de localização, ele usa a potência de sinal recebida da mensagem para estimar uma nova distância entre ele e a referência. Essa distância é passada como medição para o KF correspondente, que por sua vez filtra os ruídos e retorna uma estimativa de distância mais precisa. O KF calcula iterativamente a média das distâncias estimadas, dessa forma o nó não precisa armazenar as informações de distância e refazer o cálculo sempre que uma mensagem nova for recebida.

Em algumas situações, um nó não consegue ser localizado durante o algoritmo de localização. Isso é inevitável quando o nó está isolado de forma que não obtém

as referências necessárias para calcular a sua localização. Colisão de pacotes durante a localização também pode fazer com que o nó não complete as referências para se localizar. Isso também pode acontecer devido ao erro das estimativas de distâncias que tornam inviável o calculo de posição. Nossa abordagem, além de reduzir o erro de localização, possibilita localizar os nós livres durante o rastreamento. As mensagens do tipo *TargetDetection* possuem todas as informações necessárias para a localização. Então um nó livre adiciona os nós que originam essas mensagens como referências e depois calculam a sua localização.

Quando as referências são atualizadas, as melhores referências são escolhidas de acordo com os critérios do algoritmo de localização. No caso do RPE as melhores referências são aquelas que possuem menor valor residual [Albowicz et al., 2001]. No caso do DPE são escolhidas as duas referências mais distantes entre si e mais próximas da origem da recursão [Oliveira et al., 2009].

A abordagem apresentada reduz os erros de localização gerados por ruídos presentes nas estimativas de distância. Para isso, tira proveito das mensagens enviadas durante o rastreamento, então não há custo de comunicação adicional. Nossa abordagem foi aplicada com uma aplicação de rastreamento, mas pode ser adaptada para qualquer aplicação geográfica em que os nós precisam divulgar as suas localizações.

6.4 Avaliações

6.4.1 Metodologia

As avaliações são realizadas por meio de simulações com ns-2. A configuração padrão da rede é composta de 289 nós sensores monitorando uma região de $200 \times 200\,\mathrm{m}^2$. Cada simulação tem duração de $10^4\,\mathrm{s}$, sendo que os primeiros 10 s são reservados para o algoritmo de localização.

Os raios de comunicação dos nós da rede são de 30 m. Isso possibilita que todos os nós se comuniquem através de múltiplos saltos e garante a cobertura necessária para localizar a maioria dos nós. Já o raio de sensoriamento é de 15 m. Isso garante, na maioria dos casos, que existe pelos menos três nós detectando o alvo independente da sua posição no campo de sensores. A tabela 6.1 resume os parâmetros de simulação usados.

As localizações dos nós são estimadas com RPE ou DPE. No RPE 10% dos nós são *beacons*, enquanto no DPE são usadas quatro estruturas de *beacons*. Para simular os erros de RSSI nas distâncias estimadas entre os nós [Oliveira et al., 2009], cada amostra de distância é perturbada por uma variável Gaussiana de média zero e desvio

Tabela 6.1. Parâmetros de simulação usados nas avaliações de localização e rastreamento simultâneos.

Parâmetro	Valor
Campo de sensores	$200 \times 200\,\mathrm{m}^2$
Número de nós	289
Alcance dos nós	30 m
Alcance do alvo	15 m

padrão de 8% da distância. As distâncias estimadas entre os nós e o alvo também são perturbadas por uma variável Gaussiana de média zero e desvio padrão de 6% da distância.

A posição inicial do alvo é aleatória. A velocidade do alvo é de $1\,{}^{\mathrm{m}}/_{\mathrm{s}}$ e o seu modelo de mobilidade é o CRW com grau de correlação de 0,70, com isso o alvo percorre todo o campo de sensores.

Cada mensagem enviada pelos nós tem tamanho 32 bytes. Modelamos o consumo de energia de acordo com os diferentes modos de operação do rádio CC2420 [Suh et al., 2006; Wang et al., 2007b]. A tabela 6.2 mostra os parâmetros de energia usados nas simulações.

Tabela 6.2. Parâmetros de energia usados nas simulações de localização e rastreamento simultâneos.

Parâmetro	Valor
Energia inicial	18720 joules
Sensoriamento	2×10^{-5} watts
Transmissão	3132×10^{-5} watts
Recepção	3546×10^{-5} watts
Escutando	$76{,}68 \times 10^{-5}$ watts

Consideramos cinco métricas: erro de localização, erro de estimação, erro de previsão, mensagens e energia. O erro de localização é a distância Euclidiana entre a posição real do nó e a posição estimada pelo algoritmo de localização. O erro de estimação é a distância Euclidiana entre a posição real do alvo e e a posição estimada pelo algoritmo de rastreamento. O erro de previsão é a distância Euclidiana entre a posição real do alvo e a previsão. A métrica de mensagens contabiliza todas as mensagens enviadas para localizar os nós e para rastrear o alvo. A métrica de energia é a energia consumida pelos nós até o final da simulação.

Os resultados dos experimentos são obtidos da média de 35 execuções diferentes. As barras de erro representam um intervalo de confiança de 99%.

6.4.2 Resultados das Simulações

6.4.2.1 Avaliação Geral – Localização, Rastreamento e Tempo

A abordagem apresentada reduz o erro de localização filtrando as estimativas de distância usando as mensagens enviadas durante o rastreamento. Portanto, os erros de localização e rastreamento são reduzidos ao longo do tempo. As figuras 6.4 e 6.6 mostram a distribuição dos erros de localização ao longo do tempo em intervalos de 2500 s. Nessas figuras, os quadrados são os nós *beacons*, as cruzes são os nós livres, e os círculos representam a localização real dos nós estabelecidos, sendo que as setas apontam para a localização estimada.

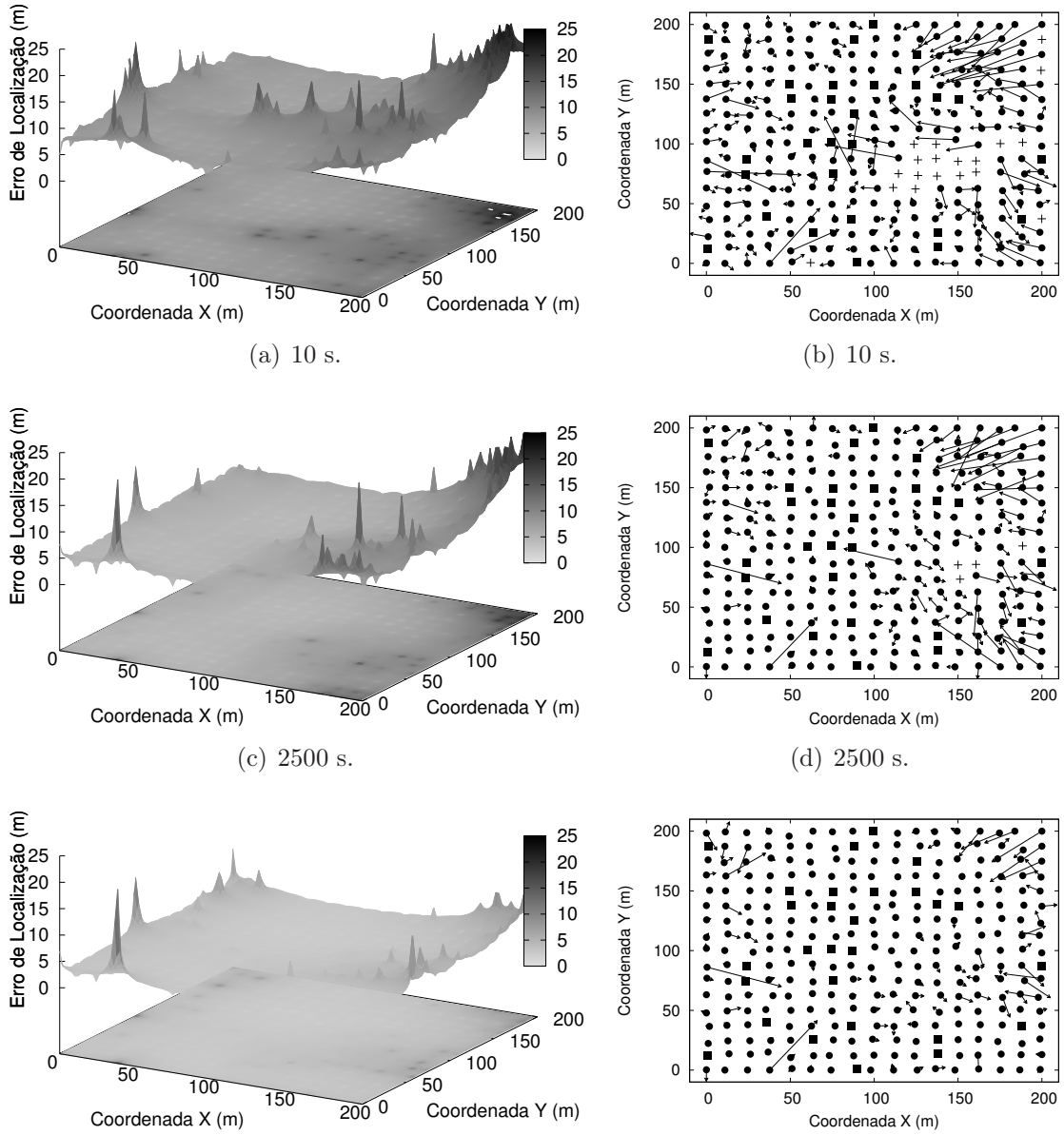
Na figura 6.4 os nós são pré-localizados com RPE, enquanto na figura 6.6 os nós são pré-localizados com DPE. As figuras que marcam 10 s de simulação mostram a localização inicial dos nós, i.e., a localização gerada apenas pelos algoritmos de localização. Nos demais tempos, a localização dos nós é refinada durante o rastreamento. O DPE apresenta erros de localização menores que o RPE, pois a forma como a posição do nó é calculada sofre menos interferência dos ruídos presentes nas estimativas de distância Oliveira et al. [2009]. Os nós mais distantes dos *beacons*, geralmente, possuem erros de localização maiores, devido ao erro acumulado nos nós estabelecidos.

Os nós que estão próximos do alvo tiram proveito da troca de mensagens, necessária para calcular a posição do alvo, para filtrar os ruídos das estimativas de distância e melhorar as suas localizações estimadas. O erro de localização é reduzido gradualmente à medida que o tempo passa, como pode ser observado na figura 6.5 e na tabela 6.3. Quando a rede é pré-localizada com RPE, a solução converge com 7500 s de rastreamento. Com DPE, a solução converge em 5000 s, pois o erro de localização inicial é menor, pois como as referências são melhores, o trabalho de reduzir o erro de localização é facilitado.

Tabela 6.3. O erro médio de localização e quantidade de nós livres são reduzidos durante o rastreamento.

Duração do Rastreamento	RPE		DPE	
	Erro Médio	Nós Livres	Erro Médio	Nós Livres
10 s	9,60 m	19	6,45 m	8
2500 s	7,07 m	4	3,64 m	0
5000 s	3,60 m	0	1,52 m	0
7500 s	2,57 m	0	1,32 m	0
10000 s	2,35 m	0	1,23 m	0

Alguns nós não conseguem se localizar porque o erro das estimativas de distância



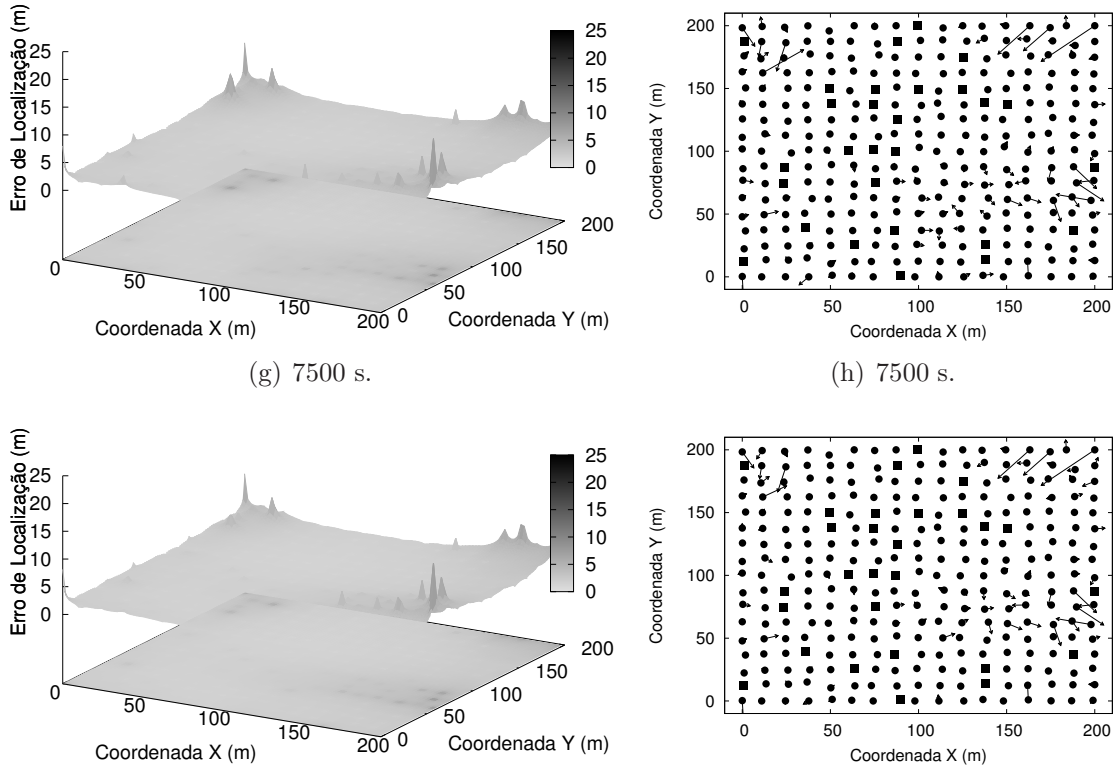
(a) 10 s. (b) 10 s.

(c) 2500 s. (d) 2500 s.

(e) 5000 s. (f) 5000 s.

Figura 6.4. Distribuição dos erros de localização do RPE ao longo do tempo (parte 1).

não possibilitam o cálculo de posição. Além disso, os nós podem não conseguir as referências necessárias para se localizarem, pois os dados sobre suas referências são perdidos em colisões, por exemplo. Aplicando nossa abordagem, os nós livres podem se localizar mesmo depois do algoritmo de localização. Isso ocorre porque as mensagens enviadas durante o rastreamento são usadas para melhorar as estimativas de distância entre os nós, possibilitando que a posição seja calculada. As mensagens também são usadas para adicionar referências aos nós que precisam. No caso do RPE, na localização



(g) 7500 s. (h) 7500 s.

(i) 10000 s. (j) 10000 s.

Figura 6.4. Distribuição dos erros de localização do RPE ao longo do tempo (parte 2).

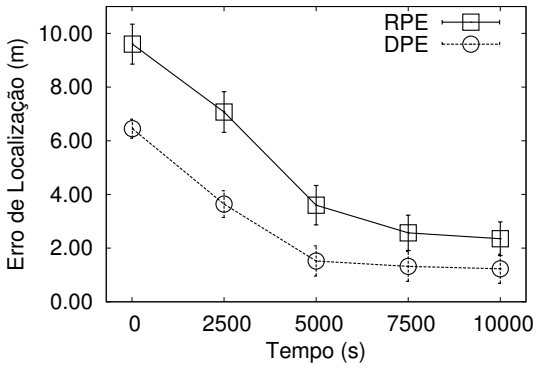
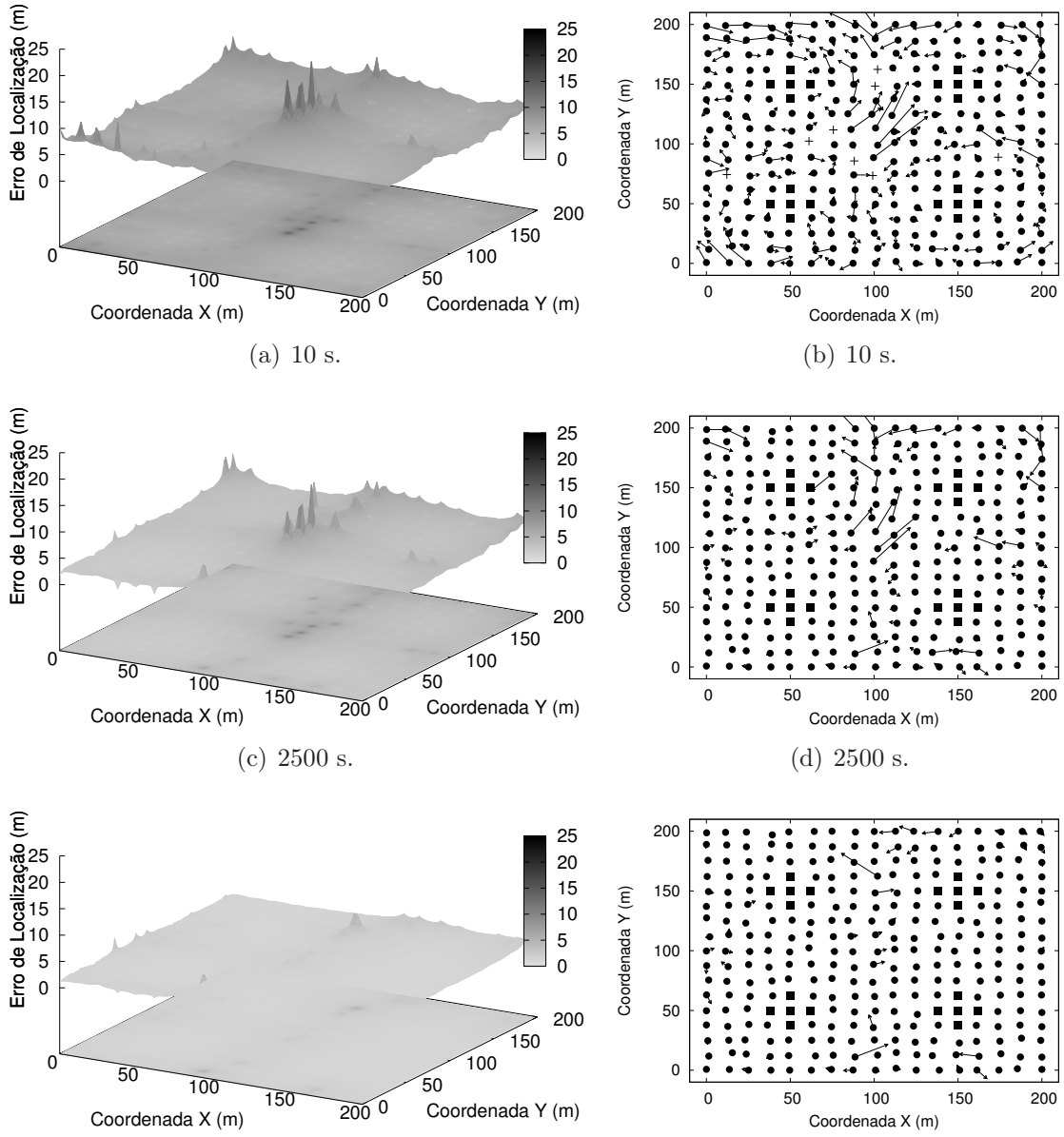


Figura 6.5. O erro médio de localização é reduzido durante o rastreamento.

inicial dos nós, há um grupo de nós que não foram localizados próximo do ponto (140, 70). Em 2500 s de rastreamento a maior parte desses nós são localizados, e em 5000 s de rastreamento todos eles são localizados. No caso do DPE, em 2500 s de rastreamento, todos os nós livres são localizados.

Mesmo com 10000 s de simulação, uma fração dos nós ainda apresenta erro de



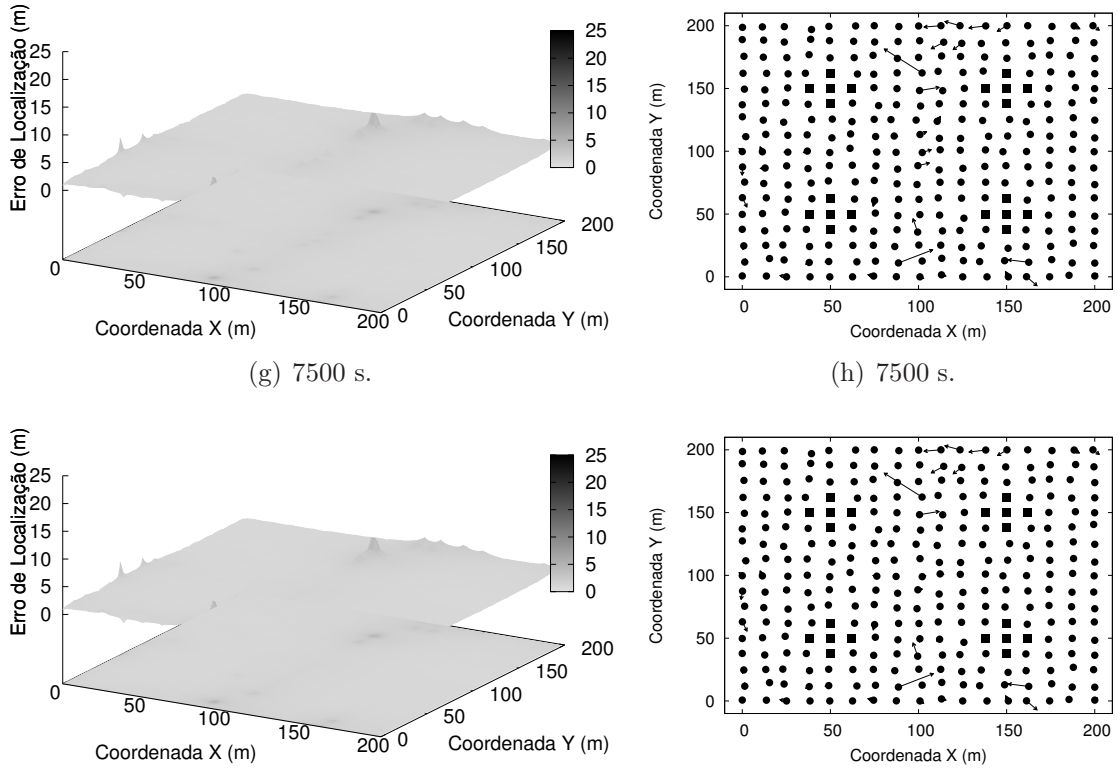
(a) 10 s. (b) 10 s.

(c) 2500 s. (d) 2500 s.

(e) 5000 s. (f) 5000 s.

Figura 6.6. Distribuição dos erros de localização do DPE ao longo do tempo (parte 1).

localização elevado. Esses nós são, geralmente, os mais afastados dos *beacons*, então os erros acumulados em suas referências durante os saltos de comunicação impossibilitam um cálculo de posição mais preciso. Futuramente, pretendemos resolver esse problema aplicando seleção de referências [Xiao, 2011].



(g) 7500 s. (h) 7500 s.

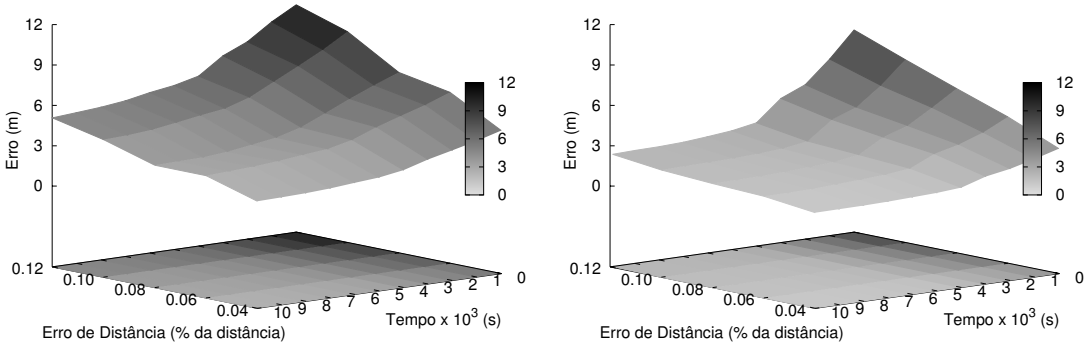
(i) 10000 s. (j) 10000 s.

Figura 6.6. Distribuição dos erros de localização do DPE ao longo do tempo (parte 2).

6.4.2.2 Imprecisão das Estimativas de Distância do Rádio

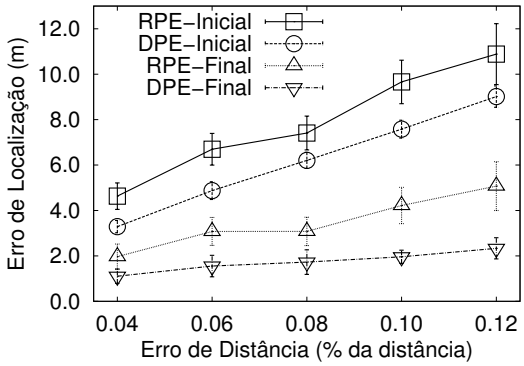
A distância estimada pelos nós sensores não é perfeita. Dependendo do ambiente monitorado, os erros associados são altos, afetando o desempenho da localização e do rastreamento. Em geral, esses erros dependem da distância e podem ser modelados como uma variável Gaussiana de média zero, em que o desvio padrão é um percentual da distância atual [Boukerche et al., 2007]. Desse modo, para avaliarmos diferentes situações, variamos o desvio padrão de 4% a 12% da distância para RPE e DPE.

As figuras 6.7(a) e 6.7(b) mostram como a localização dos nós é afetada pela imprecisão das estimativas de distância e como ela é refinada durante o rastreamento. Enquanto a aplicação de rastreamento é executada, os nós filtram os ruídos presentes nas distâncias estimadas para suas referências e com isso melhoram a localização dos nós. Entretanto, chega um ponto em que a qualidade da localização converge e não apresenta mais melhoras significativas. O RPE possui erros de localização iniciais maiores que o DPE, por isso os ajustes nos erros de localização do RPE precisam de cerca de 7000 s para convergir, enquanto o DPE converge em aproximadamente 5000 s.



(a) RPE – Localização/Tempo.

(b) DPE – Localização/Tempo.



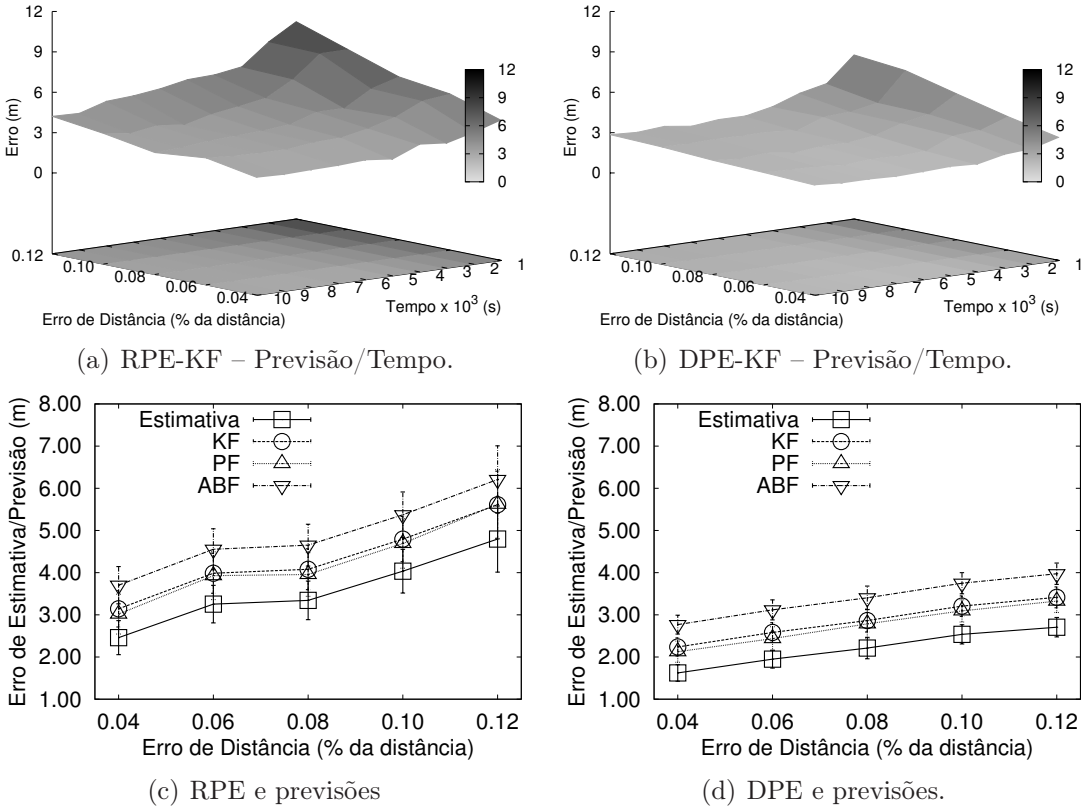
(c) Localizaçao inicial e final

Figura 6.7. Impacto das imprecisões das estimativas de distância na localização.

A figura 6.7(c) compara os erros de localização antes e depois dos ajustes. No pior caso, quando a imprecisão das estimativas de distância é de 12%, o RPE tem seus erros reduzidos em aproximadamente 54% e o DPE em 78%. O DPE é ajustado de forma mais eficiente porque direciona a recursão durante a localização, dessa forma o cálculo de posição dos nós é menos influenciado pelos erros das referências.

A figura 6.8 mostra como o rastreamento é afetado pelo imprecisão das estimativas de distância e como ele é ajustado ao longo do tempo. As figuras 6.8(a) e 6.8(b) mostram o desempenho das previsões feitas com KF. As previsões são afetadas pelo erro das estimativas de posição do alvo, que por sua vez são afetadas pelos erros de localização. Dessa forma, o desempenho das previsões melhora com o passar do tempo porque a localização dos nós usados como referências para calcular a posição do alvo também melhora.

As figuras 6.8(c) e 6.8(d) mostram as estimativas de posição do alvo e também comparam o desempenho das previsões dos métodos KF, PF e ABF. A localização dos nós são usadas como referências para estimar a posição do alvo, logo o erro das estimativas de posição do alvo crescem juntamente a imprecisão das estimativas de



(a) RPE-KF – Previsão/Tempo.

(b) DPE-KF – Previsão/Tempo.

(c) RPE e previsões

(d) DPE e previsões.

Figura 6.8. Impacto das imprecisões das estimativas de distância no rastreamento.

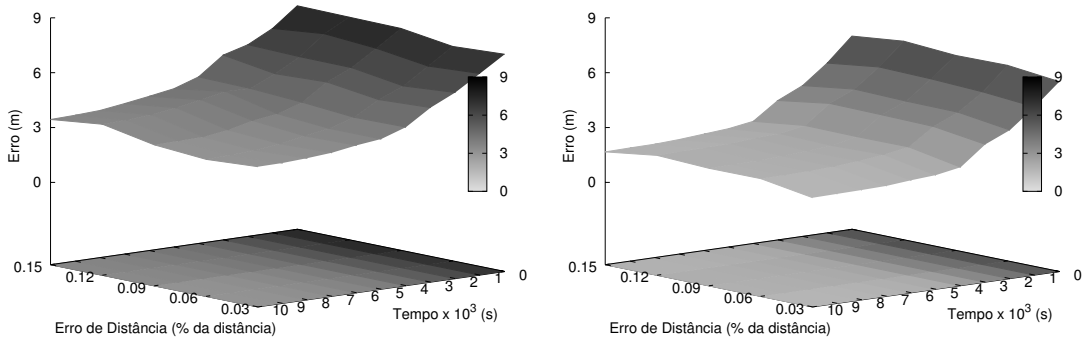
distância. Essas estimativas são usadas como medidas pelos filtros KF, PF e ABF para fazerem previsões, logo a qualidade das previsões refletem a qualidade das estimativas de posição do alvo.

O ABF é uma versão simplificada do KF, ele tem desempenho inferior, porém é menos complexo computacionalmente. O KF é a solução ótima para problemas lineares e com ruídos Gaussianos, entretanto os ruídos introduzidos pelos erros de localização são não-Gaussianos, por isso o PF possui um desempenho melhor. O PF, devido à sua natureza não-linear e não-Gaussiana, reduz uma fração do ruído não-Gaussiano introduzido pelos algoritmos de localização.

6.4.2.3 Imprecisão das Estimativas de Distância do Sensor

Os nós usam as medições dos sensores para estimar as distâncias entre eles e o alvo. Essa distância estimada não é perfeita e seu erro prejudica a posição estimada do alvo. Modelamos esse erro como uma variável Gaussiana de média zero, em que o desvio padrão é um percentual da distância real. Para avaliarmos diferentes situações, variamos o desvio padrão de 6% a 15% da distância.

A figura 6.9 mostra que a imprecisão dos sensores não afeta a localização. Dessa forma, o erro de localização só é influencido pelo tempo de rastreamento, em que os ruídos das estimativas de distância entre os nós são filtrados para melhorar a localização.



(a) RPE – Localização/Tempo. (b) DPE – Localização/Tempo.

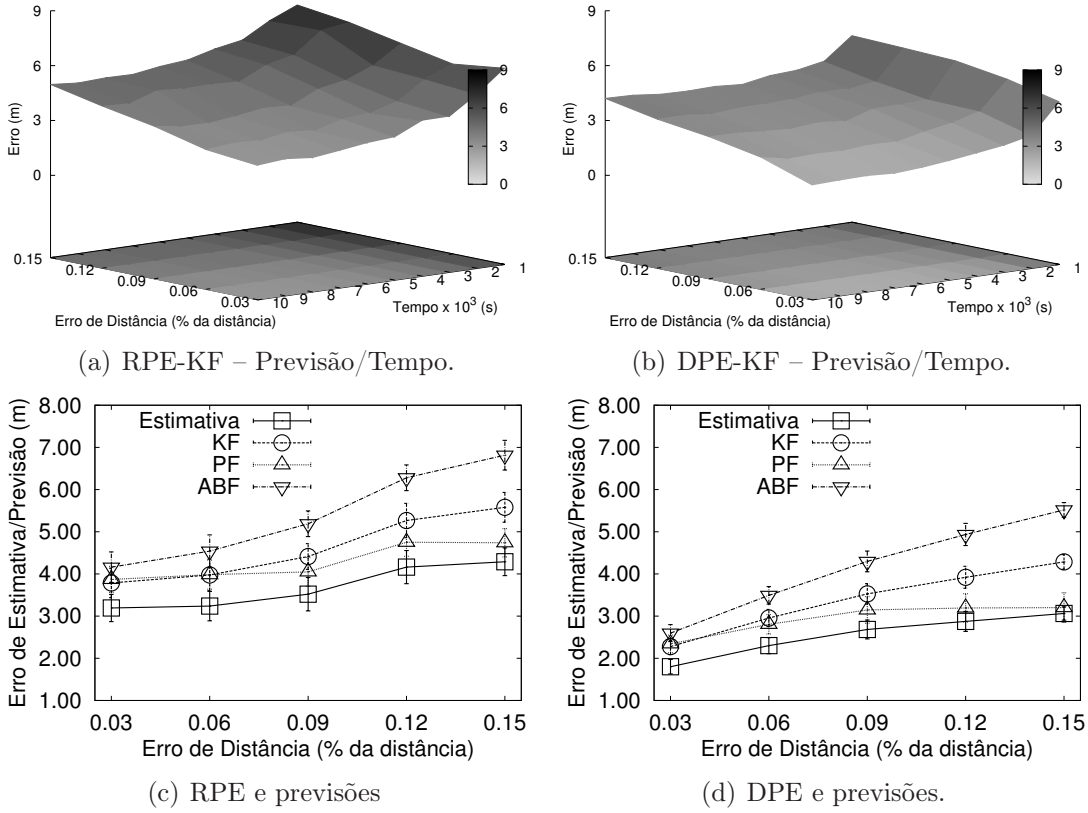
Figura 6.9. Impacto das imprecisões dos sensores na localização.

A figura 6.10 mostra o impacto da imprecisão dos sensores no rastreamento. A imprecisão dos sensores afeta a posição estimada do nó, como essa informação é o parâmetro das previsões, as previsões também são afetadas. As figuras 6.10(a) e 6.10(b) mostram as previsões feitas com KF ao longo do tempo. Ao erro das previsões são somados os erros de localização e o erro da estimativa de posição do alvo. O erros de localização são reduzidos com o tempo, por outro lado o erro da posição estimada do alvo se mantém.

As figuras 6.10(c) e 6.10(d) comparam o desempenho dos métodos de previsão. Na presença de pouco ruído (3%), os métodos apresentam desempenhos similares. Por outro lado, à medida que o ruído aumenta, o PF se sobressai, pois consegue filtrar parte dos ruídos não-Gaussianos, então é menos prejudicado pela imprecisão do sensor.

6.4.2.4 Escalabilidade

Nesta seção, avaliamos como a escala da rede afeta a combinação localização-rastreamento e a redução dos erros ao longo do tempo. Essa análise é importante porque as redes de sensores sem fio preveem situações em que precisam ser implantadas em larga escala para diversas aplicações. Sendo assim, variamos a quantidade de nós de 169 até 841, mantendo constante a densidade de 0,007 $^{\text{nós}}/_{\text{m}^2}$. Portanto, a área monitorada é redimensionada de acordo com a quantidade de nós sensores. Como o percentual de *beacons* usados no RPE é de 10%, o número de *beacons* aumenta de



(a) RPE-KF – Previsão/Tempo.

(b) DPE-KF – Previsão/Tempo.

(c) RPE e previsões

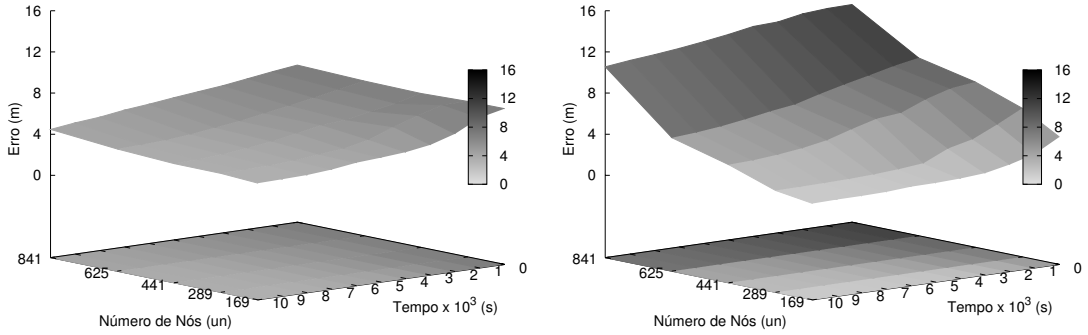
(d) DPE e previsões.

Figura 6.10. Impacto das imprecisões dos sensores no rastreamento.

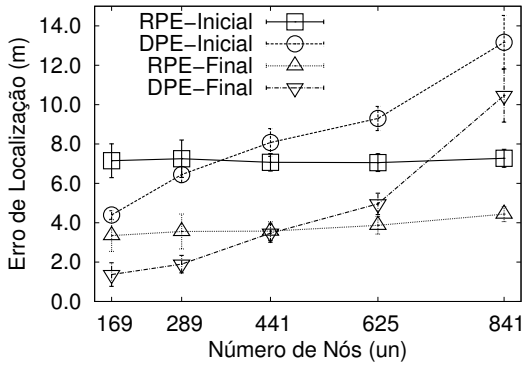
acordo com a quantidade de nós. O DPE mantém o uso de 4 estruturas de *beacons*, e passa a usar 5 estruturas quando a rede possui 441 nós ou mais.

A figura 6.11 mostra o impacto da escala da rede na localização. No RPE, a quantidade de nós *beacons* sempre é 10% da quantidade total de nós, por isso o erro de localização se mantém constante em qualquer escala usada. Já no DPE, a escala aumenta, mas a quantidade de estrutura de *beacons* utilizada não acompanha o crescimento da rede. Dessa forma, mais saltos de comunicação são necessários para alcançar toda a rede e mais erros são acumulados. Com isso, aumentar a escala da rede implica em aumentar o erro de localização. Por outro lado, a qualidade da localização do DPE seria mantida se a quantidade de estruturas de *beacons* utilizadas acompanhasse a escala da rede. Esses dois algoritmos de localização têm seus erros reduzidos ao longo do tempo, isso pode ser visto de forma mais clara na figura 6.11(c).

A figura 6.12 mostra que o rastreamento reflete o mesmo comportamento da localização com a mudança de escala. Isso ocorre porque a localização dos nós são usados como referências para calcular a posição do alvo. No RPE o erro de rastreamento permance o mesmo em qualquer escala, pois a quantidade de *beacons* cresce na mesma proporção que o número de nós. Já com o DPE, no cenário que avaliamos, a quantidade



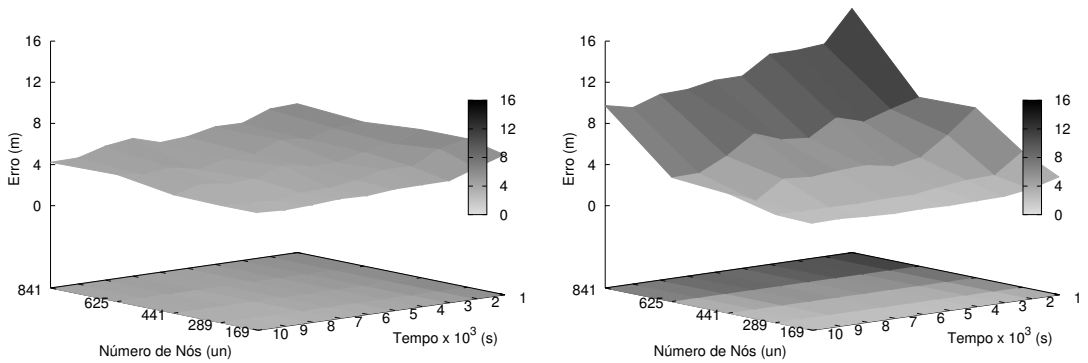
(a) RPE – Localização/Tempo. (b) DPE – Localização/Tempo.



(c) Localização

Figura 6.11. Impacto da escala da rede na localização.

de estruturas de *beacons* utilizadas não foi suficiente para para conservar a qualidade do rastreamento. Em ambos os algoritmos, o erro de localização é reduzido com o tempo, por isso o rastreamento é beneficiado e tem seus erros reduzidos à medida que o tempo passa.

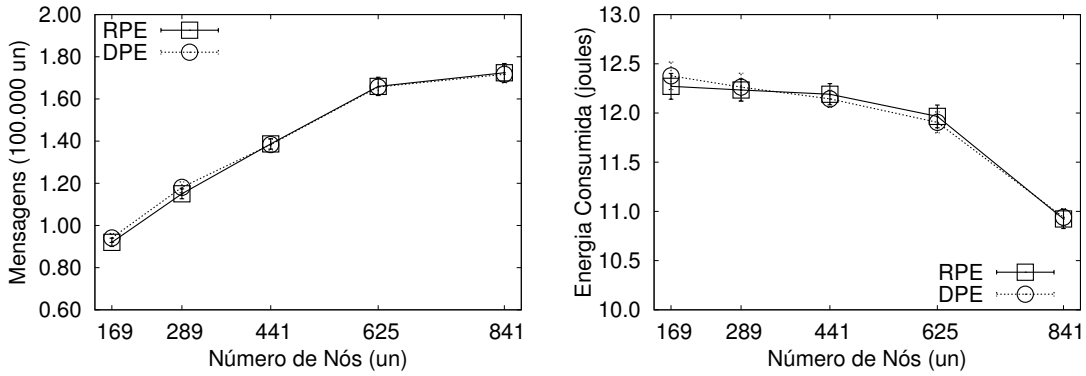


(a) RPE-KF – Previsão/Tempo. (b) DPE-KF – Previsão/Tempo.

Figura 6.12. Impacto da escala da rede no rastreamento.

A figura 6.13 mostra o impacto da escala na quantidade de mensagens enviadas e energia consumida pelos nós. A quantidade de mensagens enviadas cresce com a escala

da rede porque mais mensagens são necessárias para localizar os nós e para reportar o resultado do rastreamento para o nó *sink*. Por outro lado, a energia consumida pela rede diminui com a escala. Isso acontece porque a quantidade de nós que detecta o evento é a mesma em qualquer escala, mas o consumo médio por nó é menor em escalas maiores, uma vez que a carga de trabalho é distribuída entre mais nós.



(a) Mensagens (b) Energia

Figura 6.13. Impacto da escala da rede na energia e mensagens.

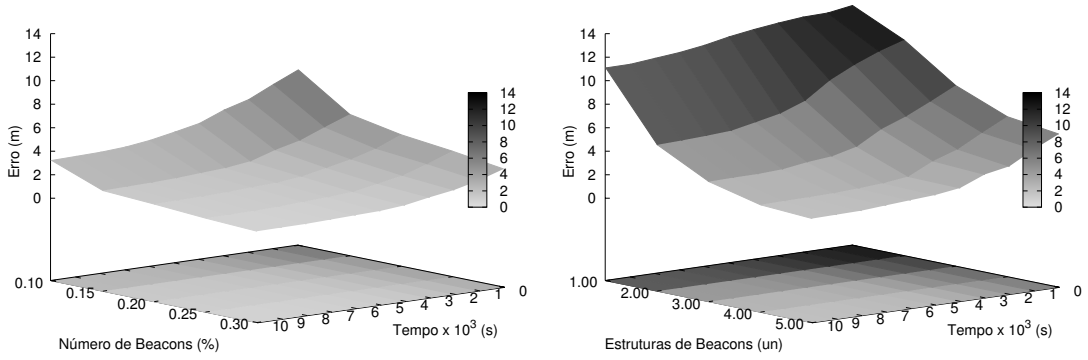
6.4.2.5 *Beacons*

A quantidade e a disposição dos *beacons* usados pelo RPE e DPE levam a diferentes erros de localização. Para o RPE variamos a quantidade de beacons de 10% até 30% da quantidade total de nós. Para o DPE variamos a quantidade de estruturas de beacons de 1 até 5.

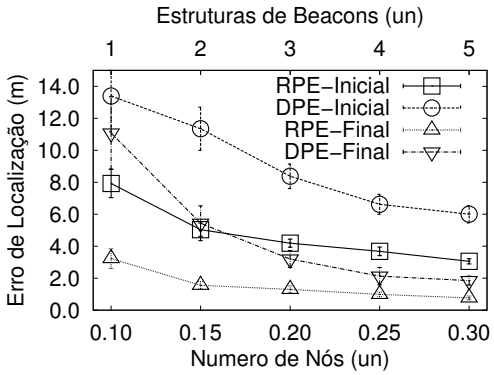
A figura 6.14 mostra como o aumento da quantidade de *beacons* influencia o desempenho da localização. Um número maior de *beacons* faz com que o algoritmo de localização tenha um maior número de referências para estimar a localização dos nós restantes, levando a erros de localização menores. No melhor caso, o DPE usa 5 estruturas de *beacons*, cada estrutura possui 5 beacons, logo apenas 25 beacons são usados. Já no DPE 30% dos nós são beacons, então no melhor caso há 87 *beacons*. Por essa razão o DPE apresenta um resultado menos preciso na localização.

Quando há poucos *beacons* na rede, nossa abordagem tem mais dificuldade de reduzir o erro de localização. Isso acontece porque as referências de localização acumulam mais erros porque há mais saltos de comunicação. Com isso, mesmo filtrando os ruídos das estimativas de distância, os erros de posição das referências persistem. Isso pode ser observado na figura 6.14(c) quando o DPE usa apenas uma estrutura de *beacons*, a diferença entre as localizações inicial e final é de cerca de 2 m, mas quando

utiliza cinco estruturas essa diferença chega a 4 m. Por outro lado, quando há muitos *beacons* (30% com RPE), os erros de localização dos nós já são próximos de zero, então nossa abordagem reduz apenas uma pequena fração desses erros.



(a) RPE – Localização/Tempo. (b) DPE – Localização/Tempo.

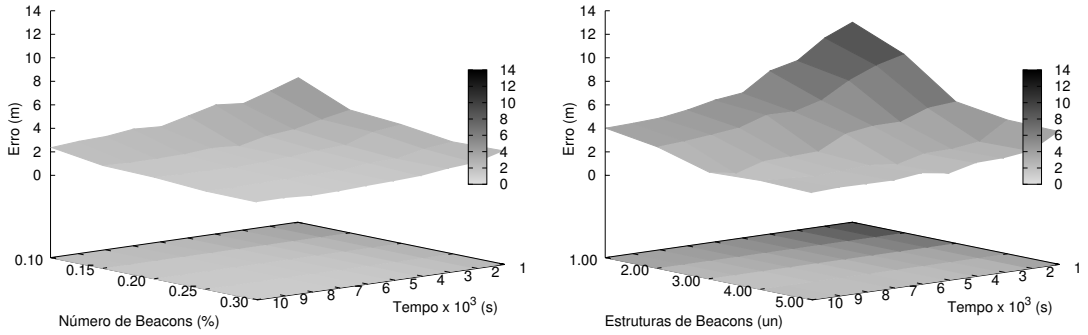


(c) Localização

Figura 6.14. Impacto da quantidade de *beacons* na localização.

A figura 6.15 mostra o desempenho das previsões quando se aumenta a quantidade de *beacons*. O desempenho do rastreamento reflete o desempenho do rastreamento, i.e., aumentar a quantidade de *beacons* reduz o erro de localização e de rastreamento. Além disso, nossa técnica reduz os erros de localização ao longo do tempo, por isso os erros de previsão são reduzidos durante o rastreamento.

A média dos erros de previsão são menores que a média dos erros de localização. Isso ocorre porque são poucos nós que possuem erros de localização acima da média, então o rastreamento utiliza as melhores referências na maioria das vezes. Além disso, os nós com maiores erros de localização geralmente estão nos limites do campo de sensores, por isso são menos utilizados durante o rastreamento.



(a) RPE-KF – Previsão/Tempo. (b) DPE-KF – Previsão/Tempo.

Figura 6.15. Impacto da quantidade de *beacons* no rastreamento.

6.5 Conclusões

Neste capítulo propomos e avaliamos uma abordagem de fusão de dados que reduz os erros de localização durante o rastreamento. Essa abordagem é focada nos algoritmos de localização baseados em alcance, em que os erros são resultado da imprecisão das estimativas de distância entre os nós. Dessa forma, filtramos os erros das estimativas de distância para reduzir o erro de localização. Esse processo não possui custo de comunicação adicional, pois é realizado durante o rastreamento, aproveitando as mensagens enviadas pela aplicação. Essa abordagem também possibilita localizar nós que não puderam ser localizados durante a etapa de localização.

Em nossas avaliações utilizamos dois algoritmos de localização baseados em alcance: RPE e DPE. Nossa abordagem reduz com sucesso o erro desses dois algoritmos de localização. Dependendo do cenário, os erros podem ser reduzidos em mais de 70%. Entretanto, se a localização precisa de muitos saltos de comunicação para ser completada, os erros acumulados nas referências geram erros que são difíceis de serem removidos.

Também comparamos três métodos de previsão em rastreamento de alvos: KF, ABF e PF. Na presença de erros de localização, o PF apresenta um desempenho superior quando comparado com as outras técnicas. Diferente do KF e do ABF, o PF pode ser aplicado para filtrar ruídos não-Gaussianos, por isso é menos influenciado pelo cálculo de posição do alvo estimado com referências que possuem erros de localização.

Capítulo 7

Considerações Finais

7.1 Conclusões

Neste trabalho demonstramos e avaliamos algoritmos para duas formulações diferentes do problema de rastreamento em redes de sensores.

Primeiramente tratamos o rastreamento de alvos em um campo de sensores discretizado em células. A rede de sensores é organizada em grade, de forma que as células formadas são as possíveis áreas de ocupação do alvo (posições). Primeiramente, esse rastreamento foi avaliado em um contexto centralizado, como prova-de-conceito, em que toda a fusão de dados de dados é realizada no nó *sink*, a função dos outros nós da rede é somente detectar o alvo e enviar os dados coletados para o *sink*. A posição do alvo é calculada por uma votação, em que cada nó vota nas quatro células ao seu redor, a célula mais votada é eleita como posição do alvo.

Posteriormente, avaliamos o rastreamento quantizado de forma distribuída usando o PRATIQUE. Esse algoritmo é baseado em um modelo de agrupamento híbrido (estático e dinâmico) para unir todos os dados coletados sobre o evento em único nó, antes de calcular o resultado e enviá-lo ao nó *sink*. As previsões são feitas com KF, PF e ABF, alimentados com as coordenadas das células que representam a posição do alvo.

Analisando as simulações concluímos que atribuir peso constate aos nós parece ser uma boa opção quando o raio de alcance do alvo é igual à distância entre os nós da grade. Entretanto quando consideramos cenários em que ocorrem falhas de comunicação, atribuir pesos com base na distância gera melhores resultados, mesmo nas situações em que a imprecisão das estimativas de distância são altas. Dentre os filtros utilizados na previsão, o KF é o mais eficiente. Entretanto, o ABF apresenta resultados próximas ao KF, apesar de ser menos complexo computacionalmente. Quando o

erro das medidas cresce, o PF tem um desempenho superior ao dos outros filtros. A quantidade de agrupamentos estáticos deve ser ajustado de acordo com as dimensões do campo de sensores para melhor reduzir a quantidade de comunicações. Pois poucos agrupamentos aumentam a quantidade de mensagens para entregar os dados ao CH. Por outro lado, muitos agrupamentos precisam de mais mensagens para configurar os agrupamentos dinâmicos e sincronizar os filtros.

Projetamos e demonstramos o algoritmo TATI que é aplicado na formulação do problema de rastreamento que inclui um objeto guiado que visa alcançar o alvo. Esse reduz a quantidade de nós ativos com base nas informações geográficas das faces e do alvo. Somente são ativados os nós das faces que o alvo pode alcançar até a próxima amostragem. O TATI também reduz a rota entre o objeto guiado e o alvo calculando pontos de consulta, que são os caminhos mais curtos entre o objeto guiado e o nós que possuem as informações sobre a rota.

Nas simulações comparamos o TATI com o DOT e analisando os resultados concluímos que o TATI economiza mais energia que o DOT quando a velocidade do alvo é inferior a $20\,^{\mathrm{m}}/_{\mathrm{s}}$, pois isso possibilita a ativação de uma quantidade menor de faces e ainda garantir que haverá nós para detectar o alvo na próxima amostragem. Essas abordagens tem desempenho equivalente nos casos em que a velocidade do alvo é igual ou superior a $20\,^{\mathrm{m}}/_{\mathrm{s}}$, pois nos dois casos, serão ativadas todas as faces adjacentes do nó mais próximo do alvo. Além disso, a utilização dos pontos do consulta possibilita que o o TATI mantenha o objeto guiado cerca de 10 m mais próximo do alvo que o DOT, pois esses reduzem o percurso a ser seguido.

Por fim, demonstramos uma abordagem que integra localização e rastreamento em que as mensagens enviadas durante o rastreamento são utilizadas para reduzir os erros das estimativas de distância usados para localizar os nós. Para isso, é realizada a fusão de dados de múltiplas distâncias estimadas de uma mesma referência para obter uma única distância estimada mais precisa. Essa abordagem foca nos algoritmos de localização baseados em alcance, sendo assim os algoritmos RPE ou DPE são utilizados para pré-localizar os nós.

Analisando as simulações concluímos que o erro de localização é reduzido ao longo do tempo e precisa de aproximadamente 5000 s para convergir. Esse tempo de conversão depende do algoritmo utilizado para pré-localizar os nós e da trajetória do alvo, pois somente os nós próximos do alvo têm suas localizações corrigidas. A pré-localização com DPE tende a convergir mais rápido, pois possui erros de localização menores. o RPE tem seus erros reduzidos em aproximadamente 54% e o DPE em 78%. O DPE é ajustado de forma mais eficiente porque direciona a recursão durante a localização, dessa forma o cálculo de posição dos nós é menos influenciado pelos erros

das referências. Os nós mais distantes dos *beacons* em alguns casos não têm seus erros de localização reduzidos, pois o erro acumulado durante os múltiplos saltos persiste.

7.2 Trabalhos Futuros

Com base nas atuais limitações presentes nas contribuições deste trabalho, apresentamos a seguir os trabalhos futuros separados por capítulo.

Capítulo 3: (i) implementar esta técnica de rastreamento de forma distribuída; (ii) utilizar fusão de dados para reduzir a quantidade de pacotes enviados pelos nós, agrupando dados em um mesmo pacote; (iii) e realizar a fusão de dados para calcular a célula do alvo de forma iterativa para reduzir a quantidade de dados nos pacotes.

Capítulo 4: (i) revezar a função de CH entre os nós de um agrupamento, dessa forma o consumo de energia é distribuído de maneira mais uniforme; (ii) e acrescentar um componente de ativação para ativar um grupo de nós responsável pelo rastreamento, enquanto o restante permanece desativado para economizar energia.

Capítulo 5: (i) aplicar uma eurística que ajuste a área de interseção do alvo com as faces a medida que o deslocamento do alvo muda; (ii) e elaborar uma técnica que reduz a quantidade de mensagens para ativar as faces, uma vez que o raio de alcance dos nós pode alcançar vários nós de suas faces adjacentes.

Capítulo 6: (i) verificar as inconsistências geométricas da localização dos nós durante a detecção do alvo para reduzir os erros de localização acumulados.

7.3 Publicações

7.3.0.6 Publicados

As publicações listadas abaixo são resultado deste trabalho e de colaboração em outros trabalhos.

- Souza, E. L.; Campos, A. N.; and Nakamura, E. F. (2011). Tracking targets in quantized areas with wireless sensor networks. *Proceedings of the 36th Local Computer Networks (LCN'11)*, pages 235-238. Bonn, Germany.

- Souza, E. L.; Campos, A. N.; and Nakamura, E. F. (2011). Algoritmo Quantizado de Rastreamento em Redes de Sensores Sem Fio. *Anais do 29º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'11)*, pag 833-846. Campo Grande, Rio Grande do Sul.

- Campos, A. N.; Souza, E. L.; Nakamura, E. F.; Nakamura, F. G.; and Rodrigues, J. J. P. C. (2012). On the Impact of Localization and Density Control Algorithms in Target Tracking Applications for Wireless Sensor Networks. *Sensors*, pages 6930-6952.

- Souza, E. L.; and Nakamura, E. F. (2012). PRATIQUE: Um Algoritmo Hierárquico para Rastreamento de Alvos em Áreas Quantizadas para Redes de Sensores. *Anais do 4º Simpósio Brasileiro de Computação Ubíqua e Pervasiva (SBCUP'12)*. Curitiba, Paraná.

- Souza, E. L.; Nakamura, E. F.; Oliveira, H. A. B. F.; and Figueiredo, C. M. S. (2013). Reducing the impact of location errors for target tracking in wireless sensor networks. *Journal of the Brazilian Computer Society (JBCS)*, pages 1-16.

- Souza, E. L.; Pazzi, R. W.; and Nakamura, E. F. (2014). Um Algoritmo de Rastreamento para Interceptação de Alvo em Redes de Sensores Estruturadas em Faces. *Anais do 32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'14)*, Florianópolis, Santa Catarina.

- Souza, E. L.; Pazzi, R. W.; and Nakamura, E. F. (2014). Reduzindo o Erro de Algoritmos de Localização Baseados em Alcance Durante o Rastreamento de Alvos em Redes de Sensores Sem Fio. *Anais do 32º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'14)*, Florianópolis, Santa Catarina.

7.3.0.7 Em Avaliação

Abaixo listamos os trabalhos que estão em avaliação.

- Souza, E. L.; Pazzi, R. W.; and Nakamura, E. F. Target Tracking in Wireless Sensor Networks. *ACM Computing Surveys (CSUR)*.

- Souza, E. L.; Pazzi, R. W.; and Nakamura, E. F. A Prediction-Based Clustering Algorithm for Tracking Targets in Quantized Areas for Wireless Sensor Networks. *Wireless Networks (WINET)*.

- Souza, E. L.; Pazzi, R. W.; and Nakamura, E. F. A Distributed Tracking Algorithm for Target Interception in Face-Structured Sensor Networks. *Local Computers Networks Conference (LCN'14).*

- Souza, E. L.; Pazzi, R. W.; and Nakamura, E. F. Reducing Range-Based Localization Algorithms Errors During Target Tracking. *Local Computers Networks Conference (LCN'14).*

- Souza, E. L.; Pazzi, R. W.; and Nakamura, E. F. Target Tracking in Face Structured Sensor Networks. *IEEE Wireless Communications.*

7.3.0.8 Em Desenvolvimento

Por fim, os trabalhos abaixo estão sendo preparados para submissão.

- Souza, E. L.; and Nakamura, E. F. Quantized Tracking Algorithm in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks (IJDSN).*

- Souza, E. L.; Pazzi, R. W.; and Nakamura, E. F. A Distributed Tracking Algorithm for Target Interception in Face-Structured Sensor Networks. *Elsevier Ad Hoc Networks.*

- Souza, E. L.; Pazzi, R. W.; and Nakamura, E. F. Reducing Range-Based Localization Algorithms Errors During Target Tracking. *Journal of the Brazilian Computer Society (JBCS).*

Referências Bibliográficas

Akyildiz, I. F.; Su, W.; Sankarasubramaniam, Y. & Cayirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422.

Al-Karaki, J. N. & Kamal, A. E. (2004). Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28.

Alawi, R. A. (2011). RSSI based location estimation in wireless sensors networks. Em *Proceedings of the 17th IEEE International Conference on Networks (ICON'11)*, pp. 118–122, Chennai, India.

Alaybeyoglu, A.; Dagdeviren, O.; Erciyes, K. & Kantarci, A. (2009). Performance evaluation of cluster-based target tracking protocols for wireless sensor networks. Em *Proceedings of the 24th International Symposium on Computer and Information Sciences (ISCIS'09)*, pp. 357 –362, Guzelyurt, Northern Cyprus.

Albowicz, J.; Chen, A. & Zhang, L. (2001). Recursive position estimation in sensor networks. Em *Proceedings of the 9th International Conference on Network Protocols (ICNP'01)*, pp. 35–41, Riverside, USA.

Arulampalam, M. S.; Maskell, S.; Gordon, N. & Clapp, T. (2002). A tutorial on Particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Aslam, J.; Butler, Z.; Constantin, F.; Crespi, V.; Cybenko, G. & Rus, D. (2003). Tracking a moving object with a binary sensor network. Em *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*, pp. 150–161, Los Angeles, USA.

Aydogmus, O. & Talu, M. F. (2012). Comparison of extended-kalman- and particle-filter-based sensorless speed control. *IEEE Transactions on Instrumentation and Measurement*, 61(2):402–410.

Bai, F. & Helmy, A. (2004). A survey of mobility models in wireless adhoc networks. Em *Wireless Adhoc and Sensor Networks*, capítulo 1, pp. 1–30. Kluwer Academic Publishers.

Basagni, S.; Carosi, A.; Melachrinoudis, E.; Petrioli, C. & Wang, Z. M. (2008). Controlled sink mobility for prolonging wsns lifetime. *Wireless Networks*, 14(6):831–858.

Bettstetter, C. (2001a). Mobility modeling in wireless networks: categorization, smooth movement, and border effects. Em *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom'01)*, pp. 55–66, Rome, Italy.

Bettstetter, C. (2001b). Smooth is better than sharp: a random mobility model for simulation of wireless networks. Em *Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (MSWIM'01)*, pp. 19–27, Rome, Italy.

Bettstetter, C. & Wagner, C. (2002). The spatial node distribution of the random waypoint mobility model. Em *Proceedings of the German Workshop on Mobile Ad-Hoc Networks (WMAN'02)*, pp. 41–58, Ulm, Germany.

Bhatti, S.; Xu, J. & Memon, M. (2011). Clustering and fault tolerance for target tracking using wireless sensor networks. *IET Wireless Sensor Systems*, 1(2):66–73.

Bhuiyan, M.; jun Wang, G.; Zhang, L. & Peng, Y. (2010). Prediction-based energy-efficient target tracking protocol in wireless sensor networks. *Journal of Central South University of Technology*, 17(2):340–348.

Boukerche, A.; de Oliveira, H. A. B. F.; Nakamura, E. F. & Loureiro, A. A. F. (2007). Localization systems for wireless sensor networks. *IEEE Wireless Communications*, 14(6):6–12.

Bovet, P. & Benhamou, S. (1988). Spatial analysis of animals' movements using a correlated random walk model. *Journal of Theoretical Biology*, 131(4):419–433.

Broch, J.; Maltz, D. A.; Johnson, D. B.; Hu, Y.-C. & Jetcheva, J. (1998). A performance comparison of multi-hop wireless ad hoc network routing protocols. Em *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom '98)*, pp. 85–97, Dallas, USA.

Campos, A. N.; Souza, E. L.; Nakamura, F. G.; Nakamura, E. F. & Rodrigues, J. J. P. C. (2012). On the impact of localization and density control algorithms in target tracking applications for wireless sensor networks. *Sensors*, 12(6):6930–6952.

Campos, C. A. V.; Otero, D. C. & de Moraes, L. F. M. (2004). Realistic individual mobility markovian models for mobile ad hoc networks. Em *Proceedings of the Wireless Communications and Networking Conference (WCNC'04)*, pp. 1980–1985, New Orleans, USA.

Cappe, O.; Godsill, S. J. & Moulines, E. (2007). An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924.

Chang, C. C. G.; Snyder, W. E. & Wang, C. (2007). Secure tracking in sensor networks. Em *Proceedings of the International Conference on Communications (ICC'07)*, pp. 3082–3087, Glasgow, Scotland.

Chang, W.-R.; Lin, H.-T. & Cheng, Z.-Z. (2008). CODA: A continuous object detection and tracking algorithm for wireless ad hoc sensor networks. Em *Proceedings of the 5th Consumer Communications & Networking Conference (CCNC'08)*, pp. 168–174, Las Vegas, Nevada, USA.

Chen, H.; Sezaki, K.; Deng, P. & So, H. C. (2008). An improved dv-hop localization algorithm with reduced node location error for wireless sensor networks. *Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E91-A(8):2232–2236.

Chen, J.; Salim, M. B. & Matsumoto, M. (2011a). A single mobile target tracking in voronoi-based clustered wireless sensor network. *Journal of Information Processing Systems*, 7(1):17–28.

Chen, W.-P.; Hou, J. & Sha, L. (2004). Dynamic clustering for acoustic target tracking in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 3(3):258–271.

Chen, Y.-L.; Wang, N.-C.; Chen, C.-L. & Lin, Y.-C. (2011b). A coverage algorithm to improve the performance of pegasis in wireless sensor networks. Em *Proceedings of the 12th Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'11)*, pp. 123–127, Sydney, Australia.

Chiani, M.; Giorgetti, A.; Mazzotti, M.; Minutolo, R. & Paolini, E. (2009). Target detection metrics and tracking for UWB radar sensor networks. Em *Proceedings of the International Conference on Ultra-Wideband (ICUWB'09)*, pp. 469–474, Vancouver, Canada.

Coates, M. (2004). Distributed Particle filters for sensor networks. Em *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*, pp. 99–107, Berkeley, USA.

Codling, E. A.; Plank, M. J. & Benhamou, S. (2008). Random walk models in biology. *Journal of The Royal Society Interface*, 5(25):813–834.

Cong, L. & Jie, W. (2009). Virtual-force-based geometric routing protocol in MANETs. *IEEE Transactions on Parallel and Distributed Systems*, 20(4):433–445.

Deldar, F. & Yaghmaee, M. (2011). Designing an energy efficient prediction-based algorithm for target tracking in wireless sensor networks. Em *Proceedings of the Conference on Wireless Communications and Signal Processing (WCSP'11)*, pp. 1–6, Nanjing,China.

Deosarkar, B. P.; Yadav, N. S. & Yadav, R. P. (2008). Clusterhead selection in clustering algorithms for wireless sensor networks: A survey. Em *Proceedings of the 17th International Conference on Computing, Communication and Networking (ICCCn'08)*, pp. 1–8, Karur, Tamil Nadu, India.

Dhillon, S. S.; Chakrabarty, K. & Iyengar, S. S. (2002). Sensor placement for grid coverage under imprecise detections. Em *Proceedings of the 5th International Conference on Information Fusion (FUSION'02)*, pp. 1581–1587, Annapollis, Maryland, USA.

Dixiao, W.; Dechao, Z. & Chuanbo, L. (2010). A new algorithm of target locating in binary wireless sensor networks. Em *Proceedings of the International Conference on Intelligent Computing and Intelligent Systems (ICIS'10)*, pp. 753–756, Xiamen, China.

Doucet, A.; Freitas, N. & Gordon, N. (2003). An introduction to sequential Monte Carlo methods. *Royal Statistical Society*, 52(4):694–695.

Doucet, A.; Godsill, S. & Andrieu, C. (2000). On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208.

Doucet, A.; Gordon, N. J. & Krishnamurthy, V. (2001). Particle filters for state estimation of jump Markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624.

Du, X. & Chen, H.-H. (2008). Security in wireless sensor networks. *IEEE Wireless Communications*, 15(4):60–66.

Einstein, A. (1956). Investigations on the theory of the Brownian movement. *Annalen der Physik*, 17(1905):549–560.

Fang, L.; Du, W. & Ning, P. (2007). A beacon-less location discovery scheme for wireless sensor networks. Em *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*, volume 30, pp. 33–55. Springer.

Fernandez, G. R.; Mateos, J. L.; Miramontes, O.; Cocho, G.; Larralde, H. & Ayala-Orozco, B. (2004). Levy walk patterns in the foraging movements of spider monkeys (ateles geoffroyi). *Behavioral Ecology and Sociobiology*, 55(3):223–230.

Figueiredo, C. M. S.; Nakamura, E. F.; Ribas, A. D.; Souza, T. R. B. & Barreto, R. S. (2009). Assessing the communication performance of wsns in rainforests. Em *Proceedings of the 2nd IFIP Wireless Days (WD'09)*, pp. 1–6, Paris, France.

Gillis, J. (1955). Correlated random walk. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(4):639–651.

Gloss, B.; Scharf, M. & Neubauer, D. (2005). A more realistic random direction mobility model. Em *Proceedings of the 4th COST 290 Management Committee Meeting*, Wurzburg, Germany.

Gordon, N.; Salmond, D. & Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing*, 140(6):107–113.

Goswami, S. (2013). Global positioning system. Em *Indoor Location Technologies*, pp. 51–63. Springer.

Group, E. D. C. (2008). Sinalgo - simulator for network algorithms. `http://dcg.ethz.ch/projects/sinalgo/`.

Gui, L.; Wei, A. & Val, T. (2012). A range-free localization protocol for wireless sensor networks. Em *Proceedings of the 9th International Symposium on Wireless Communication Systems (ISWCS'12)*, pp. 496–500, Paris, France.

Guo, D. & Wang, X. (2004). Dynamic sensor collaboration via sequential Monte Carlo. *Selected Areas in Communications*, 22(6):1037–1047.

Guo, Y.; Corke, P.; Poulton, G.; Wark, T.; Bishop-Hurley, G. & Swain, D. (2006). Animal behaviour understanding using wireless sensor networks. Em *Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN'06)*, pp. 607–614, Tampa, Florida, USA.

Gupta, A.; Patil, S. & Zaveri, M. (2012). Lost target recovery in wireless sensor network using tracking. Em *Proceedings of the Conference on Communication Systems and Network Technologies (CSNT'12)*, pp. 352–356, Rajkot, India.

Gupta, P. & Kumar, P. (2000). The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404.

Gustafsson, F. (2010). Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82.

Gustafsson, F. & Gunnarsson, F. (2007). Localization in sensor networks based on log range observations. Em *Proceedings of the 10th International Conference on Information Fusion (Fusion'07)*, pp. 1–8, Québec, Canada.

Gustafsson, F. & Hendeby, G. (2012). Some relations between extended and unscented kalman filters. *IEEE Transactions on Signal Processing*, 60(2):545–555.

Hajiaghajani, F.; Naderan, M.; Pedram, H. & Dehghan, M. (2012). HCMTT: Hybrid clustering for multi-target tracking in wireless sensor networks. Em *Proceedings of the 4th International Conference on Pervasive Computing and Communications Workshops (PERCOM'12)*, pp. 889–894, Lugano, Switzerland.

Hamouda, Y. E. M. & Phillips, C. (2011). Adaptive sampling for energy-efficient collaborative multi-target tracking in wireless sensor networks. *IET Wireless Sensor Systems*, 1(1):15 –25.

Handy, M. J.; Haase, M. & Timmermann, D. (2002). Low energy adaptive clustering hierarchy with deterministic cluster-head selection. Em *Proceedings of the 4th Conference on Mobile and Wireless Communications Networks (MWCN'02)*, pp. 368–372, Stockholm, Sweden.

He, T.; Bisdikian, C.; Kaplan, L.; Wei, W. & Towsley, D. (2010). Multi-target tracking using proximity sensors. Em *Proceedings of the Military Communications Conference (MILCOM'10)*, pp. 1777–1782, San Jose, California, USA.

Hong, S.-W.; Noh, S.-K.; Lee, E.; Park, S. & Kim, S.-H. (2010). Energy-efficient predictive tracking for continuous objects in wireless sensor networks. Em *Proceedings of the 21st International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC'10)*, pp. 1725–1730, Istanbul, Turkey.

Hong, X.; Gerla, M.; Pei, G. & Chiang, C.-C. (1999). A group mobility model for ad hoc wireless networks. Em *Proceedings of the 2nd ACM international workshop on*

Modeling, analysis and simulation of wireless and mobile systems (MSWiM'99), pp. 53–60, Seattle, USA.

Hsu, J.-M.; Chen, C.-C. & Li, C.-C. (2012). POOT: An efficient object tracking strategy based on short-term optimistic predictions for face-structured sensor networks. *Computers & Mathematics with Applications*, 63(2):391–406.

Hsu, T.-H. & Wu, J.-S. (2008). An application-specific duty cycle adjustment MAC protocol for energy conserving over wireless sensor networks. *Computer Communications*, 31(17):4081–4088.

Hu, L. & Evans, D. (2004). Localization for mobile sensor networks. Em *Proceedings of the 10th International Conference on Mobile Computing and Networking (MobiCom'04)*, pp. 45–57, Philadelphia, USA.

Huang, B.; Yu, C. & Anderson, B. D. (2009). Analyzing error propagation in range-based multihop sensor localization. Em *Proceedings of the 48th IEEE Conference on Decision and Control (CDC'09)*, pp. 865–870, Shanghai.

Huang, B.; Yu, C. & Anderson, B. D. O. (2013). Understanding error propagation in multihop sensor network localization. *IEEE Transactions on Industrial Electronics*, 60:5811–5819.

Huang, J.-W.; Hung, C.-M.; Yang, K.-C. & Wang, J.-S. (2011). Probabilistic target coverage in wireless sensor networks. Em *Proceedings of the Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA'11)*, pp. 345–350, Barcelona, Spain.

Huang, Q.; Bhattacharya, S.; Lu, C. & Roman, G.-C. (2005). FAR: Face-Aware Routing for mobicast in large-scale sensor networks. *ACM Transactions on Sensor Networks*, 1(2):240–271.

Humphries, N. E.; Queiroz, N.; Dyer, J. R. M.; Pade, N. G.; Musyl, M. K.; Schaefer, K. M.; Fuller, D. W.; Brunnschweiler, J. M.; Doyle, T. K.; Houghton, J. D. R.; Hays, G. C.; Jones, C. S.; Noble, L. R.; Wearmouth, V. J.; Southall, E. J. & Sims, D. W. (2010). Environmental context explains Lévy and Brownian movement patterns of marine predators. *Nature*, 465(7301):1066–1069.

Intanagonwiwat, C.; Govindan, R.; Estrin, D.; Heidemann, J. & Silva, F. (2003). Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking*, 11(1):2–16.

Iqbal, A. & Murshed, M. (2010). Attack-resistant sensor localization under realistic wireless signal fading. Em *Proceedings of the Wireless Communications and Networking Conference (WCNC'10)*, pp. 1–6, Sydney, Australia.

Isbitiren, G. & Akan, O. B. (2011). Three-dimensional underwater target tracking with acoustic sensor networks. *IEEE Transactions on Vehicular Technology*, 60(8):3897–3906.

Jardosh, A.; Belding-Royer, E. M.; Almeroth, K. C. & Suri, S. (2003). Towards realistic mobility models for mobile ad hoc networks. Em *Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom'03)*, pp. 217–229, San Diego, USA.

Ji, X.; Zha, H.; Metzner, J. J. & Kesidis, G. (2004). Dynamic cluster structure for object detection and tracking in wireless ad-hoc sensor networks. Em *Proceedings of the International Conference on Communications (ICC'04)*, pp. 3807–3811, Paris, France.

Ji, X.; Zhang, Y.-Y.; Hussain, S.; Jin, D.-X.; Lee, E.-M. & Park, M.-S. (2009). FOTP: Face-based Object Tracking Protocol in wireless sensor network. Em *Proceedings of the 4th International Conference on Computer Sciences and Convergence Information Technology (ICCIT'09)*, pp. 128–133, Los Alamitos, USA.

Jia, J.; Chen, J.; Wang, X. & Zhao, L. (2012). Energy-balanced density control to avoid energy hole for wireless sensor networks. *International Journal of Distributed Sensor Networks*, 2012(2012):1–10.

Jian, Z.; Chengdong, W.; Peng, J. & Yue, H. (2011). Collaborative target tracking based on energy consideration in wsns. Em *Proceedings of the 7th Conference on Wireless Communications, Networking and Mobile Computing (WiCOM'11)*, pp. 1–4, Wuhan, China.

Jiang, B. & Ravindran, B. (2011). Completely distributed particle filters for target tracking in sensor networks. Em *Proceedings of the 26th International Parallel Distributed Processing Symposium (IPDPS'11)*, pp. 334–344, Shanghai, China.

Jin, G.-Y.; Lu, X.-Y. & Park, M.-S. (2006). Dynamic clustering for object tracking in wireless sensor networks. Em *Proceedings of the 3rd International Symposium on Ubiquitous Computing Systems (UCS'06)*, pp. 200–209, Seoul, Korea.

Jin, X.; Sarkar, S.; Ray, A.; Gupta, S. & Damarla, T. (2012). Target detection and classification using seismic and pir sensors. *IEEE Sensors Journal*, 12(6):1709–1718.

Johansson, P.; Larsson, T.; Hedman, N.; Mielczarek, B. & Degermark, M. (1999). Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. Em *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom'99)*, pp. 195–206, Seattle, USA.

Julier, S. J. & Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Kapnadak, V. & Coyle, E. J. (2011). Optimal density of sensors for distributed detection in single-hop wireless sensor networks. Em *Proceedings of the 14th International Conference on Information Fusion (FUSION'11)*, pp. 1–8, Chicago, Illinois, USA.

Kareiva, P. M. & Shigesada, N. (1983). Analyzing insect movement as a correlated random walk. *Oecologia*, 56(3):234–238.

Karp, B. & Kung, H. T. (2000). GPSR: greedy perimeter stateless routing for wireless networks. Em *Proceedings of the 6th Conference on Mobile Computing and Networking (MobiCom'00)*, pp. 243–254, Boston, Massachusetts, USA.

Khare, A. & Sivalingam, K. M. (2011). On recovery of lost targets in a cluster-based wireless sensor network. Em *Proceedings of the Pervasive Computing and Communications Workshops (PERCOM'11)*, pp. 208–213, Seattle, USA.

Komagal, E.; Vinodhini, A.; Srinivasan, A. & Ekava, B. (2012). Real time background subtraction techniques for detection of moving objects in video surveillance system. Em *Proceedings of the 1st International Conference on Computing, Communication and Applications (ICCCA'12)*, pp. 1–5, Tamilnadu, India.

Koucheryavy, A. & Salim, A. (2009). Cluster head selection for homogeneous wireless sensor networks. Em *Proceedings of the 11th International Conference on Advanced Communication Technology (ICACT'09)*, pp. 2141–2146, Pyeongchang, Korea.

Kozma, R.; Wang, L.; Iftekharuddin, K.; McCracken, E.; Khan, M.; Islam, K.; Bhurtel, S. R. & Demirer, R. M. (2012). A radar-enabled collaborative sensor network integrating COTS technology for surveillance and tracking. *Sensors*, 12(2):1336–1351.

Kumar, A. & Sivalingam, K. (2012). Target tracking in a wsn with directional sensors using electronic beam steering. Em *Proceedings of the 4th Conference on Communication Systems and Networks (COMSNETS'12)*, pp. 1–10, Bangalore, India.

Kung, H. T. & Vlah, D. (2003). Efficient location tracking using sensor networks. Em *Proceedings of the 57th Wireless Communications and Networking Conference (WCNC'03)*, pp. 1954–1961, New Orleans, USA.

Lalooses, F.; Susanto, H. & Chang, C. H. (2007). An approach for tracking wildlife using wireless sensor networks. Em *Proceedings of the 7th International Conference on New Technologies of Distributed Systems (NOTERE'07)*, pp. 1–7, Morocco, Marrakesh.

Lasassmeh, S. M. & Conrad, J. M. (2010). Time synchronization in wireless sensor networks: A survey. Em *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon'10)*, pp. 242–245, Concord, North Carolina, USA.

LaViola, J. J. (2003). A comparison of Unscented and Extended Kalman filtering for estimating quaternion motion. Em *Proceedings of the 22th American Control Conference (ACC'03)*, pp. 2435–2440, Denver, USA.

Lee, D. & Choi, S. (2011). Multisensor fusion-based object detection and tracking using active shape model. Em *Proceedings of the 6th International Conference on Digital Information Management (ICDIM'11)*, pp. 108–114, Melbourne, Australia.

Lee, I.-S.; Fu, Z.; Yang, W. & Park, M.-S. (2007). An efficient dynamic clustering algorithm for object tracking in wireless sensor networks. Em *Proceedings of the 2nd International Conference on Complex Systems and Applications (ICCSA'07)*, pp. 1484–1488, Jinan, China.

Lee, W.; Yim, Y.; Park, S.; Lee, J.; Park, H. & Kim, S.-H. (2011). A cluster-based continuous object tracking scheme in wireless sensor networks. Em *Proceedings of the Vehicular Technology Conference (VTC'11)*, pp. 1–5, San Francisco, USA.

Levin, S. A. (1986). Random walk models of movement and their implication. Em *Mathematical Ecology: An Introduction*, pp. 143–154. Springer-Verlag.

Li, H. W. & Wang, J. (2012). Particle filter for manoeuvring target tracking via passive radar measurements with glint noise. *IET Radar, Sonar Navigation*, 6(3):2012.

Li, J. & Xu, C. (2010). An improved particle filter algorithm based on target tracking in wireless sensor network. Em *Proceedings of the International Conference on Communications and Intelligence Information Security (ICCIIS'10)*, pp. 187–191, NanNing, China.

Li, J. & Zhou, Y. (2010). Target tracking in wireless sensor networks. Em *Wireless Sensor Networks: Application-Centric Design*, capítulo 19, pp. 1–20. InTech.

Lin, C.-Y.; Peng, W.-C. & Tseng, Y.-C. (2006). Efficient in-network moving object tracking in wireless sensor networks. *Transactions on Mobile Computing*, 5(8):1044–1056.

Lin, C.-Y. & Tseng, Y.-C. (2004). Structures for in-network moving object tracking in wireless sensor networks. Em *Proceedings of the 1st Conference on Broadband Networks (BroadNets'04)*, pp. 718–727, San Jose, California, USA.

Lin, J.; Xiao, W.; Lewis, F. L. & Xie, L. (2009a). Energy-efficient distributed adaptive multisensor scheduling for target tracking in wireless sensor networks. *IEEE Transactions on Instrumentation and Measurement*, 58(6):1886–1896.

Lin, J.; Xie, L. & Xiao, W. (2009b). Target tracking in wireless sensor networks using compressed Kalman filter. *International Journal of Sensor Networks*, 6(3/4):251–262.

Lindsey, S. & Raghavendra, C. (2002). PEGASIS: Power-efficient gathering in sensor information systems. Em *Proceedings of the Aerospace Conference (AERO'02)*, pp. 1125–1130.

Liu, B.-H. (2010). Effective reconstruction of the message-pruning trees in wireless sensor networks. Em *Proceedings of the 4th International Conference on Genetic and Evolutionary Computing (ICGEC'10)*, pp. 695–698, Shenzhen, China.

Liu, B.-H.; Ke, W.-C.; Tsai, C.-H. & Tsai, M.-J. (2008). Constructing a message-pruning tree with minimum cost for tracking moving objects in wireless sensor networks is np-complete and an enhanced data aggregation structure. *IEEE Transactions on Computers*, 57(6):849–863.

Liu, Y.; Li, J. & Guizani, M. (2012). Lightweight secure global time synchronization for wireless sensor networks. Em *Proceedings of the Wireless Communications and Networking Conference (WCNC'12)*, pp. 2312–2317, Paris, France.

Lu, J. & Suda, T. (2003). Coverage-aware self-scheduling in sensor networks. Em *Proceedings of the 18th Workshop on Computer Communications (CCW'03)*, pp. 117–123, Dana Point, California, USA.

Machado, A. B. M.; Martins, C. S. & Drummond, G. M. (2005). *Lista da Fauna Brasileira Ameaçada de Extinção*. Biodiversitas, 1 edição.

Maignan, L. & Gruau, F. (2011). Gabriel graphs in arbitrary metric space and their cellular automaton for many grids. *ACM Transactions on Autonomous and Adaptive Systems*, 6(2):1556–4665.

Manjeshwar, A. & Agrawal, D. P. (2001). TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. Em *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01)*, pp. 2009–2015, Shanghai, China.

Mansouri, M.; Ilham, O.; Snoussi, H. & Richard, C. (2011). Adaptive quantized target tracking in wireless sensor networks. *Wireless Networks*, 17(7):1625–1639.

Mansouri, M. & Khoukhi, L. (2011). Secure quantized target tracking in wireless sensor networks. Em *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference (IWCMC'11)*, pp. 713–718, Istanbul, Turkey.

Marsh, L. & Jones, R. (1988). The form and consequences of random walk movement models. *Journal of Theoretical Biology*, 133(1):113–131.

Merhi, Z.; Elgamel, M. & Bayoumi, M. (2009). A lightweight collaborative fault tolerant target localization system for wireless sensor networks. *IEEE Transactions on Mobile Computing*, 8(12):1690–1704.

Miller, M. & Vaidya, N. (2004). Minimizing energy consumption in sensor networks using a wakeup radio. Em *Proceedings of the Wireless Communications and Networking Conference (WCNC'04)*, pp. 2335–2340, New Orleans, Louisiana, USA.

Musso, C.; Oudjane, N. & Gland, F. L. (2001). Improving regularized Particle filters. Em Doucet, A.; Freitas, N. D. & Gordon, N., editores, *Sequential Monte Carlo Methods in Practice*, capítulo 12, pp. 247–271. Springer.

Naderan, M.; Dehghan, M. & Pedram, H. (2009). Mobile object tracking techniques in wireless sensor networks. Em *Proceedings of the Conference on Ultra Modern Telecommunications Workshops (ICUMT'09)*, pp. 1–8, Moscow, Russia.

Nain, P.; Towsley, D.; Liu, B. & Liu, Z. (2005). Properties of random direction models. Em *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, pp. 1897–1907, Miami, USA.

Nakagawa, M.; Man, D.; Ito, Y. & Nakano, K. (2009). A simple parallel convex hulls algorithm for sorted points and the performance evaluation on the multicore proces-

sors. Em *Proceedings of the 10th Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'09)*, pp. 506–511, Hiroshima, Japan.

Nakamura, E. F.; Loureiro, A. A. F. & Orgambide, A. C. F. (2007). Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Computing Surveys*, 39(3):55. Article 9.

Nakamura, E. F. & Souza, E. L. (2010). Towards a flexible event-detection model for wireless sensor networks. Em *Proceedings of the IEEE Symposium on Computers and Communications (ISCC'10)*, pp. 459–462, Riccione, Italy.

Niculescu, D. & Nath, B. (2003). Dv based positioning in ad hoc networks. *Telecommunication Systems*, 22(1-4):267–280.

Oka, A. & Lampe, L. (2010). Distributed target tracking using signal strength measurements by a wireless sensor network. *IEEE Journal on Selected Areas in Communications*, 28(7):1006–1015.

Olfati-Saber, R. (2007). Distributed Kalman filtering for sensor networks. Em *Proceedings of the 46th Conference on Decision and Control (CDC'07)*, pp. 5492–5498, New Orleans, USA.

Olfati-Saber, R. & Jalalkamali, P. (2011). Collaborative target tracking using distributed kalman filtering on mobile sensor networks. Em *Proceedings of the American Control Conference (ACC'11)*, pp. 1100–1105, San Francisco, California, USA.

Olfati-Saber, R. & Jalalkamali, P. (2012). Coupled distributed estimation and control for mobile sensor networks. *IEEE Transactions on Automatic Control*, PP(99):1–6.

Oliveira, H. A. B. F.; Boukerche, A.; Nakamura, E. F. & Loureiro, A. A. F. (2009). An efficient directed localization recursion protocol for wireless sensor networks. *IEEE Transactions Computing*, 58(5):677–691.

Olule, E.; Wang, G.; Guo, M. & Dong, M. (2007). RARE: An energy-efficient target tracking protocol for wireless sensor networks. Em *Proceedings of the 36th International Conference on Parallel Processing (ICPP'07)*, pp. 76–76, XiAn, China.

Orderud, F. (2005). Comparison of Kalman filter estimation approaches for state space models with nonlinear measurements. Em *Proceedings of the Scandinavian Conference on Simulation and Modeling (SIMS'05)*, Trondheim, Norway.

Park, B.; Park, S.; Lee, E. & Kim, S.-H. (2010). Detection and tracking of continuous objects for flexibility and reliability in sensor networks. Em *Proceedings of the IEEE International Conference on Communications (ICC'10)*, pp. 1–6, Cape Town, South Africa.

Pashazadeh, S. & Sharifi, M. (2008). Simulative study of target tracking accuracy based on time synchronization error in wireless sensor networks. Em *Proceedings of the IEEE Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems (VECIMS'08)*, pp. 68–73, Istanbul, Turkey.

Pearson, K. (1905). The problem of the random walk. *Nature*, 72(1865):294–342.

Pereira, D. P.; Dias, W.; Braga, M.; Barreto, R.; Figueiredo, C. M. S. & Brilhante, V. (2008). Model to integration of RFID into wireless sensor network for tracking and monitoring animals. Em *Proceedings of the 11th Conference on Computational Science and Engineering (CSE'08)*, pp. 125–131, São Paulo, Brazil.

Pitt, M. K. & Shephard, N. (1999). Filtering via simulation: Auxiliary Particle filter. *American Statistical Association*, 94(446):590–599.

Pujolle, G.; Boussetta, K.; Achir, N. & Aitsaadi, N. (2009). Deployment in wireless sensor networks. Em *RFID and Sensor Networks: Architectures, Protocols, Security, and Integrations*, capítulo 17, pp. 477–507. CRC Press.

Rao, B. S. Y.; Durrant-Whyte, H. F. & Sheen, J. A. (1993). A fully decentralized multi-sensor system for tracking and surveillance. *International Journal of Robotics Research*, 12(1):20–44. ISSN 0278-3649.

Rhee, I.; Shin, M.; Hong, S.; Lee, K. & Chong, S. (2008). On the Levy-walk nature of human mobility. Em *Proceeding of the 27th Conference on Computer Communications (INFOCOM'08)*, pp. 924 –932, Phoenix, USA.

Rosencrantz, M.; Gordon, G. & Thrun, S. (2003). Decentralized sensor fusion with distributed Particle filters. Em *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI'03)*, pp. 493–500, Acapulco, Mexico.

Sarma, A. D.; Nanongkai, D.; Pandurangan, G. & Tetali, P. (2010). Efficient distributed random walks with applications. Em *Proceeding of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing (PODC'10)*, pp. 201–210, Zurich, Switzerland.

Schurgers, C. (2008). Wakeup strategies in wireless sensor networks. Em Li, Y.; Thai, M. T. & Wu, W., editores, *Wireless Sensor Networks and Applications*, Signals and Communication Technology, pp. 195–217. Springer.

Schurgers, C.; Tsiatsis, V. & Srivastava, M. B. (2002). STEM: Topology management for energy efficient sensor networks. Em *Proceedings of the IEEE Aerospace Conference, 2002*, pp. 1099–1108, Big Sky, Montana, USA.

Semertzidis, T.; Dimitropoulos, K.; Koutsia, A. & Grammalidis, N. (2010). Video sensor network for real-time traffic monitoring and surveillance. *IET Intelligent Transport Systems*, 4(2):103–112.

Shang, Y. & Shi, H. (2005). Coverage and energy tradeoff in density control on sensor networks. Em *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)*, pp. 564–570, Fukuoka, Japan.

Sharma, S.; Deshpande, S. & Sivalingam, K. (2011a). Alpha-beta filter based target tracking in clustered wireless sensor networks. Em *Proceedings of the 3rd Conference on Communication Systems and Networks (COMSNETS'11)*, pp. 1–4, Bangalore, India.

Sharma, S.; Deshpande, S. & Sivalingam, K. (2011b). On guided navigation in target tracking sensor networks using alpha-beta filters. Em *Proceedings of the 31st Conference on Distributed Computing Systems Workshops (ICDCSW'11)*, pp. 294 –303, Minneapolis, Minnesota, USA.

Sheng, X.; Hu, Y.-H. & Ramanathan, P. (2005). Distributed Particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network. Em *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pp. 181–188, Los Angeles, USA.

Sheu, J.-P.; Chen, P.-C. & Hsu, C.-S. (2008). A distributed localization scheme for wireless sensor networks with improved grid-scan and vector-based refinement. *IEEE Transactions on Mobile Computing*, 7(9):1110–1123.

Shi, L. & Tan, J. (2009). Distributive target tracking in sensor networks with a markov random field model. Em *Proceedings of the Intelligent Robots and Systems (IROS'09)*, pp. 854–859, Saint Louis, Missouri, USA.

Singh, J.; Kumar, R.; Madhow, U.; Suri, S. & Cagley, R. (2011). Multiple-target tracking with binary proximity sensors. *ACM Transactions on Sensor Networks*, 8(1):1–26.

Souza, E. L.; Campos, A. & Nakamura, E. F. (2011a). Tracking targets in quantized areas with wireless sensor networks. Em *Proceedings of the 36th Local Computer Networks (LCN'11)*, pp. 235–238, Bonn, Germany.

Souza, E. L.; Campus, A. N. & Nakamura, E. F. (2011b). Algoritmo quantizado de rastreamento em redes de sensores sem fio. Em *Anais do 29o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'11)*, pp. 833–846, Campo Grande, Rio Grande do Sul.

Souza, E. L.; Nakamura, E. F.; de Oliveira, H. A. B. F. & Figueiredo, C. M. S. (2013). Reducing the impact of location errors for target tracking in wireless sensor networks. *Journal of the Brazilian Computer Society*, 19(1):89–104.

Souza, E. L.; Nakamura, E. F. & Oliveira, H. A. B. F. (2009). On the performance of target tracking algorithms using actual localization systems for wireless sensor networks. Em *Proceedings of the 12th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'09)*, pp. 418–423, Tenerife, Canary Islands, Spain.

Souza, E. L.; Pazzi, R. W. & Nakamura, E. F. (2014a). Reduzindo o erro de algoritmos de localização baseados em alcance durante o rastreamento de alvos em redes de sensores sem fio. Em *Anais do 32o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'14)*, Florianópolis, Santa Catarina.

Souza, E. L.; Pazzi, R. W. & Nakamura, E. F. (2014b). Um algoritmo de rastreamento para interceptação de alvo em redes de sensores estruturadas em faces. Em *Anais do 32o Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC'14)*, Florianópolis, Santa Catarina.

Suganya, S. (2008). A cluster-based approach for collaborative target tracking in wireless sensor networks. Em *Proceedings of the 1st Conference on Emerging Trends in Engineering and Technology (ICETET'08)*, pp. 276–281, Nagpur, India.

Suh, C.; Ko, Y.-B.; Lee, C.-H. & Kim, H.-J. (2006). Numerical analysis of the idle listening problem in ieee 802.15.4 beacon-enable mode. Em *Proceedings of the 1st Conference on Communications and Networking in China (ChinaCom'06)*, pp. 1–5, Beijing, China.

Tan, G. & Kermarrec, A. (2011). Greedy geographic routing in large-scale sensor networks: A minimum network decomposition approach. *IEEE/ACM Transactions on Networking*, 20(3):1–14.

Taylor, C.; Rahimi, A.; Shrobe, H.; Bachrach, J. & Grue, A. (2006). Simultaneous localization, calibration, and tracking in an ad hoc sensor network. Em *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN'06)*, pp. 27–33, Nashville, Tennessee, USA.

Teng, J.; Snoussi, H.; Richard, C. & Zhou, R. (2012). Distributed variational filtering for simultaneous sensor localization and target tracking in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 61(5):2305–2318.

Tian, J.; Haehner, J.; Becker, C.; Stepanov, I. & Rothermel, K. (2002). Graph-based mobility model for mobile ad hoc network simulation. Em *Proceedings of the 35th Annual Simulation Symposium (ANSS'02)*, p. 337, Washington, USA.

Tran, S. P. M. & Yang, T. A. (2006). OCO: Optimized communication and organization for target tracking in wireless sensor networks. Em *Proceedings of the Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC'06)*, pp. 428–435, Taichung, Taiwan.

Tsai, H.-W.; Chu, C.-P. & Chen, T.-S. (2007). Mobile object tracking in wireless sensor networks. *Computer Communications*, 30(8):1811–1825.

Vairo, C.; Amato, G.; Chessa, S. & Valleri, P. (2010). Modeling detection and tracking of complex events in wireless sensor networks. Em *Proceedings of the Conference on Systems Man and Cybernetics (SMC'10)*, pp. 235–242, Istanbul, Turkey.

Vasanthi, V.; Kumar, M. R.; Singh, N. A. & Hemalatha, M. (2011). A detailed study of mobility models in wireless sensor network. *Journal of Theoretical and Applied Information Technology*, 33:7–14.

Vercauteren, T.; Guo, D. & Wang, X. (2005). Joint multiple target tracking and classification in collaborative sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):714–723.

Viswanathan, G. M.; Afanasyev, V.; Buldyrev, S.; Murphy, E.; Prince, P. & Stanley, H. E. (1996). Levy flight search patterns of wandering albatrosses. *Nature*, 381(6581):413–415.

Wahlström, N.; Callmer, J. & Gustafsson, F. (2010). Magnetometers for tracking metallic targets. Em *Proceedings of the 13th Conference on Information Fusion (FUSION'10)*, pp. 1–8, Edinburgh, UK.

Wahlström, N.; Callmer, J. & Gustafsson, F. (2011). Single target tracking using vector magnetometers. Em *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP'11)*, pp. 4332–4335, Prague, Czech Republic.

Walchli, M.; Skoczylas, P.; Meer, M. & Braun, T. (2007). Distributed event localization and tracking with wireless sensors. Em *Proceedings of the 5th international Conference on Wired/Wireless Internet Communications (WWIC'07)*, pp. 247–258, Coimbra, Portugal.

Wan, E. A. & Merwe, R. V. D. (2000). The unscented Kalman filter for nonlinear estimation. Em *Proceedings of the Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC'00)*, pp. 153–158, Lake Louise, Canada.

Wang, C.-L.; Chiou, Y.-S. & Dai, Y.-S. (2007a). An adaptive location estimator based on alpha-beta filtering for wireless sensor networks. Em *Proceedings of the Wireless Communications and Networking Conference (WCNC'07)*, pp. 3285 –3290.

Wang, H.; Zhang, X.; Naït-Abdesselam, F. & Khokhar, A. (2007b). DPS-MAC: an asynchronous MAC protocol for wireless sensor networks. Em *Proceedings of the 14th international conference on High performance computing (HiPC'07)*, pp. 393–404, Goa, India.

Wang, X.; Fu, M. & Zhang, H. (2012). Target tracking in wireless sensor networks based on the combination of KF and MLE using distance measurements. *IEEE Transactions on Mobile Computing*, 11(4):567–576.

Wang, Z.; Bulut, E. & Szymanski, B. K. (2010a). Distributed energy-efficient target tracking with binary sensor networks. *ACM Transactions on Sensor Networks*, 6(4):1–32.

Wang, Z.; Lou, W.; Wang, Z.; Ma, J. & Chen, H. (2010b). A novel mobility management scheme for target tracking in cluster-based sensor networks. Em *Proceedings of the 6th IEEE Conference on Distributed Computing in Sensor Systems (DCOSS'10)*, pp. 172–186, Santa Barbara, California, USA.

Wu, H.; Li, B.-L.; Springer, T. A. & Neill, W. H. (2000). Modelling animal movement as a persistent random walk in two dimensions: expected magnitude of net displacement. *Ecological Modelling*, 132(2):115–124.

Xiao, Q. (2011). *Range-Free and Range-Based Localization of Wireless Sensor Networks*. Tese de doutorado, Hong Kong Polytechnic University.

Xingbo, W.; Huanshui, Z. & Xiangyuan, J. (2011). Collaborative target tracking in WSNs based on maximum likelihood estimation and Kalman filter. Em *Proceedings of the 30th Chinese Control Conference (CCC'11)*, pp. 4946–4951, Yantai, China.

Xu, J.; Li, J. & Xu, S. (2012). Data fusion for target tracking in wireless sensor networks using quantized innovations and kalman filtering. *Science China - Information Sciences*, 3(3):530–544.

Xu, Y. & Lee, W.-C. (2007). Compressing moving object trajectory in wireless sensor networks. *Journal of Distributed Sensor Networks*, 3(2):151–174.

Xu, Y.; Winter, J. & Lee, W. (2004a). Dual prediction-based reporting for object tracking sensor networks. Em *Proceedings of the 1st Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS'04)*, pp. 154–163, Boston, Massachusetts, USA.

Xu, Y.; Winter, J. & Lee, W.-C. (2004b). Prediction-based strategies for energy saving in object tracking sensor networks. Em *Proceedings of the 5th International Conference on Mobile Data Management (MDM'04)*, pp. 346–357, Berkeley, California, USA.

Xue, W.; Luo, Q. & Pung, H. K. (2011). Modeling and detecting events for sensor networks. *Information Fusion*, 12(3):176–186.

Yan, D. M. & Wang, J. K. (2011). Sensor scheduling algorithm target tracking-oriented. *Wireless Sensor Network*, 3(8):295–299.

Yang, H. & Sikdar, B. (2003). A protocol for tracking mobile targets using sensor networks. Em *Proceedings of the 1st Workshop on Sensor Network Protocols and Applications (SNPA'03)*, pp. 71–81, Anchorage, Alaska, USA.

Yang, W.; Fu, Z.; Kim, J. & Park, M.-S. (2007). An adaptive dynamic cluster-based protocol for target tracking in wireless sensor networks. Em Dong, G.; Lin, X.; Wang, W.; Yang, Y. & Yu, J., editores, *Advances in Data and Web Management*, volume 4505 of *Lecture Notes in Computer Science*, pp. 157–167. Springer Berlin / Heidelberg.

Yektaparast, A.; Nabavi, F. H. & Sarmast, A. (2012). An improvement on LEACH protocol (Cell-LEACH). Em *Proceedings of the 14th International Conference on Advanced Communication Technology (ICACT'12)*, pp. 992–996, Pyeongchang, Korea.

Yen, L.-H.; Ye, B. W. & Yang, C.-C. (2010). Tree-based object tracking without mobility statistics in wireless sensor networks. *Wireless Networks*, 16(5):1263–1276.

Yeong-Sung, F.; Cheng-Ta & Hsu, Y.-Y. (2010). An energy-efficient algorithm for object tracking in wireless sensor networks. Em *Proceedings of the Conference on Wireless Communications, Networking and Information Security (WCNIS'10)*, pp. 424–430, Beijing, China.

Yifeng, Z. & Lamont, L. (2011). A map registration localization approach based on mobile beacons for wireless sensor networks. Em *Proceedings of the Global Telecommunications Conference (GLOBECOM'11)*, pp. 1–5, Houston, Texas, USA.

Yik-Chung, W.; Chaudhari, Q. & Serpedin, E. (2011). Clock synchronization of wireless sensor networks. *IEEE Signal Processing Magazine*, 28(1):124–138.

Yoon, J.; Liu, M. & Noble, B. (2003). Random waypoint considered harmful. Em *Proceeding of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM'03)*, pp. 1312–1321, San Francisco, USA.

Yuen, D. & MacDonald, B. A. (2002). A comparison between extended kalman filtering and sequential monte carlo techniques for simultaneous localisation and map-building. Em *Proceedings of the Australasian Conference on Robotics and Automation (ACRA'02)*, pp. 111–116, Auckland, New Zealand.

Yupho, D. & Kabara, J. (2007). The effect of physical topology on wireless sensor network lifetime. *Journal of Networks*, 2(5):14–23.

Zahedi, S.; Srivastava, M. B.; Bisdikian, C. & Kaplan, L. M. (2010). Quality tradeoffs in object tracking with duty-cycled sensor networks. Em *Proceedings of the 31st Real-Time Systems Symposium (RTSS'10)*, pp. 160–169, San Diego, California, USA.

Zhang, H. & Hou, J. C. (2005). Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc & Sensor Wireless Networks*, 1(1):89–124.

Zhang, J.; Yan, T. & Son, S. H. (2006). Deployment strategies for differentiated detection in wireless sensor networks. Em *Proceedings of the 3rd Conference on Sensor and Ad Hoc Communications and Networks (SECON'06)*, pp. 316–325, Reston, Virginia, USA.

Zhang, W. & Cao, G. (2004a). DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks. *IEEE Transactions on Wireless Communications*, 3(5):1689–1701.

Zhang, W. & Cao, G. (2004b). Optimizing tree reconfiguration for mobile target tracking in sensor networks. Em *Proceedings of the 23rd IEEE Computer and Communications Societies (INFOCOM'04)*, pp. 2434–2445, Hong Kong, China.

Zhao, Y.; Yang, Y. & Kyas, M. (2011). Comparing centralized Kalman filter schemes for indoor positioning in wireless sensor network. Em *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, pp. 1–10, Guimarães, Portugal.

Zheng, Q.; Hong, X. & Ray, S. (2004). Recent advances in mobility modeling for mobile ad hoc network research. Em *Proceedings of the 42nd annual Southeast regional conference (ACM-SE'04)*, pp. 70–75, Huntsville, USA.

Zhou, F.; Trajcevski, G.; Avci, B. & Scheuermann, P. (2012). Sensor synchronization for energy efficient multiple object tracking. Em *Proceedings of the 9th IEEE International Conference on Networking, Sensing and Control (ICNSC'12)*, pp. 139–144, Beijing, China.

Zhou, J.; Chen, Y.; Leong, B. & Sundaramoorthy, P. S. (2010). Practical 3d geographic routing for wireless sensor networks. Em *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys'10)*, pp. 337–350, Zurich, Switzerland.

Zhu, Y.; Vikram, A. & Fu, H. (2011). On topology of sensor networks deployed for tracking. Em *Proceedings of the 6th international conference on Wireless algorithms, systems, and applications (WASA'11)*, pp. 60–71, Chengdu, China.

Zou, Y. & Chakrabarty, K. (2004). Sensor deployment and target localization in distributed sensor networks. *ACM Transactions in Embedded Computing Systems*, 3(1):61–91.