# Minimizando Hot Spots no Roteamento em Redes de Sensores Sem Fio

**Fernando Henrique Gielow**[1]*, **Aldri L. dos Santos**[1]

[1]NR2 – Departamento de Informática – Universidade Federal do Paraná
Caixa Postal 19.081 – 81.531-980 – Curitiba – PR – Brasil

{fhg07,aldri}@inf.ufpr.br

***Resumo.*** *Nas redes de sensores sem fio, diversas técnicas têm sido empregadas para garantir a entrega dos dados e diminuir o custo de comunicação. Na abordagem que utiliza agrupamentos (clusters) de tamanhos iguais, líderes próximos da base constantemente fazem parte de rotas e morrem prematuramente. Tais áreas sobrecarregadas com tráfego intenso de dados são conhecidas como hot spots, e medidas precisam ser tomadas para minimizar seus impactos. Esse trabalho propõe um protocolo de roteamento, denominado RRUCR, que mitiga as regiões de hot spot por meio de clusters desiguais, criados a partir de diferentes potências de transmissão. A manutenção dinâmica do backbone repara os enlaces quebrados nas rotações de líderes, e economiza energia por não utilizar pacotes de controle. Simulações mostram que os efeitos do hot spot foram minimizados devido à melhor distribuição do tráfego de dados e de consumo de energia. Quando comparado ao protocolo UCR, que também emprega clusters desiguais, nosso protocolo apresenta um número menor de clusters e rotações, o que resultou em um aumento de 21.36% no tempo de vida da rede. Também, foram alcançadas maiores taxas de entrega de dados devido à manutenção do backbone.*

## 1. Introdução

As Redes de Sensores Sem Fio (RSSF) são utilizadas em diversas aplicações, como no monitoramento ambiental ou em sistemas de segurança [Bulusu and Jha 2005]. Para obter uma comunicação eficiente entre os sensores (nós), o estabelecimento de rotas até uma estação-base é necessário, especialmente em rotas com fluxos de dados contínuos. Assim, a perda de dados no roteamento precisa ser reduzida, evitando trabalho desnecessário. O consumo de energia é essencialmente diferente para cada nó em uma RSSF, visto que nós distribuídos de maneira homogênea resultam em uma espécie de efeito funil nas rotas quando a abordagem multi-salto é utilizada. Na medida que sensores se aproximam da base, o número de rotas diminui, resultando em um processo gradual que cria e expande um buraco de energia centrado na base [Liu et al. 2008]. *Hot spots* compreendem estas áreas sobrecarregadas por tráfego de dados.

Os protocolos de roteamento baseados em agrupamentos (*clusters*) para RSSF buscam reduzir o custo de comunicação e prover a entrega dos dados, tais como LEACH [Heinzelman et al. 2000], EECS [Ye et al. 2005], HEED [Younis and Fahmy 2004], TPC [Choi et al. 2004], VCA [Qin and Zimmermann 2007] e UCR [Chen et al. 2009]. Dentre estes, apenas o UCR se preocupa com os *hot spots*. Como eles não podem ser completamente eliminados, o UCR define *clusters* de tamanhos proporcionais às suas distâncias

*Número BANPESQ/THALES da bolsa de IC: 2007021683 - Fundação Araucária

até a estação-base (base) para aumentar o tempo de vida da rede e balancear a energia. Assim, mais *clusters* são criados próximos da base, oferecendo mais rotas até ela.

Entretanto, o uso de *clusters* desiguais apenas ameniza os efeitos do *hot spot*. Mesmo com mais rotas criadas, os *clusters* próximos à base têm menos membros e, assim, menos rotações de líderes podem ser feitas. Além disso, essas rotações podem provocar quebras de enlaces quando líderes mais distantes são escolhidos. Desta maneira, nós que se comunicavam com o antigo líder não podem transmitir seus dados ao novo líder.

Este trabalho apresenta um protocolo de roteamento para RSSF baseado em *clusters* desiguais que repara o *backbone* de maneira dinâmica. Em contraste à outras abordagens que empregam medições de RSSI (*Receive Signal Strength Indicator*), ou a distância exata entre nós para criar *clusters* desiguais, nosso protocolo, denominado *Rotation Reactive Unequal Cluster-based Routing protocol* (RRUCR), utiliza diferentes potências de transmissão. Medições de RSSI [Parameswaran et al. 2009] são imprecisas, e determinar a distância exata entre nós muitas vezes não é possível[1].

Além disso, um algoritmo de manutenção que não usa pacotes de controle foi utilizado para garantir taxas de entrega satisfatórias, visto que enlaces podem ser quebrados em virtude das rotações de líderes. Toda informação necessária é carregada em *piggybacking* pelas mensagens responsáveis pela coleta de dados, economizando energia. Simulações no NS-2.30 mostram que o RRUCR minimiza os efeitos dos *hot spots*, visto que o tempo de vida da rede aumentou e menos nós morreram perto da base. Além disso, a manutenção dinâmica dos enlaces resultou em uma maior taxa de entrega de dados.

O artigo está organizado da seguinte forma: a Seção 2 detalha o funcionamento do RRUCR. A Seção 3 mostra a avaliação do desempenho do protocolo proposto. Finalmente, a Seção 4 apresenta as conclusões.

## 2. RRUCR

Nosso protocolo, chamado *Rotation Reactive Unequal Cluster-based Routing protocol* (RRUCR), busca minimizar o *hot spot* centrado na base através de *clusters* desiguais, como mostrado na Figura 1. Ele emprega também rotações de líderes, e integra a manutenção do *backbone* em fluxos de dados. O RRUCR consiste de cinco operações: definição do raio de atuação (escopo) dos nós, criação dos *clusters*, criação do *backbone* inicial, rotação de líderes e coleta de dados, que suporta a manutenção do *backbone*. As primeiras duas operações ocorrem apenas na criação da rede. Dessa forma, o número de *clusters* apenas diminui ao passar do tempo, com a morte dos nós. Um *backbone* inicial é estabelecido após a criação dos *clusters*, e as demais operações ocorrem várias vezes, provendo balanceamento de energia e maiores taxas de entrega de dados.

As próximas subseções descrevem as operações do RRUCR. Note que as potências de transmissão são ordenadas crescentemente em uma tabela guardada por todos os nós, e apenas os índices das potências são enviados nas mensagens. Uma melhor descrição dos algoritmos pode ser encontrada em [Gielow and dos Santos 2009b]. Note também que, embora o UCR possua fases de propósitos similares, os algoritmos são diferentes. Para o cálculo do raio de atuação o UCR utiliza medidas de distância física, seu algoritmo de agrupamento é mais simples e, além disso, ele não possui uma fase de reparo de rotas.

[1]O uso de GPS poderia solucionar este problema, mas por um preço caro

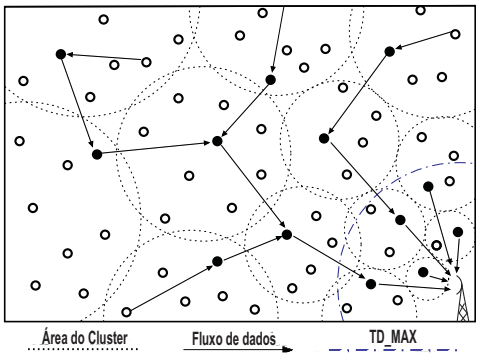


**Figura 1. Organização da rede em clusters de tamanhos desiguais**

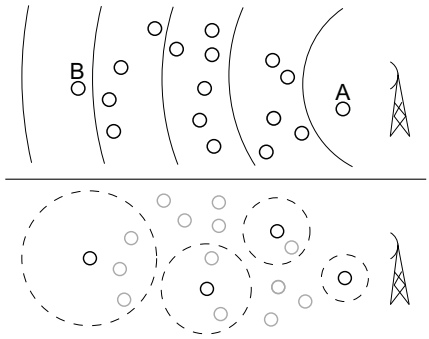


**Figura 2. Operação de definição de escopos de transmissão**

### 2.1. Definição dos escopos

A base inicialmente envia (por simplicidade, sempre que não for especificado um destino para a mensagem enviada, assumiremos que ela foi enviada em *broadcast*) mensagens *INCR_POT* cobrindo todas as potências de transmissão utilizadas (Figura 2-superior). Os nós ao receberem esta mensagem pela primeira vez, salvam a potência utilizada pela base em uma variável $RBase$, e respondem com uma mensagem de confirmação. Ao final, a base saberá quais foram a menor ($RFMin$, empregada para alcançar um nó A) e a maior ($RFMax$, empregada para atingir um nó B) potências de transmissão usadas para atingir um nó. Em seguida, a base envia uma mensagem *SETUP_CONFIG* na potência de transmissão máxima, informando os valores de $RFMin$ e $RFMax$.

Dado que alguns nós podem não ser alcançados por essa mensagem, os nós mais distantes (cujo $RBase = RFMax$) retransmitirão a mensagem *SETUP_CONFIG*. Os nós restantes, que não haviam sido alcançados, também retransmitirão a mensagem até que a rede inteira seja coberta. Essa mensagem contém um campo $cont$ que informa quantos saltos a mensagem percorreu até atingir o nó atual.

A operação de formação de *clusters* pré-estipula dois limites máximos para as potências de transmissão: $pot\_limit$, o índice da potência máxima que pode ser utilizada por nós atingidos na primeira onda de mensagens *SETUP_CONFIG*, e $pot\_max\_global$, o índice da potência máxima para os demais nós. Esses limites existem pois com as rotações de líderes, é possível que a distância entre os líderes de dois *clusters* aumente e a potência de transmissão *inter-clusters* não seja o bastante para eles se comunicarem. Com esses limites, os *clusters* terão diâmetro inferior ao alcance da potência de transmissão, tornando mais difícil a quebra de enlaces.

Nós atingidos pela primeira onda de mensagens *SETUP_CONFIG* usarão a potência do seguinte índice durante a operação de criação de *clusters* (o raio de transmissão de alguns nós é ilustrado na Figura 2-inferior):

$$Scope \ = \ \lfloor \ ((1 \ - \ \frac{(RFMax - RBase)}{(RFMax - RFMin)}) * pot\_limit) \ \rfloor \qquad (1)$$

Para os outros nós, $Scope \ = \ (pot\_limit + cont)$, tendo como limite máximo $pot\_max\_global$.

### 2.2. Criação de clusters

Após a definição do escopo de cada nó, a operação de criação de *clusters* começa. Baseando-se em uma probabilidade pré-definida $pBeTHead$, nós são sorteados como candidatos a líder. Então esses nós enviam uma mensagem informando suas energias para todos os nós em seu escopo (definido na Equação 1). Nós que não receberem uma mensagem de candidatos a líder também se candidatarão - essa medida evita áreas sem cobertura na rede. Após os nós candidatos receberem essas mensagens, eles verificam se sua energia é maior do que a de seus vizinhos, caso seja, se consolidam como líderes, enviando uma mensagem *FINAL_HEAD*. Todos os nós contam essas mensagens e salvam a informação em uma variável $finals$.

Após o tempo alocado à esta fase, os nós que não receberam nenhuma destas mensagens também se candidatarão, enviando uma mensagem *FINAL_HEAD*. Os nós com $finals > 2$ desistirão da eleição. Essa medida aumenta a cobertura da rede, sem criar um número excessivo de *clusters*. Os nós remanescentes são definidos como líderes e enviam mensagens de anúncio. Assim, os nós comuns poderão escolher um líder com base no RSSI da mensagem de anúncio. Cada nó guarda o identificador (ID) do líder selecionado, e envia uma mensagem *JOIN_CLUSTER*, informando sua energia. Ao receber estas mensagens, o líder guarda a maior energia para futuras operações de rotação.

### 2.3. Criação do backbone inicial

Esta operação estabelece um *backbone* inicial válido, que permite aos nós alcançar a base em poucos saltos. A base inicia o processo enviando uma mensagem *BEACON_ROUTE* para todos os nós na área *TD_MAX* (Figura 1), definida por uma potência de transmissão. Os líderes nesta área se comunicarão diretamente com a base, e esta é normalmente uma área de surgimento de *hot spots*.

A mensagem *BEACON_ROUTE* carrega um campo $counter$ que informa quantos saltos ela percorreu, possibilitando aos nós saber o quão longe estão da base. Cada nó guarda tal informação em uma variável $wave$. Desta forma, ao receber uma mensagem *BEACON_ROUTE*, a $wave$ do nó é atualizada se $counter$ for menor do que seu atual valor, e o campo $next\_hop$ passa a guardar o ID do nó que enviou esta mensagem. $next\_hop$ é também atualizado se $counter = wave$ e o RSSI da mensagem é maior do que o RSSI da mensagem que causou a última atualização em $next\_hop$. A mensagem *BEACON_ROUTE* é então retransmitida, incrementando o campo $counter$.

### 2.4. Rotações de líderes

Um limite percentual de energia, chamado $pRotate$, é pré-definido e utilizado na rotação dos líderes. Quando a energia do líder se torna inferior do que a porcentagem $pRotate$ da maior energia do *cluster* (originalmente obtida por meio das mensagens *JOIN_CLUSTER*), este líder envia uma mensagem solicitando uma rotação, informando também sua energia. Os nós responderão se eles pertencerem ao *cluster* deste líder, e as suas energias forem superiores à informada. O líder que solicitou a rotação selecionará como novo líder o nó que informou maior energia, adotando seu ID como $next\_hop$ (Figuras 3-I e 3-II)[2]. Então, o líder enviará uma mensagem *DENOMINATE_CH* contendo seu $next\_hop$ anterior, e o ID do novo líder. Ao receber esta mensagem, o novo líder atualiza seu $next\_hop$ para

[2]Esta figura representa apenas uma parte da rede

o informado na mensagem recebida e considera a maior energia do *cluster* como sendo a sua própria. Os nós restantes passarão a se comunicar com o novo líder.

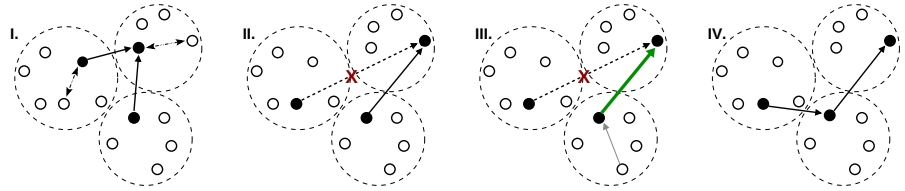


**Figura 3. Reparando um enlace quebrado em uma rotação de líderes**

Como enlaces quebrados podem ser gerados devido às rotações (linha pontilhada na Figura 3-II, marcada com um 'X'), ao receber uma mensagem *DENOMINATE_CH*, tanto o novo líder quanto os líderes que se comunicavam com o antigo líder serão forçados a atualizar suas rotas assim que possível.

### 2.5. Coleta de dados & Manutenção das rotas

Para enviar os dados coletados à base, os nós enviam uma mensagem *DATA_GATHERED* com os dados coletados e o valor de $next\_hop$ (linha cinza e fina da Figura 3-III). O líder cujo ID for igual ao $next\_hop$ informado na mensagem atualizará o $next\_hop$ da mensagem para o seu próprio e a reencaminhará (Linha grossa da Figura 3-III). Nesta mensagem, o líder também envia um valor $wr = 100 * wave + RBase$ que identifica sua distância até à base, e ajudará nas atualizações de rota.

Quando um líder receber uma mensagem *DATA_GATHERED*, ele poderá também atualizar sua rota. Quando a manutenção for obrigatória em virtude da rotação, o líder apenas verificará se o líder que enviou a mensagem está entre ele e a base (seu $wr$ é menor), e caso esteja, seu ID será imediatamente adotado como $next\_hop$. Se a manutenção não for obrigatória, a rota será atualizada apenas se a nova possibilidade de rota tiver um custo ($wr$) inferior do que o da atual. Embora a manutenção das rotas não garanta inicialmente o menor caminho, ela conserva energia e diminui o *overhead*, devido à ausência de pacotes de controle neste processo dinâmico.

## 3. Avaliação de desempenho

Embora existam vários protocolos de roteamento baseados em *clusters* para RSSF , o nosso foi comparado apenas com o UCR [Chen et al. 2009], pois ambos empregam *clusters* de tamanhos desiguais para mitigar o *hot spot* centrado na base, com algoritmos distintos. A mobilidade dos nós foi desconsiderada, pois além de ser uma técnica alternativa para combater o *hot spot*, é pouco viável em cenários reais [Vlajic and Stevanovic 2009]. Os protocolos foram implementados no simulador NS-2.30, e operam com o IEEE 802.11b na camada de enlace. Uma RSSF homogênea foi estabelecida e o rádio foi configurado de acordo com o CC1000, utilizado por nós Mica2. As leituras de dados são geradas nos nós com probabilidade de 0.1% por segundo, simulando medições de temperatura, luminosidade, e umidade em uma floresta, por exemplo. Eles são transmitidos sem nenhuma técnica de agregação ou compressão de dados.

A RSSF opera por 5000 segundos, e consiste de 700 sensores distribuídos em uma área quadrada, de 1000m em cada lado. A localização de todos os nós é aleatória em cada

simulação. A energia inicial de cada nó compreende valores entre 0.9 e 1.1 joules. Para ambos os protocolos, a probabilidade de um nó se candidatar a líder é de 35%. Os dados coletados têm 32 bytes, e a potência de transmissão *inter-cluster* é de 3.16227mW[3]. A área $TD\_MAX$ tem 149m centrados na base, onde os nós se comunicam diretamente com ela. No UCR, o limite máximo do raio dos *clusters* é de 140m. No RRUCR foram utilizados $pot\_limit = 0.25118mW$, $pot\_max\_global = 0.63095mW$[4] e $pRotate = 65\%$.

Foram realizados três tipos de simulação para cada protocolo: operação da rede sem falhas, com falhas perto da base, e com falhas longe da base (distância quantificada em saltos). No caso de falhas perto da base, são desativados aleatoriamente 8 sensores que levem de 0 a 2 saltos para se comunicar, e mais 8 sensores que levem de 1 a 5 saltos. No caso de falhas longe da base são desativados 12 nós que levem de 2 a 5 saltos para se comunicar, e mais 13 que levem de 3 a 6 saltos. As falhas ocorrem aos 400s da simulação.

As métricas consideradas são *número de saltos de cada líder até a base*, *total de energia da rede*, *tempo de vida da rede*, *número de nós mortos em relação à sua distância até à base*, *taxa de entrega de dados* (que considera a porcentagem de entrega dos últimos 30 pacotes de dados enviados), e o *número de rotações*. Essas métricas medem a eficiência das rotas criadas e suas manutenções, a mitigação do *hot spot*, e o balanceamento de energia da rede. Todos os resultados foram obtidos de 35 simulações realizadas para cada protocolo e cada tipo de rede descrita, obtendo-se um intervalo de confiança de 95%.

O código de uma implementação do RRUCR para NS-2.30, segundo a licença GLPL, é encontrado em `www.nr2.ufpr.br/~fernando/rrucr/rrucr_codes`.

### 3.1. Distribuição dos clusters & Consumo de energia

O número de *clusters* é um fator importante em RSSF. Muitos *clusters* implicam em maior consumo de energia devido à troca de mensagens, e uma quantidade pequena implica em em maior *overhead* e consumo de energia devido à maior potência de transmissão usada. Como o UCR não repara suas rotas, seus *clusters* precisam ser menores, para que as rotações realizadas tenham menos probabilidade de selecionar líderes muito distantes.

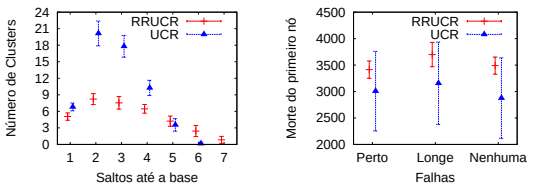


**Figura 4. Número de clusters vs. Distância & Tempo de vida**

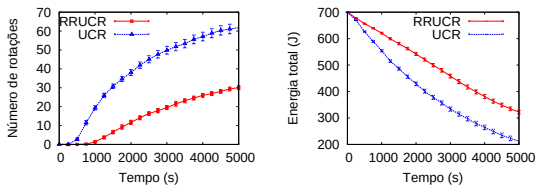


**Figura 5. Rotações & Energia ao decorrer do Tempo**

Na Figura 4-esquerda, pode ser visto que embora o RRUCR tenha *clusters* que precisam de mais saltos para atingir a base, ele cria menos *clusters*. Em média, o RRUCR cria 43 *clusters*, enquanto o UCR cria 67. Essa diferença de 35.82% gera um maior consumo de energia no UCR, como visto na Figura 5-direita. Isso acontece porque com mais *clusters*, mais mensagens serão trocadas devido ao maior número de rotações.

[3]A maior potência suportada por nós Mica2

[4]Valores obtidos do *datasheet* do rádio CC1000 e indexados, como anteriormente descrito

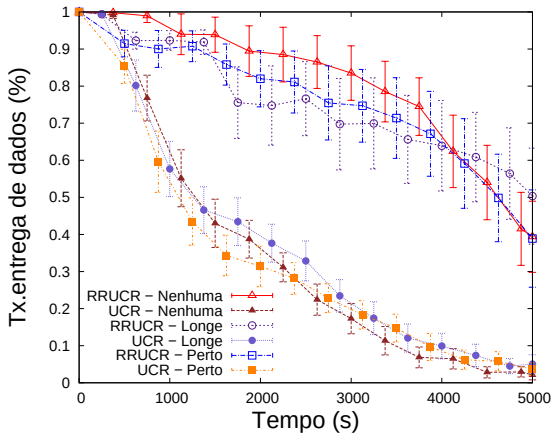


**Figura 6. Tx. de entrega dos dados com falhas**

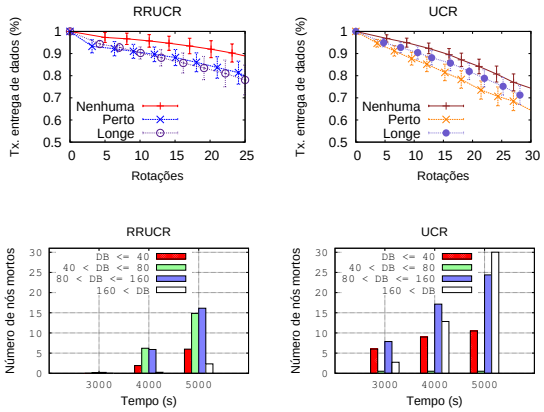


**Figura 7. Impacto das rotações & Nós mortos vs. Tempo**

### 3.2. Tempo de vida da rede

O tempo de vida de uma RSSF é o tempo decorrido até a morte do primeiro nó. Uma maneira de estender este tempo em RSSF organizadas em *clusters* é utilizar a rotação de líderes, para distribuir o consumo de energia interno. Comparado ao UCR, o RRUCR conseguiu estender o tempo de vida da rede em 21.36% em redes sem falhas. Em redes com falhas longe da base houve um aumento de 17.16%, e em redes com falhas perto da base, 13.55% (Figura 4-direita). Assim, a criação de mais rotas e suas manutenções equilibram o consumo de energia da rede como um todo. Em ambos os protocolos o melhor tempo de vida foi obtido em redes com falhas longe da base, pois são gerados menos dados que precisam trafegar longas distâncias.

Para avaliar o desempenho dos protocolos na mitigação do *hot spot*, o número de nós mortos ao longo da distância em metros até à base (DB) foi medido (Figura 7-inferior). Ambos os protocolos minimizaram os efeitos do *hot spot*, diminuindo o número de mortes perto da base. Desta forma, mais nós podem ser utilizados como último salto. Entretanto, uma quantidade maior de nós morre no UCR, devido ao maior número de *clusters* e as conseqüentes rotações, que consomem mais energia.

### 3.3. Taxa de entrega de dados

Apenas a distribuição de energia e a existência de uma rota inicial não garantem taxas de entrega satisfatórias. A Figura 6 mostra que o RRUCR apresentou uma taxa de entrega superior ao UCR, que se equiparou com o RRUCR apenas no começo das simulações, antes de qualquer falha ou rotação. Além disso, devido a maior quantidade de *clusters* criados (Figura 4-esquerda), o UCR precisa realizar mais rotações (Figura 5-esquerda), e elas geram enlaces quebrados, prejudicando sua taxa de entrega (Figura 7-superior).

Diferente do UCR, que não realiza a manutenção de rotas, as taxas de entrega do RRUCR permanecem mais estáveis. Na medida que os enlaces são quebrados, a coleta de dados dinamicamente repara os enlaces. No UCR, quando dois líderes que podiam se comunicar rotacionam e saem do alcance um do outro, o enlace entre estes *clusters* é definitivamente perdido. Isto acontece pois mesmo que novas rotações ocorram em ambos os *clusters* e seus novos líderes entrem no escopo um do outro, eles não saberão que devem se comunicar.

## 4. Conclusão

Estre trabalho apresentou um protocolo de roteamento baseado em *clusters* desiguais, chamado RRUCR, que mitiga os efeitos dos *hot spots* em RSSF. O RRUCR faz manutenção dinâmica em suas rotas, sem o uso de pacotes de controle, economizando energia. O protocolo tem cinco operações, das quais, a criação balanceada de clusters desiguais emprega diferentes potências de transmissão, e as operações de rotação de líderes e coleta de dados oferecem um melhor balanceamento de energia e a manutenção das rotas.

Os resultados das simulações mostraram que o RRUCR aumentou o tempo de vida da rede em cerca de 21.36% em relação ao UCR. O número de *clusters* criados foi 35.82% inferior ao UCR, gastando menos energia em rotações. A avaliação sob a tolerância à falhas mostrou a eficiência do RRUCR em relação ao UCR. Trabalhos futuros incluem operações que chequem a integridade dos enlaces, realizando reparos mais complexos. O desenvolvimento desse trabalho possibilitou as seguintes publicações: [Gielow and dos Santos 2009a, Gielow and dos Santos 2009b]. Atualmente, uma versão do RRUCR está sendo implementada no TinyOS 2.x.

## Referências


Bulusu, N. and Jha, S. (2005). *Wireless sensor networks - A system perspective*. Artech House.

Chen, G., Li, C., Ye, M., and Wu, J. (2009). An unequal cluster-based routing protocol in wireless sensor networks. *Wirel. Netw.*, 15(2):193–207.

Choi, W., Shah, P., and Das, S. K. (2004). A framework for energy-saving data gathering using two-phase clustering in wireless sensor networks. *Mobile and Ubiquitous Systems, Annual International Conference on*, 0:203–212.

Gielow, F. H. and dos Santos, A. L. (2009a). Protocolo de roteamento reativo que ameniza o hot spot. Technical report, 17o Evinci, Brasil. Ver http://www.nr2.ufpr.br/˜ fernando/rrucr/rrucr_pub.php.

Gielow, F. H. and dos Santos, A. L. (2009b). Um protocolo de roteamento baseado em clusters desiguais para minimizar hot spots em rssf. In *SBRC 2009 - WGRS*.

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

Liu, A.-F., Ma, M., Chen, Z.-G., and hua Gui, W. (2008). Energy-hole avoidance routing algorithm for wsn. *International Conference on Natural Computation*, 1:76–80.

Parameswaran, A., Husain, M. I., and Upadhyaya, S. (2009). Is rssi a reliable parameter in sensor localization algorithms: An experimental study. In *Field Failure Data Analysis Workshop (F2DA'09)*.

Qin, M. and Zimmermann, R. (2007). Vca: An energy-efficient voting-based clustering algorithm for sensor networks. *Journal of Universal Computer Science*, 13(1):87–109.

Vlajic, N. and Stevanovic, D. (2009). Sink mobility in wireless sensor networks: a (mis)match between theory and practice. In *IWCMC*, pages 386–393.

Ye, M., Li, C., Chen, G., Wu, J., and Al, M. Y. E. (2005). Eecs: An energy efficient clustering scheme in wireless sensor networks. In *In: Proc. of the IEEE Int'l Performance Computing and Communications Conf*, pages 535–540. IEEE Press.

Younis, O. and Fahmy, S. (2004). Heed: A hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3:366–379.