

**SPATIAL QUERY PROCESSING FOR WIRELESS
SENSOR NETWORKS**

RONE ILÍDIO DA SILVA

**SPATIAL QUERY PROCESSING FOR WIRELESS
SENSOR NETWORKS**

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: JOSÉ MARCOS S. NOGUEIRA
COORIENTADOR: DANIEL FERNANDES MACEDO

Belo Horizonte
26 de março de 2012

RONE ILÍDIO DA SILVA

**SPATIAL QUERY PROCESSING FOR WIRELESS
SENSOR NETWORKS**

Thesis presented to the Graduate Program
in Ciência da Computação of the Univer-
sidade Federal de Minas Gerais in partial
fulfillment of the requirements for the de-
gree of Doctor in Ciência da Computação.

ADVISOR: JOSÉ MARCOS S. NOGUEIRA
CO-ADVISOR: DANIEL FERNANDES MACEDO

Belo Horizonte
March 26, 2012

© 2012, Rone Ilídio da Silva.
Todos os direitos reservados.

Silva, Rone Ilídio da

S586p Processamento de Requisições Espaciais em Redes de Sensores Sem Fio = Spatial Query Processing for Wireless Sensor Networks / Rone Ilídio da Silva. — Belo Horizonte, 2012
xxxiv, 108 f. : il. ; 29cm

Tese (doutorado) — Universidade Federal de Minas Gerais— Departamento de Ciência da Computação

Orientador: José Marcos S. Nogueira
Coorientador: Daniel Fernandes Macedo

1. Computação - Teses. 2. Redes de sensores sem fio - Teses. I. Orientador. II. Coorientador. III. Título.

CDU 519.6*22(43)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO


FOLHA DE APROVAÇÃO

Processamento de requisições espaciais em redes de sensores sem fio

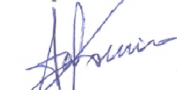
RONE ILÍDIO DA SILVA


Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:


PROF. JOSÉ MARCOS SILVA NOGUEIRA - Orientador
Departamento de Ciência da Computação - UFMG


PROF. DANIEL FERNANDES MACEDO - Co-orientador
Departamento de Ciência da Computação - UFMG


PROF. ALDRI LUIZ DOS SANTOS
Departamento de Informática - UFPR


PROF. ANTONIO ALFREDO FERREIRA LOUREIRO
Departamento de Ciência da Computação - UFMG


PROF. CLODOVEU AUGUSTO DAVIS JÚNIOR
Departamento de Ciência da Computação - UFMG


PROF. YACINE GHAMRI -Doudane
LIGM - University Paris-Est Marne-la-Vallée

To my wife, my parents and my brothers.

Acknowledgments

Antes de qualquer coisa é imprescindível agradecer a Deus. Foi Ele quem colocou cada uma das pessoas abaixo mencionadas no lugar correto e na hora correta. Foi Ele quem forneceu a força necessária para a superação de cada barreira. É Ele que nos envia seus mensageiros carregados de sabedoria para nos auxiliar nas tarefas mais árduas. Abençoe tais mensageiros. Obrigado, meu Deus, por essa fantástica oportunidade de crescimento. Espero fazer jus a todo o conhecimento técnico e pessoal adquirido durante essa caminhada. Pretendo utilizá-lo com o intuito de auxiliar no crescimento de outras pessoas, sem a necessidade de receber bônus por isso. No entanto, também fico feliz em ser um eterno aprendiz. Aprender, sinônimo de evoluir, nos mostra que existe algo superior a ser alcançado, uma existência mais plena, mas perto do Senhor, fonte de felicidade, paz e harmonia.

Rosângela, não existem palavras para descrever o quanto estou agradecido de sua presença na minha vida. Sua doçura foi, é e sempre será para mim, fonte de tranquilidade e motivação nos momentos mais difíceis. Sempre ao meu lado, sempre apoiando mesmo sem receber benefícios diretos, sempre entendendo a distância necessária para o cumprimento de minhas obrigações. Você é a luz divina que foi lançada para me fortalecer e me tornar feliz. Que você seja iluminada sempre e que esta doçura de menina perdure. Amo você com uma intensidade na qual as palavras não são capazes de expressar.

Waldyr e Hilda, meu pai e minha mãe amados, tudo que faço na vida é uma continuação da obra de vocês. Tenho a felicidade de tê-los sempre ao meu lado, mesmo que as vezes a distância, mas em oração. Suas preces foram atendidas e os conselhos se mostraram reais: paciência, perseverança e fé. Denise, minha querida irmã e amiga, como sua presença é confortante, como suas orações foram valiosas. Deus lhe dará em paz toda força que você me transmitiu durante todo esse tempo. Sandro, obrigado pelo exemplo de força de vontade. Você sempre foi para mim um ícone de inteligência.

Ao demais, gostaria de me desculpar caso esqueça alguém. Matilde e Gino, não caberia nesta página o quanto vocês fizeram por mim e pela Rosângela. Recebam em

dobro do Senhor tudo o bem que praticaram. Alexandre, obrigado pela presença sempre alegre e confortante. Marisa, obrigado pela força. Pedro, Lucas, Rafael, André, Mateus, Helena, Gustavo, Guilherme e Daniel somente crianças são capazes de passar alegria como vocês me passaram. Dona Luzia, obrigado pelas orações e pelas palavras de ânimo. Obrigado também à família Canaan, sempre presente. Obrigado aos amigos da Academia Santana, em especial ao amigo Sensei Vicente, o qual é exemplo de determinação. Obrigado aos amigos do Laboratório Winet, sempre presentes em todos os momentos. Obrigado aos professores e funcionários do Departamento de Computação da UFMG pela ajuda imprescindível.

Finalizando, gostaria de agradecer a meu orientador José Marcos e a meu co-orientador Daniel, pessoas sensacionais que tive o prazer de trabalhar. Vocês serão sempre para mim modelos de profissionais e pessoas. Deus, em sua infinita sabedoria, coloca entre nós pessoas como vocês para nos mostrar o caminho da evolução. Continuem transmitindo aos demais tudo o conhecimento que passaram para mim.

Resumo

Redes de sensores sem fio (RSSF) são tipicamente utilizadas para realizar coleta de dados em regiões geográficas bem definidas. As requisições que utilizam atributos geográficos para definir a região de coleta são denominadas requisições espaciais. Tais requisições são definidas em linguagens derivadas do SQL, as quais expressam a região de coleta de dados (chamada região de interesse) por meio de atributos geográficos, como pontos e polígonos. Esse tipo de requisição economiza energia, uma vez que restringe a coleta de dados somente à região de interesse. Este trabalho propõe uma discussão geral sobre processamento de requisições espaciais em RSSF e um novo mecanismo, eficiente em relação ao consumo de energia, para realizar esse tipo de processamento. Na discussão sobre o tema, o processamento de requisições espaciais é dividido em estágios, o que facilita seu entendimento e a proposta de novas soluções. Verifica-se que os trabalhos relacionados consideram regiões de interesse retangulares ou circulares. Até o presente momento, não foram encontrados trabalhos que considerem regiões de interesse em forma de polígonos, os quais podem representar objetos reais plotados sobre mapas ou imagens de satélite. Além disso, tais trabalhos geralmente consideram nós sempre ativos. Entretanto, eles podem falhar por diversos motivos ou adormecer para economizar energia. Isso muda a topologia da rede, o que também aumenta o consumo de energia uma vez que os nós devem enviar mensagens de controle no intuito de atualizar as tabelas de rota de seus vizinhos. O mecanismo aqui proposto foi desenvolvido para processar requisições que possuem regiões de interesse em forma de polígono, RSSF com nós propícios a falhas e nós que adormecem para economizar energia. Os experimentos mostram que, quando comparado ao estado da arte, o mecanismo proposto aumenta consideravelmente o número de requisições que obtêm resposta e consome cerca de quatro vezes menos energia.

Palavras-chave: Requisições Espaciais, Redes de Sensores Sem Fio, Tolerância a Falhas.

Abstract

Wireless sensor networks (WSN) are particularly useful for obtaining data concerning events limited to a well-defined geographic region. Applications for this task typically use spatial queries, which are SQL-like queries where location constraints are imposed on the collected data. Spatial queries save energy since only nodes inside this region collect data. This work provides a general discussion of in-network spatial query processing in WSN. We propose to divide spatial query processing in to well-defined stages. This division helps understanding the spatial query processing mechanisms found in the literature and makes it easier to propose new solutions. This survey shows that the state of the art of spatial query processing considers regions of interest shaped as rectangles or circles. To the best of the authors' knowledge, no related work considers regions of interest shaped as of polygons, which can represent real objects plotted on maps or satellite images. A polygon can be represented in the query by a sequence of several points, which can occupy several network packets during the query transmission, increasing the energy consumption. Furthermore, the related works, in general, consider that nodes are always powered on. However, nodes can go to sleep mode (turn off the radio in duty cycles) in order to save energy or can fail during their operation. This changes the network topology and, consequently, increases the amount of control message to upgrade routing tables. This work proposes a new energy-efficient fault-tolerant in-network spatial query processing mechanism that assumes polygonal regions of interest, fail-prone nodes and nodes employing duty cycles. Exhaustive experiments show that the proposed mechanism processes more queries than the state of the art and consumes four times less energy.

Keywords: Spatial Query, Wireless Sensor Network, Window Query, Duty Cycle, Fault Tolerance.

Resumo Estendido

Originalmente o texto desta tese foi redigido em Inglês com o título “Spatial Query Processing for Wireless Sensor Networks”. Entretanto, com o intuito de facilitar o acesso aos leitores de língua portuguesa, e para atender às normas da Universidade Federal de Minas Gerais, apresento aqui um resumo em Português de cada um dos capítulos contidos na tese.

Capítulo 1 - Introdução

Redes de Sensores Sem Fio (RSSF) são redes formadas por inúmeros componentes minúsculos denominados nós sensores. Elas são normalmente utilizadas para realizar monitoramento de regiões onde o acesso humano é difícil ou impossível. Tradicionalmente, nós sensores possuem recursos computacionais bastante limitados, um conjunto de sensores e se comunicam entre si através de rádio frequência. Nesse tipo de rede, a maioria das comunicações ocorre entre um nó denominado *sink* (também conhecido como sorvedouro ou estação base) e os demais nós da rede. Tal nó pode ser idêntico aos demais, mas realizando uma função diferente, ou ser um dispositivo sem as limitações de recursos que geralmente são encontradas nos nós sensores descritos na literatura. Dentre tais limitações recursos, a energia consumida pelo nó se apresenta como a mais importante. Como os nós sensores são alimentados por baterias, o projeto de aplicações para RSSF deve considerar o consumo de energia no intuito de aumentar ao máximo possível o tempo de vida da RSSF [Akyildiz et al., 2002].

Pesquisas recentes mostram que RSSF deixaram de ser apenas ferramentas que realizam coletas periódicas de dados. Em várias situações, tais redes são utilizadas para monitorar regiões geográficas onde os dados a serem coletados não são conhecidos *a priori*. Por isso, elas se tornaram mecanismos que respondem a requisições variadas criadas pelos usuários. Bons exemplos são o TinyDB [Madden et al., 2005] e o Cougar [Demers et al., 2003], plataformas para RSSF que permitem a criação de requisições em uma linguagem semelhante ao SQL. Com tais mecanismos, o usuário é capaz de definir

quais são os dados de interesse e, se necessário, os intervalos entre coletas periódicas de dados. A principal vantagem desse tipo de mecanismo é que todo esse procedimento é realizado sem a necessidade de criação e instalação de uma nova aplicação para definir quais dados são de interesse e a periodicidade das coletas.

Tais mecanismos, entretanto, não suportam requisições com restrições espaciais, nas quais o usuário define a região onde os dados devem ser coletados. Isso consome energia de nós sensores que estão fora da região de onde se deseja realizar a coleta, uma vez que as requisições são enviadas para todos os nós da rede [Deshpande et al., 2007]. Como a energia é um dos fatores mais importantes para RSSF, esse consumo desnecessário deve ser evitado ou minimizado. Uma forma para tratar tal problema é a criação de mecanismos para RSSF que suportem consultas com restrições espaciais. Tais ferramentas encaminham as consultas para os nós que estão dentro da região definida pelo usuário e limitam a coleta de dados somente aos nós contidos nela.

As requisições que restringem os dados a serem coletados por meio de atributos espaciais são denominadas **requisições espaciais** [Manalopoulos and Papadopoulos, 2004] e são tipicamente criadas em bancos de dados geográficos, como o PostGIS e o Oracle Spatial [Casanova et al., 2005]. Esse tipo de requisição se difere das demais em dois principais pontos. Primeiro, elas incorporam tipos espaciais de dados, tais como pontos e polígonos. Segundo, elas consideram o relacionamento espacial entre os dados, como um ponto *dentro* de um polígono ou um polígono que *sobrepo* outro. RSSF podem ser modeladas como bancos de dados geográficos com o intuito de capacitá-las a responder requisições espaciais.

Na literatura, podem ser encontrados trabalhos que definem mecanismos para processamento de requisições espaciais em RSSF. Entretanto, existem várias questões em aberto para serem tratadas em pesquisas futuras. Uma das questões que merece destaque é o formato das regiões onde os dados serão coletados, denominada **região de interesse**. Nesses trabalhos, tais regiões possuem formato retangular ou circular. Até o presente momento, nenhum trabalho encontrado na literatura tratou o processamento de requisições espaciais cuja região de interesse tenha formato irregular, como por exemplo um polígono. Esse tipo de figura pode representar objetos reais plotados sobre mapas ou fotos de satélites e expressar de melhor forma a região onde o usuário deseja que a coleta de dados seja realizada. Entretanto, a representação de um polígono pode ser composta por uma sequência muito grande de pontos. Com isso, uma requisição que possua a representação desse polígono utilizaria vários pacotes da RSSF para ser transmitida, o que consumiria considerável montante de energia dos nós sensores para o seu processamento.

Outro ponto que merece destaque nos trabalhos sobre processamento de requisi-

ções espaciais em RSSF é que tais trabalhos consideram os nós sensores sempre ligados, podendo assim enviar e receber mensagens a qualquer momento. Entretanto, essa afirmação não é verdadeira em todas as situações. A maioria dos protocolos da camada MAC para RSSF possui algoritmos que periodicamente desligam o rádio dos nós sensores no intuito de economizar energia. Além disso, devido aos lugares inóspitos onde as redes de sensores são lançadas, os nós sensores são propícios a falhas. Verifica-se então que o conhecimento dos nós sensores sobre a topologia da rede muda durante a operação da RSSF. Portanto, os nós sensores precisam periodicamente enviar mensagens de controle para atualizar informações sobre a topologia da rede, o que pode consumir muita energia.

Os objetivos deste trabalho são investigar o processamento de requisições espaciais em RSSF e propor um novo mecanismo para processamento desse tipo de requisição, com o menor consumo de energia possível. Na literatura, tal processamento é realizado por mecanismos compostos por conjuntos de algoritmos que trabalham de forma complementar. Por isso, pretende-se definir uma metodologia que facilite o entendimento desses mecanismos e os classifique de forma clara. Além disso, o mecanismo proposto deve considerar RSSF com nós que podem estar em estado de dormência e nós propícios a falhas. Tais características são encontradas em redes de sensores reais e não são consideradas na maioria dos trabalhos relacionados.

Este trabalho propõe a divisão em estágios dos mecanismos para processamento de requisições espaciais em RSSF. Cada estágio representa um subproblema a ser resolvido pelos mecanismos. Por isso, os algoritmos que solucionam os problemas de cada um dos estágios são analisados separadamente. Ou seja, todos os algoritmos que fazem parte dos mecanismos encontrados na literatura e que foram utilizados para a solução de um determinado estágio são descritos e comparados entre si. Essa divisão facilita o entendimento dos mecanismos e simplifica a proposta de novas soluções. A partir desses estudos, são apresentados novos algoritmos para os estágios. Tais algoritmos formam um mecanismo que têm o intuito de processar requisições espaciais consumindo a menor quantidade de energia possível. Além disso, o mecanismo proposto considera que vários nós da rede podem estar dormindo durante o processamento de requisições e que tais nós ainda podem falhar.

Capítulo 2 - Mecanismos para Processamento de Requisições Espaciais

Neste capítulo foram analisados os mais relevantes trabalhos sobre processamento de requisições espaciais em RSSF encontrados na literatura. Para isso, o proces-

samento de requisições espaciais foi dividido em seis estágios: Pré-Processamento, Encaminhamento, Disseminação, Agregação, Sensoriamento e Retorno. O **Pré-Processamento** é realizado no computador do usuário. Nele, prepara-se a requisição antes que ela seja enviada para RSSF. O primeiro nó sensor que recebe uma requisição é denominado **Origem**. Durante o Estágio de Encaminhamento, a requisição é propagada do nó Origem até um nó dentro da região de interesse, denominado **Coordenador**. No estágio de **Disseminação**, a requisição é transmitida do Coordenador para todos os nós contidos dentro da região de interesse. Tais nós coletam informações do ambiente no estágio de **Sensoriamento**. O estágio de **Agregação** ocorre durante o percurso da informação a partir de sua origem até um nó denominado **Agregador**, o qual calcula o resultado final do processamento da requisição. Nesse estágio, cada nó recebe as informações de seus vizinhos e a de seus sensores, aplica a função de agregação estabelecida na requisição e envia o resultado. Finalmente, no estágio de **Retorno**, o Agregador retorna o resultado da requisição para o nó Origem.

Os mecanismos para processamento de requisições espaciais encontrados na literatura são classificados de acordo com a **Definição do Problema** tratado por eles e com a forma que eles realizaram a **Resolução do Problema**. A Definição do Problema classifica os mecanismos a partir das características da RSSF considerada e do tipo de requisição processada. A Resolução do Problema classifica cada mecanismo em relação a seu funcionamento. Para isso, utiliza como critérios de classificação as estruturas criadas por cada mecanismo durante o processamento de requisições e os algoritmos utilizados por cada um deles em seus estágios.

Verifica-se que os mecanismos descritos pelos trabalhos analisados somente consideram regiões de interesse em formato retangular ou circular. Essas formas geométricas facilitam o processamento de requisições, pois necessitam de apenas dois pontos para serem representados na requisição. Tais trabalhos utilizam uma estratégia gulosa ou o protocolo GPSR [Karp and Kung, 2000] nos estágios Encaminhamento e Retorno. Esses protocolos de roteamento criam e mantêm uma tabela de vizinhos em cada nó, a qual é utilizada para definir o próximo nó durante o repasse de dados. Entretanto, em RSSF com ciclos de dormência e nós propícios a falhas, o consumo de energia para manter tais tabelas atualizadas tende a ser alto. No estágio de Disseminação, os mecanismos estudados utilizam, principalmente, Inundação Restrita e protocolos baseados na criação de itinerários. Esses últimos se mostram mais eficientes em relação ao consumo de energia, uma vez que menos nós transmitem a requisição dentro da região de interesse.

Foram também analisados mecanismos que criam índices espaciais distribuídos com o intuito de facilitar o processamento de requisições. Tais mecanismos têm a

limitação de iniciar o processamento somente a partir da estação base. Além disso, as mudanças de topologia causadas por falhas de nós ou por nós em estágio de dormência podem levar à transmissão de um grande número de mensagens de controle, aumentando consideravelmente o consumo de energia. Isso ocorre pois as informações do índice salvas por um determinado nó sensor sofrem alterações com a mudança de topologia em qualquer um dos descendentes diretos ou indiretos desse nó. Ou seja, uma pequena alteração de topologia pode gerar várias mensagens de atualização de índice.

Capítulo 3 - Reduzindo o Número de Pacotes Ocupados por uma Requisição

Esse capítulo apresenta um algoritmo para o estágio de Pré-Processamento. Tal algoritmo foi desenvolvido para diminuir o número de pacotes utilizados para transmitir uma requisição. Este trabalho considera regiões de interesse em forma de polígonos, os quais podem ser formados por um número grande de pontos. O algoritmo proposto simplifica a forma de tais polígonos. Isso reduz o número de pacotes que uma requisição ocupa ao ser transmitida. Como cada requisição é transmitida várias vezes durante seu processamento, ao se reduzir o tamanho da requisição o consumo de energia tende a ser menor. O algoritmo proposto é baseado no conhecido algoritmo de *Douglas-Peucker* [Douglas and Peucker, 1973], com as devidas alterações para se adequar ao processamento de requisições espaciais.

Capítulo 4 - O Protocolo ABF

Este capítulo descreve um novo protocolo de roteamento geométrico denominado ABF (*Ask Before Forwarding*) para ser utilizado nos estágios de Encaminhamento e Retorno. Tal protocolo foi concebido para RSSF que possuem nós propícios a falhas ou nós que utilizam algoritmos de *duty cycle* para economizar energia. Nesse tipo de rede, protocolos que mantêm tabelas de vizinhos tendem a consumir muita energia. Isso ocorre pois os nós devem periodicamente transmitir pacotes de controle para que seus vizinhos mantenham suas tabelas de rota atualizadas. Entretanto, as constantes mudanças de topologia ocasionadas pela dormência ou falhas de nós tendem a aumentar o número de mensagens de controle, conseqüentemente aumentando o consumo de energia da rede.

ABF assume que os nós não possuem informações sobre a topologia da rede. Por isso, quando um determinado nó S possui um pacote para ser encaminhado (uma

requisição ou seu resultado) ele primeiro verifica qual vizinho é capaz de retransmitir o pacote e depois o envia. Essa verificação é realizada com a transmissão de um **pacote de solicitação** de vizinhos. Quando um nó R recebe um pacote de solicitação, ele envia para S um **pacote de anúncio** que informa a S sua disponibilidade para recebimento. R atrasa o envio do pacote de anúncio por um período de tempo proporcional à sua distância até o destino. Isso faz com que os nós mais próximos do destino tenham maior probabilidade de serem escolhidos para participar de rotas, pois ao receber o primeiro desses pacotes, S escolhe o emissor como o próximo nó na rota e transmite para esse nó o pacote a ser encaminhado. Consequentemente, ABF evita o envio de pacotes para nós adormecidos ou que falharam. Além disso, os nós não precisam manter tabelas de vizinhos.

Capítulo 5 Algoritmos para os Estágios de Disseminação e Agregação

Este capítulo propõe três algoritmos para os estágios de Disseminação e Agregação. Tais algoritmos são denominados: Inundação Restrita Clássica (*Classic Restricted Flooding*), Inundação Restrita Por Tempo (*Delayed Restricted Flooding*) e Itinerário (*Itinerary*). Eles foram concebidos para serem executados em RSSF que possuem nós propícios a falhas e nós em estado de dormência. O algoritmo Inundação Restrita Clássica dissemina a requisição dentro da região de interesse por meio de uma inundação. Esse procedimento cria uma árvore de roteamento dentro da região de interesse e esta árvore é utilizada para a agregação. Durante a disseminação, cada nó escuta os pacotes disseminados por todos os seus descendentes na árvore de roteamento. Com isso, um nó que não possui descendentes é um nó-folha. Os nós-folhas iniciam o estágio de Agregação enviando as informações coletadas por eles para os nós imediatamente superiores na árvore. Os nós não folha esperam receber todas as leituras realizadas por todos os seus descendentes, aplicam o operador de agregação definido na requisição sobre tais dados e sobre seus próprios dados e enviam o resultado para os nós superiores a eles. Esse processo é repetido até que a raiz da árvore receba as leituras de seus descendentes e calcule o resultado da requisição.

O algoritmo Inundação Restrita por Tempo realiza a disseminação da mesma forma que o algoritmo Inundação Restrita Clássica, entretanto a agregação é realizada de uma forma distinta. Várias colisões de pacotes ocorrem durante a disseminação da requisição por inundação dentro da região de interesse. Isso ocorre pois, após um determinado nó S enviar uma requisição, todos os seus vizinhos imediatos tentam reenviá-la ao mesmo tempo. Uma colisão faz com que S não receba a requisição vinda

de um de seus descendentes. Consequentemente, durante a Agregação do algoritmo Inundação Restrita Clássica, S envia o resultado antes que todos os seus vizinhos tenham enviado suas leituras. Entretanto, o algoritmo Inundação Restrita por Tempo define um tempo no qual cada nó espera as leituras de seus descendentes. Esse tempo de espera é inversamente proporcional ao nível do nó na árvore de roteamento. Com isso, os nós folhas possuem os menores tempos e o nó raiz o maior. Tal algoritmo evita colisões pois o tempo de espera de um determinado nó é configurado suficientemente grande para que todos os seus descendentes possam enviar suas leituras.

O terceiro e último algoritmo proposto é denominado Itinerário. Ele define um caminho em forma de zigzag (itinerário) dentro da região de interesse, por onde a requisição é transmitida. Somente os nós que compõem o itinerário transmitem a requisição. Os demais nós somente a recebem e enviam suas leituras para o nó sobre o itinerário que lhe enviou a requisição. Como menos nós transmitem a requisição em comparação aos dois algoritmos anteriores, o consumo de energia tende a ser consideravelmente menor. No entanto, a criação do itinerário depende muito da topologia da rede no momento em que a requisição é transmitida. Uma baixa densidade de nós pode impedir a continuação do itinerário e diminuir o número de nós que recebe a requisição. Nos algoritmos anteriores, a disseminação sofre menos influência da topologia pois todos os nós dentro da região de interesse transmitem a requisição.

Capítulo 6 - Avaliação

Os algoritmos propostos neste trabalho compõem um novo mecanismo para processamento de requisições espaciais em RSSF. A avaliação deste mecanismo foi realizada por simulação, no simulador de redes NS versão 2.34. Foram criadas três variações do mecanismo proposto. Todas elas utilizam o protocolo ABF (proposto no Capítulo 4) nos estágios de Encaminhamento e Retorno. Porém, cada variação utiliza um dos algoritmos propostos no Capítulo 5 para os estágios de Disseminação e Agregação. Tais variações são:

- Classic: utiliza o algoritmo Inundação Restrita Clássica;
- DRF (*Delayed Restricted Flooding*): utiliza o algoritmo Inundação Restrita por Tempo, e;
- Itinerary: utiliza o algoritmo Itinerário.

Os experimentos foram realizados em três fases. Na primeira fase foi avaliado o desempenho do mecanismo proposto ao utilizar o algoritmo de Pré-Processamento

descrito no Capítulo 3. Nesta fase também foram definidos os melhores valores para os parâmetros de cada uma das variações do mecanismo proposto. Verificou-se que a variação do mecanismo denominada Classic teve seu desempenho prejudicado devido ao grande número de colisões que ocorreram nos estágios de Disseminação e Agregação. Por isso, tal variação do mecanismo não foi considerada nos demais experimentos. Verificou-se também que o algoritmo de Pré-Processamento economizou cerca de 80% da energia consumida por DRF e 60% da energia consumida por Itinerary. Além disso, o tempo de processamento da requisição e a área sensoriada durante o processamento da requisição não apresentaram significava perda de desempenho.

Na segunda fase de experimentos foi analisado o desempenho do mecanismo proposto em RSSF que possuem nós em estado de dormência. Com o intuito de comparar o mecanismo proposto com o estado da arte do processamento de requisições espaciais em RSSF, foi implementado um mecanismo denominado SWIP-2. Tal mecanismo é uma variação de SWIP, que foi proposto em [Coman et al., 2007]. SWIP utiliza um protocolo guloso para os estágios de Encaminhamento e Retorno, nos estágios de Disseminação e Agregação utiliza o algoritmo de Inundação Restrita Clássica e considera regiões de interesse em forma de retângulo. SWIP-2 utiliza o protocolo GPSR [Karp and Kung, 2000] nos estágios de Encaminhamento e Retorno, utiliza Inundação Restrita por Tempo no estágio de Disseminação e Agregação e considera regiões de interesse em forma de polígono. Tais modificações foram necessárias para adaptar SWIP ao modelo de redes considerado neste trabalho.

Na segunda fase de experimento, SWIP-2 não processou requisições em redes cuja porcentagem de tempo ativo dos nós era menor que 60%. DRF e Itinerary processaram mais de 90% das requisições em praticamente todos os cenários analisados. Verificou-se também que o consumo de energia do DRF e do Itinerary foram, respectivamente, 80% e 85% menores que consumo apresentado por SWIP-2. Entretanto, Itinerary apresentou tempo de processamento 55% superior a SWIP-2, enquanto DRF apresentou tempo 15% inferior a SWIP-2.

A terceira fase de experimentos analisou o desempenho do mecanismo em RSSF com nós propícios a falhas. Foram considerados dois tipos de falhas: transitórias (os nós ficam temporariamente inativos) e permanentes (após falhar, os nós permanecem inativos). Para ambos os tipos de falha, variou-se a probabilidade de falha dos nós entre 0% e 90%. Ambos os tipos de falhas comprometeram o processamento de SWIP-2, o qual obteve resposta em apenas 30% das requisições quando a probabilidade de falha era de apenas 10% e nenhuma resposta para probabilidade de falha superior a 40%. DRF e Itinerary obtiveram resposta em respectivamente 95% e 90% das requisições quando a probabilidade de falha era de 10%. Com probabilidade de falha

igual a 40%, tais variações do mecanismo proposto obtiveram resposta em mais de 80% das requisições. O consumo de energia de SWIP-2 foi até 400% superior ao consumo de DRF e Itinerary em praticamente todos os cenários.

Capítulo 7 - Conclusão

O capítulo final da tese resume as contribuições e conclusões da pesquisa e traz sugestões para trabalhos futuros. A primeira contribuição foi o estudo sobre processamento de requisições espaciais em RSSF. Neste estudo, o processamento de requisições foi dividido em estágios. Essa divisão facilitou a compreensão e a descrição dos mecanismos para processamento de requisições espaciais até então encontrados na literatura. Verificou-se nestes estudos que tais mecanismos não consideraram regiões de interesse em formato de polígono. Além disso, eles assumiam que os nós da rede sempre estão aptos a receber e transmitir mensagens. Tal afirmação não é válida em todos os cenários e aplicações de RSSF, pois os nós sensores podem falhar ou adormecer para economizar energia.

A partir da análise dos trabalhos relacionados foi proposto um novo algoritmo para o estágio de Pré-Processamento, um novo protocolo de roteamento geográfico denominado ABF para os estágios de Encaminhamento e Retorno e três algoritmos para os estágios de Disseminação e Agregação. O algoritmo para o estágio de Pré-Processamento reduziu consideravelmente o consumo de energia do processamento de requisições espaciais. Foram criadas três versões do mecanismo de processamento de requisições espaciais aqui propostos, denominadas: Classic, DRF e Itinerary. Todas essas versões utilizaram o algoritmo proposto para o estágio de Pré-Processamento e o protocolo ABF para os estágios de Encaminhamento e Retorno. Entretanto, cada uma utiliza um dos três algoritmos propostos para os estágios de Disseminação e Agregação.

As três variações do mecanismo proposto foram comparadas com SWIP-2, uma versão aprimorada do mecanismo SWIP proposto em [Coman et al., 2007]. Classic apresentou baixo desempenho nos experimentos preliminares, por isso foi descartado nos demais experimentos. Verificou-se que DRF e Itinerary possuem consumo de energia cerca de quatro vezes menor que SWIP-2. Além disso, as variações do mecanismo proposto aumentaram consideravelmente o número de requisições que obtiveram resposta após seu processamento. A versão do mecanismo denominada Itinerary apresentou o menor consumo de energia, porém também apresentou o maior tempo de processamento em praticamente todos os cenários. Esses resultados mostram que os algoritmos propostos atingiram os objetivos estabelecidos no início desta tese.

Como trabalhos futuros, pretende-se considerar várias características de RSSF

reais que não são consideradas pelos trabalhos até então encontrados na literatura. Podem ser citadas características como a variação da qualidade dos enlaces sem fio, imprecisão na localização de nós e mobilidade de nós. Outro ponto a ser analisado é a utilização de coordenadas em três dimensões. Tais coordenadas influenciariam o processamento de requisições espaciais principalmente porque alterariam a definição da região de interesse, a qual deveria ser considerada como um poliedro. Finalmente, pretende-se implementar o mecanismo proposto em redes de sensores reais, com o intuito de comparar os resultados obtidos nas simulações com os resultados da implementação real.

List of Figures

1.1	Forest contours on a satellite photo.	2
2.1	(a)Window query. (b)K-nearest neighbor query.	8
2.2	Query classification.	9
2.3	Six stages of spatial query processing.	14
2.4	(a) Greedy: hole in the network. (b) GPSR: the query bypasses the hole.	21
2.5	Window query processing based on itinerary.	24
2.6	(a)Itineraries created by DIKNN. (b)Itineraries created by PCIKNN. (c)Spiral itineraries created by IKNN mechanism. (d)Parallel itineraries created by IKNN mechanism.	25
2.7	Number of packets transmitted during the Aggregation stage using a naive strategy.	27
2.8	(a) Aggregation performed outside the region of interest. (b) Aggregation only inside the region of interest.	27
2.9	Query dissemination with Georouting Tree.	30
2.10	Parent-switching verification phase.	31
2.11	Semi-distributed index: subregions, nodes and cluster-heads.	31
2.12	(a)MBR created by SPIX; (b)MBR for a more efficient aggregation.	32
2.13	The algorithm and the stages they are performed.	33
2.14	Problem Statement classification.	35
2.15	Problem Solution classification.	35
3.1	(a)Contours of a lake. (b) Simplified contours of a lake.	44
3.2	Collateral effects of the Pre-Processing algorithm.	48
4.1	Reference point in Forwarding: (a) when Restricted Flooding is used to disseminate, (b) when itinerary is used to disseminate	51
4.2	State diagram of a forwarder node.	52
4.3	State diagram of a node that received a request-packet.	53

4.4	Points defined in the Forwarding stage.	53
4.5	Hole in the network and node position projected on the line \overline{SR}	54
4.6	Forward failure.	55
5.1	State diagram of the Classic Restricted Flooding.	60
5.2	The value of T_{Agg} in the routing tree.	61
5.3	State diagram of Delayed Restricted Flooding.	62
5.4	Itinerary definitions.	63
5.5	State diagram: dissemination using itinerary.	63
5.6	(a)Node calculate the new direction (RIGHT) and new DII (b) Node calculate the new direction (DOWN) and new DII	66
6.1	Irregular regions of interest over satellite photos (Lake).	70
6.2	End-to-end delay of Classic and DRF using different values of $T_{Decrease}$	74
6.3	Coverage area of Classic and DRF using different values of $T_{Decrease}$	75
6.4	Energy consumption of Classic and DRF using different values of $T_{Decrease}$	75
6.5	End-to-end delay of Itinerary.	77
6.6	Coverage area of itinerary.	77
6.7	Energy consumption of itinerary.	78
6.8	End-to-end delay for different values of ID.	79
6.9	Coverage area for different values of ID.	80
6.10	Energy consumption for different values of ID.	81
6.11	Comparing the query processing success of SWIP-2, DRF and Itinerary.	82
6.12	Comparing the end-to-end delay of SWIP-2, DRF and Itinerary.	82
6.13	Comparing the coverage area of SWIP-2, DRF and Itinerary.	83
6.14	Comparing the energy consumption of SWIP-2, DRF and Itinerary.	84
6.15	Query processing success under transient failures.	85
6.16	End-to-end delay under transient failures.	86
6.17	Coverage area under transient failures.	87
6.18	Energy consumption under transient failures.	87
6.19	Query processing success under permanent failures.	88

List of Tables

2.1	Mechanisms for Spatial Query Processing	36
6.1	Iris Sensor Node Characteristics	71
6.2	Experiments Summary	90

Contents

Acknowledgments	xi
Resumo	xiii
Abstract	xv
Resumo Estendido	xvii
List of Figures	xxvii
List of Tables	xxix
1 Introduction	1
1.1 Motivation	3
1.2 Objectives	4
1.3 Thesis Contributions	5
1.4 Thesis Outline	6
2 Spatial Query Processing For Wireless Sensor Networks	7
2.1 Spatial Query Classification	7
2.2 Problem Statement	8
2.2.1 The Focus This Thesis	10
2.3 Definitions	11
2.4 Dynamic Topologies in WSN	12
2.4.1 Duty Cycle in WSN	13
2.4.2 Fault Tolerance in WSN	13
2.5 Spatial Query Processing Organization	13
2.5.1 Spatial Query Processing Stages	14
2.5.2 Stage Formulation	15
2.6 Algorithms for the Spatial Query Stages	18

2.6.1	Pre-Processing Stage	19
2.6.2	Forwarding Stage	19
2.6.3	Dissemination Stage	22
2.6.4	Sensing Stage	26
2.6.5	Aggregation Stage	26
2.6.6	Return Stage	28
2.6.7	Distributed Spatial Indexes	29
2.6.8	Spatial Query Processing Stage Remarks	32
2.7	Spatial Query Processing Mechanisms	34
2.7.1	Spatial Query Processing Classification	34
2.7.2	Overview of the Mechanism	35
2.7.3	Window Query Processing	37
2.7.4	KNN Query Processing	39
2.8	Chapter Remarks	41
3	Simplifying the Region of Interest	43
3.1	Problem Definition	43
3.2	Relative Geographical Coordinates	44
3.3	Simplifying the Regions of Interest	44
3.4	Complexity Analysis	46
3.5	Impact of the Simplification in the Query Processing	47
3.6	Chapter Remarks	48
4	ABF - Ask Before Forwarding Routing Protocol	49
4.1	Problem Definition	50
4.2	Coordinator Selection	50
4.3	ABF Strategy	51
4.4	The Delay to Answer a Request-Packet	53
4.5	Avoiding Holes in the Network	54
4.6	Node Failure Detection	54
4.7	Forwarding Failure	55
4.8	Chapter Remarks	56
5	Dissemination and Aggregation Stages	57
5.1	Dissemination/Aggregation Problem Definition	58
5.2	Classic Restricted Flooding	58
5.3	Delayed Restricted Flooding	60
5.4	Itinerary	62

5.4.1	Itinerary Terminology	63
5.4.2	Itinerary Algorithm	64
5.5	Chapter Remarks	66
6	Evaluation	69
6.1	Simulated Experiments Design	69
6.1.1	Assumptions	69
6.1.2	Simulation Setup	70
6.2	Implementations	70
6.3	Metrics	72
6.4	Pre-Processing Algorithm and Parameter Evaluation	73
6.4.1	Evaluating Classic vs DRF	73
6.4.2	Evaluating the Itinerary	76
6.5	Varying the Duty Cycle	80
6.6	Evaluation Under Node Failures	84
6.6.1	Failure Model	84
6.6.2	Performance Evaluation Under Transient Failures	85
6.6.3	Performance Evaluation under Permanent Failures	87
6.7	Simulated Experiment Remarks	88
7	Final Remarks	93
7.1	Conclusions	93
7.2	Directions for Future Research	96
7.3	List of Thesis-related Publications	97
A	Abbreviations	99
	Bibliography	101

Chapter 1

Introduction

Wireless Sensor Networks (WSN) consist of multiple autonomous tiny devices called **sensor nodes**. This kind of network is used to collect environmental data in regions that are inaccessible or present risks to human safety. Each sensor node has a microprocessor, memory, a set of sensors, batteries and can communicate with other sensors by radio. Usually, these nodes present severe resource constraints. Among them, the limited energy supply is considered the most restrictive limitation. Generally, the users' interface with the network is a special node with more resources, called **Base Station** or **Sink**. The network commands come from this node and the data collected by the nodes flows to it. When the sink is not within the radio range, nodes create routes with multiple hops in order to deliver the collected data. Hence, each sensor node can work as a sensor (collecting environmental data) and as a router (retransmitting data collected by other nodes) [Akyildiz et al., 2002].

WSN are useful in a variety of monitoring applications, such as health, environmental, industrial and structural. WSN applications must be able to adapt themselves to eventual topology changes caused by nodes' energy depletion, environmental changes, node damages and external sources of interference. These events often occur during the WSN operation, since these networks traditionally monitor inhospitable areas [Chen et al., 2010]. Furthermore, these applications have to handle efficiently the energy supply, since a battery replacement is impractical due to the number of nodes or the place where they were deployed. Hence, fault tolerance and energy efficiency are concepts that permeate the design of all WSN.

As WSN technology matures, its role as a tool for periodic data gathering is expanding to provide data gathering in response to user queries. Several works model WSN as a distributed database and define energy-aware mechanisms for in-network query processing, such as TinyDB [Madden et al., 2005] and Cougar [Demers et al.,

2003]. They process declarative SQL-like queries, which simplify the way users obtain data from the WSN. In-network query processing mechanisms can use location information to answer a special type of query, called **spatial query**. In these queries the users' interests are expressed by geographical predicates. Thus, the user can establish a sub-region of the monitored area and ask for data collection only inside this sub-region.

Spatial queries are supported by geographic database [Felice et al., 2008]. They differ from traditional queries in two main points. First, they incorporate geometric data types, such as points and polygons. Second, they consider the spatial relationships between the defined geometries, such as a point *inside* a polygon or a polygon that *overlaps* another. WSN can be modeled as distributed geodatabases in order to process queries containing spatial information, which request data collected at a **region of interest (RoI)** defined by the user.

In general, the works about spatial query processing in WSN have rectangular RoIs, which need only two points to be represented. To our knowledge, none of these works treat the processing of queries in which the RoI has irregular contours (such as polygons). Polygon shapes need more than two points for their representation. Thus, the query processing uses more packets, increasing the energy consumption. However, polygons can represent real objects plotted on maps or satellite photos, i.e., users can specify exactly the sub-region where they want sensors to collect data. Figure 1.1 illustrates the contour of a forest in a satellite photo that is defined by a polygon.



Figure 1.1. Forest contours on a satellite photo.

Works about spatial queries in WSN also consider that the sensor nodes are always able to transmit and receive data. However, nodes can fail due to interference and energy depletion or can be destroyed during the network operation. Usually, these

works assume that no packets are lost. Nodes also can enter in sleep mode (turn off their radio) in order to avoid the energy consumption of **idle listening**. This kind of consumption occurs when a node is active (the radio is on) and is not transmitting or receiving packets [Silva et al., 2006]. Thus, during a spatial query processing, when a node has a packet to transmit, it cannot attest that one of its immediate neighbors is active since they can be sleeping or have failed. Spatial query processing mechanisms need to manage such characteristics in order to efficiently process queries.

The core research question of this thesis is “how to efficiently process spatial queries in wireless sensor networks?” We consider to be efficient the query process that consumes just a small part of the energy supply when compared to other spatial query mechanisms found in the literature. Moreover, we consider that the query result must come from a region that represents the largest possible part of the region of interest defined by the user and the time to perform this processing must be comparable to related works. Finally, all proposed algorithms must assume WSN having failure-prone nodes, nodes employing duty cycle algorithms and RoIs shaped as polygons.

This work proposes a new fault-tolerant energy-efficient in-network spatial query processing mechanism for WSN. We consider that the RoIs are polygons that represent the contours of real objects plotted on maps or satellite photos. We also consider WSN employing duty cycle algorithms to save energy, thus there are some nodes in sleep mode during the query processing. Furthermore, we assume that nodes can be destroyed during the network operation or can fail due to several factors, such as external sources of interference and energy depletion. Hence, the proposed mechanism works properly when nodes are transitioning between active and sleep modes and works properly in failure-prone WSN.

1.1 Motivation

The initial motivation of this work was to use WSN as a tool to detect hazards during disaster situations. However, we verified that the contributions could be applied in any WSN application that during the network operation enables the user to choose the necessary data and to define the region where these data are to be collected. Since the required data are not known a priori, periodic data collection is not important. These characteristics can be found in applications such as micro weather monitoring, pollution analysis, fire hazard detection, etc. Another characteristic of these applications is the possible use of Geographic Information Systems (GIS) in order to provide graphical interfaces in which the user can see the collected data and trace the contour of the

regions of interest. For all of these applications, the WSN must be able to process spatial queries efficiently.

During studies about disaster situations, we verified that the ability of the first responders to solve problems quickly can save lives and reduce the material damage caused by the disaster. Information about the available resources and about the region where the disaster has occurred can help them. Several types of data are important, such the ones collected by a human observer located inside the crisis region, maps, satellite photos, videos and data collected by sensors. The latter are important because sensors can provide an overview of the region, informing them of imminent dangers. However, the network infrastructure that could be used to transmit this data usually is destroyed during the disaster, there is interference in the wireless communication, sensors can be destroyed during their operation and all data are closely associated with the location where they were collected. Hence, the sensors used in these situations need to be easily deployed, rapidly self-configured, aware of their location and able to cope with failures [Rao et al., 2007].

WSN are an alternative for sensing in disaster regions, since WSN have all the characteristics mentioned above. This kind of network is simple to be deployed and configured since sensor nodes can be randomly spread on the monitored area and they automatically create routes to deliver data to users [Akyildiz et al., 2002]. WSN also can be fault tolerant, since more than one route is possible for each destination. Moreover, if a node fails, it can be replaced by another node, creating a new route and maintaining the network operational [Macedo et al., 2005]. Finally, node location can be obtained by several solutions found in the literature [Wang et al., 2010].

However, we verify the absence of a spatial query processing mechanism that considers all these characteristics at the same time. To the best of the authors' knowledge, works about spatial query processing do not consider RoIs having polygon shapes, they assume that nodes are always able to send and receive messages and, in the majority of these works, only the base station can start a query processing. However, this is not a realistic scenario, since nodes fail due to several factors and most of WSN employ duty cycle algorithms to save energy. Hence, we consider all of the above mentioned characteristics in order to propose a new spatial query mechanism.

1.2 Objectives

The first goal of this work is to advance the state of the art of spatial queries, both its theory and practice, by means of the two goals below.

The main goal is the design and evaluation of a novel in-network spatial query processing mechanism for WSN. In order to design a mechanism adapted to more realistic network characteristics, the following assumptions were considered: First, spatial queries should support regions of interest in the form of polygons. Second, the mechanism should support dynamic topologies, which occur due to the use of duty cycle algorithms in order to reduce the energy consumption of the network. Finally, the mechanism should consider fault tolerance, since sensor nodes are prone to fail.

Our second goal is to provide a comprehensive overview of spatial queries for WSN. This should be achieved by means of a survey of the literature, the formal definition of the stages of a spatial query, as well as the categorization of the existing solutions using a new taxonomy.

1.3 Thesis Contributions

This thesis presents three main contributions for in-network spatial query processing in WSN:

An understanding of spatial query processing in WSN. We studied the main spatial query processing mechanisms for WSN. We divide the problem of spatial queries into stages representing sub-problems, and then formalize the objectives of each stage by describing them as graph optimization problems. The algorithms found for each stage were described and analyzed separately. We also classified the mechanisms and queries found in the literature. We created a survey that provides a broad view of this type of processes that, by analyzing the stages separately, we can understand the individual characteristics of each mechanism and identify open issues.

A novel energy-efficient fault-tolerant mechanism for in-network spatial query processing. The mechanism proposed here assumes some characteristics that should be considered in WSN for disaster situations and were not considered in the literature. These characteristics are: RoIs shaped as polygons, WSN with nodes in sleep mode due to duty cycle, and failure-prone nodes. The WSN considered here are closer to the real WSN than those described in the literature. Furthermore, the proposed spatial query processing mechanism saves energy since it considers these characteristics of WSN. By assuming node failures and by having no knowledge about the sleeping schedule, nodes do not need to maintain routing tables in order to process queries. Thus, WSN avoid several beacon messages intended for routing table updates.

A performance evaluation of the proposed mechanism. We evaluated the proposal using simulations. We implemented four variations of the proposed mechanism

and a mechanism that represents the state of the art in spatial query processing. Exhaustive experiments were performed in order to evaluate the performance of the proposal and to compare the proposal against the state of the art. In most scenarios, the proposed mechanism presents 300% more successful queries and 80% less energy consumption. The experiments showed that the proposed mechanism is more fault-tolerant and consumes less energy than the mechanisms presented in the related works.

1.4 Thesis Outline

Chapter 2 provides an overview of spatial query processing in WSN. This chapter first describes the main characteristics of this type of processing and then divides it in stages. Next, it describes separately the algorithms found in the literature that implement each stage. From these descriptions, we classify and analyze the most relevant spatial query processing mechanisms. We also present the taxonomy used during the remaining of the text.

In Chapters 3, 4 and 5 we propose algorithms that constitute a new energy-aware fault-tolerant spatial query processing mechanism. Chapter 3 describes an algorithm that reduces the energy consumption by simplifying the contours of the RoI and, consequently, reducing the amount of packets used to process the query. In Chapter 4 we propose a geographic routing protocol to forward the query to the RoI and to forward back the query result to the user. This protocol avoids failed and dormant nodes during the packet transmission. It assumes no knowledge of the network topology, hence nodes must request the next node to forward packets among their active neighbors. Chapter 5 presents three new algorithms to disseminate the query inside the RoI and to calculate the query result. They also are devised to avoid failed and dormant nodes.

In Chapter 6 we evaluate the proposed spatial query mechanism using simulations. We first define the best parameter values for the proposed algorithms, then we analyze the performance of the proposal in WSN employing duty cycles and in WSN having failure-prone nodes. Finally, in Chapter 7 we present the thesis final remarks and give directions to future research.

Chapter 2

Spatial Query Processing For Wireless Sensor Networks

This chapter presents a study of spatial query processing mechanisms for wireless sensor networks. Section 2.1 presents the classification of the queries found in the literature. Section 2.2 presents the problem definition. Section 2.3 presents some definitions of terms that are used during this thesis. Section 2.4 discusses topology changing in WSN and presents an overview about duty cycle and fault tolerance in WSN. In Section 2.5 we propose to divide spatial query processing into six stages and Section 2.6 separately describes a set of algorithms for each stage. This division simplifies the understanding of spatial queries and facilitates the proposal of new solutions. Section 2.7 surveys the literature describing the characteristics of the most important spatial query processing mechanisms. Finally, in Section 2.8 we present Chapter Remarks.

2.1 Spatial Query Classification

This section classifies the queries found in the literature. This classification is based on the selection of the nodes that will answer the query (called **Selection Classification**) and the way they will answer the query (called **Answer Classification**). Considering the Selection Classification, there are two types of spatial queries: window queries and k-nearest neighbors queries (KNN). **Window queries** are the most common type of spatial queries. The user defines a region of interest (also called **window**) and asks for data collected by the sensor nodes inside this region. In the literature, regions of interest are defined by rectangles [Soheili et al., 2005; Li et al., 2008; Park et al., 2007]

or circles [Lu et al., 2005]. These queries are also called range queries¹[Li et al., 2008]. Figure 2.1 (a) illustrates a Window query in a WSN.

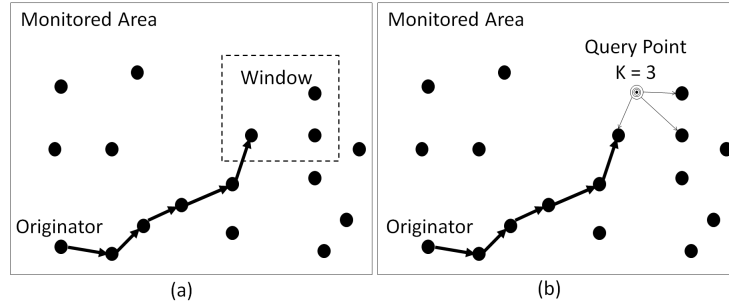


Figure 2.1. (a) Window query. (b) K-nearest neighbor query.

K-Nearest Neighbor queries (KNN) retrieve data from the K nodes that are closest to a point defined by the user, called **query point** [Wu et al., 2007; Xu et al., 2007; Fu et al., 2010]. Figure 2.1 (b) illustrates a KNN query. We also found in the literature a type of query called **Nearest Neighbor queries (NN)**. It is a special case of KNN, with $K = 1$ (such as in [Demirbas and Lu, 2007]).

Considering the Answer Classification, we found three types of spatial queries: snapshot, spatiotemporal and longlife. **Snapshot** queries retrieve data once. Sensor nodes sense the environment, transmit the collected data and wait for a new query. **Spatiotemporal** queries specify a window of time (a lower and an upper limit) for the data to be retrieved. These queries manipulate data stored at the flash memory of the sensor nodes or data collected periodically by the nodes, which are stored at the sink node memory. **Longlife** queries generate periodic answers. In these queries, the users specify the reporting period and the query lifetime.

It is important to mention that Selection Classification and Answer Classification do not interfere with each other. Window query can be snapshot, spatiotemporal or longlife. The same can be said for KNN queries. Figure 2.2 illustrates the query classification presented here. We analyze mechanisms that process all these types of queries, but we focus on window and snapshot queries.

2.2 Problem Statement

We consider queries such as: “What is the average temperature collected by the nodes inside the region defined by the user?” Nodes inside the selected region collect their

¹In traditional database literature, range queries select data between an upper and lower boundary; however, these boundaries may not be related to spatial constraints.

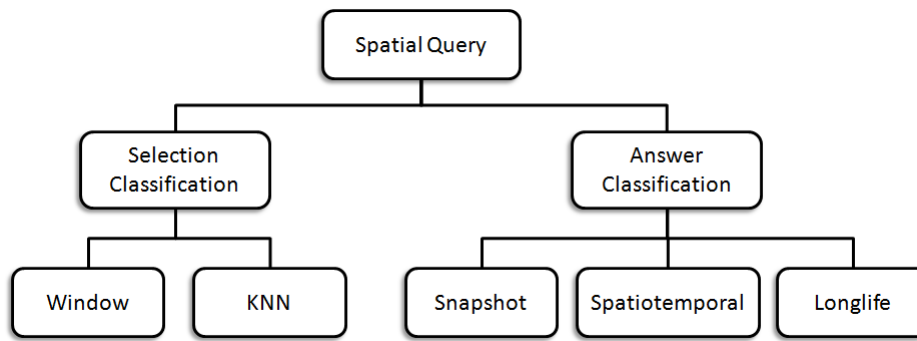


Figure 2.2. Query classification.

reading and perform in-network aggregation (described in Section 2.6.5) in order to reduce the number of packets to be transmitted [Sharifzadeh and Shahabi, 2004]. The **query result** is the value expected by the user as the final result of the spatial query processing. Here, this value represents the average reading collected by the nodes inside the RoI. In the SQL-Like language described by Madden et al. [2005], this spatial query is expressed as:

```

SELECT avg(Temp)
FROM sensors
WHERE inside(region)
ONCE
  
```

WSN are modeled as a database with a single table called *sensors*. Each sensor reading is an attribute of this table. The aggregation functions considered are *average*, *count*, *maximum value*, *minimum value*, and *sum*. All of them can be performed by the same algorithm [Sharifzadeh and Shahabi, 2004]. *Inside* is a function that verifies if a node is within the RoI. This region is defined in the variable *region*, which is a sequence of points that defines a polygon. The number of points can be large, thus the query may require more than one packet to be transmitted between neighboring nodes².

The monitored area is a 2D square region, where the nodes are uniformly randomly deployed. Each node has a GPS device or can calculate its location using one of the methods found in the literature [Wang et al., 2010]. The sensor nodes are homogeneous, static and links are symmetric (if *A* hears *B*, *B* hears *A*). We consider that nodes can fail during the spatial query processing. The nodes do not know neither when they will fail nor how long they will stay in such state. During failure,

²Packets in WSN usually have a small size, about thirty bytes [Crossbow Technology, Inc., 2010a; Moteiv Corporation, 2006; Crossbow Technology, Inc., 2010b].

nodes are unable to transmit or receive messages. Since any node can fail anytime, it is impossible for a node to define exactly the state of a specific neighbor. We also consider nodes sleeping during query processing due to duty cycle algorithms present in the MAC protocols [Demirkol et al., 2006]. Since there are several types of duty cycle algorithms, we assume that nodes have no knowledge about their sleep schedule.

The region of interest is represented by a polygon (*RoI*) that is defined as a set of n points $(p_0, p_1, \dots, p_{n-1})$. The point p_i is defined by two coordinates x_i and y_i , where x_i is the longitude and y_i is the latitude of p_i . Two consecutive points p_i and p_{i+1} form a line that represents one of the sides of the *RoI*. The line connecting the first point p_0 to the last point p_{n-1} also represents one of the sides of the *RoI*.

The spatial query processing problem considered in this work is to process queries that are described as $Q = \{Reading, Agg, RoI\}$, where *Reading* = $\{Temperature, Humidity, Light, \dots\}$ is one reading that the WSN can sense in the environment, *Agg* = $\{avg, count, sum, max, min\}$ is the aggregation operator and *RoI* is the region of interest. The expected query result (*QR*) is the value defined by the equation:

$$QR = Agg\left(\bigcup_{v_x \in I} Data_x\right) \quad (2.1)$$

where $Data_x$ is the reading collected by the node v_x and I is the set of nodes inside the *RoI*.

2.2.1 The Focus This Thesis

The literature presents several algorithms that obtain data from nodes inside regions defined by users. However, we consider them out of focus of this work, since they cannot be applied directly to solve the problem considered here and we are sure they would not present good results. One of them is Directed Diffusion [Intanagonwiwat et al., 2003], which is intended to delivery data from the sensor nodes to the sink. Using this algorithm, the sink broadcasts a packet called *interest*. This packet defines the event that must be monitored. During the *interest* dissemination, nodes create routes to deliver data to sink. If a node sense an event, it will start sending data to the sink using one of these routes. The route used to delivery data to the sink changes periodically in order to balance the energy consumption among the nodes. The *interest* packet can be replaced by a query packet, which defines the *RoI*. However, all nodes would receive all queries, increasing the energy consumption. Furthermore, Directed Diffusion is devised to obtain data from all nodes that sensed a event. It does not

aggregate data and aggregation is one of the requirements of this work.

Another algorithm that would be used to process spatial query is TBF+ (Trajectory Based Forwarding) [Niculescu and Nath, 2003]. This algorithm uses the energy map of the network to defined routes to forward packets. The energy map is the information about the remaining available energy in each part of the network [Mini et al., 2005]. All nodes send to the sink their remaining available energy in order to create the energy map. Knowing these data, the sink node is able to create equations that establish trajectories that must be followed by queries to reach nodes inside the RoI. However, this solution would only process queries starting in the sink. Furthermore, even using predicted techniques to generate the energy map, all nodes send their energy supply state to the sink, consuming substantial energy of the network.

This work focused on spatial query processing mechanisms that perform all spatial query processing, i.e., we analyze mechanisms those forward queries to nodes inside RoIs and receive the query results. We did not analyze algorithms that perform only part of the spatial query processing, such as routing protocols to forward queries or algorithms to disseminate. Otherwise, the research of this thesis would analyze a very large number of algorithms found on the literature. Furthermore, most of these algorithms would need many improvements in order to adapt them to perform part of the spatial query processing.

2.3 Definitions

In this section we present a set of definitions in order to help the reader to understand this thesis. We also describe how we use some common terms of computer networks in the context of spatial query processing.

- **Spatial Query Processing in WSN:** is a sequence of steps that are performed by sensor nodes in order to collect data only inside a region or next to a point defined by the user.
- **Spatial Query Mechanism:** we use this term to define a set of algorithms that work together to process spatial queries. This definition is based on Cobuild [2003] who defines a mechanism as “a set of parts that work together”.
- **RoI:** or region of interest is a geographical region having a set of sensor nodes that collect environmental data in response to a spatial query. In the literature, the RoI is frequently called window because they usually have rectangular shapes. We consider polygonal RoIs.

- **Aggregation:** math operation that is applied to a set of information and generate only one information. As example, the calculation of the average, which generate one value from a set of values.
- **Protocol:** during this text we employ the word *protocol* to define algorithms that create routes or algorithms used in the MAC layer. It is important to mention that we do not employ protocol to define the algorithms that perform both query dissemination inside the RoI and readings aggregation.
- **Duty Cycle:** is the fraction of time that a sensor node is in active state.
- **Duty Cycle Algorithms:** algorithms that control the transition between the active and sleep state. Traditionally, they are part of MAC protocols.
- **Length of the RoI or $length_{RoI}$:** is the length of the polygon that represents the region of interest. In this work, $length_{RoI}$ is the largest distance between any two points in the RoI.
- **Sensing Range:** is the radius of a circular region called sensing area, in which a sensor node collect environmental data. We consider that each sensor node is in the center of its sensing area. The data collected by a sensor node represents the average data collected in all points inside the sensing area. We consider that the sensing range of the nodes is the half the radio range, according to [Cardei and Wu, 2006].

2.4 Dynamic Topologies in WSN

The literature presents several works that analyze factors that change the WSN topology, such as duty cycle, node failures, link quality variation and node mobility. Duty cycles periodically turn off the radio of the sensor nodes in order to reduce the energy consumption with idle listening. Node failures are common events in WSN since nodes can be destroyed, they can die due to energy depletion and external sources of interference can block the communication. The energy supply and the interference can vary the quality of the links during the WSN operation. Hence, we assume that the maintenance of information about the WSN topology is very expensive in terms of energy consumption or even impossible in several scenarios for sensor nodes. In the following subsections, we present an overview about duty cycle and failures in WSN. This overview is intended to affirm our assumption. We do not analyze link quality variation and node mobility because it is not the focus of this work.

2.4.1 Duty Cycle in WSN

We found two main types of duty cycle algorithms: synchronous or asynchronous. Synchronous algorithms put all nodes of the WSN into sleep or active mode at the same time, such as the algorithms described in [Du et al., 2007; Ye et al., 2004; Sun et al., 2008a; van Dam and Langendoen, 2003]. Asynchronous algorithms consider that nodes change to sleep mode periodically, but not all at the same time. Nodes can use preambles to wake up their vicinity [Polastre et al., 2004; Buettner et al., 2006; El-Hoiydi and Decotignie, 2004] or can send beacons to inform that they are active [Sun et al., 2008b]. Other algorithms use routing or location information to define which nodes will sleep and when they will wake up [Swain et al., 2010; Yen and Cheng, 2009]. Moreover, some algorithms use random strategies to choose which nodes will sleep, such as in [Silva et al., 2006]. Recent works consider a new model of sensor nodes that maintains all nodes in sleep mode. They use RFID tags to turn on the radios in the vicinity before a transmission [Jurdak et al., 2010].

2.4.2 Fault Tolerance in WSN

Generally, works about fault tolerance assume that nodes have no knowledge about node failures (when and which node will fail). Hence, nodes cannot attest the state of their neighbor during packets forwarding. Furthermore, the literature presents several works concerning fault tolerance in WSN, which study several types of failure and techniques to deal with them. In [Paradis and Han, 2007] the authors classify the techniques that deal with different types of faults in distributed system as: fault prevention, fault detection, fault isolation, fault identification and fault recovery. Then, they analyzed these techniques in several applications and protocols of WSN. In [Macedo et al., 2005] the authors study the performance of routing protocols in failure-prone WSN and classify failures by their causing agents: atmospheric phenomena, mobile sources of interference, natural disasters, accidental breakage, processor crash, interference attacks, collision attacks and sinkhole attacks. Furthermore, faults are classified as persistent and resilient. We verify several types of failure, hence nodes having packets to forward cannot deduce the state of their neighborhood.

2.5 Spatial Query Processing Organization

In this section, we propose to divide the spatial query processing in stages. Each stage represents a sub-problem that the spatial query processing mechanism must solve in

order to process spatial queries. The related works define different numbers of stages for spatial query processing in WSN. Coman et al. [2007] and Li et al. [2008] divide their proposed mechanisms into two stages and Lu et al. [2005] divide theirs into three. However, these stages can be subdivided into simpler stages. We analyze spatial queries in six stages: Pre-Processing, Forwarding, Dissemination, Aggregation, Sensing and Return. This finer grained division facilitates the understanding of the mechanisms, making it easier to identify problems and to propose new solutions.

2.5.1 Spatial Query Processing Stages

Figure 2.3 illustrates the six spatial query processing stages proposed here. The user, represented by the computer, defines the RoI. The **Pre-Processing** stage is performed in the user's computer. It prepares queries to be sent to the WSN. The first sensor node to receive the query is named **Originator**. In the **Forwarding** stage, the query is forwarded from the Originator to a node within the RoI. This last node in the Forwarding stage is called **Coordinator**. In the **Dissemination** stage, the query is disseminated from the Coordinator to all nodes inside the RoI. These nodes sense environmental data in the **Sensing** stage. In the **Aggregation** stage, these nodes forward this information to a node called **Aggregator**, which applies the aggregation operator on the received data in order to calculate the query result. Finally, in the **Return** stage, the Aggregator forwards the query result to the Originator.

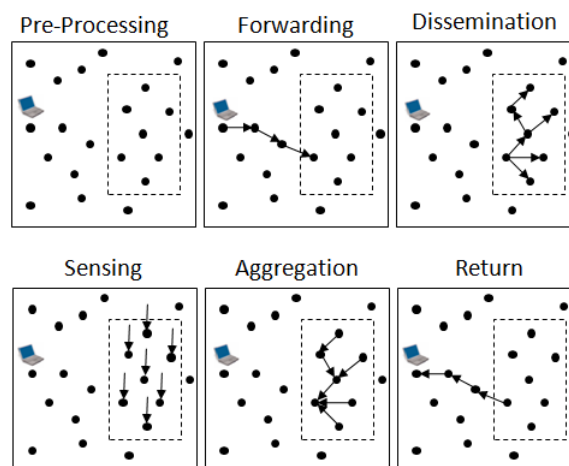


Figure 2.3. Six stages of spatial query processing.

In the following, we formally describe these stages using equations. However, we do not describe the Pre-Processing and Sensing stages, since we focus on the network operation and these stages only depend on the application. Then, we present a set of

algorithms for each stage. These algorithms are part of the spatial query processing mechanisms found in the literature.

2.5.2 Stage Formulation

We now give a formal description of the problem that the spatial query processing mechanism must solve in each stage. We represent the time spent by the network to process a query as $T = \{t_0, t_1, \dots, T_{end}\}$, where t_0 is the beginning of the query processing (the Originator received the query) and t_{end} the end of the query processing (the Originator received the query result). We represent the WSN as a directed graph $G = (V, E, s, w)$ where:

- $V = \{v_1, v_2, \dots, v_n\}$ is a set of vertices representing the sensor nodes;
- E is a set of edges representing the radio links;
- the directed edge $e_{i,j} \in E$ iff $\|(x_i, y_i) - (x_j, y_j)\| \leq r$, where r is the radio range of the nodes, $e_{i,j}$ is directed from v_i to v_j and $\forall (e_{i,j} \in E) \exists (e_{j,i} \in E)$ since we consider symmetric links;
- $s : (T, E) \rightarrow \{0, 1\}$ is the function that determines the state of node v_i at time t_x . s follows the equation:

$$s(t_x, v_i) = \begin{cases} 1, & \text{if node } i \text{ is active at time } t_x; \\ 0, & \text{otherwise} \end{cases}$$

We represent the set of 1-hop active neighbors of node v_i at time t_x by $N_{i,x}$. Hence, $(v_y \in N_{i,x})$ iff $(e_{i,y} \in E) \wedge (s(t_x, v_i) = 1) \wedge (s(t_x, v_y) = 1)$. Since we consider that all transmissions occur in broadcast, when node v_i transmits a packet at time t_x , v_i consumes energy to transmit and all nodes in $N_{i,x}$ consume energy to receive this packet. We represent the energy consumption of the WSN when a node transmits a packet using $w : (T, V) \rightarrow \mathfrak{R}$, which follows the equation:

$$w(t_x, v_i) = E_{TR(v_i)} + \sum_{v_y \in N_{i,x}} E_{RE(v_y)} \quad (2.2)$$

where $E_{TR(v_i)}$ is the energy consumed by node v_i to transmit a packet, and $E_{RE(v_y)}$ is the energy consumed by node v_y to receive a packet. Next, we present the formulation of each proposed stage.

2.5.2.1 Forwarding and Return Stage Statement

In the Forwarding stage, a spatial query processing mechanism must forward queries from the Originator to a node inside the RoI, consuming the least amount of energy. In the Return stage, the problem is practically the same, but the source node is a node inside the RoI and the destination is the Originator. The destination in the Forwarding stage and the source in the Return stage depend on the dissemination algorithm. We can formally define the problem of the Forwarding and Return stages as follows:

Forwarding/Return Problem. Given a graph $G = (V, E, s, w)$, a source node $v_s \in V$, a destination node $v_d \in V$ and a period of time $T_f = \{t_0, t_1, \dots, t_{end}\}$ in which a set of packets are transmitted from v_s to v_d , find a sequence of nodes $v_s \rightsquigarrow v_d = \{v_s, v_0, \dots, v_d\}$ that minimizes the energy consumption, as stated in the following equation:

$$\arg \min_{v_s \rightsquigarrow v_d} \sum_{v_i \in v_s \rightsquigarrow v_d} w(t_x, v_i) \quad (2.3)$$

The sequence $v_s \rightsquigarrow v_d$ should also satisfy the following constraints:

1. Path constraint: $(\forall (v_i, v_{i+1}) \in v_s \rightsquigarrow v_d) \exists (e_{i,i+1} \in E)$;
2. Node state constraint: a node v_i transmits a packet to node v_{i+1} at time $t_x \in T_f$ if $(s(t_x, v_i) = 1) \wedge (s(t_x, v_{i+1}) = 1)$

The path constraint describes a sequence of nodes forming a route between v_s and v_d . It ensures that there is an edge $(e_{i,i+1})$ between all consecutive nodes $(v_i$ and $v_{i+1})$ in the path $v_s \rightsquigarrow v_d$. The node state constraint guarantees that a transmission from v_i to v_{i+1} at time t_x occurs only if both v_i and v_{i+1} are active.

2.5.2.2 Dissemination Stage Statement

In the Dissemination stage, the spatial query processing mechanism must disseminate the query from the Coordinator to all nodes inside the RoI, consuming the least amount of energy. We can formally define this problem as follows:

Dissemination Problem. Given the graph of the network $G = (V, E, s, w)$, $V' \in V$ a subset of nodes representing the nodes inside the RoI, a subset of edges $E' \subset E$, such that $E' = \{e_{i,j} | (e_{i,j} \in E) \wedge (v_i \in V') \wedge (v_j \in V')\}$ representing the edges connecting the nodes of V' , a period of time $T_d = \{t_0, t_1, \dots, t_{end}\}$ in which the dissemination occurs, U_x representing the set of nodes that transmitted the query at time $t_x \in T_f$, and a node $v_c \in V'$ representing the Coordinator, which is the only node that holds the query at time t_0 , find a subset of vertices $D \subset V'$ that minimize the

energy consumption, as described by the following equation:

$$\arg \min_D \sum_{v_i \in D} w(t_x, v_i) \quad (2.4)$$

The subset D should satisfy the following constraints:

1. Coordinator constraint: $(v_c \in D) \wedge (U_0 = v_c)$;
2. Coverage constraint: $V' = \bigcup_{v_i \in D} Ni, x$;
3. Node state constraint: a node v_i transmits a packet in broadcast at time t_x if $s(t_x, v_i) = 1$.
4. Dissemination order constraint: $(\forall v_i \in U_x) \exists (v_j \in U_y | (t_y < t_x) \wedge (v_i \in N_{j,y}))$, $x = 1, 2, \dots, end$

The coordinator constraint guarantees that the coordinator transmits the query at the beginning of the dissemination. The coverage constraint states that all nodes in the RoI must receive the query. The node state constraint guarantees that only active nodes transmit the query. The dissemination order constraint guarantees that all nodes that transmit the query have received it before.

2.5.2.3 Aggregation Stage Statement

In the Aggregation stage, the spatial query processing mechanism must forward the data collected by all nodes inside the RoI to the Aggregator, which calculates the query result. During this forwarding, the data are aggregated node by node. The application of the sensor network defines which node is the Aggregator. It is not necessarily be the sink or even a node inside the RoI. Hence, the data aggregation can be performed by any node in the WSN.

The problem that the spatial query processing mechanism must solve in the Aggregation stage is to reduce the number of transmissions necessary to deliver all the collected readings to the Aggregator. It is important to mention that the aggregation functions considered here receive two or more data packets and generate just one data packet with the summarization of the received data. We can formally define this problem as follows:

Aggregation Problem. Given the graph of the network $G = (V, E, s, w)$, $V' \in V$ a subset of nodes representing the nodes inside the RoI, $v_a \in V$ representing the Aggregator node, a period of time $T_a = \{t_s, t_{s+1}, \dots, t_{end}\}$ where $T_a \subset T$, t_s and t_{end} are

the beginning and the end of the aggregation, respectively, $S_{t_y} \subset V$ a set of vertices representing all nodes that hold their readings or readings from their neighbors at time t_y (assuming that nodes which have transmitted the reading do not hold it anymore), U_x representing the set of nodes that transmitted readings at time $t_x \in T_a$, $t_{i \rightarrow j, x}$ stating that node $v_i \in U_x$ sent readings to v_j , find a sequence $W = \{U_s, U_{s+1}, \dots, U_{end}\}$ that minimizes the energy consumption, as stated in the following equation:

$$\arg \min_W \sum_{U_x \in W} \sum_{v_y \in U_x} w(t_x, v_y) \quad (2.5)$$

The sequence W should satisfy the following constraints:

1. Beginning constraint: $S_{t_s} = V'$;
2. Finish constraint: $S_{t_{end}} = v_a$;
3. Reading transmission constraint:
 $(v_i \in U_x) \text{ if } (v_i \in S_{t_x}) \wedge (v_i \notin S_{t_{x+1}}) \wedge (\exists v_y \in S_{t_{x+1}} | v_y \in N_{i,x});$
4. Aggregation order constraint:
 $(v_i \in U_x) \text{ if } (\exists v_j \in U_y | (t_y < t_x) \wedge (v_i \in N_{j,y})) \vee ((\forall t_y \in T_a) \wedge (v_j \in V) \nexists t_{j \rightarrow i, y})$

The beginning constraint states that in the start of the aggregation (t_s) each node only holds the reading collected by itself. The finish constraint states that at the end of the aggregation (t_{end}), only the Aggregator node holds readings. The reading transmission constraint guarantees that when node v_i transmits its readings to a neighbor node, v_i does not hold the reading anymore. The Aggregation order constraint ensures that a node transmits a reading if it has received readings before or if through all the aggregation stage this node does not receive readings (it is a leaf node).

2.6 Algorithms for the Spatial Query Stages

This section presents the most important algorithms found in the literature that solve the problems of each spatial query processing stages previous described. For each stage we analyze a set of algorithms. Then, we describe algorithms that create spatial distributed indexes in order to process queries. We present these algorithms separately because they do not divide the Forwarding/Dissemination and the Aggregation/Return stages. Finally, we present a diagram that summarizes when each algorithm is performed.

2.6.1 Pre-Processing Stage

In the Pre-Processing stage, queries are optimized and formatted, in order to improve the network operation and reduce the energy consumption. This processing is performed in the user's computer because this computer usually has more resources than the sensor nodes. Normally, queries are written in a declarative language based on SQL [Madden et al., 2005]. Hence, queries must be transformed into a sequence of bytes to be transmitted to the nodes. Moreover, packets transmitted by traditional WSN using the 802.15.4 protocol usually have at most 88 bytes [IEEE, 2006]. This sequence of bytes must be as small as possible, since a big sequence may require several packets to be transmitted and, consequently, would consume so much energy during its processing.

Algorithms for the Pre-Processing stage depend on the way the RoI is represented in the query. This work considers RoIs represented by polygons, which are sequences of geographical coordinates. However, queries can be represented in different ways, such as fractals or curves [Goussevskaia et al., 2005]. The only requirement of these representations is that nodes have to be able to calculate their distance from the RoI. Hence, the Pre-Processing algorithms have to provide the smallest RoI representation as possible considering the RoI representation in the query.

To the best of the authors' knowledge, none of the current spatial query mechanisms considers this stage. Chapter 3 proposes an algorithm for this stage. The proposal reduces the amount of packets used by a query in order to reduce the energy consumption in query processing.

2.6.2 Forwarding Stage

In this stage, nodes forward the query from the **Originator** (the first node to receive the query in the network) to the RoI. This is the main difference between conventional query processing and spatial query processing in WSN. In conventional query processing, Flooding disseminates the query to all nodes in the WSN [Deshpande et al., 2007]. In spatial query processing, nodes first forward the query and then disseminate it only to the nodes within the RoI. This strategy reduces the number of nodes participating in query processing.

The task of the Forwarding stage can be performed by adapting one of the routing protocols found in the literature [Jin et al., 2009]. However, there is a difference between packet routing and query forwarding. Packet routing protocols establish routes between two predefined nodes or points. In query forwarding, the end-point is not known a priori. Thus, a route is built between a specific node (the **Originator**) and any node

inside the RoI. The Dissemination algorithm defines the last node in the Forwarding stage.

Another important characteristic to be considered in Forwarding is node location. Spatial query processing mechanisms assume that nodes know their location. Location-based routing protocols use this information to create their routes, consuming less energy than the protocols that do not employ location information [Jin et al., 2009]. Thus, these protocols are usually the best choice for the Forwarding stage. Furthermore, each protocol is adapted to a specific network mobility pattern. Hence, the spatial query processing mechanisms must choose the routing protocol suited to the target network.

The spatial query processing mechanisms found in the literature are based on the Greedy and GPSR location-based protocols. However, Felice et al. [2008] and Xu et al. [2007] presented spatial query processing mechanisms that use Flooding to forward the query to the RoI. In these mechanisms, all the nodes receive the query, but just those within the RoI process it. The following subsection analyzes the advantages and disadvantages of Flooding to forward spatial queries. Then, we present an overview of the location-based routing protocols used in spatial query processing.

2.6.2.1 Flooding in WSN

Flooding is an algorithm to disseminates data to all nodes in a network. When a node receives a packet, it verifies if this packet has been sent before. If not, it retransmits the packet. If so, the packet is dropped. Flooding is used for spatial query processing mechanisms, such as described in [Felice et al., 2008; Xu et al., 2007]. Furthermore, Flooding is compared to the algorithms proposed in [Xu et al., 2006; Coman et al., 2007]. Its advantages are: (1) if there is a path between the Originator and the RoI, the query will always reach it; (2) it does not demand many computational resources; (3) the query packet converges quickly to the region of interest; (4) the Forward and Dissemination stages are performed together, and; (5) it builds a routing tree that can be used to return the query result.

The disadvantages are: (1) it consumes too much energy, because nodes can receive the same query more than once (from all their neighbors); (2) nodes outside the RoI receive and retransmit the query unnecessarily, and; (3) the query reaches the RoI through more than one node, thus there is more than one route to send back the query result. It impairs the aggregation process, since an efficient aggregation is performed only inside the RoI, according to [Li et al., 2008].

In [Coman et al., 2007], the authors compared two strategies to process spatial

queries. In the first, they used Flooding to forward the queries. In the second, they used the Greedy protocol (Subsection 2.6.2.2) for forwarding and Restricted Flooding for dissemination (Subsection 2.6.3.1). The experiments showed that Flooding consumed more energy than the second strategy in all the analyzed scenarios. Hence, Flooding should be avoided in spatial query processing mechanisms in order to save energy.

2.6.2.2 Greedy Protocol

The Greedy Protocol is the most simple and intuitive location-based protocol. Nodes forward the packet to the neighbor closest to the destination (in spatial query processing the destination is the RoI). This strategy is very simple and requires few computational resources. However, this protocol depends on a neighbor table, which contains the location of all neighbors of a node. This table must be periodically updated, which can consume a substantial part of the network energy supply, mainly in WSN with failure-prone nodes or using duty cycle algorithms. The spatial query processing mechanism described in [Coman et al., 2007] uses the Greedy protocol in the Forwarding stage.

The Greedy protocol, however, does not guarantee packet delivery. If a node has no neighbor closer to the RoI than itself, the query will be dropped. This problem is called as **hole in the network** [Zhao and Guibas, 2004] and is illustrated in Figure 2.4 (a). The query reached a node that is the closest node to the RoI and it does not have any neighbor closer than itself or inside this region. Hence, the query will be dropped.

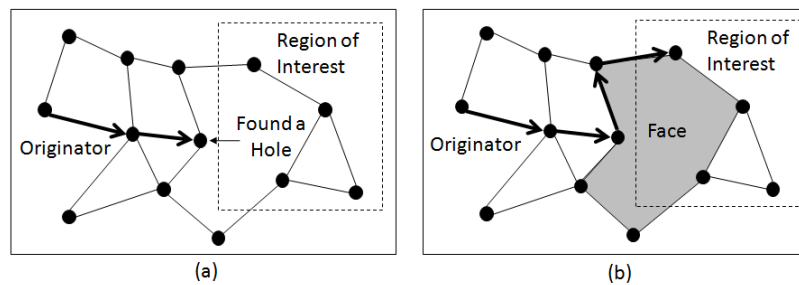


Figure 2.4. (a) Greedy: hole in the network. (b) GPSR: the query bypasses the hole.

2.6.2.3 GPSR - Greedy Perimeter Stateless Routing

GPSR (Greedy Perimeter Stateless Routing) is one of the most referenced location-based routing protocols [Karp and Kung, 2000]. It forwards the packets like the Greedy protocol. However, packets are not dropped if they reach a hole in the network. GPSR constructs a planar graph using neighborhood information [Zhao and Guibas, 2004].

Packets near a hole are transmitted over the contour of the faces of the planar graph. Figure 2.4 (b) illustrates a query contouring a face to reach the RoI. Hence, this protocol delivers more packets than the Greedy protocol and presents the same energy consumption. However, it also depends on neighbor tables, and thus consumes energy to update them. The spatial query processing mechanisms described in [Wu et al., 2007; Demirbas and Lu, 2007; Fu et al., 2010; Xu et al., 2006] use GPSR to forward queries.

2.6.2.4 Analyzing the Location-Based Routing Protocols

The analyzed works on spatial query processing showed that Flooding can be used in static or mobile networks, but it has a higher energy consumption. Greedy and GPSR protocols are good options, since they do not require much computational resources and consume little energy. Moreover, GPSR has more advantages since this protocol can avoid holes in WSN. However, both depend on the neighbor tables, whose maintenance may consume too much energy, mainly in failure-prone networks since the topology can change constantly.

2.6.3 Dissemination Stage

In the Dissemination stage, the query is disseminated to all nodes within the RoI. We found three protocols in the literature of spatial query processing in WSN: Restricted Flooding, itinerary-based and WinDeph. Other protocols for dissemination can be found, such as TBF [Niculescu and Nath, 2003] and TEDD [Goussevskaja et al., 2005]. However we focus on dissemination protocols used by spatial query processing mechanisms. In the specific case of TBF and TEDD, to use these protocols in the dissemination stage would consume so much energy, since all nodes would know the energy map of the sensor network.

It is important to mention that the Aggregation stage depends on the manner that the dissemination is performed. In some cases, these stages are performed by the same algorithm, such as in itinerary-based protocols [Xu et al., 2006]. Here we present an overview of dissemination protocols for spatial query processing.

2.6.3.1 Restricted Flooding

Restricted Flooding is an algorithm that performs flooding inside the RoI. The mechanisms described in [Xu et al., 2007; Coman et al., 2007] use this algorithm. When a node A receives a query, it verifies if it is inside the RoI. If true, it broadcasts the

query to its neighbors, which save A as their parent node. Nodes outside the RoI drop the query. This algorithm creates a routing tree inside the RoI with the Coordinator as the root. The Aggregation stage uses this tree to calculate the query result.

A drawback of this algorithm is the energy consumption. The network consumes a substantial part of its energy with unnecessary query transmissions and receptions. It is because the nodes close to the region of interest also receive the query from the nodes inside this region. Moreover, each node receives the query from all its neighbors during the dissemination.

2.6.3.2 WinDepth

WinDepth is a algorithm proposed by Coman et al. [2007]. It disseminates the query node by node until all nodes have received the query. When a node receives a query to disseminate, it adds its node identifier in the query header in order to create a return path for the query result. Then, it selects a neighbor located inside the RoI that has not received the query yet and sends it to this neighbor. When this neighbor returns the partial query result, the node verifies again if other neighbors are also inside the RoI and have not yet received the query. If such neighbor exists, the node sends the query to this neighbor and waits for its answer. This process is repeated until all the node's neighbors located inside the RoI have answered the query. When all neighbors send their data, the node merges this information with the information collected by itself and sends the partial query result to the neighbor that sent the query to it.

2.6.3.3 Itinerary-based Algorithms

An itinerary is a sequence of nodes within the RoI that forms a path over which nodes forward the queries. Each node that receives a query follows these steps: asks its neighbors to sense environmental data, senses its own data, receives the information from its neighbors and retransmits the query and the partial query result to the next node in the itinerary. When the query reaches the last node in the itinerary, the result is forwarded to the Originator.

The main challenge of those methods is how to find the best itinerary, which depends on the density and topology of the network. The quality of the data collected by itinerary-based protocols can decrease if queries reach holes in the network. In this case, queries cannot continue the itinerary. Thus, the query result, calculated until the moment that the query reaches the hole, needs to be sent to the Originator. Consequently, fewer nodes participate in query processing, reducing the quality of the data collected.

Xu et al. [2006] proposed a mechanism to process spatial window queries called IWQE (Itinerary-base Window Query Execution). Figure 2.5 shows an example of an itinerary created by this mechanism. IWQE tries to avoid holes in the network as follows. If the query reaches a hole, the query propagation switches to a perimeter forwarding mode on the progressively closer face of the planar graph, like in GPSR. However, IWQE does not guarantee that the query always reaches the end of the itinerary. If the query in the face of the graph reaches a node outside the RoI, the process stops and the partial query result is sent to the Originator.

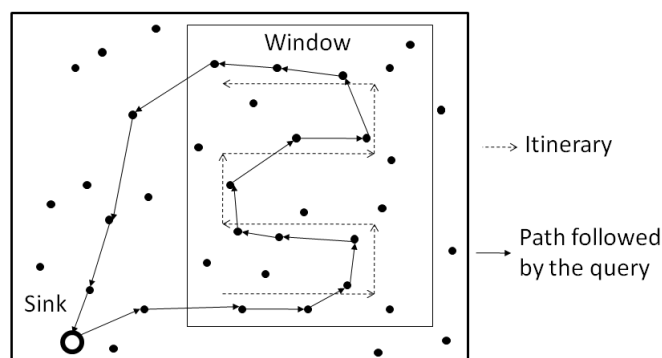


Figure 2.5. Window query processing based on itinerary.

Dissemination by itinerary is mostly used in KNN query processing because the itinerary facilitates counting the number of nodes that processed the query. We found three mechanisms that process KNN queries and use itineraries: IKNN [Xu et al., 2007], DIKNN [Wu et al., 2007] and PCIKNN [Fu et al., 2010]. The mechanisms based on itinerary differ from each other mainly in the format of the itineraries. DIKNN divides the ROI into cone-shaped areas centralized at the query point. An itinerary is created in each area. Hence, the query process occurs in a parallel fashion. Figure 2.6(a) presents the division of the ROI and the itineraries. PCIKNN also divides the region in cone-shapes and creates several itineraries. However, the itineraries have a different format, as shown in Figure 2.6(b).

IKNN proposed two formats for the itinerary. The first is called Spiral Itinerary. The query dissemination starts in the node closest to the query point, following an itinerary in the form of a spiral, as shown in Figure 2.6 (c). Queries have a field that stores the number of nodes that answered the query. When this number reaches K , the query dissemination stops and the result is sent back to the Originator. The second format is called Parallel Itinerary. The dissemination starts in the same node, but queries follow two itineraries, shown in Figure 2.6 (d). The query also stores the number of nodes that answered the query. Neighbor nodes in different itineraries sum

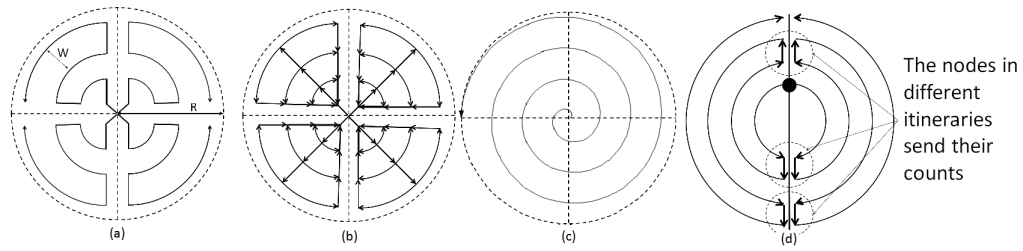


Figure 2.6. (a)Itineraries created by DIKNN. (b)Itineraries created by PCIKNN. (c)Spiral itineraries created by IKNN mechanism. (d)Parallel itineraries created by IKNN mechanism.

their counts to calculate the total number of nodes that answered the query so far. The parts of the itineraries where these nodes send their count to each other are shown in Figure 2.6 (d).

In those works, the RoI is circular, because it facilitates the identification of the nodes closest to the query point. The range of the region of interest is calculated based on the WSN density. During the Forwarding stage, each node puts in the query packet its number of neighbors. The coordinator uses this information to define the region of interest that probably contains K nodes.

2.6.3.4 Analyzing the Dissemination Protocols

In window and KNN query processing, the itinerary-based techniques present the best results. In [Xu et al., 2006], a window query mechanism that uses an itinerary-based algorithm to disseminate the query was compared with three other mechanisms: Restricted Flooding, MBR (Subsection 2.6.7.1) and Flooding. In most of the experiments, the mechanism with itinerary-based dissemination presented the smallest energy consumption and query latency. Moreover, the query result accuracy was similar to the other analyzed mechanisms. In [Xu et al., 2007] the authors compared three KNN query processing mechanisms. They used itinerary-based, Restricted Flooding and Flooding to disseminate the query. The itinerary-based mechanisms also presented the smallest energy consumption, the smallest query latency and similar query accuracy. However, all the analyzed works consider RoIs having rectangular or circular shapes. To our knowledge, none of the works about spatial queries in WSN define itineraries in RoIs having polygonal shapes.

2.6.4 Sensing Stage

In the Sensing stage, the nodes within the RoI collect the data required by the query. Some applications require a coarse spatial granularity of the data [Gao et al., 2007]. Since nodes close to each other tend to collect similar data, it is possible to devise strategies to decrease the number of nodes that sense the environment without compromising the quality of the reported data [Zhou et al., 2008]. However, none of the spatial query mechanisms found in the literature define energy-efficient algorithms to the Sensing stages. In this work, we consider that nodes only collect environmental data, hence we do not define algorithms to optimize the network operation in the Sensing stage.

2.6.5 Aggregation Stage

In the Aggregation stage, nodes inside the RoI transmit the collected data to a node called **Aggregator**, which calculates the query result. During the transmissions, each node aggregates the data received from its descendents with its own data and forwards the result. This technique is called **in-network aggregation** (or just aggregation). The main idea behind in-network aggregation is that, rather sending individual data items from sensor nodes to the Aggregator, multiple data items aggregated as they are forwarded by the WSN [Solis and Obraczka, 2006]. The works on spatial query processing consider five aggregation operators: *avg*, *count*, *sum*, *max* and *min*, however other methods could also be employed in this stage [Nakamura et al., 2007]. Generally, the Aggregator is the sink node. However, in this work the definition of the Aggregator depends on the Dissemination stage. For Restricted Flooding this node is the Coordinator, for itinerary this node is the last node on the itinerary.

In a naive strategy to aggregate, all nodes within the RoI would send their data to the Aggregator using multi-hop routes. Nodes next to the Coordinator consume more energy than the others, because all the data passes through them. Figure 2.7 illustrates such naive strategy, showing that these nodes send more packets than the others. Employing in-network aggregation, the sensor nodes combine the sensing data during the propagation, reducing the number of transmitted packets and saving energy.

When Flooding is used to disseminate queries to all nodes in the network, the aggregation process can transmit more packets than necessary. In this case, part of the aggregation is performed outside of the RoI, as shown in Figure 2.8(a). Nodes inside the RoI collect data. Node 5 finished the aggregation process and the WSN transmitted nine packets. In Figure 2.8(b) the aggregation process is performed only inside the RoI. In this example, the WSN transmitted only seven packets. The mechanism described

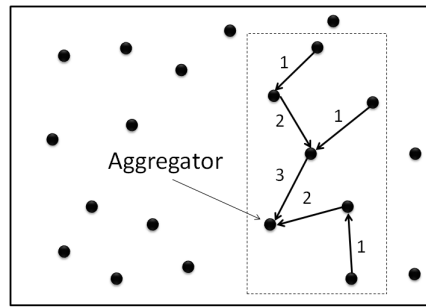


Figure 2.7. Number of packets transmitted during the Aggregation stage using a naive strategy.

in [Felice et al., 2008] disseminates the queries by Flooding and aggregates as in Figure 2.8(a).

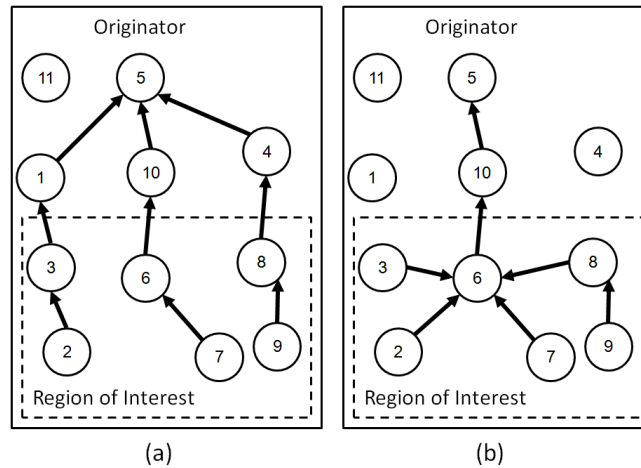


Figure 2.8. (a) Aggregation performed outside the region of interest. (b) Aggregation only inside the region of interest.

The mechanisms that disseminate the query by itineraries perform the dissemination and the aggregation together. Nodes that receive the query on the itinerary ask their neighbors to sense and to transmit the sensed data. Then, it receives this data, merges it with its own data, applies the aggregation operator and forwards the partial result and the query to the next node on the itinerary. Hence, the last node on the itinerary can calculate the query answer. The mechanisms described in [Fu et al., 2010; Xu et al., 2006; Wu et al., 2007; Xu et al., 2007] use itineraries.

2.6.5.1 Analyzing the Aggregation Strategies

In-network aggregation strategies substantially reduce the energy consumption during the Aggregation stage. However, all the aggregation strategies consume more energy

in failure-prone or mobile WSN, as shown in [Xu et al., 2006, 2007]. The trees created by Flooding or Restricted Flooding need to be maintained in order to guarantee the delivery of the partial query results to the Aggregator. Hence, nodes continuously verify if the communication with their parents is possible. If not, they look for another parent. When the aggregation occurs jointly with the dissemination (as in protocols based on itineraries), node failure requires less link verification because nodes need to establish the next node on the itinerary once. Hence, spatial query mechanisms that use itineraries to disseminate and aggregate the collected data in failure-prone WSN tend to consume less energy than mechanisms based on routing trees to aggregate data.

2.6.6 Return Stage

In the Return stage, the Aggregator sends the query result back to the Originator. The spatial query processing mechanisms use either the same route created during the Forwarding stage or the same protocol used in this stage. However, in failure-prone WSN or in networks having duty cycle algorithms the route used in the Forwarding stage may not be available to forward back the query result since the nodes may be unavailable. Hence, the Return stage must deal with these questions.

The majority of the works on spatial query processing use GPSR in this stage, such as the mechanisms described in [Xu et al., 2007, 2006; Fu et al., 2010; Demirbas and Lu, 2007; Wu et al., 2007]. The mechanisms that use the same route created in the Forwarding stage, but in the opposite direction, are described in [Coman et al., 2007; Felice et al., 2008; Xu et al., 2007]. It is important to mention that these mechanisms consider that the Originator is static. Thus, the query results are always sent to the point where the queries were issued.

2.6.6.1 Analyzing the Strategies for the Return Stage.

In window query processing, the use of the opposite route created in the Forwarding stage to return the query result can be inefficient due to node failures and duty cycle. To use this route, the spatial query mechanism needs to transmit periodic beacons in order to maintain all node links. This can cause substantial energy consumption. Moreover, this strategy can be used only when Restricted Flooding is used to disseminate the query, since the first node to receive the query inside the RoI (Coordinator) is also the node that will forward back the query result (Aggregator). When itineraries are created to disseminate queries, the Aggregator is the last node in the itinerary. Thus, the opposite route cannot be used. An alternative is to define another route, which can be performed by the same algorithm used in the Forwarding stage.

2.6.7 Distributed Spatial Indexes

Traditional spatial databases use spatial indexing techniques to improve spatial query processing. These indexes are created in a centralized manner. In WSN, however, queries are processed in a distributed manner. Moreover, WSN topologies can change over time, i.e. nodes can fail or sleep to save energy. Hence, traditional indexes cannot be applied to WSN in an energy-efficient manner since several control messages would be required in order to update the index [Kalogeraki and Soheili, 2008].

Some works define location-based distributed spatial indexes for spatial query processing in WSN in order to reduce the number of nodes that process spatial queries. The data that constitute the indexes are distributed over the nodes. The index relieves the WSN of forwarding and disseminating queries to the nodes inside the RoI. In the following, we present an overview of spatial indexes in WSN for spatial query processing. This review is not presented in stages because the Forwarding/Dissemination and Aggregation/Return stages are performed by the same algorithms.

2.6.7.1 GRT - Georouting Tree

Nodes in a routing tree in WSN save the identifier of the node immediately above them in the tree (its parent). Hence, when a node needs to send a message to the sink, it sends this message to its parent, which will send the message to its parent too. This process is repeated until the root is reached. In GRT [Goldin et al., 2003], each node saves who is its parent as well as a bounding box that covers itself and all its children (nodes below it in the tree). This information constitutes a spatial index that helps the forwarding and dissemination of the queries from the sink to the nodes within the RoI.

The bounding box information helps to save energy during query dissemination. When a node receives a query, it verifies if the RoI overlaps its bounding box. If so, it transmits the query to its children. If not, it drops the query. Hence, this strategy avoids sending the queries to all nodes in the WSN. Figure 2.9 illustrates GRT. Only the black nodes transmit the query.

2.6.7.2 SPIX - Spatial Index

SPIX is a mechanism to process window queries [Soheili et al., 2005]. It defines a spatial index that is used to locate sensor nodes that have relevant data. This index is created over a routing tree having the sink as the root. SPIX stores a two-dimensional minimum bounding area (**MBA**) in each sensor node, which covers the node and the

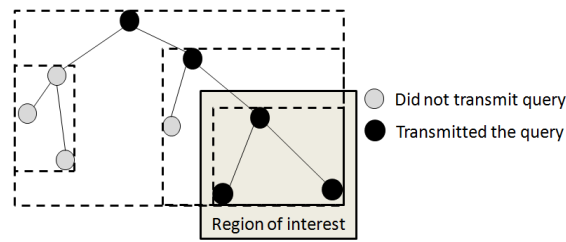


Figure 2.9. Query dissemination with Georouting Tree.

MBAs of its children in the routing tree. When a node receives a query, it verifies if the RoI overlaps each child’s MBA. If so, it prepares to receive the result and forwards the query. If not, the query is not forwarded. Also, if the node is within the RoI, it will execute the query. If not, the query will be dropped. Hence, the same technique performs the Forwarding and Dissemination stages.

SPIX exploits two models for creating the MBA, called rectangular model and angular model. In the **rectangular model**, the MBA is the minimum bounding rectangle (MBR) that covers the node and all nodes below it (similar to the GRT). In the **Angular model**, the MBA is the minimum bounding pie slice represented by a start/end radius and start/end angles. SPIX tries to minimize the MBA, because nodes with smaller MBA have fewer chances to forward or answer a query. An evaluation using simulations showed that the best performance was reached by the rectangular model.

After the index construction, SPIX performs an “energy optimization phase” in order to minimize the nodes’ MBAs. In this phase, if a node realizes that it is located in another node’s MBA (different of its parent), it runs a “parent-switching verification” process. It asks its parent: “What would be your MBA if I left you?” If the current parent’s MBA will decrease and new parent’s MBA will not change, this node switches to the new parent and notifies its old parent. Figure 2.10 presents an example of a node that switches its parent to decrease the old parent’s MBA.

2.6.7.3 Cluster and Index

Park et al. [2007] described a mechanism for window query processing that creates a semi-distributed spatial indexing structure to disseminate a query into the network and to retrieve data using a localized tree building algorithm. The mechanism divides the WSN into square sub-regions. Nodes in the same square constitute a cluster. The cluster-head is the node closest to the top right corner of each sub-region. Then, each cluster-head receives the location of all nodes in its cluster. Thus, the cluster-

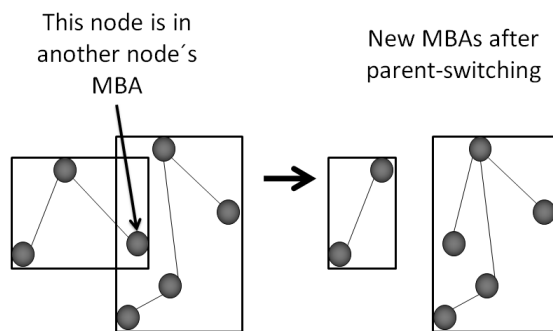


Figure 2.10. Parent-switching verification phase.

head knows the whole topology of its section and runs either Dijkstra's shortest path algorithm or Prim's minimum spanning tree algorithm to build their energy-efficient local tree. Figure 2.11 shows an area monitored by a WSN, their sub-regions, nodes and cluster-heads.

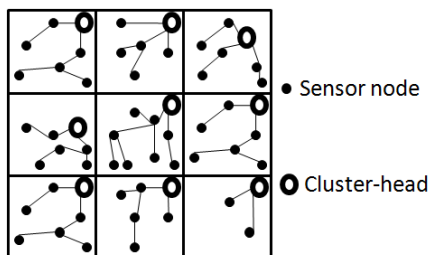


Figure 2.11. Semi-distributed index: subregions, nodes and cluster-heads.

All queries are issued from the sink, located on the top right corner of the area covered by the WSN. To disseminate queries to the cluster-heads, the mechanism uses an algorithm based on MBR, such as SPIX. After receiving a query, each cluster-head disseminates the query in its cluster, aggregates all data collected in the sub-region and sends back the query result.

This mechanism works in a centralized manner within the clusters and in a distributed manner among the clusters. This strategy allows the use of the best routes inside the cluster and avoids the high energy consumption of a totally centralized algorithm. However, this approach overloads the cluster-heads, since these nodes centralize all communication in their clusters.

2.6.7.4 Back Forwarding Method

Li et al. [2008] described a mechanism also based on the routing tree created by SPIX. In SPIX, a node A adds its MBR to the MBR of its parent's candidate. Then, node

A chooses as parent the node which will have the smallest new MBR. However, this mechanism does not guarantee the best aggregation process. In some cases, the query can reach the RoI by more than one node. Figure 2.12(a) shows an example. Thus, part of the aggregation can occur outside the RoI, increasing the amount of message transmissions.

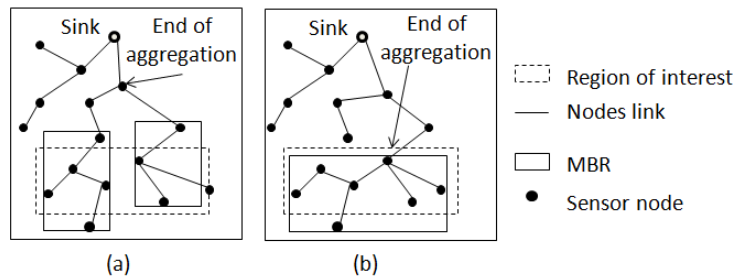


Figure 2.12. (a) MBR created by SPIX; (b) MBR for a more efficient aggregation.

In the mechanism described by Li et al. [2008], the Forwarding stage uses MBR, but in the Aggregation stage a node chooses a different parent for each query. Nodes have a list of parent candidates. They prefer a parent that is within the RoI, so the aggregation only occurs inside this region. This strategy saves a substantial part of the energy consumed in the Aggregation stage. Figure 2.12(b) shows an example of aggregation limited to the RoI.

2.6.7.5 Analyzing the Distributed Spatial Indexes

The mechanisms found in the literature on distributed spatial indexes are based on minimum bounding rectangles. Hence, they have the following drawback in query processing: queries must start only in the sink. This is because the indexes are created based on the sink's location. Moreover, the WSN consume too much energy if the network topology changes. When a node detects that a descendent is not available, it needs to inform its parent of the index change. If a node receives a message informing that its children's index changed, it will update its index and notify its parent. Hence, when a node changes its state, it can produce changes that are propagated up to the sink. So, node failures and duty cycle algorithms increase the number of update messages and, consequently, consume too much energy.

2.6.8 Spatial Query Processing Stage Remarks

As mentioned before, each stage represents a sub-problem of the spatial query processing. The previously described algorithms represent solutions for these sub-problems.

However, some algorithms solve more than one sub-problem and the use of some algorithms define the use of another algorithm in certain stages. Figure 2.13 presents a diagram that illustrates all the described algorithms and the stage(s) each algorithm implements. Moreover, this diagram also shows the algorithms that define the use of another algorithms in other stages.

In the related works, the Pre-Processing and Sensing stages represent sub-problems solved by very simple algorithms. These works do not consider the Pre-Processing algorithm since they do not present algorithms to prepare the queries before their processing in the WSN. Furthermore, in the Sensing stage nodes only collect environmental data. There is no algorithm to optimize this data collection. However, all related works present algorithms for the other states, which represent complex sub-problems that must be solved in order to process spatial queries.

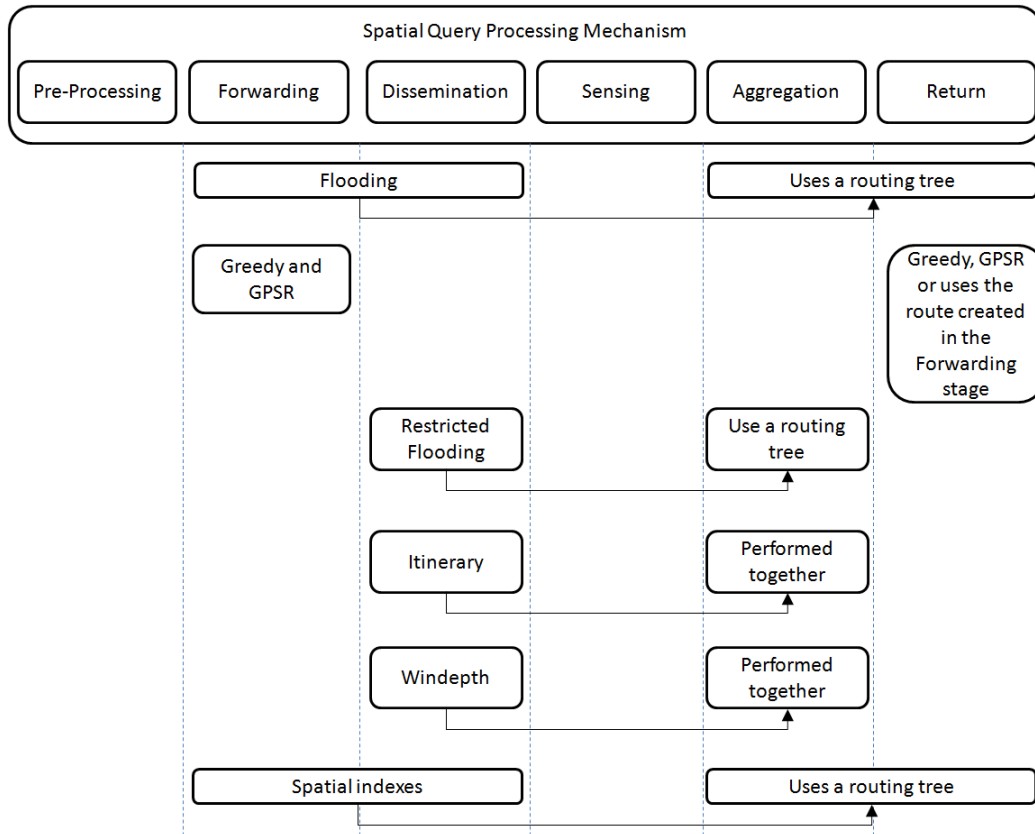


Figure 2.13. The algorithm and the stages they are performed.

Analyzing this diagram, we verify that:

- The Flooding algorithm may be used in the Forwarding and Dissemination stages. Since it creates a routing tree, WSN use such structure to perform the Aggregation and Return stages as well.

- When a spatial query processing mechanism uses Greedy or GPSR in the Forwarding stage, it creates a new route in the Return stage or uses the route previously created in the Forwarding stage.
- Restricted Flooding creates a routing tree inside the RoI, hence nodes use this structure in the Aggregation stage.
- Itinerary based algorithms disseminate queries and the partial query results to the next node in the itinerary. Thus, the Dissemination and Aggregation stages are solved by the same algorithm.
- Nodes using the Windepth algorithm disseminate the query and wait for the partial query result from each of their neighbors. Hence the Dissemination and Aggregation stages are implemented by the same algorithm.
- Spatial indexes create routing trees during the Forwarding and Dissemination stages and use this structure to implement the Aggregation and Return stages.

2.7 Spatial Query Processing Mechanisms

In this section we first classify the problem of processing spatial queries and then we present an overview of the most relevant spatial query processing mechanisms for WSN found in the literature. The criteria used in the classification are the mobility pattern of the nodes, the characteristics of the processed query and the structure and operation of the mechanisms. We describe the spatial query processing mechanisms using the stages and algorithms previously described in this chapter.

2.7.1 Spatial Query Processing Classification

We classify the spatial query processing problem into two classes: Problem Statement and Problem Solution. The criteria used in the **Problem Statement** classification are based on the scenario in which the mechanism is performed (the WSN mobility pattern and the Originator node) and the characteristics of the processed query (described in Subsection 2.1). Figure 2.14 illustrates this classification. The WSN mobility pattern describes if all WSN nodes are mobile or static and whether the Originator is mobile or static. The classification by the Originator describes if the mechanisms start the query processing only in the sink or if any node of the WSN can start processing queries. The query classification follows the structure of Section 2.1.

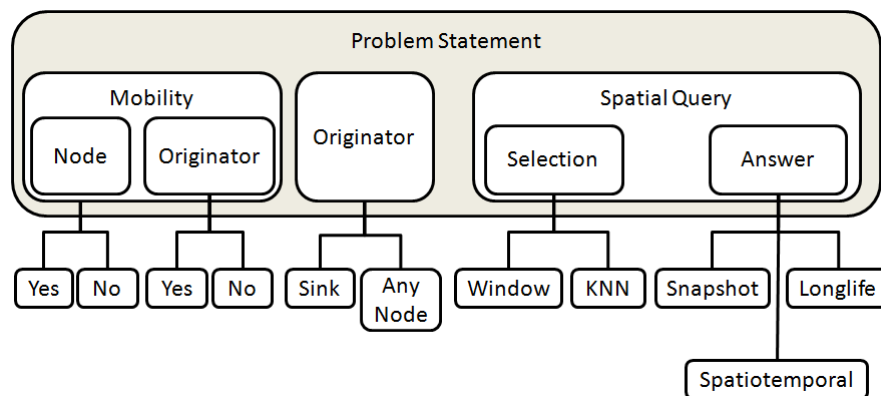


Figure 2.14. Problem Statement classification.

The Problem Statement classification considers how the mechanism processes queries. First, we classify the mechanisms according to the structure created to process queries. This criterion verifies if the mechanism creates index and/or cluster. Second, the mechanisms are classified based on the algorithms used in their stages. Figure 2.15 illustrates this classification.

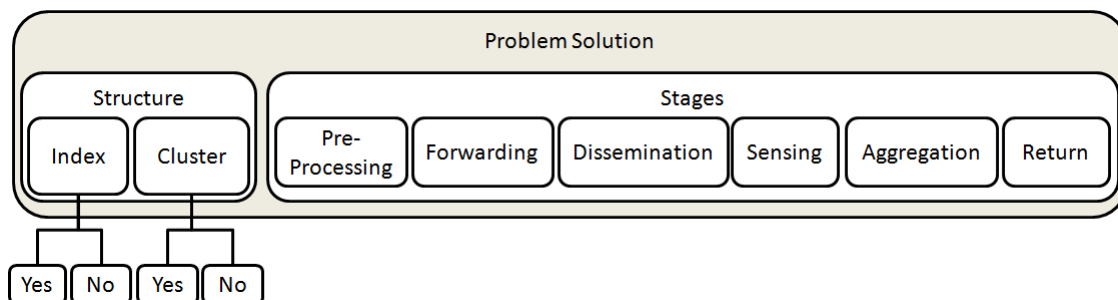


Figure 2.15. Problem Solution classification.

2.7.2 Overview of the Mechanism

Table 2.7.2 presents a summary of the mechanisms analyzed in this work. Each mechanism is classified based on the criteria described in the previous subsection. We did not include the Pre-Processing and Sensing stages in the table. Pre-Processing is one of the thesis contributions and none of the analyzed works consider either the Sensing stage or the Pre-Processing stage.

Table 2.1. Mechanisms for Spatial Query Processing

Name	Problem Statement				Originator	Problem Solution					
	Mobility		Query			Structure		Stages			
	Node	Originator	Selection	Answer		Index	Cluster	Fwd.	Diss.	Agg.	Ret.
Georouting Tree [Goldin et al., 2003]	No	No	Window	Snapshot	Sink	Yes	No	MBR	MBR	Yes	MBR
SPIX [Soheili et al., 2005]	No	No	Window	Snapshot	Sink	Yes	No	MBR	MBR	Yes	MBR
Cluster Index* [Park et al., 2007]	No	No	Window	Snapshot Longlife	Sink	Yes	Yes	MBR	MBR	Yes	MBR
Back Forwarding* [Li et al., 2008]	No	No	Window	Snapshot	Sink	Yes	No	MBR	MBR	Yes	Optimized MBR
SWIP [Coman et al., 2007]	No	Yes	Window	Spatio- temporal	Any node	No	No	Greedy/ Flooding	WinDepth	Yes	Created in Forwarding
IWQE [Xu et al., 2006]	Yes	No	Window	Snapshot	Sink	No	No	GPSR	Itinerary	Yes	GPSR
SWOP [Jin and Nirtel, 2008]	No	No	Window	Snapshot	Sink	No	Yes	N/A	N/A	No	N/A
Extension of TinyDB* [Felice et al., 2008]	No	No	Window	Spatio- temporal	Sink	No	No	Flooding	Flooding	Yes	Created by Flooding
MobiQuery [Lu et al., 2005]	Yes	Yes	Window	Spatio- temporal	Sink	No	No	No	Restricted	Yes	No
GRT [Xu et al., 2007]	Yes	No	KNN	Snapshot	Any node	No	No	Flooding	Flooding	Yes	Created by Flooding
KBK [Xu et al., 2007]	Yes	No	KNN	Snapshot	Any node	No	No	GPSR	Restricted	Yes	GPSR
IKNN [Xu et al., 2007]	Yes	No	KNN	Snapshot	Any node	No	No	GPSR	Itinerary	Yes	GPSR
DIKNN [Wu et al., 2007]	Yes	No	KNN	Snapshot	Sink	No	No	GPSR	Itinerary	Yes	GPSR
PCIKNN [Fu et al., 2010]	Yes	Yes	KNN	Snapshot	Any node	No	No	GPSR	Itinerary	Yes	GPSR

*The original paper does not propose a name, this name is based on the mechanism's characteristics.

2.7.3 Window Query Processing

The mechanisms analyzed here are divided into structured and unstructured. The structured mechanisms create indexes or clusters to help processing the queries. We do not consider as structured the mechanisms that create routing trees, because nodes only need to have information about their neighbor, which can be obtained during normal WSN operation. In the unstructured mechanisms, nodes only have their neighbors' information, no global structure is defined in the WSN. Table 2.7.2 presents the citations for the following spatial query mechanisms.

2.7.3.1 Structured Mechanisms

Georouting Tree: creates a global structure based MBR. The drawbacks of this mechanism, and of the others based on MBR, is that queries only start in the sink and node failures or nodes sleeping can increase the amount of control messages, increasing the energy consumption. Moreover, this mechanism does not optimize the index, such as more recent mechanisms.

SPIX - Spatial Index: the main contribution of this work is the definition of the optimization phase that decreases the nodes' MBR. Nodes having small MBRs decrease their probability to participate in spatial query processing. Hence, less nodes transmit and receive queries, reducing the energy consumption. This mechanism is the most referenced distributed index in the literature.

Cluster and Index: the main contribution of this work is the definition of a semi-distributed spatial index. This algorithm performs in a centralized manner inside the clusters (it can find the best routes within a cluster) and in a distributed manner among clusters (avoids all nodes having to send their information to the sink). The main drawback is the energy consumption to create and maintain all the cluster information in the cluster-head.

2.7.3.2 Unstructured Mechanisms

SWIP - Spatial Window Processing: the main contribution of this work is the study of Forwarding and Dissemination algorithms. It compares Flooding against mechanisms that first forward the query to the RoI and then disseminate it. In the Dissemination stages it analyzes two algorithms: Windepth and Restricted Flooding. The drawback of this work is that it does not consider overhearing in nodes' transmissions. Moreover, it uses a greedy protocol in the Forwarding stage, which does not

guarantee the query delivery and increases the energy consumption, since it needs to periodically refresh the neighbor tables in failure-prone WSN.

IWQE - Itinerary-based Window Query Execution: the mechanism is presented as an “structure-free window query processing technique” based on itineraries [Xu et al., 2006]. However, only the Dissemination and Aggregation stages are structure-free, since the mechanism uses GPSR in the Forwarding stage, which creates neighbor tables to forward queries. Moreover, the RoIs analyzed have rectangular shapes that simplify the itinerary definition. A contribution of this work is the performance analysis of some itinerary shapes for query dissemination.

SWOP - Spatial Window Query Over Phenomena: is a spatial window query processing mechanism that manages the sensing of spatial continuous phenomena in WSN. A spatial continuous phenomenon is a spatial scalar field that represents the variation of a scalar property over a geographical space, such as temperature or humidity. The result query is the representation of a digital surface having a value for each point. The mechanism calculates each of these points performing interpolation in the data collected by the sensor nodes. The main contribution of SWOP is the use of statistical methods to reduce the number of nodes necessary to calculate the digital surface. A drawback of SWOP is that it assumes that the ranges of the sensor node radios are bigger than the cluster radius. Thus, there is no multi-hop route in a cluster.

Spatial Extension of TinyDB: TinyDB [Madden et al., 2005] is a query processing system for extracting information from WSN. It allows to see the network as a relational database and provides a simple SQL-like interface, called TinySQL. Felice et al. [2008] created an extension that enables TinyDB to process window queries. The main contribution was to show the viability and the energy saved with the creation of a spatial extension for TinyDB. In this work, queries are disseminated to all nodes in the network (Flooding), as is performed by TinyDB in every query processing. Nodes within the window periodically collect and send data to the sink. This strategy was evaluated (by simulations) against a naive strategy, in which all nodes send their data to the sink to be saved in a database (PostGIS [Ramsey, 2001]). In this naive strategy, spatial queries are not processed in-network. The experiments show that the ameliorated TinyDB has substantial energy savings, because the number of nodes that sense and transmit data tends to be smaller. A drawback of this protocol is the use of Flooding to disseminate the queries, because nodes outside the RoI also receive/transmit the query, increasing the energy consumption.

MobiQuery: the motivation to create MobiQuery [Lu et al., 2005] was mission-critical applications in which mobile users (such as firemen) need to continuously gather real-time information from WSN within their vicinities. The authors consider static

sensor nodes and a mobile Originator. The path followed by the Originator is known in advance. Hence, queries are forwarded to the future position of the Originator and disseminated to the nodes inside a predefined range. When the Originator reaches this position, the nodes send to it their sensed data. The main contribution of this mechanism is the dissemination in two stages. Nodes can switch to sleep mode, hence a query is first disseminated to the awake nodes and when the other nodes wake up, the query is disseminated to them. A drawback is the necessity of previously knowing the Originator's path.

2.7.3.3 Analyzing the Window Query Processing Mechanisms

Xu et al. [2006] presented a good analysis of window query mechanisms. The authors used three metrics (energy consumption, latency and query accuracy) to compare four mechanisms: Flooding (Subsection 2.6.2.1), GRT (Subsection 2.6.7.1), WSI (Window Spanning Infrastructure - forwards using GPSR and disseminates using Restricted Flooding) and the proposed IWQE (creates itineraries). Flooding presented good performance, but its energy consumption was the highest. GRT presented the worst performance. The experiments showed that GRT is not suitable for mobile networks. WSI presented an average performance, except in extreme conditions (high query load or large window area). These experiments showed that itinerary-based mechanisms, such as IWQE, are the most promising for window query. They can be used in static or mobile WSN, since this mechanism does not maintain neighbor tables and obtains the next node on the itinerary on the fly. IWQE presented low energy consumption, but its accuracy depends on the WSN topology. During the experiments, the query processing was not analyzed neither in failure-prone WSN nor in networks using duty cycle algorithms.

2.7.4 KNN Query Processing

All mechanisms found in the literature that process KNN queries are unstructured. Despite the fact that KNN query processing is not the focus of this work, the mechanisms presented here are similar to the window query processing mechanisms previously described, since KNN mechanisms present the same stages defined in Section 2.5. The analyzed mechanisms follow:

GRT, KBT and IKNN: Xu et al. [2007] proposed and evaluated these three KNN processing mechanisms. The main contribution of this work is to evaluate different strategies to process KNN queries. The first proposed a mechanism called GRT (GeoRouting Tree) which employs Flooding to disseminate the queries. The second,

called KBT (KNN Bounded Tree), uses GPSR to forward the queries and Restricted Flooding to disseminate. The third, called IKNN, uses GPSR to forward queries and two different itineraries to disseminate queries. The experiments showed that IKNN provides the smallest energy consumption, due to its itinerary shape. The best query result accuracy was provided by GRT, mainly in sparse WSN. However, the Flooding used to disseminate the query consumes too much energy.

DIKNN - Density-aware Itinerary KNN query processing: proposed in [Wu et al., 2007], the main contribution of this work is an algorithm for parallel KNN query processing that can increase or decrease the radius of the RoI during query processing. DIKNN calculates the network density during the Forwarding stage. This information is used to estimate the radius of the RoI that probably contains K nodes. However, a random deployment of nodes and node mobility can degrade the accuracy of the estimation. To handle this problem, when the query reaches two neighbor nodes in different itineraries, these nodes sum the number of queried nodes in each itinerary up to that moment. The dissemination stops if K nodes were queried. If not, a new radius for the RoI will be calculated.

PCIKNN - Parallel Concentric Itinerary KNN: this mechanism also processes KNN queries using parallel itineraries [Fu et al., 2010]. The main differences between it and DIKNN are the itinerary shape and the strategy to handle the random deployment of nodes. This mechanism calculates the network density during the traversal of the itinerary, hence the radius of the RoI can be increased or decreased to compensate variations in nodes density and node mobility. The experiments showed that PCIKNN presents lower energy consumption and query latency than DIKNN.

2.7.4.1 Analyzing the KNN Query Processing Mechanisms

In the analyzed works about KNN query processing, the main difference is the itinerary shape. Parallel itineraries presented the best performance. To the best of the authors' knowledge, these mechanisms use GPSR to forward the query. Hence, nodes need to maintain neighbor tables in order to obtain the next node on the itinerary. In failure-prone WSN or in networks having duty cycle algorithms, nodes have to transmit periodic control messages in order to refresh the data in the node tables, increasing substantially the energy consumption.

2.8 Chapter Remarks

This chapter presented a survey on spatial query processing in WSN. We have divided the query processing into six stages and defined the tasks that are performed in each of these stages. The division facilitates the understanding of spatial query processing and simplifies the proposal of new solutions. We also have classified the queries found in the literature according to the selection of the nodes that will answer the query and the way they will answer it. Then, we described the algorithms found in the literature for each stage and presented an overview of distributed spatial index for spatial query processing. Thus, we surveyed the spatial query processing mechanisms describing the algorithms used in each of their stages.

We verify that the spatial query processing mechanisms found in the literature use the Greedy or GPSR routing protocol in their Forwarding and Return stages. Nodes using these protocols create neighbor tables in order to choose the next node to forward packets. The maintenance of such tables can consume a substantial amount of the energy supply since all nodes must send beacons periodically. However, this operation is not sufficient to guarantee updated neighbor tables. Since most of the WSN employ duty cycles to save energy, a node can affirm the state of a specific neighbor if it knows the sleep schedule and is perfectly synchronized with this neighbor. Furthermore, WSN are prone to failure and nodes have no previous knowledge about failures. Therefore, nodes cannot attest that all nodes in their neighbor table do not fail, hence they cannot guarantee packet delivery.

In the Dissemination stage, we found two main algorithms: Restricted Flooding and Itinerary. According to the related works, Itinerary presents the best energy consumption, an acceptable rate of processing success and good precision in the query result. Furthermore, we verify that the mechanisms found in the literature consider rectangular or circular RoIs. To the best of the authors' knowledge, no spatial query processing mechanism considers polygonal RoIs, which can represent the contours of real objects. The number of points that form polygons depends on the level of details of the RoI. Hence, queries can occupy several packets when the polygon that represents the RoI is formed by many points. Consequently, this tends to increase the energy consumption.

Chapter 3

Simplifying the Region of Interest

This chapter presents a new algorithm for the Pre-Processing stage. It was devised to reduce the number of packets required to represent a query and thus save energy during query processing. We consider spatial queries having RoIs with polygonal shapes. The proposed algorithm simplifies the polygon that defines the RoI, reducing the amount of points necessary for its representation. Hence, the query requires fewer packets, reducing the energy consumption during query processing. Section 3.1 shows the problem definition. Section 3.2 describes the geographical coordinates used here. Section 3.3 presents the proposed algorithm. Section 3.4 presents the complexity analysis of this algorithm. Section 3.5 discusses the impact of this algorithm in the query processing. And finally, Section 3.6 presents the Chapter Remarks.

3.1 Problem Definition

Regions of interest are represented by polygons. A polygon is represented by a set of consecutive points, in which the last is connected to the first. Depending on the level of detail in a figure representing a region, the number of points of the polygon can be large. Since packets in WSN have only about thirty bytes of payload [Crossbow Technology, Inc., 2010a; Moteiv Corporation, 2006; Crossbow Technology, Inc., 2010b], queries with this type of RoI can occupy several packets. The energy consumption to process these queries tends to be high, since a query is transmitted several times during its processing. Figure 3.1(a) presents an example. It shows the contours of a lake defined by 140 points and representing a RoI.

This work considers 2D geographical coordinates in the decimal degrees notation, with four decimal places. It gives an accuracy of about 11m on the Equator (worst case). Each point occupies 64 bits (two float values). Assuming WSN packets having

28 bytes of payload (such as the sensor node Iris [Crossbow Technology, Inc., 2010a]), the query occupies at least 40 packets. The energy consumption to process a query having this RoI tends to be high.



Figure 3.1. (a)Contours of a lake. (b) Simplified contours of a lake.

3.2 Relative Geographical Coordinates

In order to reduce the number of bits used to represent a point, we use relative geographical coordinates. At the initialization of the network, a node identified with the number one, called $N1$, initiates a flood. Each transmitted packet contains the location of $N1$. Hence, this location becomes the network **initial point**. All points handled by the nodes are represented as the position relative to $N1$. This work considers degree decimal coordinates using four places after the point. This provides precision of 11 meters on the Equator (the worst case). Using this setup, the number of bits to represent a point is reduced from 64 (two float point values) to 16 bits, without losing precision. Hence, using relative geographical coordinates, two bytes can represent every point in a square region of 2816m of side. Let's give an example. Suppose that $N1$ is located at *longitude* = -43.9643 and *latitude* = -19.8690 . This is considered the initial point, i.e., position $X = 0$ and $Y = 0$. A node located at the coordinate *longitude* = -43.9657 and *latitude* = -19.8688 represents its position by $X = -14$ and $Y = 2$.

3.3 Simplifying the Regions of Interest

In order to minimize the energy consumption, we propose a new algorithm that reduces the amount of packets necessary to transmit a query. The algorithm simplifies the polygon that represents the RoI by eliminating points in its representation. Thus, a

query requires less packets and, consequently, consumes less energy to be processed. Many algorithms have been created to reduce the number of points in the representation of real objects plotted in maps, such as described in [Tobler, 1964] [Robinson et al., 1978] [Jenks, 1981]. These algorithms are useful to reduce the amount of data in spatial databases and to show these objects with less details on large scale maps [Casanova et al., 2005]. Normally, these algorithms are used to simplify line segments, such as rivers, roads and other objects with complex shapes.

The **Douglas-Peucker Algorithm** [Douglas and Peucker, 1973] is the most referenced algorithm to simplify lines formed by a set of consecutive points. It starts by selecting the first and the last points (called A and Z) in the set of points (called L). After this, it seeks for the point P in L that is the most distant from the line segment formed by the points A and Z (called \overline{AZ}). If this distance is smaller than a threshold ts , given by the user, all points between A and Z will be dropped. If not, the algorithm will be recursively called twice, for the line segments \overline{AP} and \overline{PZ} .

The Douglas-Peucker Algorithm is presented by the pseudo code in Algorithm 1. In line 1, the function *douglas_peucker()* receives as parameter a set of points (L), the threshold (ts) and a line segment (\overline{AZ}) formed by the first ($start$) and the last point (end) to be analyzed. The function *most_distant* returns p , which is the most distant point from \overline{AZ} (line 2). If p is closer to \overline{AZ} than ts , all points between $start$ and end will be dropped in the set of points L (line 4). Else, the function is recursively called twice for the intervals $[start, p]$ and $[p, end]$ (lines 6 - 7).

Algorithm 1 Douglas-Peucker algorithm.

```

1: procedure DOUGLAS_PEUCKER( $L[], ts, start, end$ )
2:    $p \leftarrow most\_distant(L[start], L[end])$ 
3:   if  $distance\_point\_line(p, L[start], L[end]) \leq ts$  then
4:      $drop\_points\_between(L[start], L[end]) \vee end - start \leq 1$ 
5:   else
6:      $douglas\_peucker(L, ts, start, p)$ 
7:      $douglas\_peucker(L, ts, p, end)$ 

```

The algorithm proposed here to simplify RoIs is based on the Douglas-Peucker Algorithm. It eliminates points in the polygon that defines the RoI. The user establishes the number of packets that the query will occupy ($N_{Packets}$). The maximum number of points of the RoI ($Size_{Max}$) is calculated by multiplying $N_{Packets}$ by the number of points that can be stored in a WSN packet. Hence, the Douglas-Peucker algorithm is called for $ts = 1$. At the end, it verifies if the returned polygon has at most $Size_{Max}$ points. If so, the algorithm returns this polygon. If not, the process is executed again increasing values of ts . Hence, the threshold is increased until the polygons has at most $Size_{Max}$ points. The proposed algorithm is presented in Algorithm 2.

Algorithm 2 Reducing the number of points of the RoI.

```

1: procedure REDUCE_ROI(qry.roi, size_max)
2:   ts  $\leftarrow$  1
3:   while qry.roi.size > size_max do
4:     DOUGLAS_PEUCKER(qry.roi, ts, 0, qry.roi.size - 1)
5:     ts ++
6:   return qry.roi

```

3.4 Complexity Analysis

In this section, we analyze the time complexity of the proposed algorithm. We first analyze the Douglas-Peucker algorithm, which is the basis of our algorithm. We consider the Douglas-Peucker algorithm described in [Douglas and Peucker, 1973] and presented in Algorithm 1. Its complexity depends on the function *most_distant()* (line 2 - Algorithm 1) and the two recursive calls (lines 6 and 7 of Algorithm 1). The function *most_distant()* returns the most distant point from the line segment \overline{AZ} . This Function has complexity $O(n)$, since it analyzes the $n - 2$ points between the points L[start] and L[end] that represent the polygon. In the average case, *most_distant()* returns the point in the middle of the line \overline{AZ} . Hence, the complexity of the Douglas-Peucker algorithm is obtained by the following recurrence relation:

$$\begin{cases} T(2) = 0 \\ T(n) = 2T(\frac{n}{2}) + (n - 2) \end{cases}$$

According to the Master Theorem, the solution of this recurrence is $T(n) \in \Theta(n \log n)$. The worst case of Douglas-Peucker algorithm occurs when the function *most_distant* always returns the first point of the line \overline{AZ} . Thus, the complexity is obtained by the recurrence relation below:

$$\begin{cases} T(2) = 0 \\ T(n) = T(n - 1) + T(2) + (n - 2) \end{cases}$$

Hence, the solution of this recurrence relation shows that the time complexity in the worst case of the Douglas-Peucker algorithm is $T(n) \in \Theta(n^2)$.

The algorithm presented in Algorithm 2 calls the *DOUGLAS_PEUCKER* algorithm several times. For each call, the value of *ts* is incremented in one unit *U* (in this work we consider $U = 1$ meter). The biggest value of *ts* is the length of the RoI (*length_{RoI}* - Section 2.3), since using values bigger than *length_{RoI}* the *DOUGLAS_PEUCKER* algorithm drops all the points in the polygon representing the RoI. Let's call *m* the number of times Algorithm 2 calls the

DOUGLAS_PEUCKER algorithm. Hence, assuming that ts increases up to the value of $length_{RoI}$ (worse case), the value of m is calculated by the following equation:

$$m = \left\lceil \frac{length_{RoI}}{U} \right\rceil \quad (3.1)$$

Thus, the complexity of the Algorithm 2 is $T(m, n) \in \Theta(m * n^2)$, where m depends on the length of the RoI and n is the number of points forming the RoI. This algorithm can be changed in order to obtain the complexity $T(m, n) \in \Theta((\log m) * n^2)$. However, despite the better complexity, this new algorithm would present processing time larger than the algorithm used here. We can create the new algorithm using a binary search (complexity $T(m) \in \Theta(\log m)$) in order to find the best value for ts . However, in practice the best value for ts is so much smaller than m . Since the algorithm used here find ts using a sequential algorithm, it performs faster in than the algorithm using a binary search.

3.5 Impact of the Simplification in the Query Processing

The proposed algorithm simplifies the polygon that represents the RoI. Hence, the area where the sensor nodes collect data can change, which can change the query result. We call **coverage area** the portion of the RoI that is sensed by one or more sensor nodes during the query processing. The ideal coverage area covers the entire RoI informed by the user (called original RoI). When the Pre-Processing algorithm simplifies the original RoI, it creates a new RoI. Thus, some nodes can be excluded or included of this new RoI, such as illustrated by Figure 3.2. In this figure, we verify a node that is outside of the original RoI and inside the new RoI. This node unnecessarily collects data during the query processing. Moreover, we also verify a node inside the original RoI and outside the new RoI. In the last case, the nodes should collect data, but they do not. The influence of these nodes in the query result is analyzed in our experiments.

Another impact of the Pre-Processing algorithm is its execution time. However, it is not significant if compared to the total time spent to process spatial queries. The Pre-Processing algorithm is performed in the user's computer. We consider that the user has a conventional computer, attached to a sensor node in order to connect the user's computer into the sensor network. This computer has no processing restrictions such as in the sensor nodes. Thus, the user computer can generate the new shape of the RoI in some milliseconds, while the spatial query processing spends some seconds in

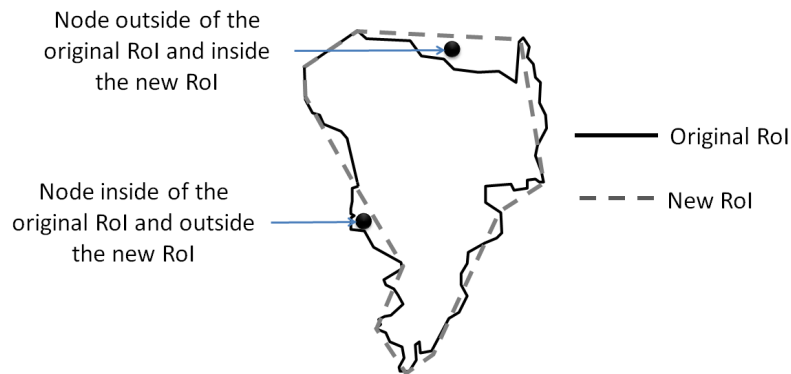


Figure 3.2. Collateral effects of the Pre-Processing algorithm.

the sensor nodes. Thus, it is more interesting to perform an expensive pre-processing in the user's computer in order to save resource in the sensor nodes. Consequently, we do not consider the execution time of the Pre-Processing algorithm in our analysis.

3.6 Chapter Remarks

This chapter presents a new algorithm to reduce the number of packets necessary to represent queries, in order to reduce the energy consumption of the query processing. Depending on the level of details of the RoI, the polygon that represents it can be formed by a larger number of points. Thus, queries having these types of RoIs can occupy many packets and consume substantial part of the WSN energy. Simplifying the RoIs reduces the number of points of the polygon and the queries use less packets, reducing the energy consumption.

We propose to use relative geographical coordinates in order to reduce the amount of bits necessary to represent a point. A geographical coordinate is defined by two values (longitude and latitude) occupying two float values (32 bits each). Using relative geographical coordinates, each point occupies only 16 bits (one byte to longitude and one byte to latitude), without losing precision. Hence, queries occupy less packets and, consequently, consume less energy during their processing.

We also propose a Pre-Processing algorithm that simplifies the contours of the polygons that represents the RoI. The user computer performs the algorithm before the WSN start processing the query. This algorithm simplifies the polygon reducing the amount of points in the polygon representation. Hence, queries occupy less packets and their processing consume less energy.

Chapter 4

ABF - Ask Before Forwarding Routing Protocol

This work proposes a new location-based routing protocol called ABF (Ask Before Forwarding) to be used by spatial query processing mechanisms in the Forwarding and Return stages. It is devised to forward queries from the Originator to a node inside the RoI and to forward back the query result. We assume WSN nodes executing duty cycle algorithms in order to save energy and WSN having failure-prone nodes.

In order to work properly in this type of scenarios, ABF uses a receiver-based strategy. The neighbors of the node having a packet to forward choose which node will be the next to forward. The majority of the spatial query processing mechanisms found in the literature uses sender-based routing protocols, such as Greedy or GPSR (Subsection 2.6.2). Using these protocols, nodes build neighbor tables in order to choose the next node to forward packets. Duty cycle algorithms and node failures change the network topology, hence the WSN must to periodically update the neighbor tables. The neighbor table maintenance consumes a substantial part of the network energy supply. Moreover, it does not guarantee successful forwarding, since nodes cannot attest that nodes in their tables have not failed since the last update.

In the following, Section 4.1 presents the problem definition. Section 4.2 describes the policies to select the Coordinator. Section 4.3 describes the strategy used by ABF to forward queries and query results. Section 4.4 presents how the protocol forwards packets towards the destination. Section 4.5 describes how ABF avoids holes in the network. Section 4.6 describes the ABF strategy to detect and manage node failures. Finally, Section 4.7 describes how ABF detects forwarding failures in order to avoid loops.

4.1 Problem Definition

In this chapter we focus on two similar problems: forwarding queries from the Originator to the RoI and forwarding back the query result from the RoI to the Originator. The first problem begins when the Originator receives a query from the user. The sensor network needs to forward the packets representing the query from this node to a node inside the RoI, called Coordinator. We assume that the Originator can be any node of the WSN. This assumption is plausible since the majority of the sensor nodes are easily coupled with laptops in order to create interfaces between the user and the WSN [Crossbow Technology, Inc., 2010a,b; Moteiv Corporation, 2006]. The Coordinator selection depends on the algorithm used in the Dissemination stage, which is discussed later.

The second problem begins when a node inside the RoI (the Aggregator) calculates the query result at the end of the Aggregation stage and needs to forward back this data to the Originator. The Aggregator selection depends on the dissemination/aggregation algorithm used by the spatial query mechanism. We verify that both problems must be solved by a geographic routing protocol.

The proposed protocol must create routes in WSN having nodes that maintain no knowledge about their vicinity. This assumption is because we consider WSN employing duty cycle algorithms, in which the sleep schedule is not known. Furthermore, we assume that these networks have failure-prone sensor nodes. Hence, when a node has a packet to be forwarded, it cannot attest if any of its neighbors can receive this packet since the neighbors can be sleeping or can be failed.

4.2 Coordinator Selection

The selection of the last node in the Forwarding stage (called Coordinator) and the last node in the Aggregation stage (called Aggregator) depends on the dissemination/aggregation algorithm used by the spatial query mechanism. However, these nodes are important to ABF because the Coordinator is the last node in the Forwarding stage and the Aggregator is the first node in the Return stage. Since we evaluate two algorithms for the Dissemination stage (presented in the following), we present two policies for Coordinator and Aggregator selection. When Restricted Flooding is used to disseminate queries, the first node to receive the query inside the ROI becomes both the Coordinator and the Aggregator. At the beginning of the Forwarding stage,

the Originator establishes the centroid¹ of the ROI as **reference point** to forward the query. If a node inside the ROI receives the query, the Forwarding stage finishes and this node becomes the Coordinator. Since this node will receive readings from all nodes in the ROI, it also acquires the task of Aggregator. Figure 4.1-a illustrates the forwarding for Restricted Flooding. The reference point is the centroid of the ROI and the Coordinator is the first node to receive the query inside this region.

When the spatial query processing mechanism uses an itinerary-based algorithm to disseminate queries, the reference point is the leftmost point (LMP) of the ROI, i.e., the point of the ROI having the lowest value in the X axis (longitude). The Coordinator is a node inside the ROI with a distance to the LMP smaller than a predefined value (ID - defined in Subsection 5.4.1). The Aggregator is the last node in the itinerary (defined in Subsection 5.4.2). These choices help the itinerary definition, since the itinerary sweeps the entire polygon (as is shown in the next section). Figure 4.1-b illustrates a query forwarding when itinerary-based dissemination is used.

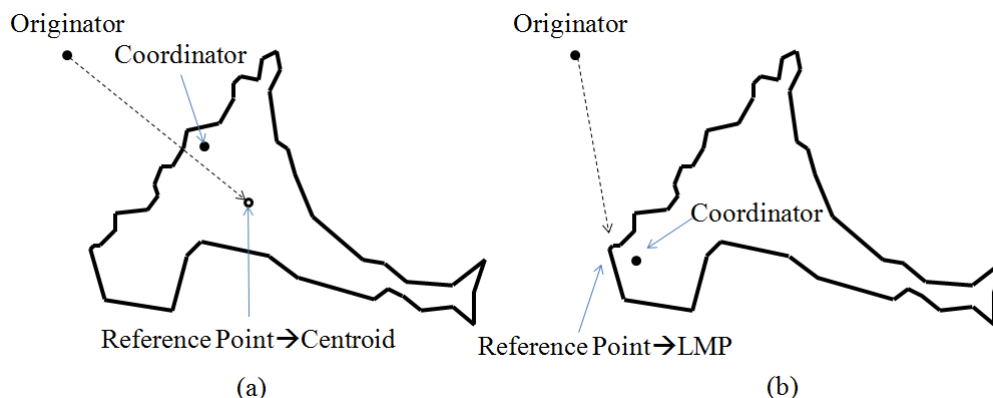


Figure 4.1. Reference point in Forwarding: (a) when Restricted Flooding is used to disseminate, (b) when itinerary is used to disseminate

4.3 ABF Strategy

During the Forwarding and Return stages, the sender node first verifies which neighbors can receive the query or the query result and then forwards it to the next node in the route. ABF performs this verification to ignore failed and inactive nodes. Figure 4.2 presents a state diagram of the verification process. It is performed by a node that

¹The centroid of a 2D figure is the center of mass of a geometric object of uniform density. Also called geometric center or barycenter, the centroid of a plane figure or two-dimensional shape X is the intersection of all straight lines that divide X into two parts of equal momentum about the line [Cobuild, 2003].

holds a query to be forwarded (**Holding Query** state). This node sends a **request-packet** to its neighbors and waits for **announce-packets** (**Waiting for Announces** state). A request-packet notifies that there is a node in the neighborhood that has a packet to forward. This packet has the location of its sender and the reference point of the query. An announce-packet notifies that the node who sent it is active. If the node in the Waiting for Announces state receives no announce-packet, it resends the request-packet after a period of time. This node goes to the **Sending Query** state after receiving the first announce-packet. In this state, this node sends all the query-packets (one query can be composed of more than one packet) to the node that sent the first announce-packet received. Then, the sender goes to the **Idle** state waiting for a new query. Subsequent announce-packets are ignored, since probably the first announce packet received came from the neighbor closest to the destination (more details in the next subsection).

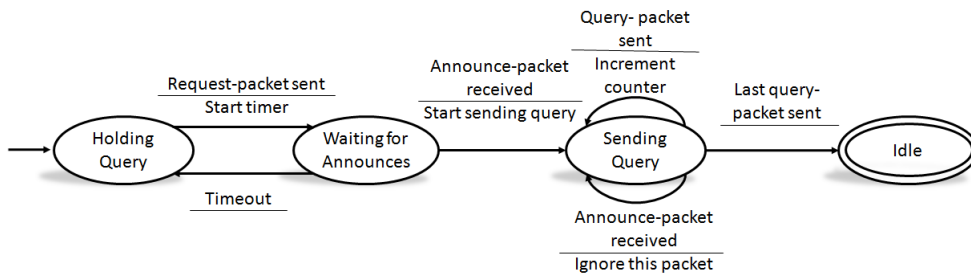


Figure 4.2. State diagram of a forwarder node.

The receiver node must answer a request-packet (sending an announce-packet) in order to inform the sender that it is able to be the next node in the route. However, the receiver inserts the delay T_{Wait} before sending a response. The value of T_{Wait} is proportional to the nodes' distance to the RoI (T_{Wait} is defined in the next subsection). The node that calculate the smallest T_{Wait} probably is the next node in the route. This is because the sender defines as the next node the source of the first announce-packet received. Figure 4.3 presents the state diagram of the receiver. When a node in the **Idle** state receives a request-packet, it goes to the **Waiting T_{Wait}** state, stays in this state for T_{Wait} seconds, sends an announce-packet and goes to the **Waiting for Query** state. In the **Waiting T_{Wait}** state, if the node receives a query-packet and the query is not directed to it, this node cancels the announce-packet transmission and goes to the **Sleep** state in order to avoid overhearing. After sending an announce-packet, if this node receives a query-packet directed to it, it will go to the **Receiving Query** state and stays in this state until it receives all the query-packets.

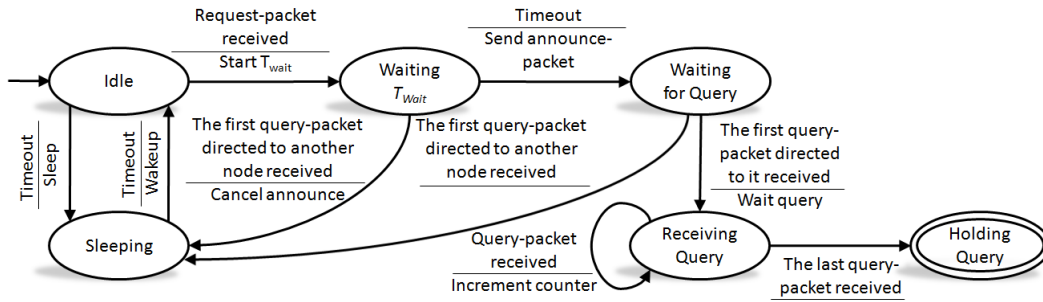


Figure 4.3. State diagram of a node that received a request-packet.

4.4 The Delay to Answer a Request-Packet

Nodes insert the T_{Wait} delay before sending an announce-packet. This delay is proportional to the node's distance from the reference point. This point depends on the dissemination algorithm, described in the next chapter. For Restricted Flooding, the reference point is the centroid of the RoI and for Itinerary this point is the leftmost point (LMP) of the RoI.

Let's call S the sender, R the reference point, D a neighbor of S and r the radio range. We define as **ideal position** (IP) a point that is r meters distant from S and on the line segment \overline{SR} . The ideal position is the closest point to the destination that S can send packets. Figure 4.4 illustrates these definitions. Since the request-packet has the location of S and R , all nodes can calculate IP . The value of T_{Wait} is based on the distance between the node and IP , as follows:

$$T_{Wait} = \frac{d}{r} * T_{Max} \quad (4.1)$$

where d is the distance between D and IP and T_{max} is the highest value of T_{Wait} . Using this equation, nodes closer to IP send their announce-packets before nodes farther from IP . Thus, the closer nodes have a higher probability of being the next node to forward the query.

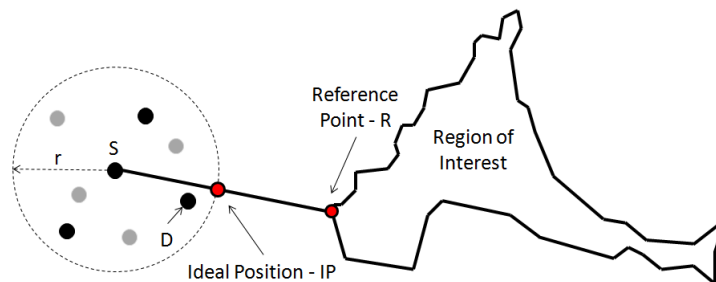


Figure 4.4. Points defined in the Forwarding stage.

4.5 Avoiding Holes in the Network

Holes in the network happen when a node needs to send a packet and it has no neighbor closer to the destination than itself [Zhao and Guibas, 2004]. Figure 4.5 illustrates this situation. In order to avoid holes, ABF applies the nodes' location on the planar graph to define T_{Wait} . The nodes obtain the network planar graph using the Gabriel Algorithm [Gabriel and Sokal, 1969]. Nodes of the planar graph bypass the network hole forming a face. These nodes transmit the forwarding packets in order to go around the hole. Hence, if a node is farther from IP than the sender S (forming a hole), it calculates T_{Wait} as follows:

$$T_{Wait} = T_{max} + T_{max} * \frac{dp_i}{r} \quad (4.2)$$

where r is radio range and dp is the distance between IP and the projection of the node location on the line \overline{SR} . Figure 4.5 illustrates how dp is calculated. Node i defines a line L_i that is perpendicular to \overline{SR} and passes through i 's location. P_i is the point where L_i crosses \overline{SR} . dp_i is the distance between P_i and IP . Using these equations, nodes use a greedy forwarding strategy when possible and avoid holes without using a neighbor table.

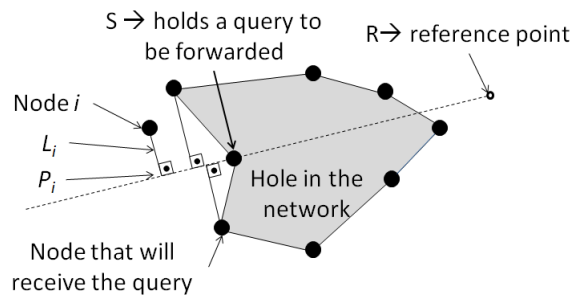


Figure 4.5. Hole in the network and node position projected on the line \overline{SR} .

4.6 Node Failure Detection

Nodes using ABF overhear the request-packet from the next node in the route in order to ensure that the query was forwarded correctly. After sending the query, the sender waits T_{Ret} seconds for the request-packet from the receiver. If this timer finishes and the expected packet was not received, the sender concludes that a failure happened and sends another request-packet in order to choose another neighbor to send the query. This situation can be caused by a failure in the receiver (which can be solved by the

request-packet retransmission) or by a failure in the sender (which happens after the sender obtains the query from the previous node in the route). The latter is identified when the sender receives no announce-packets from its neighbor. In order to manage this situation, ABF retransmits the request-packet up to R times. If no announce-packet is received, the query will be canceled. Empirically, we established $T_{Ret} = 3s$ and $R = 10$. When we used larger values for these parameters, the query processing time increased substantially.

4.7 Forwarding Failure

A **Forwarding Failure** happens when a node cannot forward the query towards the RoI and drops it. During query processing, nodes using ABF identify a forwarding failure when they have no neighbor to send the query. Each node maintains two lists of nodes: nodes that sent the query to it and neighbors from which the node received the query. When a node receives a query more than once, it sends the request-packet normally, but it waits for an announce-packet sent by a node that is not in these lists. Moreover, nodes only send announce-packets to nodes out of its lists. Hence, a node can receive a query more than once from different sources, but it sends or receives it once for each of its neighbors. This guarantees that queries can find loops during the Forwarding stage, but loops are followed just once. Figure 4.6 illustrates when a node identifies a forwarding failure. This node has three neighbors, but it cannot send the query to them because it sent the query to one and received the query from the other two neighbors.

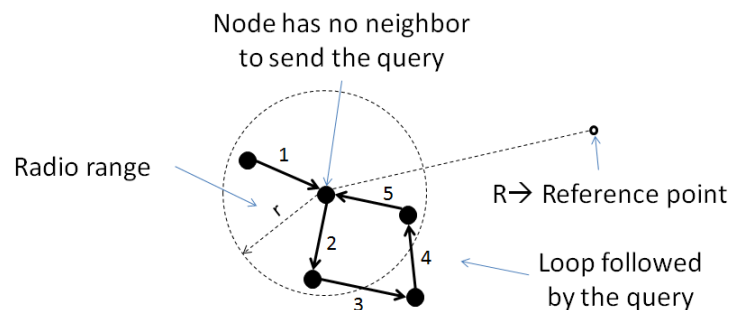


Figure 4.6. Forward failure.

4.8 Chapter Remarks

This chapter proposes ABF, a new geographic routing protocol to be used in the Forwarding and Return stages of spatial query processing mechanisms. This protocol is devised to forward packets in failure-prone WSN and networks having some nodes in sleep mode. It first sends a packet requesting for active neighbors, then receives the first neighbor announce-packet and sends the packet to this neighbor. Therefore, only active nodes participate in the route. Since sleeping and failed nodes do not receive requests, they are ignored in this process.

When a node receives a request to forward a packet, it inserts a small delay (T_{Wait}) before responding to this request. T_{Wait} is proportional to the node distance to the reference point of the query. Since the first node to retrieve the request is chosen as the next node in the route, nodes closer to the destination have higher probability to be chosen as the next node in the route. However, when the packet reaches a node having no neighbor closer to the destination than itself (hole in the network) the nodes in the contour of the hole forward the packet. These nodes have a higher value of T_{Wait} than the nodes closer to the destination. Hence, the forwarding packets usually follow a greedy strategy, but they are able to cope with holes in the network.

ABF detects failures overhearing the request-packet sent by the next node in the route. After sending a packet, the sender waits for the request-packet sent by the next node in the route. The sender uses this request to confirm that the packet was forwarded successfully. When the sender does not receive this message, it sends another request-packet in order to find another neighbor to forward the packet.

The advantage of ABF is that it does not maintain neighbor tables. The routing protocols used by the spatial query processing mechanisms found in the literature (Greedy and GPSR) create neighbor tables in order to obtain the next node in the route. Thus, in WSN having a dynamic topology, all nodes must periodically transmit beacon messages in order to update the neighbor tables. These protocols consume substantial part of the energy supply to maintain neighbor tables. Nodes employing ABF maintain no data about the network topology, reducing the energy consumption by eliminating beacon messages.

Chapter 5

Dissemination and Aggregation Stages

In the Dissemination stage, the query is transmitted to all nodes inside the RoI. In the Aggregation stage, the readings collected by these nodes are transmitted to the Aggregator. During this process, the query is aggregated and the query result is calculated node by node. The way the WSN disseminates queries determines how the nodes aggregate the readings. This is because the query dissemination creates routes, which are used in the Aggregation stage. Hence, we propose one algorithm to perform these two stages.

In this chapter, we propose three algorithms for the Dissemination and Aggregation stages: Classic Restricted Flooding, Delayed Restricted Flooding and Itinerary. The first two are based on Restricted Flooding (Subsection 2.6.3.1), but they use different strategies to aggregate the nodes' readings. The last algorithm creates itineraries to transmit queries. These three algorithms were created in order to evaluate the best dissemination/aggregation strategy in different network topologies, duty cycles and failure models.

The remaining of the chapter is organized as follows. Section 5.1 presents the dissemination and aggregation problem definitions. The following sections present the proposed algorithms: Section 5.2 describes Classic Restricted Flooding, Section 5.3 describes Delayed Restricted Flooding and Section 5.4 describes the Itinerary. Finally, Section 5.5 presents the chapter remarks.

5.1 Dissemination/Aggregation Problem Definition

The algorithms proposed here were created to solve two problems: disseminate queries to all nodes inside the RoI and perform in-network aggregation in the nodes' reading in order to obtain the query result. The Dissemination stage starts in the Coordinator and then the query is transmitted to reach the entire RoI. Nodes outside the RoI ignore received queries. The in-network aggregation (described in Subsection 2.6.5) reduces the amount of packets transmitted during the Aggregation stage. When all readings are transmitted to one node and only this node performs the aggregation, the amount of transmitted packets increases. Using in-network aggregation, nodes receive readings from its descendents, apply the aggregation operator in this data and send the result to another node, which will perform the same algorithm. The last node to perform the aggregation can calculate the query result. Hence, each node sends only one packet during the aggregation process.

5.2 Classic Restricted Flooding

This algorithm is called "Classic" because it is based on Flooding [Akyildiz et al., 2002]: each node receives the query and rebroadcasts it. However, in Restricted Flooding only nodes inside the RoI broadcast the query. During query dissemination, each node saves as its **parent** the node that first sent the query to it. This operation creates a structure called *Routing Tree* [Zhao and Guibas, 2004], which is used in the Aggregation stage. A node i puts the identifier of its parent p into the query it is disseminating in order to notify p that i is its son. Hence, nodes having no children are leaf nodes in the routing tree. A leaf node only collects data and sends it to its parent. The other nodes collect data, receive data from their sons, apply the aggregation operator and send the result to their parents. This process is performed until the data reach the root node, which assumes the tasks of Coordinator and Aggregator.

When a node receives a query to be disseminated, it executes Algorithm 3. Line 2 verifies if the node is inside the RoI (function $in_polygon()$); if not, the query is dropped (line 13). A node inside the RoI verifies if this query is a new query (line 3). If so, the node that sent the query is saved as parent (line 4), the query is saved as *old query* (line 5), it is disseminated (line 6) and the node starts a T_{Leaf} timer (line 7). If the query is not new, the node verifies if the parent of the source node is itself and saves this node as its son (line 9). The complexity of the Algorithm 3 is $O(n)$ where n is number of points representing the RoI. It is determined by the function $in_polygon()$, which is $O(n)$ according to [Casanova et al., 2005].

Algorithm 3 Classic Restricted Flooding.

```

1: procedure RECEIVE_QUERY(qry)
2:   if in_polygon(qry.roi) then
3:     if qry.id  $\notin$  old then
4:       parent_id  $\leftarrow$  qry.source
5:       old  $\leftarrow$  old  $\cup$  qry.id
6:       resend_query(qry)
7:       tleaf.start()
8:     else if qry.parent = my_id then
9:       son_id  $\leftarrow$  son_id  $\cup$  qry.source
10:    else
11:      drop(qry)
12:  else
13:    drop(qry)

```

Nodes use tables of sons and two timers (T_{Leaf} and T_{Sons}) to perform the Aggregation stage. T_{Leaf} is used to identify if a node is a leaf node. T_{Sons} defines the period of time a node waits for their sons' readings. Figure 5.1 presents the state diagram for the Classic Restricted Flooding. After receiving a query, a node is in the **Holding Query** state. This node must disseminate the query, thus it sends the first query-packet and goes to the **Sending Query** state. After disseminating the query (it sent all the query-packets), this node needs to know if it is a leaf node. So, it waits for T_{Leaf} seconds (**Waiting T_{Leaf}** state) for its sons' query-packets. When the timer T_{Leaf} expires, if this node received no query-packet from its sons, it realizes that it is a leaf node, so it sends to its parent an aggregation-packet having only its reading and goes to the **Idle state**. However, when a node received a query-packet from a son during the **Waiting T_{Leaf}** state, it goes to the **Waiting T_{Son}** state. In this stage, the node receives the query-packets sent by its sons and count how many sons it has. Moreover, it also receives the aggregation-packets from these nodes. When the timer T_{Son} expires, the node senses the environment, applies the aggregation operator, sends to its parent an aggregation-packet and goes to the **Idle state**.

When a node receives an aggregation-packet after sending its reading (in the **Idle state**), this node sends to its parent another aggregation-packet containing only the received data. We call these packets **late aggregation packets - LAP**. This kind of packet avoids to lose readings collected by nodes far from the Coordinator, since these nodes are the last to send readings. Nodes next to the Coordinator are propitious to receive LAPs from their more distant descendents, increasing the energy consumption.

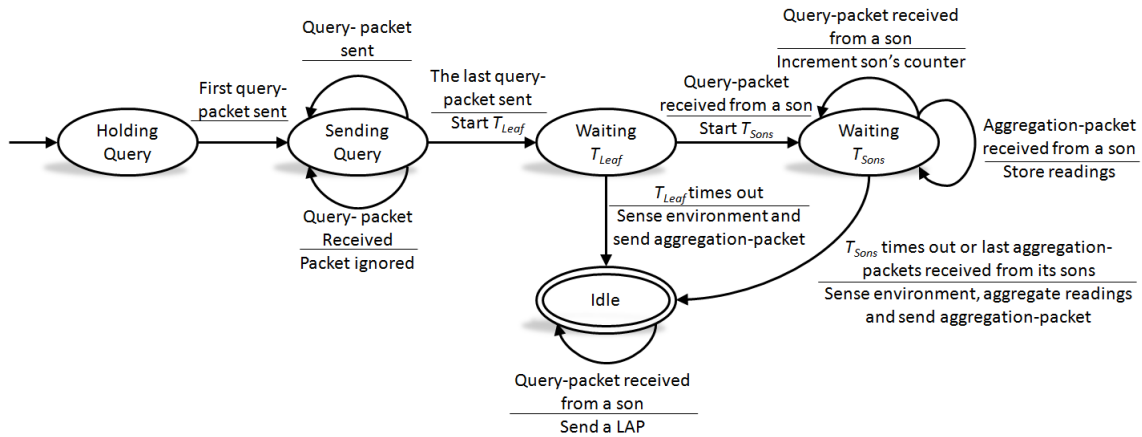


Figure 5.1. State diagram of the Classic Restricted Flooding.

5.3 Delayed Restricted Flooding

Delayed Restricted Flooding disseminates the query as in Classic Restricted Flooding, however the aggregation is performed in a different way. This algorithm was created in order to avoid packet losses during the Dissemination stage. Algorithms based on Flooding are prone to packet collisions, since all neighbors of a node try to transmit the query at the same time [Akyildiz et al., 2002]. Thus, due to collisions, the number of perceived sons of a node is usually smaller than the number of actual sons. Consequently, nodes performing Classic Restricted Flooding may send aggregation results before receiving all readings from their sons, increasing the amount of LAPs.

Nodes performing the Delayed Restricted Flooding algorithm do not create lists of sons. The algorithm defines the **coordinator timer** T_{Coord} on the Coordinator and the **aggregation timer** T_{Agg} on the other nodes. After disseminating a query, each node starts its timer. When T_{Agg} expires in a node, it sends to its parent the result of the aggregation operator (aggregation-packet). When T_{Coord} expires in the Coordinator, it calculates the query result and starts the Return Stage. The Coordinator calculates the value of T_{Coord} using the following equation:

$$T_{Coord} = T_{Pred_diss} + T_{Pred_agg} \quad (5.1)$$

where T_{Pred_diss} is the predicted time required to disseminate the query and T_{Pred_agg} is the predicted time required to aggregate. T_{Pred_diss} is calculated by the following equation:

$$T_{Pred_diss} = predicted_levels * T_{Send} * query_packets \quad (5.2)$$

T_{Send} is the approximate time to send a packet, $query_packets$ is the number of packets needed to hold the query, and $predicted_levels$ is the predicted number of levels in the routing tree created during the dissemination, which is estimated by the equation:

$$predicted_levels = \frac{length_{RoI}}{r} * 2 \quad (5.3)$$

where $length_{RoI}$ is the length of the RoI (Section 2.3), r is the radio range and 2 is an adjustment value obtained empirically. T_{Pred_agg} is calculated by the following equation:

$$T_{Pred_agg} = predicted_levels * T_{Decrease} \quad (5.4)$$

where $T_{Decrease}$ is a default value used to decrease the value of T_{Agg} proportionally to the node's level in the routing tree. Since the Coordinator has the longest timer (T_{Coord}), the other nodes calculate T_{Agg} using the following equation:

$$T_{Agg-i} = T_{Coord} - Level_i * T_{Decrease} \quad (5.5)$$

where T_{Agg-i} is the timer of node i and $Level_i$ is the level (in hops) of this node in the routing tree. The best value for $T_{Decrease}$ was empirically obtained by simulation (Subsection 6.4.1). Its value should be large enough to avoid collisions and should not increase too much the end-to-end delay. Figure 5.2 illustrates the value of T_{Agg} decreasing $T_{Decrease}$ each level down of the routing tree.

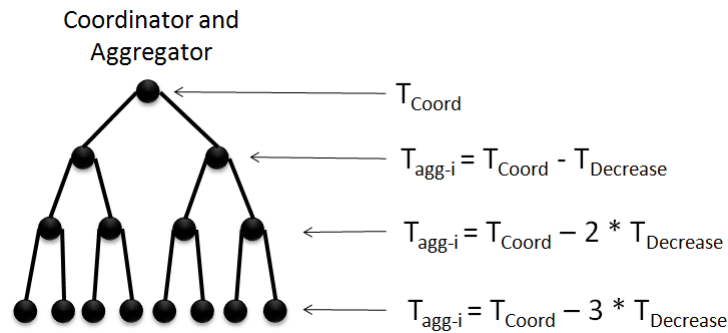


Figure 5.2. The value of T_{Agg} in the routing tree.

Usually, all descendants of a node try to send the aggregation-packet at the same time. Thus, these packets may collide. In order to avoid these collisions during the Aggregation stage, each node waits a small random period of time before sending the aggregation result. This period of time varies between 0 and 40% of $T_{decrease}$. This

value was obtained empirically during the development and testing of this algorithm. Values larger than 40% increased the end-to-end delay, and smaller values increased the amount of collisions.

Figure 5.3 presents the state diagram of Delayed Restricted Flooding. A node goes to the ‘‘Holding a Query’’ state after receiving all the query-packet. Thus, it sends the query to its descendents in the Sending Query state, starts the timer T_{Agg} and goes to the ‘‘Waiting T_{Agg} ’’ state. While this node is in this state, it receives the readings collected by its descendents. So, after T_{Agg} expires, it collects environmental data, waits a random period of time, aggregates its collected data with the data sent by its descendents, sends a aggregation-packet to its parent and goes to the Idle state.

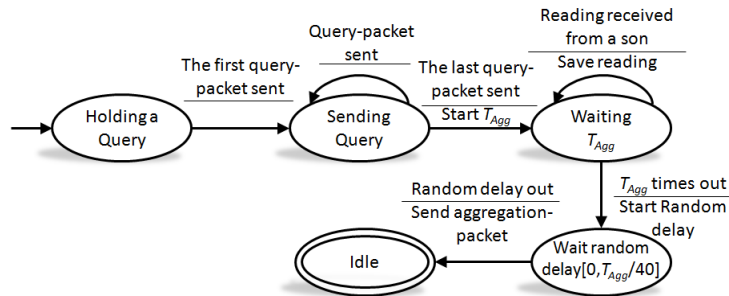


Figure 5.3. State diagram of Delayed Restricted Flooding.

5.4 Itinerary

This algorithm defines an itinerary within the RoI. It has a zigzag pattern, as illustrated in Figure 5.4. Since we consider RoIs having irregular shapes, this pattern is the most flexible among the patterns found in the literature [Xu et al., 2006]. Moreover, since nodes are deployed randomly, the path shown in Figure 5.4 is only the ideal itinerary used as a basis to generate the real itinerary.

The itinerary definition employs the ABF strategy, that is, it looks for active neighbors and chooses one of them to continue the query processing. However, a node on the itinerary does not use announce-packets. It sends the query to its neighbors and receives the packets with their readings (**reading-packets**). Next, it applies the aggregation operator on the data received from its neighbors and its own collected data, chooses the best neighbor, and sends an **aggregation-packet** with the partial result of the query. When a node receives an aggregation-packet addressed to it, this node realizes that it is the next node on the itinerary. Hence, it repeats the process. Figure 5.5 presents a state diagram that illustrates the process described above. First,

a node receives an aggregation-packet and goes to the **Holding Query** state. Thus, it broadcasts all the query-packets to its neighbors during the **Sending Query** state. Hence, it stays in the **Waiting for Packet-readings** state for T_{Wait} seconds, chooses the best neighbor and sends the aggregation-packet to the next node on the itinerary. T_{Wait} is the same timer used by ABF. The best value of this parameter was determined empirically using simulations.

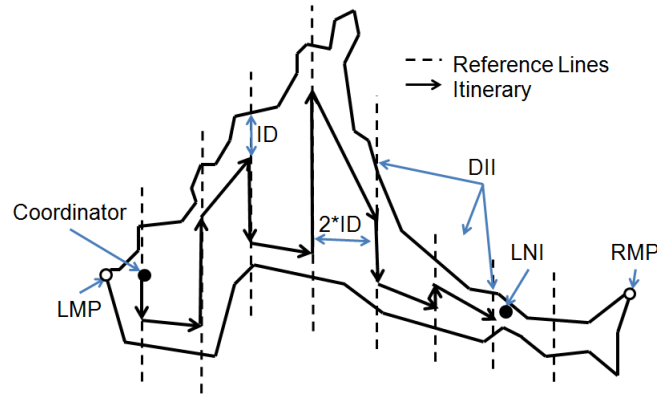


Figure 5.4. Itinerary definitions.

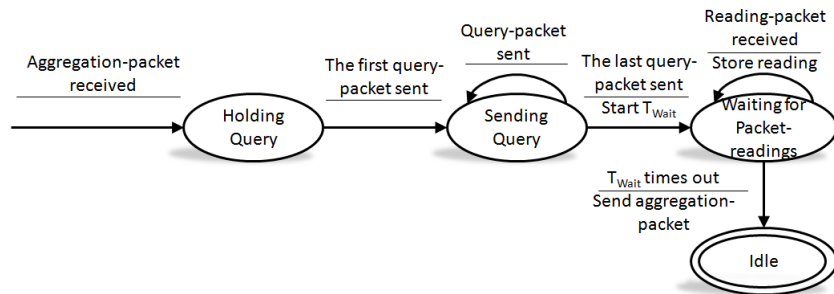


Figure 5.5. State diagram: dissemination using itinerary.

5.4.1 Itinerary Terminology

Before describing the algorithm, it is necessary to define some of the terms employed here. Figure 5.4 illustrates them. The **Itinerary Distance (ID)** is an adjustable parameter in the algorithm that defines the distance between parallel lines of the itinerary ($2 \times ID$) and between the itinerary and the boundaries of the RoI. The itinerary starts from left to right (other directions are possible with simple rotations in the X and Y axes). The proposed algorithm defines parallel guidelines, called **reference lines**, over which the itinerary will pass. Those lines are spaced by $2 \times ID$. The point where these

lines cross the RoI is called **Destination In the Itinerary (DII)**. The **Coordinator** initiates the itinerary. This node is chosen among the nodes that are inside the RoI whose distance from the **Leftmost Point (LMP)** – the point in the RoI with the smallest X value) is less than ID . The **Last Node on the Itinerary (LNI)** finishes the itinerary. It is defined by the halting criteria defined in the following subsection.

The itinerary is defined in a distributed way. A node chooses as the next node on the itinerary the neighbor closest to the current DII contained in the query. Moreover, when a node is closer to the DII than the distance ID , it changes the direction of the itinerary and calculates another DII. In order to do so, the following data need to be disseminated: points that define the RoI, the location of the Originator, Direction and the current DII. The **Location of the Originator** is the point where the query starts processing, and is disseminated because the LNI will forward the query to this point. **Direction** defines the current direction of the itinerary: **UP**, **DOWN** or **RIGHT** (the direction is represented as arrows in Figure 5.4).

Not all nodes participate in the itinerary. There are three types of nodes during dissemination: nodes on the itinerary and on the RoI (they send the query and their readings), nodes out of the itinerary and on the RoI (neighbor of a node on the itinerary - only send their readings to a node on the itinerary), and nodes out of the itinerary and out of the RoI (they ignore the query).

5.4.2 Itinerary Algorithm

This subsection describes how a node defines the next node in the itinerary. First, a node verifies if it is the last node in the itinerary, then it verifies if the direction of the itinerary must be changed and finally it chooses a node among its neighbors. Algorithm 4 performs such verifications. It is executed when T_{Wait} expires and a node on the itinerary must select the next node. First, it verifies if its distance to the RMP is smaller than the value of ID . If so, this node is the *LNI* and will start the Return stage (lines 2 and 3). If not, the node verifies its distance to the current *DII* (line 5). If this distance is smaller than ID , the node changes the current direction and *DII* (lines 6 up to 15), chooses the next node on the itinerary (line 16) and sends the aggregation-packet (line 17). The complexity of this algorithm is $O(n)$, where n is the number of neighbor of the node. It is determined by the function *choose_neighbor()*, which must return the id of the neighbor closest to the current DII.

The functions *next_down()*, *next_up()* and *next_right()* define new DIIs. They create the zigzag pattern of the itinerary. The new DII is the point where the reference line crosses the boundaries of the RoI. The reference line is a perpendicular line that

Algorithm 4 Next node itinerary definition.

```

1: procedure TDISS_OVER(qry)
2:   if my_distance_to(RMP)  $\leq$  ID then
3:     start_return_stage()
4:     return
5:   else if my_distance_to(qry.dii)  $\leq$  ID then
6:     if qry.direction = UP  $\vee$  qry.direction = DOWN then
7:       qry.dii  $\leftarrow$  next_right(qry.dii)
8:       qry.direction  $\leftarrow$  RIGHT
9:     else if qry.direction = RIGHT then
10:      if closest_top(qry.dii) then
11:        qry.dii  $\leftarrow$  next_down(qry.dii)
12:        qry.direction  $\leftarrow$  DOWN
13:      else
14:        qry.dii  $\leftarrow$  next_up(qry.dii)
15:        qry.direction  $\leftarrow$  UP
16:      next_neighbor  $\leftarrow$  choose_neighbor(qry.dii)
17:      send_aggregation_packet(next_neighbor)
18:      return

```

crosses the RoI twice, in an upper and in a lower point. The function *next_down*() maintains the reference line and defines the new DII as the lower point. The function *next_up*() also maintains the reference line and defines the new DII as the upper point. The functions *next_right*() defines a new reference line as a perpendicular line in a distance of $2 \times ID$ from the current DII. The new DII is the closest point from the current DII, between the two points on the reference line that crosses the RoI. The function *choose_neighbor*() return the identifier of the neighbor closest to the current DII. This node receives the aggregation-packet and realizes that it is the next node on the itinerary.

Figure 5.6(a) illustrates when a node receives a query close to the current DII. The query direction is UP, so the node executes the function *next_right*() (line 7). The new DII is the upper point where the new reference line crosses the RoI. The node also changes the query direction from UP to RIGHT (line 8). Figure 5.6(b) illustrates when the current direction is RIGHT. The node executes the function *next_down*() (line 11), which maintains the reference line, defines the new DII as the lower point where this line crosses the RoI, and changes the direction to DOWN (line 12).

The algorithm has two halting criteria: ideal and failure. The **ideal criterion** happens when the query reaches a node in which its distance to the RMP is smaller than ID. It means that the query covered the entire RoI. The **failure criterion** avoids loops in the itinerary. It finishes the itinerary before the query reaches the whole RoI, causing a reduction in the coverage. This criterion happens when the aggregation-packet reaches a node more than once for the same DII and all of its neighbors were

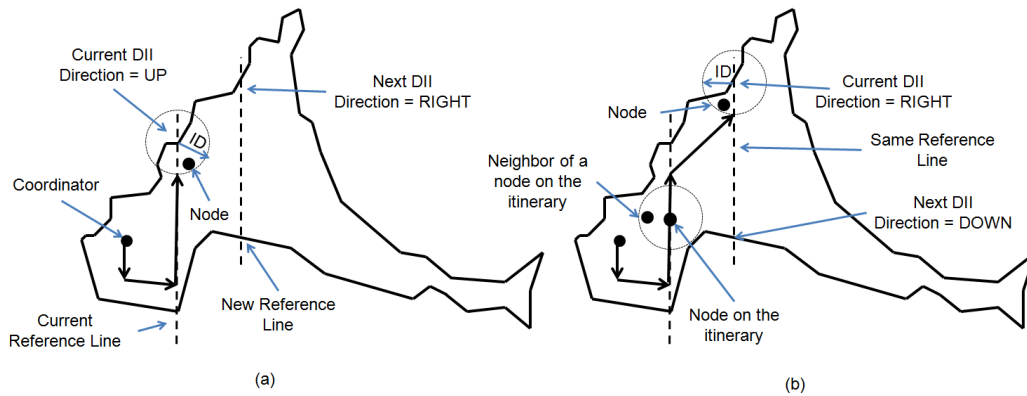


Figure 5.6. (a) Node calculate the new direction (RIGHT) and new DII (b) Node calculate the new direction (DOWN) and new DII

already chosen as the next node on the itinerary for this DII.

5.5 Chapter Remarks

This chapter presents three algorithms for the Dissemination and Aggregation stages. The proposed algorithms are Classic, Delayed Restricted Flooding and Itinerary. The two first algorithms disseminate using flooding inside the RoI and use a routing tree to aggregate the collected data. The last algorithm creates itineraries to transmit the query inside the RoI. All of them are able to process queries with RoIs forming polygons.

The Classic algorithm performs conventional flooding inside the RoI. In the Aggregation stage, each node waits for the reading from all its sons, aggregates the received data with its own reading and sends the result to its parent. The Delayed Restricted Flooding algorithm disseminates like the Classic algorithm does. However, each node waits a small period of time in which it receives readings from its sons. This period of time is inversely proportional to the node's level in the route tree. The Itinerary algorithm uses the ABF strategy to create a path (forming a zigzag) inside the RoI where the query will pass. Nodes out of the itinerary only send their readings to a node on the itinerary. A Node participating of the itinerary receives readings from their neighbors, aggregates them with its own reading and sends the result to the next node in the itinerary.

These algorithms are created in order to evaluate different types of dissemination/aggregation algorithms using RoIs forming polygons. The Classic Algorithm represents a naive strategy, since it follows the traditional flooding and its aggregation is very simple and intuitive. It is prone to packet collisions because nodes in a neighbor-

hood try to transmit queries at the same time. Collisions can decrease the algorithm performance since several packets can be lost. The Delayed Restricted Flooding algorithm is devised to deal with the effects of the collisions in the Dissemination stage. The Itinerary algorithm tries to reduce the amount of nodes that transmit the query, since only nodes on the itinerary transmit the query.

Chapter 6

Evaluation

The proposed algorithms were evaluated by simulations. This chapter presents the simulated experiments divided into three rounds. The first round evaluates the performance of the proposed spatial query mechanism using the algorithm created for the Pre-Processing stage, which was presented in Chapter 3. These experiments also evaluate the best parameter values for each proposed algorithm. In the second round of the experiments, our simulations consider WSN employing duty cycle algorithms in order to evaluate the algorithms proposed in Chapters 4 and 5. Finally, the last round of experiments evaluates the proposed mechanism in WSN having failure-prone nodes.

Section 6.1 presents the experimental setup. Section 6.2 describes the implementations created for this work and Section 6.3 presents the metrics considered here. Section 6.4 presents the first round of experiments, Section 6.5 the second round and Section 6.6 the third round. Finally, Section 6.7 presents our analysis of the experiments.

6.1 Simulated Experiments Design

This section presents the WSN and environmental characteristics considered in this work and the parameter configuration for all the simulated experiments.

6.1.1 Assumptions

We assume that all packet transmissions occur in broadcast, using symmetric links. Hence, all the neighbors of a sender receive a sent packet and are able to send packets to this sender. We consider WSN composed by Iris sensor nodes [Crossbow Technology, Inc., 2010a] running the BMAC medium access control protocol [Polastre et al.,

2004]. We chose this type of sensor nodes because we could empirically verify their characteristics in order to use them in the simulations. We assume that all sensor nodes are static and are able to obtain their location. This assumption is plausible since there are many location solutions for WSN [Wang et al., 2010]. The coordinates used are in the standard decimal degrees notation with four decimal places, giving a worst case accuracy of around 11m. The radio range (80 m), in the Table 6.1.2, was obtained empirically. We measured the indoor and outdoor radio range of real Iris sensor nodes (without obstacles) and obtained the average value. We consider that the sensing range of the nodes is 40 m, according to the definition presented in Section 2.3.

6.1.2 Simulation Setup

This work considers the two-ray ground propagation model for wireless transmissions [Rappaport, 2002]. We also consider the random deployment of 2000 sensor nodes in a uniform distribution. This deployment occurs on a square monitored area having one kilometer of side. We chose this area because it is large enough to contain a disaster, since initially we intended to process spatial queries in disaster regions. Due to the infinite possibilities for regions of interest, we chose a real contour, that of the lake in Figure 6.1 represented by 140 points, requiring 10 packets of 28 bytes to be transmitted.

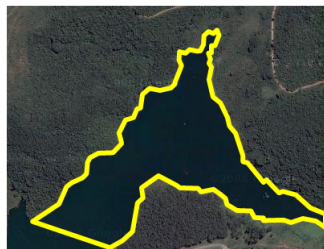


Figure 6.1. Irregular regions of interest over satellite photos (Lake).

Every point plotted on the graphics represents the mean of 60 simulations, with confidence interval of 95%. In each simulation, the WSN only processes one query. The query processing starts in a randomly chosen node . We do not consider that nodes consume energy when they are sensing, processing or sleeping. The nodes considered here have the characteristics shown in Table 6.1.2.

6.2 Implementations

The simulations of this work use the Network Simulator (NS) version 2.34 [ns, 2010]. In Chapter 5 we propose three algorithms for the Dissemination/Aggregation stages.

Parameter	Value
Radio range	80 m
Sensing range	40 m
TX power	0.048 W
RX power	0.051 W
Battery voltage	3 V
Payload length	28 bytes

Table 6.1. Iris Sensor Node Characteristics

Hence, we implemented three variations of the proposed spatial query processing mechanism. All of them use the ABF location-based protocol in their Forwarding and Return stages. However, each of them uses one of the proposed Dissemination/Aggregation algorithms. The variations of the spatial query processing mechanism are:

- Classic: uses Classic Restricted Flooding (Section 5.2);
- DRF: uses Delayed Restricted Flooding (Section 5.3), and;
- Itinerary: creates an itinerary on the RoI (Section 5.4).

We implemented SWIP-2 in order to compare the proposed algorithm against the state of the art of spatial query processing. This is a spatial query processing mechanism based on SWIP [Coman et al., 2007]. We chose SWIP because it is the only window query processing found in the literature that starts the process in any node. However, it does not consider RoIs with irregular shapes, duty cycle algorithms or packet collisions. SWIP uses a greedy strategy to forward the query to the RoI. This algorithm does not cope with holes in the network (definition in Subsection 4.5). In the Dissemination/Aggregation stages, it uses Classic and in the Return stage it uses the reverse route created during the Forwarding stage.

SWIP-2, on the other hand, uses GPSR [Karp and Kung, 2000] in the Forwarding stage, because this protocol can surround holes in the network. In the Dissemination/Aggregation stage it uses Delayed Restricted Flooding, since Classic does not present a good performance. In the Return stage, it creates new routes using GPSR. The reverse route created in the Forwarding stage should not be used because nodes can sleep. Moreover, all nodes using SWIP-2 transmit a beacon every 30 seconds in order to update the neighbor table used by GPSR. The original paper of GPRS [Karp and Kung, 2000] established 3 seconds as the beacon interval, however we increased the beacon interval since nodes do not move.

We also created an application in Java that enables the user to generate the RoI on top of satellite photos and performs the Pre-Processing algorithm that is described

in Chapter 3. In this application, the user chooses the picture file, creates the polygon that represents the contour of the RoI in the picture and executes the Pre-Processing algorithm. This algorithm simplifies the RoI, reducing the amount of points in the RoI representation in order to decrease the amount of packets necessary to transmit the query. The application asks the user to inform the number of packets that the query must occupy and generates a polygon (based on the RoI) that is used in the simulations.

6.3 Metrics

The experiments analyze four metrics: coverage area, end-to-end delay, energy consumption and query processing success. The **coverage area** represents the quality of the query result. This metric is the portion of the RoI that was covered by one or more sensor nodes during a query processing. The **end-to-end delay** is the time elapsed from the Originator receiving the query from the user and the reception of the query result. It measures the transmission time and the delays of the mechanism. For simplicity, we do not consider the time used for processing the query inside each node. **Energy consumption** is the energy consumed by the WSN as a whole in order to process one query. We do not consider the energy consumed when nodes are in sleep or idle states. In our conception, MAC protocols should manage the energy consumption during these states. The **query processing success** is the percentage of queries that the Originator node obtained a query result among all the requested queries. Hence, this metric represents the efficiency of the mechanism. The query processing success (*QPS*) is determined by the following equation:

$$QPS = \frac{NQR}{NQ} * 100 \quad (6.1)$$

where *NQR* is the number of query results obtained in the experiments and *NQ* is the number of queries processed.

In the graphics showing the metrics coverage area, end-to-end delay and energy consumption, we only considered data from successful queries processing. The data plotted in these graphics came from queries processing which the Originator obtained a query result. We ignored the data obtained from queries processing that did not presented a result.

6.4 Pre-Processing Algorithm and Parameter Evaluation

These experiments were devised to evaluate the energy consumption of the WSN using the proposed Pre-Processing algorithm and to analyze the best parameter values for the proposed Dissemination/Aggregation algorithms. The experiments ran the Pre-Processing algorithm in order to reduce the query size from 10 up to 1 packet. Hence, the graphics presented ahead have the X-axis varying between 1 and 10. Assuming packets having 28 bytes of payload, each packet can hold up to 14 points of the polygon that represents the RoI. We first evaluated the best value of $T_{Decrease}$ in DRF (Subsection 5.3) against Classic. Then, we evaluated the best value of T_{Max} and ID (Subsection 5.4.1) for Itinerary.

In order to approximate the experiments to the characteristics of real WSN, we employed a simple duty cycle control algorithm based on asynchronous MAC protocols, such as in [Polastre et al., 2004; Buettner et al., 2006; El-Hoiydi and Decotignie, 2004; Sun et al., 2008b]. All nodes have the same sleep and active times ($T_{off} = 0.5s$ and $T_{on} = 0.5s$, respectively), consequently they spend $T_{Cycle} = T_{off} + T_{on}$ seconds each cycle. However, the nodes operate in different phases in order to simulate the asynchronous MAC. In our simulations, during the network startup, each node chooses randomly its start time. Hence, these nodes transits between sleep and active mode in different times.

6.4.1 Evaluating Classic vs DRF

In this scenario we compared Classic against DRF using $T_{Decrease} = 0.5, 1.0$ and 2.0 seconds (described in Subsection 5.3). It is important to mention that Classic uses $T_{Leaf} = 5$ seconds and $T_{Son} = 5$ seconds (described in Subsection 5.2). These values were obtained empirically. Figure 6.2 presents the end-to-end delay. As expected, when we reduce the query size the end-to-end delay is also reduced. When the query size increases, the amount of packets used by the query increases too. Thus, nodes spend more time to transmit the query. This increases the end-to-end delay of all algorithms.

The experiments also show that Classic processes queries faster than DRF in practically all scenarios. It is because the timers used by Classic are smaller than the timer defined by the Coordinator (T_{Coord}), the longest timer used by DRF. Moreover, nodes using Classic can send aggregation-packets immediately after receiving all the aggregation-packets from their descendents. Using DRF, nodes wait T_{Agg} seconds before sending aggregation-packets. In the following graphics we show the tradeoff

between end-to-end delay and coverage area and between end-to-end delay and energy consumption.

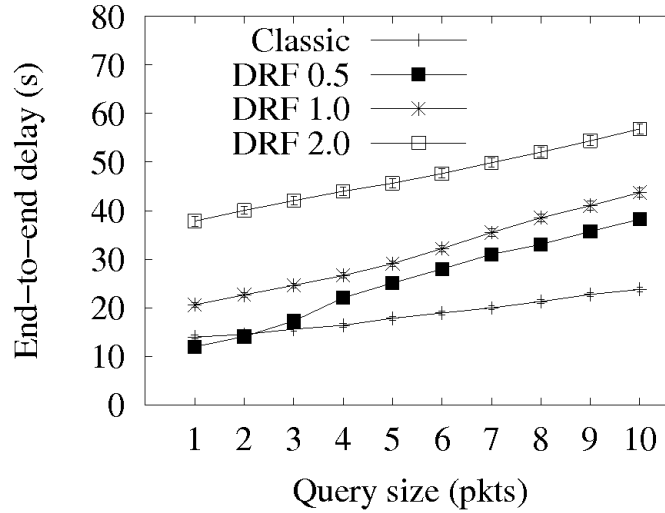


Figure 6.2. End-to-end delay of Classic and DRF using different values of $T_{Decrease}$.

Figure 6.3 presents the results concerning coverage area. DRF 2.0 presents the largest coverage area for all the analyzed query sizes. The query size does not influence DRF using this configuration. DRF 1.0, DRF 0.5 and Classic had their performance decreased due to packet collisions caused by the increased number of late packets (LAPs, Subsection 5.2). Classic is propitious to collisions since all nodes in a neighborhood try to retransmit the query at the same time. It decreases the number of descendents in the nodes' tables of sons, increasing the number of LAPs. When DRF uses small values of $T_{Decrease}$, several nodes do not define large enough aggregation timers. Hence, these nodes do not wait for all their descendents' packets before sending the aggregated readings. Thus, the number of LAPs in DRF 0.5 and DRF 1.0 increases for queries having more than 2 and 5 packets, respectively. Moreover, DRF using small values of $T_{Decrease}$ (0.5 and 1.0 seconds) also increases the number of collisions during the Aggregation stage. This happens due to the random delay used to avoid collisions before sending the aggregation-packet (Subsection 5.3). This delay is proportional to $T_{Decrease}$, so the collisions tend to increase if $T_{Decrease}$ is not large enough.

The Pre-Processing algorithm presents substantial energy savings for all analyzed algorithms, as shown in Figure 6.4. The reduction of the query size decreases the amount of packets transmitted by the WSN and consequently the energy consumption. However, LAPs increase the energy consumption of the query processing. If $T_{Decrease}$ is large enough, all nodes will wait for all their descendents' readings and will transmit

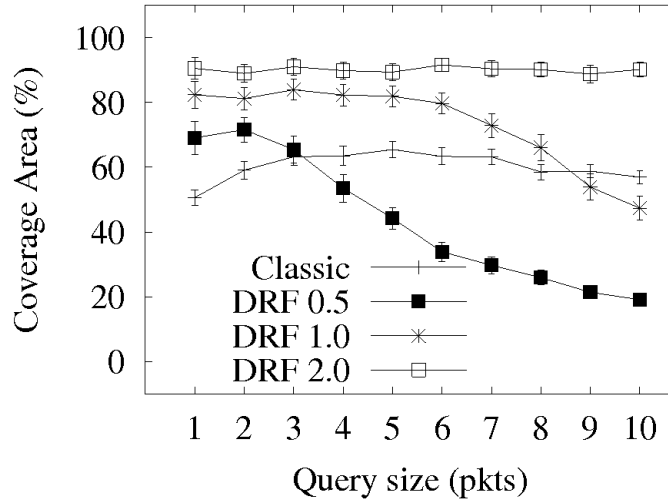


Figure 6.3. Coverage area of Classic and DRF using different values of $T_{Decrease}$.

aggregation-packets only once. But, if $T_{Decrease}$ is small, nodes can transmit several LAPs, increasing the energy consumption. DRF 0.5 and 1.0 present low energy consumption when the query is small. However, when the query size increases, the value of $T_{Decrease}$ is not sufficient for receiving the readings of all the descendents. Hence, the number of LAPs increases. The increment in the energy consumption accentuates for queries using more than two and five packets in DRF 0.5 and 1.0, respectively. Since the number of LAPs is small for DRF 2.0, the energy consumption is also small.

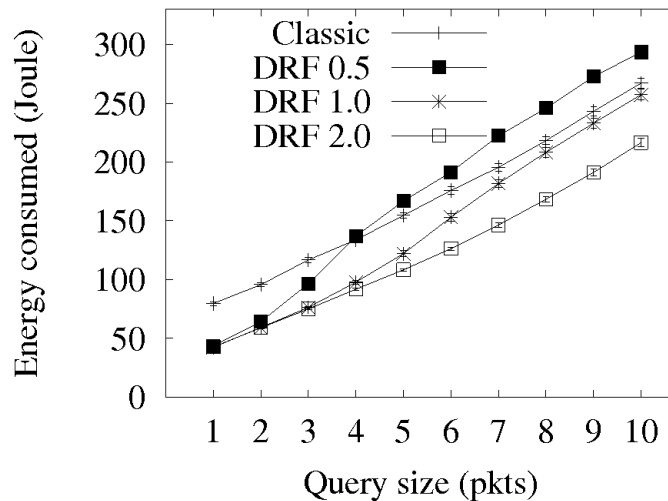


Figure 6.4. Energy consumption of Classic and DRF using different values of $T_{Decrease}$.

Analyzing the proposed Pre-Processing algorithm, these scenarios show that this

algorithm saves up to 80% of the energy consumed during the query processing, decreases the end-to-end delay and does not influence the coverage area when the parameter $T_{Decrease} = 2.0s$. Its performance is due to the decreased amount of packets transmitted during the query processing. We verify that the best configuration of DRF occurs when $T_{Decrease} = 2.0s$. The other configurations increase the number of LAPs and, consequently, increase the energy consumption and decrease the coverage area. In experiments not presented here, we increased the value of $T_{Decrease}$ to 3, 4 and 5 seconds. The end-to-end delay increased during these experiments, however the energy consumption and coverage area were practically the same. Classic presents the smallest end-to-end delay, however the amount of collisions increases its energy consumption and decreases its coverage area.

6.4.2 Evaluating the Itinerary

These experiments were performed in order to evaluate the best configuration of the Itinerary algorithm defined in Subsection 5.4 and the influence of the proposed Pre-Processing algorithm in the Itinerary performance. Here, we also use $T_{off} = 0.5$ seconds and $T_{on} = 0.5$ seconds for the duty cycle. The parameters that most affect the performance of the query processing are the itinerary distance (ID - Subsection 5.4.1) and the maximum delay to send an announce-packet (T_{Max} - Subsection 4.4). ID defines the distance between two parallel lines on the itinerary as well as the distance between the itinerary and the bounds of the RoI. The value of T_{Max} influences the end-to-end delay and the energy consumption, since larger values of T_{Max} reduce the amount of announce-packets.

6.4.2.1 The Longest Waiting Time

Figure 6.5 presents the end-to-end delay of Itinerary using different values of T_{Max} and query sizes. The curves of the graphics represent $T_{Max} = 0.2, 0.4, 0.6, 0.8$ and 1.0 seconds. When T_{Max} and the query size increase, the end-to-end delay also increases, as expected. However, the linear increase of the end-to-end delay shows that T_{Max} influences the most, since this parameter must be large enough for nodes to send the query and receive all readings from their neighbors. T_{Max} also influences the processing success. The figure shows that T_{Max} using 0.2 and 0.4 seconds are not able to process queries larger than 4 and 8 packets, respectively. This happens in these scenarios because a node does not have enough time to disseminate the query and wait for the readings from its neighbors. So, a node can consider that it has no neighbor to forward the query, hence the query processing finishes without a query result.

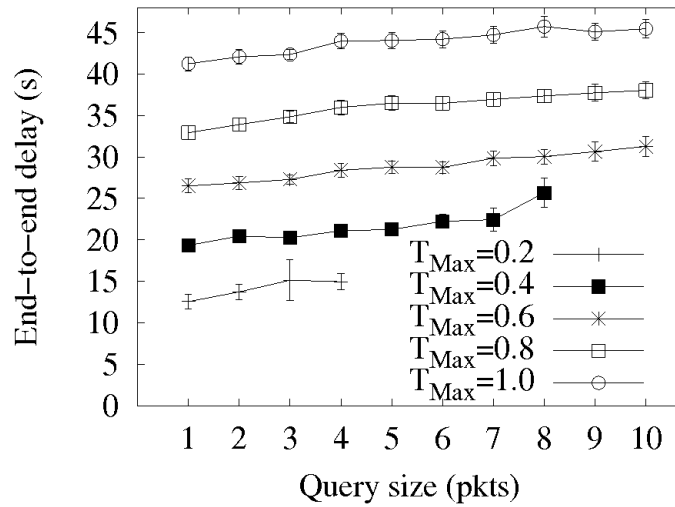


Figure 6.5. End-to-end delay of Itinerary.

The coverage area of the queries as function of the query size having T_{max} as parameter is showed in Figure 6.6. The Pre-Processing algorithm practically does not influence this metric in the analyzed scenarios. Moreover, different values of T_{Max} also presented practically the same coverage area. The performance of the itinerary algorithm decreases only for $T_{Max} = 0.2$ and $T_{Max} = 0.4$ when the query size is larger than 4 and 8, respectively. It occurs because the value of T_{Max} is not enough for all nodes to receive announce-packets during the Forwarding and Return stages, halting the query processing.

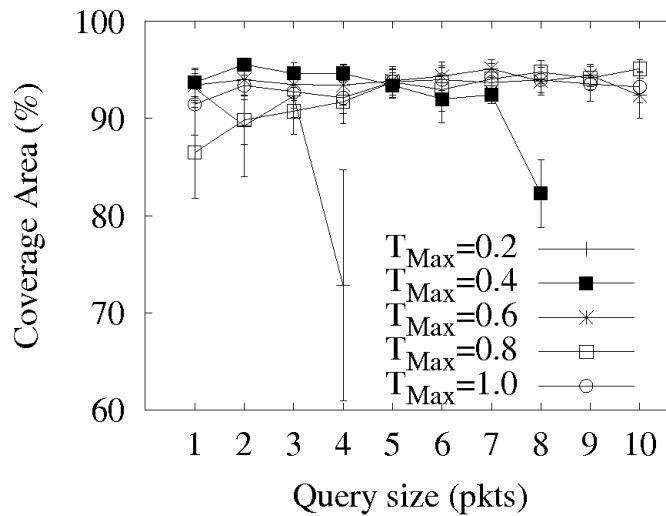


Figure 6.6. Coverage area of itinerary.

Figure 6.7 presents the energy consumption. The Pre-Processing algorithm saves around 60% of energy during query processing, for all values of T_{Max} . Itinerary using $T_{Max} = 1.0$ has the smallest consumption for small queries, because this value reduces the number of announce packets. Nodes holding a query send the query packets to the first node that sent an announce-packet. When a node has an announce-packet scheduled to be sent and it hears a query packet not directed to it, this node cancels the packet transmission (Subsection 4). Hence, large values of T_{Max} increase the amount of announce-packet cancellations, consequently reducing the energy consumption.

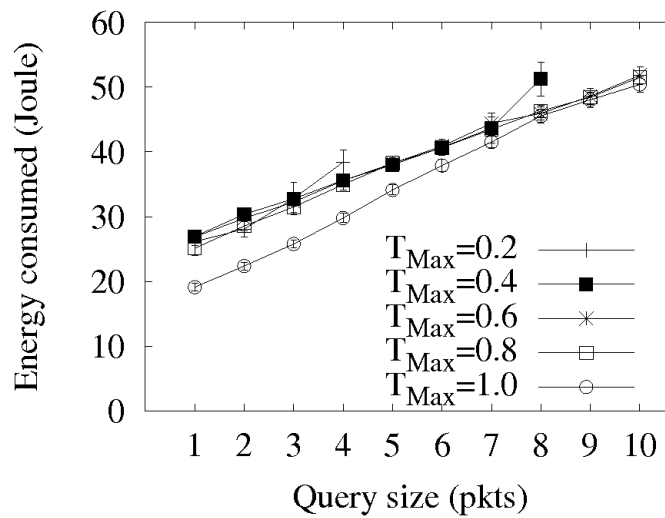


Figure 6.7. Energy consumption of itinerary.

We use $T_{Max} = 1.0$ second for the remainder of the experiments. This value increases the end-to-end delay, but presents the smallest energy consumption and the coverage area is equivalent to that of other values of T_{Max} . We also performed experiments using bigger values of T_{Max} . These experiments are not presented to improve the readability of the text, since the energy consumption and coverage area do not change significantly when compared to $T_{Max} = 1.0$ second.

6.4.2.2 The Value of Itinerary Distance - ID

Here we evaluate the Pre-Processing algorithm and the impact of the parameter ID (Subsection 5.4.1) in the performance of Itinerary. The results presented in the following graphics considered the radio range $r = 80$ meters. Thus, varying the value of the parameter ID we also vary the reason $R = ID/r$. R is the most important value for the network performance. It represents the relation between the format of the itinerary and the radio range. Hence, if we consider experiments using the values for R used in

the experiments presented here and different values for r and ID , the results will be the same. Thus, we varied only the value of ID during the experiments.

The end-to-end delay of the mechanism is presented in Figure 6.8. The Pre-Processing algorithm reduces only about 5% of the end-to-end delay. Increasing the value of ID , the end-to-end delay decreases because this parameter defines the distance between parallel lines in the itinerary. Hence, if ID increases, the number of hops on the path followed by the query during the dissemination will decrease, thus decreasing the end-to-end delay.

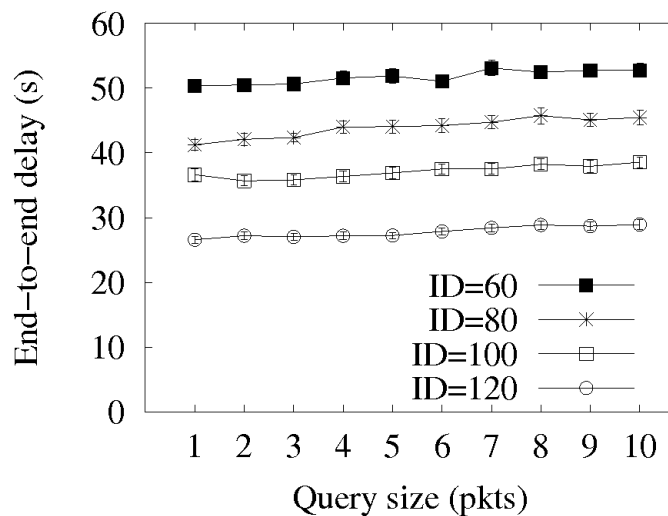


Figure 6.8. End-to-end delay for different values of ID .

The coverage area is presented in Figure 6.9. The Pre-Processing algorithm does not influence the coverage area. However, the value of ID influences directly this metric. Itinerary using $ID = 60$ m presents the best performance, since the parallel lines in the itinerary are closer to each other. When $ID = 120$ m, the mechanism presents the smallest coverage area. It occurs because queries cannot reach some sections of the RoI due to the large distance between parallel lines of the itinerary. $ID = 100$ m and $ID = 80$ m presented practically the same performance.

The energy consumption is directly affected by ID . Using higher values of this parameter, the path followed by the query decreases, reducing the energy consumption. However, these values also decrease the coverage area, since higher ID increases the distance between the parallel lines in the itinerary and, consequently, reduces the amount of nodes that receive the query during the Dissemination stage. Analyzing the graphics, we verify that using $ID = 120$ m the itinerary presents the smallest energy consumption and end-to-end delay. However, it also presents the smallest cov-

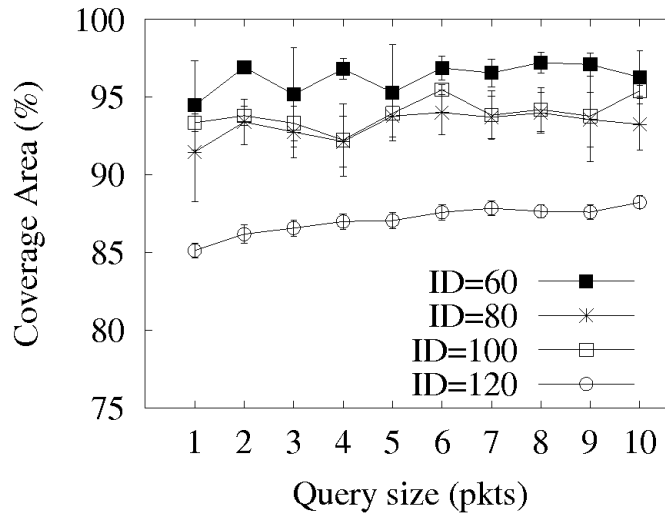


Figure 6.9. Coverage area for different values of ID.

verage area. We verify that $ID = 100$ m (a medium value) is the best value for this parameter. Using this value of ID, the itinerary presents the second smallest energy consumption, while the coverage area is higher than 90%.

It is important mention that the performance of the mechanism varying the values of ID depends on the radio range. During these experiments, the radio range was 80 m. Hence, using $ID = 80$ m is equivalent to say that ID is 100% of the radio range. These rate influences the number of nodes receiving queries. When this rate increases, nodes in the middle of parallel lines do not receive queries, since the distance between these lines is higher than the radio range. Decreasing this rate, nodes on the itinerary can transmit packets to nodes in the middle of two parallel lines. Thus, the number of nodes receiving the query tends to increase. However, when we used $ID = 100$ m, i.e. 125% of the radio range, some nodes did not receive the query, but this number of nodes was not significant. We verified that 125% is the best rate between ID and radio range.

6.5 Varying the Duty Cycle

These experiments compare the proposed mechanism against a spatial query processing mechanism that represents the state of the art. Moreover, we evaluate the performance of both mechanisms in different duty cycles. We present three algorithms in the graphics: DRF, Itinerary and SWIP-2. During these experiments, we fix $T_{Cycle} = 1$ second and vary T_{on} from 0.1 up to 1.0 second. Thus, we vary the duty cycle from 10% up to

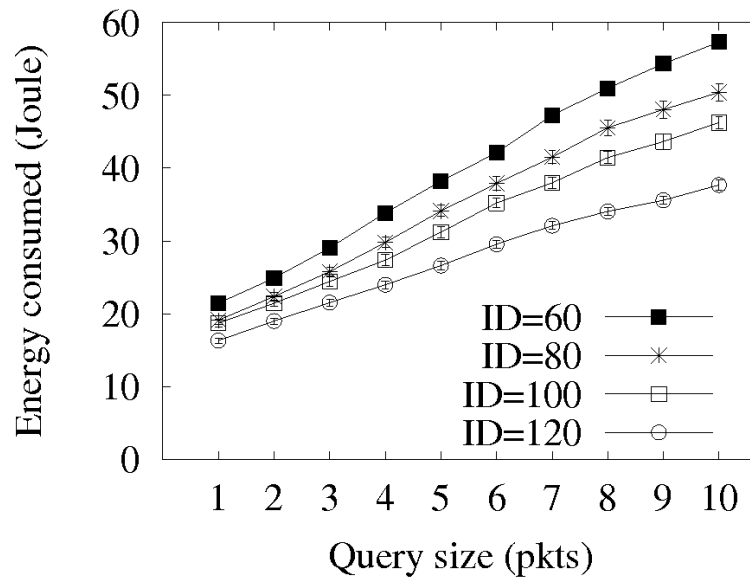


Figure 6.10. Energy consumption for different values of ID.

100% of T_{Cycle} . The X-axis of all graphics represent this variation. In order to evaluate all stages of spatial query processing, we fixed the Originator in the top left corner of the monitored area. This forces the mechanisms to execute the Forwarding and Return stages, otherwise the Originator could be inside the RoI and start processing queries in the Dissemination stage.

Figure 6.11 presents the query processing success of the three algorithms. When the duty cycle is smaller than 60%, SWIP-2 is not able to perform the Forward and Return stages since the query and the query result are often sent to sleeping nodes and, consequently, the query processing fails (Subsection 4.7). Hence, in the following graphics, the line that represents SWIP-2 starts in 60% since there is no query completed with smaller duty cycles. DRF and Itinerary presented more than 90% of query processing success in practically all scenarios, since they use only active nodes. When the duty cycle is smaller than 10%, Itinerary processes less than 80% of the queries. This is because the number of active nodes influences the Itinerary definition.

Figure 6.12 shows the end-to-end delay of the three algorithms. Itinerary presents the worst performance since the value of T_{Wait} (which was set to 2 seconds) adds a delay for each hop on the Forwarding, Dissemination and Return stages. Thus, this timer increases the time necessary to forward the query and the query result. DRF presents the smallest end-to-end delay because it uses ABF in the Forward and Return stages and does not need to transmit periodic beacons. SWIP-2 needs to transmit beacons during the query process in order to update the neighbor tables of GPSR. Hence, it

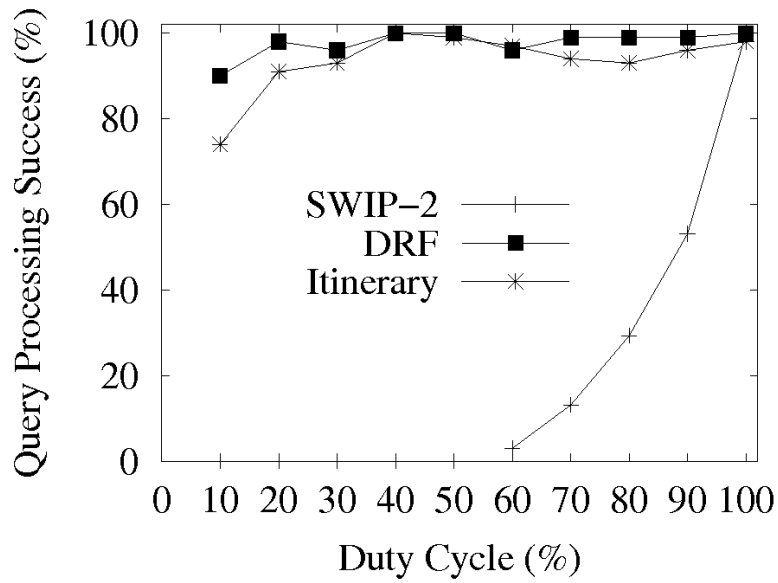


Figure 6.11. Comparing the query processing success of SWIP-2, DRF and Itinerary.

suffers from congestion, which increases the amount of packet collisions during the Dissemination stage, increasing the end-to-end delay.

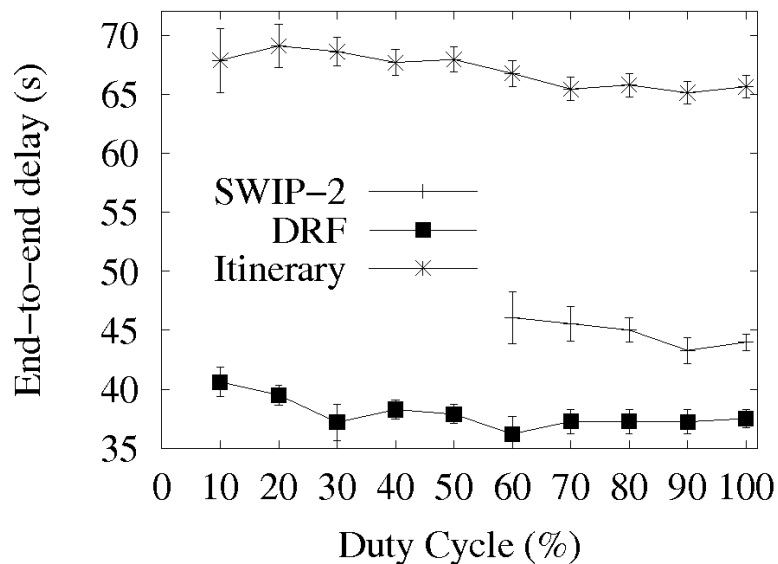


Figure 6.12. Comparing the end-to-end delay of SWIP-2, DRF and Itinerary.

The coverage area of the three algorithms is shown by Figure 6.13. SWIP-2 presents a coverage area next to 100% with 60% and 70% of duty cycle. When the duty cycle increases, the amount of packet collisions increases too, since more nodes are

active during the query processing. Hence, the coverage area decreases. DRF presents only 40% of coverage area with duty cycle in 10% because there are few nodes active to disseminate the query. However, with duty cycle in 20% more nodes stay active during the query dissemination. Thus, the coverage area grows to around 90% and maintains this value for larger duty cycles. The coverage area of Itinerary increases if the duty cycle increases, since there are more active nodes to create the itinerary. However, this mechanism presents coverage larger than 90% only for duty cycles larger than 30%. When small duty cycles are used the nodes have less active neighbors. Hence, the second halting criterion occurs more often, reducing the coverage area (Subsection 5.4.2).

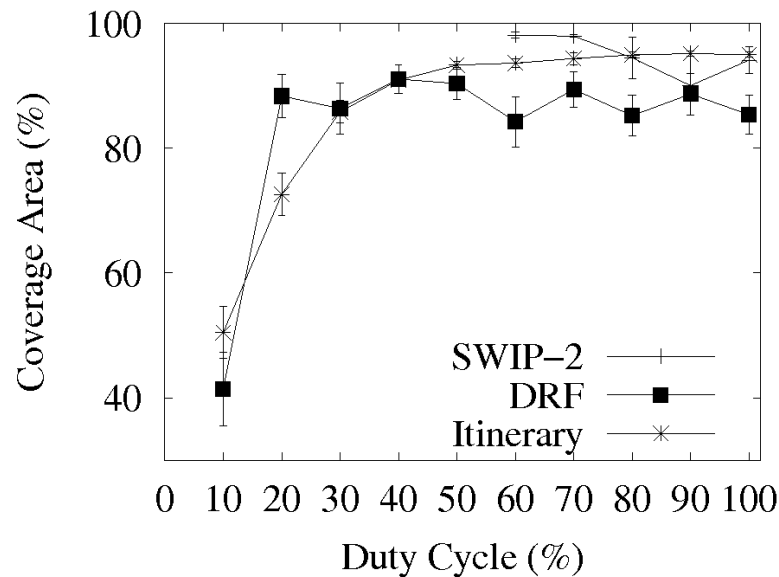


Figure 6.13. Comparing the coverage area of SWIP-2, DRF and Itinerary.

The energy consumption is presented in Figure 6.14. The beacon transmissions increase the consumption of SWIP-2 since nodes that do not participate in the query processing also transmit and receive packets. Using DRF and Itinerary, only nodes that participate in the query processing transmit packets. Itinerary presents the best results because the amount of nodes transmitting the query during the Dissemination stage is smaller than DRF, since only nodes on the itinerary transmit queries. Moreover, DRF suffers from unnecessary packet receptions during the Dissemination stage since a node can receive the query from several sources. It is important to mention that DRF and Itinerary present energy consumption 75% and 85% smaller than SWIP-2, respectively.

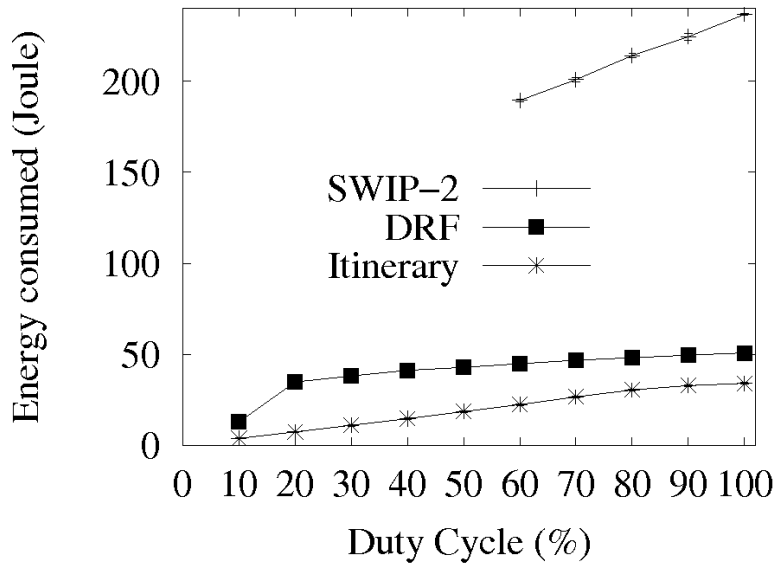


Figure 6.14. Comparing the energy consumption of SWIP-2, DRF and Itinerary.

6.6 Evaluation Under Node Failures

These experiments evaluate the proposed algorithm in WSN having failure-prone nodes. We used the Pre-Processing algorithm in order to reduce the query size from 10 up to 1 packet. In order to evaluate all stages in the query processing, the queries start processing in the top left corner of the monitored area, such as in Section 6.5.

6.6.1 Failure Model

This section considers transient and permanent failures caused by node failures, interference, node crashes or energy depletion, which are common events in disaster situations. Transient failures are failures that occur in a finite interval of time. Nodes stay inactive during failures and then resume their normal activities. When a permanent failure occurs, the node does not resume its activity. It stays inactive until the end of the WSN operation.

We simulate these failures turning off the radios of the nodes to disable the transmission and reception of packets. In order to simulate permanent failures, each node chooses randomly if it will fail and the instant of the failure during the query processing. Transient failures are implemented as cycles. When a node starts a new cycle, it chooses randomly if it will fail and the length of its cycle. We establish as parameter the failure probability ($10\% \leq P_{Fail} \leq 90\%$) and the cycle length ($15s \leq T_{Fail} \leq 20s$). According to [Macedo et al., 2005], this model may simulate failures caused by acci-

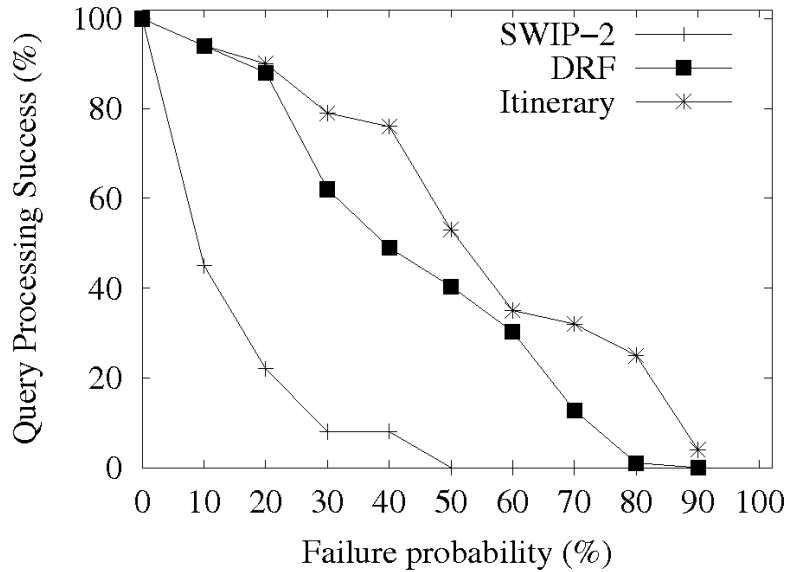


Figure 6.15. Query processing success under transient failures.

dental breakage, mobile sources of interference, processor crashes, collision attacks and sinkhole attacks.

6.6.2 Performance Evaluation Under Transient Failures

Figure 6.15 presents the percentage of successful queries processed by the three mechanisms. The performance of SWIP-2 decreases with the increase of the failure probability. This is because queries are sent to failed nodes, since the neighbor tables do not update as fast as the network topology changes. Nodes cannot process queries when the failure probability is bigger than 50%. DRF and Itinerary do not guarantee that the query processing will retrieve a result in all scenarios, because nodes in the Forwarding and Return stage can receive the query and fail two or more times consecutively, causing the query processing to stop (Subsection 4.6). Itinerary presents the best performance since queries are sent only to active nodes and nodes can retransmit queries upon detecting a failure.

Figure 6.16 shows the end-to-end delay. When the failure probability increases, Itinerary spends more time to disseminate the query since nodes need to retransmit the query several times due to the failure detection used by Itinerary (described in Subsection 4.6). For failure probabilities higher than 50%, Itinerary finishes the dissemination before it covers all the RoI due to the second halting criterion (Subsection 5.4.2). Hence, the end-to-end delay grows for failure probabilities smallest than 50%

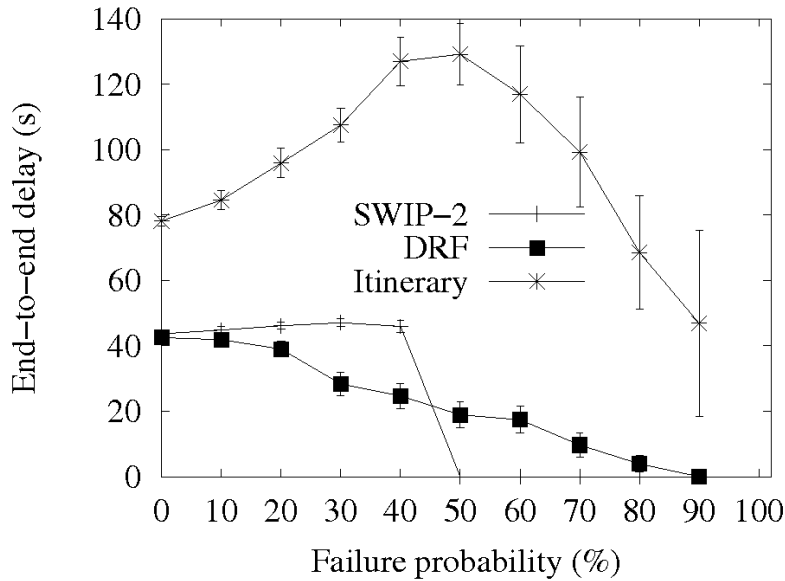


Figure 6.16. End-to-end delay under transient failures.

and reduces for values higher than 50%. DRF and SWIP-2 decrease the end-to-end delay when the failure probability increases because the amount of active nodes to disseminate the queries decreases.

The coverage area of the three mechanisms is presented in Figure 6.17. DRF and SWIP have their performance compromised due to nodes that fail during the Aggregation stage. When nodes next to the Coordinator fail, all data received from its descendents is lost. Moreover, they suffer from collisions during the Dissemination and Aggregation stages, reducing the amount of received readings. Itinerary is less prone to collisions since only nodes on the itinerary transmit the query. Furthermore, it uses only active nodes and can retransmit queries during dissemination, increasing the coverage area.

The energy consumption is presented in Figure 6.18. All mechanisms decrease the energy consumption when the failure probability increases because the amount of active nodes decreases. SWIP-2 consumes more energy than the others due to beacon messages transmitted during the query processing. Nodes that do not participate in the processing also consume energy. Using DRF and Itinerary, only nodes that participate in the query processing transmit and receive packets, since the nodes do not need to update their neighbor tables. Itinerary presents the best performance since it uses less nodes than DRF during the Dissemination stage.

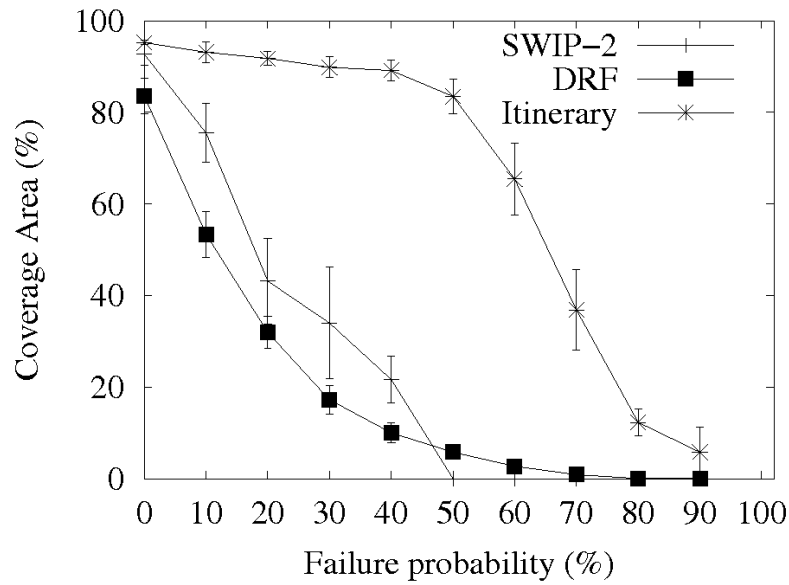


Figure 6.17. Coverage area under transient failures.

6.6.3 Performance Evaluation under Permanent Failures

During these experiments, the three algorithms obtained practically the same performance presented under transient failures. The main difference is the percentage of successful queries of DRF, as illustrated by Figure 6.19. Itinerary halts the dissemination if two consecutive nodes on the itinerary fail at the same time. DRF does not

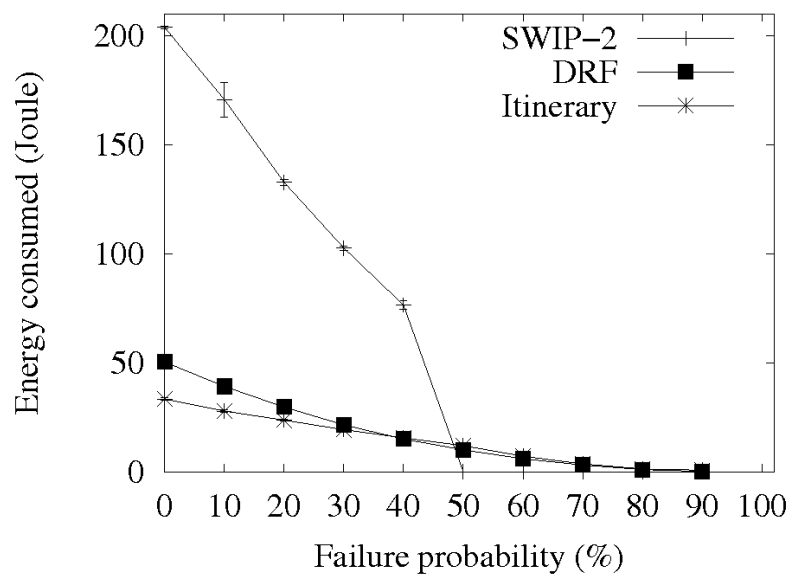


Figure 6.18. Energy consumption under transient failures.

suffer from this kind of failure since queries are disseminated by Restricted Flooding. Hence, Restricted Flooding outperforms Itinerary.

6.7 Simulated Experiment Remarks

We did three simulations studies (three rounds). **The first round of experiments** were performed in order to obtain the best parameter values for the proposed mechanism and to evaluate the performance of the Pre-Processing algorithm. We evaluated three variations of the proposed mechanism: Classic, DRF and Itinerary. The Classic mechanism presented the smallest coverage area and the highest energy consumption in practically all scenarios. Thus, we did not consider this variation of the proposed mechanism in the second and third rounds of the experiments. The parameter value that provided the best performance for DRF was $T_{Decrease} = 2.0$ seconds. The Itinerary's parameter values that provided the best performance were $T_{Wait} = 1.0$ seconds and $ID = 100$ meters. These parameter values were used in the remainder of the experiments.

Considering the best configuration of the mechanism variations, the proposed Pre-Processing algorithm reduced substantially the energy consumption of DRF and Itinerary. Furthermore, this algorithm decreased the end-to-end delay and did not decrease significantly the coverage area. With this algorithm we reached one of the objectives of this thesis, which is to provide an energy-efficient mechanism to process

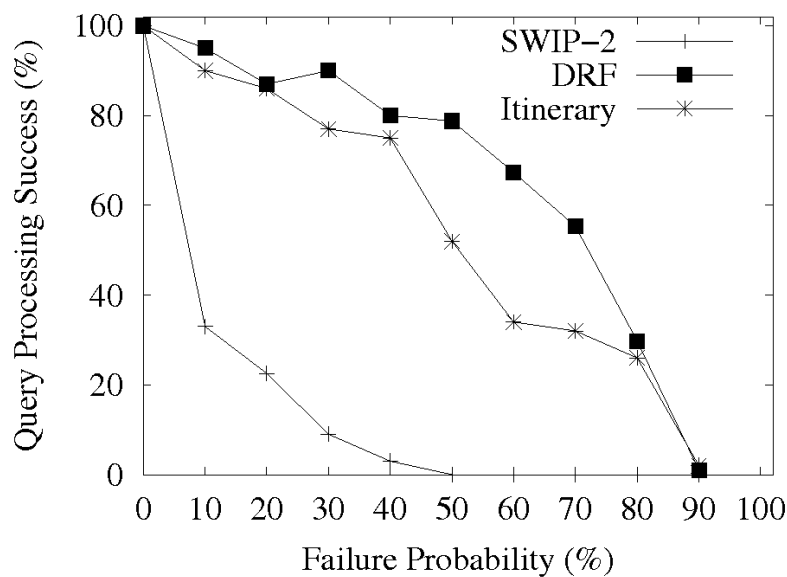


Figure 6.19. Query processing success under permanent failures.

spatial queries having regions of interest shaped as polygons.

We are sure that the performance improvement achieved by the proposed Pre-Processing algorithm is not so significant when the polygon representing the RoI is formed by a small amount of points. When a query has this type of RoI, it occupies less packets. Thus, the reduction provided by the Pre-Processing algorithm is not significant. If we apply this algorithm in RoIs with rectangular or circular shapes (such as the queries found in the literature), we will verify no performance improvement. This is because these queries occupy only one packet, hence the algorithm cannot reduce the number of packets employed to transmit these queries. However, as larger the number of points forming the polygon that represents the RoI more efficient is the spatial query processing using the proposed Pre-Processing algorithm.

The second round of experiments showed the performance of the proposed mechanism in WSN having inactive nodes due to duty cycles. SWIP-2 is not prepared to deal with sleeping nodes, thus it processed no queries in the majority of the scenarios. Moreover, it consumed about four times more energy than the proposed mechanism. Itinerary presented the smallest energy consumption and the best coverage area in the majority of the scenarios. Furthermore, it also presented the biggest end-to-end delay. However, DRF presented the best performance for all metrics in the smallest duty cycle. Hence, we verify that DRF is the best algorithm for small duty cycles, while Itinerary is the best for large duty cycles.

DRF and Itinerary consumed less energy than SWIP-2 mainly due to the ABF routing protocol used in the Forwarding and Return stages. SWIP-2 uses the GPSR routing protocol, which creates routing tables in order to obtain the next node in the route. Thus, all nodes must transmit beacon messages periodically to update their neighbor tables. This maintenance consumes a substantial part of the energy supply when the network topology is dynamic. The ABF protocol does not maintain neighbor tables, thus nodes do not transmit periodic beacons.

The third round of experiments evaluated the proposed mechanism in WSN having failure prone nodes. We analyzed transient failures (nodes fail during a finite period of time) and permanent failures (nodes fail and stay inactive up to the end of the simulations). The performance of DRF was similar for all metrics in both types of failures, except concerning the query processing success. DRF presented the best query processing success for permanent failures and Itinerary presented the best for transient failures. Itinerary also presented similar performance for both types of failures. Comparing these two variations of the proposed mechanism, Itinerary presents the best coverage area and energy consumption, but its end-to-end delay was the largest, as expected. SWIP-2 is not able to deal with failures, hence its performance

Table 6.2. Experiments Summary

	Duty Cycle	Transient Failures	Permanent Failures
Query processing success	DRF presented the best performance for duty cycle values smaller than 20% and performance equivalent to Itinerary for other scenarios. SWIP-2 only processed queries when duty cycle was higher than 60%.	Itinerary presented the best performance. SWIP-2 only processed queries when the failure probability was smaller than 50%.	DRF presented the best performance. SWIP-2 only processed queries when the failure probability was smaller than 50%.
End-to-end delay	DRF presented performance 15% smaller than SWIP-2, while Itinerary presented performance 50% higher than SWIP-2	DRF presented the best performance. SWIP-2 only processed queries when the failure probability was smaller than 50%.	DRF presented the best performance. SWIP-2 only processed queries when the failure probability was smaller than 50%.
Coverage area	Itinerary presented the best performance in the majority of the scenarios.	Itinerary presented the best performance in all scenarios.	Itinerary presented the best performance in all scenarios.
Energy consumption	Itinerary presented the best performance in all scenarios.	Itinerary presented the best performance when the failure probability was smaller than 40% and performance equivalent to DRF for bigger failure probabilities.	Itinerary presented the best performance when the failure probability was smaller than 40% and performance equivalent to DRF for bigger failure probabilities.

decreases fast when the failure probability increases. GPSR did not manage node failures, thus nodes would send packets to failed nodes, finishing prematurely the query processing. This mechanism consumed up to 400% more energy than Itinerary and DRF due to beacon transmissions.

We summarize the simulated experiment results in Table 6.2. It briefly discusses the performance of SWIP-2, DRF and Itinerary regarding to the considered metrics and scenarios. Each line presents a metric and each column presents a scenarios.

For simplicity, in the simulated experiments we did not consider WSN processing more than one query at the same time. Each query is processed separately, thus we did not evaluate the effect of one query processing on another query processing. We believe that the number of collisions would increase if more than one query are processed in parallel in the sensor network, since the amount of nodes transmitting packets also increases. However, we also believe that the proposed Itinerary mechanism would

present better performance than the other mechanism. Using Itinerary, the query processing passes from a node to the next node, hence only the neighbors of the node holding the query receives and transmits packets. Using Classic, DRF or SWIP-2, nodes flood the query inside the RoI. Thus, several nodes transmit and receive packets at the same time, increasing the probability of collisions if the WSN process more than one query at the same time.

This work considers the two-ray ground radio propagation model. However the literature presents several propagation models, which could be used to incorporate real characteristics in the simulated WSN [Scott et al., 2006a]. We could consider propagation over irregular terrain (which causes irregular attenuation of the signal), obstacles and near ground deployment of the nodes (which cause reflections) [Scott et al., 2006b]. Moreover, this work considers symmetric links. Asymmetric links could change the network topology and influence the behavior of the nodes during the query processing. All these radio characteristics are found in real WSN and, consequently, can influence the spatial query processing.

A more detailed analysis of the energy consumption of spatial query processing mechanisms can be performed using metrics such as network lifetime, number of sent/received packets and number of packet collisions. The network lifetime would be useful in longlife queries, since this type of query is processed for a long time in the WSN. The number of sent/received packets shows how the sensor network spends its energy, hence new solutions would concentrate effort where is more necessary. As an example, algorithms for the Dissemination stages can change the sleep schedule, avoiding the reception of the query more than once in nodes that have received the query and knows when their neighbors will transmit it again. Finally, the number of packet collisions would show if the spatial query processing mechanism is spending energy with packet retransmissions.

Chapter 7

Final Remarks

This chapter presents the thesis conclusions and future directions. We analyze the contributions and present open issues concerning spatial query processing in WSN. Section 7.1 presents the thesis conclusions. Section 7.2 presents our future works in spatial query processing. Finally, Section 7.3 lists the publications we achieved during the PhD.

7.1 Conclusions

In the majority of WSN applications found in the literature, the parameters for data collection are established before node deployment. These applications have nodes using predefined cycles for data collection, and data transmission either occurs periodically or responding to events. However, recent works have presented applications in which users inform the parameter for data collection during WSN operation. These applications enable users to create queries that establish what data will be collected, the frequency of data collection and where the collection will be performed. Queries which employ location information in order to express the users' interest are called spatial queries. The literature presents some works describing spatial query processing mechanisms for WSN. However, these works are not conclusive.

The first contribution of this thesis was a survey on spatial query processing for WSN. We divided and formally described the spatial query processing into six atomic stages: Pre-Processing, Forwarding, Dissemination, Sensing, Aggregation and Return. This division simplifies the understanding of spatial query processing mechanisms, facilitates the detection of problems and simplifies the proposal of new solutions. Then, we described the algorithms found in the literature for each of these stages and presented an overview of the algorithms that create distributed spatial indexes for WSN.

Afterwards, we described the most relevant spatial query processing mechanisms and how they implement each of the query stages. We verified that the related works consider only RoIs forming rectangles or circles. To the best of the authors' knowledge, no spatial query processing mechanism considers RoIs forming polygons. This shape can represent real objects plotted on maps or satellite photos and expresses better the regions where data will be collected. Furthermore, most of these works assume that all nodes are always available for data transmission and reception. However, nodes can sleep to save energy due to duty cycle algorithms or may fail due to several factors. Thus, we verified that the spatial query processing mechanisms must consider duty cycles and node failures in order to work properly in traditional WSN.

Another contribution was an algorithm for the Pre-Processing stage. It was devised to reduce the energy consumption of spatial query processing. This work considers queries having RoIs forming polygons. The amount of points in the polygon representation may be large, thus queries can occupy various packets. WSN that process this type of query tend to consume more energy, since each query is transmitted several times during its processing. The proposed algorithm simplifies the polygon representation in order to reduce the amount of packets occupied by the query. The experiments showed that our algorithm reduces the network energy consumption from 60% up to 80% and presents less than 5% of loss of precision in the query result.

This thesis also proposed ABF, a location-based routing protocol, which was used in the Forwarding and Return stages of spatial query processing mechanisms. It was devised to create routes in WSN having failed or inactive nodes. Using the protocol, a node holding a packet to forward first requests candidates among their neighbors and then forwards the packet to the first neighbor that replies to the request. Since failed and inactive nodes do not receive requests, only active nodes participate in the routes. Using ABF, nodes do not maintain data about the network topology. This strategy avoids the energy consumption with the transmission of periodic control messages used for routing table updates. Experiments showed that the spatial query processing mechanisms using ABF can reduce energy consumption by up to 75%.

In the Dissemination and Aggregation stages we proposed three algorithms. Thus, the proposed spatial query processing mechanism has three variations: Classic, DRF and Itinerary. All of them employ the proposed ABF routing protocol in their Forwarding and Return stages. Classic disseminates queries by Restricted Flooding. DRF disseminates queries like Classic, however DRF uses a different algorithm in the Aggregation stage. Itinerary creates in the Dissemination stage an itinerary having a zigzag pattern, in which the query sweeps the entire RoI. We implemented these variations in the NS 2.34 simulator and performed three rounds of experiments. In the first round,

we analyzed the best parameter values for the proposed algorithms. In the second round, we varied the duty cycle and compared these algorithms against SWIP-2. In the third round, we simulated transient and permanent failures and also compared the proposed mechanism against SWIP-2.

The first round of experiments showed that Classic suffered from packet collisions during the Dissemination and Aggregation stages, hence it did not present a good performance. In the second round, varying the duty cycle, DRF and Itinerary consumed only 25% and 15% of the energy consumed by SWIP-2, respectively. We verified that SWIP-2 completes less than 10% of the queries when the duty cycle was smaller than 50%. DRF presented up to 85% of coverage area, even in scenarios in which nodes stayed only 20% of the time active. Itinerary also presented more than 85% of coverage area in WSN having nodes staying more than 30% of the time active. Itinerary presented the worst end-to-end delay, however it presented the best energy consumption.

In the third round of the experiments we varied the failure probability. We simulated transient and permanent failures. In both types of failure, SWIP-2 finished less than 8% of the queries when the failure probability was higher than 30%. Moreover, it consumed much more energy than the proposed mechanisms due to periodic beacon transmissions, which are necessary in order to remove failed neighbors from the routing tables. DRF presented up to 75% less energy consumption than SWIP-2. However, DRF presented less than 20% of coverage area when the failure probability was less than 30%. This poor performance was similar to SWIP-2 due to nodes failing during the Dissemination stage. Itinerary completed up to 80% of the queries when the failure probability was up to 50% (SWIP-2 processed no queries in these scenarios). Furthermore, this algorithm presented the smallest energy consumption (up to 90% less than SWIP-2) since it used less nodes than the others mechanisms to disseminate queries in the RoI and it did not need to update neighbor tables. However, it presented the highest end-to-end delay (up to 50% more than SWIP-2).

The experiments showed that the proposed algorithms improved the performance of the spatial query processing mechanisms. The energy savings were obtained in consequence of the ABF strategy to forward packets (avoiding periodic beacon transmissions), the reduction on the number of packet transmissions performed by the proposed Pre-Processing algorithm and the itinerary created inside the RoI. The proposed spatial query processing mechanism increased the amount of successful queries when compared to SWIP-2 in WSN having failure-prone nodes or nodes in sleep mode. The experiments showed a tradeoff between end-to-end delay and energy consumption. Hence, we consider Itinerary as the best variation of our mechanism, since it presented the small-

est energy consumption and the best coverage area in the majority of the scenarios. However, Itinerary presents the highest end-to-end delay, confirming the previously mentioned tradeoff.

7.2 Directions for Future Research

In future works, we intend to consider several requirements that are found in real WSN, but are not taken into account in the literature. One of them is the variation of the link quality, which depends on various factors, such as node distance, external sources of interference, energy supply and transmissions occurring next to the sender [Hänninen et al., 2011]. The link quality can create unidirectional links, which must be considered by routing protocols. We also intend to consider imprecision in the nodes' position because all the location methods for WSN found in the literature present non negligible imprecision [Wang et al., 2010]. Furthermore, we intend to consider mobile sensor nodes. This characteristic complicates the query processing since nodes depend on their location to retrieve queries. The literature presents several mobility patterns, hence we must evaluate the performance of the proposed mechanism with different types of mobility. Hence, we intend to implement the proposed mechanism on real sensor nodes and compare the simulation results against the real implementation.

Future researches can also consider other ways to represent regions of interest. Using the RoI representation considered in this thesis, nodes know the direction they have to send a query because a RoI is a polygon composed by a set of geographic coordinates. However, there are other ways to represent RoIs, such as fractals and curves [Goussevskaia et al., 2005]. An advantage of using these kinds of representation is that a query transmissions use less packets than when the RoI is represented by polygons, since polygons can be represented by a large number of points. The challenge of using these representations is the algorithm complexity, since sensor nodes have small computational resources and memory size. Hence, using fractals and equations, the Pre-Processing algorithms must create lightweight representations of RoIs, otherwise the query processing time may increase too much.

Another characteristic that must be considered is the location of sensor nodes in three dimensions (3D), such as considered in [Luo et al., 2011; Ortiz et al., 2007]. To the best of the authors' knowledge, all spatial query processing mechanisms consider coordinates in two dimensions (2D). 3D coordinates influence all spatial query processing, mainly in the definition of the RoI. These regions can be defined as a polyhedron, which can employ several 3D points in its representation. We believe that the

processing of this type of query consumes much more energy, since the polyhedron representation would occupy several packets in the network. Furthermore, the algorithms for all stages must be adapted for 3D processing.

7.3 List of Thesis-related Publications

We present below a list of the published papers and the articles submitted during the PhD. The works presented here are direct results of the thesis.

Papers

Silva, R.I., Almeida, V.D.D, Poersh, A.M., Nogueira, J.M.S (2009). Spatial Query Processing in Wireless Sensor Network for Disaster Management. In Proceedings of the 2th IFIP/IEEE International Wireless Days (WD 2009), pages 194-198, Paris, France, December, 2009. (Full Paper)

Silva, R.I., Almeida, V.D.D, Poersh, A.M., Nogueira, J.M.S (2010). Wireless Sensor Network for Disaster Management. In Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010), pages 870-873, Osaka, Japan, April, 2010. (Short Paper)

Silva, R.I., Macedo, D.F., Nogueira, J.M.S. (2011). Contornos Irregulares no Processamento de Requisições Espaciais para Redes de Sensores Sem Fio. In Proceedings of the 29th Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2011), Campo Grande - MS, Brazil, may-june, 2011. (Full Paper)

Silva, R.I., Macedo, D.F., Nogueira, J.M.S. (2012). Fault Tolerance in Spatial Query Processing for Wireless Sensor Networks. In Proceedings of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS 2012), Maui, Hawaii, USA, April, 2012. (Full Paper)

Articles under evaluation

Silva, R.I., Macedo, D.F., Nogueira, J.M.S. (2010). Spatial Query Processing in Wireless Sensor Networks - A Survey. Information Fusion - Elsevier

(submitted).

Silva, R.I., Macedo, D.F., Nogueira, J.M.S. (2011). Duty Cycle Aware Spatial Query Processing in Wireless Sensor Networks. *Computer Communications* - Elsevier (submitted).

Appendix A

Abbreviations

DII	Destination In the Itinerary
DIKNN	Density-aware Itinerary KNN
GPSR	Greedy Perimeter Stateless Routing
GRT	GeoRouting Tree
ID	Itinerary Distance
IKNN	Informative K-Nearest Neighbor
IP	Ideal Position
IWQE	Itinerary-baseb Window Query Execution
KBT	KNN Bounded Tree
KNN	K-Nearest Neighbors
LAP	Last Aggregation Packet
LMP	LeftMost Point
LNI	Last Node on the Itinerary
MBA	Minimum Bounding Area
MBR	Minimum Bounding Rectangle
PCIKNN	Parallel Concentric Itinerary KNN
QR	Query Result
r	Radio Range
R	Reference Point
RoI	Region of Interest
S	Sender
SPIX	Spatial IndeX
SWIP	Spatial Window Processing
SWOP	Spatial Window Query Over Phenomena
WSI	Window Spanning Infrastructure

WSN Wireless Sensor Networks

Bibliography

- Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38:393–422.
- Buettner, M., Yee, G. V., Anderson, E., and Han, R. (2006). X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, SenSys '06, pages 307–320, New York, NY, USA. ACM.
- Cardei, M. and Wu, J. (2006). Energy-efficient coverage problems in wireless ad-hoc sensor networks. *Elsevier Computer Communications*, 29(4):413–420.
- Casanova, M. A., Câmara, G., Davis, C. A., Vinhas, L., and de Queiroz, G. R. (2005). *Banco de Dados Geográfico*. MundoGeo.
- Chen, X., Kim, Y.-A., Wang, B., Wei, W., Shi, Z. J., and Song, Y. (2010). Fault-tolerant monitor placement for out-of-band wireless sensor network monitoring. *Elsevier Ad Hoc Networks*, 10:62–74.
- Cobuild, C. (2003). *Collins COBUILD English Dictionary*. HarperCollins, Glasgow, 5nd edition.
- Coman, A., Sander, J., and Nascimento, M. A. (2007). Adaptive processing of historical spatial range queries in peer-to-peer sensor networks. *Distributed and Parallel Databases*, 22(2-3):133–163.
- Crossbow Technology, Inc. (2010a). Iris mote datasheet. <http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=135:iris>.
- Crossbow Technology, Inc. (2010b). Micaz Mote Datasheet. <http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148:micaz>.

- Demers, A., Gehrke, J., Rajaraman, R., Trigoni, N., and Yao, Y. (2003). The cougar project: a work-in-progress report. *ACM Special Interest Group on Management of Data (SIGMOD)*, 32(4):53--59.
- Demirbas, M. and Lu, X. (2007). Distributed quad-tree for spatial querying in wireless sensor networks. In *IEEE International Conference on Communications (ICC)*, pages 3325--3332.
- Demirkol, I., Ersoy, C., and Alagoz, F. (2006). Mac protocols for wireless sensor networks: a survey. *IEEE Communications Magazine*, 44:7.
- Deshpande, A., Ives, Z., and Raman, V. (2007). Adaptive query processing. *Foundations and Trends in Databases*, 1(1):1--140.
- Douglas, D. and Peucker, T. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10:112--122.
- Du, S., Kumar, A., David, S., and Johnson, B. (2007). Rmac: a routing-enhanced duty cycle mac protocol for wireless sensor network. In *IEEE International Conference on Computer Communications*, pages 1478--1486.
- El-Hoiydi, A. and Decotignie, J.-D. (2004). WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks. In *First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, pages 18--31.
- Felice, P. D., Ianni, M., and Pomante, L. (2008). A spatial extension of TinyDB for wireless sensor networks. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 1076--1082.
- Fu, T.-Y., Peng, W.-C., and Lee, W.-C. (2010). Parallelizing itinerary-based knn query processing in wireless sensor networks. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22:711--729.
- Gabriel, K. R. and Sokal, R. R. (1969). A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259--278.
- Gao, J., Guibas, L., Milosavljevic, N., and Hershberger, J. (2007). Sparse data aggregation in sensor networks. In *6th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 430--439.

- Goldin, D., Song, M., Kutlu, A., Gao, H., and Dave, H. (2003). Georouting and delta-gathering: Efficient data propagation techniques for geosensor networks. In *First Workshop on Geo Sensor Networks*, pages 9--11.
- Goussevskaia, O., Machado, M. D., Mini, R. A., Loureiro, A. A., Mateus, G. R., and Nogueira, J. M. (2005). Data dissemination based on the energy map. *Comm. Mag.*, 43(7):134--143.
- Hänninen, M., Suhonen, J., Hämäläinen, T. D., and Hännikäinen, M. (2011). Link quality-based channel selection for resource constrained wsns. In *6th International Conference Advances in Grid and Pervasive Computing*, pages 254--263.
- IEEE (2006). Wireless MAC and PHY specifications for low rate WPAN, IEEE std 802.15.4-2006 (revision of IEEE std 802.15.4-2006). *IEEE*, 1:1--305.
- Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., and Silva, F. (2003). Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2--16.
- Jenks, G. F. (1981). Computers and human frailties. In *Computers and Human Frailties*, pages 1--10. Association of American Geographers.
- Jin, G. and Nittel, S. (2008). Toward spatial window queries over continuous phenomena in sensor networks. *IEEE Transactions of Parallel and Distributed Systems*, 19(4):559--571.
- Jin, Z., Jian-Ping, Y., Si-Wang, Z., Ya-Ping, L., and Guang, L. (2009). A survey on position-based routing algorithms in wireless sensor networks. *Algorithms*, 2(1):158--182.
- Jurdak, R., Ruzzelli, A. G., and O'Hare, G. M. P. (2010). Radio sleep mode optimization in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9:955--968.
- Kalogeraki, V. and Soheili, A. (2008). Data collection, reliable real-time. In *Encyclopedia of GIS*, pages 204--209. Springer.
- Karp, B. and Kung, H. T. (2000). GPSR: greedy perimeter stateless routing for wireless networks. In *6th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 243--254, New York, NY, USA.

- Li, Y., Eo, S.-H., Lee, D.-W., Oh, Y.-H., and Bae, H.-Y. (2008). An energy efficient adaptive back forwarding method for spatial range query processing in sensor networks. In *International Conference on Advanced Language Processing and Web Information Technology (ALPIT)*, pages 373--379.
- Lu, C., Xing, G., Chipara, O., Fok, C.-L., and Bhattacharya, S. (2005). A spatiotemporal query service for mobile users in sensor networks. In *25th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 381--390.
- Luo, J., Li, F., and He, Y. (2011). 3DQS: Distributed Data Access in 3D Wireless Sensor Networks. In *Proc. of the IEEE International Conference on Communications (ICC 2011)*, pages 1--5.
- Macedo, D. F., Correia, L. H. A., dos Santos, A. L., Loureiro, A. A., Nogueira, J. M., and Pujolle, G. (2005). Evaluating fault tolerance aspects in routing protocols for wireless sensor networks. In *Fourth Annual Mediterranean Ad Hoc Networking Workshop*.
- Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2005). TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122--173.
- Manalopoulos, Y. and Papadopoulos, A. N. (2004). *Spatial Databases: Technologies, Techniques and Trends*. Idea Group Publishing.
- Mini, R. A. F., Val Machado, M. d., Loureiro, A. A. F., and Nath, B. (2005). Prediction-based energy map for wireless sensor networks. *Ad Hoc Netw.*, 3(2):235--253.
- Moteiv Corporation (2006). Tmote Sky Datasheet. <http://sentilla.com/files/pdf/eol/tmote-sky-datasheet.pdf>.
- Nakamura, E. F., Loureiro, A. A. F., and Frery, A. C. (2007). Information fusion for wireless sensor networks: Methods, models, and classifications. *ACM Comput. Surv.*, 39(3):1--55.
- Niculescu, D. and Nath, B. (2003). Trajectory based forwarding and its applications. In *Proceedings of the 9th annual international conference on Mobile computing and networking, MobiCom '03*, pages 260--272, New York, NY, USA. ACM.
- ns (2010). The Network Simulator NS-2. www.isi.edu/nsnam/ns.

- Ortiz, C. D., Puig, J. M., Palau, C. E., and Esteve, M. (2007). 3d wireless sensor network modeling and simulation. *Sensor Technologies and Applications, International Conference on*, 0:307–312.
- Paradis, L. and Han, Q. (2007). A Survey of Fault Management in Wireless Sensor Networks. *J. Netw. Syst. Manage.*, 15:171--190.
- Park, K., Lee, B., and Elmasri, R. (2007). Energy efficient spatial query processing in wireless sensor networks. In *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, pages 719--724.
- Polastre, J., Hill, J., and Culler, D. (2004). Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, pages 95--107, New York, NY, USA. ACM.
- Ramsey, P. (2001). Postgis manual. Reflection Research Corporation.
- Rao, R. R., Eisenberg, J., and Schmitt, T. (2007). *Committee on using information technology to enhance disaster management Improving disaster management: The role of IT in mitigation, preparedness, response, and recovery*. National Academies Press.
- Rappaport, T. S. (2002). *Wireless Communications: Principles and Practice (2nd Edition)*. Prentice Hall, 2 edition.
- Robinson, A. H., Sale, R. D., and Morrison, J. L. (1978). *Elements of Cartography*. John Wiley & Sons.
- Scott, T., Wu, K., and Hoffman, D. (2006a). Radio propagation patterns in wireless sensor networks: new experimental results. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, IWCMC '06, pages 857--862, New York, NY, USA. ACM.
- Scott, T., Wu, K., and Hoffman, D. (2006b). Radio propagation patterns in wireless sensor networks: new experimental results. In *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, IWCMC '06, pages 857--862, New York, NY, USA. ACM.
- Sharifzadeh, M. and Shahabi, C. (2004). Supporting spatial aggregation in sensor network databases. In *12th Annual ACM International Workshop on Geographic Information Systems (GIS)*, pages 166--175.

- Silva, R., Leite, J. C. B., and Fernandez, M. P. (2006). Extending the lifetime of ad hoc wireless networks. In *International Conference on Intelligent Computing (ICIC)*, pages 1324--1331.
- Soheili, A., Kalogeraki, V., Gunopulos, D., et al. (2005). Spatial queries in sensor networks. In *3th Annual ACM International Workshop on Geographic Information Systems (GIS)*, pages 61--70.
- Solis, I. and Obraczka, K. (2006). In-network aggregation trade-offs for data collection in wireless sensor networks. *International Journal of Sensor Networks*, 1:200--212.
- Sun, Y., Du, S., Gurewitz, O., and Johnson, D. B. (2008a). DW-MAC: a low latency, energy efficient demand-wakeup MAC protocol for wireless sensor networks. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing, MobiHoc '08*, pages 53--62, New York, NY, USA. ACM.
- Sun, Y., Gurewitz, O., and Johnson, D. B. (2008b). RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks. In *Proceedings of the 6th ACM conference on Embedded network sensor systems, SenSys '08*, pages 1--14, New York, NY, USA. ACM.
- Swain, A. R., Hansdah, R. C., and Chouhan, V. K. (2010). An energy aware routing protocol with sleep scheduling for wireless sensor networks. In *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications, AINA '10*, pages 933--940, Washington, DC, USA. IEEE Computer Society.
- Tobler, R. W. (1964). An experiment in the computer generalization of maps. Technical report, Department of Geography - University of Michigan.
- van Dam, T. and Langendoen, K. (2003). An adaptive energy-efficient MAC protocol for wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171--180, New York, NY, USA. ACM.
- Wang, J., Ghosh, R. K., and Das, S. K. (2010). A survey on sensor localization. *Journal of Control Theory and Applications*, 8(1):2--11.
- Wu, S.-H., Chuang, K.-T., Chen, C.-M., and Chen, M.-S. (2007). DIKNN: An itinerary-based KNN query processing algorithm for mobile sensor networks. In *International Conference on Data Engineering (ICDE)*, pages 456--465.

- Xu, Y., Chien Lee, W., Xu, J., and Mitchell, G. (2006). Processing window queries in wireless sensor networks. In *22th IEEE International Conference on Data Engineering (ICDE)*.
- Xu, Y., Fu, T.-Y., Lee, W.-C., and Winter, J. (2007). Processing K nearest neighbor queries in location-aware sensor networks. *Signal Processing*, 87(12):2861--2881.
- Ye, W., Heidemann, J., and Estrin, D. (2004). Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transaction Network*, 12:493--506.
- Yen, L.-H. and Cheng, Y.-M. (2009). Range-based sleep scheduling (rbss) for wireless sensor networks. *Wireless Personal Communications*, 48:411--423.
- Zhao, F. and Guibas, L. J. (2004). *Wireless sensor networks : an information processing approach*. The Morgan Kaufmann series in networking. Morgan Kaufmann.
- Zhou, X., Xue, G., Qian, C., and Li, M. (2008). Efficient data suppression for wireless sensor networks. In *14th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 599--606.

