Contornos Irregulares no Processamento de Requisições Espaciais para Redes de Sensores Sem Fio*

Rone Ilídio da Silva[1,2], Daniel Fernandes Macedo[2], José Marcos Silva Nogueira[2]

[1]Universidade Federal de São João del-Rei (UFSJ)
Campus Alto Paraopeba – Ouro Branco, MG – Brasil

[2]Universidade Federal de Minas Gerais (UFMG)
Departamento de Ciência da Computação – Belo Horizonte, MG – Brasil

rone@ufsj.edu.br, {damacedo,jmarcos}@dcc.ufmg.br

Abstract. *The recent evolution in sensor node location technology has spurred the development of a special type of in-network processing for Wireless Sensor Network (WSN), called spatial query processing. These queries require data from nodes within a region (called region of interest) defined by the users. The state of the art of spatial query processing consider only regions of interest having rectangular or circular shapes. This work proposes an energy efficient itinerary based in-network spatial query processing mechanism and a strategy to reduce the energy consumption of overhearing during query processing. The proposals were developed to process queries within irregular regions of interest, which can represent real objects on maps. The simulations show that the proposed mechanism presents energy consumption substantially smaller than one of the mechanisms found in the literature (about 80%) and also show that the strategy to avoid overhearing saves substantial energy during the query processing.*

Resumo. *A evolução da tecnologia na localização dos nós em Redes de Sensores Sem Fio (RSSF) levou à definição de mecanismos que processam requisições espaciais nessas redes. Tal tipo de requisição busca dados em regiões definidas pelo usuário (denominadas regiões de interesse -* ***RDI****). No estado da arte, essas regiões possuem formato retangular ou circular. Este trabalho descreve duas propostas que reduzem o consumo de energia no processamento de requisições espaciais cujas regiões de interesse possuem formatos irregulares, como polígonos, que podem representar objetos reais plotados sobre mapas. A primeira proposta é uma estratégia que reduz o consumo de energia com overhearing durante o processamento de requisições. A segunda é um mecanismo para processamento de requisições espaciais em RSSF baseado na criação de itinerários. Simulações mostraram que a estratégia para evitar overhearing apresenta uma economia substancial de energia durante o processamento de requisições espaciais e o mecanismo proposto apresenta consumo de energia 80% menor que um mecanismo encontrado na literatura.*

*O presente trabalho foi realizado com o apoio da CAPES, CNPQ e FAPEMIG, entidades do Governo Brasileiro e do Governo do Estado de Minas Gerais voltadas ao desenvolvimento científico e tecnológico.

1. Introdução

Pesquisas recentes mostram que **Redes de Sensores Sem Fio (RSSF)** deixaram de ser apenas ferramentas que realizam coletas periódicas de dados. Em várias situações, tais redes são utilizadas para monitorar regiões 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 [1]. Podem ser citados como exemplos o TinyDB [2] e o Cougar [3], mecanismos de consultas para RSSF que permitem a criação de requisições em uma linguagem semelhante ao SQL. Entretanto, esses mecanismos 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 onde se deseja realizar a coleta, uma vez que as requisições são enviadas para todos os nós da rede [4]. Uma forma para tratar esse problema é a criação de mecanismos para RSSF que suportem consultas com restrições espaciais, os quais encaminham as consultas para dentro da região definida pelo usuário, limitando a coleta de dados somente aos nós contidos nela.

As requisições que utilizam atributos espaciais são denominadas **requisições espaciais** [5] e são tipicamente empregadas em bancos de dados geográficos, como o PostGIS e o Oracle Spatial [6]. 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 *sobrepõe* outro. RSSF podem ser modeladas como bancos de dados geográficos com o intuito de responder a requisições, tais como: "Colete a temperatura média da região da Lagoa da Pampulha".

Na literatura, podem ser encontrados trabalhos que definem mecanismos para processamento de requisições espaciais em RSSF. Entretanto, esses trabalhos consideram **regiões de interesse (RDI)** com formato retangular ou circular. Até o presente momento, somente nosso trabalho anterior [7] 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 que são exibidos por Sistemas de Informações Geográficas [6]. A Figura 1 ilustra duas fotos de satélite nas quais foram destacados os contornos de parte de um lago e de uma mina.

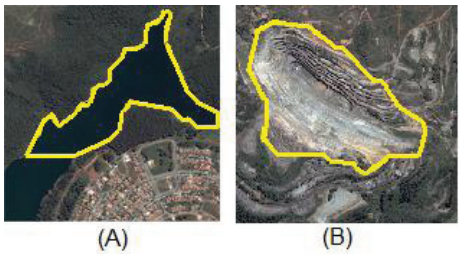

Figure 1. (A) Contorno de parte de um lago. (B) Contorno de uma mina.

Este trabalho propõe um mecanismo para processamento de requisições espaciais em RSSF que considera requisições com regiões de interesse em forma de polígono. Tal mecanismo utiliza um algoritmo que cria um itinerário por onde a requisição é disseminada e por onde os dados sensoriados são agregados. O trabalho também propõe a redução do consumo de energia com *overhearing* durante o processamento de requisições.

A maior contribuição da proposta é a redução do consumo de energia em requisições com RDI irregular. Nenhum dos trabalhos relacionados considera esse tipo de região.

2. Processamento de Requisições Espaciais em RSSF

2.1. Objetivo

Considerando o amadurecimento das técnicas que fornecem a localização dos nós em RSSF [8], o principal objetivo deste trabalho é reduzir o consumo de energia no processamento de requisições espaciais nesse tipo de rede. São consideradas requisições como: "Qual a média de temperatura a ser coletada pelos nós sensores contidos dentro da região definida pelo usuário?" Esse tipo de requisição pode ser expressa na linguagem **SQL-Like**, definida pelo mecanismo TinyDB [2], da seguinte forma:

```
"SELECT avg(Temp) FROM sensors WHERE Inside(RDI) ONCE"
```

Como parâmetro da função de agregação $avg()$ informa-se qual dado será agregado. A função $Inside$ verifica se o nó sensor está dentro da região definida pela variável RDI. Tal variável contém as informações de um polígono, o qual é formado por uma sequência de m pontos. Como m pode ser grande, a requisição pode ocupar vários pacotes da RSSF. Considera-se que qualquer nó pode iniciar o processamento de uma requisição.

2.2. Divisão do Processamento em Estágios

Como proposto em nosso trabalho anterior [7], o processamento de requisições espaciais pode ser dividido em seis estágios. Tais estágios são ilustrados na Figura 2. O **Pré-Processamento** é realizado no computador do usuário, antes da requisição ser enviada para a RSSF. Nesse computador, a requisição é preparada para ser processada pela rede. O primeiro nó sensor que recebe uma requisição é denominado **Originador**. Durante o Estágio de **Encaminhamento**, a requisição é propagada do Originador até um nó dentro da RDI, denominado **Coordenador**. A política de escolha do nó Coordenador varia entre os mecanismos encontrados. No Estágio da **Disseminação**, a requisição é transmitida do Coordenador para todos os nós contidos dentro da RDI. Tais nós coletam informações do ambiente no Estágio de **Sensoriamento** e mandam tais informações para o Coordenador. O Estágio de **Agregação** ocorre durante o percurso da informação a partir de sua origem até o Coordenador. 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 Coordenador calcula o resultado da requisição e o encaminha para o Originador.

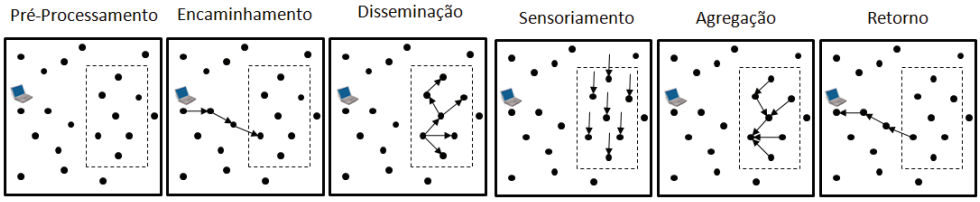

Figure 2. Os seis estágios para processamento de requisições espaciais em RSSF.

3. Trabalhos relacionados

Foram encontrados na literatura dois principais tipo de requisição: KNN e em Janela. Nas **Requisições KNN** o usuário define um ponto na região de interesse e requisita a leitura

dos K sensores mais próximos desse ponto. Os trabalhos [9, 10] descrevem mecanismos que processam esse tipo de requisição. Nas **Requisições em Janela**, que são o foco deste trabalho, o usuário define uma região onde ele deseja realizar a coleta de informações. As requisições devem então ser encaminhadas para a RDI, para que os nós contidos nessa região realizem a coleta de dados.

Em [11] são descritos mecanismos para processamento de requisições em janela que se diferenciam pelo algoritmo utilizado no Estágio de Disseminação. Foram testados três algoritmos: Inundação, Inundação Restrita e Windepth. Na Inundação, todos os nós da rede transmitem a requisição uma vez. Consequentemente, os nós a recebem a partir de cada um de seus vizinhos. Por esse motivo, seu consumo de energia tende ser alto. A Inundação Restrita é uma Inundação na qual somente os nós dentro da RDI transmitem o pacote. Com Windepth, quando um nó X recebe uma requisição, ele a envia para um vizinho dentro da RDI que ainda não a tenha recebido. Quando esse vizinho retorna com o resultado parcial da requisição, X repete a operação com outro vizinho. Esse processo é repetido até que todos os vizinhos de X tenham respondido a requisição. Após isso, ele lê as informações de seus sensores, agrega às informações recebidas e envia os resultados para o nó que lhe enviou a requisição. Os mecanismos analisados nesse trabalho utilizam no Estágio de Encaminhamento o protocolo de roteamento geográfico Guloso, o qual não garante que o encaminhamento terá sucesso [12]. A economia de energia limita-se à redução do número de mensagens transmitidas, o *overhearing* não é considerado e as RDI são retangulares.

Os mecanismos descritos em [13–16] criam índices espaciais distribuídos para processar requisições em janela. Tais índices baseiam-se na criação da MBR (*Minimum Bounding Rectangle*). A **MBR** de um nó é o menor retângulo que envolve esse nó e todos os seus dependentes na árvore de roteamento cuja raiz é a estação base [14]. A Figura 3 ilustra a MBR e o processamento de uma requisição. Quando um nó recebe uma requisição, ele verifica se a RDI sobrepõe sua MBR. Se houver uma sobreposição, tal nó retransmite a requisição para seus descendentes. Caso não ocorra sobreposição, a requisição é ignorada. Em comparação a uma Inundação, essa estratégia reduz o número de nós que transmitem a requisição. Porém, uma limitação desse tipo de mecanismo é que somente a estação base pode iniciar o processamento de requisições. Além disso, o consumo com *overhearing* não é evitado e as RDI também são retangulares.

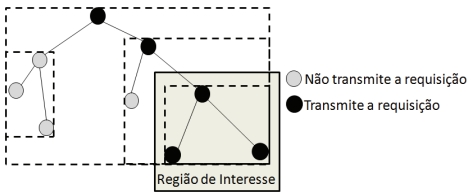

Figure 3. Exemplo de MBR criada sobre a árvore de roteamento e os nós que participam do processamento de uma requisição.

Em [17] os autores definem um mecanismo que dissemina a requisição por itinerário e mostram que o consumo energético desse tipo de mecanismo é menor que o consumo de mecanismos baseados em Inundação Restrita. Entretanto, as RDIs consideradas são retangulares, cujas faces são paralelas às faces da região monitorada. Esse formato reduz o tamanho da requisição e facilita a criação de itinerários. Somente nosso trabalho anterior

[7] considera requisições em janela com RDIs em forma de polígono. Nele foi proposto um algoritmo que reduz o número de pontos na representação da RDI para que a RSSF utilize menos pacotes para a transmissão da requisição. Entretanto, ele utiliza como base o mecanismo proposto em [11] que utiliza um algoritmo geográfico de roteamento que utiliza uma heurística gulosa no Encaminhamento e Inundação Restrita na Disseminação. Com isso, o Encaminhamento não é garantido e o consumo de energia é elevado.

4. Mecanismo Básico

Esta seção descreve o mecanismo para processamento de requisições espaciais desenvolvido em nosso trabalho anterior [7]. Tal mecanismo é utilizado para a descrição das contribuições propostas neste artigo e como base de comparação nas simulações. A seguir tem-se as descrições dos algoritmos utilizados em cada um de seus estágios.

Pre-Processamento: reduz o número de pontos da RDI com a utilização do algoritmo de Douglas-Peucker [7].

Encaminhamento: utiliza o protocolo de roteamento geográfico Guloso para encaminhar a requisição do Originador até um nó dentro da RDI. Tal protocolo sempre transmite a requisição para o vizinho que estiver mais próximo da RDI, até que a requisição alcance um nó dentro de tal região.

Disseminação: utiliza Inundação Restrita, ou seja, quando um nó recebe uma requisição, ele verifica se está dentro da RDI. Se estiver, ele retransmite a requisição e salva como pai o nó que a enviou. Com isso forma-se uma árvore de roteamento dentro da RDI. Se o nó estiver fora da RDI, a requisição é ignorada. Após transmitir a requisição, um nó inicia um contador de tempo T_{agr}. Ao seu final será iniciado o Estágio de Agregação. O Coordenador possui o maior valor para T_{agr} (denominado T_{max}), pois a cada nível da árvore tal valor decai um fator FD (fator de decaimento), ou seja, quanto maior o nível da árvore menor o valor de T_{agr}.

Sensoriamento: os nós coletam dos dados requisitados.

Agregação: quando o contador de tempo T_{agr} expira, o nó aplica o operador de agregação sobre os dados recebidos de seus descendentes e os dados coletados por seus sensores. Os resultados dessa operação são transmitidos para seu pai. Esse estágio sempre inicia em nós-folha, pois como estão nos maiores níveis da árvore de roteamento, esses nós possuem os menores valores do contador de tempo T_{agr}.

Retorno: após realizar a agregação, o Coordenador envia para o Originador o resultado da requisição. Ele utiliza a rota criada no Estágio de Encaminhamento.

5. IBIS- Algoritmo para Processamento de Requisições Espaciais Baseado em Itinerário

IBIS (Itinerary Based dissemination for Irregular Shapes) é o mecanismo proposto neste trabalho para processamento de requisições em janela em RSSF. Ele foi criado para que o processamento de tais requisições seja energeticamente eficiente, reduzindo o mínimo possível a qualidade dos resultados obtidos pela rede. Tal mecanismo considera RDI com contornos irregulares, o que dificulta a criação do itinerário.

Em seu Estágio de Disseminação, IBIS cria um itinerário em forma de ziguezague sobre o Fecho Convexo da região de interesse. Essa estratégia é exemplificada pela Figura

4(A). Ela exibe o contorno de parte de um lago, no qual sua porção inferior direita possui formas estreitas que dificultam a criação de itinerários. Mesmo em situações como esta, IBIS é capaz de criar o itinerário e disseminar a requisição para todos os nós da RDI. A Figura 4 (B) apresenta o Fecho Convexo gerado a partir desse lago e a Figura 4 (C) apresenta um itinerário criado sobre a região. Os trabalhos relacionados que definem itinerários consideram regiões de interesse retangulares ou circulares, nas quais torna-se mais simples a definição do itinerário.

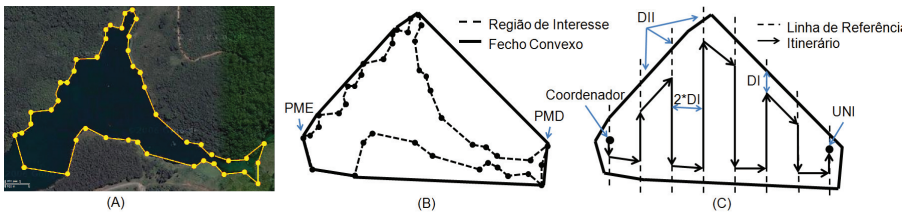

Figure 4. (A) Contorno de parte de um lago. (B) Fecho Convexo sobre a região. (C) Itinerário criado sobre o Fecho Convexo.

Para facilitar o entendimento dos estágios que compõem IBIS, faz-se necessário definir alguns elementos que compõem o itinerário, Figura 4(B) e (C). São eles:

- **PME**: Ponto Mais à Esquerda da região de interesse.
- **PMD**: Ponto Mais à Direita da região de interesse.
- **UNI**: Último Nó no Itinerário.
- **Linhas de Referência**: linhas verticais que auxiliam a definição do itinerário.
- **DI**(Distância do Itinerário): valor de referência que define a distância entre duas linhas de referência (2 x DI) e a distância entre o itinerário e a borda do fecho convexo da região de interesse.
- **DII**(Destino Imediato no Itinerário): são pontos onde a linha de referência atual cruza com o Fecho Convexo.

5.1. Estágios de IBIS

O funcionamento de IBIS é explicado a partir dos estágios definidos anteriormente para processamento de requisições espaciais.

Pre-Processamento: reduz o número de pontos da RDI com a utilização do algoritmo definido em nosso trabalho anterior [7].

Encaminhamento: utiliza GPSR [18], pois diminui falhas no encaminhamento. Define-se como Coordenador o primeiro nó dentro da RDI que receber a requisição e que estiver a uma distância de PME menor que o valor de DI.

Disseminação: é criado um itinerário dentro do fecho convexo da RDI. O primeiro nó é o Coordenador e último o UNI. Este último é um nó dentro da RDI que possui sua distância até PMD menor que o valor definido para DI. Cada nó sobre o itinerário define quem é o próximo nó e envia a requisição para ele. Durante esse processo, os nós que forem vizinhos daqueles que fazem parte do itinerário também receberão a requisição. Além de encaminhar a requisição, o Coordenador inicia a contagem de um cronômetro (15 segundos) que será utilizado no Estágio de Agregação.

Sensoriamento: todos os nós dentro da RDI, que receberam a requisição, coletam as informações de seus sensores.

Agregação: todos os nós dentro da RDI, mas que não participaram do itinerário, enviam suas leituras para os nós sobre o itinerário que lhes enviaram a requisição. Quando o Coordenador termina o cronômetro iniciado na Disseminação, ele envia um pacote para o nó que ele definiu como próximo no itinerário. Esse pacote contém o resultado do operador de agregação aplicado sobre os dados coletados por ele e por seus vizinhos. Ao receber um pacote desse tipo, um nó sobre o itinerário realiza a mesma operação. Esse estágio finaliza quando o UNI aplica o operador de agregação sobre os dados recebidos do nó anterior a ele no itinerário.

Retorno: utiliza o GPSR para enviar a resposta do nó UNI até o nó Originador.

IBIS utiliza o protocolo de roteamento geográfico GPSR para os estágios de Encaminhamento e Retorno, enquanto o Mecanismo Básico utiliza o protocolo Guloso. Como mencionado anteriormente, o protocolo Guloso não garante a entrega da requisição. Durante as simulações realizadas para este trabalho, verificou-se que ao se utilizar o protocolo Guloso 8,3% das requisições não obtiveram sucesso no Encaminhamento. Com a utilização do GPSR, todas as requisições foram encaminhadas corretamente.

5.2. Algoritmo para Criação do Itinerário

Definidos os estágios de IBIS, torna-se necessária a descrição do algoritmo que cria o itinerário durante o Estágio de Disseminação. Esse algoritmo distribuído define um caminho para a requisição e envolve os seguintes nós (Figura 5(A)) durante a disseminação:

- Nós sobre o itinerário e dentro da região de interesse (**SID**): retransmitem a requisição no itinerário e as leituras de seus sensores.
- Nós sobre o itinerário e fora da região de interesse (**SIF**): somente retransmitem a requisição no itinerário, não fazem coleta de informação de seus sensores.
- Nós fora do itinerário e dentro da região de interesse (**FID**): enviam as leituras de seus sensores para o SID que lhe enviou a requisição.
- Nós fora do itinerário e fora de região de interesse (**FIF**): ignoram a requisição.
- Nós que não receberam a requisição (**NRR**): são os nós que estão fora da RDI ou nós que, mesmo dentro dessa região, não receberam a requisição devido às imperfeições do itinerário criadas pelo posicionamento aleatório dos nós sensores.

Os dados disseminados com as requisições são: todos os pontos da RDI, a localização do Originador, o DII e a Direção. Apesar de utilizar o Fecho Convexo para criar o itinerário, todos os pontos da RDI são necessários para que os nós saibam se estão dentro ou fora desta região e com isso definir se devem ou não coletar dados. A localização do Originador é importante para que o UNI saiba para onde ele deve transmitir o resultado da requisição. O DII é utilizado para que um nó sobre o itinerário defina qual será o próximo a receber a requisição. Por último, a **Direção** é necessária para que o itinerário tenha o formato de ziguezague (ilustrado pela Figura 4) e consiga cobrir toda a região delimitada pelo Fecho Convexo. Tal variável define se a requisição está indo para **CIMA**, para **BAIXO** ou para a **DIREITA**. Quando um nó recebe uma requisição, ele verifica se está a uma distância de DII menor que o parâmetro DI. Se estiver, ele muda a Direção, escolhe um novo DII e envia a requisição para seu vizinho mais próximo deste ponto. Caso contrário, ele simplesmente envia a requisição para o vizinho mais próximo do DII contido na requisição.

O Algoritmo 1 define como o itinerário é criado. Ele descreve a função executada quando um nó X recebe uma requisição para ser disseminada. Na linha 2, X verifica se

a mensagem é destinada a ele. Caso não seja, X sabe que o itinerário não passará por ele, ou seja, ele é um nó FID ou FIF. Se ele estiver dentro da região de interesse, ele é um nó FID e por isso envia os dados coletados por seus sensores para o nó que enviou a requisição para ele (linhas 3 e 4). Se ele estiver fora da região de interesse, X ignora a requisição (linhas 5 e 6). A partir da linha 8, X sabe que ele está sobre o itinerário, pois a mensagem é destinada a ele. Com isso, ele verifica se sua distância até o PMD é menor que DI (linha 9) . Se for, tal nó finaliza o itinerário pois é o nó UNI (linhas 9 e 10). Na linha 11, X verifica se tal mensagem já foi enviada por ele e, se caso positivo, a função $manage\_sent\_message(msg)$ é chamada para verificar critérios de parada (linha 12). Na linha 13, o nó sabe que está no itinerário, que não é o nó UNI e que a mensagem nunca foi enviada por ele, ou seja, ele sabe que deve encaminhar a mensagem para o próximo nó do itinerário.

O próximo nó do itinerário é calculado de acordo com a Direção e com o DII contidos na requisição. Se a distância de X até o DII for maior que o valor de DI (linha 13), tal nó encaminha a requisição para o vizinho que estiver mais próximo do DII (linhas 24 a 36). Caso a distância de X até o DII for menor que o valor de DI, ele muda a Direção e calcula outro DII baseado nessa nova Direção (linhas 14 a 23). Se a Direção atual for CIMA ou BAIXO (linha 14), é calculado um novo DII para a nova direção que será DIREITA. Se a Direção atual for DIREITA (linha 17), a nova Direção será CIMA se X estiver mais próximo borda inferior da figura (linhas 18 a 20), e será BAIXO se X estiver mais próximo da borda superior (linhas 21 a 23). Após calculada a nova Direção e o novo DII, X envia a requisição para o vizinho que estiver mais próximo do novo DII, o qual será o próximo nó no itinerário (linhas 24 a 25).

Algorithm 1 Algoritmo para definição do itinerário.

```
1:  procedure RECEIVE_DISSEMINATION(msg)
2:      if msg.destination ≠ my_id then                  /*Se a mensagem não é destinada ao nó, ele não está no itinerário.*/
3:          if inside(msg.region) then
4:              send_reading(msg.previous)               /*Somente os nós dentro da RDI enviam seus dados.*/
5:          else
6:              drop(msg)
7:          return
8:      else if my_distance_to(PMD) ≤ DI then            /*Verifica se é o UNI.*/
9:          last_in_itinerary()
10:         return
11:     else if is_sent_msg(msg) then
12:         manage_sent_message(msg)
13:     else if my_distance(msg.dii) ≤ DI then           /*Verifica se deve mudar a Direção.*/
14:         if msg.direction = CIMA ∨ msg.direction = BAIXO then
15:             msg.direction ← DIREITA
16:             msg.dii ← next_right(msg.dii)
17:         else if msg.direction = DIREITA then
18:             if closest_top() then
19:                 msg.direction ← BAIXO
20:                 msg.dii ← next_down(msg.dii)
21:             else
22:                 msg.direction ← CIMA
23:                 msg.dii ← next_up(msg.dii)
24:     msg.destination = next_neighbor(msg.dii)         /*Obtém o identificador do vizinho mais próximo do DII corrente.*/
25:     msg.previous = myaddr
26:     send(msg)                                        /*Envia a requisição.*/
27: end procedure
```

A Figura 5(B) ilustra quando um nó perto do DII atual recebe uma requisição. Tal nó muda a Direção de CIMA para DIREITA e define uma nova linha de referência e um

novo DII. Esse nó enviará a requisição para o vizinho que estiver mais próximo do novo DII. A Figura 5(C) ilustra quando a Direção atual é DIREITA. O nó mantém a linha de referência, define o novo DII e a nova Direção para BAIXO.

O algoritmo tem dois critérios para definir o final do itinerário. O primeiro ocorre quando a requisição alcança um nó que está a uma distância de PMD menor que o valor definido para o parâmetro DI. Isso significa que a requisição percorreu toda a RDI (finalização ideal - linha 8). O segundo critério define que o itinerário finalizará se a requisição atingir um nó mais de uma vez com a mesma Direção (linha 12). Esse critério foi estabelecido para evitar ciclos no itinerário. Entretanto, quando ele ocorre, alguns nós não participam do processamento da requisição. Isso diminui a qualidade dos resultados obtidos. Quando a requisição encontra um buraco na rede (nó que não possui vizinhos mais próximos do DII que ele mesmo) utiliza-se a mesma estratégia do protocolo GPSR [18] para contorná-los.

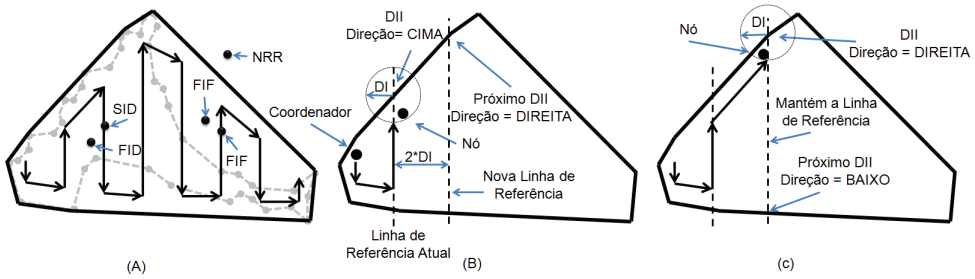

Figure 5. (A) Tipos de nós durante a disseminação. (B) Nova Direção (DIREITA) e um novo DII. (C) Nova Direção (BAIXO) e novo DII, mas mantém linha de referência.

6. AO - Estratégia para Evitar Overhearing na Disseminação

Considera-se neste trabalho que o sinal dos rádios são emitidos por difusão. Com isso, quando um dado nó transmite um pacote, todos os seus vizinhos o escutam, gastando energia desnecessariamente. Essa escuta indevida é denominada ***overhearing*** [19]. AO (*Avoid Overhearing*) é uma estratégia que reduz o consumo de energia durante o processamento de requisições espaciais evitando *overhearing*. Ela diminui o consumo de energia desligando o rádio de nós que sabem quando um de seus vizinhos realizará uma transmissão. Assim, os nós evitam receber mensagens que não são destinadas a eles. Porém essa estratégia varia de acordo com o algoritmo utilizado para processamento de requisições espaciais, como visto a seguir.

6.1. AO para o Mecanismo Básico

Em mecanismos que utilizam Inundação Restrita, como o Mecanismo Básico descrito na Seção 4, durante o Estágio de Disseminação os nós escutarão a mesma requisição várias vezes, uma de cada um de seus vizinhos, gerando um consumo de energia desnecessário. Porém, no algoritmo para Inundação Restrita aqui adotado, os nós sensores são capazes de prever essa situação e desligar seus rádios por um breve período de tempo para evitar *overhearing*. O tempo adotado neste trabalho foi de 1 segundo

A estratégia aqui utilizada desliga o rádio logo após a transmissão da requisição durante o Estágio de Disseminação, pois nesse momento todos os vizinhos desse nó também transmitirão a requisição. A Figura 6(A) ilustra parte de uma disseminação sem

a utilização de AO. Alguns nós recebem a requisição mais de uma vez. A Figura 6(B) mostra que se AO for utilizada, algumas recepções podem ser evitadas.

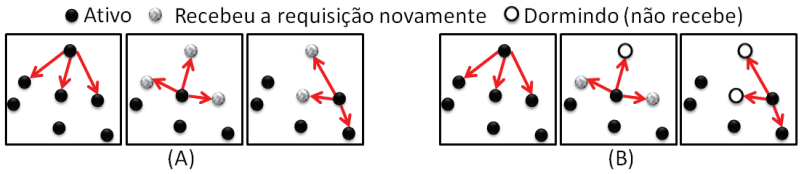

Figure 6. (A) Nós recebendo a requisição mais de uma vez. (B) AO evita recepções desnecessárias.

6.2. Evitando *Overhearing* em IBIS

Em IBIS, a estratégia para evitar *overhearing* adormece os nós no Estágio de Agregação. Com o término do contador de tempo que é iniciado no Coordenador após a disseminação da requisição, tal nó envia para o próximo nó do Itinerário o resultado da agregação dos dados coletados por ele e por seus vizinhos. O nó que recebe esse pacote realiza a mesma operação. Todos os nós sobre o itinerário participarão desse processo. Consequentemente, os vizinhos desses nós também escutam essas transmissões (*overhearing*).

Como o tempo utilizado no contador do Coordenador é conhecido por todos os nós sensores, os nós FID conseguem prever o instante em que os pacotes com os dados agregados trafegarão pela rede e podem adormecer para evitar o *overhearing* dessas transmissões. A Figura 7 ilustra os nós SID que participarão da agregação e os nós FID que adormecerão para evitar *overhearing*.

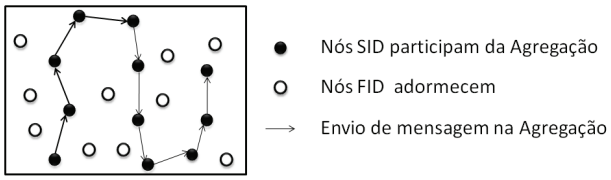

Figure 7. Nós FID adormecem durante a Agregação com AO.

7. Simulações

As propostas apresentadas neste trabalho foram avaliadas por simulações. Considera-se uma região quadrada com 1000 metros de lado. O protocolo MAC utilizado foi o 802.11. Esse protocolo possui o algoritmo de controle de colisão CSMA/CA, da mesma forma que o protocolo 802.15.4 utilizado nos nós reais. Tais redes possuem nós idênticos, estáticos, com links simétricos, com raio de alcance de 80m e GPS. As coordenadas utilizadas estão no padrão grau-decimal, com quatro casas após a vírgula, que fornece uma precisão de cerca de 11m sobre o Equador (pior caso). No início das operações da rede, o nó N_1 inicia uma Inundação. Cada pacote transmitido contém as localizações do emissor e de N_1. Com isso, todos os nós adquirem as posições de seus vizinhos e a posição do nó N_1 é utilizada como ponto de referência. Todos os pontos manipulados pela rede são posições relativas a N_1. Utilizando essa configuração, 2 bytes podem representar qualquer ponto sobre uma região quadrada de 2816m, que excede a região considerada aqui.

As propostas foram implementadas em NS-2.34 [20]. Cada ponto plotado nos gráficos representa a média de 60 simulações, com intervalo de segurança de 95%. Foi desenvolvido um aplicativo, em Java, que habilita a criação de RDIs sobre fotos de satélite.

Os experimentos analisaram duas métricas: consumo de energia e imprecisão do resultado. A **imprecisão** é definida por $I = ((NRND + NNRD)/NDR) \times 100$ onde NRND é o número de nós que responderam a requisição, mas não deveriam; NNRD é o número de nós que não responderam, mas deveriam; e NDR o número de nós dentro da RDI.

Devido à infinita possibilidade de RDI, foi escolhido o contorno real de um lago exibido na Figura 1(A), o qual é representado por 139 pontos. O algoritmo definido em [7] foi utilizado para reduzir o número de pontos dessa representação. Com isso, utilizando pacotes com 28 bytes (tamanho típico em nós sensores reais), a requisição ocupa no mínimo 1 e no máximo 10 pacotes. O eixo X em cada um dos gráficos representa o número de pacotes necessários para transmitir uma requisição. Com a redução do número de pontos, o contorno da RDI se modifica englobando nós que não deveriam participar do processamento (NNRD) e/ou excluindo nós que não estão dentro da RDI original (NRND). Porém, como visto nos gráficos, a economia de energia é significante.

O tempo gasto para RSSF processar um requisição, ou latência do processamento, não foi descrito em gráficos porque ele depende quase que exclusivamente do valor dado ao contador de tempo do Coordenador (T_{max}). Tal valor define quanto tempo após o início da Disseminação o Coordenador iniciará o Retorno do resultado da requisição. No Mecanismo básico, esse valor segue a seguinte equação: $T_{max} \geq FD * H$, onde H é a altura da árvore de roteamento formada na RDI. Nas simulações realizadas com o mecanismo básico, a árvore com maior altura possuía 25 níveis. Por isso, definiu-se $T_{max} = 25$ segundos. Em IBIS, para garantir o sucesso do processamento, o Coordenador deve esperar que a requisição alcance o UNI e só então iniciar a Agregação. Nas simulações, 16 segundos foi o maior tempo gasto para a requisição percorrer todo o itinerário. Nos dois mecanismos, o tempo para encaminhamento e retorno foi sempre menor que 1 segundo. Com isso, o tempo de simulação do mecanismo básico é de aproximadamente 25 segundos e de IBIS de aproximadamente 16 segundos.

O modelo de energia baseia-se em [21] que utiliza um modelo de rádio que elimina o consumo de energia do nó no estado ativo, mas sem transmitir ou receber (*idle listening*). Além disso, como o consumo no estado dormindo é pequeno, ele também não foi considerado. Considera-se neste trabalho transmissões em *broadcast*, ou seja, todos os vizinhos do emissor escutam o pacote. Com isso, a energia consumida pela rede para transmitir um pacote (E_{PT}) é igual à energia para transmissão de um pacote (E_{TX}), mais a soma da energia consumida por cada um dos n vizinhos do emissor para recebê-lo ($\sum_{x=1}^{n} E_{RE}$). Esse consumo pode ser modelado pela equação: $E_{(i \rightarrow j)PT} = E_{(i)TR} + \sum_{x=1}^{n} E_{(x)RE}$. Considera-se a potência de 0.048W para recepção e 0.051W para transmissão. Esses valores estão de acordo com o nó sensor Iris que utiliza duas pilhas AA de 1,5V [22].

7.1. Análise do Mecanismo Básico

Este cenário faz a estimativa do melhor valor para FD. Durante a disseminação, os nós iniciam o cronômetro T_{agr} que reduz FD segundos a cada nível na árvore de roteamento. Ou seja, $T = T_{max} - nivel_i * FD$, onde T_{Max} é o maior tempo de espera e $nivel_i$ indica o nível do nó i na árvore de roteamento. A Figura 8 exibe a imprecisão (eixo Y) para diferentes valores de FD. As requisições não podem ser disseminadas corretamente quando FD possui valores pequenos, pois os cronômetros dos nós podem terminar antes que seus descendentes disseminem a requisição. Consequentemente, a imprecisão se torna alta pois as informações coletadas por esses descendentes são perdidas. Tal situação acon-

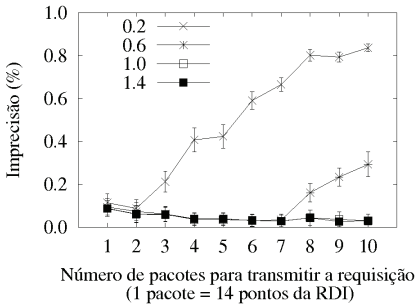

Figure 8. Analise do melhor valor para FD.

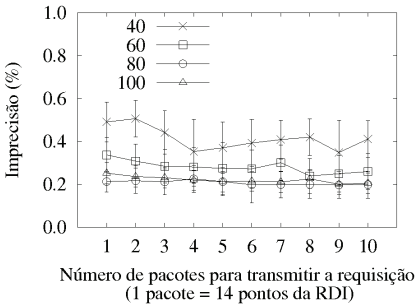

Figure 9. Imprecisão para diferentes valores de DI.

tece, principalmente, quando a requisição é formada por vários pacotes. A partir dessas simulações, definiu-se o valor padrão de $FD = 1.0$ segundo.

7.2. Avaliação de IBIS

Foram realizadas simulações com o intuito de avaliar o melhor valor para DI. A Figura 9 exibe o gráfico com a imprecisão obtida para diferentes valores de DI com o aumento do número de pacotes necessários para transmitir a requisição. Quando DI possui valores pequenos, o itinerário passa sobre o mesmo nó várias vezes. Com isso, a imprecisão é maior, uma vez que a disseminação é interrompida antes de alcançar o UNI (segundo critério de parada). Consequentemente, tem-se menor imprecisão quando DI assume valores maiores (80m e 100m), próximo ao alcance do rádio. A Figura 10 exibe o consumo de energia para a mesma situação. Quando os maiores valores de ID são utilizados, a distância entre as linhas de referência no itinerário é maior. Com isso, o consumo de energia é menor devido ao menor número de nós que recebem a requisição. O gráfico também mostra que o algoritmo apresentado em nosso trabalho anterior [7] reduz consideravelmente o consumo de energia quando empregado com IBIS, uma vez que reduz o número de pacotes necessários para transmitir a requisição.

7.3. Comparação entre o algoritmos

Esta seção compara o Mecanismo Básico (Basic), o Mecanismo Básico com AO (Basic-AO), o mecanismo IBIS sem AO (IBIS) e IBIS com AO (IBIS-AO). A Figura 11 mostra que IBIS e IBIS-AO são mais imprecisos. Isso ocorre devido ao segundo critério de parada que termina o itinerário antes que requisição alcance o UNI. Entretanto, esta imprecisão assume valores pequenos. A Figura 12 exibe o baixo consumo de energia de IBIS e IBIS-AO. Devido à criação do itinerário, a requisição é transmitida menos vezes que no Mecanismo Básico.

8. Conclusão e Trabalhos Futuros

Este trabalho apresenta duas contribuições que reduzem o consumo de energia em RSSF que realizam o processamento de requisições espaciais com RDIs em formato irregular. Simulações mostraram o baixo consumo de energia e a pequena imprecisão de IBIS. Mostraram também que AO, diminui consideravelmente o consumo de energia quando aplicada sobre o mecanismo básico. O algoritmo apresentado em nosso trabalho anterior [7], que reduz o número de pontos necessários para representar a requisição, também reduz o consumo de energia quando aplicado a IBIS. Com isso, conclui-se que IBIS é um

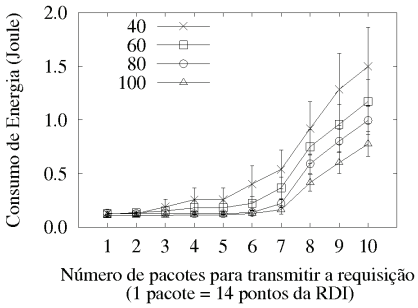

Figure 10. Consumo de energia para diferentes valores de DI.

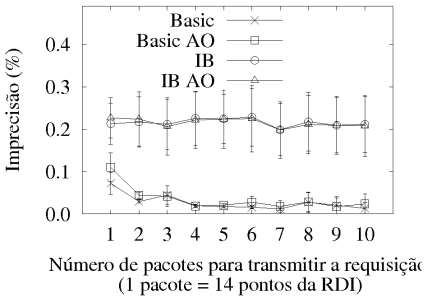

Figure 11. Imprecisão dos mecanismos.

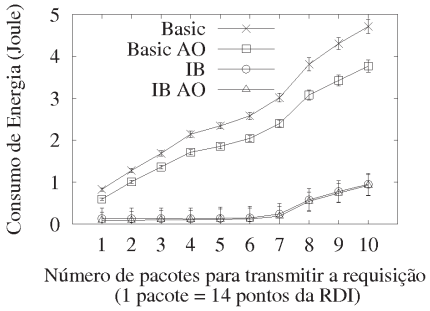

Figure 12. Consumo de energia dos mecanismos.

bom algoritmo quando o tempo de vida da rede é a prioridade. Quando a precisão é a métrica mais importante, mecanismos baseados em Inundação Restrita apresentam melhores resultados. Nessas situações, AO reduz consideravelmente o consumo de energia.

Em trabalhos futuros, pretende-se reduzir a latência no processamento dos mecanismos propostos. O segundo critério de parada de IBIS, criado para evitar ciclos, deve ser redefinido para reduzir a imprecisão. Além disso, AO necessita ser melhor adaptado para reduzir o consumo de energia quando utilizado com IBIS.

Referências

[1] M. Demirbas and X. Lu, "Distributed quad-tree for spatial querying in wireless sensor networks," in *IEEE International Conference on Communications (ICC)*, 2007, pp. 3325–3332.

[2] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, March 2005.

[3] A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao, "The cougar project: a work-in-progress report," *ACM Special Interest Group on Management of Data (SIGMOD)*, vol. 32, no. 4, pp. 53–59, Dec. 2003.

[4] A. Deshpande, Z. Ives, and V. Raman, "Adaptive query processing," *Foundations and Trends in Databases*, vol. 1, no. 1, pp. 1–140, 2007.

[5] Y. Manolopoulos, A. Papadopoulos, and M. Vassilakopoulos, Eds., *Spatial Databases: Technologies, Techniques and Trends*. Idea Group, 2005.

[6] M. A. Casanova, G. Câmara, C. A. Davis, L. Vinhas, and G. R. de Queiroz, *Banco de Dados Geográfico*. MundoGeo, May 2005.

[7] R. I. da Silva, V. del Duca Almeida, A. M. Poersch, and J. M. S. Nogueira, "Spatial query processing in wireless sensor network for disaster management," in *2nd IFIP Wireless Days*, december 2009.

[8] J. Wang, R. K. Ghosh, and S. K. Das, "A survey on sensor localization," *Journal of Control Theory and Applications*, vol. 8, no. 1, pp. 2–11, January 2010.

[9] S.-H. Wu, K.-T. Chuang, C.-M. Chen, and M.-S. Chen, "DIKNN: An itinerary-based KNN query processing algorithm for mobile sensor networks," in *International Conference on Data Engineering (ICDE)*, 2007, pp. 456–465.

[10] T.-Y. Fu, W.-C. Peng, and W.-C. Lee, "Parallelizing itinerary-based knn query processing in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 22, pp. 711–729, 2010.

[11] A. Coman, J. Sander, and M. A. Nascimento, "Adaptive processing of historical spatial range queries in peer-to-peer sensor networks," *Distributed and Parallel Databases*, vol. 22, no. 2-3, pp. 133–163, 2007.

[12] F. Zhao and L. J. Guibas, *Wireless sensor networks: an information processing approach*. The Morgan Kaufmann series in networking., 2004.

[13] D. Goldin, M. Song, A. Kutlu, H. Gao, and H. Dave, "Georouting and deltagathering: Efficient data propagation techniques for geosensor networks," in *First Workshop on Geo Sensor Networks*, 2003, pp. 9–11.

[14] A. Soheili, V. Kalogeraki, D. Gunopulos *et al.*, "Spatial queries in sensor networks," in *3th Annual ACM International Workshop on Geographic Information Systems (GIS)*. ACM, 2005, pp. 61–70.

[15] K. Park, B. Lee, and R. Elmasri, "Energy efficient spatial query processing in wireless sensor networks," in *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW)*. IEEE Computer Society, 2007, pp. 719–724.

[16] Y. Li, S.-H. Eo, D.-W. Lee, Y.-H. Oh, and H.-Y. Bae, "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)*. IEEE Computer Society, 2008, pp. 373–379.

[17] Y. Xu, W. chien Lee, J. Xu, and G. Mitchell, "Processing window queries in wireless sensor networks," in *22th IEEE International Conference on Data Engineering (ICDE)*, 2006.

[18] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *6th Annual International Conference on Mobile Computing and Networking (MobiCom)*. New York, NY, USA: ACM, 2000, pp. 243–254.

[19] P. Basu and J. Redi, "Effect of overhearing transmissions on energy efficiency in dense sensor networks," in *IPSN '04: Proceedings of the 3rd international symposium on Information processing in sensor networks*. New York, NY, USA: ACM, 2004, pp. 196–204.

[20] "The Network Simulator NS-2," http://www.isi.edu/nsnam/ns/.

[21] R. Jurdak, A. G. Ruzzelli, and G. M. P. O'Hare, "Radio sleep mode optimization in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 955–968, 2010.

[22] *Iris Mote Plataform*, MEMSIC Corporation Std.