Marcos Aurélio Pereira

Simple Ant Colony Routing Algorithm: Um Algoritmo de Roteamento Energeticamente Eficiente para Redes de Sensores Sem Fio

Florianópolis - SC, Brasil

15 de maio de 2012

Marcos Aurélio Pereira

Simple Ant Colony Routing Algorithm: Um Algoritmo de Roteamento Energeticamente Eficiente para Redes de Sensores Sem Fio

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de bacharel em ciências da computação

Orientador:
Antônio Augusto Medeiros Fhrölich

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DE SANTA CATARINA

Florianópolis - SC, Brasil

15 de maio de 2012

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de bacharel em ciências da computação.

Prof. Dr. Antônio Augusto Medeiros Fhrölich
Orientador
Universidade Federal de Santa Catarina

M.Sc Alexandre Massayuki Okazaki
Co-orientador
Universidade Federal de Santa Catarina

M.Sc Arliones Stevert Hoeller Júnior
Universidade Federal de Santa Catarina

B.Sc Rodrigo Vieira Steiner
Universidade Federal de Santa Catarina

Agradecimentos

Agradeço inicialmente a meus pais Pedro e Maria que me forneceram muito mais do que o necessário para ser capaz de realizar esta consquista. Que me acompanharam em cada etapa e cujo orgulho sempre foi, e sempre será, motivo mais do que suficiente para ter força e consquistar qualquer coisa em minha vida. Pelo apoio incansável em cada momento de minha graduação, agradeço minha namorada Amanda que me inspira a querer ser alguém melhor a cada dia.

Ao professor Guto pela confiança depositada e a oportunidade de realizar um trabalho tão desafiador e prazeroso. Além disso, por toda a dedicação em fazer com que suas disciplinas tragam lições importantes e duradouras para o resto da vida de quem tem o privilégio de cursá-las, sem dúvida, algumas das mais interessantes de toda minha graduação. Também não teria chegado tão longe ao longo deste trabalho sem o apoio do Alexandre, por quem desenvolvi profundo respeito e admiração. Não menos importante, agradeço o restante do pessoal do LISHA, em especial Arliones, Rodrigo e Mateus por suas preciosas sugestões para este trabalho.

Resumo

Algoritmos de roteamento constituem uma peça chave no comportamento de uma rede de sensores sem fio. Os recursos limitados dos sensores sem fio, por sua vez, obrigam tais algoritmos a serem pensados de forma a consumir o mínimo possível dos recursos como memória e energia de um nó sensor. Neste trabalho é apresentado o **SACRA** (Simple Ant Colony Routing Algorithm), um algoritmo de roteamento que busca a redução do consumo de energia em redes de sensores sem fio.

Ant Colony Optimization é uma técnica de swarm intelligence[1] que vem sendo bastante utilizada em algoritmos de roteamento para *RSSF* afim de otimizar algum determinado aspecto das mesmas. Esta técnica é utilizada pelo **SACRA** durante a fase de descoberta de rotas.

O diferencial do algoritmo proposto, frente a outros que fazem uso de *ant colony optimization*, é procurar a máxima simplificação para a aplicação da técnica sem que ela perca suas características. O aspecto a ser otimizado é o consumo de energia. Para isto é utilizada uma métrica intimamente ligada a energia restante dos nós ao longo de um caminho e ao número de rotas que cada nó já mantém ativas no momento da descoberta. De posse deste valor, as heurísticas aplicadas pela técnica buscam formar rotas de forma que a distribuição do consumo de energia entre os nós de uma rede seja a melhor possível. Além disso, o algoritmo determina proativamente, quando um nó consome uma quantidade considerável de energia servindo como intermediário de rotas de dados, forçando estas a serem movidas para caminhos alternativos e com melhores saldos de energia restante.

Além disso o **SACRA** busca a utilização de mecanismos que, apesar da simplicidade, sejam comprovadamente eficientes na realização do roteamento. A manutenção de rotas, por exemplo, é baseada na utilizada pelo algoritmo **AODVjr**, porém, é introduzido um mecanismo que trabalha em prol da distribuição da carga de roteamento entre os nós da rede.

O trabalho conta ainda com uma análise dos dados obtidos em simulações realizadas nos simuladores **sinalgo** e **omnet++** e a comparação do **SACRA** com outros algoritmos conhecidos em *RSSF*.

Sumário

1 Introdução

O consumo eficiente da bateria de um nó sensor em uma rede de sensores sem fio é de suma importância, visto que este pode estar instalado em um local de difícil acesso. O alto consumo do emissor de rádio comparado aos demais componentes de um sensor[2], torna o roteamento um ponto importante ao se considerar este problema.

O foco do **SACRA** (Simple Ant Colony Routing Algorithm) é realizar o roteamento de dados utilizando os recursos energéticos da rede de forma inteligente e com o mínimo possível de overhead no tráfego de pacotes. Para isto ele utiliza a técnica "*Ant Colony Optimization*".

A principal motivação para o desenvolvimento deste trabalho é contribuir com o desenvolvimento de redes de sensores sem fio energeticamente eficientes, uma vez que esta tecnologia é aplicada para o desenvolvimento de pesquisas em diversas áreas como , por exemplo, *"Internet Of Things"*[3], que constitui o futuro da disponibilização e troca de dados sem intervenção humana pela internet. Além disso, a característica de eficiência energética do **SACRA** pode contribuir muito com o desenvolvimento de *smart buildings*, visto que estes também buscam tal eficiência. Portanto, dadas as possíveis aplicações para RSSF as contribuições de seus algoritmos de roteamento vão além de sua importância para as redes em si.

Em seguida são apresentados os objetivos geral e específicos do trabalho, bem como, uma descrição de sua estrutura.

1.1 Objetivo Geral

Construir um algoritmo ciente de energia para redes de sensores sem fio, capaz de oferecer uma boa taxa de entrega de pacotes, com pouco overhead de pacotes de controle e um consumo de energia baixo e balanceado.

1.2 Objetivos Específicos

O algoritmo deve ser capaz de fornecer uma taxa de entrega de pacotes que garanta, ao menos, um minimo de qualidade da rede.

Os níveis médios de consumo de energia devem ser inferiores aqueles verificados nos demais algoritmos utilizados para comparação em condições de igualdade de ambientes e configurações.

1.3 Metodologia

Conhecer diferentes algoritmos de roteamento para *RSSF*'s afim de obter informações a respeito de abordagens já utilizadas neste contexto para atacar o problema da minimização do consumo de energia.

Prototipar o algoritmo, focando em simplicidade e idéias criativas e eficientes para auxiliar no balanceamento do consumo de energia.

Comparar o algoritmo com outros já conhecidos por meio de simulações. Além disso, realizar uma análise dos dados obtidos.

1.4 Estrutura do Trabalho

O texto está organizado da seguinte maneira: o capítulo 2 mostra um visão geral de *swarm intelligence* e descreve a técnica *ant colony optimization*. O capítulo 3 traz uma descrição de alguns algoritmos conhecidos em redes de sensores sem fio que serão utilizados nas comparações com o **SACRA**. O capítulo 4 descreve o algoritmo proposto, suas tabelas de rotas e mecanismos utilizados para realizar o roteamento visando a otimização do consumo de energia dos nós da rede. O capítulo 5, a respeito dos resultados obtidos em simulações, traz uma comparação entre o algoritmo proposto e os demais algoritmos descritos no capítulo 3. Os capítulos 6 e 7 apresentam, respetivamente, as conclusões obtidas com o trabalho realizado e as propostas de trabalhos futuros para a melhoria do **SACRA**.

2 *Ant Colony Optimization*

Swarm Inteligence é uma ferramenta de inteligência artificial baseada no comportamento de enxames de insetos e outros grupos de animais. Ela baseia-se na interação entre os agentes e o seu ambiente para a obtenção de um objetivo. Por não possuir uma estrutura de controle centralizada, e ter como foco a obtenção de um comportamento inteligente global do sistema[1], esta técnica adequa-se perfeitamente as necessidades de um algoritmo de roteamento para redes de sensores sem fio onde se busca um balanceamento do consumo de energia global da rede.

Este capítulo descreve a técnica conhecida como *ant colony optimization* que, dentre as técnicas de *Swarm Intelligence*, é a que mais se destaca na aplicação em redes de sensores sem fio. O principal aspecto de destaque desta é permitir a inserção de heurísticas a serem utilizadas para determinar o que é ou não um bom caminho a ser escolhido. Isto permite a consideração da energia na escolha de rotas o que serve perfeitamente aos objetivos deste trabalho.

Grassé usa o termo estigmergia para definir o processo de comunicação indireto mediado pelo ambiente onde os indivíduos estão inseridos[4]. Algumas espécies de formigas se comunicam modificando o ambiente ao seu redor com o intuito de chegarem a um objetivo comum. Informações stigmérgicas podem ser acessadas apenas localmente, ou seja, elas só interferem no comportamento do indivíduo que o está acessando ou de sua vizinhança mais próxima.

Ant Colony Optimization é inspirada no comportamento de algumas espécies de formiga em busca de alimento. Essas formigas depositam feromônios no chão com o intuito de marcar o caminho da toca até o alimento enquanto o procuram. Ao encontrar o alimento a formiga volta pela trilha de feromônios que ela previamente deixou depositado no chão. Neste processo de volta até a toca ela segue depositando seus feromônios para manter a trilha ativa. As novas formigas que partirão da toca perceberão que já existe uma ou mais trilhas partindo dali e seguirão por uma delas também depositando feromônios na trilha que escolheram percorrer. Os feromônios depositados evaporam ao longo do tempo, assim as trilhas que não são utilizadas tendem a desaparecer. Quanto maior o número de formigas caminhando por uma trilha, maior será a quantidade de feromônios depositados na mesma. As formigas se sentirão mais atraídas

pelas trilhas com maiores níveis de feromônios e consequentemente estas trilhas acumularão cada vez mais feromônios e se tornarão mais atraentes a elas. Isso faz com que o menor caminho até o alimento seja naturalmente escolhido e de forma inconsciente por parte dos agentes, no caso as formigas, isto porque a formiga que retornar primeiro à toca após encontrar o alimento, acaba tornando sua trilha mais rica em feromônios atraindo novas formigas para ela.

O comportamento das formigas é um exemplo claro de como se dá a comunicação através da interação com o ambiente e como esta interação leva a resolução de um problema, no caso, a busca por comida.Uma meta-heurística é em síntese um método heurístico utilizado para resolver de forma genérica um problema de otimização. Trata-se de um método computacional que busca uma solução iterativamente, considerando uma solução candidata a cada iteração e realizando continuas tentativas de aprimoramento desta solução, onde é utilizada uma medida de qualidade previamente definida para determinar o quão boa é a solução encontrada em uma dada iteração[5].Ant Colony Optimization, é uma meta-heurística, e como tal, define um método capaz de resolver uma série de problemas através da realização de relativamente poucas modificações na base do método. Um modelo P para um problema de otimização combinatória pode ser definido como:

$$P = (S, \Omega, f)$$

onde:

- S é o espaço de busca, definido com um conjunto finito de variáveis discretas de decisão $X_i, i = 1, \ldots, n$;
- Ω é um conjunto de restrições sobre as variáveis;
- f é uma função objetivo $f : S \to \mathbb{R}^+$, a ser minimizada (ou maximizada tomando $g = -f$ como função objetivo). Uma variável X_i toma valores num conjunto

  $$Di = \{v(i, j) | j = v_i^1, ..., v_i^{|D_i|}\}$$

  Uma possível solução $s \in S$ consiste na atribuição de valores a todas as variáveis X satisfazendo as restrições contidas em Ω .

Uma solução $s^* \in S$ é uma solução ótima global se, e somente se:

$$f(s*) \leq f(s) \, \forall \, s \in S$$

O modelo P é utilizado para definir o modelo de feromônios em *ant colony optimization*. Um valor de feromônio τ_{ij} é dado a cada variável envolvida em uma possível solução c_{ij} do problema, o que consiste na atribuição:

$$X_i = v_{ij}$$

O conjunto de todas as possíveis soluções é denotado por C.

Dado um grafo $G_c(V,A)$, onde V é um conjunto de vértices e A é um conjunto de Arestas. Este grafo é definido a partir dos componentes de solução c que podem ser representados tanto pelos vértices quanto pelas arestas de G. Formigas artificiais buscarão uma solução caminhando vértice a vértice através das arestas do grafo e construindo incrementalmente uma solução parcial. Durante este processo elas depositarão feromônios nos componentes c, fazendo com que a quantidade de feromônios Δ_τ depositada em um componente seja dependente da qualidade da solução encontrada. Com isso as demais formigas utilizarão as quantidades de feromônios já determinadas como um ponto de partida para sua tomada de decisão em relação as melhores áreas a percorrer dentro do espaço de busca S.

A meta-heurística *ant colony optmization* consiste em uma fase de inicialização seguida por iterações que, por sua vez, são divididas em duas fases principais.

Na fase de inicialização são setados os parâmetros do algoritmo como, por exemplo o fator de evaporação do feromônio, e inicializar o feromônio das trilhas.

Dentro de cada iteração existe a fase de construção de soluções e a fase de atualização dos ferômonios.

Algoritmo 1: Metaheurística Ant Colony Optimization

```
while condição de parada não satisfeita do
    ContruirSoluções;
    AtualizarFeromônios;
end
```

Na fase de construção, um conjunto de m formigas artificiais constrói suas soluções a partir de um conjunto finito de variáveis em:

$$C = \{c_{ij} | i = 1,\ldots,n \wedge j = 1,\ldots,|D_i|\}$$

O Ponto de partida para a construção de uma solução é uma solução parcial vazia $s^p = \emptyset$. Define-se $N(s^p) \in C$ como o conjunto dos componentes que podem ser adicionados a atual

solução parcial s^p sem violar nenhuma das restrições de Ω. A solução s^p inicialmente vazia vai sendo incrementada a cada iteração através da adição de uma possível solução presente no conjunto $N(s^p)$.

Na fase de atualização dos feromônios são realizadas duas operações:

1. Subtração dos valores de feromônios segundo a equação que rege a evaporação dos mesmos utilizando um valor que indica o fator de evaporação.

2. Adição de feromônios aos valores associados a boas soluções, ou seja, trilhas percorridas por formigas na iteração corrente[6].

A escolha de uma possível solução presente em $N(s^p)$ é guiada por um mecanismo estocástico regulado pelos feromônios relacionados a cada elemento $N(s^p)$. O mecanismo pode variar para cada algoritmo onde é aplicada a técnica, porém, deve refletir algum modelo do comportamento real de formigas como o visto em [6].

3 *Algoritmos de Roteamento*

Este capítulo tem por objetivo fornecer uma breve análise de alguns pontos inerentes a redes de sensores sem fio e uma descrição básica dos principais aspectos de alguns algoritmos que serão citados e utilizados para comparação ao longo do trabalho.

Uma das questões a ser tratada para cada rede de sensores sem fio, é a aplicação a qual a mesma servirá. Assim, diferentes algoritmos de roteamento poderão se comportar de maneira adequada, ou não, dependendo da aplicação a ser considerada. Porém, alguns aspectos são comuns a maioria das aplicações em redes de sensores sem fio. Entre eles estão:

- Recursos limitados de memória, processamento e energia.
- Topologia dinâmica devido a mobilidade. e possibilidade de falhas dos nós.
- Possibilidade de difícil manutençao, fazendo com que o tempo de vida da rede seja um ponto crítico a ser considerado para sua implantação.

Tendo em vista estas características, podemos concluir que alguns dos principais pontos a serem buscados por um algoritmo de roteamento para redes de sensores sem fio são[rotsensores]:

- Minimizar o consumo de energia.
- Distribuir tarefas de roteamento entre os nós.
- Ser tolerante a falhas.

3.1 AODVjr

O Algoritmo de roteamento **AODVjr** (AODV simplificado)[7], trata-se de uma simplificação do muito estudado[8] e utilizado **AODV** (Ad-hoc On Demand Distance Vector)[9]. Este protocolo se enquadra na família dos protocolos reativos. Nesta categoria as rotas são formadas

sob demanda, ou seja, o protocolo só faz uma busca por rotas quando lhe é solicitado enviar dados a um destino para o qual ele não conhece um caminho. O **AODVjr**, modifica alguns aspectos do **AODV** tradicional afim de diminuir a quantidade de pacotes de controle e simplificar os mecanismos de descoberta e manutenção de rotas.

Este algoritmo retira os sequintes itens da especificação do **AODV** original[9]:

- Números de sequencia.
- RREP's desnecessários.
- Contagem de saltos.
- Mensagens "HELLO".
- Mensagens de falha de link (RERR).
- Lista de precursores.

O mecanismo de descoberta de rotas realiza o tradicional *flood* de pacotes RREQ solicitando uma rota para o destino desejado. Para se livrar da necessidade de manter números de sequencia para garantia da ausencia de loops de roteamento, o algoritmo faz com que apenas os destinos das RREQ's possam responder com um RREP. Como nenhum nó intermediário poderá responder a um RREQ, os RREP's "gratuitos" também são eliminados. Além disso, toda descoberta de rotas, gera rotas bidirecionais[7].

A manutenção das rotas é feito por meio de um tempo de vida pré determinado para uma rota. Este tempo de vida só poderá ser atualizado através do recebimento de pacotes e nunca pelo envio. Desta forma, é necessário que o nó destino envie algum pacote para o nó origem de uma rota periodicamente, caso contrário, a mesma se tornará inválida ao fim de seu tempo de vida. Caso a comunicação seja unidirecional, o nó destino envia pacotes "CONNECT" para a origem da rota, afim de atualizá-la. No caso em que a comunicação é bidirecional, as rotas são atualizadas sem a necessidade de um pacote extra o que implica em um overhead de pacotes de controle ainda menor.

Caso o nó origem deixe de receber dados ou "CONNECT" do nó destino, após algum tempo sem ser atualizada, a rota será retirada da tabela de rotas do nó origem. Este mecanismo é eficiente em determinar uma quebra de rota, visto que neste caso o nó origem deixa de receber informações do nó destino. Desta forma, as mensagens "RERR" não são mais necessárias. Após a eliminação de uma rota, caso o nó origem deseje enviar dados para o mesmo destino, ele dará inicio a um novo processo de descoberta de rotas.

Este algoritmo tem um foco muito grande na simplicidade, e diversos aspectos do **SACRA** foram inspirados nas simplificações por ele apresentadas.

3.2 AOER

O **AOER** (Ant-based On-demand Energy Route Protocol)[8] também é caracterizado como um algoritmo reativo. Além disso este algoritmo pertence a família dos algoritmos cientes de energia. Este tipo de algoritmo é caracterizado por buscar um uso eficiente dos recursos energéticos dos nós de uma rede buscando aumentar seu tempo de vida. Este algoritmo utiliza *ant colony optimization* para alcançar seus objetivos de otimização do consumo de energia dos nós.

Na fase de descoberta de rotas, são enviados pacotes de requisição denominados *"forward ants"* afim de encontrar uma rota eficiente até o destino. Para isto o pacote formiga carrega informações a respeito da energia restante do último nó visitado, além de outras métricas, como a média de energia restante dos nós por onde ela já passou. Enquanto percorre os nós a formiga faz depósitos de feromônios proporcionais ao quanto a participação de um dado nó em uma rota, pode torná-la energeticamente eficiente. Estes valores de feromônios, por sua vez, são armazenados em tabelas de feromônios inversas. O termo "inversas" se referem ao fato de estas tabelas apontarem caminhos para o nó origem da rota a ser descoberta.

Ao receber uma *"forward ant"* o nó destino envia uma *"backward ant"*. Esta percorre o caminho de volta até o nó origem decidindo quais nós intermediários utilizará baseada nas tabelas de feromônios previamente preenchidas. A escolha do próximo destino em cada ponto da rota, é realizada de forma probabilistica, onde quanto maior a quantidade feromônios associada a um dado destino, maior a probabilidade desta rota ser escolhida pela formiga.

Um mecanismo particularmente interessante deste algoritmo ocorre quando um nó recebe uma *"forward ant"*. Caso a energia restante deste nó esteja muito abaixo da média de energia calculada pela formiga recebida, este nó não retransmitirá esta formiga, impedindo que ele seja escolhido para estabelecimento de uma rota. Apesar de interessante para o aumento do tempo de vida da rede e distribuição uniforme do consumo de energia entre os nós, é possível que este mecanismo tenha um impacto negativo na taxa de entrega de dados. Isto pode ocorrer no caso em que o nó que se recusa a aceitar uma rota, seja a única alternativa para o estabelecimento da mesma. Indícios deste comportamento podem ser verificados nas análises realizadas em [10] e nos resultados obtidos neste trabalho. O **SACRA** utiliza um mecanismo parecido com este, porém, propõe uma importante modificação que permite minimizar o impacto desta decisão na

taxa de entrega de pacotes.

A manutenção de rotas no **AOER** é semelhante a utilizada no **AODVjr**. A principal diferença reside na capacidade do algoritmo de determinar momentos oportunos para forçar uma redescoberta de rotas baseado na energia consumida por um nó enquanto serve uma rota. Isto permite que, periodicamente, a comunicação entre dois nós, utilize diferentes nós intermediários, distribuindo melhor o consumo de energia.

3.3 ADHOP

ADHOP (Ant-based Dynamic hops Optimization Protocol) é um protocolo reativo que busca eficiencia na taxa de entrega de dados assim como a minimização do overhead no tráfego de pacotes da rede[10]. Para atingir seus objetivos, também é utilizada a técnica *ACO*.

Neste algoritmo as rotas não são pre-determinadas. Os saltos de uma rota vão sendo selecionados dinamicamente em cada nó até o destino, o que facilita a adaptação as mudanças constantes de topoliga de uma rede móvel. Além disso, todo pacote de dados é uma formiga, ou seja, ainda que uma formiga necessite explorar uma área para encontrar o destino, uma vez localizado, o pacote de dados é entregue.

Para sua operação o algoritmo utiliza dois tipos de pacotes (formigas). São eles:

- **Exploratory Transport:** reponsável por buscar o destino de um pacote quando, em um dado nó da rota, não é possível encontrar um próximo salto. As formigas exploratórias avançam através dos nós buscando o destino ou um nó que já possua uma trilha para o destino. Assim elas possuem a função de descobrir uma rota ou religar trilhas (rotas) evaporadas.
- **Forward Transport:** Carrega os dados utilizando as rotas e reforçando as trilhas de feromônios das rotas deixadas pelas formigas exploratórias.

Além disso o algoritmo permite a aplicação de heuristicas para otimizar um determinado aspecto da rede. Em [11] podemos ver resultados obtidos com a aplicação de heurísticas relacionadas a otimização da energia ou da latência na comunicação.

4 O algoritmo SACRA: Simple Ant Colony Routing Algorithm

Este capítulo descreve o algoritmo **SACRA** (Simple Ant Colony Routing Algorithm), suas estruturas e mecanismos de prevenção de loops destacando os principais pontos relacionados ao consumo eficiente de energia da rede. Como o próprio nome sugere, o algoritmo tem foco na simplicidade. O que se pretende é utilizar mecanisnos simples, porém, que apresentem potencial na tentativa de otimizar o consumo de energia da rede.

4.1 Tabelas de rotas

O algoritmo proposto possui duas tabelas de rotas. Uma delas é utilizada para armazenar as rotas inversas, utilizadas na primeira fase da descoberta de rotas, enquanto a outra armazena as rotas efetivas por onde trafegarão os dados.

A tabela de rotas inversas é especialmente importante para o algoritmo, visto que é nela que estarão armazenados os valores de feromônios a serem utilizados na aplicação de *ant colony optimization* durante a descoberta de rotas.

O uso de uma tabela de rotas inversas, bem como, a utilização de um campo *time to live* nas rotas efetivas foi inspirado pelo algoritmo **AODVjr**[7], anteriormente descrito.

A seguir encontra-se uma descrição completa dos campos de cada tabela:

4.1.1 Tabela de rotas inversas

Definição 1 *Sejam A e B dois nós da rede e A deseja enviar dados para B. Uma rota inversa é definida como o caminho de retorno de B para A a ser percorrido por uma backward ant afim de estabelecer uma rota bidirecional entre A e B.*

Cada entrada na tabela de rotas inversas (tabela 4.1) possui os seguintes campos:

- **Destino**: Armazena o identificador do nó que iniciou a descoberta de rotas.
- **Próximo**: Para cada **Destino** há uma lista de um ou mais vizinhos que podem ser escolhidos para constituir uma rota efetiva. Estes vizinhos são identificados pelo campo **Próximo**. Cada par (destino,próximo), identifica uma rota inversa.
- **Feromônios**: A cada vizinho identificado como **Próximo** é associado um valor de feromônios, armazenado neste campo.
- **Saltos**: Armazena o menor número de saltos registrado para o destino.

Destino	**Próximos**		**Saltos**
A	**Próximo**	**Feromônios**	1
	A	130023	
	B	10989	
	D	11985	
H	**Próximo**	**Feromônios**	5
	F	21565	
	G	1054	
	X	36775	

Tabela 4.1: Exemplo entradas na tabela de rotas inversas.

4.1.2 Tabela de rotas

Cada entrada na tabela de rotas efetivas (tabela 4.2) possui os seguintes campos:

- **Destino**: Armazena o identificador do ponto final da rota efetiva, ou seja, o nó que receberá os dados enviados pelo nó que solicitou uma rota.
- **Próximo**: Armazena o identificador do próximo nó que um pacote deve visitar afim de alcançar o destino por meio desta rota.
- **TTL**: Significa *"time to live"* e trata-se de um timeout para determinar se uma rota continua ativa. O campo é decrementado continuamente e, ao ser zerado, a rota é removida da tabela de rotas. Este campo é utilizado pelo mecanismo de manutenção de rotas, desempenhando um papel importante em diversos aspectos do algoritmo.

Destino	Próximo	TTL
D	B	5
X	G	8
C	F	2

Tabela 4.2: Exemplos de entradas na tabela de rotas.

4.2 Descoberta de rotas

Afim de manter a simplicidade, decidiu-se utilizar *ant colony optimization* apenas na fase de descoberta de rotas. Assim esta fase é a responsável pela otimização do consumo de energia.

Quando um determinado nó deseja enviar dados a um outro nó da rede para o qual ele não possui uma rota ativa, é iniciado um processo de descoberta de rotas afim de obter o melhor caminho até o destino desejado. Nesta fase há um intensa troca de informações entre todos os nós da rede, devido ao *flooding* de pacotes de requisição de rota. A aplicação de *ant colony optimization* permite que isto seja realizado com pouco overhead de pacotes de controle.

Após o estabelecimento das rotas inversas a técnica utilizada permite que os nós possuam consciência, ainda que de forma indireta, de qual vizinho possível para o estabelecimento de uma rota é o que possui maior quantidade de energia. Detalhes a respeito da obtenção esta "consciência" serão explicados ao longo da descrição do algoritmo nas próximas seções.

Os pacotes envolvidos nas descobertas de rotas são os seguintes:

- **Forward ant** (tabela 4.3): partem do remetente estabelecendo rotas rumo a este. São elas ainda que carregam as informações relevantes para os mecanismos de prevenção de loops. A estrutura de uma forward ant é a seguinte:
  - **Origem**: Identificação do nó que inicia a descoberta de rotas, ou seja, aquele que deseja enviar dados.
  - **ReqId**: Número que junto ao campo origem, identifica unicamente uma requisição de rotas.
  - **Destino**: Identificação do nó para o qual a origem deseja estabelecer uma rota.
  - **Rank**: Valor que determina o quanto um nó está apto a receber uma rota de dados. Este é o valor mais importante da heurística aplicada para o consumo inteligente dos recursos energéticos da rede. Este campo deve estar fortemente ligado a informações como a quantidade de energia restante do nó.
  - **Último**: Identificação do último nó visitado pela **Forwad Ant**.

  - **Saltos**: O número de nós visitados pela *fant* em uma rota. Este campo é incrementado sempre que a fant é recebida por um nó.

Origem	ReqId	Destino	Rank	Último	Saltos

Tabela 4.3: Estrutura de uma forward ant

- **Backward ant** (tabela 4.4): Seguem do destino para o remetente utilizando as rotas estabelecidas pelas *forward ants*. Elas selecionam probabilisticamente as melhores rotas armazenadas nas tabelas de rotas inversas, estabelecendo assim as rotas efetivas entre os nós. A estrutura de uma backward ant é a seguinte:

  - **Origem**: Identificação do nó que lançou a backward ant, ou seja, o nó destino do processo de descoberta de rotas.
  - **Destino**: Identificação do nó destino da backward ant, ou seja, o nó que iniciou a descoberta.
  - **Último**: Identificação do último nó visitado pela **Backward Ant**.

Origem	Destino	Último

Tabela 4.4: Estrutura de uma backward ant

A partir deste ponto trataremos as forward ants e backward ants pelas abreviações **fant** e **bant** respectivamente.

O desafio desta etapa é gerar o maior número possível de rotas inversas criadas pelas **fants**, para que as **bants** possuam um bom número de alternativas de rotas. Além disso, estas rotas devem ser livres de loops de roteamento.

O mecanismo esolhido para evitar os loops é baseado na contagem de saltos das rotas inversas geradas pelas fants e será explicado em detalhes mais a frente.

4.2.1 Envio de uma Forward Ant

O processo de descoberta de rotas inicia no momento em que um nó deseja enviar dados a um destino para o qual não possui uma rota ativa. O nó incrementa o seu contador de requisições e cria uma fant preenchendo seus campos da seguinta forma:

- Origem: Seu identificador.

- ReqId: o valor de seu contador de requisições interno.
- Destino: O identificador do nó para o qual deseja enviar dados.
- Rank: Seu valor de Rank dado pela equação 4.1.
- Último: Seu próprio identificador.
- Saltos: 0.

Em seguida o nó envia fant (figura 4.2.1) a todos os seus vizinhos[1].

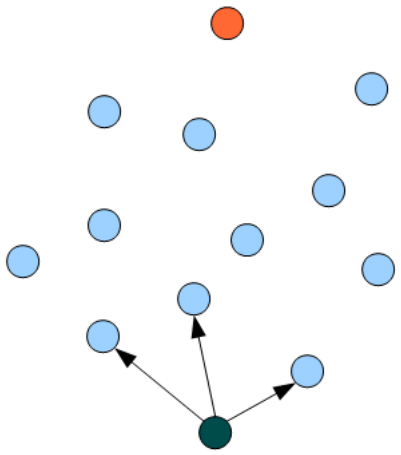

Figura 4.1: O nó origem (em verde) envia uma fant afim de estabelecer uma rota para o nó destino (em laranja).

4.2.2 Recebimento de uma Forward ant

Ao receber uma *forward ant*, o nó encontra-se no ponto crucial para a determinação do maior número possível de rotas inversas sem loops. Após incrementar o número de saltos da fant o nó decide se deve ou não adicionar uma rota em direção ao nó origem e apontando para o último nó visitado, a sua tabela de rotas inversas.

Os critérios para que se adicione uma possível rota inversa são os seguintes:

1. O nó ainda não possui nenhuma rota inversa para a origem da fant.

[1] por vizinhos, entenda-se todos os nós capazes de receber uma mensagem enviada por broadcast a partir de um dado nó.

2. Já existe uma rota inversa para a origem, a tabela de próximos saltos da rota existente ainda não possui o nó identificado pelo campo **Último** e o contador de saltos da **fant** é menor ou igual ao campo **Saltos** da rota. Isto garante que as rotas inversas sejam livres de loop.

3. O nó atual é o destino da fant. O nó destino pode adicionar todas as rotas visto que ele é o ponto final da fant e como as rotas até o ponto imediatamente anterior são livres de loops, não é possível que seja gerado um loop a partir do destino.

Após analizar a rota percorrida pela fant e determinar se ela deve ser inserida na tabela de rotas inversas, o nó verifica se já existe uma rota com a mesma assinatura, ou seja, partindo da mesma origem e chegando até o nó atual através do mesmo nó vizinho. Neste caso, a rota existente recebe um incremento em sua quantidade de feromônios, tornando-se mais atraente a uma bant. Caso trate-se de uma rota nova, ela é inserida contendo a mesma quantidade de feromônios que receberia como incremento no caso anterior. Este valor é dado pela seguinte equação:

$$\tau_i = \omega r_i - \gamma s \tag{4.1}$$

Sendo τ_i a quantidade de feromônios a ser depositado na rota cujo próximo salto é o nó i, ω o peso relacionado ao valor de *Rank* carregado pela **fant** recebida, r_i é o *Rank* calculado no nó i, γ o peso que o número de saltos deve ter no valor de τ e s o número de saltos carregado pela **fant**, assim, caso seja desejado que as rotas com menor número de saltos sejam preferidas, este valor deve ser incrementado.

Após esta etapa, é verificado se a fant já foi recebida e processada pelo nó atual. Isso é possível armazenando uma lista dos pares (origem, reqId) atendidos recentemente e verificando se o par da fant recebida já existe nesta lista. Qualquer fant recebida que já tenha sido processada será descartada neste ponto. A cada nova requisição recebida, os pares de requisições anteriores podem ser excluídos afim de poupar memória.

Aqui age o primeiro mecanismo responsável pela distribuição do consumo de energia. Caso o nó atual seja apenas um intermediário para o estabelecimento da rota, é verificado se este encontra-se em um estado de "baixa energia". O estado citado é determinado com base em um threshold configurável que indica o limite a partir do qual a quantidade de energia restante é considerada pouca. Quando neste estado, o nó possui setada uma flag global que o impede de processar a primeira requisição recebida em um processo de descoberta. Após negar o processamento de uma fant devido a esta flag, a mesma é resetada de forma a permitir o processamento

da próxima requisição, o que é necessário pois este nó pode ser a única opção para o estabelecimento da rota desejada.

Este mecanismo tem a intenção de dificultar a formação de rotas passando por um nó que possui pouca energia restante. A eficiência deste procedimento reside no fato de que a negação do estabelecimento de algumas rotas inversas passando por este nó, aumenta as chances de que uma rota mais longa, porém com mais energia, seja escolhida.

A flag ainda é setada novamente para negar o processamento de requisições em dois momentos:

1. Quando o nó detecta a evaporação completa de alguma trilha de feromônios, indicando que um processo de descoberta de rotas já foi terminado a um certo tempo.

2. Quando o nó recebe uma bant, pois neste caso uma rota já foi estabelecida passando pelo nó.

Em um nó intermediário, o processamento de uma fant desempenha um papel crucial na aplicação da heurística de energia aplicada em *ant colony optimization*. Isto porque, antes de ser reenviada a todos os vizinhos por broadcast, o campo **Rank** da fant é atualizado com os dados do nó atual. O valor aplicado a este campo é dado pela seguinte equação:

$$Rank = E - \lambda |R| \tag{4.2}$$

Sendo E a energia restante, R o conjunto das rotas efetivas e λ o peso relativo a penalidade do *Rank* em relação ao número de rotas efetivas dado por $|R|$.

Como o valor de Rank da fant reflete na quantidade de feromônios da rota inversa apontando para o nó atual, a utilização da quantidade de energia na equação faz com que nós com maior quantidade de energia se tornem mais atraentes para uma bant. Pode-se perceber ainda que o valor recebe uma penalização relacionada a quantidade de rotas efetivas que o nó já está atendendo. A intenção desta penalidade é fazer com que nós que já possuem um grande número de rotas passando por eles, se tornem menos atraentes e consequentemente possuam uma probabilidade menor de serem selecionados para uma rota efetiva.

Além da atualização do *"rank"* da fant, ela recebe o identificador do nó atual em seu campo **Último**.

O processo de envio de fants e formação de rotas inversas, é ilustrado pela figura a seguir:

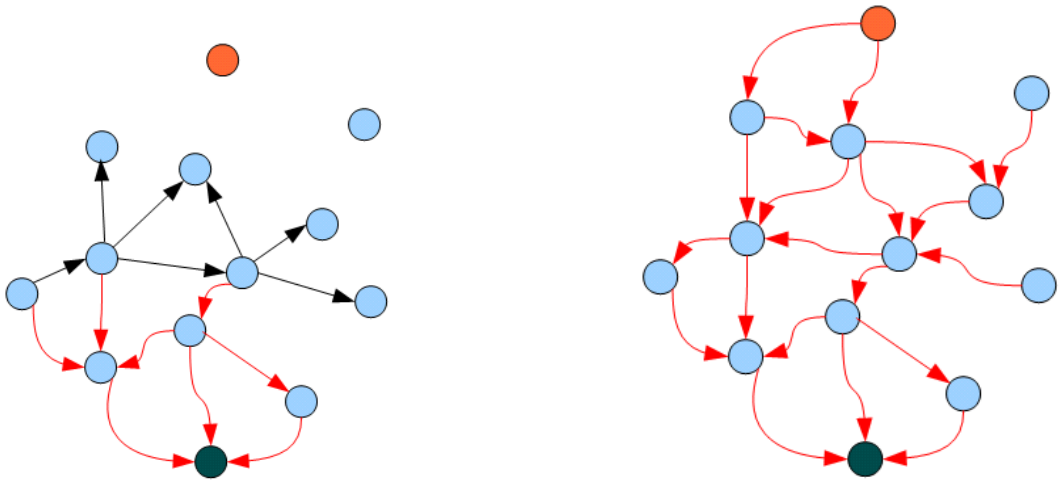

Figura 4.2: Processo de envio de fants e formação de trilhas de feromônios.

Ao chegar ao nó destino a fant encontra seu ponto final. Ao receber a primeira destas, o nó destino inicia um timer para o envio de uma bant. A intenção de atrasar o envio de uma bant é permitir que outras fants cheguem até o destino e, desta forma, fornecer um número maior de opções de rotas no momento em que a bant iniciar sua jornada até a origem para o estabelecimento de uma rota efetiva.

Ao fim da contagem do timer, o nó gera então uma **bant** com os seguintes valores em seus campos:

- Origem: O identificador no nó destino da **fant**, ou seja, o nó atual.
- Destino: Recebe o campo Origem da **fant**.
- Último: Recebe o identificador do nó atual.

As próximas seções mostram como é feito o envio da **emph** bant e como as rotas efetivas são estabelecidas. No entanto é importante entender, anteriormente, como a *bant* seleciona os melhores nós para compor uma rota.

4.2.3 Escolha da Melhor Rota Inversa

Com a tabela de rotas inversas devidamente preenchida nos passos anteriores, a **bant** possui agora diversas opções de rotas a seguir. A cada passo ela deve decidir qual o melhor caminho escolhendo, estocasticamente, uma entre todas as rotas possíveis até o seu destino.

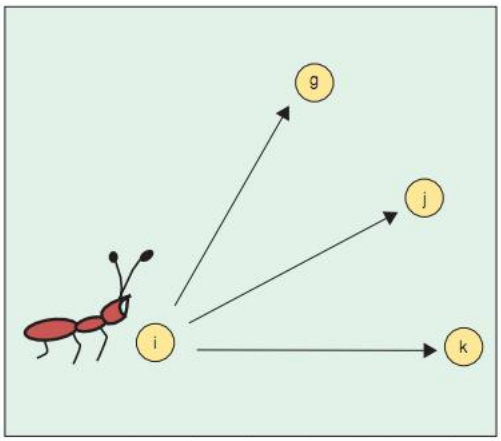

Figura 4.3: Uma bant no nó *i*, decidindo qual seu próximo destino. O nó que possui maior quantidade de feromônios na tabela deve possuir maior probabilidade de ser escolhido.

Para aumentar as probabilidades de que a trilha com maior concentração de feromônios seja selecionada, é utilizado uma estratégia de seleção chamada "seleção proporcional pseudo-randômica". A escolha desta estratégia é baseada na escolhida pelo algoritmo **AOER** descrito em [8].

A **bant** analisa a rota cujo campo *Destino* coincide com seu campo homônimo. A lista de **próximos** desta rota, constitui o conjunto dos possíveis próximos saltos da *bant*.

Seja N o conjunto de todos os nós vizinhos ao nó atual que podem ser escolhidos como próximo salto rumo ao nó destino e $\tau_n, n \in N$, o valor que representa a quantidade de feromônios associada ao nó vizinho n. A probabilidade p_j de uma rota ser escolhida, onde j é o próximo salto da mesma, é dada pela seguinte equação:

$$p_j = \frac{\tau_j}{\sum_{i \in N} \tau_i}$$

Uma vez calculadas as probabilidades, é sorteado um número pseudo-aleatório α no intervalo $[0,1]$ e o próximo salto k é selecionado através dos seguintes critérios:

$$k = \begin{cases} n \in N \wedge n = \max_{i \in N} \tau_i, & \alpha < p_n \\ m, & \text{caso contrário} \end{cases}$$

Onde m é selecionado pelo método da roleta, descrito na sequência.

Define-se A, como o conjunto das probabilidades acumuladas para cada nó $n \in N$. Onde a probabilidade acumulada a de um nó i ser escolhido é dada por:

$$a_i = \sum_{\substack{k \in N \\ k<i}} p_k, i \in N$$

O critério de ordenação dos nós é baseado na ordem que os mesmos aparecem na tabela de rotas inversas que, por sua vez, é determinado pela ordem de inserção na mesma. Uma vez definido o conjunto A, o nó m pode finalmente ser selecionado como:

$$m = j \in N \quad | \quad a_j \in A \wedge a_j > \alpha$$

A utilização deste método para a seleção de k permite que a rota com o maior valor de feromônios depositados tenha prioridade de escolha entre as demais rotas. Esta escolha, aliada a forma como o valor dos feromônios é composto (vide equações (4.1) e (4.2)) favorece a escolha dos nós com maior quantidade de energia restante e menor número de rotas já ativas passando por eles.

4.2.4 Envio de uma Backward Ant

Ao fim do timer no destino da rota desejada, a **bant** inicia sua jornada rumo ao ponto que originou a descoberta de rotas. O nó deve, baseado nas informações armazenadas nas tabelas de rotas inversas, decidir probabilisticamente o próximo passo da **bant**. O mecanismo responsável por decidir qual rota a **bant** deve seguir será descrito na seção 4.2.3.

Após selecionar qual a rota inversa desejada, o nó adiciona uma entrada em sua tabela de rotas efetivas referente a direção do *Destino* à *Origem* dos dados. Esta rota é referente a uma das direções da rota bidirecional que será formada.

- **Destino**: O identificador do nó que consta como *Destino* da **bant**.
- **Próximo**: O identificador do próximo salto da rota inversa selecionada.
- **TTL**: Recebe um valor configurável referente ao tempo que uma rota deve ser mantida ativa, ainda que não utilizada.

Uma vez que a rota efetiva tenha sido adicionada, o nó aramazena seu identificador no campo **último** da **bant** e, em seguida, a envia por unicast ao vizinho fornecido como próximo salto na rota inversa selecionada anteriormente.

O processo de envio de uma **bant** é o mesmo para todos os nós intermediários da rota a ser estabelecida.

4.2.5 Recebimento de uma Backward Ant

Ao receber uma **bant** o nó adiciona em sua tabela de rotas efetivas uma entrada referente ao caminho que será estabelecido partindo do nó que deu origem a descoberta de rotas em direção ao nó destino desejado. A rota criada recebe os seguintes dados:

- **Destino**: O identificador do nó que consta como *Origem* da **bant**.
- **Próximo**: O identificador do nó que enviou a **bant** ao nó atual. Valor obtido através do campo **último** da formiga.
- **TTL**: Recebe o valor de ttl conforme explicado na descrição da rota formada na seção 4.2.4.

Em um nó intermediário a **bant** é encaminhada ao próximo salto conforme descrito na seção 4.2.4.

A figura a seguir ilustra o caminho de uma bant até o no origem:

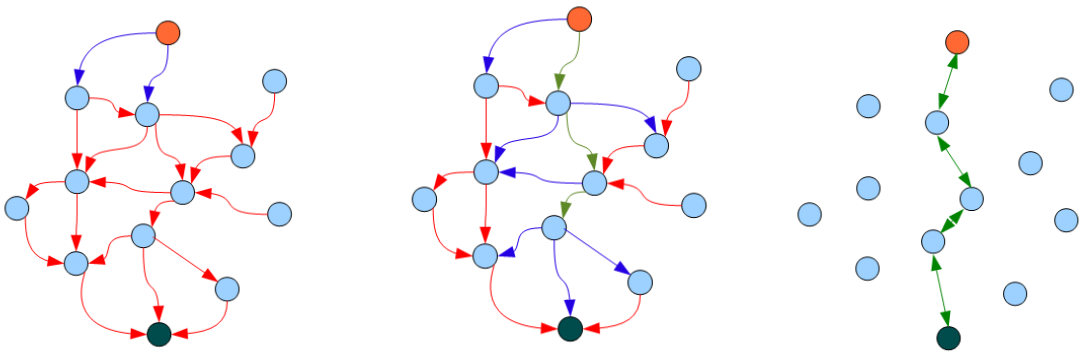

Figura 4.4: Caminho de uma bant. As flechas azuis representam as rotas possíveis de serem escolhidas em cada nó, as verdes representam aquelas que foram efetivamente escolhidas pela bant. Mais a esquerda, temos a rota efetiva estabelecida ao fim do processo.

A chegada ao nó apontado como destino da **bant**, ou seja, o nó que iniciou o processo de descoberta de rotas, marca o fim deste processo. Neste ponto o nó *Origem* possui uma rota ativa para o envio de dados ao destino desejado, assim como o *Destino* possui uma rota passando pelos mesmos nós intermediários até o nó *Origem*. A necessidade desta rota será explica na seção *Manutenção de Rotas*.

4.3 Manutenção de Rotas

4.3.1 Rotas Inversas

Rotas inversas são abstrações das trilhas de feromônios deixadas pelas formigas enquanto buscam seu "alimento", ou seja, o nó destino de uma transmissão de dados. Dessa forma, elas se mantém ativas enquanto houver feromônios depositados nela. No entanto, os feromônios possuem um fator de evaporação, o que faz com que seu valor diminua em uma trilha com o passar do tempo. Para simular este fenômeno, a cada determinado período, todas as rotas inversas tem seus valores de feromônios atualizados pela seguinte atribuição:

$$\tau_i \leftarrow (1-\rho)\tau_i$$

Sendo ρ o fator de evaporação dos feromônios que também pode ser um valor configurável no intervalo $[0,1]$.

4.3.2 Rotas Efetivas

Uma vez estabelecidas as rotas, os mecanismos responsáveis por buscar a distribuição eficiente do consumo de energia atuam de forma pró ativa utilizando informações locais, evitando a necessidade de troca de informações entre os nós e, consequentemente, mantendo baixo o overhead de pacotes de controle. O alicerce do processo de manutenção de rotas foi baseado no mecanismo utilizado pelo algoritmo AODVjr [7].

O campo *TTL* de todas as rotas efetivas é decrementado periodicamente. Assim que este campo é zerado, a rota referente a ele é considerada inválida e retirada da tabela de rotas.

As rotas em direção a origem são mantidas através dos pacotes de dados enviados por ela. Cada vez que um nó *a* recebe um pacode de dados de um nó *b*, o nó *a* atualiza o campo *TTL* de todas as rotas cujó campo *Próximo* corresponde ao nó *b*, atribuindo a este, seu valor inicial. Estas rotas são utilizadas pelo destino para o envio de pacotes de verificação de conexão entre origem e destino. Chamaremos estes pacotes de **Connect**.

No sentido do destino para origem, as rotas são mantidas através do recebimento de **Connects**. Ao receber um destes pacotes o nó realiza o mesmo procedimento do caso dos pacotes de dados, a diferença entre os dois surge no momento de enviar o pacote ao próximo destino da rota. O encaminhamento de um **Connect** pode ser negado para que a rota para o destino seja quebrada, forçando um novo processo de descoberta e, consequentemente, a possibilidade do

estabelecimento de uma nova rota com maior quantidade de energia restante.

O funcionamento básico deste sistema é mostrado na figura a seguir:

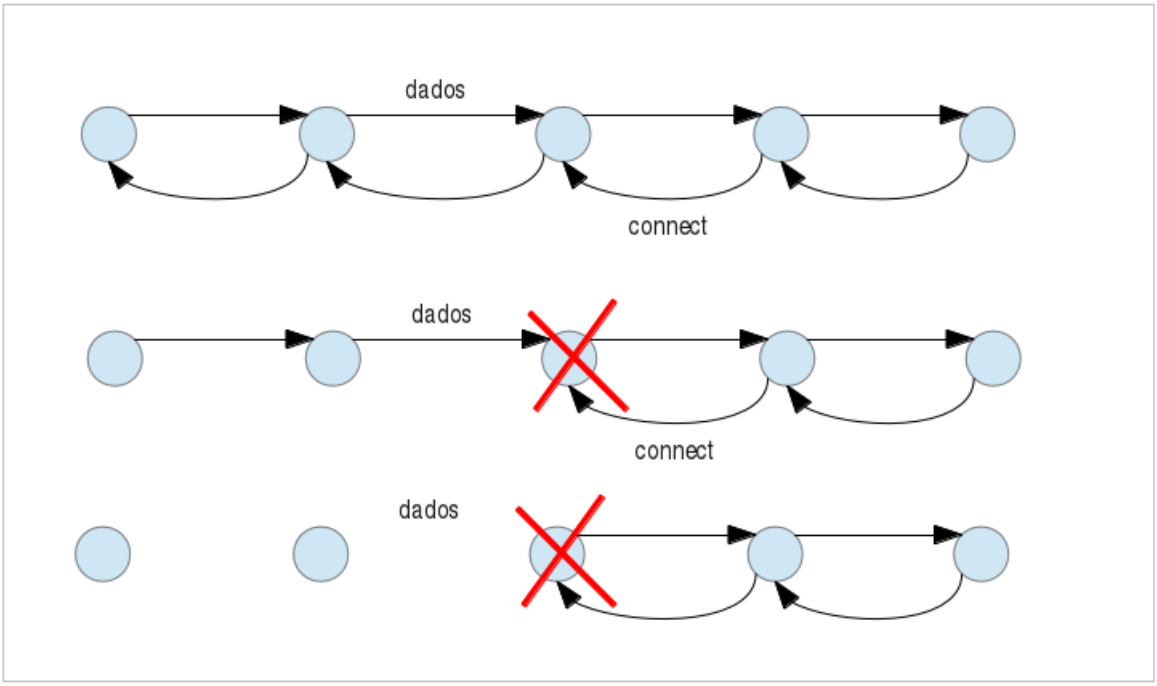

Figura 4.5: Manutenção de rotas efetivas do SACRA.

Cada nó possui um segundo *threshold* configurável que especifica o quanto de energia este nó está "autorizado" a consumir a partir do último momento em que for estabelecida uma rota passando por ele. Para que esta verificação seja possível, o nó armazena o estado atual de sua bateria sempre que uma **bant** é recebida. Assim, quando um pacote **Connect** é recebido, o nó o processa devidamente e somente o encaminha ao próximo destino caso seja satisfeita a condição:

$$E_{\text{checagem}} - E_{\text{atual}} < threshold$$

Ou seja, caso o consumo de energia desde a última checagem até o momento atual seja maior que o treshold definido, o nó deixa de encaminhar os *connects* pois ele já consumiu uma boa quantidade de energia servindo esta ou mais rotas. Visto que são estes pacotes que atualizam o *ttl* das rotas utilizadas pelos pacotes de dados e que o nó *Origem* deixará de recebê-los, logo a rota para o nó destino será invalidada e retirada da tabela de rotas da origem provocando uma redescoberta caso este ainda deseje enviar dados ao destino.

Este mecanismo trabalhando junto ao processo de descoberta de rotas, é o principal res-

ponsável pela distribuição do consumo de energia pelos nós da rede uma vez que ele promove a renovação periódica das rotas, permitindo assim, que *ant colony optimization* entre em ação selecionando os melhores nós.

A modificação do valor atribuído a este *threshold*, bem como ao aplicado na negação do processamento de uma **fant** (ver seção 4.2.2), podem modificar significamente o comportamento do algoritmo. Caso configurado com um valor baixo para o *threshold* da checagem de energia, o algoritmo provocará mais descobertas de rotas, consumindo mais energia da rede como um todo, porém distribuindo melhor este consumo entre os nós. Por outro lado, se este valor for alto, serão provocadas menos descobertas, porém, um nó poderá ter grande parte de sua energia consumida enquanto serve a uma rota, o que prejudica a distribuição do consumo de energia. Um dos desafios da configuração deste algoritmo é buscar o melhor equilíbrio possível entre estes parâmetros.

Vale ressaltar que esta negação de encaminhamento de mensagens é realizada apenas com os pacotes *connect*, mantendo a entrega de dados até a exaustão do ttl da rota que corre em sentido oposto a dos *connects*. Procedendo desta maneira, busca-se que o impacto deste procedimento na taxa de entrega de dados seja o mínimo possível.

É possível perceber que este mecanismo entrará em ação apenas caso a comunicação seja unidirecional, ou seja, apenas uma das pontas da rota envie dados a outra. Não pode-se negar o encaminhamento de pacotes de dados pois isto geraria um impacto negativo na taxa de entrega de pacotes o que não é desejado. Sendo assim, ambas direções terão seus valores de *ttl* atualizados enquanto durar a troca bidirecional de dados.

Para que não sejam enviados pacotes *connect* desnecessários, cada nó guarda uma lista dos identificadores para os quais eles enviam dados. Assim os *connect* são enviados apenas para os destinos da tabela de rotas que não estão presentes nesta lista.

5 *Resultados de Simulações*

Este capítulo mostra os resultados obtidos com o **SACRA** em simulações realizadas em dois simuladores de redes, o *Sinalgo* e o *Omnet++*. Cada simulador foi utilizado em uma fase distinta do desenvolvimento do algoritmo. A primeira fase foi marcada pela necessidade de construir um protótipo que fornecesse facilidade para obeservação do comportamento do algoritmo. Neste contexto o *Sinalgo* serviu perfeitamente com suas ferramentas gráficas altamente intuitivas. A segunda fase, por sua vez, trouxe o amadurecimento do algoritmo. Um trabalho realizado já com bases mais sólidas devido ao conhecimento em **RSSF**'s obtido na primeira fase. Assim, para a validação dos resultados pretendidos com o **SACRA**, fez-se necessária a utilização de um ambiente mais robusto para testes, o qual foi fornecido pelo *Omnet++*.

5.1 Primeira Fase - Sinalgo

O *Sinalgo* é um poderoso framework de simulação para testes e validações de algoritmos de rede [12]. Escrito em java, este simulador permite a prototipação de algoritmos nesta linguagem, fornecendo uma interface gráfica amigável para verificação visual do comportamento do algoritmo.

O tempo no *Sinalgo* é medido em *rounds* de simulação. Este simulador não fornece qualquer abstração de um sistema energético e também não leva em consideração aspectos como as colisões de pacotes em uma rede, porém, a possibilidade de acompanhar visualmente e de forma extremamente intuitiva a troca de dados entre os nós, faz deste um ótimo simulador para iniciar o desenvolvimento de um algoritmo de roteamento.

Para as simulações foi configurado um ambiente em duas dimensões de 500x500 metros. A aplicação considerada consiste em uma estação base recebendo dados de outros 9 nós da rede, totalizando assim, 10 nós comunicantes distribuídos de forma que o nó base estivesse no centro do ambiente e os 9 nós *origem* formassem um "círculo" ao redor dele. Os demais nós da redes foram uniformemente distribuídos formando uma grade. O algoritmo utilizado para a

comparação de resultados foi o **AODVjr**.

O envio de dados por parte dos 9 nós é realizado a cada 15 rounds de simulação enquanto o nó base envia um *connect* a cada 90 rounds. O ttl foi configurado para 120 rounds.

As figuras a seguir mostram 3 cenários distintos de uma simulação do algoritmo proposto: A figura 5.1 temos a disposição dos nós antes do início da simulação, a figura 5.2 mostra as rotas inversas deixadas por **fants** logo após um processo de descoberta de rotas e a figura 5.3 as rotas efetivas formadas e a troca de dados ocorrendo [1].

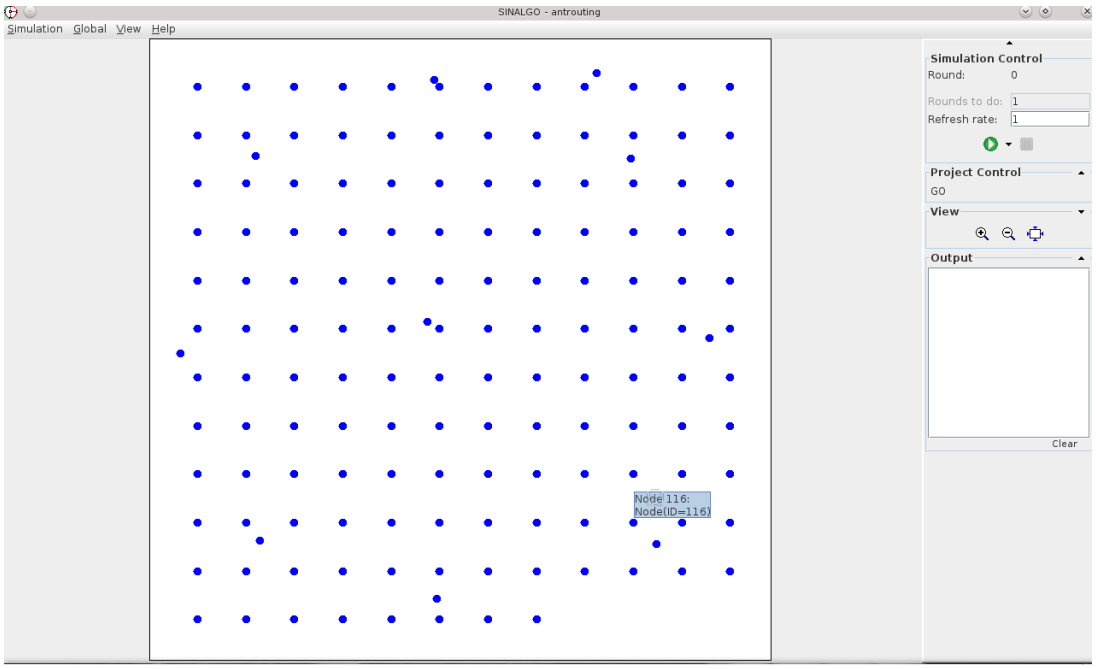

Figura 5.1: Disposição dos nós para o inicio das simulações. Os nós desalinhados da grade são aqueles que realizam comunicação.

[1] Foram realizadas apenas simulações de ambientes estáticos nesta fase.

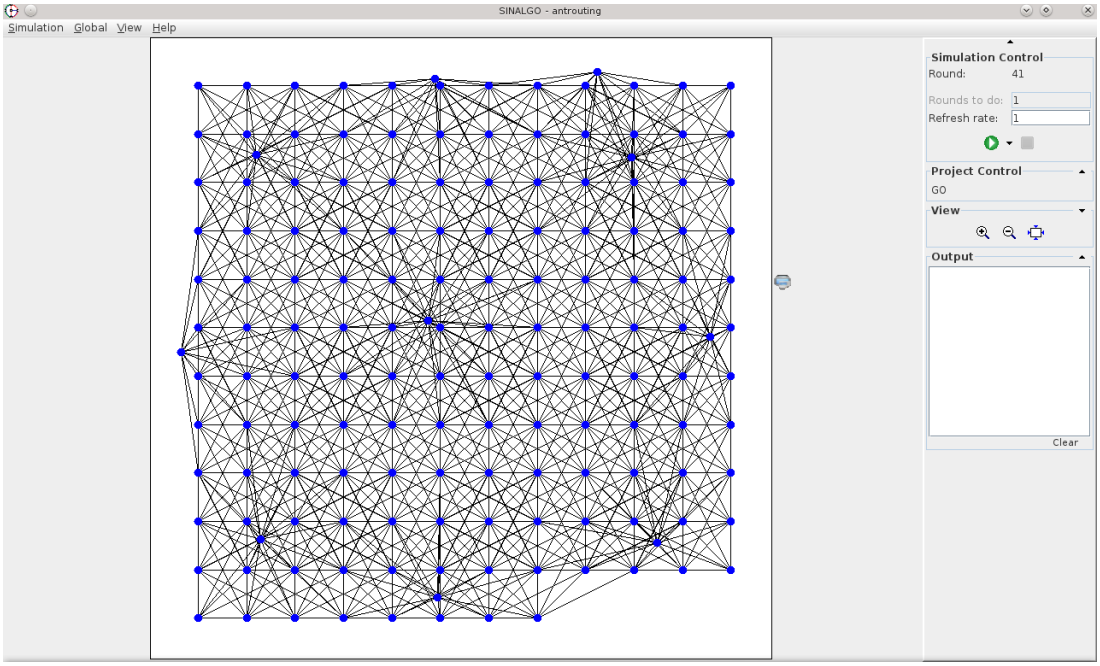

Figura 5.2: Os 9 nós comunicantes já receberam bants e acabam de estabelecer suas rotas para o destino. A imagem ilustra o grande número de rotas inversas presentes, demostrando a grande quantidade de opções para o estabelecimento de rotas efetivas.

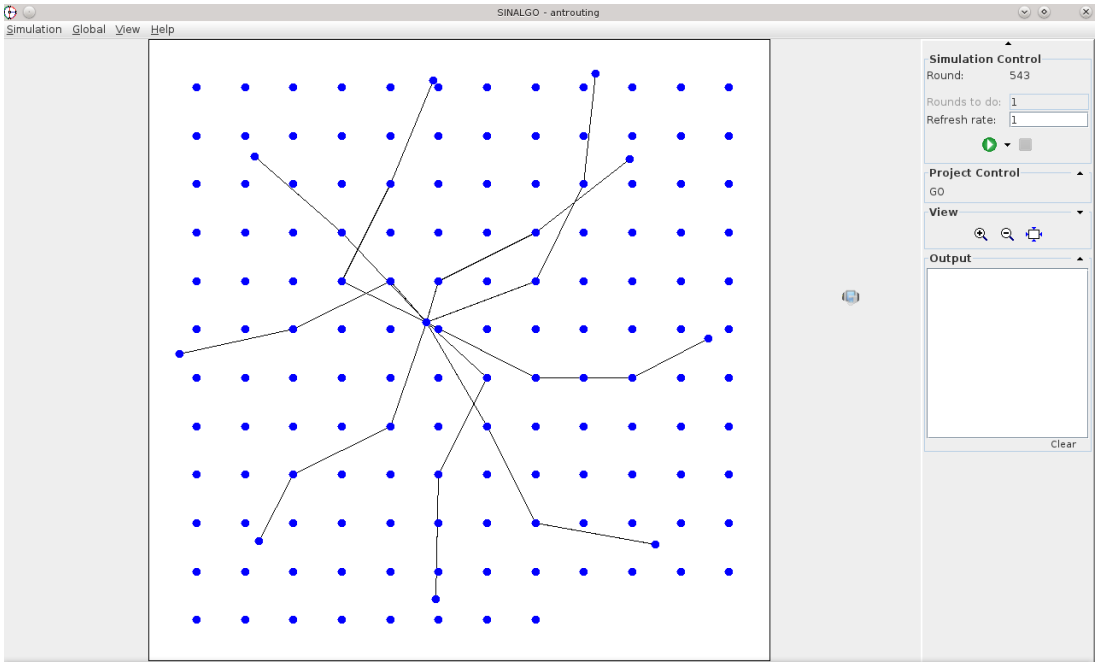

Figura 5.3: Após a evaporação completa das rotas inversas (trilhas de feromônios), restam as rotas efetivas entre os nós comunicantes, estas mantidas por um mecanismo de timeout (time to live) e dinamicamente modificadas através dos mecanismos anteriormente descritos.

Para simular o consumo de energia, foi atribuído a cada nó um valor de energia inicial de-

crementado a cada mensagem recebida ou enviada. Este mesmo valor foi utilizado nas equações onde se fazia necessário informações a respeito da energia restante de um nó. Para os pacotes de dados, foi sorteado um número aleatório que determina o seu tamanho, valor este, que foi utilizado para determinar a quantidade de bateria a ser consumida para este tipo de pacote. Para uma comparação justa foram utilizados os mesmos valores de consumo por mensagem na implementação do **AODVjr** e do **SACRA**.

Um dos aspectos mais importantes a se considerar em uma rede de sensores sem fio é o "tempo de vida" da rede, definido em [8] como o tempo decorrido até que um primeiro nó tenha sua bateria completamente consumida. Este aspecto foi medido em rounds do simulador para diferentes números de nós e foram obtidos os resultados mostrados na figura 5.4.

Como esperado, quanto maior o número de nós da rede, maior a eficiência do **SACRA** em relação ao **AODVjr**. Isto ocorre pois o aumento do número de nós em um mesmo espaço implica em um maior número de rotas inversas formadas na primeira fase da descoberta. Assim uma **bant** possui mais opções de rotas a seguir, melhorando a distribuição do consumo de energia entre os nós.

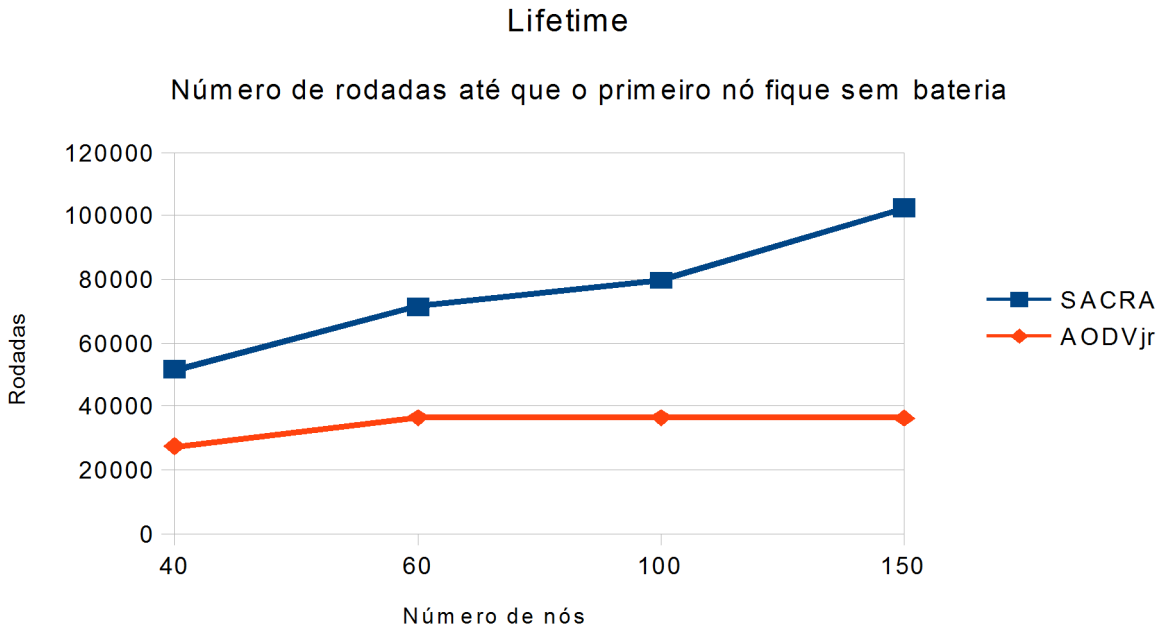

Figura 5.4: Tempo de vida

Outros dois aspectos verificados nas simulações foram a média global de energia restante na rede e o desvio padrão da quantidade de energia dos nós em relação a média. Este é particularmente importante por ter relação com a variância da quantidade de energia entre os nós

da rede, ou seja, é uma medida que reflete a distribuição do consumo em relação a média. As figuras 5.5 e 5.6 ilustram os resultados obtidos.

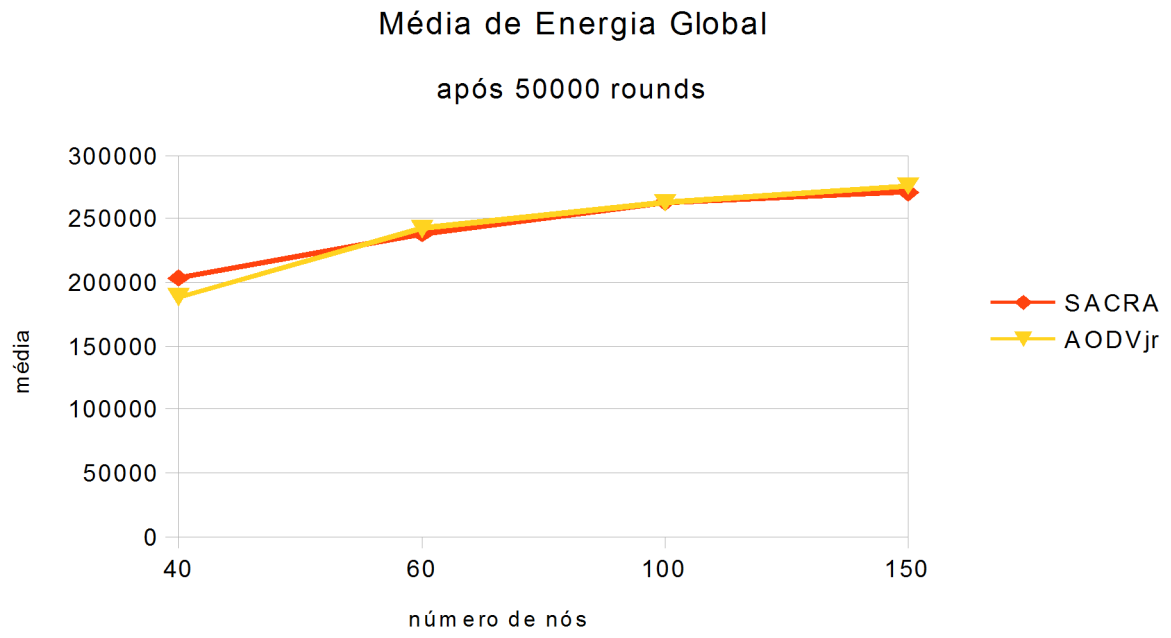

Figura 5.5: Média de energia da rede após 50000 rounds de simulação

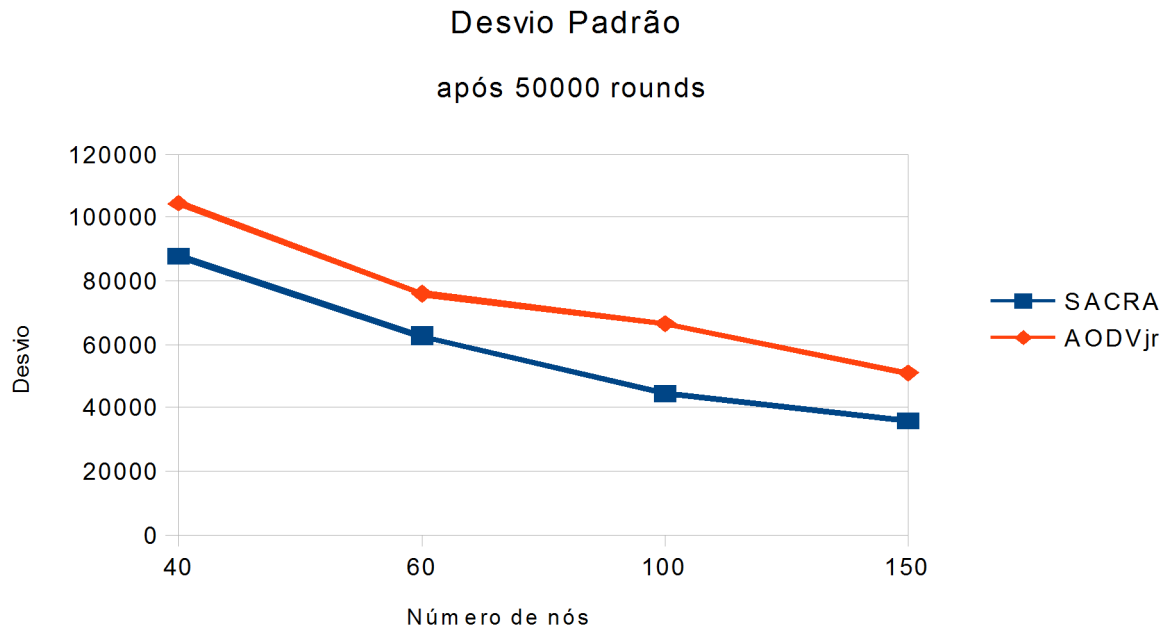

Figura 5.6: Desvio padrão da energia dos nós após 50000 rounds de simulação

Quanto menor o desvio padrão apresentado, melhor a distribuição da energia entre os nós da rede. Dado que o *AODVjr*, em um cenário onde um nó só falha caso sua bateria seja esgotada,

vai manter este nó em uma rota até que sua bateria seja completamente consumida, a distribuição da energia tende a ser bastante prejudicada. Esse fato foi comprovado na comparação entre os algoritmos, onde o **SACRA** apresentou uma distribuição melhor para diferentes quantidades de nós compondo a rede.

5.2 Segunda Fase - Omnet++

A segunda fase de simulações foi realizada no simulador omnet++. Escrito em *C++*, ele permite a implementação de algoritmos nesta linguagem. Foi utilizado o framework "inet" implementado para o simulador. Este framework fornece todo o necessário para realizar simulações de redes de sensores sem fio.

Nesta fase do desenvolvimento, as simulações foram realizadas em um cenário muito mais realista a fim de verificar o comportamento do algoritmo quando sujeito a todas as variáveis de uma RSSF real como colisões de pacotes e mobilidade dos nós. Todo os parâmetros de energia do framework foram configurados para que o consumo representasse de forma adequada o consumo de um EPOSMote[13], um nó sensor real desenvolvido no LISHA/UFSC.

Neste simulador, fatores presentes em redes reais são simulados com grandeza de detalhes. É possível simular ambientes críticos, com grande taxa de perda de pacotes. Este tipo de cenário é interessante para verificar a eficiência do algoritmo simulado em situações extremas.

Para comparações, foram utilizados os algoritmos **AODV**, **AOER** e **ADHOP**. Nesta fase, também foram analisados os resultados de média global de consumo de energia e desvio padrão da energia dos nós. A novidade fica por conta da análise da taxa de entrega de pacotes presente nesta fase.

5.2.1 Redes Móveis

Para as simulações de redes móveis foi configurada uma área de 1000m x 1000m e a posição inicial dos nós foi determinada aleatoriamente. A comunicação é realizada por meio de 10 nós enviando dados para outros 10 a cada 4 segundos. O tempo de simulação neste caso foi de 1500 segundos e o tempo de espera para o envio de uma bant no nó destino foi de 100ms. Para as rotas efetivas, foi configurado um TTL de 10 segundos e os pacotes connect configurados para serem enviados a cada 6 segundos.

O esquema de mobilidade utilizado foi o *"MassMobility"*. Neste modelo um nó se movimenta em linha reta durante um determinado tempo escolhido aleatoriamente segundo uma

mobilityType	"MassMobility"
changeInterval	truncnormal(7s, 5s)
changeAngleBy	normal(0deg, 15deg)
speed	truncnormal(5mps, 1mps)
waitTime	7s
updateInterval	100ms

Tabela 5.1: Parâmetros de configuração do esquema de mobilidade

distribuição normal e, ao fim deste período, ele muda de direçao e segue seu movimento retilíneo. A alteração no sentido do movimento também é determinada randomicamente seguindo uma distribuição normal. Os parâmetros utilizados para o modelo de mobilidade são apresentados na tabela 5.1:

Os dados obtidos para a taxa de entrega de dados neste cenário de grande mobilidade podem ser visualizados na figura 5.7:

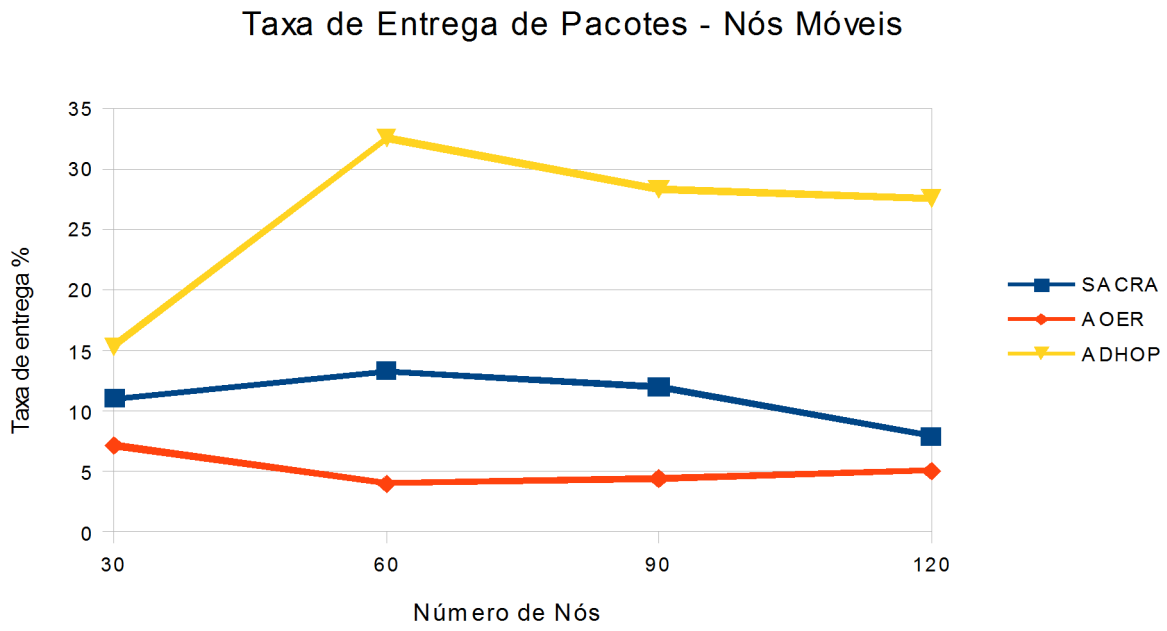

Figura 5.7: Taxa De Entrega de Pacotes - Mobilidade

O cenário criado para as simulações apresenta grande mobilidade e uma taxa de envio de pacotes relativamente alta. Dessa forma as baixas taxas de entrega de pacotes apresentadas pelo **SACRA** e o **AOER** podem ser justificadas, em parte, pela grande quantidade de rotas quebradas devido a mudança constante de posição dos nós. As taxas superiores apresentadas pelo **ADHOP** são alcançadas por sua mudança no paradigma tradicional de manter rotas pré definidas, adaptando-se com mais eficiência ao deparar-se com uma falta de rotas conhecidas

para um destino. Além disso, as formigas exploratórias do **ADHOP** carregam pacotes de dados para serem entregues. Este comportamento garante maior probabilidade de um pacote ser entregue, porém, tem um impacto negativo no consumo de energia, uma vez que estes pacotes tendem a ser retransmitidos e multiplicados diversas vezes enquanto procuram um caminho para o destino. Este impacto pode ser conferido na figura 5.8.

Outro fator agravante para a taxa de entrega de dados em um cenário de grande mobilidade é a utilização, na camada **MAC**, do padrão **802.15.4**. Este padrão não foi necessariamente projetado para obter um comportamento ótimo em redes com nós móveis. A presença de enlaces assimétricos aumentam com a mobilidade, o que traz sérios problemas para o sucesso do estebelecimento de rotas. Além disso, o aumento da densidade de uma rede, aumenta a incidência de nós escondidos, implicando em um maior número de colisões de pacotes e, consequentemente, a diminuição nas taxas de entrega de pacotes[14][15].

Os dados obtidos para a média de consumo global de energia para este cenário podem ser visualizados na figura 5.8.

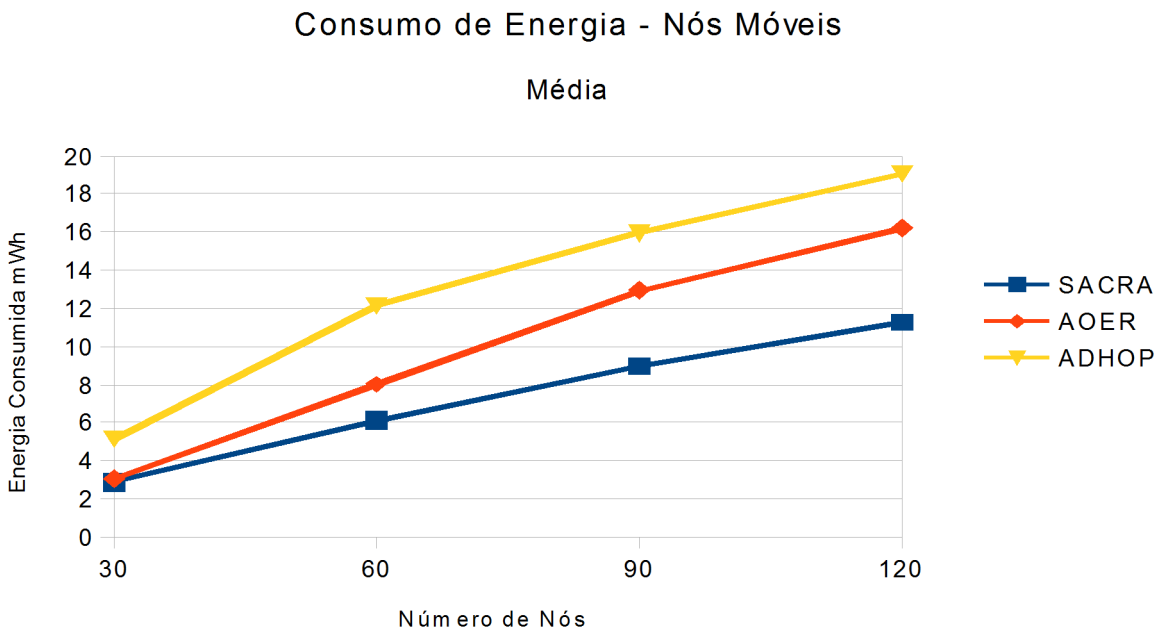

Figura 5.8: Média Global de Consumo - Mobilidade

É possível perceber que o **SACRA** manteve as melhores médias de consumo em relação aos demais algoritmos. Além disso, a tendência ao aumento do consumo com o aumento da densidade da rede, apesar de apresentado por todos os algoritmos testados, é menor no **SACRA**. Visto que, quanto maior o número de nós da rede, maior o número de pacotes de requisição

de rotas que serão retransmitidos, este resultado nos dá indícios de que o algoritmo proposto consegue consumir menos energia que os demais durante a realização dos *floods* de descoberta de rotas. Os pacotes *fant* (requisição de rota) do **SACRA** foram concebidos para carregar o mínimo de dados possível para, além do estabelecimento de rotas, aplicar *ACO* sobre o consumo de energia. Levando em consideração que a quantidade destes pacotes trafegando em uma rede durante esta fase constitui uma enorme fatia de tudo o que trafega em uma rede, esta decisão mostrou sua eficiência com o resultado acima.

Finalmente, o desvio padrão do consumo de energia dos nós pode ser visualizado no gráfico a seguir:

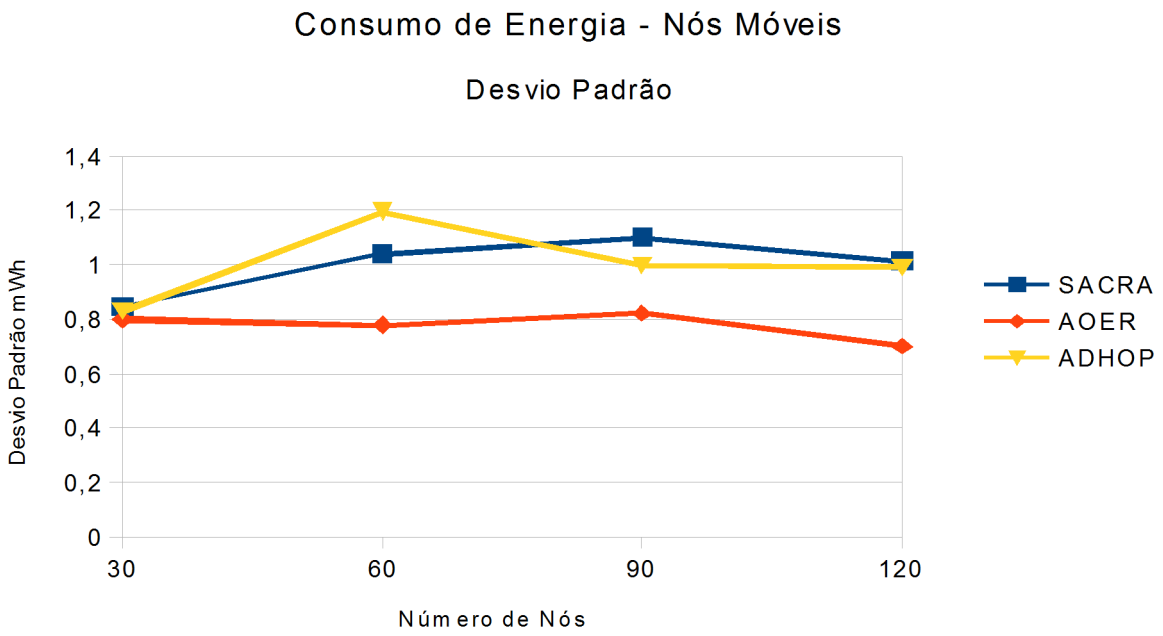

Figura 5.9: Desvio Padrão do Consumo de Energia - Mobilidade

Os baixos valores para o desvio padrão obtidos pelo **AOER** podem ser justificadas pela baixa entrega de pacotes. O baixo desempenho em obter sucesso na entrega de um pacote, indica que este algoritmo gerou uma grande quantidade de requisições de rotas, que por serem realizadas por *flooding* de pacotes, fazem com que todos os nós da rede consumam quantidades parecidas de energia, diminuindo assim, a discrepância entre os valores de consumo de energia dos nós. Sendo assim, os resultados obtidos pelo **SACRA** para a distribuição do consumo de energia, podem ser considerados positivos, uma vez que o algoritmo obteve valores bastante próximos daqueles registrados pelo **ADHOP** e conseguiu taxas de entrega significativamente maiores que o **AOER**.

5.2.2 Redes Estáticas

Para a simulação de redes estáticas, foi configurada uma área de 1000mx1000m. Os nós foram distribuídos de forma aleatória por esta área. Para a comunicação, foram escolhidos 10 nós da rede incumbidos de enviar dados para outros 10. Os nós "origem" enviam um pacote de dados a cada 3 segundos para seus "destinos". O tempo de simulação para todos os cenários foi de 2000 segundos e o tempo de espera para o envio de uma bant no nó destino foi de 100ms. Para as rotas efetivas, foi configurado um TTL de 10 segundos e os pacotes connect configurados para serem enviados a cada 8 segundos.

Para a taxa de entrega de pacotes, o **SACRA** apresentou resultados bastante positivos, ficando a frente dos demais algoritmos na grande maioria dos cenários simulados. Estes resultado podem ser observados na figura 5.10.

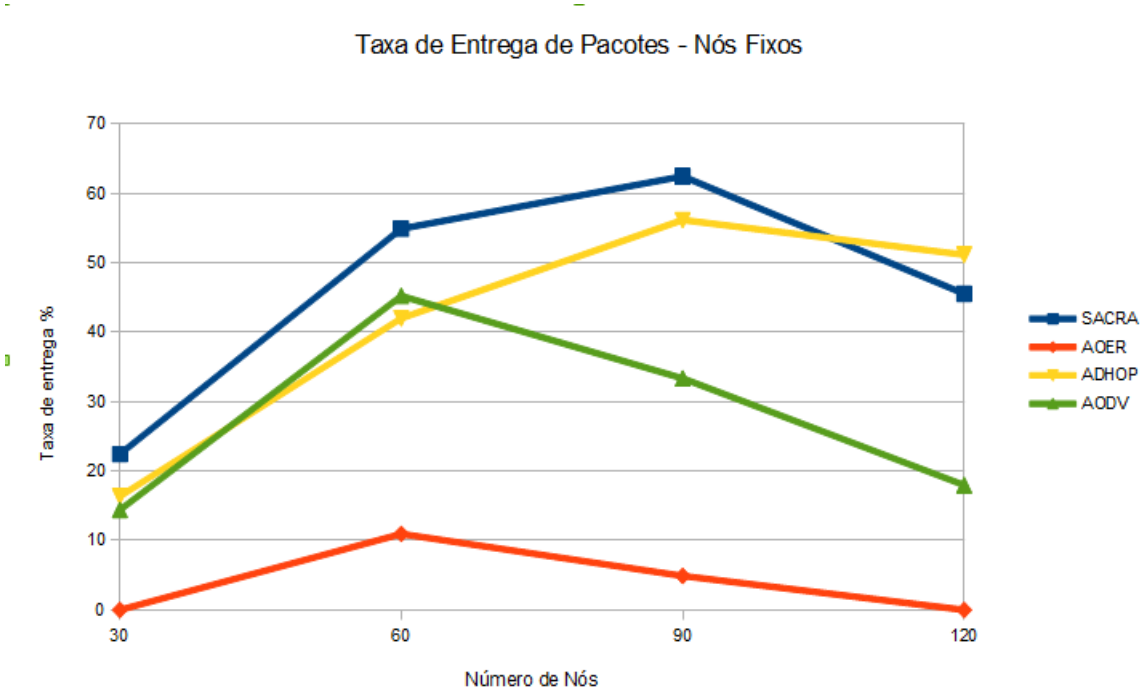

Figura 5.10: Taxa De Entrega de Pacotes - Nós Fixos

A queda de desempenho na taxa de entrega, apresentada no cenário com 120 nós na rede, pode ser justificada pelo aumento do overhead de pacotes de controle. O aumento da densidade da rede, contribui para o aumento de colisões de pacotes, provocando quebras de rotas e perda dos próprios pacotes envolvidos nas descobertas. Uma análise dos resultados obtidos para o overhead de pacotes de controle (ver figura 5.11), pode reforçar estas afirmações.

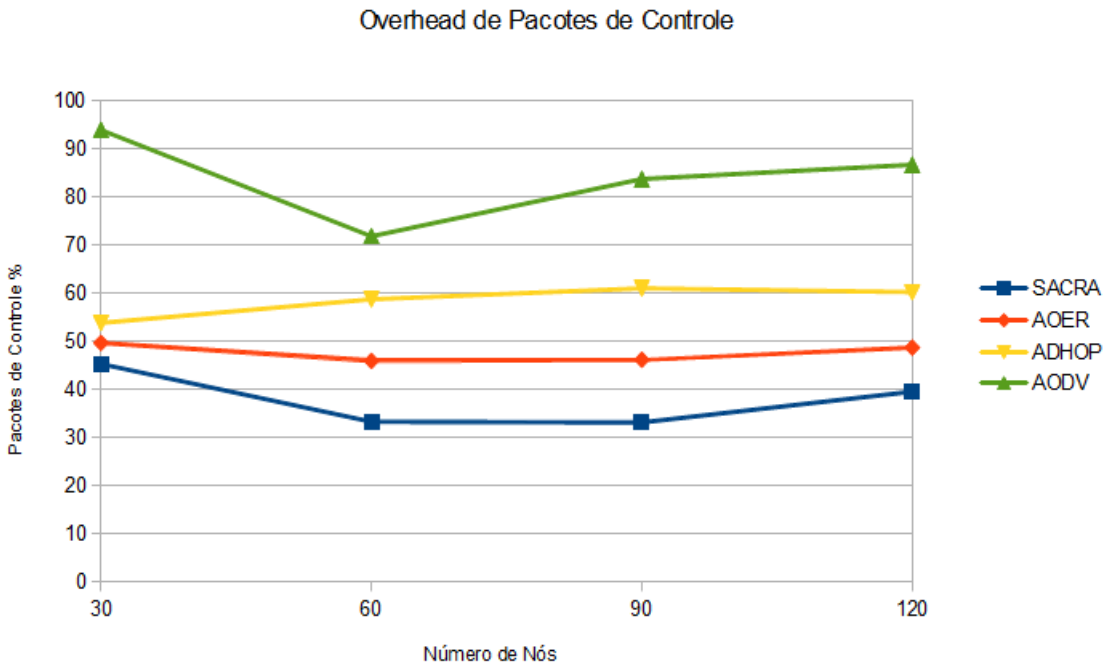

Figura 5.11: Overhead de Controle - Nós Fixos

A figura 5.12 mostra os resultados obtidos para a média de consumo global da rede:

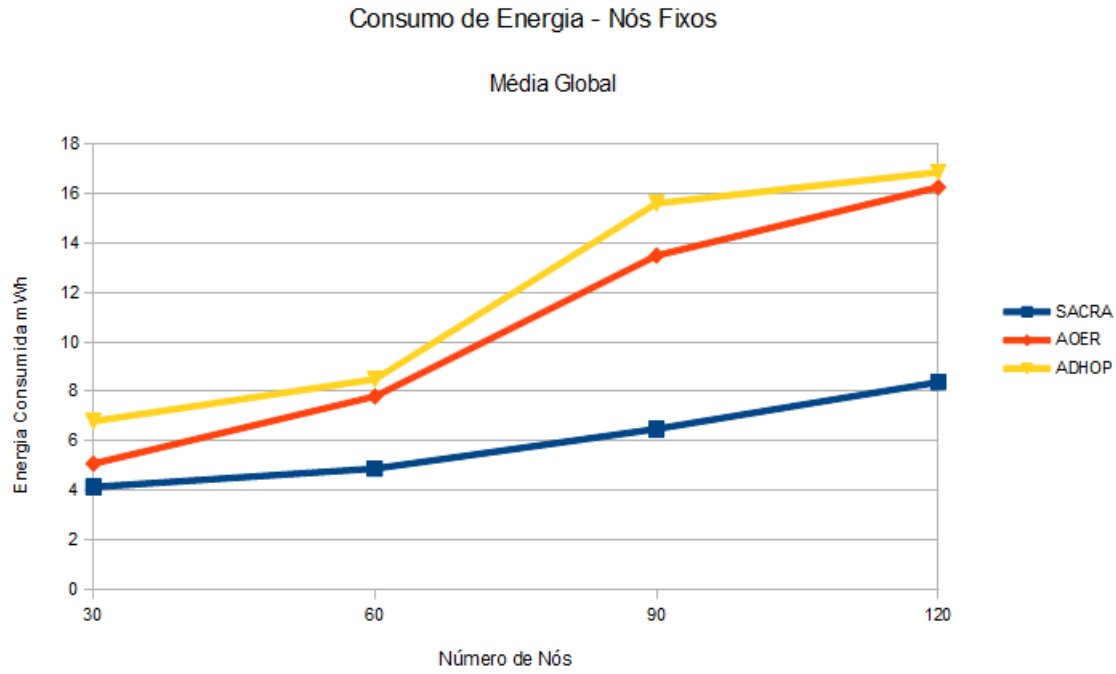

Figura 5.12: Média de Consumo Global - Nós Fixos

O **SACRA** mostra um ótimo desempenho para diferentes números de nós na rede. Além

disso, é possível verificar um consumo bastante controlado como o aumento da quantidade de nós, ao passo que, os demais algoritmos apresentaram uma tendência mais acentuada de aumento do consumo de energia nesta mesma situação. Este fato comprova a eficiência alcançada com a diminuição da quantidade de dados trafegados durante os *floods* de descoberta de rotas.

Os resultados obtidos para o desvio padrão são apresentados pela figura 5.13.

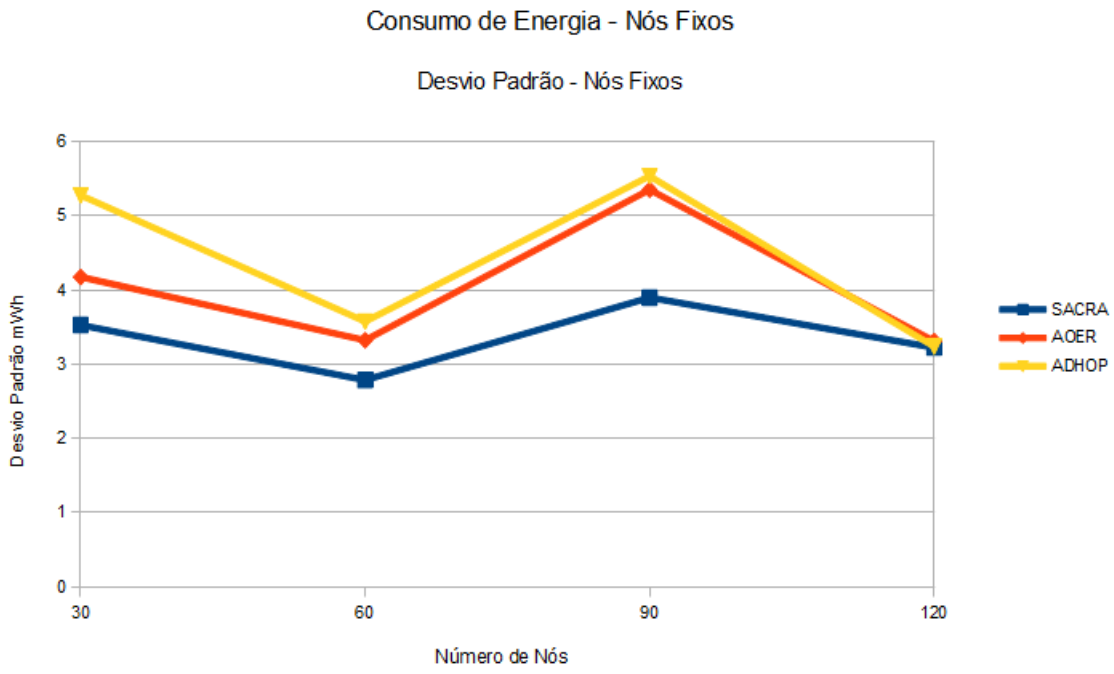

Figura 5.13: Desvio Padrão - Nós Fixos

Para esta medida de desempenho, o **SACRA** demonstrou uma taxa pouco pior do que os demais algoritmos analisados para alguns casos, o que indica uma menor eficiência em distribuir o consumo entre os nós da rede. No entanto, a pequena diferença mostra que o impacto não é suficiente para considerar esta distribuição ruim. Este fator pode ser compensado pelo excelente desempenho na média de consumo global.

Ainda na figura 5.13, pode-se perceber uma excessão ao comportamento geral do **SACRA**. Em simulações com 100 nós compondo a rede, o algoritmo se comportou melhor do que os demais algoritmos testados na distribuição do consumo de energia. Com este número de nós, o algoritmo também obteve uma ótima taxa de entrega de pacotes (ver figura 5.10). Estes dois fatos somados, nos permitem chegar a conclusão de que as rotas formadas ficaram bastante distribuídas, balanceando o consumo de energia e consequentemente reduzindo o número de colisões de pacotes, o que contribuiu positivamente para a taxa de entrega.

As taxas de entrega de dados apresentadas, aliado ao excelente consumo de energia, fazem do **SACRA**, uma ótima alternativa para aplicações de RSSF's estáticas onde o tempo de vida dos nós da rede constituem um fator crítico.

6 *Conclusões*

Neste trabalho foi apresentado o **SACRA**, um algoritmo de roteamento energeticamente efiente para redes de sensores sem fio. Para alcançar tal eficiência, foi aplicado a técnica de *ant colony optimization*, trabalhando junto a outros mecanismos intimamente ligados a quantidade de energia e número de rotas servidas por cada nó da rede. Todos os detalhes do algoritmo foram escolhidos de forma a apresentarem eficiência aliada a simplicidade de implementação.

Foram estudados diversos algoritmos para RSSF, principalmente aqueles considerados cientes de energia. Foram identificadas diversos aspectos interessantes em cada um. Alguns detalhes presentes no **SACRA** foram inspirados no funcionamento destes algoritmos conhecidos, o que mostra a grande importância deste levantamento para a realização do trabalho.

A prototipação do algoritmo foi realizada com o auxilio do simulador de redes **Sinalgo**, que mostrou ser uma ótima ferramenta para este tipo de atividade. O suporte visual fornecido pela ferramenta foi essencial para auxiliar na compreensão das particularidades envolvidas no desenvolvimento de um algortimo de roteamento. Durante a criação do protótipo foram definidos todos os mecanismos utilizados na composição do **SACRA**. O prótotipo ainda passou por uma validação, através de uma comparação com o algoritmo **AODVjr**. Foi utilizada uma abstração bastante simples para simular o consumo de energia com base nos pacotes de dados e roteamento enviados e recebidos. Neste cenário o **SACRA** apresentou resultados positivos para para a distribuição do consumo de energia, apesar de o consumo de energia em si ter se mostrado muito próximo ao obtido pelo **AODVjr**. A melhor distribuição do consumo energético no **SACRA** mostrou, ainda, seu impacto positivo no tempo de vida da rede. Estes resultados demonstraram que o protótipo construído atendeu as expectativas de maximizar o tempo de vida dos nós de uma RSSF.

A técnica *ant colony optimization* mostrou ser bastante eficiente na aplicação da heurística de energia presente no algoritmo, fornecendo um mecanismo poderoso capaz de ser implementado sem a necessidade de qualquer alteração no número de pacotes de controle planejado. Os dados referentes a aplicação da técnica, carregados por uma formiga de requisição de rotas no

SACRA, foram reduzidos a um único campo, minimizando o impacto da técnica na quantidade de dados efetivamente transmitidos.

A simplicidade alcançada no desenvolvimento do **SACRA**, foi posteriormente testada em um ambiente mais realista, através do simulador **Omnet++**. Com este simulador foi possível verificar o comportamento do algoritmo sujeito aos efeitos da mobilidade dos nós, colisões de pacotes e sua consequente perda, e os efeitos da camada *MAC* em algoritmos de roteamento. Para a validação do algortimo nesta fase, foram realizadas comparações com os algoritmos **AOER**, **ADHOP** e **AODV**. Os resultados obtidos para o consumo médio de energia foram extremamente positivos, alcançando a eficiência energética desejada.

As taxas de entrega de pacotes em ambientes de alta mobilidade mostraram-se satisfatórias se comparadas ao **AOER**. A comparação com o **ADHOP** neste cenário, mostram que o paradigma tradicional de manter rotas pré-definidas para a comunicação, utilizado no **SACRA**, mostrou não ser tão interessante quanto o roteamento baseado apenas no próximo salto, utilizado no **ADHOP**. Em ambientes com nós fixos, este quesito mostrou resultados excelentes.

A análise dos resultados obtidos com simulações e a comparação do algoritmo proposto com outros já conhecidos em redes de sensores sem fio, permitem concluir que o objetivo central de construir um algoritmo simples e capaz de obter eficiência energética no roteamento foi alcançado satisfatoriamente, transformando o **SACRA** em um algoritmo promissor para aplicações onde esta variável seja de grande importância. Como exemplo de uma tecnologia que normalmente faz uso de redes de sensores sem fio cuja eficiência energética seria grandemente valorizada, podemos citar os *smart buildings*, que possuem entre suas principais características, a busca pelo consumo eficiente de energia.

7 *Trabalhos Futuros*

Alguns aspectos analisados ao longo do desenvolvimento do **SACRA**, permitem identificar possíveis melhorias que contribuiriam ainda mais para a melhoria da taxa de entrega de pacotes e a redução do consumo de energia. Entre elas pode-se destacar a aplicação de algum mecanismo de controle para o *flood* de pacotes de requisição utilizado na fase de descoberta de rotas e a mudança da reação a quebras de rotas, fazendo com que estas possam ser reparadas ao invés totalmente redescobertas.

Permitir que um nó intermediário seja capaz de solicitar uma reparação de rotas, e que um nó que já possua uma rota para o destino desejado responda a uma requisição de rotas, faria com que o algoritmo pudesse reutilizar "partes" de rotas quebradas. Esta modificação pode ter impacto positivo na taxa de entrega de pacotes uma vez que a reconstrução de uma rota pode ser significativamente menos custoso para a rede, com relação a pacotes de controle, quando comparada a uma descoberta completa de rotas.

A respeito do controle de *flood*, a principal contribuição seria na redução do consumo de energia. Um *flood* massivo e sem controle de pacotes de requisição de rotas, torna-se proibitivo em redes de grande densidade uma vez que uma parcela importante da energia de todos os nós da rede é consumida durante este processo. Durante as pesquisas para o desenvolvimento do **SACRA** foram analisados alguns mecanismos para o controle do *flooding* na rede. Um deles mostrou-se particularmente promissor para a utilização junto com *ACO*. O nome da técnica é *percolation driven flood* e ela utiliza um mecanismo estocástico para a definição de quando um pacote de requisição de rotas deve, ou não, ser retransmitido. A métrica originalmente utilizada pela técnica, é baseada no número de vizinhos do nó que deve decidir a respeito da retransmissão do pacote[16]. Considerando que é possível utilizar outras métricas para determinar a propabilidade de um pacote ser retransmitido, seria possível adicionar questões referentes a energia neste calculo. O ponto negativo da especificação original da técnica é a adição de um pacote de controle para a determinação do número de vizinhos de um nó, porém, se a métrica for substituida por algo capaz de ser calculado localmente, este pacote tornaria-se desnecessário.

É necessário, ainda, algum trabalho com respeito a determinação de parâmetros ótimos para a configuração do algoritmo em cada cenário. Além disso, um estudo mais cuidadoso das equações e parâmetros envolvidos nos cálculos relacionados a feromônios, penalidade para número de rotas e *rank* de um nó. Simples modificações nestas equações podem modificar radicalmente o comportamento geral do algoritmo.

Referências Bibliográficas

1 WIKIPEDIA. *Swarm intelligence — Wikipedia, The Free Encyclopedia*. 2011. [Online; accessed 4-December-2011]. Disponível em: <http://en.wikipedia.org/w/index.php?title=Swarm_intelligence&oldid=464012605>.

2 PASSOS, R. M. *Gerenciamento Dinâmico de Energia em Redes de Sensores Sem Fio: Uma Abordagem Orientada à Aplicação*. Dissertação (Pós-graduação) — Universidade Federal de Minas Gerais, 2009.

3 THE Internet of Things. janeiro 2011. [Online; acessado em 18-maio-2012]. Disponível em: <http://oecdinsights.org/2012/01/31/the-internet-of-things/>.

4 GOSS, S. et al. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, Springer-verlag, n. 76, p. 579–581, 1989.

5 WIKIPEDIA. *Metaheuristic — Wikipedia, The Free Encyclopedia*. 2012. [Online; accessado em 15-Maio-2012]. Disponível em: <http://en.wikipedia.org/wiki/Metaheuristic>.

6 DORIGO, M.; BIRATARI, M.; STüTZLE, T. Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, IEEE, p. 28–38, November 2006.

7 CHAKERES, I. D.; KLEIN-BERNDT, L. Aodvjr, aodv simplified. *Mobile Computing and Communications Review*, v. 6, n. 3.

8 SHUANG, B.; LI, Z.; CHEN, J. An ant-based on-demand energy route protocol for ieee 802.15.4 mesh network. *Int J Wireless Inf Networks*, Springer, n. 16, p. 255–236, 2009.

9 PERKINS, C. E.; ROYER, E. M. The ad hoc on-demand distance vector protocol. *Charles E. Perkings, editor, Ad hoc Networking*, p. 173–219.

10 OKAZAKI, A. M.; FRöLICH, A. A. Ant-based dynamic hop optimization protocol: a routing algorithm for mobile wireless sensor networks. *Proceedings of the Joint Workshop of SCPA 2011 and SaCoNAS 2011 - IEEE GLOBECOM 2011*, p. 1179–1183, 2011.

11 OKAZAKI, A. M. *Algoritmo de Roteamento baseado em Colônia de Formigas com Heurísticas Configuráveis para Redes Sensores Sem Fio de Topologia Dinâmica*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2012.

12 SINALGO - Simulator for Network Algorithms. [Online; accessed 4-December-2011]. Disponível em: <http://disco.ethz.ch/projects/sinalgo/news.html>.

13 EPOSMOTE. [Online; accessed 11-December-2011]. Disponível em: <http://epos.lisha.ufsc.br/EPOSMote>.

14 DENARDIN, G. W. et al. Impacto das camadas inferiores para o desenvolvimento de protocolos de roteamento para redes de sensores sem fio em grande escala. *XVIII congresso Brasileiro de Automática*, Setembro 2010.

15 SILVA, L. fabiano da; BRANQUINHO, O. C.; ASSUMPçãO, R. M. da. Mobility impact on ieee 802.15.4 network through a simulation platform.

16 VAKULYA, G.; SIMON, G. Energy efficient percolation-driven flood routing for large-scale sensor networks. *Proceedings of International Multionference on Computer Science an Information Tecnology*, p. 877–883, 2008.

17 WANG, J. et al. Hopnet: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. *Ad Hoc Networks*, Elsevier, n. 7, p. 690–705, 2009.

18 WANG, Y.-C.; WU, F.-J.; TSENG, Y.-C. Mobility management algorithms and aplications for mobile sensor networks. *Wiley Wireless Communications and Mobile Computing*, p. 1–11.