Disseminação de Dados Adaptativa em Redes de Sensores sem Fio Auto-Organizáveis*

Eduardo Freire Nakamura[1,2] **, Carlos Maurício S. Figueiredo**[1,2] **, Antônio Alfredo F. Loureiro**[1]

[1] Departamento de Ciência da Computação – DCC
Universidade Federal de Minas Gerais – UFMG
Belo Horizonte, MG.

{nakamura, mauricio, loureiro}@dcc.ufmg.br

[2]Fundação Centro de Análise, Pesquisa e Inovação Tecnológica – FUCAPI
Manaus, AM.

{eduardo.nakamura, mauricio.figueiredo}@fucapi.br

Abstract. *This work introduces a superimposition routing algorithm for wireless sensor networks. The substrate algorithm is called STORM (Self-organizing TOpology discoveRy and Maintenance) and is responsible for the discovery and maintenance of the topology used by the superimposition algorithm, called Adaptive Diffusion, that routes sensory data towards the sink node. We analyze the performance and behavior of our solution comparing it with a routing scheme based on a tree built using the* earliest-first *parent selection scheme, the Flooding and the Directed Diffusion algorithms.*

Resumo. *Este artigo apresenta um algoritmo de roteamento para redes de sensores sem fio (RSSF) que se baseia no conceito de superimposição. Neste caso o algoritmo de substrato, chamado STORM (*Self-organizing TOpology discoveRy and Maintenance*), é responsável pela descoberta e manutenção da topologia utilizada pelo algoritmo de superimposição, chamado de* Adaptive Diffusion*, que roteia os dados sensoriados em direção ao nó sorvedouro. O desempenho e comportamento deste algoritmo são analisados através da comparação com um esquema de roteamento baseado em árvore que utiliza a estratégia* primeiro-pai*, os algoritmos de inundação (*Flooding*) e a Difusão Direcionada (*Directed Diffusion*).*

1. Introdução

Rede de sensores sem fio (RSSFs) [Estrin et al., 1999, Pottie and Kaiser, 2000, Akyildiz et al., 2002] é um tipo especial de rede *ad hoc* com diversas restrições, tais como limitações computacionais e de energia, e requisitos, como características de auto-organização e tolerância a falhas. A forma como os dados são disseminados depende

*O presente trabalho foi realizado com apoio parcial do CNPq, uma entidade do Governo Brasileiro voltada ao desenvolvimento científico e tecnológico. Processo 55.2111/2002-3.

fortemente da organização da rede e do algoritmo de roteamento, que por sua vez são diretamente afetados pelos requisitos de consumo eficiente de energia e tolerância a falhas.

Para minimizar o consumo de energia, uma solução natural é a redução do número de mensagens que trafega na rede, técnicas comuns usadas para este fim incluem fusão, agregação e compressão de dados. Embora a fusão de dados torne a rede mais robusta e confiável reduzindo a redundância, se todas as mensagens redundantes forem eliminadas então a falha de um único nó pode tornar a disseminação de dados impraticável.

Em redes planas, a comunicação é freqüentemente otimizada organizando a rede em uma estrutura de árvore onde a agregação dos dados é feita sempre que duas rotas se sobrepõem (um nó pai agrega os dados de seus filhos) [Krishanamachari et al., 2002]. A desvantagem clara da organização em árvore é que quando um nó falha toda a ramificação descendente daquele nó é eliminada até que o mesmo volte ao estado operacional ou a árvore seja reconstruída. Portanto, uma RSSF organizada em árvore tende a ser fracamente tolerante a falhas. Outro problema relacionado a esta abordagem é a dificuldade de determinar o melhor momento para reconstrução da árvore.

A Difusão Direcionada (*Directed Diffusion*) [Intanagonwiwat et al., 2000] apresenta uma solução que tenta encontrar os melhores caminhos entre nós fontes e sorvedouros. Neste algoritmo o nó sorvedouro recebe dados de múltiplos caminhos com diferentes freqüências de entrega e reforça o melhor caminho. Caso o melhor caminho falhe, então outro caminho alternativo é reforçado aumentado sua freqüência gradualmente. Esta abordagem é uma inundação (*Flooding*) controlada que herda a tolerância a falhas do algoritmo de inundação com uma redução do tráfego total. Contudo, comparado com a árvore, o tráfego gerado é bem superior na Difusão Direcionada. Uma desvantagem que será mostrada neste trabalho é que ao utilizar um protocolo MAC de *broadcast* CSMA/CA como o 802.11, quando um número de nós fontes é elevado (maior que 5) a Difusão Direcionada não consegue definir as rotas, pois os reforços são perdidos em colisões resultando em um tráfego ainda maior do que o da inundação simples (devido aos interesses e reforços propagados e perdidos). Contudo, em cenários com poucos nós fontes (5 ou menos) a Difusão Direcionada apresenta um desempenho superior ao apresentado neste trabalho [Figueiredo et al., 2004].

Este trabalho apresenta uma solução intermediária para redes planas que é mais tolerante a falhas do que a árvore e mais econômica (em relação à energia) do que a Difusão Direcionada e a inundação. Nesta solução, a rede é organizada em um grafo acíclico direcionado, de forma que múltiplos caminhos são usados para disseminar os dados sensoriados. Assim, caso um nó falhe, a rede automaticamente seleciona outro caminho sem demandar a reconstrução do grafo. Esta solução é composta de um algoritmo para descoberta e manutenção da topologia, chamado STORM, que constrói o grafo acíclico direcionado; e um algoritmo de roteamento, chamado *Adaptive Diffusion* (Difusão Adaptativa), que repassa seletivamente os pacotes de dados baseando-se em métricas definidas pela aplicação. A idéia por trás da Difusão Adaptativa consiste na escolha do melhor caminho em direção ao nó sorvedouro de acordo com as métricas de QoS desejadas, como latência, tolerância a falhas e economia de energia. Assim, em ambientes hostis, a Difusão Adaptativa pode ser configurada para alcançar os níveis desejados de tolerância a falhas; em aplicações de tempo real ela pode ser configurada para reduzir o tempo de resposta; ou em um cenário genérico, ela pode ser configurada para balancear o tráfego da rede.

O artigo está organizado da seguinte forma. A Seção 2 descreve o comportamento dos algoritmos STORM e *Adaptive Diffusion*. Na Seção 3 são apresentados os experimentos feitos para avaliar a solução proposta. Na Seção 4 são apresentados alguns trabalhos relacionados. A Seção 5 exibe as direções e trabalhos futuros. Por fim, na Seção 6 são apresentados os comentários e conclusões finais do trabalho.

2. Proposta

Aqui, auto-organização e roteamento são consideradas duas tarefas distintas e separadas mas que são fortemente relacionadas e, por isso, devem ser analisadas conjuntamente. A proposta de roteamento aqui apresentada é um algoritmo de superimposição [Bougé and Frances, 1988] que opera sobre a topologia criada e mantida pelo algoritmo de auto-organização. O modo de operação de superimposição é mostrado na Figura 1 onde o algoritmo de auto-organização é executado continuamente no plano de substrato enquanto o algoritmo de roteamento opera no plano de superimposição utilizando a infra-estrutura disponível.

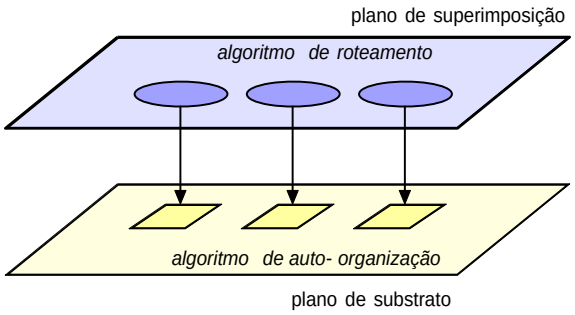


Figura 1: Interação entre auto-organização e roteamento.

2.1. STORM: Self-organizing TOpology discoveRy and Maintenance

STORM é um algoritmo distribuído para descoberta e manutenção de topologia para RSSFs. Na Figura 2, o STORM é modelado com uma máquina de estados-finitos estendida (MEFE). Nesta MEFE, as elipses representam os estados e as setas as transições. As transições estão associadas a um par evento/ação que representa a condição que dispara a transição e a ação que é executada **após** a transição. O sinal de soma (+) indica uma mensagem recebida enquanto o sinal de subtração (-) indica uma mensagem enviada pelo nó.

O nó sorvedouro se encontra permanentemente no estado READY e periodicamente difunde uma mensagem de construção de grafo: msg_build_graph com um identificador seqüencial. Quando a rede é lançada, todos os demais nós estão no estado IDLE. Ao receber uma nova mensagem msg_build_gaph o nó acrescenta o enviador na sua lista de pais parent_list, inicia seu temporizador de descoberta discovery_timer com um valor aleatório limitado (tempo de descoberta) e passa para o estado DISC.

O nó se mantém no estado DISC até que seu tempo de descoberta expire. Enquanto isso, se uma mensagem msg_build_gaph for recebida com o mesmo identificador da mensagem que fez com que o nó entrasse no estado DISC, então ele acrescenta o enviador na sua lista de pais parent_list. Ao expirar o tempo de descoberta, o nó transita para o estado READY (é neste momento que o algoritmo de roteamento torna-se operacional), difunde sua mensagem msg_build_graph e inicia seu temporizador de atualização update_timer. O

temporizador update_timer é iniciado com um valor dependente dos requisitos de QoS da aplicação.

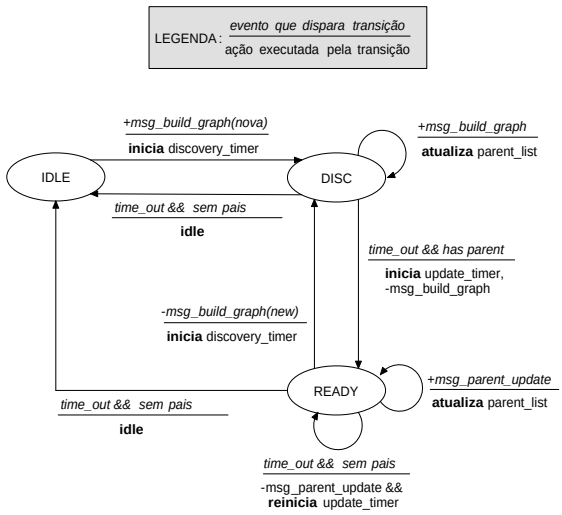


Figura 2: Algoritmo de auto-organização: STORM.

Quando o nó está no estado READY ele periodicamente envia uma mensagem do tipo msg_parent_update com seus dados atualizados (métricas utilizadas pelo roteamento). Sempre que um nó no estado READY recebe uma mensagem msg_parent_update de um vizinho em sua lista de pais parent_list, ele atualiza os dados referentes ao nó de origem. Ao expirar o temporizador update_timer, o nó remove da lista parent_list todos os nós cuja informação é mais velha do que o tempo de atualização e reinicia o temporizador update_timer. Caso todos os pais estejam fora de operação, o nó volta para o estado IDLE até que ele receba um nova mensagem do tipo msg_build_graph. Se um nó no estado READY receber um nova mensagem de reconstrução (msg_build_graph) ele volta para o estado de descoberta, DISC, reiniciando a fase de descoberta de pais.

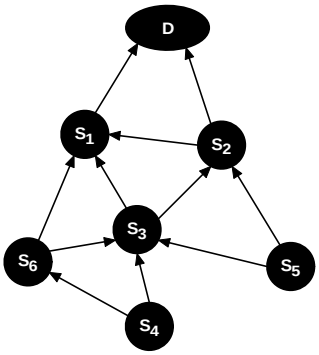
A topologia resultante deste algoritmo é um grafo acíclico direcionado onde o fluxo de dados segue dos nós fontes para o nó sorvedouro (veja Figura 3(a)). A propriedade acíclica é garantida pois um nó que se encontra no estado READY somente pode tornar-se pai de um outro nó que ainda não é pai de ninguém. Devido à aleatoriedade do temporizador discovery_timer, múltiplos caminhos são formados de cada fonte para o sorvedouro.

2.2. *Adaptive Diffusion* (Difusão Adaptativa)

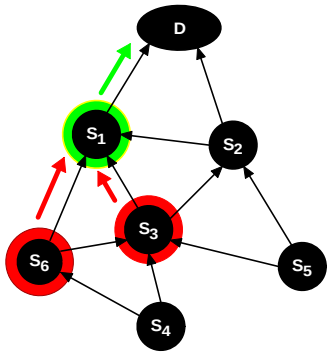
Como a topologia disponibilizada pelo STORM é um grafo acíclico direcionado, o algoritmo de roteamento pode escolher qualquer caminho sem se preocupar com a detecção de ciclos. Isto pode ser feito de uma maneira gulosa onde o nó avalia seus pais e decide para qual deles será enviado o pacote de dados. Outra alternativa é o uso de esquemas simplificados de roteamento, por exemplo:

- Escolher o pai (destino) aleatoriamente (ou seqüencialmente). Isto permite que o custo de comunicação seja homogeneizado pela rede.
- Escolher todos os pais como destinos. Esta estratégia é particularmente interessante nos casos em que o índice de perdas é elevado.
- Escolher o pai mais próximo, permite que a latência na entrega de pacotes seja reduzida.
- Escolher o pai com melhor RSSI (*Received Signal Strength Indicator*) ou melhor relação sinal/ruído. Com isso espera-se que seja reduzida a probabilidade de perdas e de que os dados sejam corrompidos.
- Escolher o pai mais próximo do nó sorvedouro para tentar reduzir o número total de pacotes trafegando pela rede.

- Escolher o pai de maior grau (mais vizinhos). Com isso a possibilidade de ganhos com a agregação é maior.



(a) Exemplo de uma possível topologia descoberta pelo STORM.



(b) Agregação ocorre quando rotas se sobrepõem.

Figura 3: Exemplo de funcionamento do STORM e do *Adaptive Diffusion*.

Como se pode observar, estes exemplos de roteamento triviais são recomendados em diferentes situações com objetivos distintos. Assim, para tornar o roteamento adaptativo e adequado a diferentes cenários (ou cenários dinâmicos) este trabalho propõe uma nova classe de algoritmos chamada *Adaptive Diffusion*, ou simplesmente Difusão Adaptativa, que compreende os benefícios das soluções acima listadas e permite ainda a mudança de comportamento perante outras métricas. Na Difusão Adaptativa um nó avalia seus pais e escolhe aquele que se espera levar ao melhor resultado.

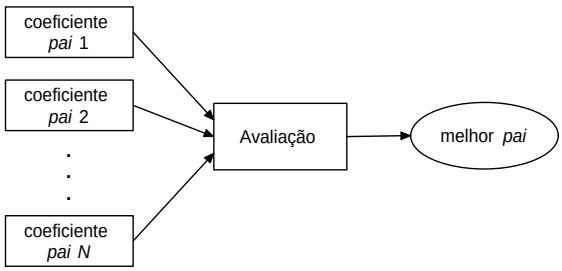


Figura 4: Funcionamento do algoritmo de roteamento.

O diagrama esquemático do algoritmo é mostrado na Figura 4. Primeiro, o nó calcula um *coeficiente de adaptação* para cada um de seus pais. A seguir, todos os coeficientes são avaliados e o pai que (supostamente) levar para o melhor caminho em direção ao sorvedouro (aquele com melhor coeficiente de adaptação) é escolhido. Os parâmetros adequados para o cálculo do coeficiente de adaptação e a função de avaliação devem ser escolhidos de acordo com os requisitos de cada aplicação. Entretanto, a título de exemplificação será apresentada uma possível implementação a seguir.

Suponha que os parâmetros relevantes para a aplicação sejam: energia residual, distância para o sorvedouro e taxa de agregação.[1] A taxa de agregação (Agg) é definida aqui como sendo a relação entre o número de mensagens enviadas e o número de mensagens recebidas pelo nó

[1] É importante observar que parâmetros podem ser modificados, trocados, removidos ou adicionados dependendo da aplicação.

$$Agg = \frac{msg_{enviadas}}{msg_{recebidas}}. \quad (1)$$

Para cada pai, o nó cria um vetor de parâmetros

$$v = \begin{pmatrix} p_{ren} & p_{dist} & p_{agg} \end{pmatrix} \quad (2)$$

onde p_{ren} é a energia residual média do caminho que aquele pai participa, p_{dist} é a distância em saltos (*hops*) daquele pai para o sorvedouro, e $p_{agg} = Agg^{-1}$ é a taxa média de agregação do caminho que o pai participa.

Além disso, para cada pai, o nó cria um vetor de peso

$$w = \begin{pmatrix} w_{ren} & w_{dist} & w_{agg} \end{pmatrix} \quad (3)$$

tal que $w_{ren} + w_{dist} + w_{agg} = 1$. Os pesos w_{ren}, w_{dist} e w_{agg} representam, respectivamente, a importância dos parâmetros energia, distância para o sorvedouro e agregação para a tarefa de disseminação dos dados.

Neste caso, o coeficiente de adaptação para cada pai pode ser computado como

$$c = vw^T \quad (4)$$

e, caso o nó possuam m pais, a função de avaliação retorna o pai com maior coeficiente de adaptação:

$$destino = \arg\max_{pai}(c_1, c_2, \ldots, c_m). \quad (5)$$

Assim, os dados são disseminados escolhendo aquele que é considerado o melhor caminho em direção ao sorvedouro. A fusão ou agregação de dados ocorrerá sempre que duas ou mais rotas se sobrepuserem (Figura 3(b)).

O melhor caminho de cada nó fonte para o sorvedouro é calculado interativamente através de uma inundação (*Flooding*) controlada onde os parâmetros são atualizados durante a formação e manutenção da topologia provida pelo STORM. Caso o sorvedouro esteja na lista parent_list, ele sempre será escolhido como a melhor opção. Quando um nó envia uma mensagem de construção msg_build_graph ou atualização msg_parent_update, o seu coeficiente é incluído de carona na mensagem. Quando um nó passa para o estado READY ele calcula seu custo (coeficiente de adaptação) que leva em consideração o coeficiente do seu melhor pai. Desta forma, quando a topologia é formada, todos os nós sabem qual pai leva para o melhor caminho. Observe que o significado de "melhor caminho" e a função de custo dependem da aplicação. Por exemplo, o custo pode ser considerado a energia para a transmissão de n bytes, a latência média, a taxa de perda de pacotes etc.

Os requisitos da aplicação podem ser representados através da definição do vetor de parâmetros e do ajuste de seus pesos. No exemplo acima, para distribuir melhor o consumo de energia entre os nós, pode-se aumentar o peso do parâmetro de energia residual.

Para diminuir a latência basta aumentar o peso do parâmetro de distância. Se o usuário desejar garantir o uso de canais mais confiáveis, pode-se acrescentar um parâmetro RSSI ao vetor de parâmetros. Os caminhos que possuem as melhores medidas RSSI (canais com sinais de melhor qualidade) podem ser priorizados configurando o peso desse parâmetro.

3. Avaliação

O STORM e a Difusão Adaptativa (STORM/AD - *STORM/Adaptive Diffusion*) foram avaliados conjuntamente através da comparação com o algoritmo de inundação (Flooding), a Difusão Direcionada (Directed Diffusion) e uma árvore (EF-Tree) construída com a estratégia *primeiro-pai* onde o pai de um nó é aquele que primeiro solicitou [Zhou and Krishnamachari, 2003]. A avaliação baseia-se na simulação dos algoritmos utilizando o simulador de eventos discretos NS-2 [NS-2, 2003]. As simulações fizeram uso dos parâmetros disponíveis para o nó sensor mica2 [Crossbow Technology, Inc, 2003]. Os demais parâmetros utilizados são mostrados na Tabela 1. Em todas as simulações, foi mantida uma densidade constante de $0,005\ nós/m^2$, independente do número de nós presentes na rede.

Parâmetro	*Valor Utilizado*
Energia de Transmissão	45mW
Energia de Recepção	24mW
Energia de Sensoriamento	15mW
Alcance do Rádio	40m
Largura de Banda	19.2Kbps
Taxa de Geração de Dados	10s

Tabela 1: Tabela de parâmetros utilizados

Para a camada MAC foi utilizado o protocolo 802.11 por possuir características semelhantes ao MAC disponível no mica2 que também é CSMA/CA. É interessante observar ao usar um protocolo CSMA/CA a quantidade de colisões deve aumentar a medida que o tráfego da rede aumenta causando a diminuição na entrega de pacotes.

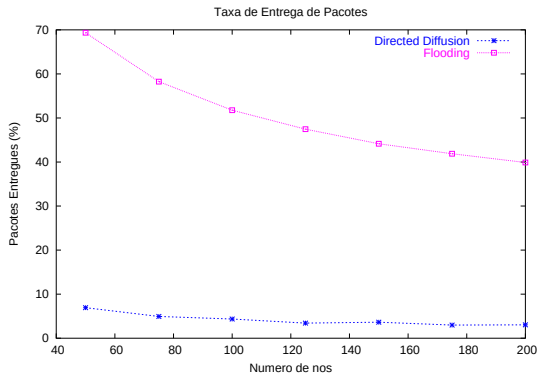
3.1. Escalabilidade

Para adaptar a rede ao dinamismo resultante de falhas, tanto o STORM/AD quanto a árvore, periodicamente reconstroem a topologia da rede com um tempo proporcional à taxa de geração de dados dos nós fonte. Esta taxa de geração de dados foi fixada em $10s$ (Tabela 1) e o período de reconstrução da topologia adotado foi $100s$.

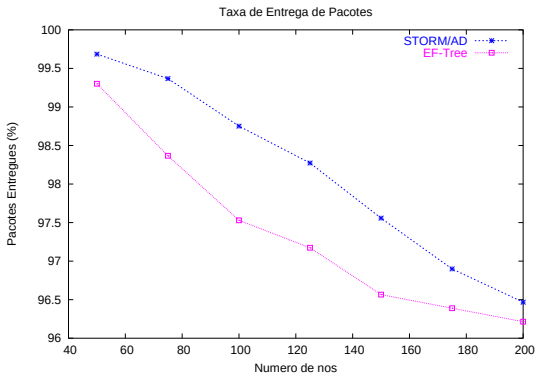
Para avaliar a escalabilidade dos algoritmos foram simulados cenários aleatórios com 50, 75, 125, 150, 175 e 200 onde cada nó possuía 10% de chance de falhar.[2] Em cada cenário destes foram escolhidos aleatoriamente 20 fontes responsáveis pela geração de dados.

Conforme mostram as Figuras 5(a) e 5(b), em relação à taxa de entrega de pacotes, o STORM/AD supera o desempenho dos demais algoritmos. A superioridade em relação à árvore ocorre porque quando um nó falha o STORM/AD pode escolher automaticamente um novo pai enquanto a árvore tem que esperar a sua reconstrução. No caso da inundação,

[2]Foram consideradas apenas falhas permanentes, ou seja, após a falha o nó jamais volta ao estado operacional.



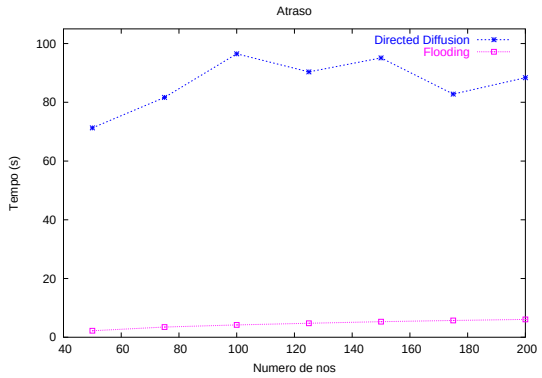
(a) Difusão Direcionada e inundação.



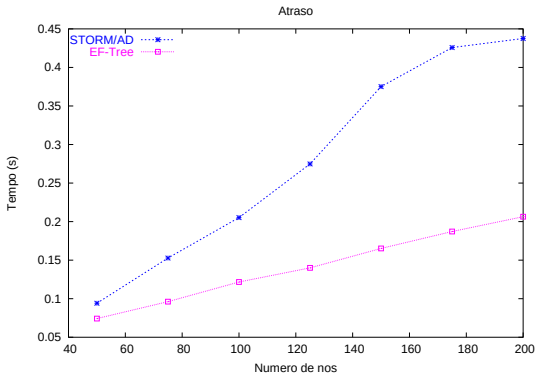
(b) STORM/AD e EF-Tree.

Figura 5: Taxa de entrega de pacotes com 20 nós fontes.

um alto tráfego é gerado (todos os nós recebem todas as mensagens pelo menos uma vez) causando um maior número de perdas devido a fatores como colisões e estouro de fila (a fila de recepção de cada nó é finita). A Difusão Direcionada apresenta o pior desempenho entregando menos de 10% dos pacotes devido ao alto tráfego gerado por reforços e pela inundação de dados e interesses.



(a) Difusão Direcionada e inundação.

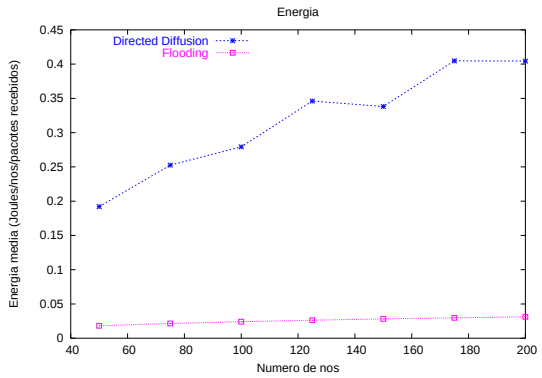


(b) STORM/AD e árvore.

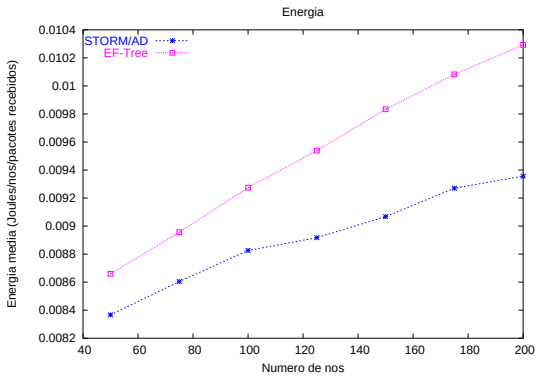
Figura 6: Atraso com 20 nós fontes.

Como se pode observar na Figura 6(a), em relação ao atraso, a inundação e a Difusão Direcionada são sacrificadas devido ao tempo que os nós gastam processando as filas de recepção. Nestas simulações o STORM/AD é superado pela árvore (Figura 6(b)). Isto ocorreu porque a árvore usa os enlaces mais rápidos e, embora, o STORM/AD tenha priorizado os caminhos mais curtos, o fator de aleatoriedade (temporizadores) do STORM resulta em enlaces de menor velocidade.

As Figuras 7(a) e 7(b) ilustram o comportamento dos algoritmos em relação à utilização da energia. Novamente, os algoritmos de Difusão Direcionada e inundação são sacrificados devido ao tráfego gerado. A árvore supera o STORM/AD porque os seus dados de controle possuem um tamanho menor (16 bytes da árvore contra 20 bytes do STORM/AD).



(a) Difusão Direcionada e inundação.

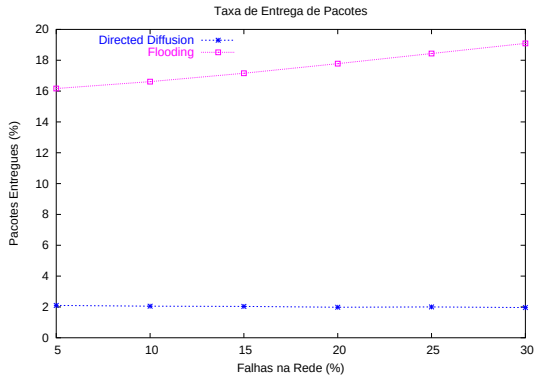


(b) STORM/AD e árvore.

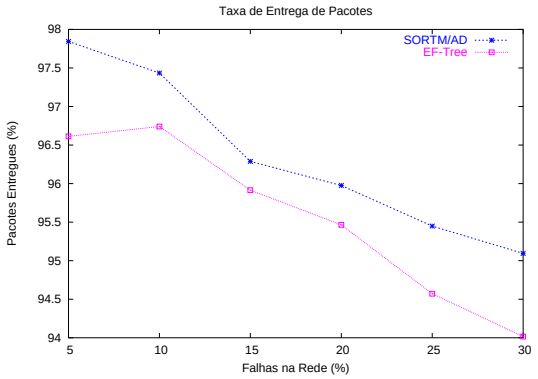
Figura 7: Consumo de energia com 20 nós fontes.

3.2. Robustez

Para explorar o comportamento dos algoritmos sob diferentes cenários de falhas, foram simuladas redes de 100 nós com um sorvedouro e 99 fontes gerando dados a cada $10s$. Com isso deseja-se avaliar o comportamento sob situações extremas de tráfego e falhas.



(a) Difusão Direcionada e inundação.

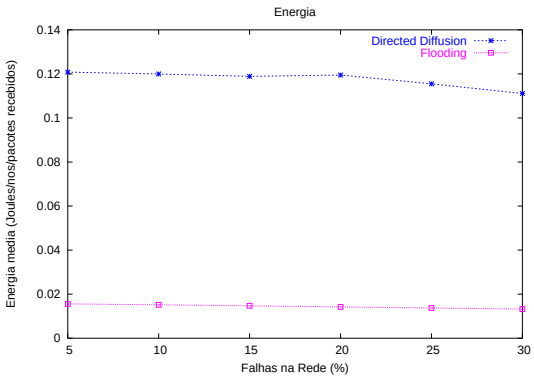


(b) STORM/AD e árvore.

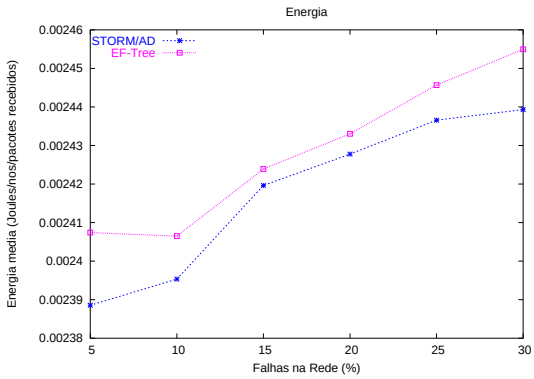
Figura 8: Taxa de entrega de pacotes para redes de 100 nós com falhas.

Nas Figuras 8(a) e 8(b) é possível observar que sob condições extremas, o STORM/AD apresenta uma taxa de entrega de pacotes superior aos outros algoritmos. Note que a diferença entre o STORM/AD e a árvore aumentam de acordo com a quantidade de falhas na rede, devido às rotas opcionais disponíveis no STORM/AD. Quando o número de falhas aumenta, o tráfego gerado pela inundação diminui aumentando ligeiramente seu desempenho. Este fato não é verificado para a Difusão Direcionada pois mesmo com as falhas o algoritmo continua operando em condições saturadas, sem conseguir definir rotas fixas entre os fontes e o sorvedouro.

Em relação à energia consumida, o STORM/AD utilizou este recurso de forma mais eficiente do que a inundação e a Difusão Direcionada (Figuras 9(a) e 9(b)). Novamente a árvore consumiu energia de forma mais eficiente devido ao tráfego de controle que possui pacotes de tamanho inferior ao STORM/AD.



(a) Difusão Direcionada e inundação.

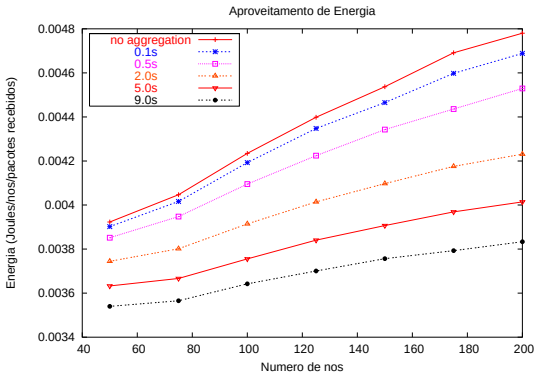


(b) STORM/AD e árvore.

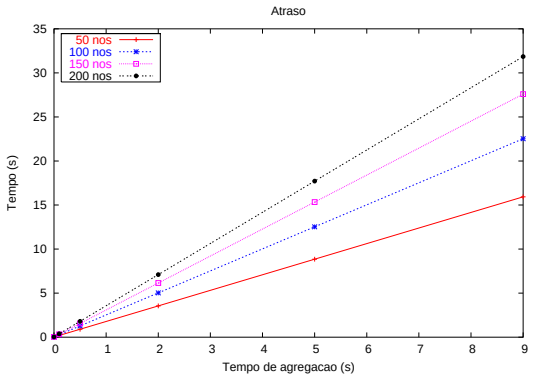
Figura 9: Energia consumida para redes de 100 nós com falhas.

3.3. Agregação

Ao introduzir agregação de dados ao STORM/AD os ganhos em relação à economia de energia são superiores. Entretanto, a agregação de dados possui algumas particularidades no caso do STORM/AD. Como os nós não sabem quem são seus filhos, foi estipulado que todo nó espera um tempo fixo e agrega todos os dados recebidos neste intervalo de tempo. Assim, quando um nó recebe um novo dado (sensoriado por ele mesmo ou recebido de outro nó) ele espera um tempo fixo (período de agregação) inferior à taxa de geração de dados e, quando este tempo expira ele agrega todos os dados recebidos durante este período e transmite o pacote agregado. Para efeito de simplificação a função de agregação utilizada foi *max* que retorna o maior valor de um conjunto de números reais.



(a) Consumo de energia.



(b) Atraso.

Figura 10: STORM/AD com agregação.

A Figura 10(a) ilustra os ganhos devido à agregação quando o período de agregação é de $0,1s$, $0,5s$, $2s$, $5s$ e $9s$. Observe que neste caso, a economia de energia torna-se mais importante quando o tamanho da rede cresce. Em relação ao atraso causado pela agregação de dados, foram feitas simulações onde cada nó gerou apenas um pacote de dados no mesmo instante de tempo e foi medido o atraso do último pacote recebido pelo nó sorvedouro. Conforme mostra a (Figura 10(b)), o atraso é maior para redes de maior escala. Isto ocorre porque os caminhos entre os fontes e o sorvedouro

aumentam com a escala da rede. Além disso, o atraso cresce linearmente com o período de agregação pois cada nó espera sempre o mesmo tempo para agregar os dados.

4. Trabalhos Relacionados

Algoritmos de auto-organização para topologia impactam diretamente no comportamento e desempenho dos algoritmos de roteamento em RSSFs. Tais algoritmos podem ser agrupados em duas classes distintas: escalonamento de nós e formação de topologia.

No escalonamento de nós, a principal preocupação é a economia de energia através do controle de densidade [Xu et al., 2001, Curt Schurgers and Srivastava, 2002, Cerpa and Estrin, 2002, Chen et al., 2002, Ye et al., 2003, Gupta et al., 2003]. Tais algoritmos gerenciam a densidade da rede determinando quando cada nó fica no estado operacional (acordado) e quando ele passa para o estado não operacional (dormência). O STORM/AD pode ser executado no topo de um algoritmo de escalonamento de nós para aumentar a economia de energia da rede.

A formação de topologia opera no topo dos algoritmos de escalonamento de nós. Estes algoritmos escolhem como cada nó (acordado) participa da infra-estrutura de rede. Dois grupos de algoritmos de formação de topologia podem ser identificados: algoritmos para topologias hierárquicas e para topologias planas. O primeiro grupo inclui os algoritmos que organizam a rede em agrupamentos onde cada nó responde somente para o líder de seu agrupamento (que pode executar operações especiais como agregação de dados). O outro grupo de algoritmos organiza a rede de forma plana, assim a comunicação é feita salto-a-salto e todos os nós possuem a mesma funcionalidade.

Para topologia hierárquica, diversos algoritmos podem ser encontrados na literatura. No LEACH [Heinzelman et al., 2000], a função de líder do agrupamento é rotativa com o objetivo de distribuir melhor o consumo de energia. O PEGASIS [Lindsey et al., 2002] é uma evolução do LEACH onde os nós sensores forma cadeias e cada nó comunica somente com um vizinho próximo e o envio de dados para o sorvedouro é feito em rodadas reduzindo a energia média gasta por cada nó. Em [Deb et al., 2002], é proposto o algoritmo TopDisc que organiza a rede em uma árvore de agrupamentos.

Enquanto nas topologias hierárquicas somente os líderes de agrupamento podem fazer agregação de dados, em redes planas qualquer nó pode agregar dados. Em [Krishanamachari et al., 2002], Krishanamachari *et al.* mostram alguns limites analíticos em relação ao custo e economia de energia que podem ser obtidos com agregação de dados usando topologias baseadas em árvores. Em [Zhou and Krishnamachari, 2003], Zhou e Krishnamachari avaliam a topologia de árvore utilizando quatro diferentes abordagens para a escolha dos pais (primeiro candidato é o escolhido, o candidato é escolhido aleatoriamente, o candidato mais próximo é escolhido, o candidato é escolhido de forma ponderada e aleatória); a análise baseia-se em métricas como grau dos nós, robustez, qualidade do canal, agregação de dados e latência. Tian e Georganas [Tian and Georganas, 2003] identificaram algumas desvantagens em esquemas de roteamento *single-path* e *multi-path* em termos de taxa de entrega de pacotes e consumo de energia.

Conforme mencionado anteriormente, o *Directed Diffusion*, proposto em [Intanagonwiwat et al., 2000], tenta encontrar os melhores caminhos entre fontes e o sorvedouro que recebe os dados de múltiplos caminhos com freqüências distintas. Caso o melhor caminho falhe, um outro caminho alternativo de menor freqüência garante a entrega dos pacotes. Em [Ganesan et al., 2001] o roteamento tenta descobrir e manter caminhos alternativos (ligando os fontes ao sorvedouro) para tornar a rede mais tolerante a falhas.

Em relação a tais esquemas de roteamento (baseados em árvores e baseados em difusão), o STORM/AD tenta oferecer uma solução intermediária que possa equalizar tolerância a falhas, economia de energia e latência de acordo com os requisitos da aplicação. De forma semelhante às soluções encontradas em [Intanagonwiwat et al., 2000, Gupta et al., 2003, Ganesan et al., 2001], o STORM constrói e mantém de forma eficiente uma infra-estrutura de múltiplos caminhos para obter maior tolerância a falhas. Por outro lado, a Difusão Adaptativa deve ser utilizada para explorar tal infra-estrutura buscando o balanceamento de carga, a economia de energia, o aumento da taxa de entrega de pacotes e a redução do atraso.

5. Trabalhos Futuros

Atualmente estão sendo incluídos na Difusão Adaptativa outros parâmetros para o cálculo do coeficiente de adaptação, como taxa de perdas e atraso (buscando encontrar os caminhos de menor atraso e de menor taxa de perdas). No próximo estágio, será avaliado o impacto da mobilidade de nós que aumenta o dinamismo da topologia da rede. Outros trabalhos futuros incluem a avaliação do STORM/AD quando executado no topo de um algoritmo de escalonamento de nó, e a exploração de outras estratégias para a construção do grafo direcionado acíclico (diferentes do STORM) usado pela Difusão Adaptativa.

6. Conclusões

Este trabalho apresentou um algoritmo distribuído para formação e manutenção da topologia de rede chamado STORM. Este algoritmo constrói um grafo acíclico direcionado que oferece caminhos alternativos para o algoritmo de roteamento. Como resultado, comparado a uma topologia baseada em árvore, esta solução permite alcançar maiores taxas de entrega de pacotes.

A Difusão Adaptativa é flexível o suficiente para se adequar às condições consideradas relevantes para a aplicação. A função de avaliação, os parâmetros e os pesos usados para o cálculo do coeficiente de adaptação podem ser selecionados baseados nos requisitos da aplicação para alcançar a qualidade de serviço (QoS) desejada. Além disso, os parâmetros, pesos e a função de avaliação podem ser modificados de forma dinâmica tornando a RSSF mais adequada à situação corrente.

Embora esta avaliação preliminar não tenha explorado todo o potencial da Difusão Adaptativa, foi possível observar que ao priorizar a distância, o atraso na entrega dos pacotes foi reduzido. Contudo, outras formas de construção do grafo possam obter melhores resultados.

Também foi possível observar que, ao utilizar um protocolo MAC de disputa ao meio como o 802.11, sob situações de alto tráfego (todos os nós da rede enviando pacotes a cada $10s$), o número de colisões é tão alto que o melhor caminho escolhido pela Difusão Adaptativa é freqüentemente trocado por um caminho alternativo. Conseqüentemente, os pesos usados para o cálculo do coeficiente de adaptação tornam-se irrelevantes pois são as falhas (colisões) que de fato determinam a rota escolhida. Este comportamento deve ser amortizado quando utilizado um protocolo MAC baseado em TDMA.

Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cyirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422.

Bougé, L. and Frances, N. (1988). A compositional approach to superimposition. In *Proceedings of the 15th Annual ACM SIGACT/SIGPLAN Symposium On Principals of Programming Languages*, San Diego, CA, USA.

Cerpa, A. and Estrin, D. (2002). Ascent: Adaptive self-configuring sensor networks topologies. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, volume 3, pages 1278–1287, New York, NY, USA.

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. (2002). Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM Wireless Networks*, 8(5):481–494.

Crossbow Technology, Inc (2003). *MPR - Mote Processor Radio Board MIB - Mote Interface / Programming Board User's Manual*. San Jose, CA, USA. Document 7430-0021-05.

Curt Schurgers, Vlasios Tsiatsis, S. G. and Srivastava, M. (2002). Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing*, 1(1):70–80.

Deb, B., Bhatangar, S., and Nath, B. (2002). A topology discovery algorithm for sensor networks with applications to network management. In *IEEE CAS Workshop on Wireless Communications and Networking*, Pasadena, CA, USA. (short paper).

Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. (1999). Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the 5th Annual International Conference on Mobile Computing and Networks (MobiCom'99)*, pages 263–270, Seattle, Washington, USA. ACM Press.

Figueiredo, C. M., Nakamura, E., and Loureiro, A. A. F. (2004). Protocolo adaptativo híbrido para disseminação de dados em redes de sensores sem fio auto-organizáveis. In *22o. Simpósio Brasileiro de Redes de Computadores (SBRC 2004)*, Gramado, RS, Brazil.

Ganesan, D., Govindan, R., Shenker, S., and Estrin, D. (2001). Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11–25.

Gupta, H., Das, S. R., and Gu, Q. (2003). Connected sensor cover: Self-organization of sensor networks for efficient query execution. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pages 189–200, Annapolis, Maryland, USA. ACM Press.

Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. (2000). Energy-efficient communication protocols for wireless microsensor networks. In *Proceedings of the Hawaiian International Conference on Systems Science (HICSS)*, Maui, Hawaii, USA.

Intanagonwiwat, C., Govindan, R., and Estrin, D. (2000). Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom'00)*, pages 56–67, Boston, MA, USA. ACM Press.

Krishanamachari, B., Estrin, D., and Wicker, S. (2002). The impact of data aggregation in wireless sensor networks. In *Proceedings of the International Workshop of Distributed Event Based Systems (DEBS)*, Vienna, Austria.

Lindsey, S., Raghavendra, C., and Sivalingam, K. M. (2002). Data gathering algorithms in sensor networks using energy metrics. *IEEE Transactions on Parallel and Distributed Systems*, 13(9):924–935.

NS-2 (2003). The network simulator - ns-2. [Online] Available: http://www.isi.edu/nsnam/ns/.

Pottie, G. J. and Kaiser, W. J. (2000). Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58.

Tian, D. and Georganas, N. D. (2003). Energy efficient routing with guaranteed delivery in wireless sensor networks. In *IEEE Wireless Communications and Networking Conference (WCNC 2003)*, volume 3, pages 1923–1929, New Orleans, Louisiana, USA.

Xu, Y., Heidemann, J., and Estrin, D. (2001). Geography-informed energy conservation for ad-hoc routing. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 16–21, Rome, Italy.

Ye, F., Zhong, G., Cheng, J., Lu, S., and Zhang, L. (2003). Peas: A robust energy conserving protocol for long-lived sensor networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 28–37, Providence, Rhode Island, USA.

Zhou, C. and Krishnamachari, B. (2003). Localized topology generation mechanisms for self-configuring sensor networks. In *Proceedins of the IEEE Globecom 2003 - Wireless Communication Symposium*, San Francisco, USA. Accepted.