Como obter o Mapa de Energia em Rede de Sensores Sem Fio? Uma Abordagem Tolerante a Falhas

Marcos Augusto M. Vieira[1], Luiz Filipe M. Vieira[1], Linnyer Beatrys Ruiz[1,3], Antônio Alfredo F. Loureiro[1], Antônio O. Fernandes[1], José Marcos S. Nogueira[1], Diógenes Cecílio da Silva Jr. [2]

[1] Departamento Ciência da Computação e Departamento de Engenharia Elétrica[2] - Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627 – 31.270-010 – Belo Horizonte – MG – Brasil
{mmvieira,lfvieira, linnyer, loureiro,Otavio, jmarcos@dcc.ufmg.br}
[3] Pontifícia Universidade Católica do Paraná
R. Imaculada Conceição, 1155 – CEP 80215-901 Curitiba-Paraná

Resumo— **Redes de Sensores Sem Fio (RSSFs) são redes compostas por um grande número de dispositivos com capacidade de processamento, comunicação e sensoriamento. Estes dispositivos, chamados nós sensores, são projetados com pequenas dimensões (cm^3 ou mm^3) o que limita sua capacidade de hardware. Neste trabalho, duas soluções são apresentadas, SmartSink e Lista de Padrasto, para corrigir a árvore de roteamento, estendendo o tempo de vida da RSSF e adicionando um mecanismo tolerante à falhas. SmartSink pode corrigir a árvore de roteamento, gastando uma mensagem por nó sensor. Um nó sensor tem uma lista de pais potenciais (chamado de lista de padrasto) para rotear mensagens. Suas vantagens são mostradas com uma aplicação para obter o mapa de energia de uma RSSF. Resultados da simulação mostram que o tempo de vida da RSSF aumenta pelo menos três vezes quando comparado com trabalhos anteriores**.

Palavras-chave— **Rede de Sensores Sem Fio, Algoritmo Distribuído, Tolerância a Falha, Mapa de Energia**

I. INTRODUÇÃO

Rede de Sensores Sem Fio (RSSF) são redes formadas por microsensores compactos com capacidade de comunicação sem fio. Estes pequenos dispositivos (nós sensores) têm o potencial de serem disseminados em grande quantidade. Como dispositivos autônomos, eles podem prover distribuição pervasiva. Vários desafios aparecem para os projetistas como poder computacional, consumo de energia, fontes de energia, canais de comunicação e capacidade dos sensores.

Uma RSSF tem o potencial de ser usada em diferentes aplicações. Por exemplo, na indústria para instrumentação das fábricas; em grandes metrópoles para monitorar densidade de tráfego; na engenharia para monitorar prédios; no meio-ambiente para monitorar florestas, oceanos, agricultura de precisão, etc. Outras aplicações incluem gerência de complexos sistemas físicos como asa de avião e ecossistemas [5] [7].

É desejável que a RSSF seja eficiente em termos de consumo de energia. Além disso, os protocolos de rede devem ser projetados para obter tolerância a falha quando um nó sensor falhar enquanto tenta minimizar o consumo de energia. Algoritmos propostos na literatura tendem a tratar apenas um dos aspectos, como energia. Este é o caso de protocolos de roteamento que tentam balancear o consumo de energia [3][4] ou usar árvore de roteamento [2] para economizar energia com agregação de dados e cache.

Neste trabalho, duas soluções são propostas, SmartSink e Lista de Padrasto, explicado abaixo, para corrigir a árvore geradora de roteamento, estendendo o tempo de vida da RSSF e sendo tolerante a falha. SmartSink é um ponto de acesso, que tem suas funcionalidades estendidas, e corrige a árvore de roteamento, gastando uma mensagem por nó sensor. Lista de Padrasto consiste em cada nó sensor ter uma lista de potências pais (chamado de lista de padrasto) para rotear mensagens. O algoritmo para descoberta da lista de padrasto também é apresentado. Quando um nó sensor perde o seu pai, o nó sensor pode substituir seu pai na árvore de roteamento usando a lista de padrasto. Este mecanismo é tolerante a falhas, corrigindo a árvore de roteamento dinamicamente. Resultados da simulação mostram que o tempo de vida da rede aumenta em pelo menos três vezes quando comparado com trabalhos anteriores.

Este trabalho está organizado como a seguir. A Seção II descreve a arquitetura de uma RSSF. Em seguida, na Seção III os trabalhos relacionados são abordados. A quarta seção descreve as duas soluções propostas. As seções seguintes descrevem o modelo de rádio e o processo de simulação, resultados obtidos e análise destes resultados. A última seção contém as conclusões deste trabalho.

II. ARQUITETURA RSSF

Esta seção descreve a arquitetura de RSSF. RSSFs são redes compostas por um grande número de nós sensores. O objetivo de tais redes é coletar informação e eventualmente controlar o ambiente. Os nós sensores são distribuídos pelo ambiente que se deseja monitorar, despertam, realizam auto-teste, descobrem sua localização, auto organizam e, a partir de então, estabelecem dinamicamente canais de comunicação entre si, formando uma rede ad hoc.

Uma RSSF não possui uma infra-estrutura de comunicação pré-construída, como ocorre, por exemplo, com redes de telefonia celular ou redes locais sem fio. As RSSFs formam uma rede ad hoc, cuja topologia é dinâmica devido ao fato de nós sensores saírem de serviço temporariamente ou definitivamente por diferentes razões como, por exemplo esgotamento de energia. Um ponto importante destas redes é que o fluxo de dados é tipicamente unidirecional, isto é, os dados coletados flui dos nós sensores fonte para um ou mais pontos de acesso (estação base, nó sorvedouro, gateways).

Um ponto de acesso é um nó sensor que na maioria dos casos difere dos outros nós sensores. O ponto de acesso tem a finalidade de coletar informação da rede e enviar para um observador. Ele conecta a RSSF com o mundo exterior. Em geral os nós pontos de acesso apresentam mais energia que os demais, maior alcance de rádio e não apresentam funcionalidade de sensoriamento.

Em função das pequenas dimensões das antenas, no caso de transceptores de rádio freqüência e das políticas de economia de energia, a comunicação entre nós fontes e o ponto de acesso é multi-hop. Na grande maioria dos casos, os nós sensores não possuem um canal de comunicação direto com o ponto de acesso, o que demanda que nós intermediários sirvam de roteadores para a comunicação. Assim, nesta arquitetura, cada nó sensor é potencialmente também um roteador.

O grande número de nós sensores na RSSF faz com que a coleta individual de dados de cada nó sensor seja inviável. Além disso, o conhecimento do estado global de uma região é mais importante que conhecimento de cada estado dos nós sensores. A fusão de dados ajuda a reduzir a quantidade de dados transmitidos, salvando energia.

O recurso mais crítico nas RSSFs é a energia. Cada nó sensor é composto por uma pequena bateria, com capacidade limitada. É praticamente inviável recarregar todas as baterias, pois as RSSFs são compostas por grande quantidade de nós sensores. Portanto, o foco principal no projeto de RSSF, do projeto de hardware até os protocolos de rede, é a economia de energia.

As redes de sensores podem ser classificadas [5] segundo a sua organização em rede hierárquica (os nós se organizam em grupos) e rede plana, segundo a sua composição em homogênea (mesmo tipo de nós sensores) ou heterogênea (diferentes tipos), mensagens enviadas ao gateway seguem o mesmo modelo de enviadas do gateway (simétricas e assimétricas).

III. Trabalhos Relacionados

Residual Energy Scan, também conhecido com eScan[2], é um método de obter a distribuição geográfica dos recursos da rede ou das atividades dos sensores, como a energia. Por exemplo, o eScan provê uma visão abstrata da distribuição de energia ao invés da informação detalhada da energia residual de cada nó. Ele agrega dados, reduzindo a quantidade de dados transmitidos, mas introduz uma distorção na medida. Existe um *tradeoff* entre a quantidade de dados transmitidos e a sua precisão.

O problema com o Residual Energy Scan é o fato do roteamento tender a sobrecarregar os nós pais. O roteamento é feito de maneira que todas as mensagens do nó são enviadas para outro nó, chamados de nó pai, no qual está mais perto do ponto de acesso, e assim por diante. Os nós pais tendem a receber e transmitir mais mensagens do que seus descendentes, levando a um consumo de energia não balanceado. Portanto, os nós pertos dos pontos de acesso irão receber e transmitir mais mensagens do que os nós mais distantes. Isto pode causar uma morte prematura da rede, mesmo que existam nós com energia suficiente para operar mas suas mensagens não chegarão ao ponto de acesso.

Outro problema com o Residual Energy Scan é que ele não é tolerante a falha. Suponha que aconteça uma disfunção no nó sensor durante uma pesquisa. Os dados enviados pelos nós descendentes não estarão disponíveis para o ponto de acesso e esses nós sensores ficarão órfãos. Em RSSF, uma falha é um caso comum, não uma exceção. Aqui é apresentada uma solução para estes dois problemas.

PEGASIS [3] foi projetado para ser um protocolo de roteamento eficiente em energia. Cada nó sensor comunica apenas com o seu vizinho, e faz rodízio para decidir quem irá transmitir para a estação base. Portanto, não é possível usar cache e agregação para reduzir o número de mensagens. PEGASIS assume que todos os nós sensores têm conhecimento global da rede.

LEACH (Low Energy Adaptive Clustering Hierarchy) [4] é um protocolo de comunicação eficiente em energia para RSSF. LEACH é um algoritmo de roteamento baseado em aglomerados (cluster) no qual o próprio sensor pode se eleger líder. Esses líderes coletam dados dos sensores que pertencem ao seu aglomerado, agregam os dados e transmitem para uma estação base. Cada líder continua como chefe do aglomerado por um período de tempo conhecido como rodada. No começo de cada rodada, cada nó determina se pode ser um líder. Se puder ser líder, o nó avisa a sua decisão para os seus vizinhos. Os nós que optaram não serem líderes ouvem os sinais de rádio e optam por entrar em um aglomerado.

As vantagens do LEACH são a coordenação local para criar e operar os aglomerados, rotação aleatória dos líderes e fusão local dos dados. As desvantagens são que o LEACH não leva em consideração a não disponibilidade dos nós sensores (neste caso, quando um líder fica indisponível, todas as mensagens enviadas são perdidas naquela rodada); não garante grupos uniformemente distribuídos pela rede; e como cada nó sensor faz rodízio para transmitir para estação base, não é possível usar cache e outras técnicas para predizer quando uma mensagem seria enviada.

IV. Princípios introduzidos

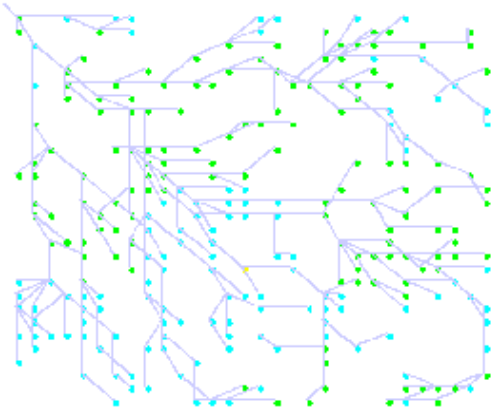
Neste trabalho, os princípios do SmartSink e Lista de Padrasto, como explicado abaixo, são aplicados em um

algoritmo que utiliza árvore de roteamento [2] para estender o tempo de vida da RSSF e se tornar tolerante a falhas.

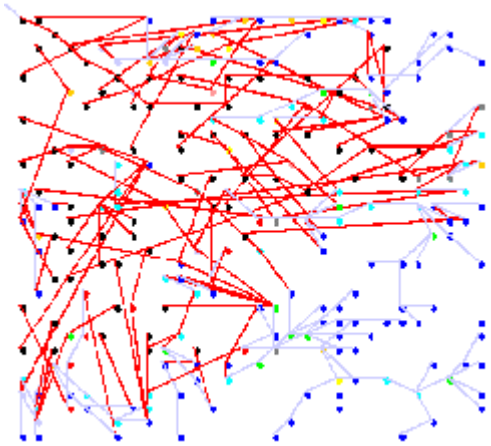
A. SmartSink

SmartSink é o nome do ponto de acesso que tem suas funcionalidades estendidas. O propósito do ponto de acesso é coletar informação da rede e enviar para um observador externo. Em geral, este nó possui mais energia, maior alcance de rádio, não precisa de sensores e possui conhecimento global da RSSF, pois conecta a RSSF com o mundo exterior. A idéia é estender as funcionalidades do ponto de acesso para corrigir a árvore de roteamento. Ele irá ajudar o algoritmo Residual Energy Scan [2] a obter tolerância a falha na presença de falha individuais dos nós e compartilhará os recursos da rede de melhor maneira. A idéia é criar o próprio mapa de energia, usando o algoritmo baseado no Residual Scan, para permitir tolerância à falha e corrigir problemas da rede. Quando o ponto de acesso recebe informação de uma consulta, ele tem conhecimento global do mapa de energia, e, portanto, pode configurar o roteamento.

O processo de configurar o roteamento é descrito a seguir. Quando o SmartSink recebe o resultado de uma consulta, ele verifica quais nós estão em estado crítico de energia. No caso de algum nó sensor poder ficar órfão, o SmartSink pode escolher um novo nó para ser seu pai e notificá-lo da mudança.



a)



b)

Figura 1: RSSF e sua árvore de roteamento sendo dinamicamente alterada pelo SmartSink.

A Figura 1 mostra a "foto instantânea" de duas RSSFs. Ela ilustra o funcionamento do mecanismo do SmartSink. Cada ponto representa um nó sensor na RSSF, e uma linha conectando dois pontos representa um canal de comunicação entre dois nós sensores. A Figura 1a mostra a RSSF no início da simulação. Os nós sensores perto do ponto de acesso (que está localizado no canto esquerdo superior) perdem energia mais rápido do que os outro nós sensores, não ocorrendo um consumo balanceado da energia. A solução é o mecanismo do SmartSink, no qual prevê que um nó irá ficar órfão e pode informar para o nó o seu novo pai. Este novo canal de comunicação está representado na Figura 1b como a linha mais escura (linha vermelha). Se for assumido que o SmartSink pode comunicar diretamente com os nós sensores, o custo extra para mudar o nó pai será de apenas uma mensagem.

A funcionalidade do nó é associada ao nível de energia. Um nó com pouca energia será um nó folha na árvore de roteamento. Se o nó tiver pouca energia, ele irá apenas sentir o ambiente e enviar dados coletados, deixando de ser um hop de comunicação. Desta maneira, o recurso de energia da rede será melhor utilizado pois o consumo de energia será melhor balanceado entre os nós.

Com a utilização do SmartSink, a aplicação para obter o mapa de energia adquire característica de tolerância à falhas. O SmartSink prevê quem vai morrer e corrige a árvore de roteamento. Desta maneira, quando um nó morre, os dados dos descendentes continuarão sendo conhecidos pois estes trocaram de pais. Se um nó prevê que irá morrer por outro motivo que não seja a falta de energia (por exemplo, temperatura muito alta), bastará o nó enviar uma mensagem de notificação avisando que irá morrer. Caso a morte de um nó não seja previsível, como no caso de destruição física, a solução proposta por Ganesan [6] pode ser usada. Portanto, a idéia principal do mecanismo do SmartSink é usar o mapa de energia para prever que um nó irá ficar órfão e mudar o seu pai, corrigindo a árvore de roteamento.

Dois fatores são considerados pelo SmartSink para escolher o novo pai: a energia e a localização dos nós. Estas informações são obtidas pelo SmartSink quando ele recebe os dados de sua consulta a RSSF. Com a localização dos nós, é possível estimar a distância entre os nós. O modelo de rádio adotado assume que o consumo de energia é proporcional ao quadrado da distância.

O novo pai será aquele que puder transmitir mais mensagens ao filho, ou seja, tiver a maior relação energia por custo de transmitir mensagem. Outra condição é que o novo pai não pode fazer um ciclo na árvore de roteamento. Isto pode levar ao cenário em que não é possível trocar o nó pai. A complexidade do algoritmo para cada nó é O(n).

Outros problemas que podem acontecer [10] e podem ser resolvidos pelo SmartSink são: um nó sensor perder a transmissão, alguns links serem mais distantes que o esperado, e o link no qual o receptor do "flooding" está mais perto da estação base do que o transmissor (backward link). O SmartSink corrige a árvore geradora ao enviar uma mensagem para cada nó sensor que é responsável por uma aresta mal formada na árvore geradora.

B. Lista de Padrasto

Outra contribuição deste trabalho é a Lista de Padrasto. Um mecanismo tolerante a falha para obter o mapa de energia de RSSF consiste em incluir uma lista, em cada nó sensor, os nós sensores que poderão substituir seu pai na árvore de roteamento. Estes nós potenciais são chamados de padrasto. Uma vez que um nó pai morre, os descendentes destes podem trocar de pai, e continuarem a funcionar na RSSF.

```
Algoritmo Descoberta_de_Padrasto
∇Variáveis:
    Lista de Padrasto_i=∅;
    EnviouMensagem_i=falso;
∇Entrada:
    mensagem_i=nulo;
    Ação se n_i ∈No:
      EnviouMensagem_i=verdadeiro;
      Enviar inf para todo n_j ∈ Vizinho_i;
∇Entrada:
    mensagem_i=inf tal que origem_i (mensagem_i)=(n_i,n_j);
    Ação:
      Se não EnviouMensagem_i então
        Início
          EnviouMensagem_i :=verdadeiro;
          Adicione n_j para Lista de Padrasto_i;
          Enviar inf para todo n_k ∈ Vizinho_i tal que n_k
∉ Lista de Padrasto_i;
        Fim
      Senão adicione n_j para Lista de Padrasto_i.
```

Figura 2: Algoritmo Descoberta_de_Padrasto

O algoritmo para construir a Lista de Padrasto é muito similar com o algoritmo de propagação da informação (PI)[9]. A primeira vez que a estação base envia uma requisição, o algoritmo de descoberta de padrasto é processado, como mostrado na Figura 2. O conjunto N_0 representa os nós sensores que recebem a requisição da estação base. O conjunto $Vizinho_i$ contém os vizinhos do nó sensor i. Todo nó que recebe uma mensagem mantém a informação do remetente na Lista de Padrasto e envia uma nova mensagem para todos os seus vizinhos que não estão na lista de padrasto. Esta última condição é necessária para evitar a formação de um ciclo na árvore de roteamento num estado futuro da RSSF.

O algoritmo assume que verificar a lista de padrasto e enviar uma mensagem é uma operação atômica. Receber uma mensagem e atualizar a lista de padrasto também é uma operação atômica. Estas condições previnem um nó sensor de enviar mensagem para seu pai, o que poderia ter acontecido ao receber uma mensagem depois de verificar a lista de padrasto e antes de transmitir a mensagem.

Outra premissa é que cada nó não pode receber e enviar uma mensagem ao mesmo tempo. Isto poderia levar a situação onde o nó A adiciona o nó B em sua lista de padrasto e o nó B faz o mesmo, criando a possibilidade de formação de ciclo no futuro. Se o canal de comunicação entre os nós sensores se comportar como uma fila (FIFO), quando o nó sensor escolher um padrasto, nunca irá ocorrer a formação de um ciclo.

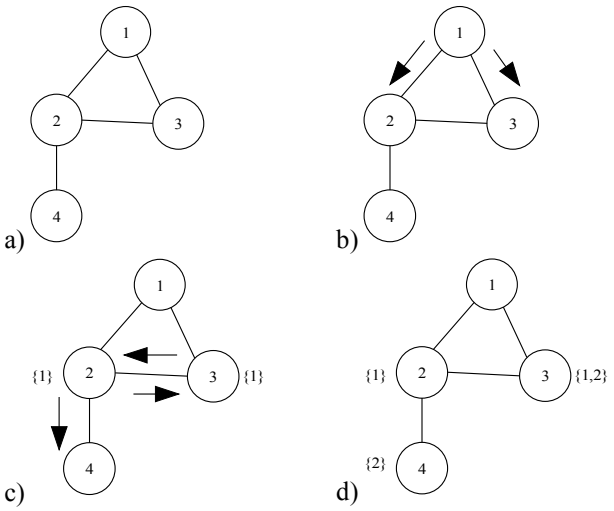


Figura 3: Exemplo do Algoritmo de Descoberta de Padrasto.

A Figura 3 ilustra o funcionamento do algoritmo. A parte a) é a topologia da RSSF. O nó sensor 1 é o responsável de iniciar o algoritmo. Ele envia mensagem para os nós sensores 2 e 3 como mostrado na parte b). Estes nós irão receber as mensagens e irão enviar mensagens para seus vizinhos exceto os nós que estão na lista de padrasto (neste instante apenas o nó sensor 1). A parte c) ilustra este momento e a respectiva lista de padrasto. Uma colisão na camada MAC poderia ocorrer, e eventualmente, um nó sensor irá transmitir primeiro. Neste exemplo, nó sensor 2 transmite a mensagem. Nó sensor 3 recebe a mensagem e atualiza sua lista de padrasto. Nó sensor 3 não irá enviar mensagem para nó sensor 2. Finalmente, nó sensor 4 recebe sua mensagem e a lista de padrasto final é mostrada na parte d).

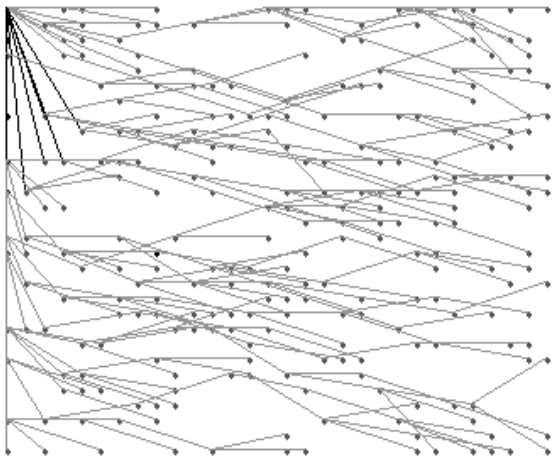


Figura 4: RSSF usando o algoritmo da lista de Padrasto.

Figura 4 mostra a foto instantânea da simulação da RSSF usando o mecanismo da lista de Padrasto. As linhas escuras no canto esquerdo superior indicam a mudança de pais. Também é possível observar a árvore de roteamento. O algoritmo foi melhorado para evitar que nós sensores mais perto do ponto de acesso morram primeiro. Se um nó perde todos os nós de sua lista de padrastos, ele fica sem nós intermediários para comunicar com o ponto de acesso. O ponto de acesso é inserido por último na lista de padrasto em todos os nós sensores. Quando um nó perde o pai, e nenhum outro na sua lista serve para ser pai, ele ainda tem como opção conversar diretamente com o ponto de acesso.

O algoritmo da lista de padrasto introduz tolerância a falha na organização da rede, sem precisar de uma entidade externa. Neste caso, os nós sensores não precisam ter conhecimento global da rede. Normalmente, o critério para escolher o novo pai é a menor distância.

V. Modelo de Rádio e Simulação

Assumiu-se que um modelo simples de rádio onde o rádio dissipa E_{elec} =180nJ/bit para executar o circuito de recepção ou transmissão e $\in_{amp}$= 10pJ/bit/m^2 para o amplificador. Estes valores foram apresentados em [1]. Também pressupõe perda de energia de r^2 devido ao canal de transmissão. Este modelo de rádio é o mesmo que foi adotado em [4].

Para avaliar os modelos, uma RSSF foi simulada para cada modelo. A rede era bidimensional (2D), não hierárquica e homogênea. O posicionamento dos sensores foi feito de maneira aleatória. Experimentos para determinar o tempo de vida da rede (número de rodadas) com 1%, 20%, 50%, 90% dos nós morrendo foram feitos. Todos os nós começavam com 0.30 Joules de energia. Redes são localizadas em um quadrado de 30x30m com 250 nós, 40x40m com 450 nós e 50x50 m com 700 nós, mantendo a densidade da rede quase constante. A métrica utilizada foi o tempo de vida da rede.

Foi pressuposto que quando um nó morre, seus descendentes também morrem. Esta premissa é válida porque quando o nó pai morre, usando o algoritmo original, não há como obter dados dos nós descendentes, ficando esses nós inúteis.

Também foi assumido que as mensagens tinham tamanho de 200 bits e que em todas as rodadas todos os sensores tinham dados a serem transmitidos ao ponto de acesso. Na versão original do Residual Energy Scan, os sensores transmitiam dados apenas quando algum evento ocorresse. Esta modificação na simulação não modifica o algoritmo, apenas faz os nós perderem energia mais rápido, permitindo o algoritmo ser independente do modelo de energia.

VI. Resultados Experimentais

A Tabela 1 mostra os resultados obtidos com a rede 30 x 30 m. A Tabela 2 mostra os resultados para 40 x 40 m e a Tabela 3 para 50 x 50 m.

Tabela 1: Número de rodadas quando 1%, 20%, 50%, 90% dos nós morrem para rede de 250 nós

Protocolo	1%	20%	50%	90%
Residual E-Scan	17	24	26	31
SmartSink	52	128	179	226
Lista de padrastos	28	168	337	1001

Tabela 2: Número de rodadas quando 1%, 20%, 50%, 90% dos nós morrem para rede de 450 nós.

Protocolo	1%	20%	50%	90%
sem SmartSink	12	15	17	23
SmartSink	37	118	194	242
Lista de padrastos	22	134	284	893

Tabela 3: Número de rodadas quando 1%, 20%, 50%, 90% dos nós morrem para rede de 700 nós.

Protocolo	1%	20%	50%	90%
sem SmartSink	7	8	8	9
SmartSink	25	93	172	225
Lista de padrastos	26	118	254	795

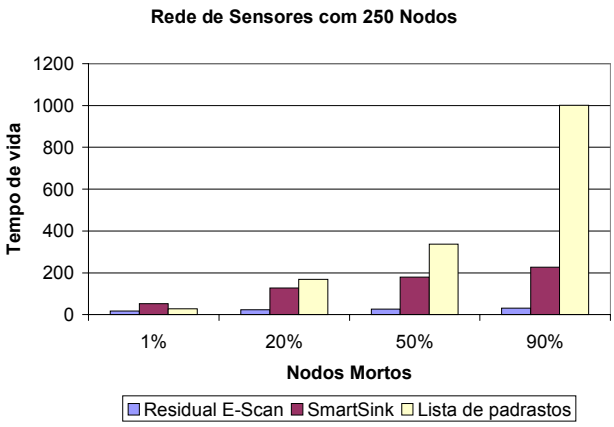


Figura 5: Resultados obtidos com a rede de 250 nós sensores.

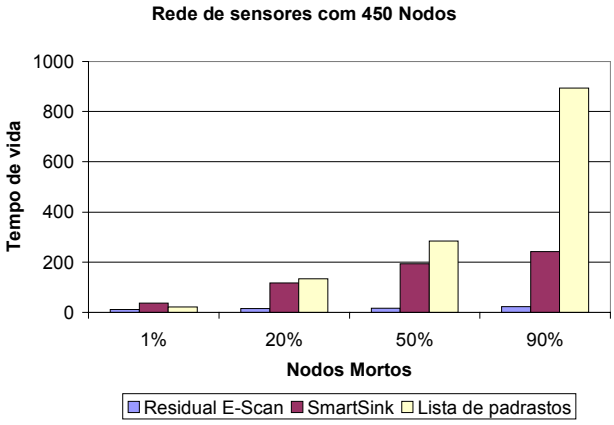


Figura 6: Resultados obtidos com a rede de 450 nós.

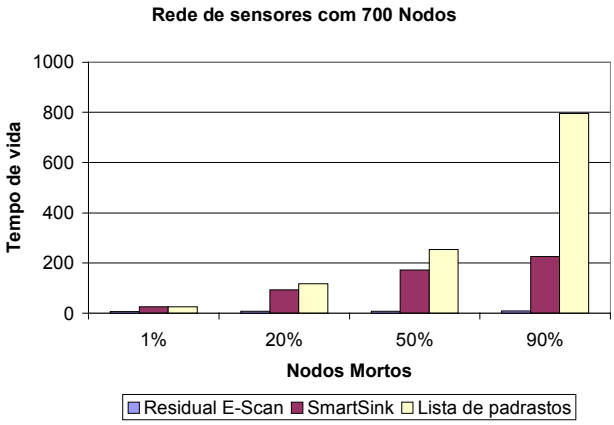


Figura 7: Resultados obtidos com a rede de 700 nós.

VII. ANALISE

A. Smart Sink

Simulações mostram que o tempo de vida da RSSF aumenta quando usando SmartSink comparado ao Residual e-scan. SmartSink pode rearranjar a árvore geradora de roteamento, usando energia como métrica principal, o que distribui a dissipação de energia entre os nós sensores. Também provê escalabilidade. Aumentando o número de nós sensores, SmartSink melhora a performance dos seus resultados também. A razão para este fato é que a chance de obter um pai mais perto é maior.

Durante os experimentos com RSSF sem SmartSink, normalmente o primeiro nó sensor a morrer é o nó mais perto do ponto de acesso, pois ele tem que rotear quase todas as mensagens ao ponto de acesso. Mesmo aumentando a porcentagem de nós que morrem, a rede sem o SmartSink tende a ter o tempo de vida constante. Isso ocorre devido à premissa que quando o nó pai morre, o filho também morre porque ele fica incomunicável. Como normalmente o primeiro nó a morrer é o nó mais perto do ponto de acesso, quase toda a rede tenderá a morrer. Este efeito não ocorre com a rede que usa o SmartSink, pois aumentando a porcentagem de nós que morrem, o SmartSink atuará, balanceando o consumo de energia da rede.

B. Lista de Padrasto

A lista de padrastos melhorou o desempenho comparado com o Smart Sink e o Residual e-Scan, conforme é mostrado nas Figuras 5,6 e 7. A explicação é que o esquema montado de padrastos escolhidos pela menor distância funciona melhor que a escolha do SmartSink que é pelo de menor energia. No aspecto de gerenciamento, além da rede ser tolerantes a falha, a própria rede se organiza, sem precisar de uma entidade externa. Outra vantagem é que nenhum nó precisa ter conhecimento global da RSSF.

Não foi simulada a fusão de dados, cache ou algum algoritmo preditivo que reduz o número de mensagens, e portanto, economiza energia, o qual iria aumentar ainda mais o número de rodada nos três cenários.

VIII. CONCLUSÃO

Neste trabalho, foram apresentadas duas soluções genéricas, SmartSink e lista de padrasto, no qual podem ser incorporados em diferentes sistemas e protocolos. Suas vantagens foram exemplificadas com a aplicação de obter o mapa de energia da RSSF. SmartSink é um ponto de acesso, no qual tem suas funcionalidades estendidas. SmartSink pode corrigir a árvore de roteamento, gastando apenas uma mensagem por nó sensor. Resultados da simulação mostram que o tempo de vida aumenta em pelo menos três vezes comparado ao Residual e-scan. Lista de Padrasto usa uma lista de nós sensores, no qual contem potenciais pais, fazendo o nó sensor ser mais tolerante a falha e trabalha como o SmartSink, corrigindo a árvore de roteamento dinamicamente. Seus resultados foram melhores que o SmartSink, devido à política de escolher o

nó mais próximo. Este resultado é importante pois mostra que é possível à gerência da rede se auto organizar.

Para trabalhos futuros, estudar outras políticas para o SmartSink escolher o novo nó pai ao corrigir o roteamento e considerar também o SmartSink móvel.

REFERÊNCIAS

[1] Manish Bhardwaj, Timothy Garnett, Anantha P. Chandrakasan, "Upper Bounds on the Lifetime of Sensor Networks", Proc. ICC 2001, Junho 2001.

[2] Y. Zhao, R. Govindan, D. Estrin, "Residual Energy Scan for Monitoring Sensor Networks", IEEE Wireless Communications and Networking Conference (WCNC'02).

[3] S. Lindsey, C. Raghavendra, "PEGASIS: Power-Efficient Gathering in Sensor Information Systems", IEEE Aerospace Conference, Março 2002.

[4] W.Heinzelman, A.Chandrakasan,H. Balakrishnan, “Energy-Efficient Communication Protocols for Wireless Microsensor Networks”, Proc. Hawaaian Int'l Conf. on Systems Science, Janeiro 2000.

[5] Linnyer Beatrys Ruiz, José Marcos S. Nogueira, Antonio Alfredo Loureiro. MANNA: A Management Architecture for Wireless Sensor Networks. IEEE Communications Magazine, 41(2):116-125, February 2003.

[6] D. Ganesan, R. Govindan, S. Shenker, D. Estrin, "Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks", Mobile Computing and Communications Review, Volume 1, Number 2.

[7] Andre Motta, "X-MAC: Um Novo Protocolo de Enlace para Redes Sem Fio de Sensores", POC2-julho/2001.

[8] G. Asada, M. Doug, T. Lin, F. Newberg, G. Pottie, W. Kaiser, and H. Marcy, "Wireless integrated network sensors: Low power systems on a chip", In Proceedings of the European Solid State Circuits Conference, 1998.

[9] Barbosa, Valmir C., An Introduction to Distributed Algorithms, The MIT Press, Cambridge, Massachusetts, 1996.

[10] Deepak Ganesan, Bhaskar Krishnamachari, Alec Woo, David Culler, Deborah Estrin and Stephen Wicker, Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks, UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013.