UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL CARDOSO SOBEJEIRO

ANÁLISE DE DESEMPENHO DO PROTOCOLO *LEACH* COM ÊNFASE NA DIVISÃO DE GRUPOS E LOCALIZAÇÃO DOS SENSORES

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2020

RAFAEL CARDOSO SOBEJEIRO

ANÁLISE DE DESEMPENHO DO PROTOCOLO *LEACH* COM ÊNFASE NA DIVISÃO DE GRUPOS E LOCALIZAÇÃO DOS SENSORES

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina Trabalho de Conclusão de Curso 2, do curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Dr. Augusto Foronda.

PONTA GROSSA

2020

TERMO DE APROVAÇÃO

ANÁLISE DE DESEMPENHO DO PROTOCOLO *LEACH* COM ÊNFASE NA DIVISÃO DE GRUPOS E LOCALIZAÇÃO DOS SENSORES
por

RAFAEL CARDOSO SOBEJEIRO

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em vinte e dois de Setembro de 2020 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Augusto Foronda
Orientadora

Prof. Dr. Richard Duarte Ribeiro
Membro titular

Prof. MSc. Geraldo Ranthum
Membro titular

Prof. MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

Profa. Dra. Mauren Louise Sguario
Coordenador do curso

-O Termo de Aprovação assinado encontra-se arquivado na Secretaria Acadêmica -

Dedico este trabalho à minha mãe e meu pai (in memorian) pelo apoio ao longo da minha vida, à minha noiva que me apoiou e revisou meu trabalho, aos meus professores pelo conhecimento e atenção dentro e fora da sala de aula e aos amigos e colegas pelo suporte principalmente nos momentos de dúvida.

Agradeço ao meu orientador Prof. Dr. Augusto Foronda, pela sabedoria e paciência com que me guiou ao longo do trabalho.

Aos meus colegas de sala que auxiliam em dúvidas fora e dentro deste projeto.

A Secretaria do Curso, pela cooperação ao longo do meu permanecimento na instituição.

Aos demais professores do departamento acadêmico que compartilharam seus conhecimentos ao longo do curso.

Gostaria de agradecer muito à minha mãe, meu pai (in memoriam) e à minha noiva, pois sem o apoio deles seria muito difícil vencer esse desafio.

Enfim, a todos os que, por algum motivo, contribuíram para a realização desta pesquisa.

RESUMO

SOBEJEIRO, Rafael Cardoso. **Análise de DESEMPENHO do protocolo *LEACH* com ÊNFASE na divisão de grupos e localização dos sensores**. 2020. 50 f. Trabalho de Conclusão de Curso Bacharelado em Ciência da Computação - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2020.

As Redes de sensores sem fio (RSSF) são componentes tecnológicos essenciais para a sociedade modernizada. Sua função de receber e transmitir dados entre si por meio de ondas de rádio, não necessitando a utilização de cabeamento, possibilita uma transmissão constante, acesso e controle de informações em regiões que outras redes são inviáveis, como áreas remotas ou com grande tráfego de pessoas. Todavia, o consumo de energia é um dos principais problemas dos sensores sem fio; pelo fato de tratarem-se de pequenos e simples dispositivos que funcionam captando informações sem precisar de constante supervisão humana, e não possuírem a capacidade de recarga, sendo necessária a substituição de um sensor quando sua energia acaba, fazendo com que diminua o tempo de vida dos sensores e promovendo possíveis problemas nas transmissões de dados na rede. O protocolo *LEACH*, por sua vez, trata-se de um sistema para roteamento hierárquico o qual formam-se grupos de sensores, onde alguns deles são eleitos para desempenhar a função cabeça periodicamente, afim de minimizar o consumo de energia dos sensores em geral. O respectivo protocolo foi utilizado para simular a quantidade de dissipação de energia em dois ambientes, sendo eles, com parte sensores dispostos aleatoriamente e outro com sensores organizados estrategicamente. Os resultados obtidos demonstram que não só *LEACH*, mas também conhecer o alcance dos sensores e o ambiente o qual eles são dispostos auxilia no consumo de energia destes sensores.

Palavras-chave: Protocolo *LEACH*. Redes de Sensores em Fio. Consumo de energia. Tempo de vida dos sensores. Distribuição Uniforme. Distribuição Aleatória.

ABSTRACT

SOBEJEIRO, Rafael Cardoso. ***LEACH* protocol PERFORMANCE analysis with EMPHASIS in *cluster* division and sensor location.** 2020. 50 f. Work of Conclusion Course (Graduation in Computer Science) – Federal Technological University of Paraná. Ponta Grossa, 2020.

Wireless sensor networks (WSN) are essential technological components for modernized society. Its function of receiving and transmitting data to each other through radio waves, not requiring the use of cabling, enables constant transmission, access and control of information in regions where other networks are not viable, such as remote areas or with high traffic of people . However, energy consumption is one of the main problems with wireless sensors; due to the fact that they are small and simple devices that work by capturing information without needing constant human supervision, and do not have the ability to recharge, requiring the replacement of a sensor when its energy runs out, causing it to shorten sensor's lifetime and promoting possible problems in data transmissions on the network. The *LEACH* protocol, itself, is a system for hierarchical routing which groups of sensors are formed, where some of them are elected to perform the head function periodically, in order to minimize the energy consumption of the sensors in general. The respective protocol was used to simulate the amount of energy dissipation in two environments, which are, one with part sensors arranged at random and another with sensors strategically organized. The results obtained demonstrates that not only *LEACH*, but also knowing the range of the sensors and the environment in which they are arranged helps in the energy consumption of these sensors.

Keywords: *LEACH* Protocol. Wireless Sensor Network. Energy Consumption. Sensor's Lifetime. Uniform Distribution. Random Distribution.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

LEACH	*Low-Energy Adaptive Clustering Hierarchy*
LEACH-B	LEACH Balanced
LEACH-C	LEACH Centralized
RSSF	Redes de Sensores Sem Fio
V-*LEACH*	*Vice* LEACH

SUMÁRIO

1 INTRODUÇÃO

Desde sua primeira aparição em 1967, com a publicação da ARPANET[1], as redes de computadores têm evoluído ao longo dos anos, por uma parte pelo avanço tecnológico e por outra, para suprir as necessidades humanas (HAUBEN, 1998). Graças a isso, as Redes de Sensores Sem Fio (RSSF) puderam ser idealizadas e implementadas, contendo dispositivos miniaturizados e capazes de se comunicar sem necessidade de cabos (ZHAO; GUIBAS, 2005).

As RSSFs referem-se a um tipo de rede móvel *ad-hoc* que se forma a partir de vários dispositivos autônomos nomeados nós sensores. Um dos principais objetivos das RSSFs é o monitoramento e controle de ambientes a partir da coleta de dados por meio dos nós sensores que se comunicam diretamente. Por exemplo, para o plantio de milho em determinada área, as redes de sensores sem fio podem ser utilizadas para monitoramento do clima, umidade relativa do ar, radiação, dentre outros elementos climáticos que influenciam no crescimento do milho (RUIZ et al., 2004).

Embora as RSSFs mostrem-se como uma alternativa interessante, principalmente ao que se refere ao monitoramento de dados, este tipo de rede é bastante sensível às sua limitação de recursos. Os sensores possuem bateria limitada e são impossibilitados de recarga por se tratarem de equipamentos baratos e voltados ao descarte. Para isto, protocolos foram criados com o intuito de melhor administrar tais recursos (KAMARUDIN et al., 2016).

O protocolo *LEACH* (*Low-Energy Adaptive Cluster Hierarchy*) busca resolver a sensibilidade que a rede possui quanto ao consumo de energia, como apresentado por Heinzelman, Chandrakasan e Balakrishnam (2003). Dada sua importância para o pleno funcionamento de uma RSSF, é preciso investigar se este protocolo possui qualificação para aplicação prática através da análise dos seus pontos positivos e melhorias e na existência de tais, qual é seu efetivo impacto e se há espaço para melhora no protocolo.

[1] Rede de comutação de pacotes, além de ser a primeira rede a implementar os protocolos TCP/IP (*Transmission Control Protocol/Internet Protocol*).

1.1 OBJETIVO GERAL

Analisar o desempenho do protocolo *LEACH* com ênfase no agrupamento localização dos sensores.

1.2 OBJETIVOS ESPECÍFICOS

- Encontrar na bibliografia problemas que o *LEACH* possua e de que forma foram resolvidos para aprendizado;
- Simular o comportamento do protocolo *LEACH* em linguagem C a fim de demonstrar sua capacidade de economia de energia;
- Comparar ambientes de diferentes distribuições de sensores e analisar o impacto na capacidade energética da rede[2].

1.3 MOTIVAÇÃO E JUSTIFICATIVA

Como citado anteriormente, numa RSSF um dos principais fatores a ser considerado é o consumo de energia (EL KHEDIRI et al, 2014), pois os sensores não são recarregáveis[3], e assim, é necessário que sejam substituídos quando a sua energia acaba.

Portanto, é importante analisar protocolos que tratem esse problema de consumo, tanto para verificar se eles de fato resolvem o problema e quais exceções podem existir, bem como auxiliar em futuras elaborações que venham a considerar o uso destes protocolos.

[2] A capacidade energética da rede pode ser definida como quanto tempo essa mesma rede pode sobreviver sem que um de seus dispositivos, neste caso, os sensores, precise ser substituído devido à exaustão energética.
[3] Visando serem uma opção barata, os sensores não possuem forma de recarga.

1.4 ORGANIZAÇÃO DO TRABALHO

No capítulo 1, é descrito de forma introdutória o que será apresentado em seguida. No capítulo 2, são apresentadas as referências teóricas, descrevendo os elementos essenciais para o desenvolvimento do trabalho, em especial, o próprio protocolo *LEACH*.

O capítulo 3 ancora-se na pesquisa bibliográfica do protocolo *LEACH*. Através do levantamento e revisão de artigos, livros, dentre outros documentos, procura-se explorar a conceituação, desenvolvimento e problemas encontrados no protocolo *LEACH* por outros estudiosos da temática.

No capítulo 4, há o teste e apresentado o desempenho do protocolo *LEACH* em dois ambientes fechados fazendo uso da linguagem C, simulando cem sensores distribuídos pela rede. Os resultados destes testes, em seguida, serão comparados acerca da distribuição de sensores na rede e a quantidade de sensores que cada *header*[4] irá abranger.

[4] Nomenclatura em inglês usado para designar o sensor líder em um determinado grupo. Este conceito será melhor explorado no capítulo 2.

2 REFERENCIAL TEÓRICO

Antes de explorar o ambiente de testes do protocolo *LEACH* quanto aos problemas que este apresenta, é necessário introduzir os elementos essenciais da pesquisa, sendo assim, este capítulo descreve os conceitos teóricos necessários para a compreensão do trabalho desenvolvido.

2.1 REDES SEM FIO

Segundo FERREIRA (2013) *apud* FARIAS (2005), as redes sem fio ou *wireless* surgiram da necessidade de implementação de um método simples e seguro para a troca de informações em ambiente de combate. Com a evolução e modernização tecnológica, as redes sem fio deixaram de ser aparatos unicamente militares e passaram a ocupar as residências civis, institutos de ensino, estabelecimentos comerciais, dentre outros, como forma de priorizar a distribuição de rede, ao invés da utilização da versão cabeada.

A transferência de dados por meio de ondas de rádio sem a utilização de cabos tornou-se mais utilizada e cômoda por não ter necessidade de uma infraestrutura para funcionamento, além de possibilitar que os dispositivos conectados não necessariamente precisem estar próximos para comunicarem uns com os outros.

Para poder ocorrer a transmissão de dados, são necessários um transmissor (responsável pela origem dos dados), um canal (meio pelo qual os dados se movimentam) e um receptor (que é o destino dos dados). Para a utilização da rede, apenas são necessários os objetos que armazenam os dados, e estes são distribuídos para outros objetos através das ondas de rádio, sendo os sensores físicos tanto transmissores quanto receptores.

Existem diferentes tipos de redes sem fio, sendo alguns exemplos:

a. O Bluetooth, que é uma tecnologia sem fio de pequeno alcance, conectando normalmente pequenos dispositivos, como mouses, teclados, fones de ouvido etc.;

b. WWAN, que é conhecida como rede de longa distância sem fio, e utiliza o serviço do próprio celular para fornecer acesso à Internet para algum dispositivo, normalmente computadores;
c. WI-FI, que se refere a marca licenciada pela WI-FI Alliance[5], uma rede que permite, principalmente a mobilidade em um dado ambiente. Ela utiliza uma WLAN (rede de área local sem fio), a qual fornece acesso à Internet para um ambiente através da conexão com a operadora, normalmente por meio de cabeamento telefônico ou via fibra óptica.

2.2 REDES DE SENSORES SEM FIO

As RSSFs consistem de redes onde pequenos sensores conseguem processar dados e comunicarem-se entre si; e estão distribuídos em um dado ambiente com a finalidade de monitorá-lo (KAMARUDIN et al; 2016). Segundo LOUREIRO et al. (2003), “uma RSSF tende a ser autônoma e requer um alto grau de cooperação para executar as tarefas definidas para a rede.”

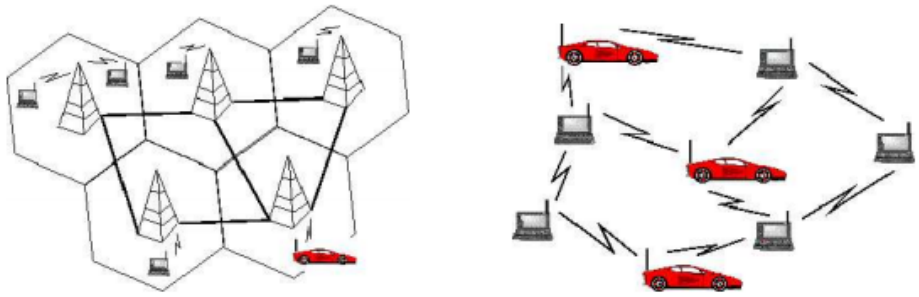
Este modelo de rede tem sido alvo de estudos devido sua aplicabilidade em diversos setores, dos quais pode-se destacar as áreas industrial, saúde, militar, agrícola e monitoramento ambiental (temperatura, pressão, etc) (OTHMAN; SHAZALI, 2012). Sua aceitação é devida à facilidade de inclusão de novos dispositivos na rede a qualquer momento, preço de implementação baixo e dispensar de uma infraestrutura fixa para fazer as configurações de rede (BATTACHARYYA; KIM; PAL, 2010).

Para seu melhor entendimento, as RSSF podem ser vistas como similares à rede móvel *ad hoc*, – que consistem em um conjunto de nós móveis que formam redes dinâmicas autônomas que trocam informações diretamente entre si, podendo ser utilizadas em qualquer tipo de infraestrutura, sem pontos de acesso (PINHEIRO, 2005) –, já que ambas não necessitam de uma infraestrutura para a troca de informações. Para fins ilustrativos, a Figura 1a refere-se as redes que necessitam de uma infraestrutura, enquanto a Figura 1b assemelha-se a uma rede móvel sem fio.

[5] https://www.wi-fi.org/

No entanto, ainda segundo LOUREIRO et al. (2003), a rede ad hoc tem como função básica prover suporte à comunicação entre os elementos computacionais que individualmente podem realizar diferentes atividades, enquanto os sensores das RSSF provêm dados para a execução de funções colaborativas.

Figura 1 - Tipos de rede sem fio de comunicação de dados



Fonte: LOUREIRO et al. (2003)

Contudo, as RSSF possuem alguns problemas devido ao *hardware*: a largura de banda de cada sensor é limitada e, consequentemente, a rede é mais lenta que a cabeada. Além do mais, é uma rede menos segura, pois *hackers* poderiam coletar os dados mais facilmente; é mais complexa de configurar que uma rede cabeada (BATTACHARYYA; KIM; PAL, 2010); e sua bateria não é recarregável (ZHAO; GUIBAS, 2005).

Como a energia se mostra um dos principais problemas de uma RSSF, surgiram protocolos que buscavam solucionar este problema, como o protocolo *LEACH* proposto por Heinzelman, Chankandrasan e Balakrishnan (2003).

Existem diversos protocolos de sensores sem fio, dentre eles, TAVARES (2002) cita

a. SMECN (*Small Minimum Energy Communication Network)*, que cria um sub-grafo da rede de sensores que contém a rota com gasto mínimo de energia;
b. *Flooding*, que transmite dados a todos os nós vizinhos independentemente de eles terem ou não recebido os dados anteriormente;
c. *Gossiping*, que envia dados a um nó vizinho selecionado anteriormente;

d. SPIN (*Sensor Protocols for Information via Negotiation*), que envia dados aos nós-sensores somente se eles têm algum interesse;
e. SAR (*Sequential Assignment Routing*), que cria múltiplas árvores onde a raiz de cada uma está a um salto do *sink*[6];
f. *LEACH* (*Low-Energy Adaptive Clustering Hierarchy*), que forma agrupamentos para minimizar a dissipação de energia;

2.2.1 Protocolo *LEACH*

O protocolo *Low-energy Adaptive Clustering Hierarchy* (*LEACH*) foi apresentado por Heinzelman et al. em 2003 e logo tornou-se um dos mais populares algoritmos de roteamento para redes de sensores sem fio (HENNING, 2013).

> Ele baseia-se na premissa de formar grupos de sensores com base no potencial do sinal recebido e utilizam de head-*clusters*, que farão o processamento de agregação/fusão e transmissão dos dados para o coletor, funcionando como um tipo de roteador, economizando assim energia dos demais. (Henning, 2013, p. 21-22).

Além disso, o *LEACH* estende a vida útil da rede através da divisão dos sensores em grupos, também chamados de *clusters*. O tamanho de cada *cluster* será estabelecido pelo alcance do sensor principal, chamado de *header*, o qual terá função também de coletar os dados dos demais sensores naquele *cluster* e transmiti-los para a estação-base (podendo ser um roteador, por exemplo), fazendo assim com que apenas os sensores-*header* consumam mais energia.

A cada determinado tempo, chamado de *round*, faz-se uma nova escolha de *header*, objetivando com que todos os sensores da rede tenham sido *header* pelo menos uma vez (HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2003).

No protocolo, existem dois principais procedimentos: a configuração e o estado estável[7]. Na fase de configuração acontece a escolha dos *headers* e a formação dos

---

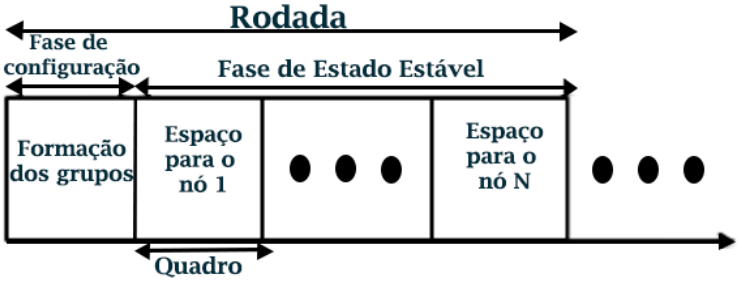
[6] Nome usado pelo autor para se referir ao nó da estação-base, local onde os sensores encaminham as informações coletadas.

7 Tradução livre de *steady state phase*.

grupos. Já na fase de estado estável, os sensores eleitos como *headers* irão receber os dados coletados pelos sensores de seus respectivos grupos e os transmitirão para a estação base.

O conjunto do procedimento da fase de configuração com a fase de estado-estável é chamado de *round*, como mostra a Figura 2. A cada *round* uma nova configuração acontece e a transmissão dos dados é feita.

Figura 2 – Fases de configuração do *LEACH*



Fonte: Energy efficient m- level *LEACH* protocol (2015)

A escolha do *header* é feita através da Distribuição de Bernolli, para que uma porcentagem P de sensores seja eleito e tenha-se *headers* por round de aproximadamente k sensores. Para isso, cada nó gera um número de 0 a 1 que é comparado com um limite variável no tempo T(n) e se o valor gerado pelo nó for abaixo desse número o nó é eleito. Dado um *round* r, tem-se a Equação 1, sendo G o grupo de nós que não se tornaram *header* nas últimas 1/p interações (KAMARUDIN et al., 2016):

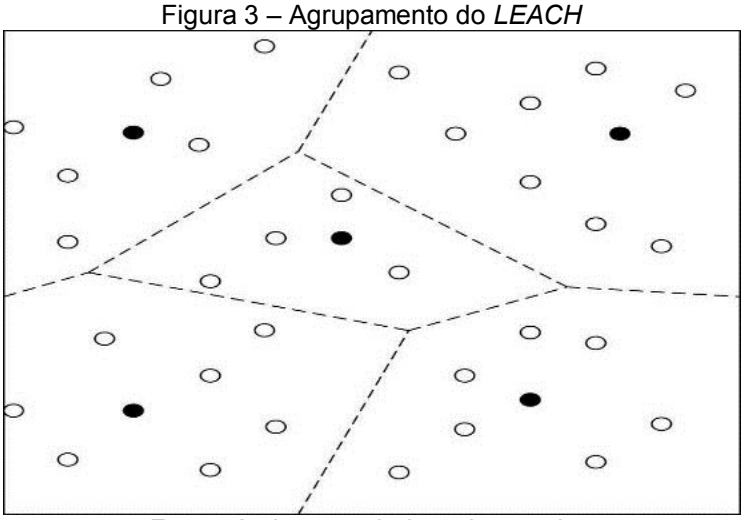
$$T(n) = \begin{cases} \frac{P}{1-P\left(rmod\left(\frac{1}{P}\right)\right)}, se\ n \in G \\ 0, casocontrário \end{cases} \quad (1)$$

Ao início do *round* 0 cada nó tem a probabilidade P de ser eleito. Os nós eleitos ficarão inelegíveis pelos próximos 1/p *rounds*, no *round* 1/p os nós ainda não eleitos terão probabilidade limite de 1 e, após o *round* 1/p, todos os nós terão a mesma probabilidade de serem eleitos novamente. Vale ressaltar que este cálculo considera uma rede onde todos os sensores iniciam com a mesma quantidade de energia.

Na Figura 3 é possível ver um exemplo de como a rede fica após a fase de configuração. Os pontos preenchidos são os *headers*, os demais estão representados por círculos vazios, já os traços delimitam cada região (*cluster*).

Deve-se destacar que o exemplo dado é de uma divisão ideal. Como a divisão é feita a partir do alcance de um sensor podem ocorrer situações onde sensores eleitos podem estar próximos entre si ou estarem na borda da rede.

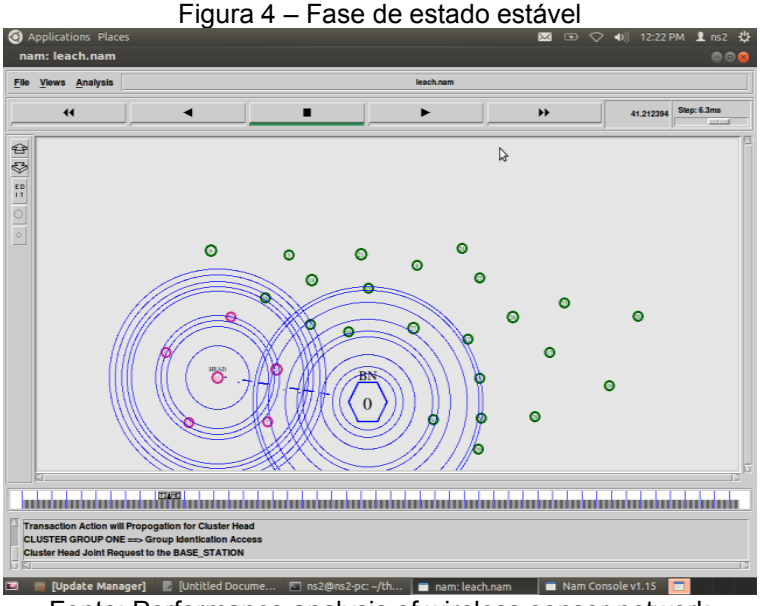
Figura 3 – Agrupamento do *LEACH*



Fonte: An improved *cluster*ing routing mechanism for wireless Ad hoc network (2017)

Na Figura 4, pode-se ver como é o comportamento da rede durante a fase de estado estável. Os círculos rosas são os *headers*, os quais farão a coleta dos outros sensores de seus grupos, representados pelos círculos verdes, e transmitirão para a estação-base que está representada pelo círculo com o hexágono em torno dela. Os círculos azuis são os raios de alcance dos sensores e da estação-base.

Figura 4 – Fase de estado estável



Fonte: Performance analysis of wireless sensor network using *LEACH* protocol (2015)

O *LEACH* adota como propagação de rádio o Modelo de Propagação de Espaço Livre (do inglês, Free Space Propagation Model, FSPM), o qual não considera obstáculos entre o transmissor e o receptor, assim, fazendo com que a comunicação entre os dois seja somente afetada pela distância (KAMARUDIN et al. 2016).

Vários problemas com o *LEACH* foram encontrados desde sua criação em 2003, dos quais podem-se destacar: a falta de mobilidade (KIM; CHUNG, 2006); perda de energia (YASSEIN et al., 2009); má escolha de *header* (TONG, 2010); controle de *headers* gerados (WU; WANG. 2010); e interferência entre sensores (KAMARUDIN et al., 2016).

3 BIBLIOGRAFIAS REFERENTES AO *LEACH*

Kim e Chung (2006) perceberam que o *LEACH* não considerava a mobilidade dos sensores e propuseram o *LEACH-Mobile*, uma variação que, durante a transmissão na fase de estado estável, deixa o sensor móvel livre para ter suas informações coletadas pelo *header* mais próximo. A fase de configuração nesta variante se mantém a mesma da usada originalmente no *LEACH*.

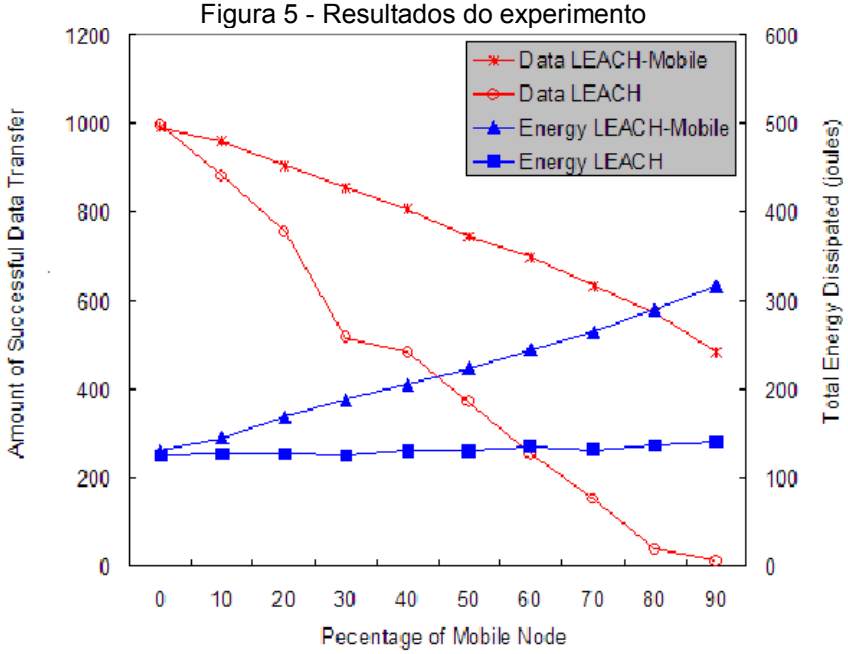
Ainda segundo os autores, o *LEACH*-Mobile se mostrou um protocolo melhor quanto à perda de pacotes de dados de sensores móveis em ambientes, nomeado por eles como *hot areas*, que são lugares com grande densidade de atividade humana numa base diária, tais como hospitais, universidades e grandes museus, por exemplo. Entretanto, essa variante tem maior quantidade de dissipação de energia devido às checagens via mensagem que o protocolo faz aos sensores móveis para garantir o envio de informação.

A Figura 5 demonstra o comparativo feito pelos autores em uma rede contendo cem sensores em um ambiente de 50mx50m onde mostra sobre o sucesso de transferência de dados e quantia de dissipação de energia conforme os nós móveis aumentam. Nele vê-se pelos traços vermelhos – com círculos para o LEACH e quadrados para o LEACH-Mobile – que a perda de dados do LEACH já é maior a partir dos dez sensores, mas a diferença se torna mais evidente a partir dos trinta sensores e que o maior consumo de energia do LEACH-Mobile – representado pelos traços azuis com triângulos – não é suficientemente maior para inviabilizar essa rede em ambientes como os citados anteriormente.

Yassein et al. (2009) verificaram que o *LEACH* tinha perda de energia – e, consequentemente uma vida útil menor – quando um *header* se extinguia, seja por uso de toda sua energia ou por alguma outra razão. A partir daí, apresentaram o *V-LEACH*, onde, além da escolha de um *header* em cada *cluster*, como o *LEACH* já fazia, ele elege também um *vice-header* na fase de seleção baseado na distância, energia residual e energia total. O *vice-header* assumirá quando o *header* parar de funcionar, recebendo os dados e transmitindo à estação-base, sem a necessidade que uma nova fase de seleção ocorra. Isso permite que menos mensagens sejam criadas e que o consumo de energia no *V-LEACH* seja menor.

Tong e Tang (2010) propuseram o protocolo *LEACH-B*, também conhecido como *LEACH-Balanced*, visando fazer uma melhor escolha do *cluster header*. Nele, além da seleção que já existe no *LEACH*, uma segunda seleção é feita baseada em quanta energia restante cada sensor possui, assim fazendo escolhas mais eficientes e estendendo a vida útil da rede.

Figura 5 - Resultados do experimento

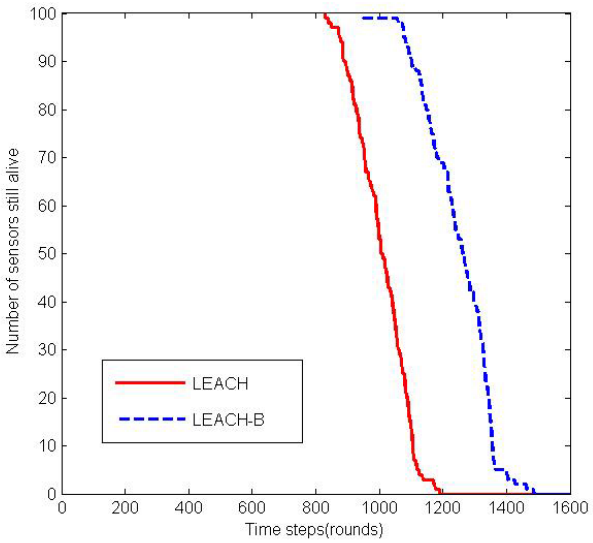


Fonte: Self-Organization Routing Protocol Supporting Mobile Nodes for Wireless Sensor Network (2006)

Na Figura 6, os autores demonstram o tempo de vida de uma rede utilizando os algoritmos *LEACH* e o *LEACH-B*. Pelos traços vermelhos percebe-se que em um ambiente com cem sensores os nós passam a esgotarem-se em torno das 800 rodadas, chegando ao fim da energia de todos os sensores em torno das 1200 rodadas. Já o *LEACH-B*, representado pelos traços azuis, permite com que os nós decaiam em torno das 1000 rodadas e o esgotamento de energia de todos os sensores chegue apenas em torno da rodada número 1500.

Wu e Wang (2010) ao notarem que o *LEACH* não controlava o número de *headers* gerados devido ao fato de a eleição destes *headers* ser feita pelos sensores presentes na rede. Eles propuseram o protocolo *LEACH-C* que faz a escolha dos *headers* de cada *cluster* ser determinada pela estação-base, a qual coleta a energia residual e a localização dos sensores. Isto permite não somente controlar quantos grupos e *headers* a rede terá, como também proporcionaria qual sensor é o mais adequado como cabeça.

Figura 6: Tempo de vida do sistema usando *LEACH* e *LEACH-B*



Fonte: *LEACH-B*: An Improved *LEACH* Protocol for Wireless Sensor Network (2010)

O tempo de vida da rede do *LEACH* e *LEACH-C* são influenciados pela localização da estação-base, como os autores demonstram na Tabela 1 em um ambiente com cem sensores; entretanto mesmo em condições onde o *LEACH-C* tem tempo de vida menor, ele ainda transmite mais dados, como ocorre no caso da estação-base localizada em (49,175).

Tabela 1: Simulação de Resultados entre *LEACH* e *LEACH-C*

Coordenadas da Estação-Base	Parâmetros de Desempenho	*LEACH*	*LEACH-C*
(49, 175)		505	350
(49, 195)		480	405
(49, 215)		451	455
(49, 225)		440	470
(49, 245)	Tempo de Vida da	390	491
(49, 265)	Rede	374	513
(175, 47)	Nós Sobreviventes	565	498
(195, 47)	em 20 segundos	549	512
(215, 47)		540	543
(225, 47)		500	557
(255, 47)		470	566
(275, 47)		443	579

Fonte: Performance Comparison of *LEACH* and *LEACH-C* Protocols by NS2 (2010)

O Gráfico 1 mostra um comparativo, trazendo vantagens e desvantagens das variantes do *LEACH* em relação ao mesmo, mostrando que mesmo cada versão alternativa tendo um desempenho satisfatório, podendo não ser o adequado para determinados ambientes ou objetivos.

Gráfico 1: Comparativo de protocolos baseados no *LEACH*

Protocolo	Vantagem sobre o LEACH	Desvantagem
LEACH-M	Suporta mobilidade e possui menos perda de pacote	Ainda possui perda de pacote em caso do header ser móvel e é menos eficiente em relação à energia
LEACH-B	Melhor tempo de vida da rede e escolha do header.	Maior sobrecarga na seleção do header.
LEACH-C	Mais rounds em áreas pequenas	Mais sobrecarga na estação-base e menor eficiência em redes grandes.
V-LEACH	Vice-header atua como header em caso do header morrer.	Maior sobrecarga na seleção do vice-header

Fonte: Survey of *LEACH* variants (2015, adaptado)

Existem problemas que a bibliografia aponta e que não foram investigados a fim de solucioná-los, tais como a distribuição dos *clusters*, a disposição dos sensores[8] e interferência entre sensores (Kamarudin, 2016). Além disso, não há nenhuma solução que adote *clusters* mais homogêneos, ou seja, grupos com quantidades de sensores similares entre si ou que se preocupe com a quantidade de sensores inclusos no grupo.

[8] Embora o *LEACH-C* considere a localização para a escolha do *header* e, assim, melhore a seleção.

4 PROPOSTA DO TRABALHO E ANÁLISE EXPERIMENTAL DO PROTOCOLO *LEACH*

O presente capítulo analisa a distribuição de energia entre os sensores de *LEACH*, problema já apresentado durante a breve análise do referencial teórico no capítulo anterior.

4.1 METODOLOGIA DO TRABALHO

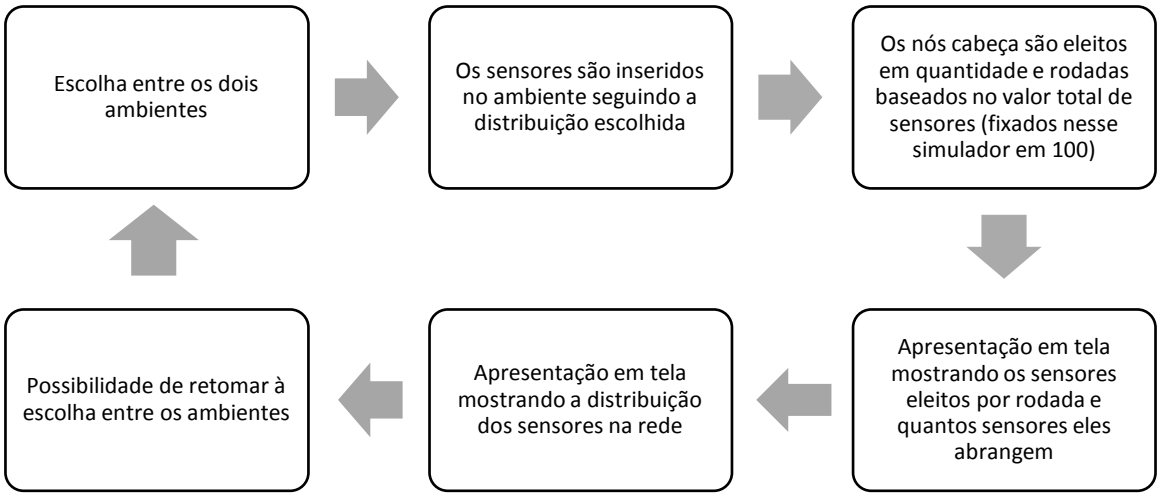
Inicialmente, foi levantada a bibliografia referente a Rede de Sensores Sem Fio e o protocolo *LEACH* para poder embasar a análise experimental aplicada a este trabalho. Para isto, foram coletadas informações acerca dos seus respectivos fundamentos básicos, funcionamento e aplicabilidade, vantagens e desvantagens, bem como a forma para a simulação do protocolo em linguagem de programação C.

A análise do presente trabalho refere-se ao estudo do protocolo *LEACH*, simulando-o em linguagem de programação C, investigando seu consumo de energia quando comparado em um ambiente que o desconsidere ou que a distribuição não seja perfeitamente ordenada. A linguagem de programação em C foi escolhida devido a familiaridade do autor com a linguagem.

O escopo do código foi dividido em cinco partes: definição dos ambientes, inserção dos sensores nos ambientes, seleção dos sensores-cabeça, delimitação das regiões que os sensores-cabeça atuarão e quantia de sensores que cada sensor-cabeça abrange.

A definição dos ambientes é a escolha de um espaço para a implementação do protocolo *LEACH*, permitindo a limitação de certos tipos de ambientes para servir de amostra para a simulação do *LEACH* e, através desses ambientes é possível determinar se o protocolo possui viabilidade de implementação.

Gráfico 2 – Diagramação do funcionamento do simulador



Fonte: Autoria própria (2020)

A inserção dos sensores é a etapa na qual são introduzidos os sensores em geral, possibilitando testar diferentes formas de distribuição destes na rede e como isso impacta no funcionamento do protocolo.

A seleção dos sensores-cabeça mostra como o *LEACH* elege os *headers*, quantos sensores serão escolhidos e o custo que eles terão a mais. E por fim, a delimitação das regiões onde os sensores-cabeça atuarão mostra uma ideia do funcionamento do alcance dos sensores eleitos. A quantidade de sensores que cada sensor-cabeça abrange proporciona uma avaliação de quais sensores recolhem mais dados em um dado ambiente.

4.2 ANÁLISE EXPERIMENTAL

O código, como dito anteriormente, contempla a definição dos ambientes, a inserção dos sensores e o comportamento do *LEACH* sobre esses ambientes. Para desenvolvê-lo em C foi usado o ambiente de trabalho integrado (do inglês, *IDE*) Dev C++ que compila programas para o sistema operacional *Microsoft Windows*.

A Figura 7 apresenta as bibliotecas padrões necessárias para o funcionamento do código – como, por exemplo, o *time.h* que auxilia na aleatoriedade da distribuição e escolha de sensores –, bem como as definições de variáveis globais criadas. Este segundo tem como objetivo facilitar algumas das operações, e caso seja de interesse, alterar o tamanho do ambiente ou a quantidade de sensores, sendo *DIM* para o tamanho e *QNT* para o número de sensores.

Figura 7 - Bibliotecas

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <conio.h>
4   #include <time.h>
5   #include <string.h>
6
7   #define DIM 20
8   #define QNT 100
```

Fonte: Autoria própria (2020)

Na Figura 8, é possível ver a parametrização de cada função auxiliar. Algumas delas são utilizadas na função principal, outras auxiliam no pleno funcionamento certas funções auxiliares – como é o caso da *preencherMatriz* que visa diminuir linhas de código nas funções *inserirSensor* e *inserirSensorR* –.

Figura 8 – Parametrização de cada função auxiliar

```
10  void inserirSensor(int matriz[][DIM], int* vetor);
11  void inserirSensorR(int matriz[][DIM], int vetor[QNT]);
12  void preencherSensor(int* sensor, int vlr_sensor);
13  void Rodadas(int* vetor, int matriz[][DIM]);
14  int Distancia(int x, int matriz[][DIM]);
15  void embaralhar(int* vetor);
16  void preencherMatriz(int matriz[][DIM]);
```

Fonte: Autoria própria (2020)

Devido à dimensão da codificação, esta foi inserida completa como anexo, denominado de Anexo A, e desmembrada nesse respectivo subcapítulo (4.2.) através das linhas de código.

A função principal do código situa-se entre as linhas 17 a 69. Nela, seleciona-se qual dos ambientes deseja-se que seja simulado dentre os dois disponíveis, sendo um com os sensores perfeitamente distribuídos (1), referente ao *inserirSensor*, e outro com parte dos sensores distribuídos aleatoriamente (2), referente ao *inserirSensorR*. Em seguida, encontram-se as funções – que serão explicadas detalhadamente mais adiante –, como a geração das rodadas de sensores-cabeça, o preenchimento e inserção dos sensores no ambiente.

Nas linhas 71 a 87, demonstra-se o funcionamento da inserção do sensor em um ambiente de forma ordenada, sendo a opção (1), simulando a distância de 2 metros entre eles – representando o que seria a diferença entre um objeto e outro contendo o sensor–.

Nas linhas, 89 a 218, vê-se a função de inserção dos sensores no segundo ambiente, sendo a opção (2). Ela simula a inserção pseudoaleatória da seguinte forma: na primeira metade do ambiente são colocados 75 sensores aleatórios com distâncias de zero, um e dois metros entre eles, essas distâncias são sorteadas para imitar a distância em que um objeto contendo o sensor teria de diferença para outro em uma fábrica, por exemplo. Na segunda metade do ambiente são colocados os 25 sensores restantes sorteados com distribuição aleatória, buscando imitar um estacionamento de fábrica ou qualquer outro ambiente que não tenha a distribuição definida.

Nas linhas 220 a 254, mostra a geração das rodadas. Para o cálculo de quantos sensores seriam eleitos por rodada, foi usada a Equação 2 disposta por Heinzelmann et al. (2003).

$$\sum_{i=1}^{N} P_i(t) * 1 = k \qquad (2)$$

A letra *i* refere-se a cada um dos sensores que elegem a si mesmos como *headers* no início de cada rodada com tempo *t* e com uma probabilidade *Pi(t)*, sendo assim, *k* o número aproximado de sensores-cabeça por rodada. Neste caso, para cem sensores terão cinco sensores-cabeça eleitos, ocasionando em vinte rodadas até que todos os sensores sejam escolhidos.

A Tabela 2 mostra as probabilidades de cada sensor não eleito tem de ser escolhido conforme as rodadas avançam.

Tabela 2: Probabilidade de escolha de um sensor ainda não eleito

Número da rodada	Chance de ser escolhido (em %)	Sensores restantes	Número da rodada	Chance de ser escolhido (em %)	Sensores restantes
1	5%	95	11	10%	45
2	5%	90	12	11%	40
3	6%	85	13	12%	35
4	6%	80	14	14%	30
5	7%	75	15	16%	25
6	7%	70	16	20%	20
7	8%	65	17	25%	15
8	8%	60	18	33%	10
9	9%	55	19	50%	5
10	10%	50	20	100%	0

Fonte: Autoria própria (2020)

Nas linhas 254 a 298, mostram parte da função que trata o alcance de um sensor no ambiente – visto que o restante do código segue análogo, variando apenas a direção a ser verificada pelo sensor –. Quando um sensor é designado como cabeça, essa função faz uma varredura dos sensores próximos – considerando os três metros de alcance – e contabiliza quantos sensores serão agrupados na região daquele sensor, embora aconteça de uma região se sobrepor a outra.

Nas linhas 508 a 533, encontram-se as funções de embaralhamento e preenchimento. A função de embaralhamento faz com que os valores ordenados sequencialmente – neste caso os cem sensores – sejam misturados de forma aleatória baseado no algoritmo de mistura de Fisher-Yates (1938), porém sorteando dois valores ao invés de um.

O algoritmo de mistura de Fisher-Yates, criado em 1938 por Ronald Fisher e Frank Yates, é um método simples de permutação dada uma quantia finita de números, podendo, inclusive, ser aplicada usando papel e caneta, cuja finalidade é gerar um conjunto de números aleatórios.

As funções de preenchimento, como o nome sugere, inserem valores para os sensores, como no caso do *preencherSensor*, ou apenas ocupam o ambiente com valores zerados como acontece com o *preencherMatriz*; isso garante que valores desconhecidos – conhecidos também como lixo de memória – sejam carregados e tratados como parte do código.

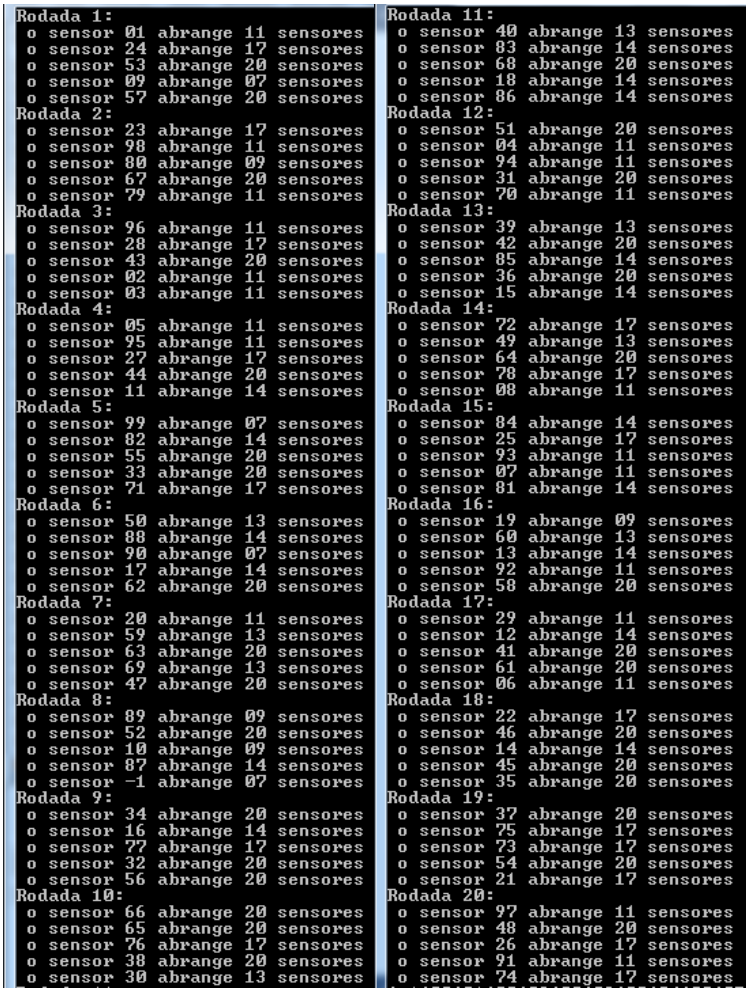
4.3 RESULTADOS DA SIMULAÇÃO

Ao executar o código mostrado anteriormente, o resultado em tela apresentado pela Figura 9 mostra quais foram os sensores eleitos como cabeça e quantos sensores eles cobrem baseado num alcance de três metros, no caso de um ambiente com os sensores perfeitamente ordenados[9], como mostra a Figura 10, sendo possível também, ver quais os sensores serão contidos em cada grupo de *headers*. Por exemplo, o sensor 21 quando escolhido como cabeça alcançará dezessete sensores da rede – sendo eles -1, 1, 2, 10, 11, 12, 20, 22, 30, 31, 32, 40, 41, 42, 50, 51 e 52 -.

Vale ressaltar que o sensor -1 refere-se ao sensor 0 que, ao desenvolver a codificação, teve que ser representado por -1 por 0 ser utilizado como separador entre os sensores, para facilitar a visualização.

[9] Como, por exemplo, uma plantação onde a distância entre os elementos segue um padrão uniforme e formam fileiras.

Figura 9 – Sensores escolhidos em ambiente organizado

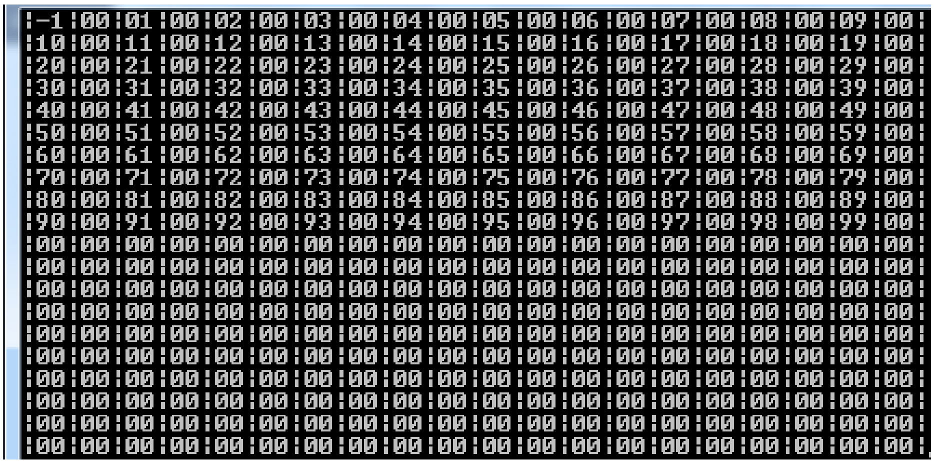


Fonte: Autoria própria (2020)

Com isso, ao atribuir um valor **5** ao custo energético de um sensor cabeça considerando a coleta dos dados dos sensores de seu grupo, tratamento e envio para a estação-base, e valor **1** ao custo dos demais sensores, ao fim das **20** rodadas, os sensores com o uso do protocolo *LEACH* terão gasto 5 + (1 * 19), sendo **19** a quantidade de rodadas restantes, totalizando **24**.

Enquanto que em uma rede que desconsidere o uso do protocolo, todos os sensores teriam gasto de **100**, devido ao fato que todos eles, a cada rodada, teriam que coletar os dados, tratar e enviar para a estação, ou seja, a cada vinte rodadas o uso do protocolo *LEACH* permite uma economia de **76** de valor energético à uma RSSF com cem sensores. Mas e se comparado com outra rede com distribuição diferente dos sensores contendo igualmente o protocolo *LEACH*?

Figura 10 – Distribuição dos sensores no ambiente perfeitamente distribuído



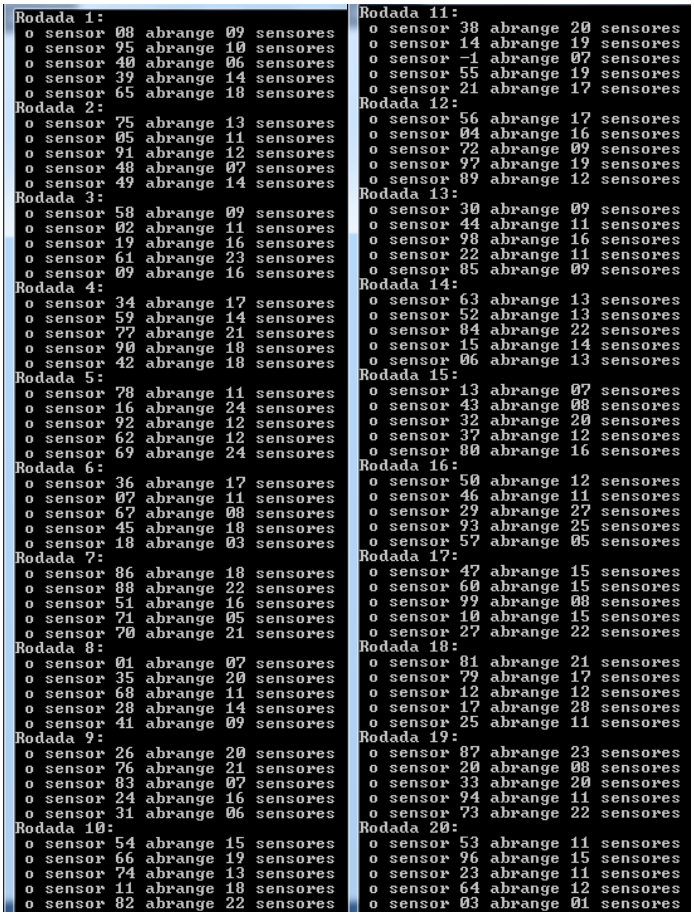
```
|-1|00|01|00|02|00|03|00|04|00|05|00|06|00|07|00|08|00|09|00|
|10|00|11|00|12|00|13|00|14|00|15|00|16|00|17|00|18|00|19|00|
|20|00|21|00|22|00|23|00|24|00|25|00|26|00|27|00|28|00|29|00|
|30|00|31|00|32|00|33|00|34|00|35|00|36|00|37|00|38|00|39|00|
|40|00|41|00|42|00|43|00|44|00|45|00|46|00|47|00|48|00|49|00|
|50|00|51|00|52|00|53|00|54|00|55|00|56|00|57|00|58|00|59|00|
|60|00|61|00|62|00|63|00|64|00|65|00|66|00|67|00|68|00|69|00|
|70|00|71|00|72|00|73|00|74|00|75|00|76|00|77|00|78|00|79|00|
|80|00|81|00|82|00|83|00|84|00|85|00|86|00|87|00|88|00|89|00|
|90|00|91|00|92|00|93|00|94|00|95|00|96|00|97|00|98|00|99|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|00|
```

Fonte: Autoria própria (2020)

As Figuras 11 e 12 mostram as mesmas informações dispostas anteriormente, porém, considerando uma distribuição onde 75% dos sensores terão distância de zero, um ou dois entre eles e os 25% restantes serão colocados aleatoriamente na segunda metade do ambiente[10].

[10] Para exemplo, pode-se pensar nesse ambiente como uma indústria, onde a primeira parte de sensores distribuídos seria a parte interna, onde encontram-se os maquinários e a parte aleatória o estacionamento.

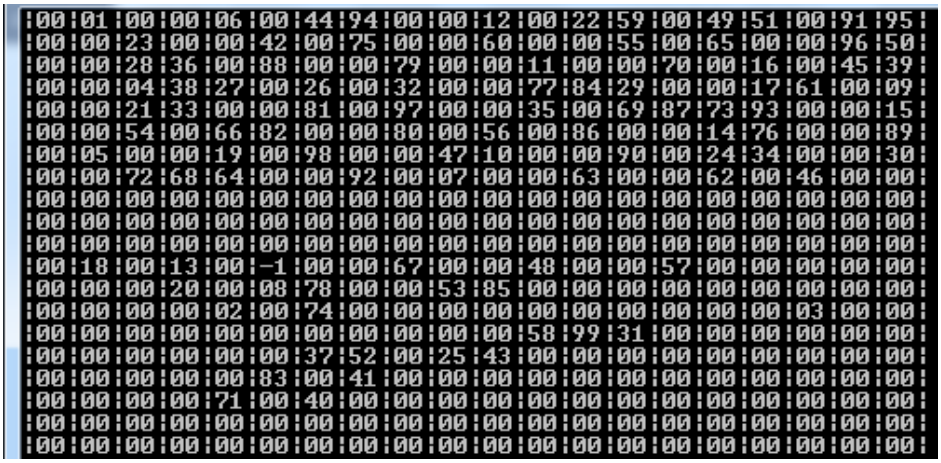
Figura 11 – Sensores escolhidos em ambiente com parte dos sensores organizados aleatoriamente



Fonte: Autoria própria (2020)

Pela abrangência dos sensores cabeça que a Figura 11 demonstra, já é possível identificar diferenças principalmente a quantidade mínima de sensores. O sensor 03, por exemplo, abrange apenas um sensor, enquanto no ambiente perfeitamente organizado a quantidade menor é de 07 sensores, característica que ocorre com os sensores das extremidades do ambiente.

Figura 12 – Distribuição dos sensores no ambiente com parte dos sensores organizados aleatoriamente



Fonte: Autoria própria (2020)

Se verificar-se o sensor disposto na mesma região que o sensor 21 se encontrava no exemplo anterior, temos o sensor 28 que, quando eleito na rodada oito, continha catorze sensores, isto por si só não mostra uma falha do protocolo, entretanto, ao somar a quantidade de sensores que são abrangidos nas duas redes temos no primeiro ambiente 1524, já no segundo temos 1444. Isto dá um valor de 80 entre cada exemplo, o que mostra que no primeiro caso mais sensores estarão contidos por cada *cluster* gerado, sendo assim, a forma com que são colocados no ambiente afeta o consumo total da rede e a transmissão de dados – visto que alguns dos sensores terão o envio de suas informações mais frequentemente -.

5 CONCLUSÃO

Como visto anteriormente, o protocolo LEACH tornou-se uma ferramenta de roteamento para RSSFs por permitir a extensão da vida útil da rede através da divisão dos sensores em grupos, permitindo a estabilidade energética dos próprios sensores em um ambiente. Manter a constância energética é significativa principalmente quando o ambiente é extenso e/ou requer funcionamento constante. Como, por exemplo, os sensores de capacitação de dados meteorológicos, que funcionam 24 horas por dia e demandam de vários componentes para captação de informações. Devido essa atualização constante de dados é possível determinar as mudanças climáticas com maior precisão de uma dada região.

A partir da codificação em C proposta neste trabalho averiguou-se o desempenho do protocolo *LEACH*, principalmente verificando acerca do comportamento energético do protocolo. O problema quanto a sobrecarga de energia entre os sensores é algo contínuo e perceptível na bibliografia a respeito de *LEACH*. O ambiente o qual utilizará do protocolo é um fator fundamental para a economia – ou não - de energia dos sensores, o funcionamento, a captação e a distribuição de informações entre sensores e estação-base. Quanto mais organizado é o ambiente, melhor será o funcionamento da rede.

Pode-se afirmar, assim, que conhecer previamente como se dá a distribuição dos sensores na rede e levar em conta o alcance desses sensores podem ser medidas que amenizariam o consumo de energia da rede e prolongariam seu tempo de vida, portanto, necessários para qualquer aplicação prática de uma RSSF. Inclusive, é possível verificar que os sensores agrupados sofrerão de sobrecarga e repetição de informação, pois serão requisitados mais vezes durante um mesmo *round*. O oposto ocorre aos sensores mais distantes fazendo com que sofram de acúmulo de informação.

5.1 TRABALHOS FUTUROS

A existência de diversas variantes do *LEACH* buscando resolver problemas mostra a vulnerabilidade dele em uso prático, mas revelam também sua influência como base teórica. Caberia, assim, um comparativo entre estas variantes, visto que algumas delas resolvem o mesmo problema do *LEACH*, como no caso do *LEACH-B* e *LEACH-C*, por exemplo.

Os problemas de sobrecarga, repetição e acúmulo de informação podem ser trabalhados em uma divisão dos sensores distribuídos na rede em dois planos que se alternariam a cada fase de seleção, permitindo que sensores mais próximos não precisassem enviar as informações tão frequentemente.

Também, como é apresentado no fim do capítulo 3, a não consideração de interferência entre os sensores e falta de homogeneidade dos grupos criados são questões sobre o *LEACH* que não foram amplamente explorados, afim de apresentar uma solução, podendo ser objetos de trabalhos futuros.

6 ANEXOS

ANEXO A – CODIFICAÇÃO DOS AMBIENTES E SENSORES

Pastebin.com - Printed Paste ID: https://pastebin.com/6Rg2mabq https://pastebin.com/print/6Rg2mabq

```c
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#include <string.h>

#define DIM 20
#define QNT 100

void inserirSensor(int matriz[][DIM], int* vetor);
void inserirSensorR(int matriz[][DIM], int vetor[QNT]);
void preencherSensor(int* sensor, int vlr_sensor);
void Rodadas(int* vetor, int matriz[][DIM]);
int Distancia(int x, int matriz[][DIM]);
void embaralhar(int* vetor);
void preencherMatriz(int matriz[][DIM]);
int main (){
inicio: int ambiente[DIM][DIM];
    int a = 0, b = 0, i, j, sensor[QNT];
    char k[QNT];
    printf("pressione 1 para gerar um ambiente com os sensores perfeitamente distribuidos"
    " e 2 para gerar um ambiente com parte dos sensores distribuidos aleatoriamente \n");
    gets(k);
    system("cls");
    preencherSensor(sensor, QNT);
    srand ((unsigned) time (NULL));
    if(strcmp(k, "1") == 0){
        inserirSensor(ambiente, sensor);
    }else{
        if(strcmp(k, "2") == 0){
            inserirSensorR(ambiente, sensor);
        }else{
            goto inicio;
        }
    }
    Rodadas(sensor, ambiente);
    for(j=0;j<20;j++){
        if(j%1 == 0){
            if(j == 0){
                printf("\n");
            }else{
                printf("|\n");
            }
        }
        for(i=0;i<40;i++){
            if(i%2 == 0){
                printf("|");
            }else{
                if(ambiente[a][b] < 10 && ambiente[a][b] != -1){
                    printf("0%d", ambiente[a][b]);
                }else{
                    printf("%d", ambiente[a][b]);
                }
                b++;
            }
        }
        a++;
```

```
58.         b=0;
59.     }
60.     printf("|");
61.     getch();
62.     if(1 != 0){
63.         i=0;
64.         j=0;
65.         system("cls");
66.         goto inicio;
67.     }
68.     return 0;
69. }
70. 
71. void inserirSensor(int matriz[][DIM], int* vetor){
72.     int i, j, k = 0;
73.     preencherMatriz(matriz);
74.     for(i = 0; i < 20; i++){
75.         for(j = 0; j < 20;j++){
76.             if(j % 2 == 0 && k < 100){
77.                 if(k == 0){
78.                     matriz[i][j] = -1;
79.                     k++;
80.                 }else{
81.                     matriz[i][j] = vetor[k];
82.                     k++;
83.                 }
84.             }
85.         }
86.     }
87. }
88. 
89. void inserirSensorR(int matriz[][DIM], int vetor[QNT]){
90.     int i, j, k = 0, x, temp[QNT], a = 0, b = 0;
91.     preencherSensor(temp, QNT);
92.     preencherMatriz(matriz);
93.     embaralhar(temp);
94.     for(i = 0; i < 20; i++){
95.         for(j = 0; j < 20; j++){
96.             if(k < 100){
97.                 if(k < 75){
98.                     x = rand() % 3;
99.                     switch(x){
100.                        case 0:
101.                            if(temp[k] == 0){
102.                                matriz[i][j] = -1;
103.                                k++;
104.                            }else{
105.                                matriz[i][j] = temp[k];
106.                                k++;
107.                            }
108.                            break;
109.                        case 1:
110.                            if(temp[k] == 0){
111.                                if(matriz[i][j] > 100 || matriz[i][j] < -1){
112.                                    matriz[i][j] = 0;
113.                                }
114.                                matriz[i][j+1] = -1;
```

```
                                j++;
                                k++;
                            }else{
                                if(matriz[i][j] > 100 || matriz[i][j] < -1)
                                    matriz[i][j] = 0;
                                if(j+1 < 20){
                                    matriz[i][j+1] = temp[k];
                                    j++;
                                }else{
                                    if(i < 20){
                                        i+=1;
                                        j = 0;
                                        matriz[i][j] = temp[k];
                                    }
                                }
                                k++;
                            }
                            break;
                        case 2:
                            if(temp[k] == 0){
                                if(matriz[i][j] > 100 || matriz[i][j] < -1){
                                    matriz[i][j] = 0;
                                }
                                if(matriz[i][j+1] >100 || matriz[i][j+1] < -1){
                                    matriz[i][j+1] = 0;
                                }
                                if(j+2 <= 20){
                                    matriz[i][j+2] = -1;
                                    j+2;
                                }else{
                                    if(i < 20){
                                        if(j == 19){
                                            i+=1;
                                            j = 1;
                                        }else{
                                            if(j == 20){
                                                i+=1;
                                                j = 0;
                                            }
                                        }
                                        matriz[i][j] = -1;
                                    }
                                }
                                k++;
                            }else{
                                if(matriz[i][j] > 100 || matriz[i][j] < -1){
                                    matriz[i][j] = 0;
                                }
                                if(matriz[i][j+1] > 100 || matriz[i][j+1] < -1){
                                    matriz[i][j+1] = 0;
                                }
                                if(j+2 <= 20){
                                    matriz[i][j+2] = temp[k];
                                    j = j + 2;
                                }else{
                                    if(i < 20){
                                        if(j == 20){
```

```
                                        i += 1;
                                        j = 1;
                                    }else{
                                        if(j == 21){
                                            i+= 1;
                                            j = 0;
                                        }
                                    }
                                    matriz[i][j] = temp[k];
                                }
                            }
                            k++;
                        }
                        break;
                    }
                }else{
                    for(i=i; i< 20; i++){
                        for(j = 0; j < 20; j++){

                        }
                    }
another:
                a = rand() % 20;
                b = rand() % 20;
                    if(a > 10){
                        i = a;
                        j = b;
                        if(matriz[i][j] == 0){
                            if(temp[k] == 0){
                                matriz[i][j] = -1;
                                k++;
                            }else{
                                matriz[i][j] = temp[k];
                                k++;
                            }
                        }else{
                            goto another;
                        }
                    }else{
                        goto another;
                    }

                }
            }
        }
    }
}

void Rodadas(int* vetor, int matriz[][DIM]){
    int temp[QNT], i, j = 0, x, y, count = 1;
    for(i = 0; i < 20; i++){
        printf("\nRodada %d:", count);
        count++;
        for(j = 0; j < 5; j++){
            x = rand() % QNT;
            if(vetor[x] != -1){
                vetor[x] = -1;
```

```
                if(x == 0){
                    x = -1;
                    temp[i] = x;
                }else{
                    temp[i] = x;
                }
                y = Distancia(temp[i], matriz);
                if(temp[i] < 10 && y < 10 && x != -1){
                    printf("\n o sensor 0%d abrange 0%d sensores", x, y);
                }else{
                    if(y < 10){
                        printf("\n o sensor %d abrange 0%d sensores", x, y);
                    }else{
                        if(temp[i] < 10 && x != -1){
                            printf("\n o sensor 0%d abrange %d sensores", x, y);
                        }else{
                            printf("\n o sensor %d abrange %d sensores", x, y);
                        }
                    }
                }
            }else{
                j--;
            }
        }
    }
}

int Distancia(int x, int matriz[][DIM]){
    int i, j, count = 0;
    for (i = 0; i < DIM; i++){
        for(j = 0; j < DIM; j++){
            if(matriz[i][j] == x){
                if(j-1 >= 0){
                    if(j-2 >= 0){
                        if(j-3 >= 0){
                            if(matriz[i][j-3] != 0){
                                count++;
                            }
                            if(i-1 >= 0){
                                if(i-2 >= 0){
                                    if(matriz[i-2][j-3] != 0){
                                        count++;
                                    }
                                }
                                if(matriz[i-1][j-3] != 0){
                                    count++;
                                }
                            }
                        }
                        if(matriz[i][j-2] != 0){
                            count++;
                        }
                        if(i-1 >= 0){
                            if(i-3 >= 0){
                                if(matriz[i-3][j-2] != 0){
                                    count++;
                                }
```

```
                        }
                        if(matriz[i-1][j-2] != 0){
                            count++;
                        }
                    }
                }
                if(matriz[i][j-1] != 0){
                    count++;
                }
                if(i-2 >= 0){
                    if(i-3 >= 0){
                        if(matriz[i-3][j-1] != 0){
                            count++;
                        }
                    }
                    if(matriz[i-2][j-1] != 0){
                        count++;
                    }
                }
            }
            if(j+1< DIM){
                if(j+2<DIM){
                    if(j+3<DIM){
                        if(matriz[i][j+3] != 0){
                            count++;
                        }
                        if(i+1 < DIM){
                            if(i+2 < DIM){
                                if(matriz[i+2][j+3] != 0){
                                    count++;
                                }
                            }
                            if(matriz[i+1][j+3] != 0){
                                count++;
                            }
                        }
                    }
                    if(matriz[i][j+2] != 0){
                        count++;
                    }
                    if(i+1 < DIM){
                        if(i+3 < DIM){
                            if(matriz[i+3][j+2] != 0){
                                count++;
                            }
                        }
                        if(matriz[i+1][j+2] != 0){
                            count++;
                        }
                    }
                }
                if(matriz[i][j+1] != 0){
                    count++;
                }
                if(i+2 < DIM){
                    if(i+3 < DIM){
                        if(matriz[i+3][j+1] != 0){
```

```
                                count++;
                            }
                        }
                        if(matriz[i+2][j+1] !=0){
                            count++;
                        }
                    }
                }
                if(i+1 < DIM){
                    if(i+2 < DIM){
                        if(i+3 < DIM){
                            if(matriz[i+3][j] != 0){
                                count++;
                            }
                            if(j-1 >= 0){
                                if(j-2 >= 0){
                                    if(matriz[i+3][j-2] != 0){
                                        count++;
                                    }
                                }
                                if(matriz[i+3][j-1] != 0){
                                    count++;
                                }
                            }
                        }
                        if(matriz[i+2][j] != 0){
                            count++;
                        }
                        if(j-1 >= 0){
                            if(j-3 >= 0){
                                if(matriz[i+2][j-3] != 0){
                                    count++;
                                }
                            }
                            if(matriz[i+2][j-1] != 0){
                                count++;
                            }
                        }
                    }
                    if(matriz[i+1][j] != 0){
                        count++;
                    }
                    if(j-2 >= 0){
                        if(j-3 >= 0){
                            if(matriz[i+1][j-3] != 0){
                                count++;
                            }
                        }
                        if(matriz[i+1][j-2] != 0){
                            count++;
                        }
                    }
                }
                if (i-1 >= 0){
                    if(i-2 >= 0){
                        if(i-3>=0){
                            if(matriz[i-3][j] != 0){
```

```
                                count++;
                            }
                            if(j+1 < DIM){
                                if(j+2 < DIM){
                                    if(matriz[i-3][j+2] != 0){
                                        count++;
                                    }
                                }
                                if(matriz[i-3][j+1] != 0){
                                    count++;
                                }
                            }
                        }
                        if(matriz[i-2][j] != 0){
                            count++;
                        }
                        if(j+1 < DIM){
                            if(j+3 < DIM){
                                if(matriz[i-2][j+3] !=0){
                                    count++;
                                }
                            }
                            if(matriz[i-2][j+1] != 0){
                                count++;
                            }
                        }
                    }
                    if(matriz[i-1][j] != 0){
                        count++;
                    }
                    if(j+2 < DIM){
                        if(j+3 < DIM){
                            if(matriz[i-1][j+3] != 0){
                                count++;
                            }
                        }
                        if(matriz[i-1][j+2] != 0){
                            count++;
                        }
                    }
                }
                if(i+1 < DIM && j+1 < DIM){
                    if(i+2 < DIM && j+2 < DIM){
                        if(i+3 < DIM && j+3 < DIM){
                            if(matriz[i+3][j+3] != 0){
                                count++;
                            }
                        }
                        if(matriz[i+2][j+2] != 0){
                            count++;
                        }

                    }
                    if(matriz[i+1][j+1] != 0){
                        count++;
                    }
                }
```

```
457.                 if(i-1 >= 0 && j-1 >= 0){
458.                     if(i-2 >= 0 && j-2 >= 0){
459.                         if(i-3 >= 0 && j-3 >= 0){
460.                             if(matriz[i-3][j-3] != 0){
461.                                 count++;
462.                             }
463.                         }
464.                         if(matriz[i-2][j-2] != 0){
465.                             count++;
466.                         }
467.                     }
468.                     if(matriz[i-1][j-1] != 0){
469.                         count++;
470.                     }
471.                 }
472.                 if(i-1 >= 0 && j+1 < DIM){
473.                     if(i-2 >= 0 && j+2 < DIM){
474.                         if(i-3 >= 0 && j+3 < DIM){
475.                             if(matriz[i-3][j+3] != 0){
476.                                 count++;
477.                             }
478.                         }
479.                         if(matriz[i-2][j+2] != 0){
480.                             count++;
481.                         }
482.                     }
483.                     if(matriz[i-1][j+1] != 0){
484.                         count++;
485.                     }
486.                 }
487.                 if(i+1 < DIM && j-1 >= 0){
488.                     if(i+2 < DIM && j-2 >= 0){
489.                         if(i+3 < DIM && j-3 >= 0){
490.                             if(matriz[i+3][j-3] != 0){
491.                                 count++;
492.                             }
493.                         }
494.                         if(matriz[i+2][j-2] != 0){
495.                             count++;
496.                         }
497.                     }
498.                     if(matriz[i+1][j-1] != 0){
499.                         count++;
500.                     }
501.                 }
502.             }
503.         }
504.     }
505.     return count;
506. }
507. 
508. void embaralhar(int* vetor){
509.     int i, a, b, temp;
510.     for(i = 0; i < QNT*2; i++){
511.         a = rand() % QNT;
512.         b = rand() % QNT;
513.         temp = vetor[a];
```

```
514.         vetor[a] = vetor[b];
515.         vetor[b] = temp;
516.     }
517. }
518. 
519. void preencherSensor(int* sensor, int vlr_sensor){
520.     int i;
521.     for(i = 0; i < 100; i++){
522.         sensor[i] = i;
523.     }
524. }
525. 
526. void preencherMatriz(int matriz[][DIM]){
527.     int i, j;
528.     for(i = 0; i < DIM; i++){
529.         for(j = 0; j < DIM; j++){
530.             matriz[i][j] = 0;
531.         }
532.     }
533. }
```

REFERÊNCIAS

BATTACHARYYA, D.; KIM, T. H.; PAL, S. A Comparative Study of Wireless Sensor Network and Their Routing Protocols. **Sensors**, v. 10, p. 10506-10523, nov. 2010.

FERREIRA, J. L. M. **Segurança em redes sem fio**. Orientador: Prof. Christian Carlos Souza Mendes. 2013. 49p. Monografia (Especialização em Configuração e Gerenciamento de Servidores e Equipamentos de Rede) – Universidade Tecnológica Federal do Paraná, Curitiba, Paraná. 2013. Disponível em <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/2412/1/CT_GESER_IV_2014_03.pdf>. Acesso em: 15 mar. 2020.

HAUBER, M. Behind the Net: The Untold Story of the ARPANET and Computer Science (Chapter 7). Disponível em: <http://pages.infinit.net/jbcoco/Arpa-Arpanet-Internet.pdf>. Acesso em: 12 fev. 2020.

HEINZELMAN, W. B; CHANDRAKASAN, A.P; BALAKRISHNAN, H. An Application-Specific Protocol Architecture for Wireless Microsensor Networks. **IEEE Transactions on Wireless Communications**, Notre Dame (IN), v. 1, n. 4, p. 660-670, out. 2002.

HENNING, M. **Protocolo de roteamento para redes de sensores sem fio baseado em políticas**. Orientador: Prof. Dr. Mauro Sérgio Pereira Fonseca. 2013. 95p. Dissertação (Mestre em Informática) – Pontifícia Universidade Católica do Paraná, Curitiba, Paraná. 2013. Disponível em: <https://www.ppgia.pucpr.br/pt/arquivos/mestrado/dissertacoes/2013/mauricio-henning.pdf>. Acesso em: 08 abr. 2020.

KABIR, M. H. et al. Detail Comparison of Network Simulators. I**nternational Journal of Scientific and Engineering Research**. v. 5, p. 203-218, nov. 2014.

KAMARUDIN, L. M.; et al. Simulation and Analysis of *LEACH* for Wireless Sensor Networks in Agriculture. **International Journal of Sensor Network**. v. 21, p. 16-26, jan. 2016.

KATOCH, P.; GUPTA, M. Survey of *LEACH* Variants. **International Journal of Computer Science Engineering & Technology**. v. 5, n. 12, p. 432-441, dez. 2005.

KIM, D. S.; CHUNG, Y. J. Self-organization routing protocol supporting mobile nodes for wireless sensor network. **First International Multi-Symposiums on Computer and Computational Sciences**. jun. 2006.

KODALI, R. K. et al. Energy efficient m-level LEACH protocol. **Advances in Computing, Communications and Informatics (ICACCI)**. ago. 2015.

LOUREIRO, A. el at. Redes de sensors sem fio. **XXI Simpósio Brasileiro de Redes de Computadores**. jan. 2003.

PENG, J. Radio Propagation Models In Wireless Networks of Unmanned Aerial Vehicles. **International Journal of Computer Networks & Communications**. v. 7, n. 3, mai. 2015.

OTHMAN, M.F; SHAZALI, K. Wireless Sensor Network Applications: A Study in Environment Monitoring System**. International Symposium on Robotics and Intelligent Sensors**. v. 41, p. 1204-1210, 2012

RUIZ, L. B. et al. Arquiteturas para Redes de Sensores Sem Fio. In: **22 Simpósio Brasileiro de Redes de Computadores**. [S.l.: s.n.]. p. 167-218, 2004.

TAVARES, P. L.; **Redes de sensores sem-fio**. Universidade Federal do Rio de Janeiro, dez. 2002. Disponível em <https://www.gta.ufrj.br/grad/02_2/Redes%20de%20sensores/Redes%20de%20Sensores%20Sem-fio.htm>. Acesso em: 18 jan. 2020.

THIAGARAJAN, N. S.; MURUGAN, M. Performance analysis of wireless sensor network using *LEACH* protocol. **International Journal of Applied Engineering Research**. v. 10. p. 16111-16116, jan. 2015.

TONG, M.; TANG, M. *LEACH-B*: An improved *LEACH* protocol for wireless sensor network. **6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)**. out. 2010.

WANG, T.; WANG, Y.; HAN, C. An improved *cluster*ing routing mechanism for wireless Ad hoc network. **Journal of Intelligent and Fuzzy Systems**. v. 32, p. 3401-3412, abr. 2017.

WU, X. H.; WANG S. Performance comparison of *LEACH* and *LEACH-C* protocols by NS2. **9th International Symposium on Distributed Computing and Applications to Business, Engineering and Science**. ago. 2010.

YASSEIN, M. B.; et al. Improvement on *LEACH* protocol of wireless sensor network (V*LEACH*). **International Journal of Digital Content Technology and its Applications**. v. 3, n. 2, p. 132-136, 2009.

ZHAO, F.; GUIBAS L. J. **Wireless Sensor Networks**: An Information Processing Approach. 1. ed. Morgan Kauffman, 2012.