UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Yelco Antonio Marante Cañizales

Consumo Energético de Animais em Pastagem baseado em um Modelo de Mobilidade Híbrido para Redes de Sensores Sem Fio

Juiz de Fora
2019

Yelco Antonio Marante Cañizales

Consumo Energético de Animais em Pastagem baseado em um Modelo de Mobilidade Híbrido para Redes de Sensores Sem Fio

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Eduardo Barrére

Juiz de Fora

2019

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF
com os dados fornecidos pelo(a) autor(a)

Marante, Yelco.
Consumo Energético de Animais em Pastagem baseado em um Modelo de Mobilidade Híbrido para Redes de Sensores Sem Fio / Yelco Antonio Marante Cañizales. – 2019.
96 f. : il.

Orientador: Eduardo Barrére
Dissertação (Mestrado) – Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2019.

1. Redes de Sensores Sem Fio. 2. Protocolos de Roteamento. 3. Modelos de Mobilidade. 4. Consumo Energético da Rede. 5. Consumo Energético de Animais em Pastagem. I. Barrére, Eduardo, orient. II. Título.

Yelco Antonio Marante Canizales

"Consumo Energético de Animais em Pastagem baseado em um Modelo de Mobilidade Híbrido para Redes de Sensores Sem Fio"

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Juiz de Fora como requisito parcial à obtenção do grau de Mestre em Ciência da Computação.

Aprovada em 06 de novembro de 2019.

BANCA EXAMINADORA

Prof. Dr. Eduardo Barrére – Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Edelberto Franco Silva
Universidade Federal de Juiz de Fora

Dr. Ricardo Guimarães Andrade
Empresa Brasileira de Pesquisa Agropecuária

"Trabalho dedicado à minha família,
que tornaram tudo isso possível."

AGRADECIMENTOS

A presente dissertação de mestrado não poderia ser realizada sem o apoio de várias pessoas e instituições.

Primeiramente, agradeço a Deus por estar sempre ao meu lado durante toda a minha carreira, cada passo que dei foi graças a ele. Coloquei meus planos nas suas mãos e me sinto muito feliz por tudo.

Quero agradecer a minha família, especialmente aos meus Pais, Antonio Marante e Elizabeth Cañizales, por me guiarem e apoiarem incondicionalmente em cada um dos projetos até este instante, apesar das saudades pelos 2 anos que estou sem vê-los.

Também, agradeço a minha esposa, Aisbeth Martínez, pelo amor, companhia e motivação para realizar o mestrado. Sem ela, não seria possível a realização deste trabalho, já que foi a pessoa que me suportou durante meu início difícil com o idioma e os conteúdos das disciplinas. Além de ter sido a pessoa que ficou mais feliz quando recebi a noticia de fazer o mestrado no Brasil.

Ao meu orientador, Prof. Dr. Eduardo Barrére, por ser uma das primeiras pessoas que me acolheu e aceitou me orientar. Agradeço pela paciência, dedicação e toda orientação durante o mestrado. Embora, seja uma pessoa muito ocupada, as vezes que precisei da sua ajuda estava disponível para mim.

Agradeço imensamente por todo o aprendizado que tive com os professores do PPGCC, e por contribuir no meu desenvolvimento profissional e pessoal. A todos aqueles que contribuíram, direta ou indiretamente, para a realização desta dissertação, o meu sincero agradecimento.

Por último, quero agradecer à Organização dos Estados Americanos (OEA), ao Programa de Pós-Graduação em Ciência da Computação (PPGCC) e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pela oportunidade de fazer o mestrado e me selecionar como aluno estrangeiro e bolsista.

A todos, muito obrigado por acreditarem em mim...

Yelco Marante

"A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo que imaginação abrange o mundo inteiro"

Albert Einstein

RESUMO

O monitoramento de animais surgiu para a pecuária como um fator importante para o controle de qualidade e produtividade. A ideia é monitorar cada animal com a finalidade de fornecer ao produtor o resumo diário das atividades do bovino. Na maioria dos casos, o monitoramento é realizado através de um colar com sensor GPS embutido. O colar armazena o posicionamento do animal em um determinado ambiente, permitindo o monitoramento em tempo real, através do envio das informações para uma estação base. No entanto, os testes e a implantação desses equipamentos no mundo real são custosos, complexos e levam tempo para serem colocados em funcionamento. Neste cenário, a utilização de simuladores de redes de sensores sem fio se torna essencial para o desenvolvimento desse tipo de tecnologia. As redes de sensores sem fio são formadas por pequenos nós sensores com limitações de processamento, memoria e energia. Cada nó sensor é alimentado por uma bateria interna, tornando o tempo de vida da rede um parâmetro importante. Nos últimos anos, o problema de roteamento nas redes de sensores sem fio tem tido destaque na literatura e muitos protocolos de roteamento foram propostos. A maioria desses protocolos têm como foco a confiabilidade da transmissão e a eficiência energética das redes. Neste trabalho apresentamos um modelo configurável capaz de calcular o consumo energético de animais em pasto baseado em um modelo de mobilidade híbrido utilizado nos modelos de mobilidade individuais (Markoviano de Percurso Aleatório e Gauss-Markov) e em grupos (com Ponto de Referência) em redes *ad hoc*. O modelo de mobilidade foi implementado no simulador Castalia 3.2 e o algoritmo que faz os cálculos do consumo energético foi implementado em Python. A solução foi avaliada, como prova de conceito, em 3 distribuições de áreas, com 3 matrizes de relevo, para observar o comportamento esperado. Os resultados mostraram que o modelo é totalmente configurável e tem um bom funcionamento, semelhante à mobilidade real dos animais, conseguindo fazer os cálculos de consumo energético esperados. Além disso, este trabalho apresenta a avaliação de desempenho dos protocolos de roteamento: Multipath Rings (Proativo), Bypass e AODV (Reativos) e LEACH (Hierárquico). Como resultado, o protocolo que gerou o menor consumo de energia foi o LEACH e o maior consumo foi o Multipath Rings, em relação ao número de pacotes recebidos pelo nó *Sink* mostraram que a menor perda foi para o Multipath Rings e a maior perda foi para o LEACH.

Palavras-chave: Redes de Sensores Sem Fio. Protocolos de Roteamento. Modelos de Mobilidade. Consumo Energético da Rede. Consumo Energético de Animais em Pastagem.

ABSTRACT

Animal monitoring has emerged for livestock as an important factor for quality control and productivity. The idea is to monitor each animal to provide the producer with a daily summary of cattle activities. In most cases, monitoring is accomplished through a collar with a built-in GPS sensor. The collar stores the animal's positioning in a given environment, allowing real-time monitoring by sending information to a base station. However, testing and deploying this equipment in the real world is costly, complex and takes time to get up and running. In this scenario, the use of wireless sensor network simulators becomes essential for the development of this type of technology. Wireless sensor networks are made up of small sensor nodes with limitations in processing, memory, and power. Each sensor node is powered by an internal battery, making network life an important parameter. In recent years, the routing problem in wireless sensor networks has been highlighted in the literature and many routing protocols have been proposed. Most of these protocols focus on transmission reliability and the energy efficiency of networks. In this work, we present a configurable model capable of calculating the energy consumption of pasture animals based on a hybrid mobility model used in the individual mobility models (Markovian Random Walk and Gauss-Markov) and groups (with Reference Point) in ad hoc networks. The mobility model was implemented in the Castalia 3.2 simulator and the algorithm that makes the energy consumption calculations was implemented in Python. The solution was evaluated as proof of concept in 3 area distributions, with 3 relief matrices, to observe the expected behavior. The results showed that the model is fully configurable and has a good functioning, similar to the actual mobility of the animals, being able to make the expected energy consumption calculations. Also, this work presents the performance evaluation of the routing protocols: Multipath Rings (Proactive), Bypass and AODV (Reactive) and LEACH (Hierarchical). As a result, the protocol that generated the lowest power consumption was LEACH and the highest power consumption was Multipath Rings. About the number of packets received by the sink node, the lowest loss was for Multipath Rings and the highest loss was for LEACH.

Key-words: Wireless Sensor Networks. Routing Protocols. Mobility Models. Network Energy Consumption. Energy Consumption of Pasture Animals.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARC	Agricultural Research Council
AODV	Ad Hoc On-Demand Distance Vector
BAN	Body Area Network
CH	Cluster Head
EM	Energia Metabolizável
ELL	Energia Líquida de Lactação
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
GPS	Global Position System
ITU	Índice de Temperatura e Umidade
IBGE	Instituto Brasileiro de Geografia e Estatística
MANET	Mobile Ad hoc Network
MPA	Modelo Markoviano de Percurso Aleatório
NRC	National Research Council
LApIC	Laboratório de Aplicações e Inovação em Computação
LR-WPAN	Low-Rate Wireless Personal Area Network
LEACH	Low-Energy Adaptive Clustering Hierarchy
RPGM	Reference Point Group Mobility Model
RSSFs	Redes de Sensores Sem Fio
RWMM	Random Walk Mobility Model
RWPM	Random Waypoint Mobility Model
UA	Unidade Animal
UFJF	Universidade Federal de Juiz de Fora
UFMS	Universidade Federal de Mato Grosso do Sul
WSN	Wireless Sensor Networks
6LoWPAN	Over Low Power Wireless Personal Area Networks

SUMÁRIO

1 INTRODUÇÃO

Segundo o IBGE, o Brasil possui um dos maiores rebanhos bovino do mundo com 214.899.796 cabeças. A representatividade econômica da pecuária nos rebanhos de leite sempre foi enorme e a tendência é que continue expandindo. A região Centro-Oeste concentra cerca de 34,50% de todo o rebanho do Brasil, seguida da Norte com 22,55%, Sudeste com 17,46%, Nordeste com 12,91% e Sul com 12,58% [1].

Para gerenciar esses rebanhos é comum a utilização da pecuária de precisão. Ela diz respeito a inovações tecnológicas que monitoram o animal no seu ambiente, sendo uma forma eficiente de gerenciar os sistemas de produção animal. Também, ela consiste na medição de diferentes parâmetros dos animais, na modelagem desses dados para selecionar a informação desejada e no uso desses modelos em tempo real, visando o monitoramento e controle de animais e rebanhos. A base da proposição da pecuária de precisão esta no monitoramento individual e remoto dos animais e pastagens, possibilitado pelos novos avanços nas áreas de tecnologia. As aplicações dessas tecnologias são potencialmente imensas, inclusive podendo contribuir para a rastreabilidade dos produtos, uma vez que oferece registros quanto à origem dos produtos e o seu meio de produção [2].

Uma das possibilidades de monitoramento mais conhecidas é por meio do uso de sensores. Um sensor é um dispositivo que recebe e responde a sinais ou estímulos e pode ser usado para medir grandezas físicas e, caso possua um meio de transmissão, pode transmitir essas informações para outro lugar. O uso de sensores para monitoramento existe há décadas, sendo usados em aplicações de climatologia, biologia, militares e indústria [3].

Na pecuária de precisão, o GPS (*Global Position System*) é um dos sistemas mais utilizados no monitoramento animal. Embutido em um aparelho carregado pelo animal, geralmente na forma de um colar sobre o pescoço, o sensor recebe em um determinado intervalo de tempo, a posição geográfica do animal. Um conjunto de dados georreferenciados coletados por esse GPS pode, além de mostrar a trajetória do animal, revelar padrões comportamentais e novos conhecimentos a respeito dele. Por meio de análise em um conjunto de dados espaço-temporais, é possível, por exemplo, interpretar as atitudes de um bovino em uma pastagem (monitorado por um colar com um GPS embutido). Se o animal vai até o cocho para lamber sal, se frequenta certas regiões do pasto, se evita alguma região do pasto, se ele tem preferência pela sombra de uma determinada árvore, entre outros [4].

Para que os dados sejam analisados sem que haja a necessidade da coleta manual, faz-se necessário a implementação de uma rede sem fio interligando os nós sensores. Essas redes são conhecidas como Redes de Sensores Sem Fio (RSSFs) e o seu funcionamento depende do desenvolvimento de protocolos que supram suas necessidades. A avaliação de desempenho de protocolos é uma fase importante e normalmente realizada usando simulações ou

experimentos reais. Embora experimentos reais forneçam resultados confiáveis e precisos, a simulação ainda é uma alternativa interessante, pois é menos dispendiosa, é escalável a redes maiores e pode ser feita em um curto espaço de tempo. Além disso, por meio da simulação pode-se observar o comportamento do protocolo e detectar potenciais erros no processo de projeto e implementação, antes dos custos da implantação, ou seja, ajudando a reduzir os custos financeiros.

O objetivo deste trabalho foi gerar um método capaz de calcular o consumo energético de animais em pastagem baseado em um modelo de mobilidade híbrido para redes de sensores sem fio. De forma complementar, avaliar o consumo energético da rede utilizando vários protocolos de roteamento com o modelo de mobilidade criado.

Vale destacar que a solução proposta é formada por dois módulos: *Mobility Manager* e *Energy Consumption*. O módulo de mobilidade original foi desenvolvido na dissertação de Daniel Prata Borges [5]. Portanto, o presente trabalho é uma continuidade desse trabalho e apresenta como contribuições no módulo de *Mobility Manager*: a otimização dos algoritmos anteriormente desenvolvidos (corrigindo alguns erros e otimizando o código para um melhor desempenho), completude dos dados de configuração (inclusão de área de relevo, e dados temporais). Já o módulo *Energy Consumption* foi totalmente implementado no contexto desta dissertação.

Neste contexto, o principal produto gerado foi um módulo denominado *Energy Consumption*, que calcula o consumo energético de animais em pastagem utilizando um novo modelo de mobilidade híbrido configurável chamado *Mobility Manager*, baseado nos modelos de mobilidade individual (Modelo Markoviano de Percurso Aleatório [6] e no Modelo de Mobilidade Gauss-Markov [7]); e também no modelo de mobilidade em grupo, Modelo de Mobilidade em Grupo com Ponto de Referência [8]. Para modelar o comportamento dos animais escolheu-se o simulador de redes de sensores sem fio Castalia 3.2 [9], baseado na plataforma OMNeT++. E para implementar o cálculo do consumo energético escolheu-se a linguagem de programação de alto nível Python utilizando o ambiente de desenvolvimento integrado JetBrains Pycharm [10].

Para a validação do trabalho foram configuradas varias instâncias baseadas nos comportamentos do rebanho leiteiro, utilizando nessas configurações, trabalhos científicos da área para encontrar as informações relacionadas com o gado leiteiro e assim fazer a configuração dos cenários, de forma mais realística possível. Como resultados das simulações tivemos a geração dos caminhos percorridos por cada nó sensor, simulando o comportamento real de um bovino leiteiro de forma individual e em grupo. Também foram realizados os cálculos do consumo energético de gado leiteiro em cenários de pastagem totalmente plana, em um plano inclinado e com variações de picos e vales. Também são apresentados mapas de calor para verificar as áreas mais visitadas pelo rebanho leiteiro. Além disso, neste trabalho, se avaliou o desempenho de quatro tipos de protocolos

de roteamento: *Bypass*, *Multipath Rings*, *AODV* e *LEACH*. Os cenários gerados foram baseados na alteração da quantidade de nós: 10, 30 e 50 nós.

O restante do trabalho foi organizado da seguinte maneira: capítulo 2 fornece uma revisão da teoria básica para compreender o trabalho. Os trabalhos relacionados são apresentados no capítulo 3. A descrição da arquitetura proposta é apresentada no capítulo 4. Os resultados das avaliações dos testes realizados se apresentam no capítulo 5. Finalmente, o capítulo 6 apresenta as conclusões e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os conceitos fundamentais para o melhor entendimento da proposta deste trabalho. Nesse sentido, a seção 2.1 é baseada nos tópicos envolvidos com as redes de sensores sem fio (RSSFs). A descrição dos protocolos de roteamento, é realizada na seção 2.2. Já a seção 2.3 apresenta os principais modelos de mobilidade que se encontram na literatura, e se destaca o modelo utilizado. A seção 2.4 descreve os simuladores de redes sem fio utilizados atualmente pela comunidade científica. Por último, os tópicos envolvidos com o gado leiteiro são apresentados na seção 2.5.

2.1 REDES DE SENSORES SEM FIO

Uma rede de sensores é composta de um grande número de nós sensores, que são densamente implantados dentro do fenômeno ou muito próximos a ele. A posição dos nós sensores não precisa ser projetada ou pré-determinada. Isso permite a implantação aleatória em terrenos inacessíveis ou operações de socorro a desastres. Por outro lado, isso também significa que os protocolos e algoritmos de rede de sensores devem possuir recursos de auto-organização. Outra característica única das redes de sensores é o esforço cooperativo dos nós sensores. Os nós sensores são equipados com um processador integrado. Em vez de enviar os dados brutos para os nós responsáveis pela fusão, os nós sensores usam suas capacidades de processamento para realizar localmente cálculos simples e transmitir apenas os dados necessários e parcialmente processados [11].

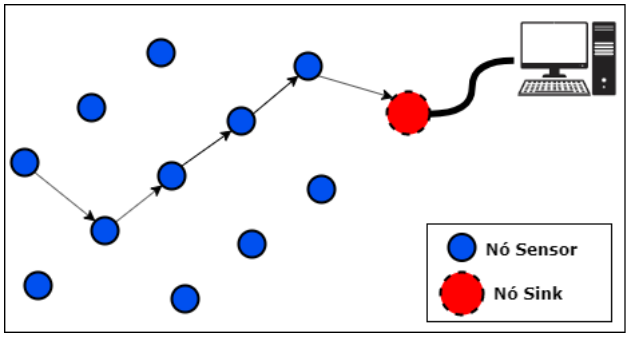
A manutenção de rede de sensores exige técnicas de rede *ad hoc* sem fio. Muitos algoritmos e protocolos foram apresentados para essas redes. Para demonstrar este ponto, as diferenças entre sistemas *ad hoc* e sistema de sensores [12] são descritas abaixo:

- O número de nós sensores em uma rede de sensores pode ser de várias ordens de magnitude e maior do que os nós em uma rede *ad hoc*.
- Nós sensores são densamente implementados.
- Nós sensores são propensos a falhas.
- Nós sensores são limitados em potência, capacidades computacionais e memória.
- Os nós sensores podem não ter identificação global (ID) devido à grande quantidade de sobrecarga e ao grande número de sensores.

As RSSFs podem ser vistas como um tipo especial de rede móvel *ad hoc* (MANET) [13]. Em uma rede tradicional, a comunicação entre os elementos computacionais é feita através de estacões base de rádio, que constituem uma infra-estrutura de comunicação.

Por outro lado, em uma rede móvel *ad hoc* os elementos computacionais trocam dados diretamente entre si. Do ponto de vista de organização, RSSFs e MANETs são idênticas, já que possuem elementos computacionais que comunicam diretamente entre si através de enlaces de comunicação sem fio. No entanto, as MANETs têm como função básica prover um suporte a comunicação entre esses elementos computacionais, que individualmente, podem estar executando tarefas distintas. Por outro lado, RSSFs tendem a executar uma função colaborativa onde os elementos (sensores) proveem dados, que são processados por nós especiais chamados de nó *Sink*.

Figura 1 – Arquitetura de uma rede de sensores sem fio.



Fonte: Elaborada pelo autor.

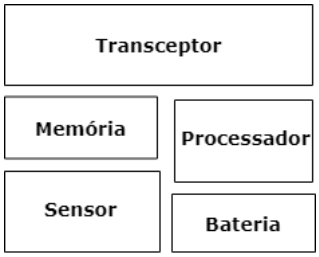
2.1.1 Componentes das Redes de Sensores Sem Fio

Os principais componentes das redes de sensores são: os nós sensores, as interfaces de comunicação sem fio e os nós *Sink* para comunicação com outras entidades [13].

- **Nós Sensores**: são dispositivos autônomos equipados com capacidades de sensoriamento, processamento e comunicação. Quando estes nós são dispostos em rede em um modo *ad hoc*, formam as redes de sensores. Os nós coletam dados via sensores, processam localmente ou, coordenadamente entre vizinhos, podendo enviar a informação para o usuário ou, em geral para um nó *Sink*. Um nó na rede tem essencialmente tarefas diferentes: sensoriamento do ambiente, processamento da informação e tarefas associadas com o trafego em um esquema de retransmissão *multi-hop*. A Figura 2 apresenta os componentes básicos de um nó sensor: transceptor, memória, processador, sensor e bateria.

- **Nós de Interface com Outras Redes**: A comunicação da rede de sensores com outras redes ocorre através de nós chamados *Sink*. As mensagens percorrem a rede de sensores até chegar a um nó *Sink* que irá encaminha-las, por uma rede como a Internet, até um computador onde roda a aplicação.

Figura 2 – Hardware básico de um nó sensor.



Fonte: Elaborada pelo autor.

2.1.2 Características das Redes de Sensores Sem Fio

As características de uma RSSF comparada com as redes de computadores tradicionais e as redes *ad hoc* são:

- Possuir um grande número de nós distribuídos.
- Ter restrições de energia, potência, capacidade computacional e de memória nos nós.
- Ser propensas a falhas.
- Alterar frequentemente a topologia da rede.
- Possuir mecanismos para auto-configuração e adaptação devido a problemas como falhas de comunicação e perda de nós.

Essas características dependem das áreas em que são aplicadas. Algumas dessas características são discutidas por Loureiro et al. [13].

- **Endereçamento dos Nós Sensores**: Dependendo da aplicação, cada sensor pode ser endereçado unicamente ou não. Por exemplo, sensores embutidos em peças numa linha de montagem ou colocados no corpo humano devem ser endereçados unicamente se deseja saber exatamente o local de onde o dado está sendo coletado. Por outro lado, sensores monitorando o ambiente numa dada região externa possivelmente não precisam ser identificados individualmente já que o ponto importante é saber o valor de uma determinada variável nessa região.
- **Agregação dos Dados**: Indica a capacidade de uma RSSF de agregar ou sumarizar dados coletados pelos sensores. Caso a rede tenha essa funcionalidade, é possível reduzir o número de mensagens que precisam ser transmitidas por ela. Os dados coletados são combinados e sumarizados ainda na rede, antes de serem enviados a estação base.

- **Mobilidade dos Sensores**: Indica se os sensores podem se mover ou não em relação ao sistema em que estão coletando dados.

- **Limitação da Energia Disponível**: Em muitas aplicações, os sensores serão colocados em áreas remotas, o que não permitirá facilmente o acesso a esses elementos para manutenção. Neste cenário, o tempo de vida de um sensor depende da quantidade de energia disponível. Aplicações, protocolos, e algoritmos para RSSFs não podem ser escolhidos considerando apenas sua capacidade, mas definitivamente a quantidade de energia consumida. Assim, o projeto de qualquer solução para esse tipo de rede deve levar em consideração o consumo, o modelo de energia e o mapa de energia da rede.

- **Auto-organização da rede**: Sensores em uma RSSF podem ser perdidos por causa de sua destruição física ou falta de energia. Também podem ficar incomunicáveis devido a problemas no canal de comunicação sem fio ou por decisão de um algoritmo de gerenciamento da rede. A situação contrária também pode acontecer: sensores inativos se tornarem ativos ou novos sensores passarem a fazer parte da rede. Em qualquer um dos casos, de sensores ficarem inoperantes ou passarem a participar de sua estrutura, é necessário haver mecanismos de auto-organização para que a rede continue a executar a sua função. Essa configuração deve ser automática e periódica já que a configuração manual não é viável devido a problemas de escalabilidade.

2.1.3 Fatores que influenciam o projeto das Redes de Sensores Sem Fio

Um projeto de rede de sensores é influenciado por muitos fatores, que incluem: tolerância a falhas, escalabilidade, custos de produção, topologia de rede de sensores e consumo de energia. Esses fatores são importantes porque servem como diretriz para projetar um protocolo ou um algoritmo para redes de sensores. Além disso, esses fatores de influência podem ser usados para comparar diferentes esquemas. Alguns desses fatores são discutidos por Akyildiz et al. [11].

- **Tolerância ao erro**: Alguns nós sensores podem falhar ou ser bloqueados devido à falta de energia, sofrer danos físicos ou interferências ambientais. A falha dos nós sensores não deve afetar a tarefa geral da rede de sensores. Esse é o problema de confiabilidade ou tolerância a falhas. A tolerância a falhas é a capacidade de sustentar as funcionalidades da rede do sensor sem qualquer interrupção devido a falhas no nó do sensor.

- **Escalabilidade**: O número de nós sensores implantados no estudo de um fenômeno pode ser da ordem de centenas ou milhares. Dependendo da aplicação, o número pode atingir um valor extremo de milhões. Os novos esquemas devem ser capazes de

trabalhar com esse número de nós. Eles também devem utilizar a natureza de alta densidade das redes de sensores. A densidade pode variar de poucos nós sensores a algumas centenas de nós sensores em uma região, que pode ter menos de 10 m de diâmetro.

- **Custos de Produção**: Como as redes de sensores consistem em um grande número de nós sensores, o custo de um único nó é muito importante para justificar o custo total das redes. Se o custo da rede é mais caro do que a implantação de sensores tradicionais, a rede de sensores não é justificada pelo custo.

- **Topologia de Rede do Sensor**: Um grande número de nós sensores, inacessíveis e não assistidos, propensos a falhas frequentes, tornam a manutenção da topologia uma tarefa desafiadora. Eles são implantados dentro de dezenas de metros um do outro. A densidade dos nós podem ser tão altas quanto 20 nós/m^3 [14]. A implantação de alto número de nós exige um tratamento cuidadoso da manutenção de topologia.

- **Consumo de Energia**: O sensor do nó sem fio, sendo um dispositivo microeletrônico, só pode ser equipado com uma fonte de energia limitada (<0,5 Ah, 1,2 V) [11]. Em alguns cenários de aplicativos, o reabastecimento de recursos de energia pode ser impossível. A vida útil do nó do sensor, portanto, mostra uma forte dependência da vida útil da bateria. Em uma rede de sensores *ad hoc multihop*, cada nó desempenha o duplo papel de originador de dados e roteador de dados. A desativação de alguns nós pode causar alterações topológicas significativas e pode exigir o reencaminhamento de pacotes e a reorganização da rede. Portanto, a conservação de energia e o gerenciamento de energia assumem importância adicional.

2.1.4 Aplicações das Redes de Sensores Sem Fio

As redes de sensores podem consistir em muitos tipos diferentes de sensores, tais como sísmica, baixa taxa de amostragem magnética, térmica, visual, infravermelha, acústica e radar, que são capazes de monitorar uma ampla variedade de condições ambientais que incluem o seguinte [15]: temperatura, umidade, movimento veicular, pressão, níveis de ruído, características atuais como velocidade, direção e tamanho de um objeto, etc.

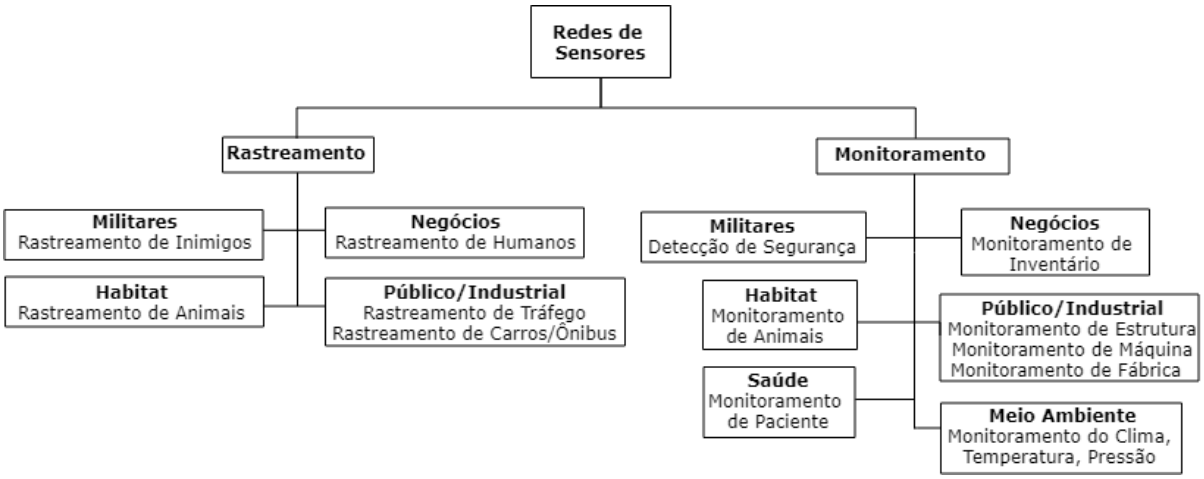
Nós sensores são amplamente utilizados para detecção sem parada, detecção de eventos, detecção de posição, entre outras.

- Aplicações Militares
- Aplicações Ambientais
- Aplicações em Saúde
- Aplicações Domésticas

- Aplicações Comerciais

As aplicações das RSSFs podem ser classificadas em duas categorias: monitoramento e rastreamento. As aplicações de monitoramento incluem monitoramento ambiental interno/externo, monitoramento de saúde e bem-estar, monitoramento de energia, monitoramento de localização de inventário, automação de fábrica e de processos e monitoramento sísmico e estrutural. As aplicações de rastreamento incluem rastreamento de objetos, animais, seres humanos e veículos. Embora existam muitas aplicações diferentes, na Figura 3 se apresentam alguns exemplos de aplicações que já foram implantadas e testadas no ambiente real [16].

Figura 3 – Visão geral de aplicações de RSSFs.



Fonte: [16].

2.1.5 **Padrões das Redes de Sensores Sem Fio**

Os padrões de sensores sem fio foram desenvolvidos com o principal requisito de baixo consumo de energia. Eles definem as funções e protocolos necessários para que os nós sensores interajam com uma variedade de redes. Alguns desses padrões incluem IEEE 802.15.4, ZigBee, IETF 6LoWPAN. Os parágrafos a seguir descrevem esses padrões em mais detalhes.

- **IEEE 802.15.4** [17] é o padrão proposto para redes de área pessoal sem fio de baixa taxa (LR-WPAN). O IEEE 802.15.4 se concentra no baixo custo de implantação, baixa complexidade e baixo consumo de energia. Foi projetado para aplicações de sensores sem fio que exigem comunicação de curto alcance para maximizar a vida útil da bateria. O padrão permite a formação da topologia em estrela e ponto-a-ponto para comunicação entre dispositivos de rede. Os dispositivos na topologia em estrela comunicar-se com um controlador central, enquanto na topologia ponto-a-ponto

podem ser formadas redes *ad hoc* e auto-configuráveis. Os dispositivos IEEE 802.15.4 são projetados para suportar os protocolos físicos e de camada de enlace de dados. A camada física suporta bandas baixas de 868/915 MHz e bandas altas de 2,4GHz. A camada MAC controla o acesso ao canal de rádio usando o mecanismo CSMA-CA. A camada MAC também é responsável por quadros de validação, entrega de quadros, interface de rede, da rede de sincronização, associação de dispositivo, e serviços seguros.

- **ZigBee** [18] define os protocolos de comunicação de camada superior construídos nos padrões IEEE 802.15.4 para LR-PAN. O ZigBee é uma tecnologia de comunicação sem fio simples, de baixo custo e baixo consumo de energia usada em aplicativos incorporados. Os dispositivos ZigBee podem formar redes mesh conectando centenas a milhares de dispositivos juntos. Os dispositivos ZigBee usam pouca energia e podem operar com uma bateria por muitos anos. Existem três tipos de dispositivos ZigBee: o coordenador ZigBee, o roteador ZigBee e o dispositivo final ZigBee. O coordenador ZigBee inicia a formação de redes, armazena informações e pode ligar redes em conjunto. Os roteadores ZigBee vinculam grupos de dispositivos e fornecem comunicação de vários saltos entre os dispositivos. O dispositivo final ZigBee consiste nos sensores, atuadores e controladores que coletam dados e se comunicam apenas com o roteador ou com o coordenador. O padrão ZigBee estava disponível publicamente a partir de junho de 2005.

- **IETF 6LoWPAN** [19] Redes pessoais sem fio de baixo consumo baseadas em IPv6, permitem a comunicação de pacotes IPv6 em uma rede baseada em IEEE 802.15.4. O dispositivo de baixa potência pode se comunicar diretamente com dispositivos IP usando protocolos baseados em IP. Usando o 6LoWPAN, os dispositivos de baixa potência têm todos os benefícios da comunicação e gerenciamento de IP. O padrão 6LoWPAN fornece uma camada de adaptação, novo formato de pacote e gerenciamento de endereço. Como os tamanhos de pacote IPv6 são muito maiores que o tamanho de quadro do IEEE 802.15.4, uma camada de adaptação é usada. A camada de adaptação realiza a funcionalidade para compactação de cabeçalho. Com a compactação de cabeçalho, pacotes menores são criados para caber em um tamanho de quadro IEEE 802.15.4. O mecanismo de gerenciamento de endereços lida com a formação de endereços de dispositivos para comunicação. O 6LoWPAN é projetado para aplicativos com dispositivos de baixa taxa de dados que exigem comunicação pela Internet.

2.2 PROTOCOLOS DE ROTEAMENTO

Os protocolos de roteamento para redes *ad hoc* podem ser categorizados em três classes: *Proativos, Reativos e Híbridos* [20]. Em geral, os protocolos diferem com base em

como eles lidam com a *descoberta de caminho* e a *manutenção da caminho*. A *descoberta de caminho* configura caminhos iniciais ou procura novos caminhos quando os antigos quebram. A *manutenção de caminhos* gerencia informações precisas sobre caminhos existentes e também suporta recuperação de erros quando um caminho quebrado é detectado.

Nos *protocolos pró-ativos*, as informações de roteamento são trocadas entre os vizinhos periodicamente ou sempre que ocorre uma mudança na topologia da rede (ou seja, a descoberta e a manutenção de caminho são realizadas continuamente). Embora os protocolos nessa categoria tenham a vantagem de que os caminhos estão imediatamente disponíveis quando solicitados, eles incorrem em alta sobrecarga de controle. Os *protocolos híbridos* reduzem a sobrecarga de controle equilibrando a proatividade e a reatividade, todos os protocolos híbridos atuais dependem da aquisição proativa de pelo menos uma informação de vizinhança de um salto para todos os nós. Os *protocolos reativos* reduzem a sobrecarga de roteamento vinculando a descoberta de caminhos à comunicação de rede. Esses protocolos iniciam uma descoberta de caminho somente quando um novo caminho é necessária para a configuração inicial do caminho ou devido a um caminho quebrado. A descoberta de caminho é alcançada inundando a rede, o que causa alta sobrecarga de roteamento e interferência no tráfego em andamento. Se um caminho quebrado puder ser reparado, a descoberta de caminho da origem, que causa a inundação em toda a rede, não será mais necessária. No entanto, é importante garantir que qualquer técnica de recuperação de caminho custa menos em termos de sobrecarga de controle e atraso do que uma nova configuração de caminho via inundação.

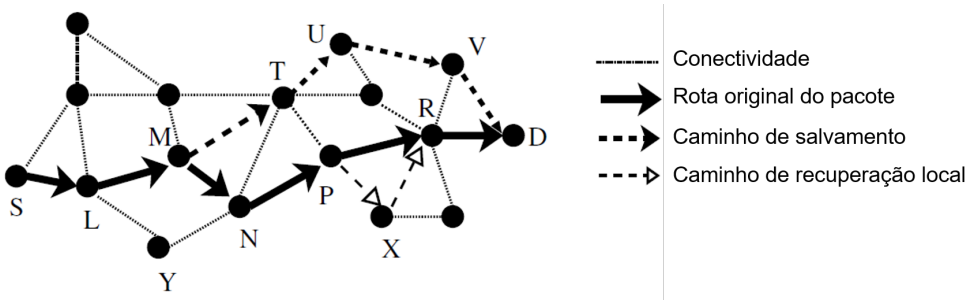
Neste trabalho foram analisados 4 protocolos de roteamento, levando em consideração a categorização deles existente na literatura. Primeiramente foram utilizados os dois protocolos de roteamento que já estão implementados no próprio simulador Castalia: Bypass (Reativo) e Multipath Rings (Proativo). Além disso, foi utilizado o protocolo de roteamento AODV (Reativo), por ser utilizado nas redes com nós móveis, e o protocolo de roteamento LEACH (Hierárquico), por estar baseado em estrutura de *cluster* que permite implementar a conservação de energia em grande escala.

2.2.1 **Protocolo de Roteamento - Bypass**

O roteamento de bypass [20] executa a recuperação de caminhos sob demanda, utilizando caches de caminho e recuperação de erros locais. Essencialmente, para recuperar uma falha de caminho, um nó primeiro salva um caminho pesquisando seu cache de caminho para um caminho alternativo até o destino (se o nó armazenar em cache vários caminhos). Se houver um caminho, o nó corrige o caminho quebrado com o caminho alternativo. Se o nó não conseguir reparar o caminho de seu cache de caminho, ele iniciará a recuperação de bypass consultando seus vizinhos para ver se eles têm um *link* para qualquer nó no caminho *downstream* para o destino (por exemplo, o próximo salto ou todo

os nós *downstream* no caso de caminhos de origem). Conforme as respostas chegam, o nó repara os caminhos afetados pela falha do *link* com as informações de conectividade recebidas. Quando esses pacotes chegam ao destino, as novas informações de caminho são adicionadas a um pacote de erros de caminho aprimorado e enviadas de volta à fonte para informá-lo sobre o *link* quebrado e a mudança de caminho bem-sucedida.

Figura 4 – Funcionamento do Protocolo de Roteamento Bypass.



Fonte: [20].

Embora o roteamento de bypass não esteja limitado a nenhum protocolo de roteamento, as especificações do mecanismo dependem das características do protocolo de roteamento subjacente sob demanda. A Figura 4 mostra o roteamento de bypass integrado a um protocolo de roteamento de origem. Inicialmente, o fluxo do nó S para o nó D usa o caminho "S – L – M – N – P – R – D". Quando o *link* entre o nó M e o nó N é interrompido, o nó M detecta a falha e tenta corrigir o caminho usando um caminho alternativo de seu cache para o destino D. Quando o nó M encontra um caminho sem *loops*, o nó M recupera o pacote com o caminho "S – L – M – T – U – V – D". A Figura 4 também ilustra um exemplo de recuperação de bypass. Novamente, o nó S inicialmente usa o caminho "S – L – M – N – P – R – D". Quando o *link* entre o nó P e o nó R é interrompido e o nó P não tem um caminho alternativo em seu cache para o destino, o nó P aciona uma consulta local para seus vizinhos. Os vizinhos respondem se possuem *links* ativos para qualquer um dos nós de recebimento de dados no caminho quebrado. Na Figura 4, o nó X relata sua conectividade com o nó R ao nó P. O nó P corrige o caminho de acordo e o pacote é primeiro encaminhado para o nó X e depois para o nó R para alcançar o destino.

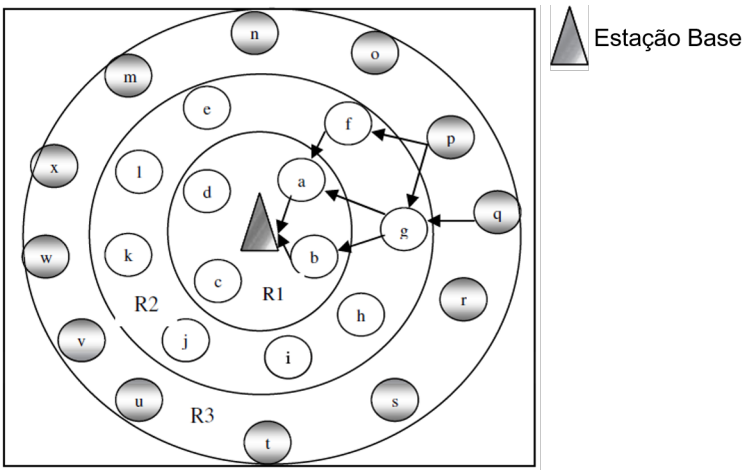
Esses exemplos mostram que o reparo local de caminhos quebrados pode resultar em um aumento nos comprimentos dos caminhos. No entanto, as fontes informadas sobre as informações de reparo não são forçadas a usar um caminho mais longo se conhecerem um caminho mais curto. Especificamente, a recuperação de desvio visa reduzir a frequência das descobertas de caminho, permitindo que o nó use caminhos mais curtos quando possível.

2.2.2 **Protocolo de Roteamento - Multipath Rings**

O protocolo de roteamento Multipath Rings é adotado de uma técnica semelhante por Nath et al. [21]. No roteamento, os nós não têm um pai específico. Um nó apenas recebe um número de nível do anel durante a formação da topologia. O número do anel indica a distância do salto do nó de origem do nó *Sink*. O primeiro pacote de configuração de topologia enviado da estação base possui o número de anel *0*. Qualquer nó que receba esse pacote de configuração da topologia, ele aumentará em *1* o número de anel do pacote recebido e retransmitirá-o. Esse processo continua até que todos os pacotes recebam o número do anel. Eventualmente, todos os nós conectados terão um número de anel.

Depois que a fase de configuração da topologia é concluída, quando um nó de origem deseja relatar um dado à estação base, ele não o envia para um determinado nó, mas o transmite, anexando seu número de anel. Qualquer nó que tenha um número de anel menor receberá esse pacote e o retransmitirá. O processo continua até o pacote chegar à estação base [22].

Figura 5 – Protocolo de Roteamento Multipath Rings.



Fonte: [23].

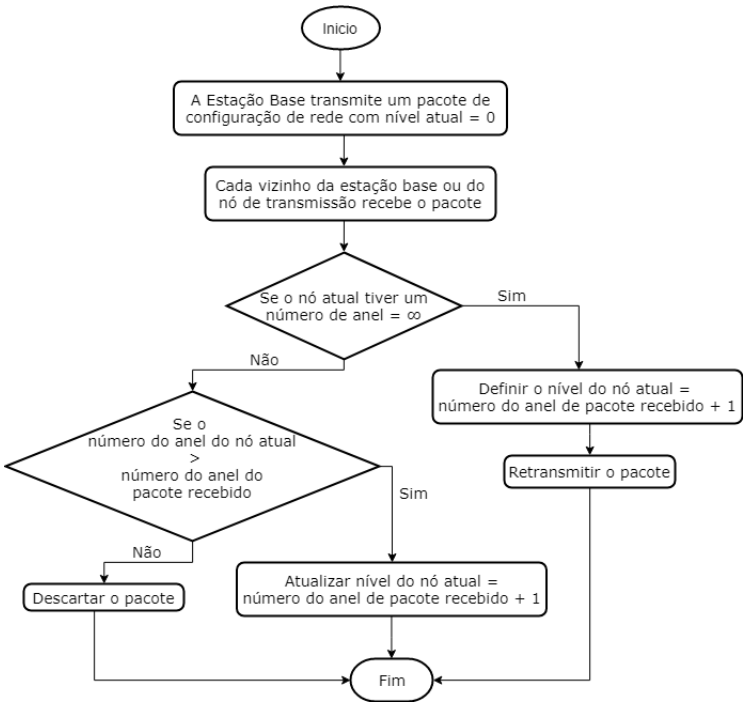
O protocolo de roteamento Multipath Rings é um tipo de protocolo de roteamento proativo porque na inicialização da rede é feita antes da disseminação de dados e a descoberta de caminhos também não é necessária antes da transmissão de dados [24]. No protocolo de roteamento Multipath Rings, é assumido que todos os nós são distribuídos aleatoriamente para detectar o campo do sensor e a estação base, que é responsável por coletar dados de todos os nós sensores e reportar à estação de coleta de dados. Também é assumido que cada nó sensor possui bateria própria para alimentação e não pode ser substituído ou recarregado.

Segundo Pandya e Mehta [23] o protocolo de roteamento Multipath Rings pode ser dividido em duas etapas básicas:

1. **Fase de configuração da topologia**: A ideia básica da fase de configuração da topologia é organizar a rede de acordo com a distância de salto de cada nó do nó *Sink*, ou seja, no final dessa fase, cada nó virá a saber que salto há.

   Na fase de configuração da topologia, o pacote de configuração de rede de transmissão da estação base com o número de anel atual está definido como *Zero*. O parâmetro do número do anel é usado para definir a distância do salto. Os nós que estão próximos da estação base podem receber este pacote de configuração de topologia. Ao receber este pacote, o nó receptor incrementará o número de anel do pacote em *1* e retransmitirá o pacote e também definirá seu próprio número de anel igual ao número do anel de pacote recebido mais *1*. Portanto, todos os nós que são vizinhos diretos do nó *Sink* estarão no anel 1 e o vizinho direto dos nós no anel *1* ficará no anel *2* e assim por diante. Em geral, se o nó estiver afastado do nó *Sink*, esse nó ficará sob o *anel n*. O processo mostrado na Figura 6 continuará até que todos os nós recebam seus números de anel.

Figura 6 – Fluxograma da fase de configuração da topologia.



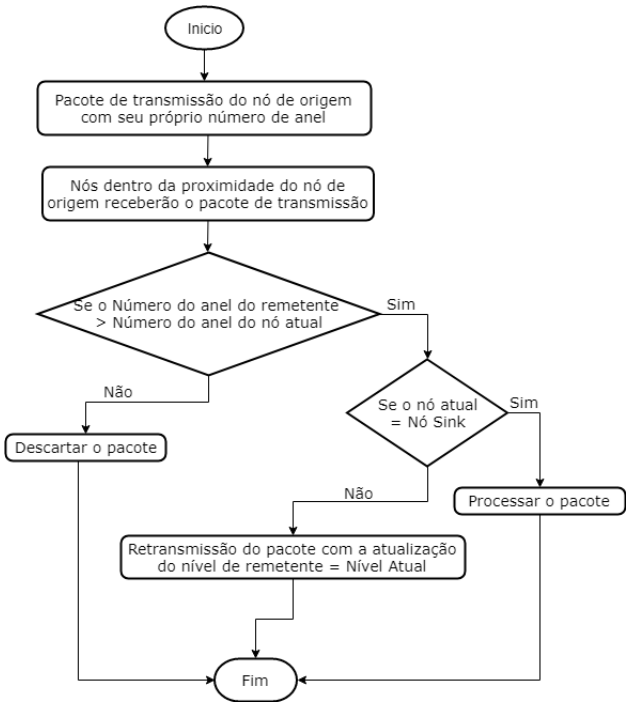
Fonte: [23].

2. **Fase de disseminação de dados**: Após a conclusão da fase de configuração da topologia, a transmissão de dados do nó de origem para o nó de recebimento pode ocorrer. Na rede de sensores sem fio, como os nós sensores não têm endereço global, quando qualquer nó deseja enviar seus dados para o coletor, ele não envia para um determinado nó, mas transmite o pacote com seu número de anel atual. Qualquer nó que tenha um valor de número de anel atual inferior ao número do anel recebido,

ele poderá retransmitir o pacote recebido. Este processo continuará até que o pacote alcance a estação base.

A Figura 7 mostra os fluxos da fase de transferência de dados no protocolo de roteamento Multipath Rings. Nesta fase, quando qualquer nó quiser enviar dados para o nó *Sink*, ele transmitirá os dados com seu número de anel atual. Assim, todo vizinho do nó de origem pode ouvir esse pacote e verificar o nível de anel desse pacote se o pacote tiver um número de anel mais alto que o nó de recebimento, então somente o nó processará esse pacote caso contrário, ele descartará esse pacote diretamente. Se o nó receptor for nó *Sink*, ele receberá esse pacote e o processará, mas se o nó receptor não for nó *Sink*, então o nó receptor definirá o número do anel do pacote como igual ao seu próprio número de anel e retransmitirá o pacote.

Figura 7 – Fluxograma da fase de disseminação de dados.



Fonte: [23].

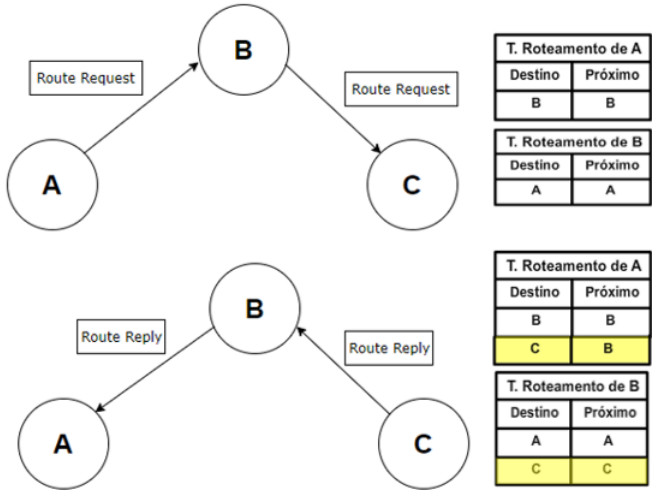
2.2.3 **Protocolo de Roteamento - AODV**

O protocolo de roteamento Ad Hoc On-Demand Distance Vector (AODV) é destinado ao uso por nós móveis em uma rede *ad hoc*. Ele oferece rápida adaptação às condições de enlace dinâmico, baixo custo de processamento e memória, baixa utilização da rede e determina rotas de unicast para destinos dentro da rede *ad hoc*. Ele usa números de sequência de destino para garantir a liberdade de loop em todos os momentos (mesmo em face da entrega anômala de mensagens de controle de roteamento), evitando problemas (como “contar até infinito”) associados a protocolos de vetor de distância clássicos [25].

AODV fornece recuperação de rota permitindo que os nós de relé iniciem uma busca para substituir uma rota falhada. No AODV, se um nó não estiver mais longe do que o reparo máximo se afastar do destino, ele tentará reparar a rota transmitindo uma solicitação de rota com um tempo de vida limitado.

São utilizadas tabelas de roteamento tradicionais, ou seja, é armazenado apenas o próximo salto para o nó destino em questão. Sendo um protocolo reativo, quando é necessário o envio de pacotes a um nó destino que não consta em sua tabela de roteamento, é dado início ao Processo de Descoberta de Rotas, cujo funcionamento é mostrado na Figura 8.

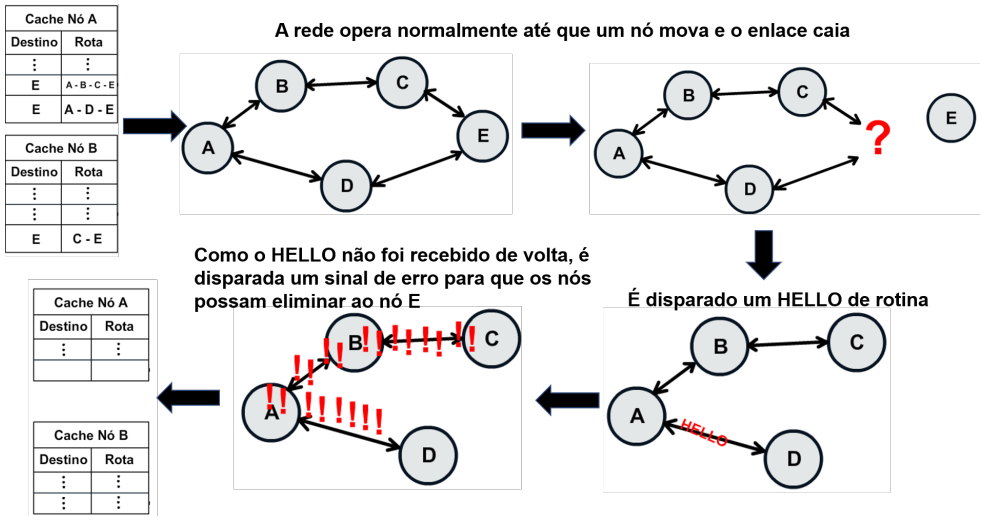
Figura 8 – Processo de Descoberta de Rotas no Protocolo de Roteamento AODV.



Fonte: Elaborada pelo autor.

No protocolo AODV, é importante ressaltar a importância da Manutenção de Rotas. Vista como um sub-protocolo do AODV, tem o objetivo de validar as rotas contidas no cache. Como os nós são móveis, sua movimentação pode ocasionar a ruptura de uma ligação antes existente e presente nas tabelas de roteamento dos nós. Pela manutenção de rotas, periodicamente são enviados pacotes chamados *HELLO* entre os nós para se verificar a existência e ruptura de rotas. Quando um desses pacotes é esperado por algum nó mas não é recebido, este detecta o erro de transmissão (por exemplo, uma quebra de ligação) e, então, é enviado imediatamente um pacote sinalizador de erro *RERR* de volta para o nó de origem e para todos os nós que forem necessários avisando sobre o erro. Um nó, ao receber este pacote, tem seu cache atualizado, de forma a remover de suas rotas armazenadas todas as sub-rotas posteriores às ligações quebradas. Na Figura 9 se mostra o funcionamento da manutenção de rotas.

Figura 9 – Processo de Manutenção de Rotas no Protocolo de Roteamento AODV.

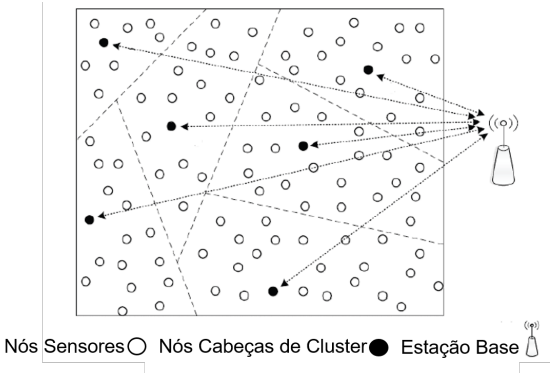


Fonte: Elaborada pelo autor.

2.2.4 **Protocolo de Roteamento - LEACH**

LEACH (Hierarquia de Clusters Adaptativos de Baixa Energia) [26] é um protocolo baseado em estrutura de classe de cluster e tecnologia em camadas que pode implementar a conservação de energia em grande escala em comparação com os protocolos tradicionais. No LEACH, todos os nós da rede são divididos em vários clusters, e os nós cabeças de cluster são selecionados aleatoriamente para a distribuição equitativa das tarefas de comunicação. Ele funciona sem quaisquer nós centrais para controlar a seleção ou para coordenar o processo de seleção, portanto, o algoritmo de seleção de cabeçalhos de cluster do LEACH é um algoritmo distribuído, no qual cada nó decide independentemente se deve se tornar um nó cabeça de cluster. Isso pode resultar em que os nós com pouca energia residual sejam escolhidos como nós principais do cluster, o que acelera a morte desses nós, encurtando assim toda a vida útil da rede.

Figura 10 – Protocolo de Roteamento LEACH.



Fonte: [26].

O protocolo LEACH, que consiste no estágio de agrupamento e no estágio de transmissão estável, é executado por rodadas. No estágio de agrupamento, os nós cabeças de cluster (CHs) são selecionados de todos os nós.

$$P(n) = \begin{cases} \frac{p}{1-p\left(r mod \frac{1}{p}\right)} \leftarrow if\ (n \in G) \\ 0 \leftarrow else \end{cases} \tag{2.1}$$

Onde p é a porcentagem de CHs, r é o número atual de rodadas, $n = 1, 2, 3, 4 \ldots N$ é o nó na rede e G é uma coleção de nós que não atuam como CHs para o próximo $1/p$ rodadas. Depois que os CHs são determinados, todos os CHs transmitem um pacote de sinais para toda a rede. Então, o nó comum envia o pacote para o CH mais próximo, de acordo com a intensidade do sinal, para unir o cluster independentemente.

No estágio de transmissão estável, os CHs recebem os dados dos nós no cluster e, em seguida, enviam os dados para a estação base após a fusão de dados.

As características típicas do LEACH:

- Estabelecimento e operação da classe de cluster através da colaboração e controle locais.
- Mudança dinâmica da classe do cluster e a aleatoriedade dos nós principais do cluster (rotação da cabeça do cluster).
- Agregação de dados local para reduzir o tráfego geral.

As suposições LEACH baseadas em:

- Todos os nós da rede são idênticos.
- Todos os nós adotam a mesma antena omnidirecional.
- O nó *Sink* está fixo e longe da rede de sensores sem fio.
- O canal é simétrico, ou seja, o consumo de energia do envio de dados do nó A para o nó B é igual ao consumo do nó B para o nó A.

2.3 MODELOS DE MOBILIDADE

Os modelos de mobilidade visam representar o comportamento de movimentação dos nós móveis em uma rede de sensores. Esses modelos são usados na avaliação do desempenho de aplicações e sistemas de comunicação, permitindo analisar o impacto causado pela mobilidade no funcionamento dos mesmos. E são classificados em dois tipos: modelos de mobilidade individual e modelos de mobilidade em grupo.

2.3.1 **Modelos de Mobilidade Individual**

Os modelos de mobilidade individual são modelos que representam o comportamento de movimentação de um nó móvel de forma independente do restante dos nós móveis da rede. Consiste em uma modelagem mais simples e de fácil implementação. Eles têm importante aplicação na predição da disponibilidade dos enlaces sem fio, afim de melhorar, como por exemplo, a eficiência dos algoritmos de roteamento e construção de rotas mais estáveis. A causa de mudanças na disponibilidade dos enlaces sem fio são as mudanças locais na topologia da rede [6].

2.3.1.1 *Modelo de Mobilidade Aleatória (RWMM)*

O modelo de mobilidade aleatória [27] foi descrito pela primeira vez matematicamente por Einstein em 1926 [28]. É o modelo de mobilidade de percurso aleatório e é um dos modelos mais utilizados pelos pesquisadores para avaliação de protocolos em redes de sensores. As características da direção e da velocidade do movimento em um novo instante de tempo não tem relação nenhuma com os valores de instantes anteriores, ou seja, é um modelo de mobilidade sem memória. Por isso, esse modelo pode gerar um comportamento não realístico, por exemplo com mudanças bruscas de direção, paradas abruptas e acelerações bruscas no movimento dos nós móveis.

2.3.1.2 *Modelo de Mobilidade Waypoint (RWPM)*

O modelo de mobilidade waypoint [29] divide o percurso de um nó móvel em períodos de movimentação e pausa. O nó móvel fica em um local por um determinado intervalo de tempo e depois move-se para um novo local escolhido aleatoriamente com uma velocidade que segue uma distribuição uniforme entre [Velocidade Miníma, Velocidade Máxima]. Esse modelo também é sem memória, com isso ele possui as desvantagens que têm o modelo de mobilidade aleatória. Apesar disso, ele também é um modelo de mobilidade muito usado pelos pesquisadores devido a sua simplicidade de implementação. Através da configuração do intervalo do tempo de pausa em zero, esse modelo se comporta de forma semelhante ao modelo de mobilidade aleatória.

2.3.1.3 *Modelo Markoviano de Percurso Aleatório (MPA)*

O modelo markoviano de percurso aleatório foi proposto por Chiang [30], onde o movimento é modelado através de uma Cadeia de Markov, e portanto, ele é um modelo de mobilidade com memória. Esse modelo possui três estados para representar as coordenadas x e y. O estado zero (0) representa a posição atual, o estado um (1) representa a posição anterior e o estado dois (2) representa a próxima posição, em x e y. O modelo utiliza a matriz 2.2 de probabilidade de transição para determinar a posição de um nó móvel

especifico no próximo instante de tempo. Nessa matriz, cada elemento $P(a, b)$ representa a probabilidade de mudança do estado a para o estado b.

$$P = \begin{bmatrix} P(0,0) & P(0,1) & P(0,2) \\ P(1,0) & P(1,1) & P(1,2) \\ P(2,0) & P(2,1) & P(2,2) \end{bmatrix} \tag{2.2}$$

Esse modelo tem a característica que ele não permite mudanças bruscas de sentido no movimento, pois para o nó móvel mudar o sentido do movimento é necessário parar o movimento para depois mudar de sentido. Porém, esse modelo é mais realista que os modelos de mobilidade Aleatória e Waypoint.

2.3.1.4 *Modelo de Mobilidade Gauss-Markov*

O modelo de mobilidade de Gauss-Markov foi projetado para se adaptar a diferentes níveis de aleatoriedade por meio de um parâmetro de ajuste [7]. Inicialmente, a cada nó móvel é atribuída uma velocidade e direção atuais. Em intervalos fixos de tempo n, o movimento ocorre pela atualização da velocidade e direção de cada nó móvel. Especificamente, o valor da velocidade e da direção na instância n^{th} é calculada com base no valor da velocidade e da direção na instância $(n-1)^{st}$, e uma variável aleatória usando as equações 2.3 e 2.4.

$$s_n = \alpha s_{n-1} + (1-\alpha)\bar{s} + \sqrt{(1-\alpha^2)s_{x_n-1}} \tag{2.3}$$

$$d_n = \alpha d_{n-1} + (1-\alpha)\bar{d} + \sqrt{(1-\alpha^2)d_{x_n-1}} \tag{2.4}$$

Onde:

- s_n e d_n são a nova velocidade e direção do nó móvel no intervalo de tempo n.
- $0 \leq \alpha \leq 1$ é o parâmetro de sintonização usado para variar a aleatoriedade.
- $\bar{s}$ e $\bar{d}$ são constantes representando o valor médio de velocidade e direção como $n \to \infty$
- s_{x_n-1} e d_{x_n-1} são variáveis aleatórias de uma distribuição gaussiana.

A cada intervalo de tempo, a próxima localização é calculada com base na localização atual, velocidade e direção do movimento. Especificamente, no intervalo de tempo n, a posição de um nó móvel é dada pelas equações 2.5 e 2.6.

$$x_n = x_{n-1} + s_{n-1}\cos d_{n-1} \tag{2.5}$$

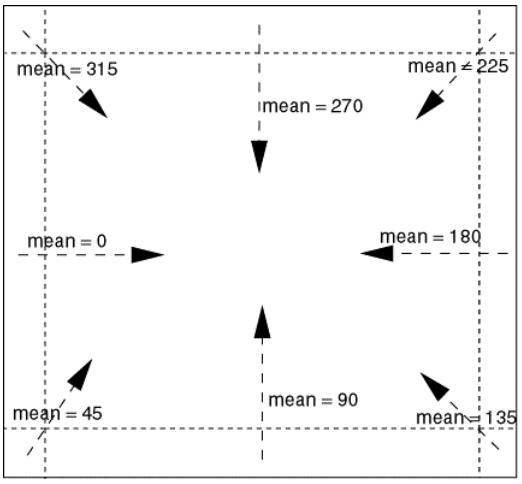
$$y_n = y_{n-1} + s_{n-1} \sin d_{n-1} \tag{2.6}$$

Onde:

- x_n e y_n são as coordenadas x e y da posição dos nós móveis no n^{th}.
- x_{n-1} e y_{n-1} são as coordenadas x e y da posição dos nós móveis no $(n-1)^{st}$.
- s_{n-1} e d_{n-1} são a velocidade e a direção do nó móvel, no intervalo de tempo $(n-1)^{st}$.

Para garantir que um nó móvel não permaneça próximo a uma borda da grade por um longo período de tempo, os nós móveis são forçados para longe de uma borda quando se movem dentro de uma certa distância da borda. Isso é feito modificando a variável de direção média d na equação 2.4. Os valores da direção média para diferentes locais na grade de simulação são mostrados na figura 11.

Figura 11 – Mudança do ângulo médio perto das bordas.



Fonte: [7]

2.3.2 **Modelos de Mobilidade em Grupo**

Os modelos de mobilidade em grupo são modelos que representam o movimento de um grupo de nós móveis, onde o comportamento de movimentação dos nós móveis é dependente, tanto em relação a intervalos de tempo quanto ao relacionamento entre eles [6]. Além desses modelos serem utilizados para predizerem a disponibilidade do enlace sem fio, desde 2002 estão sendo usados também na predição do particionamento das redes, que são mudanças de escala global na topologia da rede [31]. A principal causa desse particionamento é o próprio comportamento de movimentação em grupo dos nós móveis.

Em uma rede de sensores há muitas situações em que é necessário modelar o comportamento dos nós móveis à medida que eles se movem juntos. Por exemplo, um

indivíduo é tido como líder do grupo e os outros nós irão acompanhá-lo na maioria do tempo. Para modelar tais situações, é necessário um modelo de mobilidade de grupo para simular essa característica cooperativa. Existem vários modelos de mobilidade em grupo: Modelo de Mobilidade Aleatória Correlacionada Exponencial, Modelo de Mobilidade de Coluna, Modelo de Mobilidade Comunitária Nômade, Modelo de Mobilidade Perseguida, Modelo de Mobilidade em Grupo de Ponto de Referência, entre outros. O mais geral desses modelos é o modelo Mobilidade em Grupo com Ponto de Referência (RPGM) e foi utilizado neste trabalho.

2.3.2.1 *Modelo de Mobilidade em Grupo com Ponto de Referência (RPGM)*

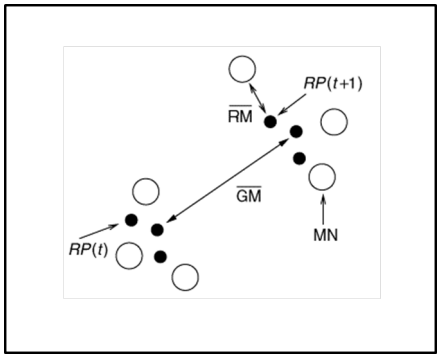
O modelo foi desenvolvido por Hong [8]. Cada grupo tem um centro de referência lógico, e o movimento do centro define o comportamento de todo o grupo incluindo localização, velocidade, direção, aceleração, etc. Desse modo, a trajetória do grupo é composta por um caminho até o centro. Os nós móveis são distribuídos uniformemente dentro da área geográfica do grupo. Para cada nó móvel, é atribuído um ponto de referência, ao qual, segue o movimento do grupo. Um nó móvel é colocado aleatoriamente na vizinhança do ponto de referência. O esquema de referência em um ponto admite o comportamento de movimentação aleatória independente para cada nó móvel, além do movimento do grupo.

A Figura 12 mostra uma ilustração de três nós móveis movendo-se com o modelo RPGM. No tempo t existem três pontos pretos para representar os pontos de referência $RP(t)$ para os três nós móveis. O modelo RPGM usa uma imagem de equação vetorial de movimento de grupo para calcular o novo ponto de referência de cada nó móvel $RP(t+1)$ no tempo $t+1$. A imagem da equação pode ser escolhida aleatoriamente ou predefinida. A nova posição para cada nó móvel é calculada pela soma de um vetor de movimento aleatório, imagem de equação, com o novo ponto de referência. O comprimento da imagem da equação é distribuído uniformemente dentro de um raio especificado centrado em $RP(t+1)$ e sua direção é distribuída uniformemente entre 0 e 2π.

2.4 SIMULADORES DE REDES DE SENSORES SEM FIO

Nos últimos anos, o contínuo interesse em redes de sensores sem fio (*RSSF's*) levou ao surgimento de novos métodos de modelagem e ambientes de simulação para aplicativos *RSSF's*. Uma ampla variedade de diferentes ferramentas de simulação foi projetada para explorar e validar os sistemas *RSSF's* antes da implementação real e da implantação no mundo real. Essas ferramentas abordam diferentes aspectos do projeto e oferecem várias abstrações de simulação para representar e modelar o comportamento do mundo real [32]. Neste âmbito, têm-se os simuladores de redes, que além de diminuírem a complexidade dos experimentos, possuem uma série de vantagens, entre elas: baixo custo

Figura 12 – Movimentos de três nós móveis usando o modelo RPGM.



Fonte: [7].

de implementação, ambiente controlado, facilidade na coleta de dados, universalização da informação e compartilhamento de experimentos e resultados.

Hoje em dia existem vários simuladores para RSSF's, tendo três categorias: Simuladores de Rede Genéricos, Simuladores Orientados à Rede e Simuladores de Nó Sensores.

2.4.1 Simuladores de Rede Genéricos

- **NS-2 (Network Simulator 2)** [33]: é um dos mais populares simuladores de rede de código aberto de uso geral. Ele fornece uma ampla variedade de modelos de simulação para protocolos de rede IP amplamente usados. Isso inclui protocolos TCP/IP, roteamento e multicasting para redes convencionais com e sem fio. O NS-2 ganhou popularidade graças à sua arquitetura orientada a objetos altamente extensível baseada em um mecanismo de eventos discretos. Isso permite que os usuários estendam a funcionalidade de simulação adicionando componentes e bibliotecas personalizados. Assim, muitas implementações de terceiros de protocolos de comunicação, modelos e complementos estão disponíveis no NS-2.

- **NS-3 (Network Simulator 3)** [34]: é uma versão recente da família de Network Simulator (NS) que se destina a substituir o NS-2. O NS-3 é uma ferramenta poderosa para modelagem e otimização de redes. Ele inclui modelos para os protocolos mais populares, incluindo API de soquete, TCP/IP, IPv6, roteamento MANET, IEEE802.11, WiMAX e assim por diante. Como o NS-2, o NS-3 também possui uma arquitetura modular orientada a objetos, combinada com um mecanismo de simulação de evento discreto escrito em C++. Devido à relativa novidade da ferramenta, não há muitas extensões disponíveis, especialmente aquelas adequadas para aplicativos WSN, e fornece consideravelmente menos modelos disponíveis do que seu predecessor.

- **OMNeT++** [35]: é uma biblioteca e estrutura de simulação C++ baseada em componentes que fornece uma análise profunda das atividades da rede. O OM-

NeT++ fornece um *front-end* de GUI para configurações de simulação e análise de interpretação de resultados. Ele explora módulos e canais para implementar e conectar componentes de simulação, onde os componentes são conectados de forma hierárquica através de interfaces genéricas. O OMNeT++ tornou-se uma estrutura de simulador popular devido à sua arquitetura simples e pura e várias extensões para sistemas convencionais e WSN.

2.4.2 Simuladores Orientados à Rede

- **Castalia** [36]: é um simulador WSN popular baseado no framework OMNeT++. Ele fornece uma análise de primeira ordem de algoritmos e protocolos antes da implementação real em uma plataforma de nó específica. Tem uma forte característica que é a modelagem detalhada de rádio e comunicação. Ele inclui um modelo de rádio-frequência ajustável e altamente preciso (PHY), juntamente com um modelo de canal de comunicação avançado baseado em dados medidos empiricamente. Para lidar com a entrega de pacotes, o Castalia fornece o cálculo dinâmico da taxa de interferência e ruído do sinal de rádio (SINR) com base no tipo de modulação, intensidade do sinal e outros parâmetros. O modelo PHY inclui múltiplos estados com atrasos de transição entre eles, diferentes níveis de transmissão e modulações, e RSSI (Received Signal Strength Indicator) configurável e CCA (Clear Channel Assessment). Além disso, o Castalia permite modelar unidades de rede móvel, inclui modelos personalizáveis dos mais populares MAC e protocolos de roteamento para aplicativos WSN. Ele também fornece modelos de sensores, incluindo efeitos como detecção de ruído e distorção do dispositivo. O Castalia oferece relatórios de modelagem e consumo de energia para componentes de rádio e pode suportar modelagem simples de desvio de relógio para CPUs.

- **MiXim** [37]: é um ambiente de simulação baseado no OMNeT++. É um projeto conjunto que combina várias ferramentas de simulação projetadas para simulações móveis e sem fio. Fornece modelos detalhados do canal de comunicação e da camada PHY do rádio. Ele oferece vários modelos de propagação de sinal de rádio, incluindo um modelo de nível de Relação Sinal-Ruído (SNR) variante no tempo, um modelo para perda de trajetória, sombreamento e modelos de desvanecimento de grande e pequena escala. A camada PHY do MiXiM oferece um alto nível de personalização em termos de tipos de modulação, sensibilidade, potência do sinal de saída e estados operacionais de hardware de rádio com parâmetros de potência e temporização. O MiXiM oferece modelagem e relatórios de energia para módulos de rádio. Além disso, as redes móveis também são suportadas. Em geral, esta ferramenta tem muito em comum com o Castalia. Como uma desvantagem relativa do MiXiM, ele realmente não possui modelos para componentes periféricos de hardware típicos de nós WSN e

uma ferramenta de representação gráfica para resultados de simulação ainda está ausente.

2.4.3 Simuladores de Nó Sensores

- **TOSSIM** [38]: O ambiente de simulação TOSSIM está incluído no TinyOS. Ele explora o modelo de componente do TinyOS e é totalmente integrado a ele. Este simulador fornece simulação em nível de código de aplicativos TinyOS que podem ser executados em hardware de rede de sensores reais. Ao substituir um pequeno número de componentes do TinyOS, o TOSSIM simula o comportamento do hardware de baixo nível. Inclui modelos para a CPU, ADCs, relógios, temporizadores e componentes de rádio. A arquitetura de simulação TOSSIM oferece um alto nível de escalabilidade e velocidade de execução para redes com um grande número de nós sensores. No entanto, o modelo de hardware abstrato disponível no TOSSIM não captura detalhes de baixo nível de temporização e interrupções, o que pode ser importante para uma análise precisa do poder do tempo. A simulação de rede no TOSSIM é limitada pela homogeneidade do modelo de execução. É capaz de executar apenas o mesmo programa em cada nó da rede. Além disso, o modelo de simulação TOSSIM não é extensível e suporta apenas um modelo de plataforma de hardware (MicaZ).

- **COOJA** [39]: é uma estrutura de simulação para o Contiki [40] sistema operacional do nó sensor. É escrita como ferramenta independente baseada em Java e permite que as WSNs sejam simuladas tanto no sistema operacional (código) quanto no nível de instrução da máquina. O COOJA fornece uma simulação em nível de aplicativo para redes que consistem em nós executando o sistema operacional Contiki. O COOJA opera de maneira semelhante ao funcionamento do TOSSIM para aplicativos TinyOs.

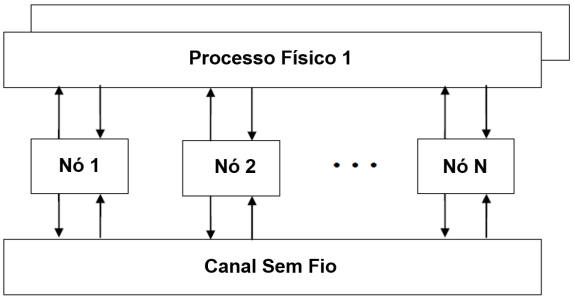
2.4.4 Simulador Castalia

O Castalia é um simulador para Redes de Sensores Sem Fio (WSN), Redes de Área Corporal (BAN) e geralmente redes de dispositivos embarcados de baixa potência. Ele é baseado na plataforma OMNeT++ e pode ser usado por pesquisadores e desenvolvedores que desejam testar seus algoritmos ou protocolos distribuídos em modelos de rádio e canal sem fio realistas, com um comportamento de nó realista, especialmente relacionado ao acesso do rádio. O Castalia também pode ser usado para avaliar diferentes características de plataforma para aplicações específicas, uma vez que é altamente paramétrico e pode simular uma ampla gama de plataformas [9].

A estrutura básica do módulo do Castalia é mostrada na Figura 13. Onde os nós não se conectam entre si diretamente, mas através dos módulos do canal sem fio. As setas

significam uma mensagem passando de um módulo para outro. Quando um nó tem um pacote para enviar, ele vai para o canal sem fio, que decide quais nós devem receber o pacote. Os nós também são vinculados por meio dos processos físicos que eles monitoram. Para cada processo físico, existe um módulo que contém a "autenticidade" da quantidade que o processo físico está representando. Os nós amostram o processo físico no espaço e no tempo (enviando uma mensagem ao módulo correspondente) para obter as leituras do sensor. Pode haver vários processos físicos, representando os múltiplos dispositivos de detecção (múltiplas modalidades de detecção) que um nó possui.

Figura 13 – Os módulos e suas conexões no Castalia.

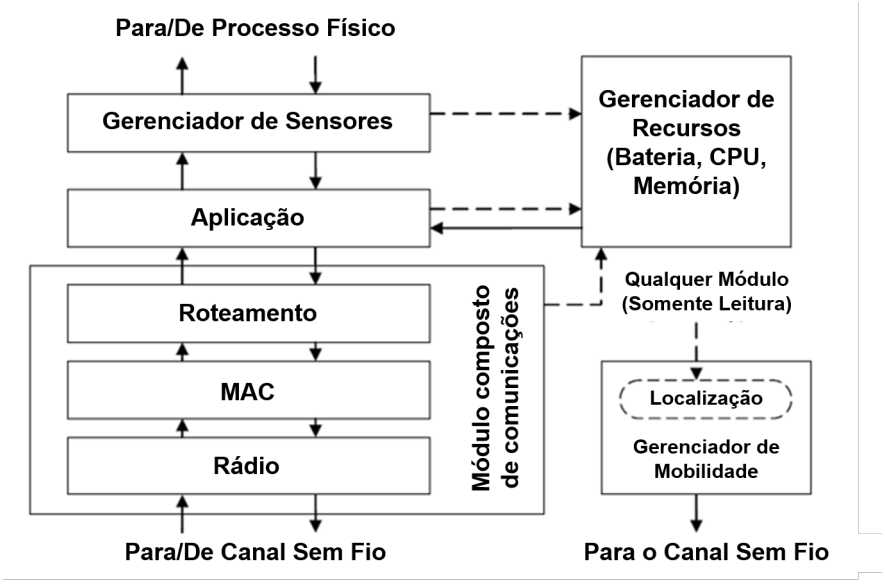


Fonte: [9].

A Figura 14 mostra a estrutura interna do módulo composto do nó. As setas sólidas significam passagem de mensagem e as setas tracejadas significam chamadas de função simples. Por exemplo, a maioria dos módulos chama uma função do gerenciador de recursos para sinalizar que a energia foi consumida. O módulo do aplicativo é aquele que o usuário normalmente altera, geralmente criando um novo módulo para implementar um novo algoritmo. Os módulos MAC e Roteamento de comunicações, bem como o módulo Gerenciador de Mobilidade, também são bons candidatos para mudanças pelo usuário, geralmente criando um novo módulo para implementar um novo protocolo ou padrão de mobilidade. O Castalia oferece suporte para construção de seus próprios protocolos ou aplicativos definindo classes abstratas apropriadas. Todos os módulos existentes são altamente configuráveis por parâmetros.

A estrutura interna do módulo composto do nó é implementada no Castalia com o uso da linguagem NED do OMNeT++. Com essa linguagem, podemos definir facilmente módulos, ou seja, definir um nome de módulo, parâmetros de módulo e interface de módulo e uma possível estrutura de submódulo (se este for um módulo composto). Os arquivos com o sufixo ".ned"contém o código de idioma NED. A estrutura do Castalia também é refletida na hierarquia de diretórios no código-fonte. Cada módulo corresponde a um diretório que sempre contém um arquivo .ned que define o módulo. Se o módulo for composto, haverá subdiretórios para definir os submódulos. Se for um módulo simples, então existe o código C++ (arquivos .cc, .h) para definir seu comportamento. Essa hierarquia completa

Figura 14 – O módulo composto do nó.



Fonte: [9].

de arquivos .ned define a estrutura geral do simulador Castalia. Normalmente, o usuário não irá alterar esses arquivos. No entanto, esses arquivos são carregados e processados dinamicamente (usando um recurso do OMNeT++) para que qualquer alteração não exija a recompilação do Castalia (a menos que novos módulos simples com novas funcionalidades apareçam).

Na pasta: /Castalia/bin encontram-se os executáveis para realizar as simulações.

- **Castalia**: executa as simulações.
- **CastaliaResults**: interpreta os resultados.
- **CastaliaPlot**: cria gráficos dos resultados proveniente do CastaliaResults, utilizando o GnuPlot.

2.5 MANEJO DO GADO LEITEIRO

Em uma propriedade leiteira, as instalações são de grande importância, porque facilitam o manejo dos animais, influindo diretamente na sua produtividade e saúde.

2.5.1 **Instalações para Gado de Leite**

Para que as instalações zootécnicas de um modo geral, não só aquelas destinadas à produção de leite, sejam consideradas satisfatórias, na fase de planejamento devem ser considerados vários pontos impontantes como: localização, orientação e distribuição dos

prédios que compõem a exploração, entre outros [41]. Neste trabalho utilizou-se um local aberto para manejo de gado que envolve os seguintes pontos:

- **Localização**

A construção das instalações para exploração do gado leiteiro depende de vários fatores, tais como: tipo de manejo do gado, tamanho da exploração, além das características do clima, de solo e da topografia. Para construção desse tipo de instalação, o local ideal deve ser bem drenado, exposto aos raios solares, o que facilita a secagem e diminui a proliferação de organismos patogênicos. O estabulo não deve ser atravessado por fortes correntes de ar frio que favorecem surtos de doenças do sistema respiratório dos animais. No entanto, deve permitir um conforto térmico, evitando o predomínio das altas temperaturas [42].

- **Pasto**

O número de animais por hectare varia muito de acordo com a raça animal, idade dos animais, espécie forrageira, tipo de solo (levando-se em consideração sua fertilidade), umidade do solo (se tem irrigação ou não), idade da forrageira, etc. Tudo isso está associado à necessidade de matéria seca que cada animal necessita.

Em bovino de leite, temos várias categorias animais. A primeira categoria é a fase de bezerros (animais até um ano de idade), em que uma cabeça equivale a 0,33 UA, seguida por garrotes (animais de um a dois anos de idade), em que uma cabeça equivale a 0,50 UA, a terceira (animais acima de 2 anos de idade para matrizes e novilhas e entre 2 a 3 anos para novilhos), e a categoria de animais adultos, em que uma cabeça equivale a 1,0 UA. servindo para matrizes bovina, novilhos e novilhas [43].

- **Água**

Para assegurar a produção de leite, manter a temperatura corporal e demais funções vitais dos animais de forma adequada, é necessário oferecer a eles água fresca e limpa, com proporções quantitativas em torno de 20 a 40 litros por dia por animal, devendo os bebedouros serem construídos de maneira a oferecerem tal demanda. A altura dos bebedouros deve ser próxima a 0,75 m e a largura deve ser de 0,70 m caso o acesso seja por apenas um lado ou de 1,0 m, caso o acesso seja pelos dois lados. De certa forma, o dimensionamento dos bebedouros obedece à relação de disponibilidade de água pelo número de animais com acesso [44].

A vazão de água é ainda mais importante do que as dimensões dos bebedouros, no sentido de evitar a restrição de água para os bovinos. A vazão não deve ser inferior a

10-20 litros por minuto, evitando um tempo longo de enchimento e renovação da água. Os bovinos possuem maior afeição para bebedouros mais rasos, com água limpa, que permitem a visualização do fundo. Outro fato é que estes animais preferem consumir a água com temperatura entre 25 e 30 °C, tendendo a diminuir o consumo quando a temperatura desta está abaixo de 15 °C.

- **Cocho**

Um dos aspectos fundamentais do conforto para as vacas é o espaço no cocho de alimentação. O espaço disponível afeta o nível de competição no cocho, que por sua vez afeta significativamente os hábitos alimentares e o consumo de alimentos. A discussão sobre a necessidade de espaço em cocho não é nova, e há muitos trabalhos realizados sobre o assunto. No mercado existe um consenso geral de que algo em torno de 0,6-0,7 metros de espaço linear no cocho é suficiente para cada vaca comer sossegada, sem estresse, e dessa forma poder expressar todo o seu potencial produtivo, pelo menos no que depender dessa variável. Mas como esse tema sempre gera algumas discussões, o interesse da comunidade científica pelo assunto ainda existe, como mostra o artigo publicado no Journal of Dairy Science [45].

Nesse artigo, os autores avaliaram o comportamento ingestivo de 4 grupos de vacas mantidas em galpão Free-Stall, sendo que cada grupo dispunha de espaços diferentes no cocho de alimentação, 0,21, 0,41, 0,61 ou 0,81 metros lineares por vaca. Os grupos também foram submetidos a dois tipos de contenção no cocho, um com canzis individuais, e outro com uma contenção simples que apenas impedia as vacas de passar para o corredor de alimentação, sem qualquer contenção lateral. As vacas foram monitoradas 24 horas por dia, através de equipamento de vídeo colocado 6m acima da linha do cocho, de forma que todas as vacas de cada grupo podiam ser observadas. Além do espaço de cocho, avaliou-se também o efeito do tipo de contenção.

Os resultados mostraram que houve uma redução linear no número de períodos de alimentação (visitas ao cocho) à medida que se reduzia o espaço disponível por vaca. O efeito foi mais pronunciado nos momentos de pico de alimentação, ou seja, cerca de 1 hora após o descarregamento da ração no cocho. Nesses momentos, a porcentagem de vacas se alimentando diminuía com a redução do espaço disponível. A presença do canzil também resultou em redução nesse parâmetro. Da mesma forma, observou-se aumento no tempo em que as vacas permaneciam inativas e em pé na área de alimentação, e na incidência de comportamentos agressivos, à medida que se reduzia o espaço disponível no cocho. A presença do canzil também contribuiu para o aumento nesses dois parâmetros. No caso da incidência de comportamentos agressivos, que eram considerados quando alguma vaca era "expulsa"por outra da linha de cocho, a presença do canzil foi útil para reduzir essas ocorrências.

O trabalho também mostrou um aspecto muito interessante que é o fato de as vacas submetidas aos canzis passaram menos tempo se alimentando, o que fatalmente compromete o consumo de matéria seca. Os autores argumentam que na contenção com canzil, a vaca pode ficar mais desconfortável, e que também é possível que os animais desenvolvam algum tipo de aversão a esse tipo de contenção, uma vez que é comumente utilizada nos troncos para inseminação artificial e manipulação por veterinários, o que pode fazer com que as vacas fiquem relutantes em passar a cabeça pelo canzil para acessar a comida. E o trabalho mostrou claramente que, principalmente no momento de pico de alimentação, menos vacas estavam presentes nos cochos com canzil. De maneira geral, a recomendação de 0,6 metros por vaca parece ser adequada, mas os resultados desse trabalho mostram que se houver mais espaço disponível, as vacas podem ser beneficiadas, tendo acesso mais livre ao cocho, favorecendo o consumo de matéria seca, o que todo produtor de leite quer que seja maximizado. Os resultados estão na Tabela 1.

Tabela 1 – Porcentagem de vacas se alimentando e inativas em pé durante o pico da atividade de alimentação.

	Metros Lineares por Vaca				**Canzis por Vaca**			
Parâmetro	0,81	0,61	0,41	0,21	1,33	1,00	0,67	0,33
Vacas se Alimentando (%)	61,1	63,2	48,0	33,8	61,3	54,6	43,9	28,5
Vacas Inativas em Pé (%)	5,1	6,2	12,5	23,3	3,8	8,0	14,4	25,6

Fonte: [45]

- **Sombra**

A questão do conforto de vacas leiteiras, e seus efeitos sobre o desempenho produtivo e reprodutivo dos animais tem sido objeto de numerosos estudos nos últimos anos. O conforto de vacas em pastejo, tem a ver com a sombra, principalmente por causa dos efeitos do calor sobre os animais. O calor excessivo prejudica a produção de leite e o desempenho reprodutivo das vacas, e esses efeitos são em parte explicados pela redução na ingestão de alimentos dos animais sob estresse calórico, mas o calor também afeta o status endócrino das vacas, reduz o tempo de ruminação e a absorção de nutrientes, além de elevar os requerimentos para mantença dos animais, o que resulta em menor disponibilidade de nutrientes e energia para as funções produtivas e reprodutivas.

Para não ter nenhum prejuízo no seu desempenho, uma vaca leiteira precisa de, no mínimo, 8-10 horas diárias de descanso em local fresco, seco e confortável. É muito comum ver vacas sem 1 m^2 de sombra disponível nas áreas de pastagem, ou com áreas de descanso mal drenadas, com grande acúmulo de barro. Se na área de descanso uma vaca permanece em pé, ou fica agitada balançando a cabeça, isso é um sinal claro de estresse [46].

Para não sofrerem stress por falta de conforto as vacas precisam de áreas de descanso com espaço suficiente, piso seco e macio, e sombra, de preferência natural, para se abrigarem do calor. Pelo menos 5 m^2 por vaca são necessários para garantir o conforto em áreas de descanso [46].

2.5.2 **Velocidades e Pesos do Gado Leiteiro**

No trabalho [47] foi feito um estudo sobre o desempenho da caminhada do gado leiteiro. Utilizaram 3 raças diferentes (Holstein, Montbéliard e Tarentaise) onde tiveram como resultados: para distâncias percorridas de 3,2 e 5,6 km a velocidade média de caminhada da raça Tarentaise foi 1,43 e 1,71 m/s, para a raça Montbéliard foi 1,10 e 1,49 m/s e a raça Holandesa foi 1,23 e 1,50 m/s.

Segundo D'Hour et al. [47] o peso vivo da raça Holstein é 577 ± 30 Kg. Neste trabalho serão utilizados os dados da raça Holstein (Gado Holandês). Originária da Europa, surgiu primitivamente entre a Frísia (norte dos Países Baixos) e o Holstein (Alemanha), há cerca de vinte séculos. Tal raça desde muitos séculos tem sido criada nos Países Baixos, no Norte da Alemanha e na Dinamarca, tanto pela carne quanto pelo leite, e destas regiões foi levada para o restante do mundo. É uma das raças de maior aptidão leiteira conhecida, sendo comum nos Açores e no Brasil, em especial no Centro-Sul do país.

2.5.3 **O Consumo de Energia do Gado Leiteiro**

Em sistemas de pastejo se tem um consumo de energia do gado leiteiro, uma vez que os animais percorrem maiores distâncias para beber água, ou até mesmo para consumirem sal.

Os primeiros trabalhos [48] abordando o tema, estimaram que o aumento na energia de manutenção foi na grandeza de 4 a 24% quando animais percorreram, em esteiras, as respectivas distâncias de 1 e 6 km. De acordo com o Conselho de Pesquisa Agrícola, o custo de locomoção é 0,00048 e 0,00067 Mcal/km/kg/peso corporal, para deslocamento horizontal e vertical, respectivamente. Em outras palavras, uma vaca de 550kg, por exemplo, caminhando 1 km em terreno de 0% de declividade tem um custo energético de 0,263 Mcal de energia metabolizável (550kg x 1km x 0,00048 + 550 x 0 km x 0,00067) [49]. Considerando que esta energia seja deslocada da produção de leite, teríamos um decréscimo de 212 g de leite para cada km percorrido (adotando o padrão NRC, sendo 1 kg de leite com 4% de gordura equivale a 1,24 Mcal de energia metabolizável).

A Figura 15 simula diferentes custos energéticos através da associação entre diferentes declividades, deslocamentos horizontal e vertical e pesos dos animais. As comparações foram feitas para distâncias percorridas horizontalmente mais curtas (1 km) e mais longas (6 km), simulando terreno plano e com declividade de 10%. Considerando que 1 kg de milho = 1,73 Mcal de Energia Líquida de Lactação e que 1 Mcal de ELL = EM * 0.644.

Figura 15 – Simulação do gasto energético em função do deslocamento horizontal e vertical.

Peso Vivo	Deslocamento	Deslocamento	Declividade	Custo Energético	Custo Energético	kg Milho
(kg)	Horizontal (km)	Vertical (km)	(%)	Mcal EM/vaca/dia	Mcal ELL/vaca/dia	
500	1	0	0	0,240	0,155	0,089
500	1	0,1	10	0,274	0,176	0,102
500	6	0	0	1,440	0,927	0,536
500	6	0,6	10	1,641	1,057	0,611
550	1	0	0	0,264	0,170	0,098
550	1	0,1	10	0,301	0,194	0,112
550	6	0	0	1,584	1,020	0,590
550	6	0,6	10	1,805	1,162	0,672
600	1	0	0	0,288	0,185	0,107
600	1	0,1	10	0,328	0,211	0,122
600	6	0	0	1,728	1,113	0,643
600	6	0,6	10	1,969	1,268	0,733

Fonte: [49]

Existem diversos dados a respeito do incremento no consumo de energia em função do caminhar dos animais. O NRC atribui um aumento de 3% para cada km percorrido, sendo considerado elevado por alguns pesquisadores. No trabalho do Penati e Corsi [49] se utilizou a técnica da taxa de diluição de dióxido de carbono (CO_2) em animais que caminharam entre 6 a 9 km por dia na velocidade de 3 km/hora e encontrou aumento no gasto energético de 4 e 6%. De acordo como os dados do trabalho do Di Marco e Aello [50], em terrenos de 0% de declividade, a energia consumida com o caminhar foi de 9 Kcal/km/kg/peso corporal e para a declividade de 6%, os valores encontrados foram de 16,4 Kcal/km/kg/peso corporal, aproximadamente.

3 TRABALHOS RELACIONADOS

A literatura reporta diversos estudos relacionados contendo os principais assuntos abordados por este trabalho, aqueles que trouxeram maiores contribuições para o desenvolvimento desta dissertação foram: os modelos de mobilidade para redes móveis *ad hoc* na seção 3.1, o monitoramento de animais com RSSFs na seção 3.2. E por último, na seção 3.3 estão os protocolos de roteamento.

3.1 MODELOS DE MOBILIDADE PARA REDES MÓVEIS *AD HOC*

Os modelos de mobilidade visam representar o comportamento de movimentação dos dispositivos móveis em uma rede ad hoc. Esses modelos são usados na avaliação do desempenho de aplicações e sistemas de comunicação, permitindo analisar o impacto causado pela mobilidade no funcionamento dos mesmos.

Existem diversos estudos que fazem um levantamento de uma série de modelos de mobilidade que representam nós móveis cujos movimentos são independentes um do outro (modelos de mobilidade individuais) e modelos de mobilidade de grupo que representam nós móveis cujos movimentos são dependentes um do outro (modelos de mobilidade em grupo) foram discutidos nos trabalhos do Camp et al. [7] e do Radha e Shanmugavel [51] com a finalidade de apresentar os modelos de mobilidade usados nas simulações de redes *ad hoc*.

Os resultados de simulação dão a importância da escolha de um modelo de mobilidade na simulação de um protocolo de rede *ad hoc*. Concluindo, a importância de escolher um modelo de mobilidade na simulação de um protocolo de rede, já que os resultados de desempenho de um protocolo de rede mudam drasticamente como resultado da mudança do modelo de mobilidade simulado.

Campos e Moraes [6] propuseram um modelo de mobilidade individual para redes móveis *ad hoc*, tendo uma maior liberdade de movimentação dos dispositivos. Através do modelo proposto se permitiu movimentos na mesma direção, em direções adjacentes, acelerações e intervalos de pausa no movimento, além de evitar mudanças bruscas de direção e paradas abruptas. Os modelos foram apresentados analiticamente através de cadeias de Markov e as comparações feitas através de simulação, entre os modelos propostos, o modelo Waypoint e o modelo Markoviano de Percurso Aleatório. Obtendo como resultados uma maior aproximação do movimento real dos usuários em um ambiente urbano e em rodovias.

3.2 REDES DE SENSORES SEM FIO PARA MONITORAMENTO DE ANIMAIS

No monitoramento de gado em espaço aberto, uma questão relevante é rastrear a localização dos animais no campo em um determinado momento. Kwong et al. [52] apresentaram uma plataforma de monitoramento de animais usando redes de sensores sem fio para lidar com os desafios da gestão da pecuária. Uns dos principais desafios foi a mobilidade pois os animais têm mudanças frequentes de localização. Os experimentos foram feitos com os movimentos de 13 vacas livres registrando os dados através de um GPS durante dois dias. Concluindo que os rebanhos não se distribuem uniformemente e nem sempre se movem como um único coletivo. Em vez disso, o rebanho pode se dividir em subgrupos independentes, cada um com sua própria dinâmica. Esse comportamento leva à necessidade da topologia de rede mudar dinamicamente. Além disso, é fundamental o conhecimento da mobilidade dos animais para possibilitar um funcionamento mais eficiente da plataforma de monitoramento.

A Universidade Federal de Mato Grosso do Sul (UFMS), especificamente a FACOM, tem uma parceria com a Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA) possuindo uma linha de pesquisa na área de pecuária de precisão. A primeira defesa de mestrado [3] construiu um sistema computacional capaz de identificar padrões comportamentais e informações relevantes através das trajetórias dos animais, produzidas durante o pastejo. Para isso, utilizou a metodologia de trajetória semântica com a finalidade de auxiliar os pecuaristas na tomada de decisões. No seguinte ano, o Leandro [53] criou um nó sensor com capacidade para coletar dados por meio de uma rede de sensores sem fio, monitorar os bovinos e inferir o comportamento animal por meio de um sistema. Cada nó possui um sensor GPS preso a um colar e colocado no bovino que circula em uma pastagem totalmente georreferenciada. Os dados gerados pelo GPS são armazenados em um cartão de memória. Neste mesmo trabalho foi criado um sistema de apoio na observação do comportamento dos animais em campo e uma abordagem para a classificação automática das atividades realizadas pelos bovinos dividida em andando, comendo/buscando, em pé e deitado.

Continuando na mesma linha de pesquisa, o Lomba [54] criou uma plataforma de identificação automática do comportamento bovino, utilizando os dados de movimentação e posicionamento do animal e algoritmos de classificação supervisionada. Por último, para seguir dando continuidade aos trabalhos anteriores o Nacer [55] desenvolveu um sistema integrado de hardware/software para aquisição e transmissão de posições de GPS para monitoramento de bovinos. Além disso, apresentou alguns protocolos da literatura e simulações de RSSFs, bem como as tecnologias utilizadas em RSSFs e como podem ser configuradas.

No ano passado foi criado um novo ramo de pesquisa no Laboratório de Aplicações e Inovação em Computação (LApIC) do Programa de Pós-graduação de Ciência da

Computação da Universidade Federal de Juiz de Fora (UFJF), na área de monitoramento e rastreamento de animais. O primeiro trabalho [5] implementou no simulador Castalia um modelo de mobilidade chamado *Pasture Mobility Manager* de animais em pasto para redes de sensores sem fio, utilizando um modelo Markoviano de percurso aleatório e estados finitos de cadeia de Markov. Seguindo essa ideia mais tentando fazer o modelo o mais parecido à realidade dos animais em pasto, se fez o presente trabalho.

3.3 PROTOCOLOS DE ROTEAMENTO

Os tipos de protocolos de roteamento reportados na literatura, estão presentes, em seu tempo, com avanços significativos na tarefa proposta, embora tenham sido superadas em eficacia por abordagens mais recentes.

Nos últimos anos, os desafios de roteamento das RSSFs atraíram grande interesse e muitos protocolos de roteamento para essas redes foram propostos: roteamento baseado em cluster, roteamento de múltiplos caminhos e roteamento ciente de energia, etc. Que têm sido amplamente utilizados para várias finalidades de rede, como para melhorar a confiabilidade da entrega de dados, fornecendo roteamento tolerante a falhas, controle de congestionamento [56]. Os nós sensores têm energia de bateria limitada e podem estar em qualquer lugar, incluindo áreas de acesso difícil, não sendo possível recarregar ou substituir suas baterias. Portanto, o consumo de energia de cada nó sensor deve ser minimizado para que a vida útil (tempo de funcionamento) total da rede seja maior. Os protocolos de roteamento são os encarregados de selecionar a melhor rota para permitir a comunicação entre os nós sensores da rede, tentar reduzir o consumo de energia e ter uma boa eficiência na entrega dos pacotes de dados.

Vidhyapriya e Vanathi [22] propuseram uma técnica de roteamento multipercurso adaptativo com eficiência energética que pretende utilizar a energia residual e a intensidade do sinal recebido para descobrir múltiplos caminhos para o nó *Sink* por transmissão de dados confiável e baixo consumo de energia. Pandya e Mehta [23] avaliaram o desempenho do protocolo de roteamento Multipath para redes de sensores sem fio implementado no simulador Castalia. Foi feita a comparação dos pacotes recebidos pelo nó *Sink* com o protocolo de roteamento Bypass implementado no Castalia. Huang et al. [57] apresentaram o algoritmo de roteamento Multipath para redes de sensores sem fio que utiliza o nível de anel para separar nós sensores em várias seções a fim de melhorar a confiabilidade da transmissão de dados. Foram feitas comparações do número de pacotes recebidos pelo nó *Sink* e o consumo de energia com dois protocolos de roteamento LEACH e Direto. Barot e Rajani [58] propuseram um novo protocolo de roteamento baseado no protocolo de roteamento Multipath e o protocolo de roteamento LEACH para fornecer confiabilidade e minimizar o congestionamento em uma rede, realizando a comparação dos pacotes recebidos pelo nó *Sink* do novo protocolo com o protocolo de roteamento Multipath.

3.4 DISCUSSÕES SOBRE O CAPÍTULO

Este capítulo apresentou trabalhos reportados na literatura e que estão relacionados com: os modelos de mobilidade, as redes de sensores sem fio para monitoramento de animais e os protocolos de roteamento.

Existem vários modelos de mobilidade que representam nós móveis cujos movimentos são independentes um dos outros e modelos de mobilidade que representam nós móveis cujos movimentos dependem um do outro. Dessa forma, os modelos existentes influenciaram para criar um novo modelo de mobilidade híbrido utilizando vários deles para atingir uma maior aproximação do movimento real dos animais em pasto. Outro fator que influenciou neste trabalho foi que várias pesquisas mostraram que uma seleção de modelo de mobilidade pode afetar o resultado da simulação de desempenho de roteamento em redes móveis. Assim, um protocolo de roteamento pode ser eficaz apenas em um modelo ou cenário de mobilidade específico, mas apresenta desempenho inferior em outro. Por causa disso, sabendo que existe um impacto dos modelos de mobilidade no desempenho do protocolo de roteamento de rede móvel, se avaliou o desempenho de vários protocolos de roteamento para saber qual é o que melhor se adapta ao modelo de mobilidade implementado.

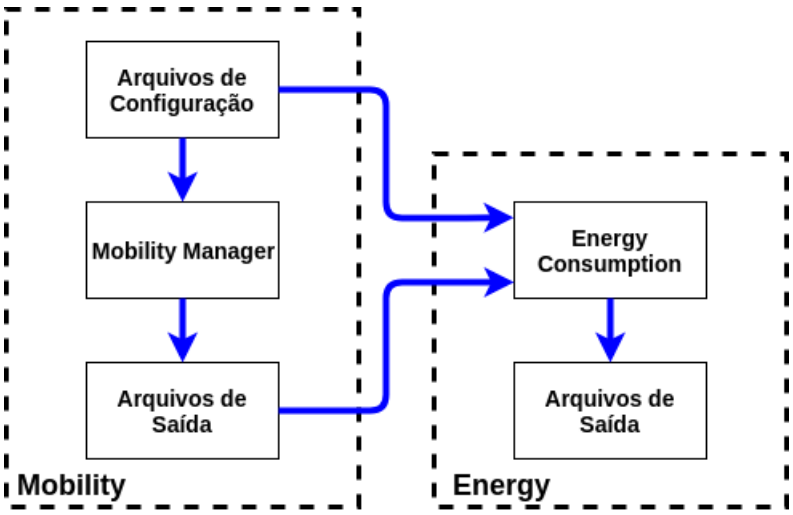
Uma lacuna de pesquisa perceptível nos trabalhos relacionados aos modelos de mobilidade e monitoramento de animais através de redes de sensores sem fio é a ausência de um modelo de mobilidade que permita simular o comportamento dos animais em pasto e também para calcular o consumo energético desses animais. Além disso, os trabalhos relacionados com os protocolos de roteamento influenciaram na avaliação da métrica do consumo de energia no modelo proposto.

4 ARQUITETURA

Este capítulo apresenta a arquitetura para calcular o consumo energético de animais em pastagem. Ela é formada por um módulo de mobilidade, chamado *Mobility Manager*, para simular os comportamentos dos animais (seção 4.1) e um outro módulo, chamado *Energy Consumption*, para realizar o cálculo do consumo energético dos animais (seção 4.2).

A arquitetura, ilustrada na Figura 16, é genérica e pode ser utilizada para simular a movimentação de quaisquer espécies de animais em pastagem. Ela foi validada em prova de conceito, utilizando uma instância específica para gado leiteiro, criada com informações coletadas na literatura científica [48] [49] [45], que tornou possível definir estados comportamentais que possam ser assumidos pelos animais.

Figura 16 – Arquitetura Proposta.



Fonte: Elaborada pelo autor.

A arquitetura é composta pelos seguintes módulos:

- **Mobility Manager**: utiliza o simulador Castalia 3.2 [9] para fazer o processamento dos arquivos de configuração e gerar as trajetórias dos animais através do modelo de mobilidade.

    1. Arquivos de configuração: fornecem todos os parâmetros para fazer a configuração da rede de sensores sem fio e a configuração das informações ligadas com os animais (subseção 4.1.1).
    2. Arquivos de saída: armazena os arquivos gerados pelo simulador Castalia 3.2 referente às posições de cada animal em cada instante de tempo. As saídas do módulo *Mobility Manager* são entradas do módulo *Energy Consumption*.

- **Energy Consumption**: faz o processamento dos arquivos de configuração e dos arquivos de saída do módulo Mobility Manager para gerar os resultados do cálculo do consumo energético dos animais. O algoritmo foi desenvolvido utilizando a linguagem de programação Python, no ambiente de desenvolvimento integrado JetBrains Pycharm [10].

    1. Arquivos de saída: submódulo encarregado de armazenar os arquivos gerados após o processamento. Esses arquivos fornecem: os cálculos do consumo energético, distâncias e figuras das trajetórias por cada nó sensor na pastagem.

4.1 MOBILITY MANAGER

O módulo de mobilidade especifica como os nós se movem pelo terreno (área de pasto). Ele mantém o estado de localização que outros módulos internos do simulador Castalia podem acessar a qualquer momento usando uma chamada de função e também notifica periodicamente o canal sem fio da posição de um nó. A notificação do canal sem fio é feita por motivos de eficiência. Como o canal sem fio precisa da localização de todos os nós com muita frequência (toda vez que temos o início de uma transmissão de pacote ou detecção de portadora), seria prejudicial se ele tivesse que solicitar explicitamente os locais de todos os nós (apenas para detectar que na maioria dos casos nada mudou), sendo mais proveitoso para o módulo de mobilidade notificar o canal somente se algo mudou (por exemplo, ele poderia notificá-lo, quando o nó acabou de alterar a localização) [9].

O módulo de mobilidade para animais em pastagem, *Mobility Manager*, é formado pelos seguintes modelos [5]:

- Modelo Markoviano de Percurso Aleatório: utilizado para realizar as transições de áreas através das matrizes markovianas.
- Modelo de Mobilidade Gauss-Markov: utilizado para realizar as transições de comportamentos dentro de cada área através das matrizes markovianas.
- Modelo de Mobilidade em Grupo com Ponto de Referência: implementa a estratégia de agrupamento dos animais, já que o módulo permite fazer agrupamento de animais até n, conforme especificado no arquivo de configuração *ConfigMobility.xml (leaderGroup)*, onde n é o número máximo de animais.

4.1.1 Arquivos de Configuração

Este submódulo contém dois arquivos de configuração, o *omnetpp.ini* e o *ConfigMobility.xml*, que permitem atribuir grande flexibilidade a solução proposta, pois todos os valores considerados nas simulações são passíveis de alteração.

O arquivo de configuração *omnetpp.ini* define os cenários de simulação, contendo as configurações da simulação que o usuário deseja executar. Esse arquivo tem como parâmetros: Biblioteca, Tempo, Nós, Área, Canal Sem fio, Gerenciador de Aplicação, Comunicação, Roteamento e Mobilidade do simulador e deve ter a estrutura apresentada na Figura 17 e estar baseado nos parâmetros da Tabela 2.

Figura 17 – Estrutura do Arquivo *omnetpp.ini*.

```
[General]
# BIBLIOTECA
include ../Parameters/Castalia.ini

# TEMPO DA SIMULAÇÃO
sim-time-limit = 36000s

# TAMANHO DA ÁREA
SN.field_x = 225     # EIXO X (metros)
SN.field_y = 225     # EIXO Y (metros)

# QUANTIDADE DE NÓS
SN.numNodes = 10

# CONFIGURAÇÃO DO CANAL SEM FIO
SN.wirelessChannel.onlyStaticNodes = false
SN.wirelessChannel.sigma = 0
SN.wirelessChannel.bidirectionalSigma = 0

# CONFIGURAÇÃO DO GERENCIADOR DE COMUNICAÇÃO
SN.node[*].Communication.Radio.RadioParametersFile = "../Parameters/Radio/CC2420.txt"
SN.node[*].Communication.Radio.TxOutputPower = "-5dBm"
SN.node[*].Communication.Routing.maxNetFrameSize = 2500
SN.node[*].Communication.MAC.maxMACFrameSize = 2500
SN.node[*].Communication.Radio.maxPhyFrameSize = 2500

# CONFIGURAÇÃO DO GERENCIADOR DE APLICAÇÃO
SN.node[*].ApplicationName = "ThroughputTest"                    # NOME DO GERENCIADOR DE APLICAÇÃO
SN.node[*].Application.packet_rate = 5
SN.node[*].Application.constantDataPayload = 2000

# CONFIGURAÇÃO DO GERENCIADOR DE MOBILIDADE
SN.node[*].MobilityManager.collectTraceInfo = true               # ARQUIVO TRACE. TRUE = CRIA ARQUIVO, FALSE = NÃO CRIA ARQUIVO
SN.node[0..9].MobilityManager.areaSelStrategy = 1                # ESTRATEGIA DE SELEÇÃO. 0 = RANDOM, 1 = GAUSS-MARKOV
SN.node[0..9].MobilityManagerName = "CattleMobilityManager"      # NOME DO GERENCIADOR DE MOBILIDADE
SN.node[0..9].MobilityManager.configFile = xmldoc("ConfigCattleMobility.xml")  # NOME DO ARQUIVO XML DOS PARAMETROS DE CONFIGURAÇÃO
SN.node[0..9].MobilityManager.updateInterval = 100
SN.node[0..4].MobilityManager.groupLeaderIndex = 0               # NUMERO DE NÓ LIDER DO GRUPO [X..Y]
SN.node[5..9].MobilityManager.groupLeaderIndex = 5               # NUMERO DE NÓ LIDER DO GRUPO [X..Y]
```

Fonte: Elaborada pelo autor.

Tabela 2 – Parâmetros Configuráveis no Arquivo *omnetpp.ini*.

Módulos	**Parâmetros**	**Padrão**	**Obrigatório**
Geral	sim-time-limit	36000	SIM
	SN.field_x	225	SIM
	SN.field_y	225	SIM
	SN.numNodes	10	SIM
Canal Sem Fio	onlyStaticNodes	FALSE	SIM
Gerenciador de Comunicação	RadioParametersFile	"../Parameters/Radio/CC2420.txt"	SIM
	TxOutputPower	-5dBm"	SIM
Gerenciador de Aplicação	ApplicationName	ThroughputTest	SIM
Protocolo de Roteamento	RoutingProtocolName	"Bypass"	NÃO
Gerenciador de Mobilidade	MobilityManagerName	"MobilityManager"	SIM
	configFile	xmldoc("ConfigMobility.xml")	SIM
	areaSelStrategy	1	SIM
	updateInterval	100	SIM
	groupLeaderIndex	0 - 5	SIM
	collectTraceInfo	TRUE	NÃO

Fonte: Elaborada pelo autor.

Para o funcionamento do módulo *Mobility Manager*, o arquivo de configuração do módulo, *ConfigMobility.xml*, deve conter os parâmetros configuráveis relacionados propriamente aos animais. Esse arquivo tem padrão preestabelecido, que deve ser informado

no arquivo de configuração *omnetpp.ini* por meio do parâmetro *configFile*. Os parâmetros configuráveis são apresentados na Tabela 3.

Tabela 3 – Parâmetros Configuráveis no Arquivo *ConfigMobility.xml*.

Tags	**Parâmetros**	**Padrão**	**Obrigatório**
initialParams	areaTypeID, behaviorID	0, 0	NÃO
leaderGroup	percentValue	2	NÃO
matrixTemperature	temperatureMatrixID	0	NÃO
matrixRelief	reliefMatrixID	0	NÃO
nodesNumber	number	10	SIM
horizontalConstant	constant	0,00048	SIM
verticalConstant	constant	0,00067	SIM
divisionsX	axisX	4	NÃO
divisionsY	axisY	4	NÃO
initialTime	initial	7	NÃO
finalTime	final	17	NÃO
DairyCattleWeight	weightID, weight		SIM
areaType	areaTypeID, name, xMin, yMin, xMax, yMax, behaviorsMatrixID		SIM
behavior	behaviorID, name, minSpeed, maxSpeed		SIM
temperatureMatrix	temperatureMatrixID, type, row, cell		SIM
reliefMatrix	reliefMatrixID, type, row, cell		SIM
markovAreaMatrix	areaMatrixID, type, row, cell		SIM
markovBehaviorMatrix	behaviorMatrixID, type, row, cell		SIM

Fonte: Elaborada pelo autor.

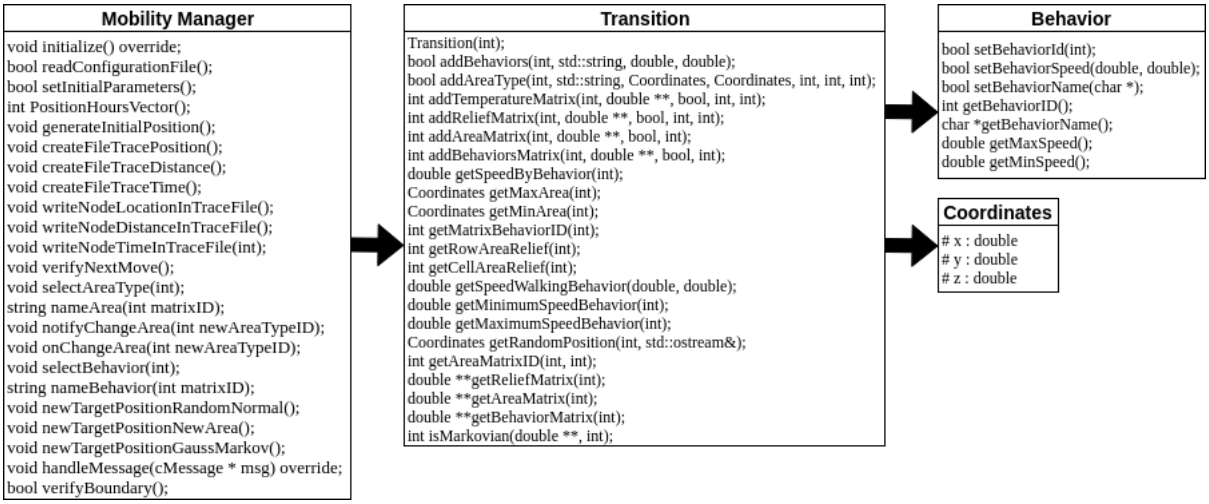
4.1.2 **Estrutura do *Mobility Manager***

O módulo é um híbrido de modelos de mobilidade para redes *ad hoc*. Ele é baseado nos modelos de mobilidade individual e em grupo. O modelo de mobilidade individual considera o Modelo Markoviano de Percurso Aleatório [6] e o Modelo de Mobilidade Gauss-Markov [7]. Já o modelo de mobilidade em grupo é baseado no Modelo de Mobilidade de Grupo com Ponto de Referência [8].

Embora tenham sido utilizadas as 4 classes do desenvolvimento da dissertação do Daniel Prata Borges [5], como apresentado na Figura 18, em cada método dessas classes foi feita a otimização dos algoritmos anteriormente desenvolvidos, com a finalidade de corrigir alguns erros e otimizar o código para um melhor desempenho. Também foram inseridos novos métodos para permitir as contribuições deste trabalho. Os métodos desenvolvidos em cada classe são apresentados no Apêndice A.

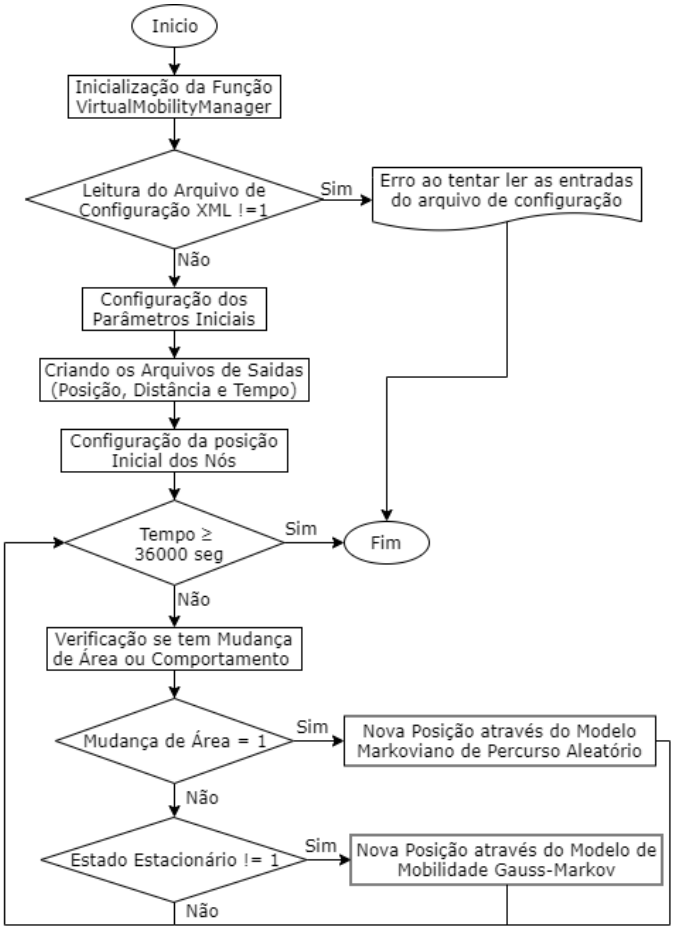
Uma visão geral da classe principal *Mobility Manager* é apresentada no diagrama de fluxo da Figura 19. Também, se apresenta o Algoritmo 1, responsável pela lógica principal desta classe.

Figura 18 – Classes e Métodos do *Mobility Manager*



Fonte: Adaptada de [5] pelo autor.

Figura 19 – Diagrama de fluxo da Classe *Mobility Manager*.



Fonte: Elaborada pelo autor.

Algoritmo 1: Módulo *Mobility Manager*

Entrada: Arquivo de Configuração .xml e omnetpp.ini
Saída: Arquivos .log: Posição e Tempo
início
Inicialização da Função VirtualMobilityManager;
if *Leitura do Arquivo de Configuração XML* **then**
Configuração dos parâmetros iniciais;
Criação dos arquivos .log de saídas;
Configuração da posição inicial dos nós;
Tempo ← 0;
repita
if *Nó = Líder* **then**
if *Verificação se tem mudança de área* **then**
Calcula a nova posição pelo Modelo Markoviano de Percurso Aleatório;
Notifica aos nós subordinados que houve mudança de área;
else
if *Estado Estacionário* ! = *1* **then**
if *Estrategia de Seleção = 0* **then**
Calcula a nova posição pelo Modelo Randômico;
else
Calcula a nova posição pelo Modelo Gauss-Markov
end
end
end
else
if *Verificação se tem mudança de comportamento* **then**
if *Estado Estacionário* ! = *1* **then**
if *Estrategia de Seleção = 0* **then**
Calcula a nova posição pelo Modelo Randômico;
else
Calcula a nova posição pelo Modelo Gauss-Markov
end
end
end
end
if *Rota de Rastreamento* **then**
Escreve nos arquivos .log: Posição, Distância e Tempo;
end
até *Tempo <= 36000*;
else
Erro ao tentar validar os parâmetros lidos no arquivo de configuração;
end
fim

4.1.3 **Funcionamento do *Mobility Manager***

O simulador Castalia começa carregando as informações contidas nos arquivos de configuração *omnetpp.ini* e o módulo mobilidade começa carregando as informações contidas no arquivo de configuração *ConfigMobility.xml*, que foram detalhados na subseção 4.1.1. Após, serem validadas essas informações constrói as áreas, os comportamentos, as matrizes de temperatura, as matrizes de relevos, as matrizes markovianas de transição entre as áreas e entre os comportamentos para cada área, etc. São criados arquivos .log onde ficarão armazenadas as informações das movimentações. Logo, é iniciado o processo de seleção da posição inicial de cada nó, essa posição será selecionada utilizando uma distribuição uniforme randômica dentro de uma área pré determinada como entrada, onde o eixo X varia no intervalo $0 \leq X \leq 2,5$ e o eixo $Y = 0$, simulando a entrada no pasto.

Uma vez que foi selecionada a posição inicial de cada nó, é informada ao canal sem fio do simulador, o gerenciador de mobilidade proposto irá verificar se o nó é um líder de grupo. No caso se o nó for líder de grupo, o gerenciador irá calcular através da matriz markoviana de transição de áreas se o grupo irá continuar ou mudar de área. A seleção da matriz markoviana a utilizar vai depender do tempo atual e da matriz de temperatura. Se vai ou não mudar de área, o nó notifica para todos os nós subordinados. No caso se o nó não for líder de grupo, o gerenciador irá calcular através da matriz de transição dos comportamentos da área em questão, se o nó irá continuar ou mudar de comportamento.

Se ocorrer mudança de área (determinado pelo nó líder de grupo), todos os nós do grupo assumem o comportamento ANDANDO e calcula a velocidade de deslocamento, utilizando uma função normal randômica que recebe como entrada a velocidade mínima e máxima do comportamento mais uma variação da velocidade dependendo da área de relevo atual e a futura. E irá se mover até uma posição, também calculada randomicamente, dentro da nova área que será frequentada pelo grupo.

Se ocorrer mudança de comportamento (determinado por cada nó individualmente), cada nó assume o comportamento selecionado pela matriz de transições dos comportamentos de cada área, calcula a velocidade igual como se fosse mudança de área. E irá se mover até uma posição, calculada pelo modelo de Gauss-Markov. O modelo de mobilidade pode utilizar duas estratégias de seleção de área ("1"= Modelo de Gauss-Markov e "0"= Modelo de mobilidade aleatória).

Quando o nó alcançar o ponto de destino ou mudar de um comportamento estacionário para um comportamento móvel, o gerenciador de mobilidade retorna para a fase de verificar se deve ocorrer uma mudança de área e segue o fluxo explicado acima novamente. Isso irá se repetir até que o tempo máximo de simulação seja alcançado.

Ao final da simulação, o módulo de mobilidade disponibiliza três arquivos trace para cada nó, onde <node_index> é o identificador único do nó. São eles:

- trace_position_<node_index>.log: contem as coordenadas X e Y separados por um ";" como mostra a Figura 20a.
- trace_distance_<node_index>.log: contem as distâncias percorridas como mostra a Figura 20b.
- trace_time_<node_index>.log: contem o tempo quando ocorreu uma movimentação e o tipo de movimentação ("1" = Mudança de Área, "0" = Mudança de Comportamento, "−1" = Não houve mudança) separados por um ";" como mostra a Figura 20c.

Figura 20 – Arquivos .log gerados após a simulação.

(a) Posição

trace_position_0.log

1	1.48211;0
2	107.899;2.3991
3	108.019;2.68381
4	108.119;3.02644
5	107.187;110.008
6	70.4873;198.359
7	70.4887;198.849
8	70.3075;199.341
9	70.1149;199.669
10	4.34277;111.238
11	105.392;108.935
12	102.416;52.1989
13	105.528;2.05744
14	69.7861;36.6446
15	2.15462;104.29
16	101.807;4.96866
17	2.04219;101.604
18	103.099;111.923
19	69.114;124.331
20	105.22;108.155
21	13.5934;168.97
22	110.157;109.035
23	3.30053;108.994
24	107.516;2.88071
25	170.453;95.0332
26	114.136;2.56815
27	48.4254;200.461
28	47.9813;201.12
29	107.738;1.43301
30	144.282;55.4746
31	143.815;55.9249
32	143.299;56.3349
33	106.009;4.1106
34	105.619;4.1931
35	102.562;112.053
36	179.712;140.581
37	179.278;140.535
38	2.53251;108.706

(b) Distância

trace_distance_0.log

1	106.478
2	0.31036
3	0.357524
4	107.026
5	95.7042
6	0.509083
7	0.526923
8	0.38161
9	110.248
10	101.105
11	56.8805
12	50.243
13	49.7662
14	95.6773
15	140.708
16	138.955
17	101.604
18	36.1795
19	39.6013
20	109.994
21	113.691
22	106.866
23	148.746
24	111.604
25	108.316
26	208.547
27	0.813794
28	208.467
29	65.2383
30	0.665233
31	0.697179
32	64.2018
33	0.399717
34	107.93
35	82.2876
36	0.438427
37	179.593
38	146.973

(c) Tempo

trace_time_0.log

1	0.1;1
2	146.3;0
3	158.2;0
4	174.6;1
5	314.5;1
6	474;0
7	475.4;0
8	495.7;0
9	510.8;1
10	662.7;1
11	919.9;1
12	990;1
13	1107.8;1
14	1206.5;1
15	1370;1
16	1688.5;1
17	1890.8;1
18	2036.7;1
19	2086.3;1
20	2151;1
21	2325.8;1
22	2484.7;1
23	2721;1
24	3133.9;1
25	3305.6;1
26	3466.8;1
27	3780.1;0
28	3782.7;1
29	4416.2;1
30	4502.5;0
31	4504.5;0
32	4506.2;1
33	4646.4;0
34	4673.9;1
35	4953.3;1
36	5064.6;0
37	5081.2;1
38	5325.2;1

Fonte: Elaborada pelo autor.

A simulação também disponibiliza mais dois arquivos. O primeiro arquivo, que recebe o nome no formato YYMMDD-HHMMSS.txt e armazena uma compilação final das informações coletadas durante a simulação (consumo de energia, pacotes perdidos, pacotes recebidos por cada nó, entre outros). Um exemplo desse arquivo é apresentado na Figura 21. O segundo arquivo, denominado Castalia-Trace.txt, disponibiliza todas as informações sobre a execução da simulação. Este arquivo é útil e foi utilizado durante

a fase de implementação devido a grande dificuldade de realizar *debug* na execução das simulações. Um exemplo desse arquivo é apresentado na Figura 22.

Figura 21 – Arquivo contendo o resultado da simulação.

190421-131539.txt

```
Castalia| what:General (1)
Castalia| when:2019-04-21 13:15
Castalia| label:General
Castalia|   module:SN.node[0].ResourceManager
Castalia|       simple output name:Consumed Energy
Castalia|           2448
Castalia|   module:SN.node[0].Communication.Radio
Castalia|       simple output name:RX pkt breakdown
Castalia|           13381 Failed with NO interference
Castalia|           321649 Failed with interference
Castalia|           256501 Failed, below sensitivity
Castalia|           33591 Received despite interference
Castalia|           13073 Received with NO interference
Castalia|   module:SN.node[0].Application
Castalia|         index:1 simple output name:Packets received per node
Castalia|           8461
Castalia|         index:2 simple output name:Packets received per node
Castalia|           12646
Castalia|         index:3 simple output name:Packets received per node
Castalia|           8330
Castalia|         index:4 simple output name:Packets received per node
Castalia|           8023
Castalia|         index:5 simple output name:Packets received per node
Castalia|           1338
Castalia|         index:6 simple output name:Packets received per node
Castalia|           2179
Castalia|         index:7 simple output name:Packets received per node
Castalia|           1668
Castalia|         index:8 simple output name:Packets received per node
Castalia|           1233
Castalia|         index:9 simple output name:Packets received per node
Castalia|           2786
Castalia|       histogram name:Application level latency, in ms
Castalia|       histogram_min:0 histogram_max:200
Castalia|       histogram_values 0 0 0 46664 0 0 0 0 0 0
Castalia|   module:SN.node[1].ResourceManager
Castalia|       simple output name:Consumed Energy
Castalia|           2263.35
```

Fonte: Elaborada pelo autor.

4.2 ENERGY CONSUMPTION

O módulo *Energy Consumption* faz o processamento dos arquivos de configuração e dos arquivos de saída do módulo Mobility Manager para gerar os resultados do cálculo do consumo energético dos animais. Utiliza o arquivo de configuração e os arquivos de saída do módulo *Mobility Manager*.

4.2.1 **Estrutura do *Energy Consumption***

O algoritmo 2 faz o cálculo do consumo energético de gado leiteiro mediante as equações apresentadas no trabalho do Penati e Corsi [49]: a Equação 4.1 para o consumo de energia metabólica (EM) e a Equação 4.2 para o consumo de energia de lactação líquida (ELL). De acordo com o ARC [59], o custo de locomoção é de 0,00048 e 0,00067 Mcal/km/kg/peso corporal, para deslocamento horizontal e vertical, respectivamente.

$$C_{usto}E_{nergetico}EM = P_{eso} * [(0,00048 * D_{eslocamento}H_{orizontal}) + (0,00067 * D_{eslocamento}V_{ertical})] \quad (4.1)$$

Figura 22 – Arquivo Castalia-Trace.txt.

Castalia-Trace.txt

```
0    SN.node[0].MobilityManager    INICIALIZAÇÃO DO NÓ : 0
0    SN.node[0].MobilityManager    Posiçao Inicial (X:Y:Z) é 0:0:0
0    SN.node[0].MobilityManager    INICIANDO A LEITURA DO ARQUIVO DE CONFIGURAÇÃO PARA O NÓ : 0
0    SN.node[0].MobilityManager    ID do Comportamento Inicial : 0, Nome do Comportamento Inicial : ANDANDO
0    SN.node[0].MobilityManager    ID da Área Inicial : 0, Nome da Área Inicial : PASTO
0    SN.node[0].MobilityManager    Porcentagem de Grupos Formados : 2%
0    SN.node[0].MobilityManager    ID da Matriz de Temperatura Selecionada  : 0
0    SN.node[0].MobilityManager    ID da Matriz de Relevo Selecionada : 0
0    SN.node[0].MobilityManager    Constante Horizontal para o Calculo do Relevo : 0.00048
0    SN.node[0].MobilityManager    Constante Vertical para o Calculo do Relevo : 0.00067
0    SN.node[0].MobilityManager    Divisões no Eixo X : 4
0    SN.node[0].MobilityManager    Divisões no Eixo Y : 4
0    SN.node[0].MobilityManager    Hora de Inicio : 7h
0    SN.node[0].MobilityManager    Hora Final : 17h
0    SN.node[0].MobilityManager    Número Total de Comportamentos : 4
0    SN.node[0].MobilityManager    ID do Comportamento : 0
0    SN.node[0].MobilityManager    Nome do Comportamento : ANDANDO
0    SN.node[0].MobilityManager    Velocidade Mínima : 0.277 m/s
0    SN.node[0].MobilityManager    Velocidade Máxima : 0.833 m/s
0    SN.node[0].MobilityManager    Comportamento --> ANDANDO Gravado com Sucesso
0    SN.node[0].MobilityManager    ID do Comportamento : 1
0    SN.node[0].MobilityManager    Nome do Comportamento : CORRENDO
0    SN.node[0].MobilityManager    Velocidade Mínima : 0.833 m/s
0    SN.node[0].MobilityManager    Velocidade Máxima : 1.388 m/s
0    SN.node[0].MobilityManager    Comportamento --> CORRENDO Gravado com Sucesso
0    SN.node[0].MobilityManager    ID do Comportamento : 2
0    SN.node[0].MobilityManager    Nome do Comportamento : DEITADO
0    SN.node[0].MobilityManager    Velocidade Mínima : 0 m/s
0    SN.node[0].MobilityManager    Velocidade Máxima : 0 m/s
0    SN.node[0].MobilityManager    Comportamento --> DEITADO Gravado com Sucesso
0    SN.node[0].MobilityManager    ID do Comportamento : 3
0    SN.node[0].MobilityManager    Nome do Comportamento : ALIMENTANDO
0    SN.node[0].MobilityManager    Velocidade Mínima : 0.013 m/s
0    SN.node[0].MobilityManager    Velocidade Máxima : 0.027 m/s
0    SN.node[0].MobilityManager    Comportamento --> ALIMENTANDO Gravado com Sucesso
0    SN.node[0].MobilityManager    Número Total dos Tipos de Área : 4
0    SN.node[0].MobilityManager    ID do Tipo de Área : 0
0    SN.node[0].MobilityManager    Nome do tipo de Área : Pasto
```

Fonte: Elaborada pelo autor.

$$C_{usto}E_{nergetico}ELL = C_{usto}E_{nergetico}EM * 0,644 \tag{4.2}$$

O módulo *Energy Consumption* foi implementado com os métodos da Figura 23 e são apresentados no Apêndice B.

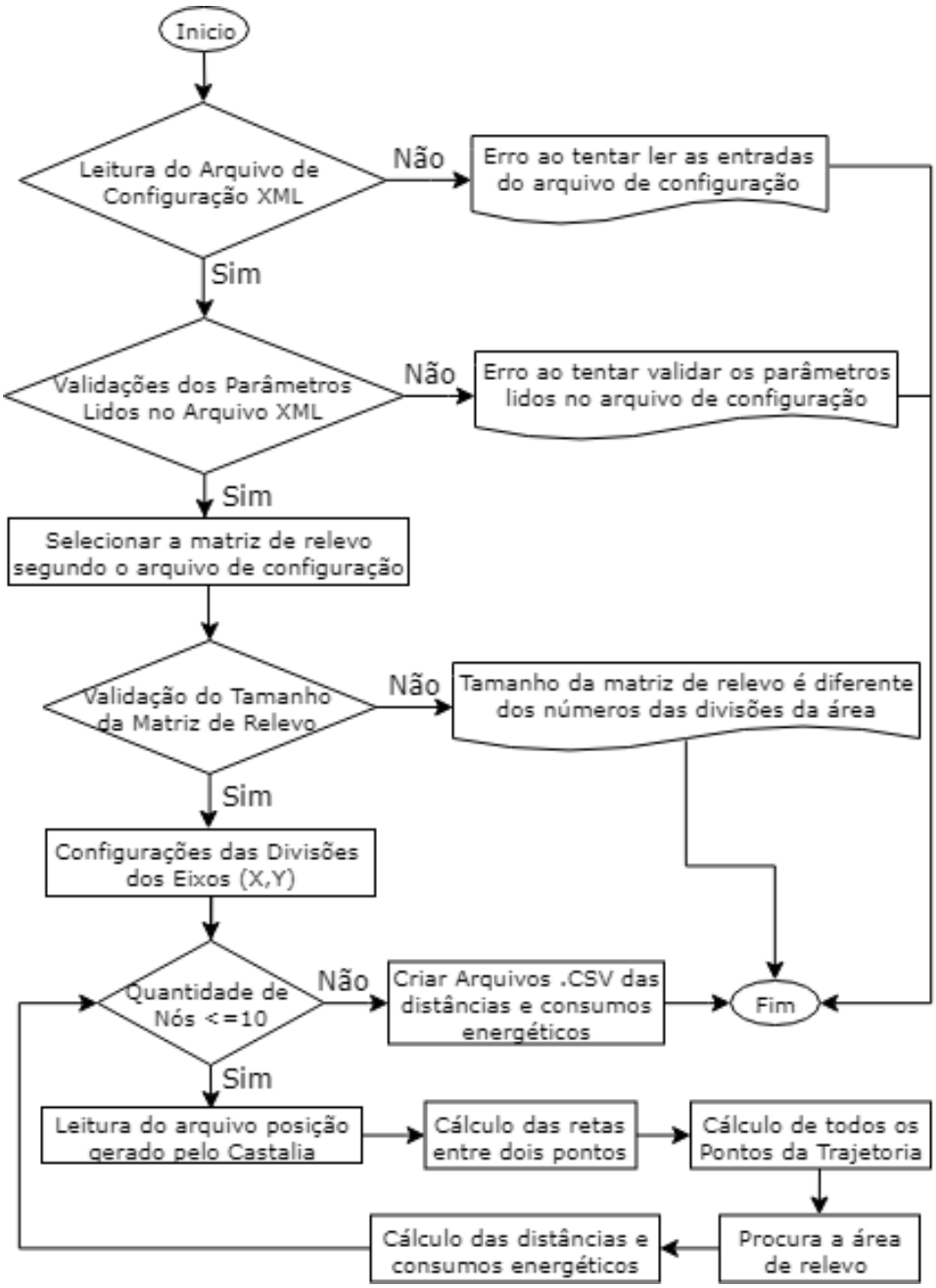
Figura 23 – Métodos do *Energy Consumption*

Energy Consumption

def ReadingFileXML(File_xml):
def ReadingFileLog(File_Log):
def LineEquation(x1,y1,x2,y2):
def Trajectory(m,b,x1,y1,x2,y2,Axis_X,Axis_Y):
def SearchReliefArea(x,y,Axis_X,Axis_Y,flag):
def DistanceCalculation(X,Y):
def EnergyConsumptionCalculation_EM(Weight,D_Horizontal,D_Vertical,C_Horizontal,C_Vertical):
def EnergyConsumptionCalculation_ELL(EnergyConsumption_EM):
def SalveFile (File,Vector):

Fonte: Elaborada pelo autor.

Figura 24 – Diagrama de fluxo do Algoritmo *Energy Consumption*.



Fonte: Elaborada pelo autor.

Algoritmo 2: Cálculo do Consumo Energético dos Animais

Entrada: Arquivo de Configuração .xml
Saída: Arquivos .CSV: Distância, Consumo Energético EM e ELL
início
  if *Leitura do Arquivo de Configuração XML* **then**
    if *Validações dos parâmetros lidos no arquivo de configuração* **then**
      Configura a matriz de relevo;
      if *Validação do tamanho da matriz de relevo* **then**
        Configuração das divisões dos eixos;
        i ← 0;
        repita
          Leitura do arquivo posição gerado pelo castalia;
          Cálculo das retas entre dois pontos;
          Cálculo de todos os pontos da trajetória;
          Procura a área de relevo;
          Cálculo das distâncias e consumos energéticos;
          i ← i + 1;
        até *i < quantidade de animais*;
        Criar Arquivos .CSV das distâncias e consumos energéticos;
      else
        Tamanho da matriz de relevo é diferente dos números das divisões da área;
      end
    else
      Erro ao tentar validar os parâmetros lidos no arquivo de configuração;
    end
  else
    Erro ao tentar ler as entradas do arquivo de configuração;
  end
fim

4.2.2 Funcionamento do *Energy Consumption*

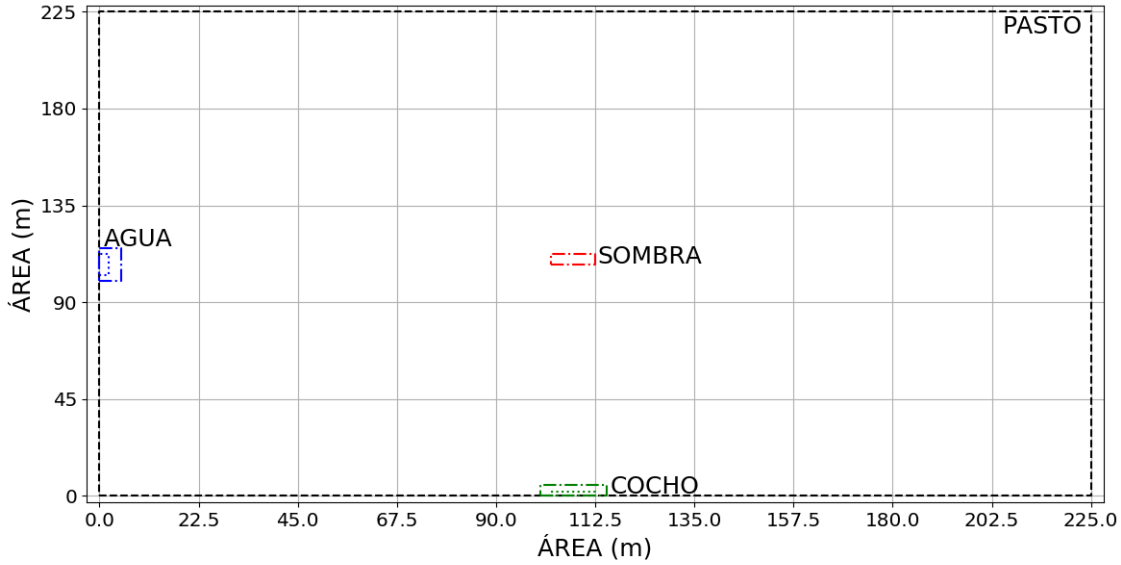
O algoritmo começa carregando as informações contidas no arquivo de configuração *ConfigMobility.xml*, que foi detalhado na subseção 4.1.1. Após, serem validadas essas informações constrói as áreas, os comportamentos, as matrizes de temperatura, relevos, as matrizes markovianas de transição entre as áreas e entre os comportamentos para cada área, etc.

Após, a leitura do arquivo de configuração é feita a verificação de vários parâmetros: os números das divisões da área do pasto com a área máxima estabelecida, o número da

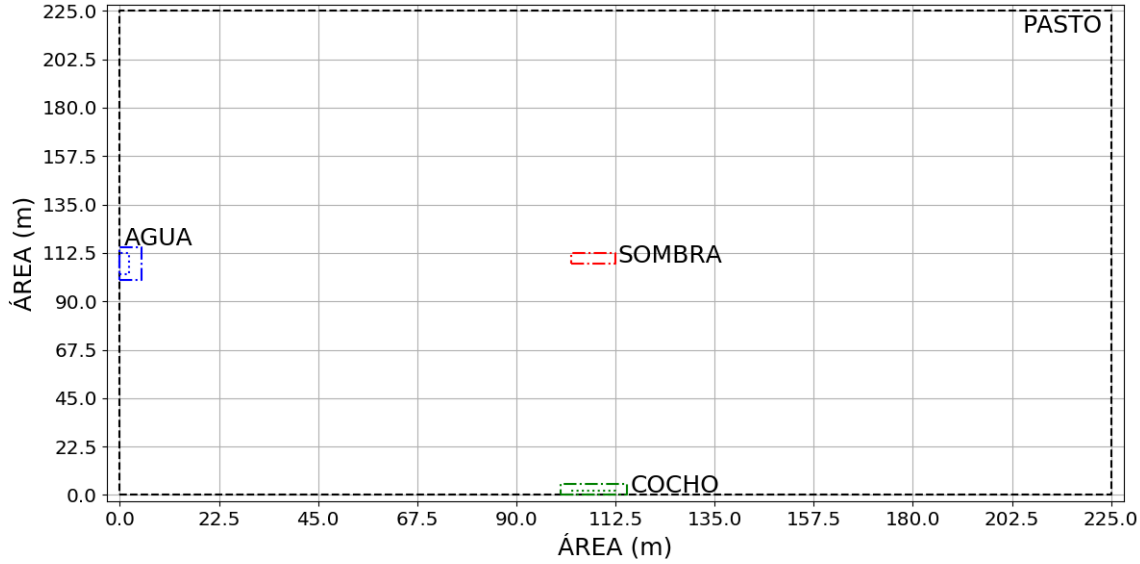
matriz de relevo e temperatura que estejam dentro da quantidade de matrizes configuradas, o tamanho da matriz de relevo com os números das divisões da área. Uma vez validadas essas informações, como a finalidade de não ter algum tipo de erro ao momento de começar a fazer os cálculos, se faz a configuração dos eixos X e Y de tal maneira que fiquem particionados pela quantidade de divisões configuradas, como por exemplos: o eixo X dividido em 10 divisões e o eixo Y dividido em 5 divisões (Figura 25a) e ambos eixos divididos em 10 divisões (Figura 25b).

Figura 25 – Exemplos das divisões dos eixos.

(a) O eixo $X = 10$ e o eixo $Y = 5$



(b) O eixo $X = 10$ e o eixo $Y = 10$

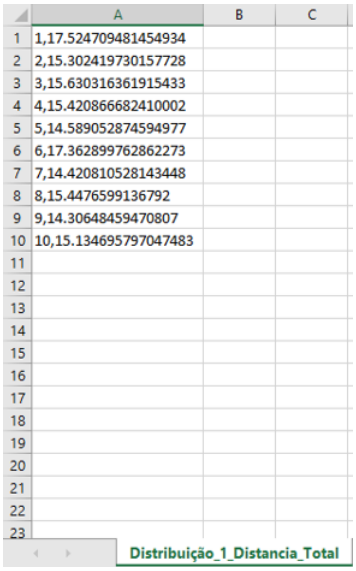


Fonte: Elaborada pelo autor.

Para cada nó configurado se faz o seguinte: leitura do arquivo “ trace_position_node_ index.log”, cálculos das retas entre dois pontos, cálculos de todos os pontos da trajetória, procura a área de relevo referente aos pontos da trajetória, e por último, calcula as distâncias percorridas, o consumo energético EM e o consumo energético ELL. Para todos os resultados são gerados arquivo .csv para armazenar as informações como os apresentados na Figura 26.

Figura 26 – Arquivos .csv gerados após os cálculos.

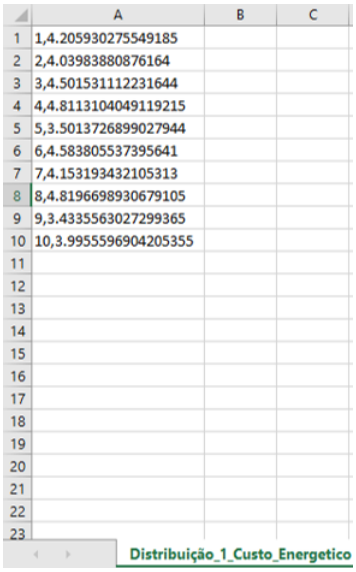
(a) Distância



	A	B	C
1	1,17.524709481454934		
2	2,15.302419730157728		
3	3,15.630316361915433		
4	4,15.420866682410002		
5	5,14.589052874594977		
6	6,17.362899762862273		
7	7,14.420810528143448		
8	8,15.4476599136792		
9	9,14.30648459470807		
10	10,15.134695797047483		
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			

Distribuição_1_Distancia_Total

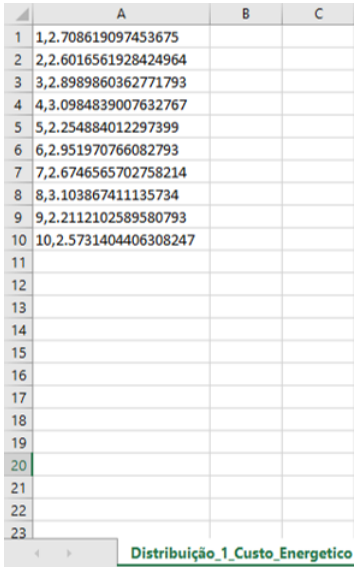
(b) Consumo Energético EM



	A	B	C
1	1,4.205930275549185		
2	2,4.03983880876164		
3	3,4.501531112231644		
4	4,4.8113104049119215		
5	5,3.5013726899027944		
6	6,4.583805537395641		
7	7,4.153193432105313		
8	8,4.8196698930679105		
9	9,3.4335563027299365		
10	10,3.9955596904205355		
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			

Distribuição_1_Custo_Energetico

(c) Consumo Energético ELL



	A	B	C
1	1,2.708619097453675		
2	2,2.6016561928424964		
3	3,2.8989860362771793		
4	4,3.0984839007632767		
5	5,2.254884012297399		
6	6,2.951970766082793		
7	7,2.6746565702758214		
8	8,3.103867411135734		
9	9,2.2112102589580793		
10	10,2.5731404406308247		
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			

Distribuição_1_Custo_Energetico

Fonte: Elaborada pelo autor.

5 AVALIAÇÃO

No capítulo anterior, foi descrita a arquitetura. Ela provê uma flexibilidade para configurar os parâmetros referentes aos animais e às áreas do pasto. Além disso, o modelo simula os comportamentos e caminhos percorridos dos animais em pastagem o mais próximo à realidade.

Assim, este capítulo tem o intuito de avaliar a arquitetura proposta, que modela os comportamentos e calcula o consumo energético para gado leiteiro, escolhido como temática para a avaliação. Além disso, calcula o consumo energético da rede através de protocolos de roteamento. Na seção 5.1 é apresentada as parametrizações básicas para avaliar a arquitetura. Após isso, a avaliação do consumo de energia de gado leiteiro é apresentada na seção 5.2 e do consumo de energia da rede é apresentada na seção 5.3.

5.1 PARAMETRIZAÇÕES BÁSICAS

Basicamente os sistemas de criação convencionais de animais podem ser classificados em três: Extensivo, Semiextensivo e Intensivo [41]. Neste trabalho foi utilizado o semiextensivo onde os animais permanecem no pasto apenas parte do dia, recebendo suplementação alimentar em cochos. Esse sistema é adotado tanto para a produção de carnes, peles, leite, em geral. Portanto as áreas escolhidas são: Pasto, Água, Sombra e Cocho.

Sabe-se que os fatores ambientais relacionam-se diretamente com o comportamento animal [54]. A temperatura, a radiação solar e a umidade relativa são os principais elementos do ambiente relacionados ao comportamento bovino [3]. Então, neste trabalho escolheu-se três faixas de temperaturas: 1) $20°C \leq T < 26°C$, 2) $26°C \leq T < 32°C$ e 3) $32°C \leq T \leq 36°C$, considerando uma matriz de transições das áreas diferentes por cada faixa de temperatura.

Os valores das matrizes markovianas da Tabela 4 que pertence às transições das áreas foram escolhidos de acordo com os valores e descrições encontrados em vários trabalhos científicos que abordam o assunto. Os valores de transições entre as áreas são diferentes em cada matriz. Por exemplo, conforme descrito no trabalho do Oliveira [3], a probabilidade dos animais procurarem a área da sombra e da área da água é alta quando existem altas temperaturas com o objetivo de reduzir a temperatura corporal.

O *Mobility Manager* possui matrizes markovianas (Tabela 4) de transições entres as áreas (Pasto, Água, Cocho e Sombra), que determinará em qual área específica um determinado grupo de nós deve se encontrar. A seleção do *ID* da matriz vai depender do tempo real (Segundos) que a sua vez está relacionada com a temperatura (°C) da Tabela 5. Também, possui matrizes markovianas (Tabela 6) de transições dos comportamentos

dentro de cada área (Andando, Correndo, Deitado e Alimentando) que irá determinar em qual comportamento específico um determinado nó deve-se encontrar e assim determinar a velocidade de movimentação.

Tabela 4 – Matrizes Markovianas de Transições das Áreas Baseadas na Temperatura.

(a) ID: 0, 20°C ≤ T < 26°C

ÁREAS	Pasto	Água	Cocho	Sombra
Pasto	0,4	0,4	0,3	0,4
Água	0,2	0,2	0,3	0,2
Cocho	0,2	0,3	0,2	0,2
Sombra	0,2	0,1	0,2	0,2

(b) ID: 1, 26°C ≤ T < 32°C

ÁREAS	Pasto	Água	Cocho	Sombra
Pasto	0,35	0,3	0,2	0,3
Água	0,3	0,35	0,3	0,15
Cocho	0,15	0,15	0,3	0,15
Sombra	0,2	0,2	0,2	0,4

(c) ID: 2, 32°C ≤ T ≤ 36°C

ÁREAS	Pasto	Água	Cocho	Sombra
Pasto	0,1	0,1	0,1	0,1
Água	0,4	0,4	0,4	0,4
Cocho	0,1	0,1	0,1	0,1
Sombra	0,4	0,4	0,4	0,4

Fonte: Elaborada pelo autor.

Os valores de temperaturas da matriz da Tabela 5 foram escolhidos de tal forma a ter variações de temperatura, sendo uma temperatura mínima = 20°C e uma temperatura máxima = 36°C, com a finalidade de fazer a prova de conceito utilizando todas as matrizes markovianas de transições das áreas da Tabela 4 e ter a influência da temperatura no comportamento do animal.

Tabela 5 – Matriz de Temperaturas Baseada na Hora do Dia.

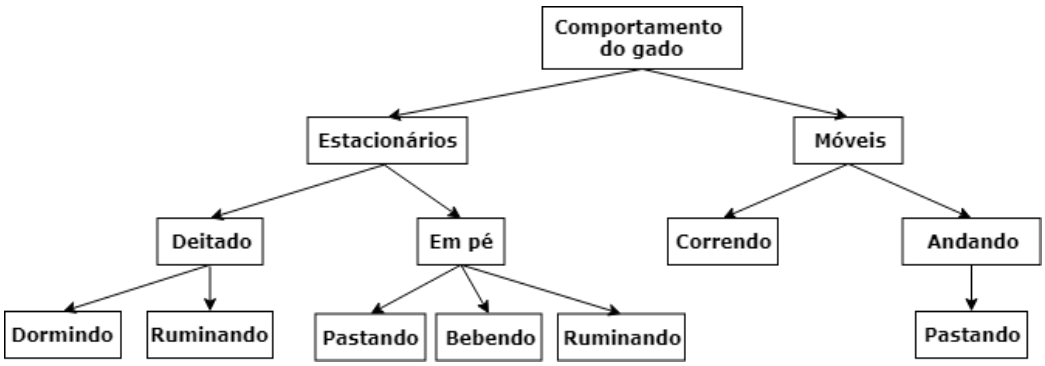
	7-8h	8-9h	9-10h	10-11h	11-12h	12-13h	13-14h	14-15h	15-16h	16-17h
Tmín (°C)	20	22	24	26	28	30	32	34	32	30
Tmáx (°C)	22	24	26	28	30	32	34	36	34	32

Fonte: Elaborada pelo autor.

Os estados de comportamentos dos animais podem ser classificados em subclasses de acordo com diferentes padrões e propósitos, mostrados na Figura 27. A estrutura hierárquica de classificação é feita a partir das atividades da camada mais alta, ou seja, para identificar entre estados estacionários e móveis [60].

Guo et al. [60] dizem que o comportamento “Pastando” foi classificado como um comportamento estático e também um comportamento móvel, sendo colocado como comportamento secundário do comportamento “Em Pé” e do comportamento “Andando”. Além disso, o comportamento “Ruminando” também foi classificado nos dos dois tipos de comportamentos estacionários: “Deitado” e “Em Pé”. Na configuração do módulo *Mobility Manager* decidiu-se agrupar três comportamentos (“Pastando”, “Ruminando” e “Bebendo”) em um só comportamento denominado “Alimentando” que possui baixa mobilidade que é menor do que o comportamento “Andando” [5].

Figura 27 – Classificação do Comportamento do Gado.



Fonte: [60].

Os valores das matrizes markovianas da Tabela 6, que pertencem às transições dos comportamentos dentro de cada área, foram escolhidos de acordo com as análises feitas na dissertação do Lomba [54]. Os valores de transições entre esses comportamentos são diferentes em cada área, por exemplo, o comportamento "Alimentando", possui probabilidade nas áreas: "Água" (100%), "Cocho" (100%), "Pasto" (60%) e "Sombra" (30%), esse comportamento é um agrupamento dos comportamentos "Bebendo", "Pastando" e "Ruminando" que se encontram na literatura.

Tabela 6 – Matrizes Markovianas de Transições dos Comportamentos nas Áreas.

(a) ID: 0, Pasto.

PASTO	Andando	Correndo	Deitado	Alimentando
Andando	0,4	0,5	0,3	0,3
Correndo	0,1	0,5	0	0
Deitado	0,1	0	0,5	0,1
Alimentando	0,4	0	0,2	0,6

(b) ID: 1, Água.

ÁGUA	Andando	Correndo	Deitado	Alimentando
Andando	0	0	0	0
Correndo	0	0	0	0
Deitado	0	0	0	0
Alimentando	1	1	1	1

(c) ID: 2, Cocho.

COCHO	Andando	Correndo	Deitado	Alimentando
Andando	0	0	0	0
Correndo	0	0	0	0
Deitado	0	0	0	0
Alimentando	1	1	1	1

(d) ID: 3, Sombra.

SOMBRA	Andando	Correndo	Deitado	Alimentando
Andando	0,3	0,7	0,3	0,2
Correndo	0,1	0,3	0	0
Deitado	0,5	0	0,6	0,5
Alimentando	0,1	0	0,1	0,3

Fonte: Elaborada pelo autor.

Para cada tipo de comportamento foi escolhida uma faixa de velocidade, apresentadas na Tabela 7. Essas faixas foram deduzidas do trabalho [47], que fez um estudo do desempenho da caminhada do gado leiteiro.

Tabela 7 – Velocidades dependendo do Tipo de Comportamento.

Comportamentos	**Velocidade Mínima (m/s)**	**Velocidade Máxima (m/s)**
Andando	0,277	0,833
Correndo	0,833	1,388
Deitado	0	0
Alimentando	0,013	0,027

Fonte: Elaborada pelo autor.

Vale lembrar que, por se tratar de uma arquitetura flexível e configurável, pode-se acrescentar quantos tipos de áreas e comportamentos forem necessários para se aproximar ao comportamento animal desejado.

5.2 CONSUMO DE ENERGIA DO GADO LEITEIRO

5.2.1 **Parâmetros das Simulações do Consumo de Energia do Gado Leiteiro**

Os parâmetros para as simulações são apresentados na Tabela 8. Elas foram feitas utilizando três distribuições de áreas diferentes, Figuras 28, 29 e 30, onde o tamanho da área total (Pasto) é a mesma e só mudam as localizações das subáreas (Água, Cocho e Sombra). As dimensões de cada distribuição estão na Tabela 9, 10 e 11. Embora o módulo *Mobility Manager* possa simular duas estratégias de seleção de área (Randômico e Gauss-Markov), nesse trabalho é apresentada a estratégia do Gauss-Markov, por apresentar movimentações mais reais.

Executou-se uma simulação com um tempo de 36000 segundos (10 horas) para simular um cenário desde das 7h até às 17h, com 10 nós móveis dividido em dois grupos, o nó 1 será o líder dos 5 primeiros nós e o nó 6 será o líder dos 5 últimos nós. A área de simulação foi de 50625 m^2 calculada com a equação 5.1. A configuração utilizada do módulo do canal sem fio do simulador foi de nenhum nó estático, e as configurações do gerenciador de comunicação, aplicação e roteamento foram as padrões do simulador, já que o objetivo principal é verificar o funcionamento do módulo *Mobility Manager* e do módulo *Energy Consumption*. A estratégia de seleção de área = 1 para utilizar o modelo de Gauss-Markov e o arquivo de configuração foi *ConfigMobility.xml* onde se encontram as informações do gado leiteiro.

Tabela 8 – Parâmetros dos Testes do Consumo de Energia de Rebanho Leiteiro.

Parâmetros		**Valores**
Tempo da Simulação		36000 Seg
Área da Simulação		225m x 225m = 50625 m^2
Número de Nós		10
Canal Sem Fio	Nós Estáticos	False
Gerenciador de Comunicação	Tipo do Rádio	CC2420
	Potência do Rádio	-5dBm
Gerenciador de Aplicação	Nome	ThroughputTest
Protocolo de Roteamento	Nome	BypassRouting
Gerenciador de Mobilidade	Nome	MobilityManager
	Arquivo de Configuração	ConfigMobility.xml
	Estratégia de Seleção de Área	1
	Número de Grupos	2
	Nó Líder	1 → [1 - 5]
	Nó Líder	6 → [6 - 10]
	Intervalo de Atualização	100

Fonte: Elaborada pelo autor.

Para definir os tamanhos das áreas de simulação: Pasto, Água, Cocho e Sombra, foram utilizadas as relações:

$$Area_{Pasto}\left(m^2\right) = 10.000m^2 * NumeroDeCabeça \tag{5.1}$$

$$Area_{Agua}\left(m^2\right) = (0,7m * NumeroDeCabeça) * 0,7m \tag{5.2}$$

$$Area_{Cocho}\left(m^2\right) = (0,8m * NumeroDeCabeça) * 1m \tag{5.3}$$

$$Area_{Sombra}\left(m^2\right) = 5m^2 * NumeroDeCabeça \tag{5.4}$$

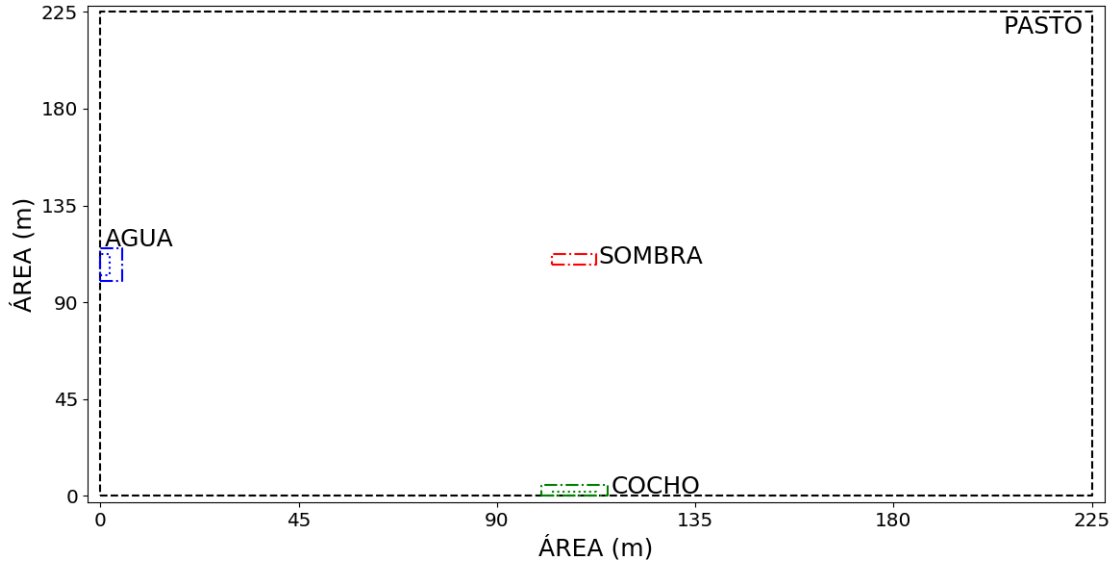
$$Peso_{Vivo}\left(K_g\right) = 557 \pm 30K_g \tag{5.5}$$

Tabela 9 – Dimensões da Distribuição 1.

Área	**Xmín**	**Xmáx**	**Ymín**	**Ymáx**
PASTO	0	225	0	225
ÁGUA	0	5	100	115
COCHO	100	115	0	5
SOMBRA	102,5	112,5	107,5	112,5

Fonte: Elaborada pelo autor.

Figura 28 – Distribuição 1.



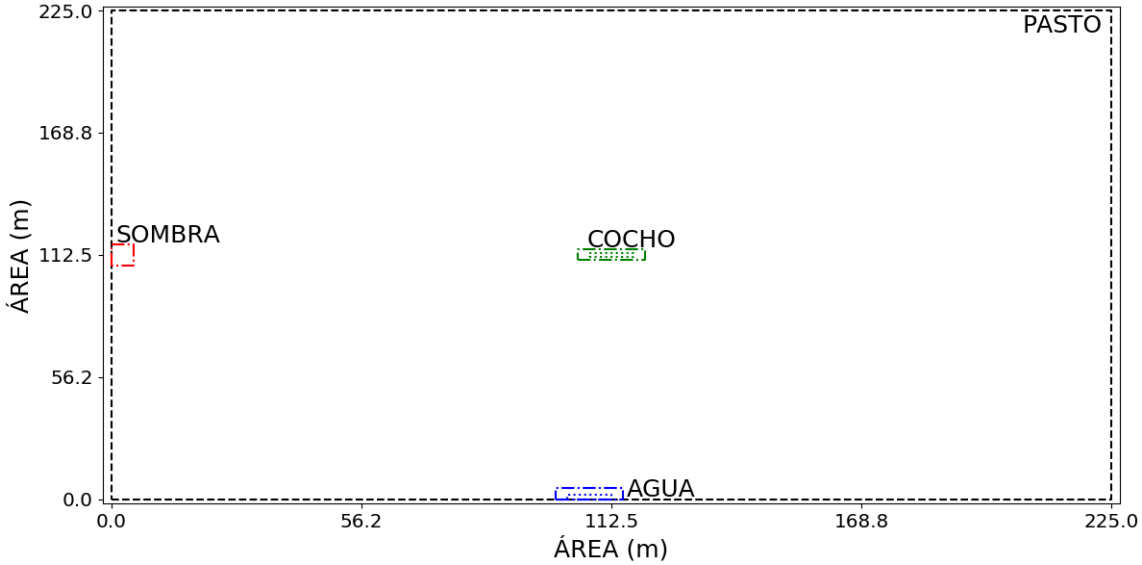
Fonte: Elaborada pelo autor.

Tabela 10 – Dimensões da Distribuição 2.

Área	**Xmín**	**Xmáx**	**Ymín**	**Ymáx**
PASTO	0	225	0	225
ÁGUA	100	115	0	5
COCHO	105	120	110	115
SOMBRA	0	5	107,5	117,5

Fonte: Elaborada pelo autor.

Figura 29 – Distribuição 2.



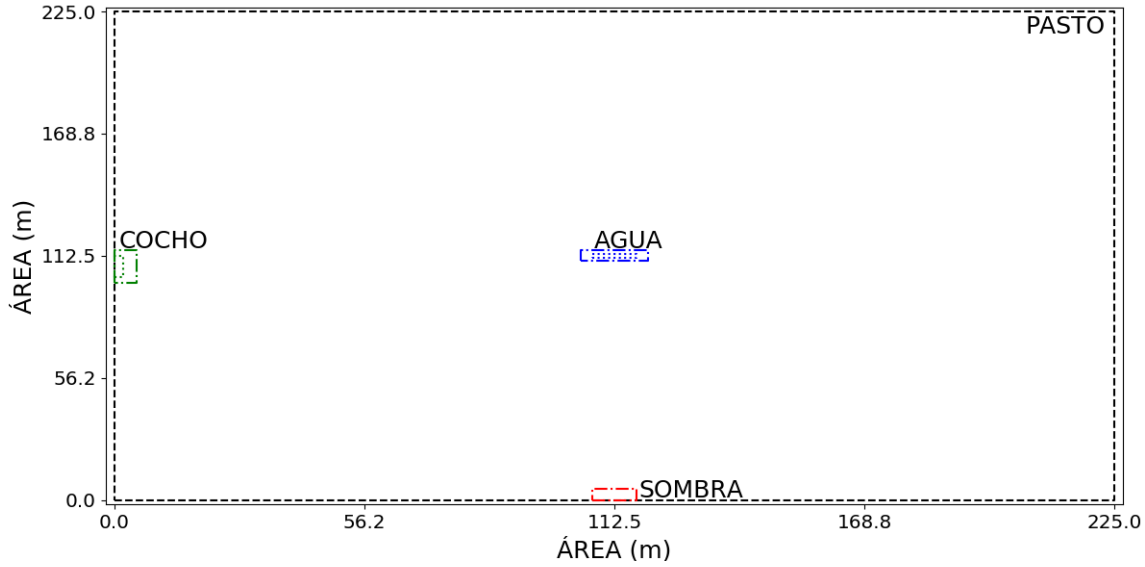
Fonte: Elaborada pelo autor.

Tabela 11 – Dimensões da Distribuição 3.

Área	Xmín	Xmáx	Ymín	Ymáx
PASTO	0	225	0	225
ÁGUA	105	120	110	115
COCHO	0	5	100	115
SOMBRA	107,5	117,5	0	5

Fonte: Elaborada pelo autor.

Figura 30 – Distribuição 3.



Fonte: Elaborada pelo autor.

5.2.2 **Resultados das Simulações**

Em cada distribuição de área, foram calculadas a distância total percorrida por cada bovino, o consumo de energia metabólico (EM) e o consumo de energia líquida de lactação (ELL). Também, um mapa de calor foi elaborado para ver as áreas mais visitadas pelo rebanho e por último para mostrar o funcionamento do modelo de mobilidade em grupo é apresentado um gráfico mostrando as mudanças de áreas e comportamentos. Para cada distribuição de áreas foram configuradas três matrizes de relevo, apresentadas na Tabela 12 para simular uma área totalmente plana (a), uma área em aclive (b) e outra área com aclive e declive (c). As divisões da área total foi feita em 4 partes para o eixo x como para o eixo y, mas esse número de divisões é configurável.

Tabela 12 – Matrizes das Áreas de Relevo.

(a) ID: 0. Matriz de Área Plana

225 x 225 m	**0 ≤ X < 56,25**	**56,25 ≤ X < 112,5**	**112,5 ≤ X < 168,75**	**168,75 ≤ X ≤ 225**
0 ≤ Y < 56,25	0 %	0 %	0 %	0 %
56,25 ≤ Y < 112,5	0 %	0 %	0 %	0 %
112,5 ≤ Y < 168,75	0 %	0 %	0 %	0 %
168,75 ≤ Y ≤ 225	0 %	0 %	0 %	0 %

(b) ID: 1. Matriz de Área em Aclive

225 x 225 m	**0 ≤ X < 56,25**	**56,25 ≤ X < 112,5**	**112,5 ≤ X < 168,75**	**168,75 ≤ X ≤ 225**
0 ≤ Y < 56,25	0 %	2,5 %	5 %	10 %
56,25 ≤ Y < 112,5	2,5 %	2,5 %	5 %	10 %
112,5 ≤ Y < 168,75	5 %	5 %	10 %	15 %
168,75 ≤ Y ≤ 225	10 %	10 %	15 %	20 %

(c) ID: 2. Matriz de Área com Aclive e Declive

225 x 225 m	**0 ≤ X < 56,25**	**56,25 ≤ X < 112,5**	**112,5 ≤ X < 168,75**	**168,75 ≤ X ≤ 225**
0 ≤ Y < 56,25	2,5 %	5 %	10 %	20 %
56,25 ≤ Y < 112,5	-5 %	10 %	15 %	25 %
112,5 ≤ Y < 168,75	-10 %	-10 %	20 %	30 %
168,75 ≤ Y ≤ 225	-20 %	-20 %	25 %	35 %

Fonte: Elaborada pelo autor.

Os resultados da distância total percorrida por todos os bovinos leiteiros, o consumo de energia EM e o consumo de energia ELL de todas as simulações da distribuição 1, distribuição 2 e distribuição 3 com as três matrizes de relevo diferentes são mostrados nas Tabelas 13, 14 e 15, respectivamente. O consumo de energia maior foi usando a ID da matriz de relevo: 2, pois possui uma grande área de relevo e o menor consumo de energia foi usando a ID da matriz de relevo: 0, porque não possui área de relevo, pois a configuração é de uma área plana.

Tabela 13 – Resultados do Rebanho utilizando a Distribuição 1

		Consumo Energético	
Matriz de Relevo	**Distância (Km)**	**EM (MCal)**	**ELL (MCal)**
ID: 0	155,139	42,045	27,077
ID: 1	157,672	44,860	28,890
ID: 2	156,844	48,178	31,026

Fonte: Elaborada pelo autor.

Tabela 14 – Resultados do Rebanho utilizando a Distribuição 2

		Consumo Energético	
Matriz de Relevo	**Distância (Km)**	**EM (MCal)**	**ELL (MCal)**
ID: 0	155,649	42,116	27,123
ID: 1	154,183	43,981	28,324
ID: 2	152,674	46,836	30,162

Fonte: Elaborada pelo autor.

Tabela 15 – Resultados do Rebanho utilizando a Distribuição 3

		Consumo Energético	
Matriz de Relevo	**Distância (Km)**	**EM (MCal)**	**ELL (MCal)**
ID: 0	155,018	41,874	26,966
ID: 1	166,633	47,738	30,743
ID: 2	155,841	48,084	30,966

Fonte: Elaborada pelo autor.

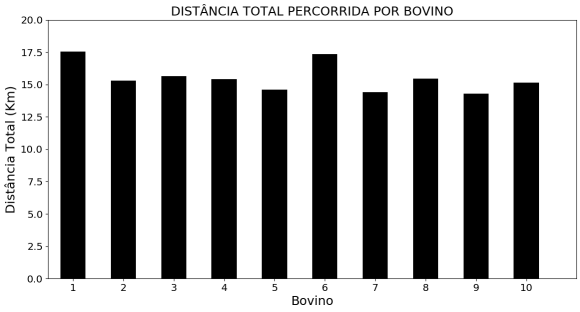
5.2.2.1 *Distribuição 1*

- Distância e Consumos Energéticos

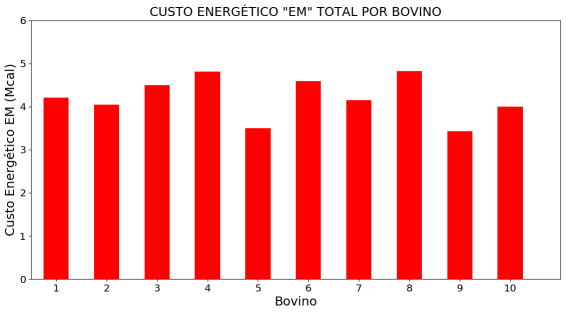
  Nas Figuras 31a, 31d, 32g se apresentam as distâncias totais percorridas, nas Figuras 31b, 31e, 32h se apresentam os consumos energéticos EM e nas Figuras 31c, 31f, 32i se apresentam os consumos energéticos ELL, por cada bovino leiteiro.

Figura 31 – Distribuição 1 por Matriz de Relevo.

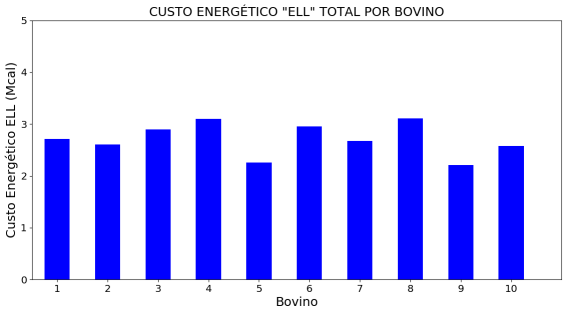
(a) ID: 0



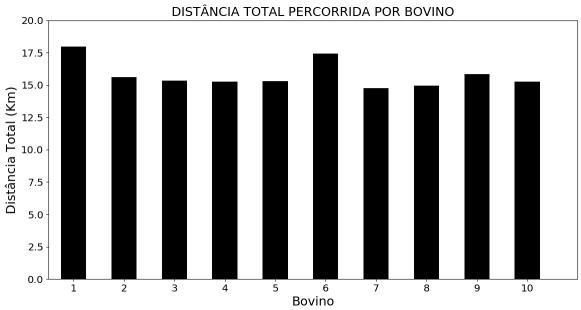
(b) ID: 0



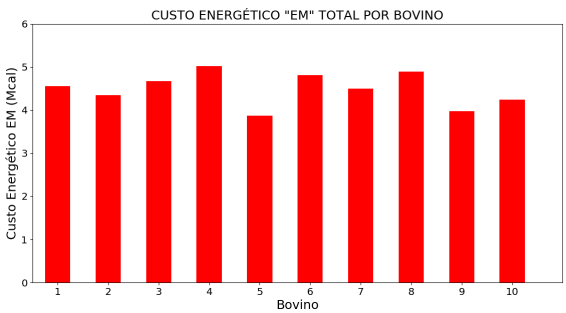
(c) ID: 0



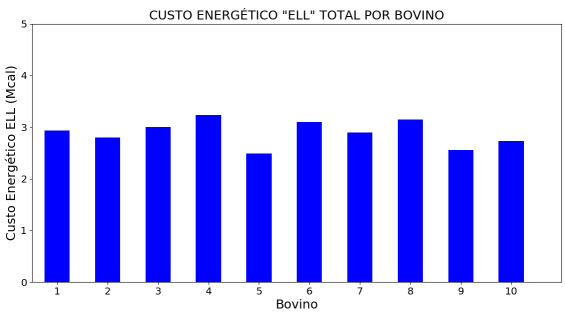
(d) ID: 1

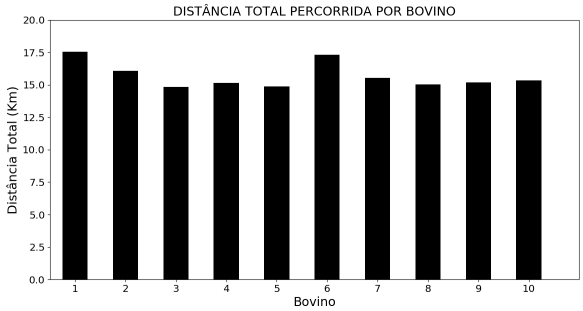


(e) ID: 1

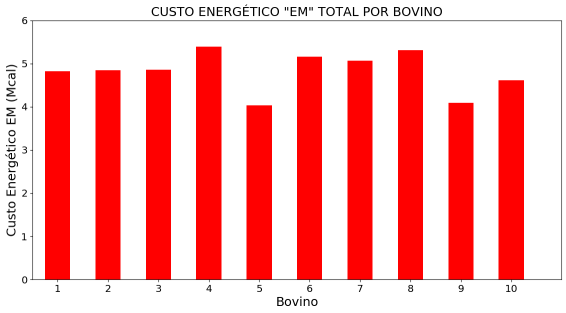


(f) ID: 1

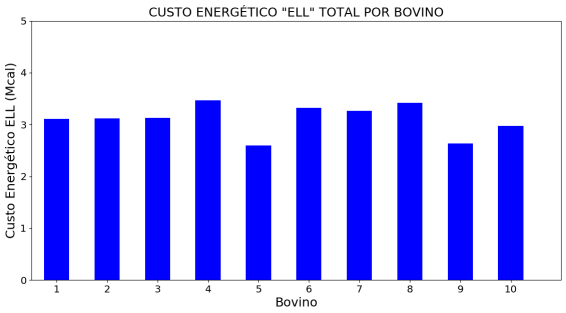




(g) ID: 2



(h) ID: 2



(i) ID: 2

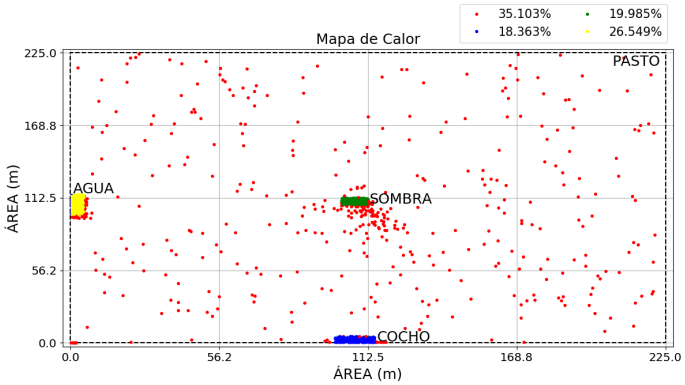
Fonte: Elaborada pelo autor.

- Mapa de Calor das Áreas Visitadas

  Nas Figuras 32a, 33c, 33e se apresentam os mapas de calor das movimentações do gado leiteiro por todas as áreas e nas Figuras 32b, 33d, 33f se apresentam as quantidades de vezes por cada hora que as áreas foram visitadas que tem a ver com a matriz de temperaturas e as matrizes de transições das áreas. Neste caso, o pasto ficou como a área mais visitada, já que é a maior área e onde os animais passam quase o dia inteiro, depois ficou a área da água e a área da sombra e o cocho ficaram como as áreas menos visitadas. Já que, a movimentação vai depender das configurações das matrizes de transições das áreas.

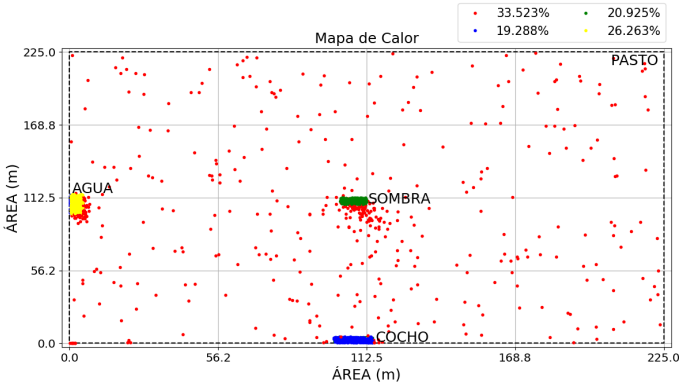
Figura 32 – Mapa de Calor da Distribuição 1.

(a) Matriz de Relevo ID: 0



(b) Áreas Visitadas

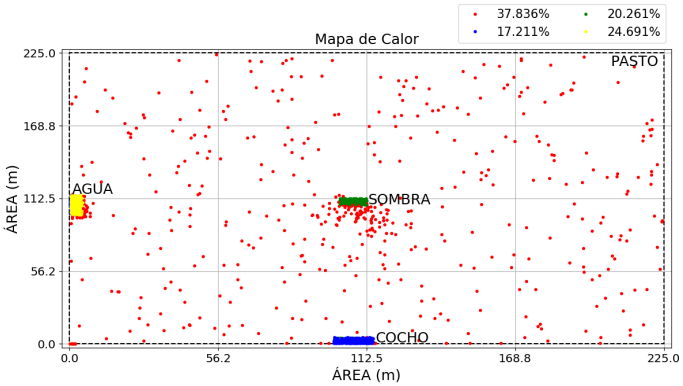
	Pasto	Agua	Cocho	Sombra
07:00 - 08:00	53	38	28	28
08:00 - 09:00	45	30	33	19
09:00 - 10:00	47	34	26	5
10:00 - 11:00	45	36	23	35
11:00 - 12:00	54	30	28	27
12:00 - 13:00	56	17	31	35
13:00 - 14:00	40	46	18	35
14:00 - 15:00	36	46	19	45
15:00 - 16:00	40	46	18	35
16:00 - 17:00	38	33	33	31



(c) Matriz de Relevo ID: 1

	Pasto	Agua	Cocho	Sombra
07:00 - 08:00	43	27	48	19
08:00 - 09:00	30	38	41	24
09:00 - 10:00	50	35	15	14
10:00 - 11:00	54	27	28	22
11:00 - 12:00	42	36	23	14
12:00 - 13:00	43	34	33	11
13:00 - 14:00	49	47	14	44
14:00 - 15:00	35	52	15	51
15:00 - 16:00	49	47	14	44
16:00 - 17:00	35	40	32	28

(d) Áreas Visitadas



(e) Matriz de Relevo ID: 2

	Pasto	Agua	Cocho	Sombra
07:00 - 08:00	65	15	47	19
08:00 - 09:00	63	18	40	31
09:00 - 10:00	58	31	15	18
10:00 - 11:00	73	16	33	17
11:00 - 12:00	58	38	12	23
12:00 - 13:00	58	38	29	25
13:00 - 14:00	54	33	21	29
14:00 - 15:00	44	60	11	29
15:00 - 16:00	54	33	21	29
16:00 - 17:00	65	21	28	35

(f) Áreas Visitadas

Fonte: Elaborada pelo autor.

5.2.2.2 *Distribuição 2*

- Distância e Consumos Energéticos

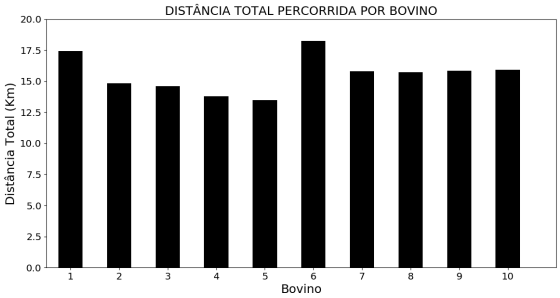
  Nas Figuras 33a, 33d, 33g se apresentam as distâncias totais percorridas, nas Figuras 33b, 33e, 33h se apresentam os consumos energéticos EM e nas Figuras 33c, 33f, 33i se apresentam os consumos energéticos ELL, por cada bovino leiteiro.

- Mapa de Calor das Áreas Visitadas

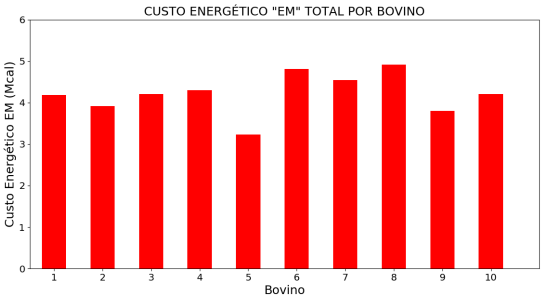
  Nas Figuras 34a, 34c, 34e se apresentam os mapas de calor das movimentações do gado por todas as áreas e nas Figuras 34b, 34d, 34f se apresentam as quantidades de vezes por cada hora que as zonas foram visitadas que tem a ver com a matriz de temperaturas e as matrizes de transições das áreas. Neste caso, o pasto ficou como a área mais visitada, já que é a maior área e onde os animais passam quase o dia inteiro, depois ficou a área da água e a área da sombra e o cocho ficaram como as áreas menos visitadas. Já que, a movimentação vai depender das configurações das matrizes de transições das áreas.

Figura 33 – Distribuição 2 por Matriz de Relevo.

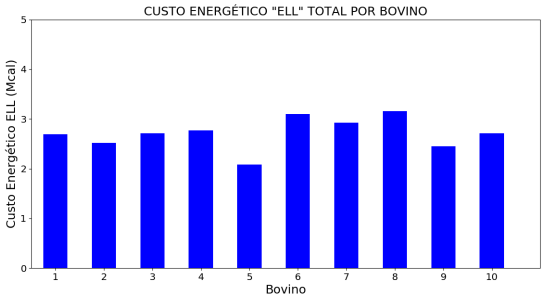
(a) ID: 0



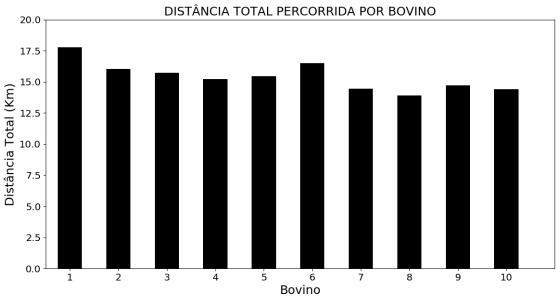
(b) ID: 0



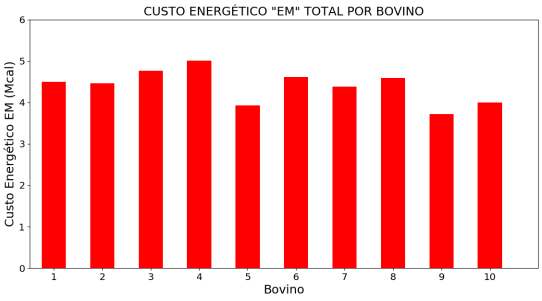
(c) ID: 0



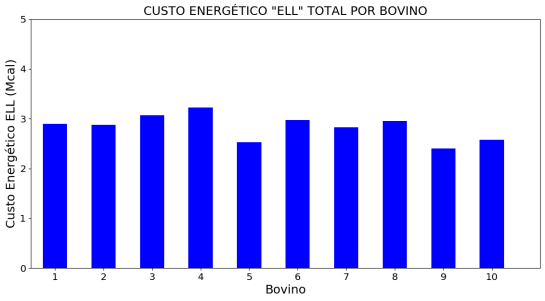
(d) ID: 1



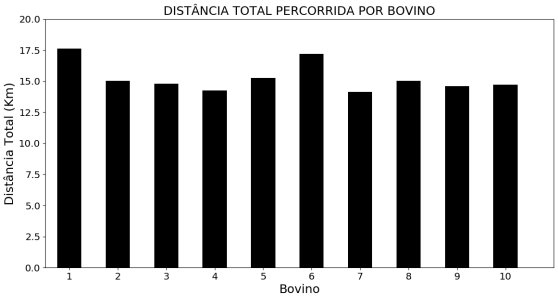
(e) ID: 1



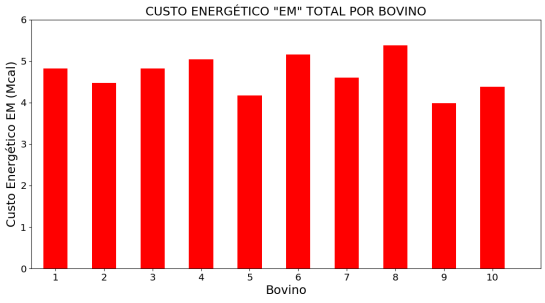
(f) ID: 1



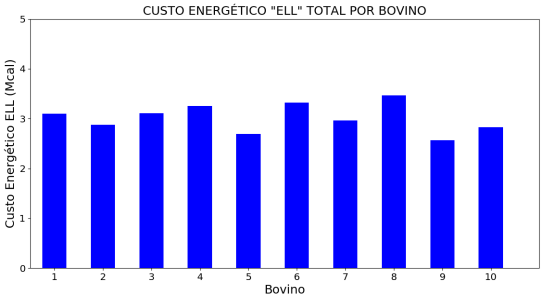
(g) ID: 2



(h) ID: 2



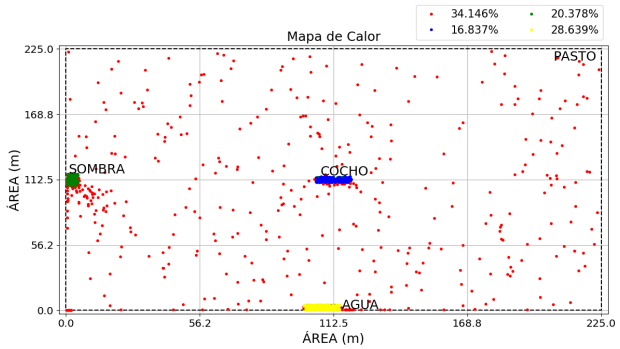
(i) ID: 2



Fonte: Elaborada pelo autor.

Figura 34 – Mapa de Calor da Distribuição 2.

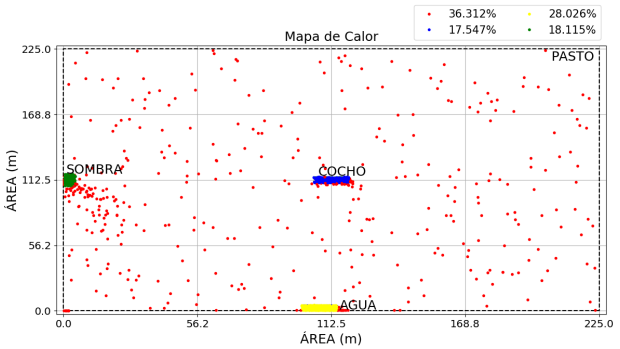
(a) Matriz de Relevo ID: 0



(b) Áreas Visitadas

	Pasto	Agua	Cocho	Sombra
07:00 - 08:00	48	29	28	25
08:00 - 09:00	53	37	20	26
09:00 - 10:00	41	45	39	15
10:00 - 11:00	46	41	28	12
11:00 - 12:00	52	27	11	36
12:00 - 13:00	37	41	21	25
13:00 - 14:00	24	43	20	28
14:00 - 15:00	41	38	10	39
15:00 - 16:00	30	43	13	34
16:00 - 17:00	62	20	24	19

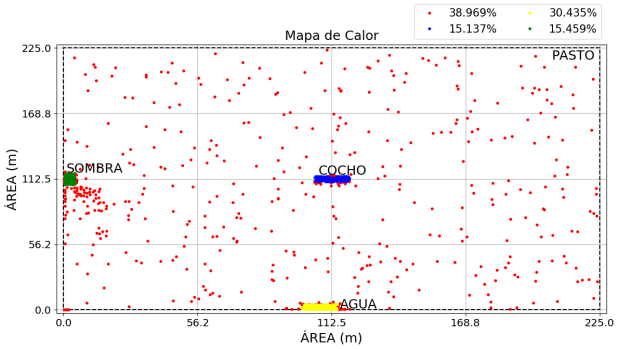
(c) Matriz de Relevo ID: 1



(d) Áreas Visitadas

	Pasto	Agua	Cocho	Sombra
07:00 - 08:00	53	36	23	13
08:00 - 09:00	48	31	34	17
09:00 - 10:00	49	36	16	19
10:00 - 11:00	57	23	27	22
11:00 - 12:00	36	33	37	25
12:00 - 13:00	51	43	28	15
13:00 - 14:00	30	38	11	49
14:00 - 15:00	29	40	13	44
15:00 - 16:00	30	38	11	49
16:00 - 17:00	42	48	25	16

(e) Matriz de Relevo ID: 2



(f) Áreas Visitadas

	Pasto	Agua	Cocho	Sombra
07:00 - 08:00	55	35	23	18
08:00 - 09:00	56	35	32	13
09:00 - 10:00	53	39	18	16
10:00 - 11:00	59	24	22	17
11:00 - 12:00	48	48	16	24
12:00 - 13:00	42	27	17	29
13:00 - 14:00	25	47	16	43
14:00 - 15:00	25	47	11	31
15:00 - 16:00	25	47	16	43
16:00 - 17:00	54	23	16	19

Fonte: Elaborada pelo autor.

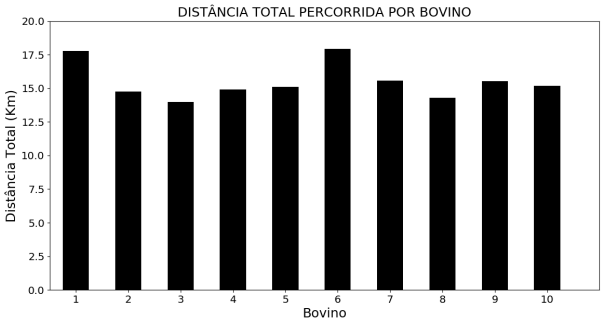
5.2.2.3 *Distribuição 3*

- Distância e Consumos Energéticos

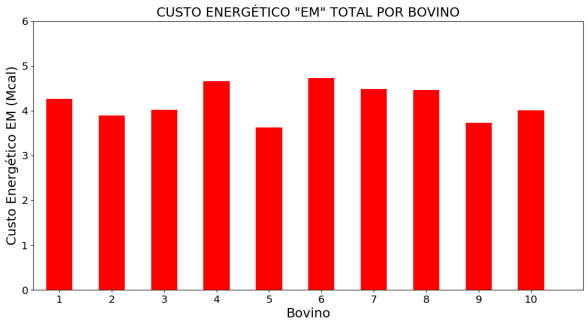
  Nas Figuras 35a, 35d, 35g se apresentam as distâncias totais percorridas, nas Figuras 35b, 35e, 35h se apresentam os consumos energéticos EM e nas Figuras 35c, 35f, 36i se apresentam os consumos energéticos ELL, por cada bovino leiteiro.

Figura 35 – Distribuição 3 por Matriz de Relevo.

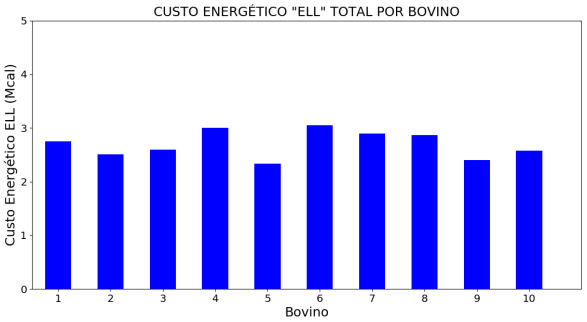
(a) ID: 0



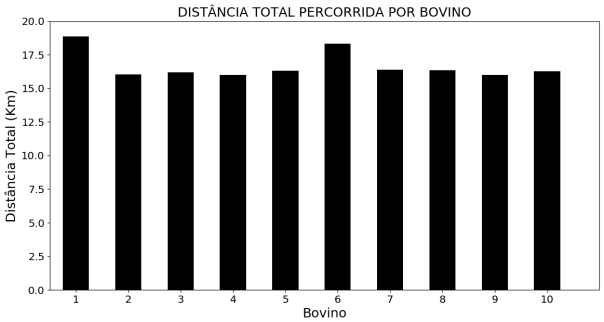
(b) ID: 0



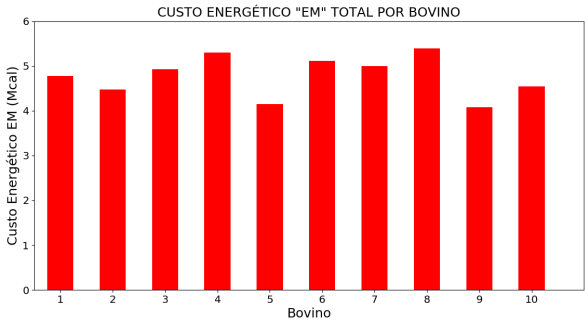
(c) ID: 0



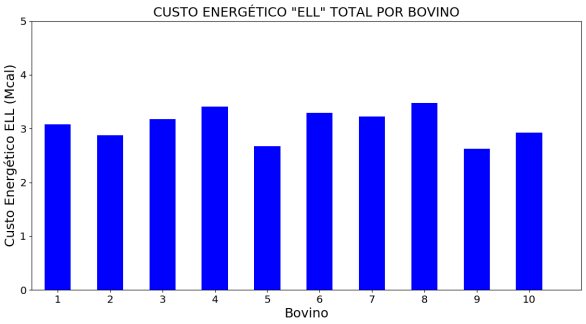
(d) ID: 1



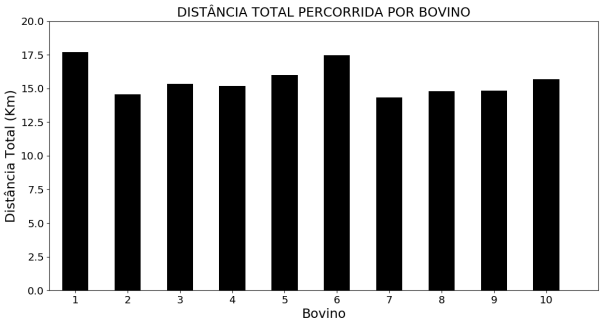
(e) ID: 1



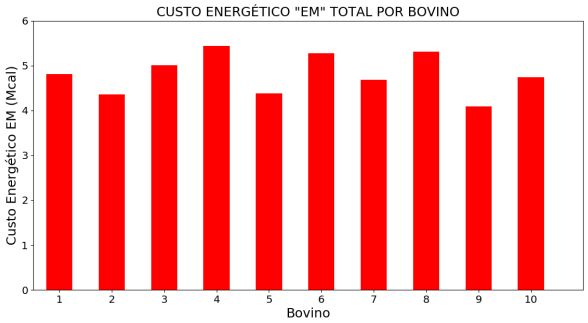
(f) ID: 1

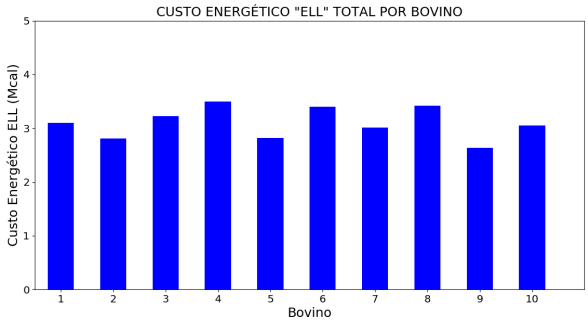


(g) ID: 2



(h) ID: 2





(i) ID: 2

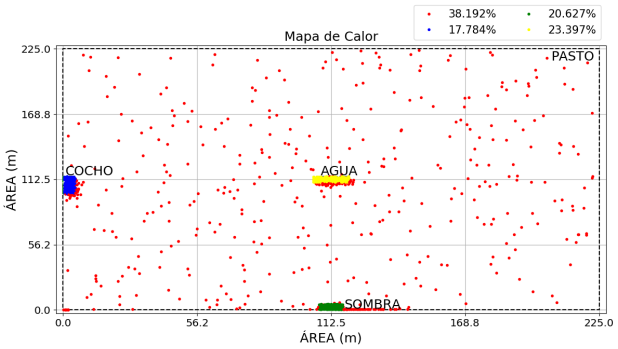
Fonte: Elaborada pelo autor.

- Mapa de Calor das Áreas Visitadas

  Nas Figuras 36a, 36c, 37e se apresentam os mapas de calor das movimentações do gado por todas as áreas e nas Figuras 36b, 36d, 37f se apresentam as quantidades de vezes por cada hora que as zonas foram visitadas que tem a ver com a matriz de temperaturas e as matrizes de transições das áreas. Neste caso, o pasto ficou como a área mais visitada, já que é a maior área e onde os animais passam quase o dia inteiro, depois ficou a área da água e a área da sombra e o cocho ficaram como as áreas menos visitadas. Já que, a movimentação vai depender das configurações das matrizes de transições das áreas.

Figura 36 – Mapa de Calor da Distribuição 3.

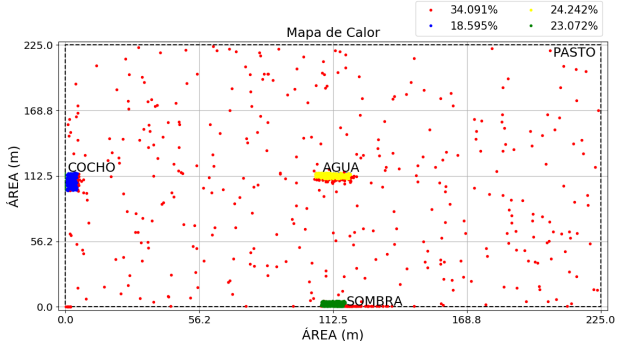
(a) Matriz de Relevo ID: 0



(b) Áreas Visitadas

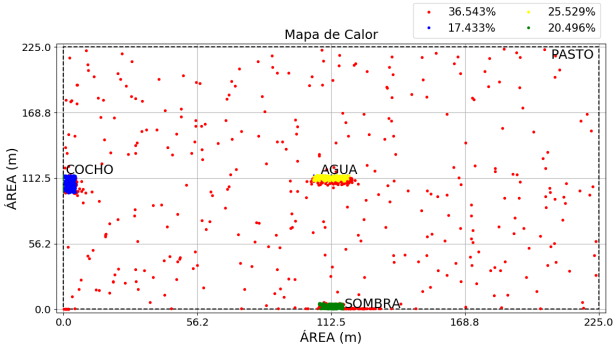
	Pasto	Agua	Cocho	Sombra
07:00 - 08:00	68	23	30	24
08:00 - 09:00	58	21	32	21
09:00 - 10:00	77	29	9	17
10:00 - 11:00	59	34	32	27
11:00 - 12:00	56	20	35	23
12:00 - 13:00	36	33	19	31
13:00 - 14:00	56	33	19	41
14:00 - 15:00	33	48	18	39
15:00 - 16:00	56	33	19	41
16:00 - 17:00	59	21	18	27

(c) Matriz de Relevo ID: 1



(d) Áreas Visitadas

	Pasto	Agua	Cocho	Sombra
07:00 - 08:00	66	30	32	30
08:00 - 09:00	48	39	31	25
09:00 - 10:00	52	26	33	31
10:00 - 11:00	40	22	37	32
11:00 - 12:00	75	44	16	25
12:00 - 13:00	63	28	29	24
13:00 - 14:00	42	32	17	47
14:00 - 15:00	25	58	16	47
15:00 - 16:00	42	32	17	47
16:00 - 17:00	48	37	20	31



(e) Matriz de Relevo ID: 2

	Pasto	Agua	Cocho	Sombra
07:00 - 08:00	63	41	39	17
08:00 - 09:00	56	30	25	21
09:00 - 10:00	49	23	31	28
10:00 - 11:00	53	39	40	12
11:00 - 12:00	44	16	26	34
12:00 - 13:00	59	50	20	23
13:00 - 14:00	47	47	10	43
14:00 - 15:00	35	47	22	37
15:00 - 16:00	47	47	10	43
16:00 - 17:00	50	39	19	36

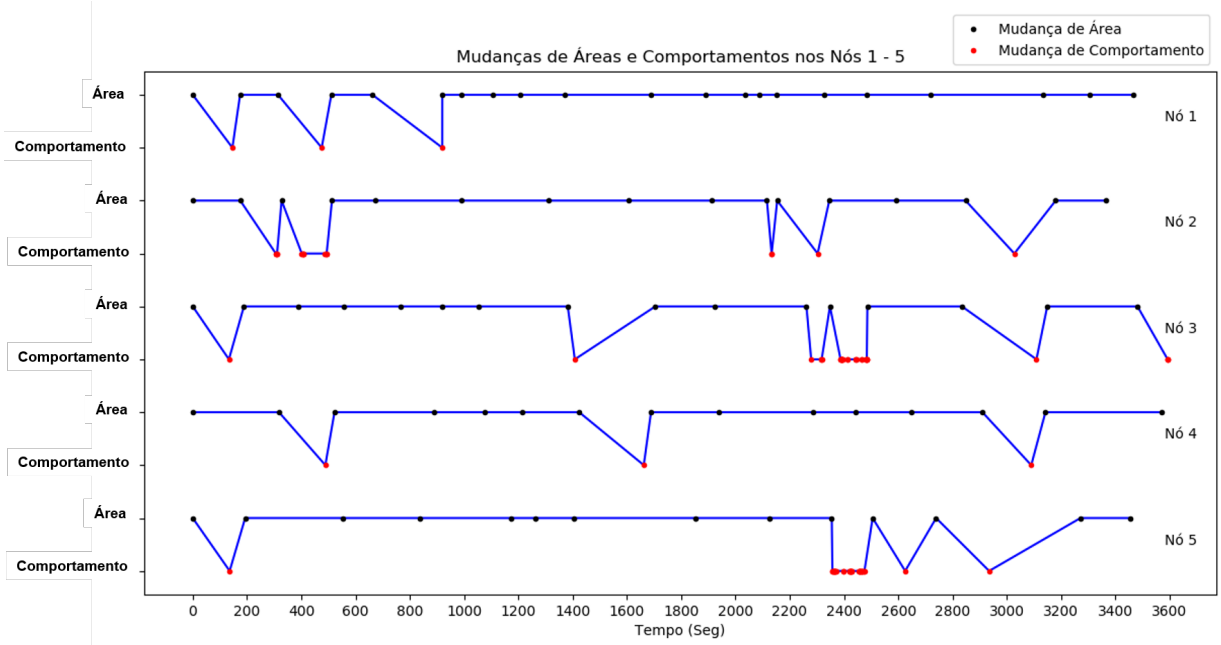
(f) Áreas Visitadas

Fonte: Elaborada pelo autor.

5.2.2.4 *Mudanças de Áreas e Comportamentos*

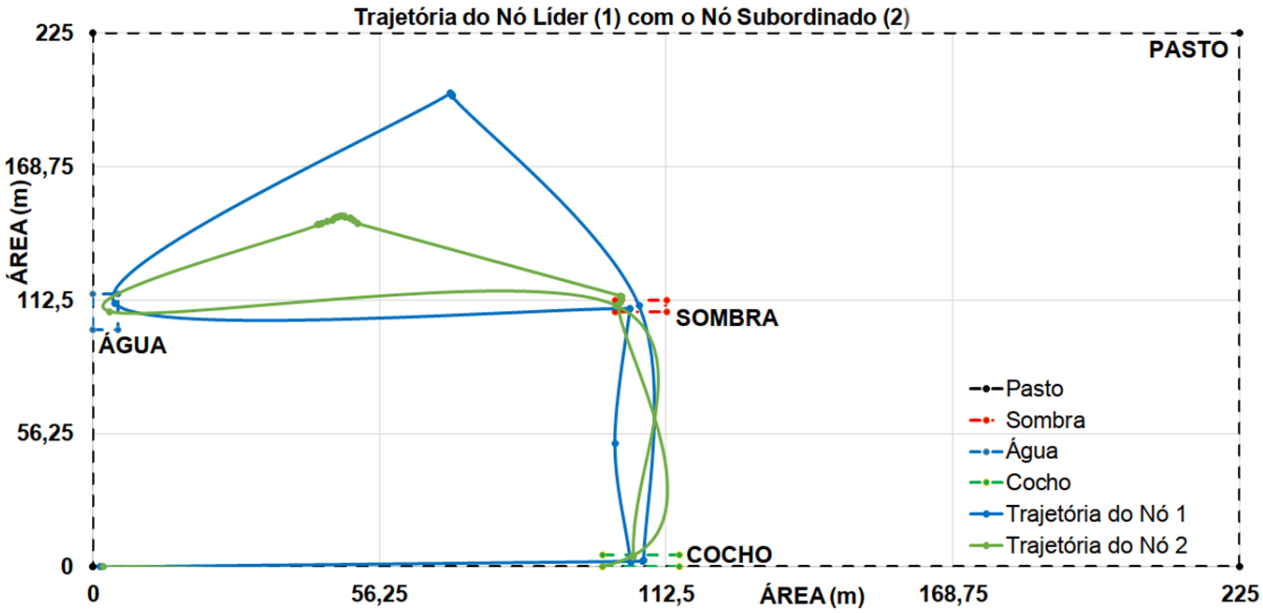
O objetivo é verificar o comportamento em grupo do gado leiteiro, frequentando o mesmo tipo de área e mantendo o comportamento individual. Dos 10 nós móveis utilizados na simulação, para facilitar a visualização gráfica, selecionou-se apenas um grupo dos nós móveis. O nó 1 é o líder do grupo 1 e o nó 2 é subordinado. Para uma melhor visualização se mostra a simulação só nos primeiros 1000 segundos. Nas Figuras 37 e 38 demonstra que as mudanças de áreas são realizadas em grupo e dependem do nó líder e as mudanças dos comportamentos são realizadas individualmente.

Figura 37 – Mudanças de Áreas e Comportamentos do Grupo 1.



Fonte: Elaborada pelo autor.

Figura 38 – Trajetória do Nó Líder (1) com o Nó Subordinado (2).



Fonte: Elaborada pelo autor.

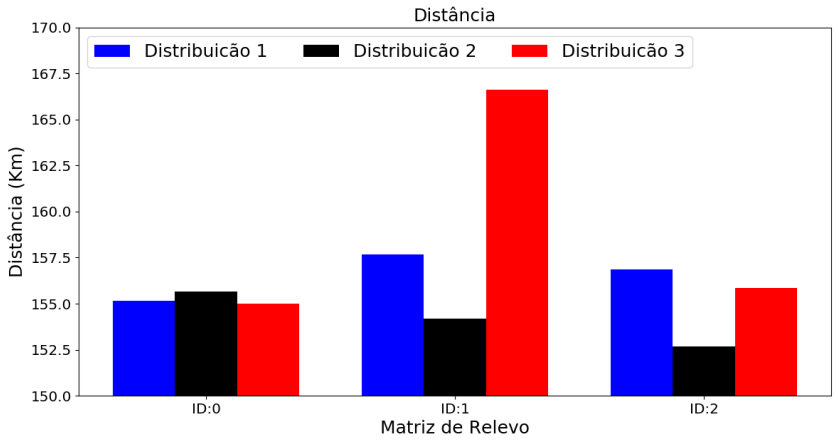
5.2.3 Comparações de Resultados

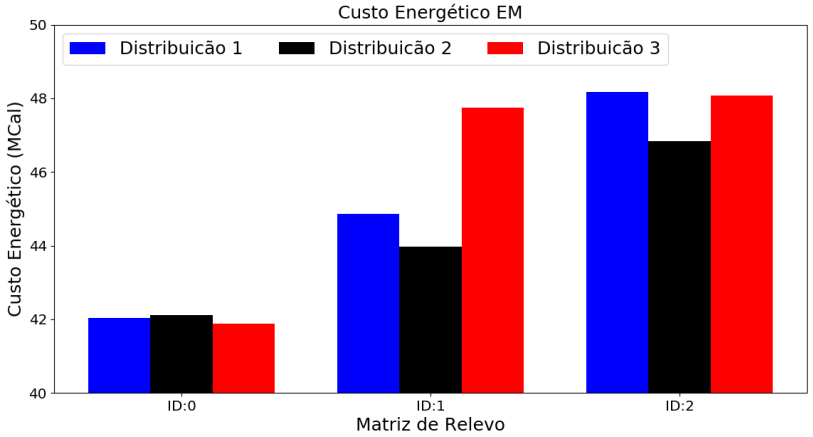
Na Figura 39 é apresentado um resumo dos resultados obtidos nas simulações: as distâncias totais percorridas, os consumos energéticos EM e os consumos energéticos ELL por cada distribuição de área e por cada matriz de relevo.

Os resultados concordam com os apresentados pelo Penati e Corsi [49], embora tenham utilizado outros métodos para calcular o consumo de energia. Y demonstra que quanto mais pesado o animal, maior o custo de energia. Além disso, quanto maior a inclinação do terreno, para a mesma distância percorrida, maior o gasto de energia. Para pequenas inclinações, o custo da energia não atua como um fator limitante.

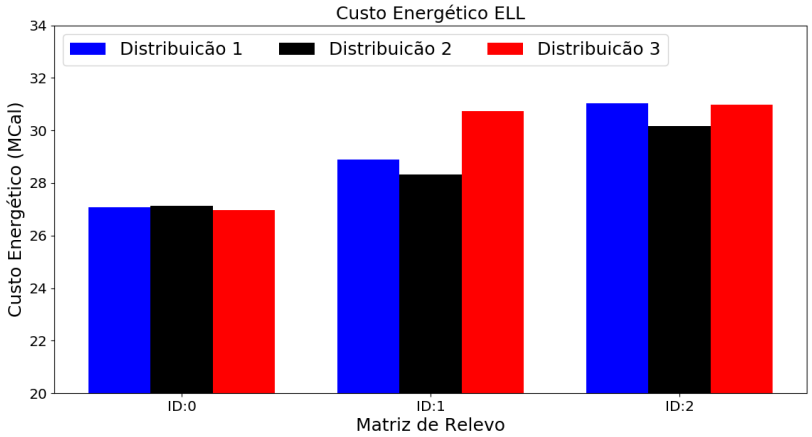
Figura 39 – Comparações dos Resultados.

(a) Distância





(b) Consumo Energético EM



(c) Consumo Energético ELL

Fonte: Elaborada pelo autor.

5.3 CONSUMO DE ENERGIA DA REDE

As simulações foram baseadas nos dois protocolos de roteamento (Bypass e MultipathRings) que já estão implementados no simulador Castalia, mais dois protocolos de roteamento (AODV e LEACH) implementados no simulador para avaliar o desempenho do consumo de energia de cada um deles.

Para obter o consumo de energia se utilizou o módulo gerenciador de recursos do Castalia que é responsável por calcular a quantidade de energia usada em diferentes operações, como transmissão, recepção, etc. A energia é linearmente subtraída com base na potência total gerada e no tempo passado. Os módulos que modelam dispositivos de hardware (ou seja, o rádio e o gerenciador de sensores) enviam mensagens para o gerenciador de recursos para sinalizar quanto de energia eles consomem atualmente. O consumo de energia por módulo de rádio é definido separadamente pelo Castalia. Para definir os principais parâmetros operacionais de um rádio, o Castalia segue um formato específico. O Castalia define dois rádios: CC1000 e CC2420 que definem os rádios reais do mesmo nome pela Texas Instruments. Para avaliar os protocolos, usamos o rádio CC2420.

5.3.1 Parâmetros das Simulações do Consumo de Energia da Rede

Os parâmetros para as simulações são apresentados na Tabela 16. Doze simulações foram realizadas, sendo cada simulação com um protocolo de roteamento e uma quantidade de nós diferentes.

Executou-se uma simulação com um tempo de 3600 segundos (1 hora) com 10, 30 e 50 nós móveis dividido em dois grupos. A área de simulação foi de 50625 m^2. A configuração utilizada do módulo do Canal Sem Fio do simulador foi se nenhum nó estático, e as configurações do gerenciador de comunicação e aplicação foram as padrões do simulador, já que o objetivo principal é verificar o funcionamento dos protocolos de roteamento (BypassRouting, MultipathRouting, LEACH, AODV) para obter o consumo de energia da rede. A estratégia de seleção de área = 1 para utilizar o modelo de Gauss-Markov e o arquivo de configuração foi *ConfigMobility.xml* onde se encontram as informações do gado leiteiro.

Tabela 16 – Parâmetros das Simulações do Consumo de Energia da Rede.

Parâmetros		**Valores**
Tempo da Simulação		3600 Seg
Área da Simulação		225m x 225m = 50625 m^2
Número de Nós		10, 30, 50
Canal Sem Fio	Nós Estáticos	False
Gerenciador de Comunicação	Tipo do Rádio	CC2420
	Potência do Rádio	-5dBm
Gerenciador de Aplicação	Nome	ThroughputTest
Protocolos de Roteamento	Nome	BypassRouting, MultipathRouting, LEACH, AODV
Gerenciador de Mobilidade	Nome	MobilityManager
	Arquivo de Configuração	ConfigMobility.xml
	Estratégia de Seleção de Área	1
	Número de Grupos	2
	Nó Líder	1 → [1 - 5], 1 → [1 - 15], 1 → [1 - 25]
	Nó Líder	6 → [6 - 10], 16 → [16 - 30], 26 → [26 - 50]
	Intervalo de Atualização	100

Fonte: Elaborada pelo autor.

5.3.2 Cenário e Resultados das Simulações

Como abordagem adotada para as simulações, foram utilizados quatro protocolos de roteamento e o módulo de mobilidade híbrido dos modelos de mobilidade para redes *ad hoc*. Nas simulações há uma distribuição de área onde ocorrerá as movimentações dos nós. Para avaliar os desempenhos dos protocolos de roteamento, simulamos com diferentes números de nós (10, 30 e 50 nós) e comparamos o seu desempenho entre eles.

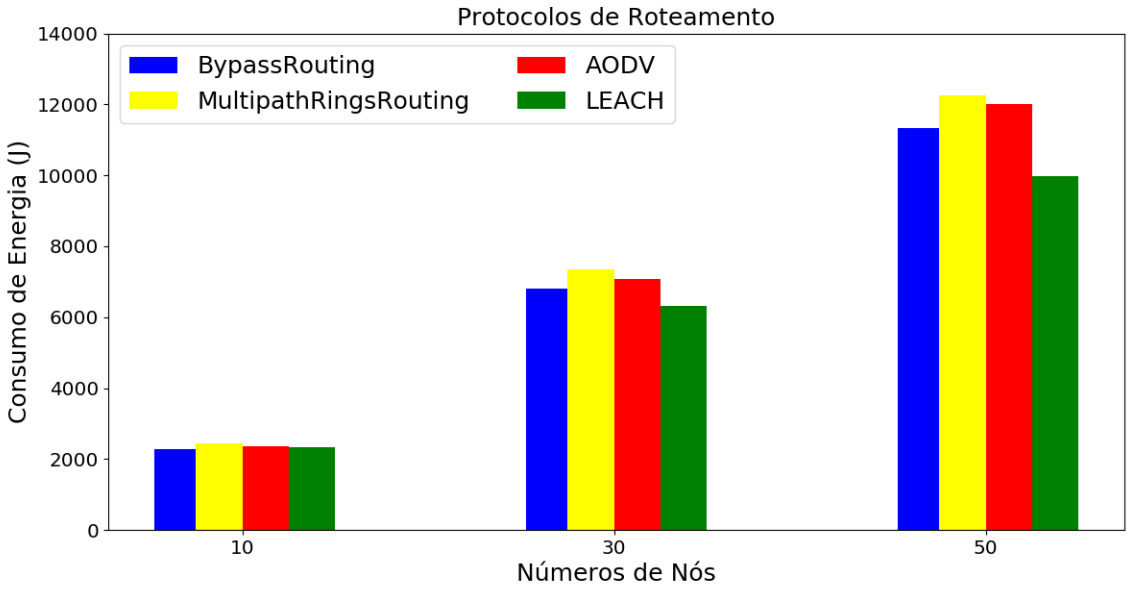
Na Figura 40a, estão apresentados os consumos de energia para os protocolos de roteamento, a partir dos quais podemos concluir que o protocolo de roteamento *Multipath* tem o maior consumo de energia em todos os cenários simulados, para pouca quantidade de nós (<= 10) o protocolo *Bypass* tem menor consumo de energia, embora os resultados

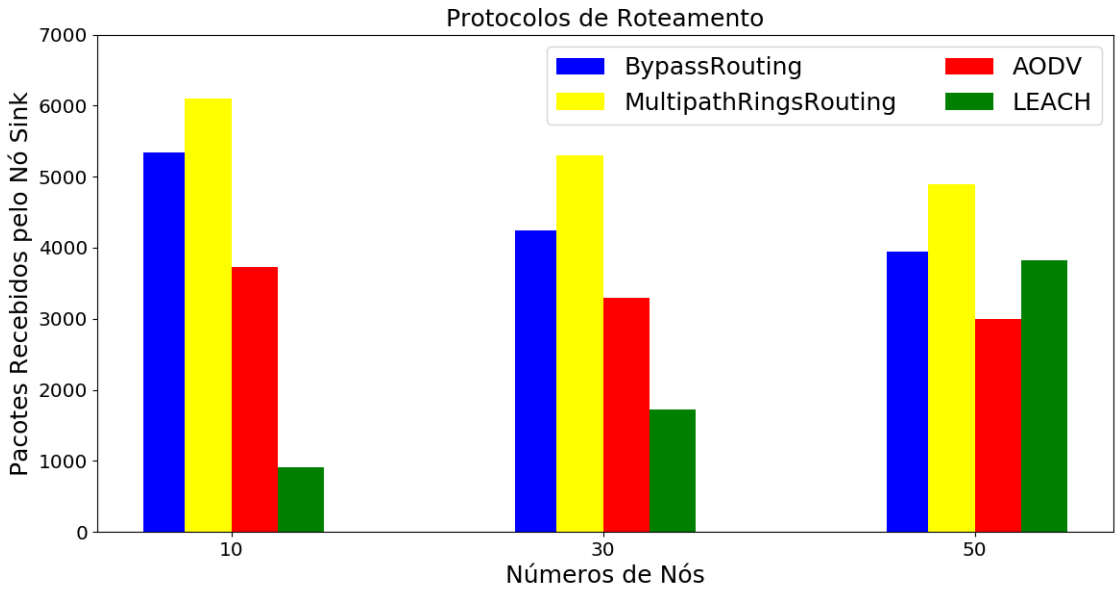
em comparação com o *AODV* e o *LEACH* foram muito parecidos, ou seja, com pouca diferença entre eles. Para mais quantidade de nós (> 10) o protocolo *LEACH* passa a ter o menor consumo de energia, já que muitos nós enviam seus pacotes para o nó cabeça do cluster, o que economiza muita energia. O protocolo de roteamento *AODV* é um protocolo adaptativo a cenários de alta mobilidade, de maneira a evitar desperdício de banda e minimizar o processamento nos nós, que atuam como roteadores na rede, então o consumo de energia aumenta à medida que aumenta o número de nós de forma proporcional.

Além do consumo de energia, também se apresentam os pacotes recebidos pelo nó *Sink* na Figura 41b. O protocolo de roteamento *Multipath* tem a maior entrega (menor perda) de pacotes no nó *Sink* quando é comparado com os outros três protocolos de roteamento. O protocolo de roteamento *LEACH* tem a menor entrega (maior perda) de pacotes no nó *Sink* devido ao algoritmo de seleção dos nós cabeças de cluster que pode resultar em que os nós com pouca energia residual sejam escolhidos como nós principais do cluster.

Figura 40 – Resultados dos Protocolos de Roteamento.

(a) Consumo de Energia da Rede





(b) Pacotes Recebidos pelo nó Sink

Fonte: Elaborada pelo autor.

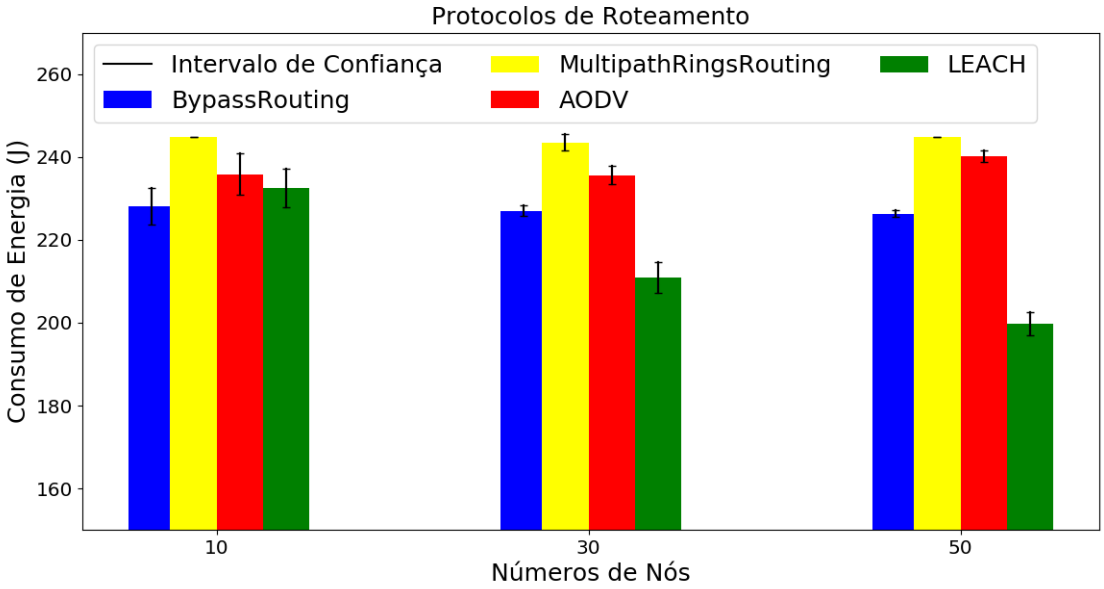
Os valores do consumo médio de energia dos protocolos de roteamento são mostrados na Tabela 17 e na Figura 41 estão apresentados os intervalos de confiança com um nível de confiança de 95%.

Tabela 17 – Consumo Médio de Energia dos Protocolos de Roteamento.

Nós	**Protocolo de Roteamento**	**Consumo de Energia (J)**
10	BypassRouting	228,180 ± 5,839
	MultipathRouting	244,798 ± 0,001
	AODV	235,819 ± 6,701
	LEACH	232,392 ± 6,153
30	BypassRouting	226,949 ± 3,371
	MultipathRouting	243,465 ± 5,074
	AODV	235,612 ± 5,852
	LEACH	210,990 ± 9,744
50	BypassRouting	226,333 ± 2,611
	MultipathRouting	244,799 ± 0,001
	AODV	240,224 ±,4,811
	LEACH	199,752 ± 9,645

Fonte: Elaborada pelo autor.

Figura 41 – Consumo Médio de Energia.



Fonte: Elaborada pelo autor.

Na tabela 18 se apresentam os tempos de processamento dos protocolos de roteamento, onde para cada quantidade de nós o maior tempo foi para o protocolo *Bypass* e o menor tempo foi para o protocolo *Multipath*.

Tabela 18 – Tempo de Processamento dos Protocolos de Roteamento.

Nós	**Protocolo de Roteamento**	**Tempo (Seg)**
10	BypassRouting	7,846
	MultipathRouting	0,727
	AODV	6,009
	LEACH	2,660
30	BypassRouting	38,608
	MultipathRouting	1,561
	AODV	26,574
	LEACH	12,065
50	BypassRouting	88,889
	MultipathRouting	2,302
	AODV	56,812
	LEACH	28,024

Fonte: Elaborada pelo autor.

6 CONCLUSÕES

Em sistemas de pastagem, o problema de testar redes sem fio tende a ser mais significativo, pois existe um gasto de energia maior do que em sistemas confinados, uma vez que os animais percorrem maiores distâncias para beberem água, ou até mesmo para consumirem sal.

Este trabalho apresentou um módulo chamado *Energy Consumption*, que calcula o consumo energético de animais em pastagem através de um novo modelo de mobilidade denominado *Mobility Manager*, que possibilita a simulação mais próxima à realidade dos animais, auxiliando assim projetos científicos que desejam trabalhar dentro deste contexto. Para o desenvolvimento do modelo de mobilidade híbrido baseado nos modelos de mobilidade para redes *ad hoc*, foi utilizado o simulador Castalia 3.2 para as redes de sensores sem fio.

O modelo proposto é totalmente configurável e adaptável para cenários de animais em pastagem, neste caso foi configurado para gado leiteiro. Duas evoluções na solução anterior foram relevantes. Primeiramente, a implementação de uma correlação com o tempo, ou seja, uma correlação de três níveis (Área x Comportamento x Tempo). Essa evolução se mostrou necessária, pois em vários artigos, os pesquisadores informam a grande diferença de comportamento dos animais no período da manhã e da tarde. Em segundo lugar, foi considerado no modelo que as áreas podem ter diferente relevo para não sempre ser uma área plana e assim calcular realmente o consumo energético dos animais que maiormente são mais afetados pelo deslocamento vertical nas áreas.

Com base no modelo de mobilidade proposto, também é possível estudar a adequação das áreas para otimizar também o posicionamento das subáreas, otimizando assim a relação de esforço energético dos animais e os elementos (cocho, água, etc).

Foram simulados 9 cenários em 3 distribuições de áreas e com relevos diferentes, visando verificar o comportamento da variação do consumo energético dos animais em pastagem. A divisões das áreas de relevo no eixo X e Y foram divididas entre 4 partes iguais, mas se podem configurar com um número de divisões maiores e podendo cada eixo ter diferente número de divisões. Em cada distribuição de área observou-se que o maior consumo de energia ocorreu nos cenários com a área de relevo com maior variação de altitude (Área com Subida e Descida), fato que demonstra a adequação do modelo proposto com o conhecimento técnico para gado leiteiro.

Este trabalho também discute sobre redes de sensores sem fio, onde cada nó sensor tem uma carga limitada de bateria, sendo que algumas vezes é impossível ou inviável recarregá-las. Portanto, hoje em dia, essas redes apresentam problemas com o consumo de energia e com os pacotes recebidos e os protocolos de roteamento ajudam a resolvê-los, já que o tempo de vida da rede é diretamente proporcional ao consumo eficiente de energia.

Além disso, as redes precisam de uma alta confiabilidade na entrega dos dados desde o nó fonte até o nó *Sink*.

Foram discutidos os protocolos de roteamento: Bypass, Multipath Rings, AODV e LEACH para as redes de sensores sem fio; avaliando os desempenhos em função do consumo de energia da rede e do número de pacotes recebidos pelo nó *Sink* e foram realizadas comparações entre esses protocolos. Com base nos resultados obtidos, foi possível observar o melhor desempenho do consumo de energia da rede com o protocolo de roteamento *LEACH* e do número de pacotes recebidos pelo nó *Sink* com o protocolo de roteamento *Multipath Rings*.

Como trabalhos futuros, é interessante criar aplicações específicas para simular os nós sensores utilizados em colares para monitoramento e rastreamento de animais em pastagem. Essas aplicações devem ser desenvolvidas no módulo de aplicações do simulador Castalia, mas deve ter grande interligação com o gerenciador de mobilidade para, por exemplo, conseguir simular variações da temperatura corporal quando o animal se desloca do pasto para a sombra. Também, considerar o índice de conforto térmico (Índice de Temperatura e Umidade - ITU), especificamente a umidade, para realizar as mudanças de áreas dos animais. Com respeito aos protocolos de roteamento, é interessante implementar outros protocolos existentes nas RSSFs e avaliar o seu desempenho para definir o uso de cada um dependendo da aplicação a utilizar. Também, usar as diferentes técnicas para melhorar o desempenho dos protocolos de roteamento.

REFERÊNCIAS

[1] IBGE, "Banco de dados," 2017. [Online]. https://sidra.ibge.gov.br [Acessada: 27-Maio-2019].

[2] P. C. d. F. Carvalho, J. K. d. Trindade, J. C. Mezzalira, C. H. E. C. Poli, C. Nabinger, T. C. M. Genro, and H. L. Gonda, "Do bocado ao pastoreio de precisão: compreendendo a interface planta-animal para explorar a multi-funcionalidade das pastagens," *Revista brasileira de zootecnia= Brazilian journal of animal science. Viçosa, MG. Vol. 38, supl. especial (2009), p. 109-122*, 2009.

[3] M. T. P. d. Oliveira, "Análise comportamental de bovinos baseada em trajetórias semânticas aplicada à pecuária de precisão," dissertação de mestrado, Universidade Federal do Mato Grosso do Sul, 2013.

[4] V. Bogorny and F. Braz, "Introdução a trajetórias de objetos móveis: conceitos, armazenamento e análise de dados," *Univille*, 2012.

[5] D. P. Borges, "Modelo de mobilidade de animais em pasto para redes de sensores sem fio utilizando modelo markoviano de percurso aleatório e estados finitos de cadeia de markov," dissertação de mestrado, Universidade Federal de Juiz de Fora, 2018.

[6] C. A. V. Campos and L. F. M. de Moraes, "Modelos markovianos de mobilidade individual para redes móveis ad hoc," *Anais do XXI Simpósio Brasileiro de Redes de Computadores (SBRC2003)*, pp. 135–150, 2003.

[7] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless communications and mobile computing*, vol. 2, no. 5, pp. 483–502, 2002.

[8] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang, "A group mobility model for ad hoc wireless networks," in *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, vol. 53, p. 60, Citeseer, 1999.

[9] A. Boulis, "A simulater for wireless sensor networks and body area network," *NICTA*, 2011.

[10] JetBrains, "Pycharm," 2010. [Online]. Available: https://www.jetbrains.com/pycharm/. [Acessada: 14-Maio-2019].

[11] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.

[12] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Personal communications*, vol. 8, no. 1, pp. 16–28, 2001.

[13] A. A. Loureiro, J. M. S. Nogueira, L. B. Ruiz, R. A. d. F. Mini, E. F. Nakamura, and C. M. S. Figueiredo, "Redes de sensores sem fio," in *Simpósio Brasileiro de Redes de Computadores (SBRC)*, pp. 179–226, sn, 2003.

[14] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan, "Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, pp. 272–287, ACM, 2001.

[15] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 263–270, ACM, 1999.

[16] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[17] I. Howitt and J. A. Gutierrez, "Ieee 802.15. 4 low rate-wireless personal area network coexistence issues," in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, vol. 3, pp. 1481–1486, IEEE, 2003.

[18] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and zigbee standards," *Computer communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[19] G. Mulligan, "The 6lowpan architecture," in *Proceedings of the 4th workshop on Embedded networked sensors*, pp. 78–82, ACM, 2007.

[20] C. Sengul and R. Kravets, "Bypass routing: An on-demand local recovery protocol for ad hoc networks," *Ad Hoc Networks*, vol. 4, no. 3, pp. 380–397, 2006.

[21] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 2, p. 7, 2008.

[22] R. Vidhyapriya and P. Vanathi, "Energy efficient adaptive multipath routing for wireless sensor networks.," *IAENG International Journal of Computer Science*, vol. 34, no. 1, 2007.

[23] A. Pandya and M. Mehta, "Performance evaluation of multipath ring routing protocol for wireless sensor network," *UACEE International Journal of Advances in Computer Networks and its Security*, vol. 2, no. 2, pp. 53–58, 2012.

[24] H. Van Luu and X. Tang, "An enhanced relay scheme for robust data collection through rings overlay in wireless sensor networks," in *2010 IEEE Wireless Communication and Networking Conference*, pp. 1–6, IEEE, 2010.

[25] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," tech. rep., 2003.

[26] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd annual Hawaii international conference on system sciences*, pp. 10–pp, IEEE, 2000.

[27] M. M. Zonoozi and P. Dassanayake, "User mobility modeling and characterization of mobility patterns," *IEEE Journal on selected areas in communications*, vol. 15, no. 7, pp. 1239–1252, 1997.

[28] M. Sánchez and P. Manzoni, "Anejos: a java based simulator for ad hoc networks," *Future generation computer systems*, vol. 17, no. 5, pp. 573–583, 2001.

[29] E. Hyytiä and J. Virtamo, "Random waypoint mobility model in cellular networks," *Wireless Networks*, vol. 13, no. 2, pp. 177–188, 2007.

[30] C.-C. Chiang and M. Gerla, *Wireless network multicasting*. PhD thesis, UCLA, 1998.

[31] K. H. Wang and B. Li, "Group mobility and partition prediction in wireless ad-hoc networks," in *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No. 02CH37333)*, vol. 2, pp. 1017–1021, IEEE, 2002.

[32] I. Minakov, R. Passerone, A. Rizzardi, and S. Sicari, "A comparative study of recent wireless sensor network simulators," *ACM Transactions on Sensor Networks (TOSN)*, vol. 12, no. 3, p. 20, 2016.

[33] I. T. Downard, "Simulating sensor networks in ns-2," tech. rep., NAVAL RESEARCH LAB WASHINGTON DC, 2004.

[34] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[35] A. Varga, "Discrete event simulation system," in *Proc. of the European Simulation Multiconference (ESM'2001)*, pp. 1–7, 2001.

[36] A. Boulis, "Castalia: revealing pitfalls in designing distributed algorithms in wsn," in *Proceedings of the 5th international conference on Embedded networked sensor systems*, pp. 407–408, ACM, 2007.

[37] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. Haneveld, T. E. Parker, O. W. Visser, H. S. Lichte, and S. Valentin, "Simulating wireless and mobile networks in omnet++ the mixim vision," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, p. 71, ICST, 2008.

[38] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. A. Brewer, and D. E. Culler, "The emergence of networking abstractions and techniques in tinyos.," in *NSDI*, vol. 4, pp. 1–1, 2004.

[39] J. Eriksson, F. Osterlind, T. Voigt, N. Finne, S. Raza, N. Tsiftes, and A. Dunkels, "Demo abstract: Accurate power profiling of sensornets with the cooja/mspsim simulator," in *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, pp. 1060–1061, IEEE, 2009.

[40] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *29th annual IEEE international conference on local computer networks*, pp. 455–462, IEEE, 2004.

[41] C. d. F. SOUZA *et al.*, "Instalações para gado de leite," *Apostila didática do Curso de Engenharia Agrícola da Universidade Federal de Viçosa*, 2003.

[42] J. B. da Veiga, A. Homma, A. Camarao, C. Goncalves, C. Ferreira, C. de Freitas, E. d. Cruz Cardoso, G. Azevedo, H. Láu, and J. Tourrand, *Criação de gado leiteiro na Zona Bragantina*. Embrapa, 2006.

[43] J. Da Silva, "Número de animais por hectare," 2013. [Online]. [Acessada: 14-Maio-2019].

[44] B. Assis, "Pecuária leiteira intensiva: Ganhos através de instalações adequadas," 2013. [Online]. [Acessada: 14-Maio-2019].

[45] J. Huzzey, T. DeVries, P. Valois, and M. Von Keyserlingk, "Stocking density and feed barrier design affect the feeding and social behavior of dairy cattle," *Journal of dairy science*, vol. 89, no. 1, pp. 126–133, 2006.

[46] A. Pedroso, "Conforto para vacas leiteiras em pastejo," 2012. [Online]. [Acessada: 14-Maio-2019].

[47] P. D'Hour, A. Hauwuy, J. Coulon, and J. Garel, "Walking and dairy cattle performance," in *Annales de zootechnie*, vol. 43, pp. 369–378, 1994.

[48] J. de CR Ribeiro, J. Brockway, and A. Webster, "A note on the energy cost of walking in cattle," *Animal Science*, vol. 25, no. 1, pp. 107–110, 1977.

[49] M. PENATI and M. CORSI, "Condições técnicas para localização e instalação da exploração leiteira," in *SIMPÓSIO SOBRE PRODUÇÃO ANIMAL*, vol. 10, pp. 7–55, 1998.

[50] O. Di Marco and M. Aello, "Energy cost of cattle walking on the level and on a gradient," *Journal of Range Management*, pp. 9–13, 1998.

[51] S. Radha and S. Shanmugavel, "Mobility models in mobile ad hoc network," *IETE journal of research*, vol. 53, no. 1, pp. 3–12, 2007.

[52] K. H. Kwong, T.-T. Wu, H. G. Goh, K. Sasloglou, B. Stephen, I. Glover, C. Shen, W. Du, C. Michie, and I. Andonovic, "Practical considerations for wireless sensor networks in cattle monitoring applications," *Computers and Electronics in Agriculture*, vol. 81, pp. 33–44, 2012.

[53] L. d. Jesus, "Identificação do comportamento bovino por meio do monitoramento animal," dissertação de mestrado, Universidade Federal de Mato Grosso do Sul, 2014.

[54] L. F. D. Lomba, "Identificação do comportamento bovino a partir dos dados de movimentação e do posicionamento do animal," dissertação de mestrado, Universidade Federal de Mato Grosso do Sul, 2015.

[55] J. F. R. Nacer, "Aspectos de roteamento em redes de sensores sem fio para monitoramento bovino," dissertação de mestrado, Universidade Federal de Mato Grosso do Sul, 2017.

[56] M. Radi, B. Dezfouli, K. A. Bakar, and M. Lee, "Multipath routing in wireless sensor networks: survey and research challenges," *Sensors*, vol. 12, no. 1, pp. 650–685, 2012.

[57] G. M. Huang, W. J. Tao, P. S. Liu, and S. Y. Liu, "Multipath ring routing in wireless sensor networks," in *Applied Mechanics and Materials*, vol. 347, pp. 701–705, Trans Tech Publ, 2013.

[58] D. J. Barot and I. Rajani, "Enhancing the performance of multipath ring routing protocol with heterogeneous system in wireless sensor network," *International Journal for Scientific Research Development*, vol. 3, pp. 1356–1359, 2015.

[59] A. R. C. ARC, "The nutrient requirements of farm livestock," no. 3, 1988.

[60] Y. Guo, P. Corke, G. Poulton, T. Wark, G. Bishop-Hurley, and D. Swain, "Animal behaviour understanding using wireless sensor networks," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pp. 607–614, IEEE, 2006.

APÊNDICE A – Métodos das Classes do Módulo de Mobilidade

Tabela 19 – Métodos da Classe *Mobility Manager*.

Métodos	Descrição
void initialize();	Inicialização da movimentação
bool readConfigurationFile();	Realiza a leitura do arquivo de configuração .xml
bool setInitialParameters();	Inicialização dos parâmetros do arquivo de configuração omnetpp.ini
int PositionHoursVector();	Retorna a Posição do Vector das Horas
void generateInitialPosition();	Calcula uma posição aleatória simulando a entrada ao pasto
void createFileTracePosition();	Cria o arquivo .Log das posições
void createFileTraceDistance();	Cria o arquivo .Log das distâncias
void createFileTraceTime();	Cria o arquivo .Log dos tempos
void writeNodeLocationInTraceFile();	Escreve no arquivo .Log das posições
void writeNodeDistanceInTraceFile();	Escreve no arquivo .Log das distâncias
void writeNodeTimeInTraceFile(int);	Escreve no arquivo .Log dos tempos
void verifyNextMove();	Verifica se irá ocorrer alguma mudança de área ou comportamentos
void selectAreaType(int);	Verifica se ocorreu mudança de área
string nameArea(int matrixID);	Recupera o nome da área
void notifyChangeArea(int newAreaTypeID);	Notifica a mudança de área
void onChangeArea(int newAreaTypeID);	Notifica se houver ou não mudança de área
void selectBehavior(int);	Verifica se ocorreu mudança de comportamentos
string nameBehavior(int matrixID);	Recupera o nome do comportamento
void newTargetPositionRandomNormal();	Calcula a próxima posição utilizando uma distribuição randômica
void newTargetPositionNewArea();	Calcula a próxima posição quando houver mudança de área
void newTargetPositionGaussMarkov();	Calcula a próxima posição utilizando o modelo Gauss-Markov
void handleMessage(cMessage * msg);	Recebe as mensagens do módulo superior
bool verifyBoundary();	Verifica se o nó está próximo dos limites máximos

Fonte: Elaborada pelo autor.

Tabela 20 – Métodos da Classe *Transition*.

Métodos	Descrição
Transition(int);	Define o número de comportamentos
bool addBehaviors(int, string, double, double);	Cria um novo comportamento e a adiciona à lista
bool addAreaType(int,string,Coordinates,Coordinates,int,int,int);	Cria uma nova área e a adiciona à lista
int addTemperatureMatrix(int, double **, bool, int, int);	Cria uma nova matriz de temperatura
int addReliefMatrix(int, double **, bool, int, int);	Cria uma nova matriz de relevo
int addAreaMatrix(int, double **, bool, int);	Cria uma nova matriz de transições das áreas
int addBehaviorsMatrix(int, double **, bool, int);	Cria uma nova matriz de transições dos comportamentos
double getSpeedByBehavior(int);	Recupera a velocidade pelo comportamento
Coordinates getMaxArea(int);	Retorna o tamanho da área máxima
Coordinates getMinArea(int);	Retorna o tamanho da área mínima
int getMatrixBehaviorID(int);	Recupera o ID da matriz de comportamento da área
int getRowAreaRelief(int);	Recupera o número da fila da área de relevo
int getCellAreaRelief(int);	Recupera o número da coluna da área de relevo
double getSpeedWalkingBehavior(double, double);	Retorna a velocidade do comportamento ANDANDO
double getMinimumSpeedBehavior(int);	Retorna a velocidade mínima comportamento
double getMaximumSpeedBehavior(int);	Retorna a velocidade máxima do comportamento
Coordinates getRandomPosition(int, std::ostream&);	Retorna uma posição aleatória dentro de uma área
int getAreaMatrixID(int, int);	Recupera o ID da matriz de transições das áreas
double **getReliefMatrix(int);	Recupera a matriz de relevo
double **getAreaMatrix(int);	Recupera a matriz de transições das áreas
double **getBehaviorMatrix(int);	Recupera a matriz de transições dos comportamentos
int isMarkovian(double **, int);	Verifica se uma matriz é markoviana

Fonte: Elaborada pelo autor.

Tabela 21 – Métodos da Classe *Behavior*.

Métodos	**Descrição**
bool setBehaviorId(int);	Setear o ID do comportamento
bool setBehaviorSpeed(double, double);	Setear as velocidades do comportamento
bool setBehaviorName(char *);	Setear o nome dado para o comportamento
int getBehaviorID();	Retorna o ID exclusivo do comportamento
char *getBehaviorName();	Retorna o nome do comportamento
double getMaxSpeed();	Retorna a velocidade máxima do comportamento
double getMinSpeed();	Retorna a velocidade mínima do comportamento

Fonte: Elaborada pelo autor.

APÊNDICE B – Métodos do Algoritmo do Módulo de Energia

Tabela 22 – Métodos do Algoritmo *Energy Consumption*.

Métodos	**Descrição**
def ReadingFileXML(File_xml):	Faz a leitura do arquivo XML para carregar toda a configuração dos animais.
def ReadingFileLog(File_Log):	Faz a leitura de todos os arquivos *.Log* gerado pelo Castalia.
def LineEquation(x1,y1,x2,y2):	Calcula todas as equações das retas entre dois pontos.
def Trajectory(m,b,x1,y1,x2,y2,Axis_X,Axis_Y):	Reconstrói toda a trajetória.
def SearchReliefArea(x,y,Axis_X,Axis_Y,flag):	Procura o valor da área de relevo de todo o percorrido que forma a trajetória.
def DistanceCalculation(X,Y):	Calcula as distâncias.
def EnergyConsumptionCalculation_EM(Weight,D_H,D_V,C_H,C_V):	Calcula o consumo energético EM dos animais.
def EnergyConsumptionCalculation_ELL(EnergyConsumption_EM):	Calcula o consumo energético ELL dos animais.
def SalveFile (File,Vector):	Armazena os vetores com os consumos energéticos EM e ELL dos animais.

Fonte: Elaborada pelo autor.