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

Daniel Prata Leite Borges

PastureManagerMobility: Modelo de Mobilidade de Animais em Pastagem para Redes de Sensores Sem Fio Utilizando Modelo Markoviano de Percurso Aleatório e Estados Finitos de Cadeias de Markov

Juiz de Fora

2018

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

Daniel Prata Leite Borges

PastureManagerMobility: Modelo de Mobilidade de Animais em Pastagem para Redes de Sensores Sem Fio Utilizando Modelo Markoviano de Percurso Aleatório e Estados Finitos de Cadeias de Markov

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

Orientador: Eduardo Barrere

Juiz de Fora

2018

Daniel Prata Leite Borges

PastureManagerMobility: Modelo de Mobilidade de Animais em Pastagem para Redes de Sensores Sem Fio Utilizando Modelo Markoviano de Percurso Aleatório e Estados Finitos de Cadeias de Markov

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

Aprovada em 17 de Dezembro de 2018.

BANCA EXAMINADORA

Prof. Dr. Eduardo Barrere - Orientador
Universidade Federal de Juiz de Fora

Prof. Dr. Wagner Antônio Arbex
Universidade Federal de Juiz de Fora

Prof. Dr. Ricardo Guimarães Andrade
Embrapa Gado de Leite

Aos meus pais e minha esposa

Sempre presentes

AGRADECIMENTOS

Primeiramente agradeço a Deus por me dar força, coragem e determinação em todos os momentos.

Aos meus pais, Geraldo e Valéria, pelo caráter e valores transmitidos e pelo esforço incondicional para proporcionar a mim e aos meus irmãos, Carina e Felipe, o melhor que poderiam prover. Aos meus irmãos e meus amigos por estarem sempre ao meu lado, incentivando e acreditando em meu potencial em momentos que até mesmo eu não acreditava.

A minha esposa Patrícia por todo amor, cumplicidade e companheirismo, sempre me incentivando e motivando durante esse percurso. Pelas noites em claro, assistindo minhas apresentações sem sequer entender sobre o que eu estava falando. Sem você, nada disso seria possível.

Ao meu orientador Prof. Dr. Eduardo Barrére pelo apoio, paciência, dedicação e incentivo ao decorrer deste mestrado.

Aos membros da Banca Examinadora pela avaliação deste trabalho.

E a todos que contribuíram, direta ou indiretamente, na realização deste trabalho.

"Talvez não tenha conseguido
fazer o melhor, mas lutei para
que o melhor fosse feito. Não sou
o que deveria ser, mas Graças a
Deus, não sou o que era antes"
Marthin Luther King

RESUMO

A utilização de colares contendo sensores para o monitoramento e rastreamento de animais em pasto, principalmente para rebanhos bovinos, vem ganhando adeptos no Brasil e no mundo. O monitoramento dos animais se mostra muito importante no que tange otimização dos ganhos, cuidados com a saúde e qualidade dos produtos obtidos em toda cadeia de produção. Todavia, o monitoramento em tempo real, possibilitado pela transmissão das informações coletadas pelos colares com estações de comunicação, se mostra ainda mais eficaz, pois possibilita a tomada de decisões mais rápidas, antes que possíveis problemas aconteçam ou oportunidades se percam. Porém, testes e implantação desses equipamentos no mundo real são extremamente custosos e complexos. Devido a isso, a utilização de simuladores de Redes de Sensores Sem Fio se torna essencial para o desenvolvimento desse tipo de tecnologia. O presente trabalho descreve e implementa um modelo de mobilidade configurável que se aproxima da mobilidade realizada por animais que se organizam em rebanhos, utilizando estados finitos em cadeias de Markov. Este modelo foi implementado no simulador de Redes de Sensores Sem Fio denominado Castalia, devido sua fidedignidade em relação ao consumo de energia dos nós sensores, um dos maiores problemas enfrentados pelas RSSF, e sua facilidade de implementação de protocolos de acesso ao meio e de roteamento, se mostrando assim um ótimo ambiente para realização dos testes desejados.

Palavras-chave: Cadeias de Markov. Redes de Sensores Sem Fio. Simulação. Monitoramento. Rastreamento. Rebanhos. Mobilidade.

ABSTRACT

The use of collars containing sensors for the monitoring and tracking of grazing animals, especially for cattle herds, has been gaining support in Brazil and worldwide. The monitoring of the animals is very important in terms of optimization of the gains, health care and quality of products obtained throughout the production chain. However, the real-time monitoring, made possible by the transmission of the information collected by the collars with communication stations, is even more effective, since it allows for faster decision making, before possible problems happen or opportunities are lost. However, testing and deploying these devices in the real world are extremely costly and complex. Because of this, the use of wireless sensor network simulators becomes essential for the development of this type of technology. The present work describes and implements a configurable mobility model that approximates the mobility performed by animals that organize themselves in herds, using finite states in Markov chains. This model was implemented in the Wireless Sensor Networks simulator called Castalia, due to its reliability with the measurement of spent energy, one of the major problems faced by the WSN, and its ease of implementation of medium access and routing protocols, showing thus a great environment for the realization of the desired tests.

Keywords: Markov Chains. Wireless Sensor Network. Simulation. Monitoring. Tracking. Cattle. Mobility.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BAN Body Area Network

CPU Central Processing Unit

HMM Hidden Markov Model

IBGE Instituto Brasileiro de Geografia e Estatística

MAC Media Access Control

MANET Mobile Ad hoc Network

MPA Modelo Markoviano de Percurso Aleatório

NS Network Simulator

RGMM Random Gauss-Markov Mobility

RPGM Reference Point Group Mobility

RSSF Redes de Sensores Sem Fio

RWMM Random Walk Mobility Model

RWPM Random Waypoint Mobility Model

SDN Software-defined Networking

UFJF Universidade Federal de Juiz de Fora

UFMS Universidade Federal do Mato Grosso do SUl

WSN Wireless Sensor Network

SUMÁRIO

1 INTRODUÇÃO

Segundo dados do IBGE (2018), o Brasil possui o segundo maior rebanho bovino do mundo, ficando atrás apenas da Índia, com um efetivo de 214,9 milhões de cabeças. A produção de leite em 2017, ainda segundo o IBGE, totalizou 33,5 bilhões de litros, sendo o estado de Minas Gerais o maior produtor com 26,6% da produção nacional. A representatividade econômica da pecuária, tanto dos rebanhos de corte quanto de leite, sempre foi enorme em nosso país e a tendência é que continue expandindo. Neste contexto, a pecuária de precisão diz respeito a inovações tecnológicas que monitoram o animal no seu ambiente, sendo uma forma eficiente de gerenciar os sistemas de produção animal (CARVALHO *et al.*, 2009). Consiste, também, na medição de diferentes parâmetros dos animais (como por exemplo, o tempo de permanência de um animal em uma região do pasto), 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 utilização de tecnologia nos rebanhos pode ocasionar enormes ganhos na qualidade dos produtos, na saúde dos animais e no crescimento econômico gerado pela agropecuária. Um exemplo desta utilização é o monitoramento do gado através de colares ou *tags* contendo 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 outros nós ou para uma estação base. O uso de sensores para monitoramento existe há décadas, sendo usados em aplicações de climatologia, biologia, militares e indústria.

No que tange ao monitoramento e rastreamento animal, essa tecnologia traz uma gama de benefícios, como: detecção antecipada de doenças ou qualquer anomalia no animal, seleção de pastagens com quantidades maiores de vegetação, segurança dos animais e aumento da produtividade e detecção de cio em tempo real.

Todavia, para que os dados coletados pelos sensores sejam analisados sem que haja a necessidade da coleta manual desses, faz-se necessário a implementação de uma rede sem fio interligando os nós (animais) dos sensores. Essas redes são conhecidas como RSSF (Rede de Sensores Sem Fio) e seu funcionamento depende do desenvolvimento de protocolos que supram suas necessidades, como, por exemplo, protocolos otimizados para

economia do consumo de energia dos nós.

A avaliação de desempenho de protocolos é uma fase importante e normalmente realizada usando simulações ou experimentos reais. Embora experimentos reais forneçam resultados mais confiáveis e precisos, a simulação pode ser uma melhor alternativa, pois é menos dispendiosa, é extensível a redes maiores (escalável) e pode ser feita em um curto espaço de tempo. Outro ponto a se destacar é que através da simulação podemos observar melhor o comportamento do protocolo e detectar potenciais erros no processo de projeto e implementação, antes dos custos da implantação.

Existem vários simuladores especializados em RSSF ou Redes *Ad-hoc* como: NS3 (HENDERSON *et al.*, 2008), Castalia (BOULIS, 2007) e o MiXiM (KÖPKE *et al.*, 2008), atualmente incorporado ao Inet (VARGA, 2012). Estes simuladores focam principalmente nas camadas Física, de Acesso ao Meio e de Aplicação, e disponibilizam poucos modelos de mobilidade dos nós. O Castalia (BOULIS, 2007), por exemplo, em sua versão 3.2 possui apenas o caso de não mobilidade e o de mobilidade linear. E, conforme sugere (NABI *et al.*, 2011), um bom modelo de mobilidade é um pré-requisito essencial para avaliação de desempenho de protocolos para RSSF contendo nós móveis. Sendo assim, para que as simulações alcancem uma maior acurácia e precisão, modelos de mobilidades mais próximos de padrões encontrados no mundo real se tornam necessários.

Modelos são bastante comuns em outras áreas da Ciência de Computação, como na Computação Gráfica, e, com certeza, podem trazer ganhos para os estudos de Redes de Computadores. Então, com o intuito de proporcionar uma avaliação de desempenho de RSSF mais próxima da realidade, este trabalho propõe um modelo de mobilidade configurável que se aproxima dos padrões de mobilidade de animais em pastagem ou ao ar livre.

Para implementar e testar o modelo, escolheu-se a plataforma de Simulação Castalia. Dentre vários, um dos principais motivos pela escolha desta plataforma foi a forma realística com que a plataforma supre as necessidades de simulação do canal sem-fio. O canal sem-fio é notoriamente um meio difícil de ser modelado, especialmente quando estamos levando em conta: nós móveis, mudanças de ambiente e comunicação entre bandas diferentes.

Outro motivo importante é que a comunicação é o aspecto mais cuidadosamente modelado pelo Simulador Castalia, desde o canal sem fio ao comportamento do rádio, além de

possuir várias implementações de protocolos de acesso ao meio, capturando assim a essência e muitos detalhes dos homólogos reais. Todavia, o módulo de simulação de movimento do Simulador Castalia nos disponibiliza inicialmente apenas dois tipos: sem movimento e movimento em linha reta. Apesar de ser um módulo totalmente programável, a inexistência de um modelo padrão de movimentação dos nós que proporcione os desafios de uma RSSF móveis no mundo real não nos permite realizar comparações mais precisas entre protocolos.

Após a implementação, como prova de conceito, uma instância foi criada e configurada baseada na mobilidade de bovinos a pasto. Escolhemos especificamente esta abordagem pelo relacionamento institucional existente entre a UFJF e a Embrapa Gado de Leite, por se tratar de um nicho muito importante para economia de nosso país, pelo grande número de trabalhos científicos que abordam essa temática e pelo fato desse cenário contemplar a maioria dos problemas e complexidade que uma RSSF móvel possa ter e que devem ser solucionados ou otimizados por protocolos (de acesso a meio ou roteamento). Ou seja, trata-se de um movimento com inúmeras nuances comportamentais que consegue trazer os desafios reais que uma RSSF móvel possui.

1.1 OBJETIVOS

Este trabalho apresenta um modelo de mobilidade de animais a pasto para ser utilizado em simuladores RSSF com nós móveis e pode-se listar como seus principais objetivos:

- Elaborar um modelo de mobilidade para animais a pasto, configurável e extensível que possibilite a criação de cenários e comportamentos que se aproximem do existente na realidade;
- Possibilitar uma maior acurácia nas simulações e uma melhor avaliação de desempenho dos protocolos;
- Reprodutibilidade. Possibilitar que execuções diferentes utilizem o mesmo *trace* de mobilidade, tornando assim os resultados comparáveis e possibilitando que simulações possam ser repedtidas (reproduzidas); Estes arquivos *trace* podem ser disponibilizados e proporcionar um padrão que possa ser adotado para outras pesquisas científicas;

- Disponibilizar uma implmentação do modelo para a plataforma de simulação Castalia e uma instância do modelo configurada para bovinos a pasto que possa ser utilizado por outras pesquisas sobre este contexto;

1.2 ESTRUTURA DO TRABALHO

Além do capítulo introdutório, essa dissertação está dividida da seguinte maneira. O Capítulo 2 apresenta a revisão da literatura, citando os principais trabalhos relacionados. O Capítulo 3 descreve os principais conceitos relacionados à proposta e às tecnologias envolvidas. O Capítulo 4 apresenta o modelo proposto e o Capítulo 5 descreve os resultados obtidos através da implementação deste modelo. E por fim, o Capítulo 6 conclui a dissertação, apresentando as considerações finais e trabalhos futuros.

2 REVISÃO DA LITERATURA

Durante a revisão da literatura científica, foram encontrados inúmeros trabalhos relacionados aos principais assuntos abordados por este trabalho. Os principais, ou seja, aqueles que trouxeram maiores contribuições para o desenvolvimento do trabalho, estão citados abaixo, contendo uma breve descrição.

A tese de doutorado de Wietrzyk (2008), descreve um sistema de emulação de movimento para gado em confinamento, por meio de dados de GPS para calcular a variação da velocidade dos bovinos e considerou que os animais poderiam assumir apenas 3 diferentes estados comportamentais (descansando no cocho, comendo/bebendo e sendo ordenhado) conforme demonstra a figura 2.1. Apesar de se tratar de um ótimo estudo, percebe-se que o contexto descrito é mais simples do que um cenário de animais em pasto, que impõe maiores desafios e complexidade para implementação de uma RSSF. Inclusive, na seção de trabalhos futuros, o autor cita o desejo de criar um padrão de movimentação de bovinos a pasto por se tratar de um cenário mais complexo e desafiador. Outra contribuição interessante desse trabalho foram as deduções realizadas sobre a velocidade de caminhada dos animais. Com base nos dados GPS, percebeu-se que a velocidade de marcha momentânea preferida difere significativamente entre as vacas e também que a velocidade preferencial de caminhada, ou seja, comportamento ANDANDO tem uma distribuição gama. Além disso, toda velocidade acima de 1,5 m/s foi considerada falha de medição do GPS.

Todavia, ao ler trabalhos de outros autores que citam o trabalho descrito acima, parece que o algoritmo de transmissão apresentado não é muito eficiente, como sugere Huang *et al.* (2009). O autor chega a chamar o método (algoritmo) de transmissão utilizado em Wietrzyk (2008) de rude, pois transmite todas as medições em um certo período de tempo fixo. O autor ainda alega que conseguiu reduzir o *overhead* de comunicação em até 20% com otimizações no algoritmo proposto por ele.

Kwong *et al.* (2012) cita a mobilidade como um dos principais desafios ao elaborar uma plataforma de RSSF para o monitoramento de gado, uma vez que estão sujeitos a mudanças frequentes de localização. A topologia de rede e os caminhos de roteamento devem, portanto, ser dinâmicos, capazes de responder ao movimento frequente dos animais, otimizando a entrega de pacotes. Após o monitoramento de 13 vacas durante 2

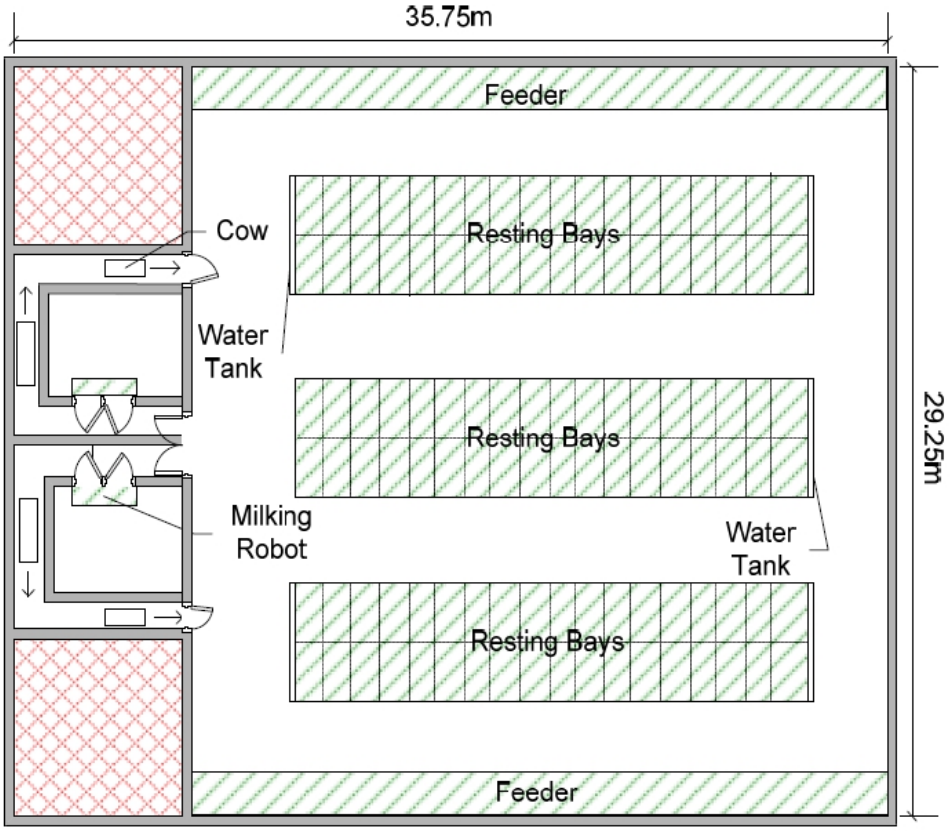

Figura 2.1: Layout da divisão de laticínios (WIETRZYK, 2008) .

dias, o autor concluiu que os rebanhos não se distribuem uniformemente e nem sempre se movem como um único coletivo ou grupo. Em vez disso, o rebanho pode se dividir em subgrupos independentes, cada um com sua própria dinâmica. Esse comportamento leva à necessidade de a topologia da rede mudar dinamicamente. O autor conclui afirmando que é fundamental o conhecimento da mobilidade dos animais para possibilitar um funcionamento mais eficiente de sua plataforma de monitoramento.

O artigo apresentado por Nkwari *et al.* (2015), sugere que algumas informações sobre o comportamento e saúde do gado devem ser interpretadas e analisadas em tempo real para serem realmente úteis e efetivas. Sugere também que uma maneira de aumentar o alcance das redes *Ad Hoc* e conseguir ter conectividade em tempo real é utilizando uma rede hibrida contendo dois transmissores, um *Wi-Fi* e um ZigBee. A combinação dos transmissores IEEE 802.11 e IEEE 802.15.4 em um nó móvel agrega a vantagem de obter rede *Ad Hoc* que possa trabalhar distante da estação de base.

A UFMS, em parceria com a Embrapa Gado de Corte, possui uma linha de pesquisa bastante interessante e que foi de extrema valia para este trabalho. Esta linha de pesquisa rendeu a publicação de alguns artigos e na defesa de quatro teses de metrado. O primeiro

trabalho publicado foi de Oliveira (2013), que elaborou um sistema computacional com a capacidade de identificar padrões comportamentais e informações relevantes através das trajetórias dos animais produzidas durante o pastejo. Para isso, utilizou metodologias de trajetória semântica. O intuito era auxiliar os pecuaristas e pesquisadores na tomada de decisões.

Então, evoluindo o trabalho de Oliveira (2013), Jesus e Pires (2014) criaram colares contendo nó sensor com capacidade para coletar dados por meio de uma RSSF para monitorar os bovinos e inferir por meio de um sistema o comportamento animal. Os nós possuiam um sensor de GPS preso a um colar que é colocado no bovino que circula em uma pastagem totalmente georreferenciada. Os dados gerados pelo sensor GPS foram armazenados em um cartão memória e após coletados eram disponibilizados em um sistema de informação.

Dando continuidade a essa linha de pesquisa, Lomba (2015) criou uma plataforma de identificação automática do comportamento bovino, utilizando os dados de movimentação e posicionamento do animal e algoritmos de classificação supervisionada. E mais recentemente, Nacer (2017) acrescentou aos trabalhos anteriores a possibilidade de transmissão dos dados para estações base e otimizou o protocolo de roteamento utilizado na RSSF. Estes trabalhos mostram a importância das Universidades possuírem linhas de pesquisas em parcerias com outras instituições, no intuito de resolver problemas reais.

Outra linha de pesquisa que serviu como alicerce para este trabalho é a realizada pela Universidade Tecnológica de Eindhoven, Holanda. O trabalho Nabi *et al.* (2011), um dos principais dessa linha de pesquisa, descreve um modelo abrangente de mobilidade configurável denominado MoBAN *(Mobility Model for BANs)* para avaliar a comunicação intra e extra-WBAN. Este modelo implementa diferentes posturas, bem como mobilidade de nó individual dentro de uma determinada postura. Além disso, o modelo pode ser adaptado para uma ampla gama de aplicações para WBANs. Os autores ainda deixam claro como os modelos de mobilidade têm um grande impacto na precisão das simulações para redes ad hoc e RSSF com nós móveis. Sugerem também que os modelos de mobilidade imitam o comportamento dos nós móveis na realidade, caracterizando padrões estocásticos de movimento de nós, e que o modelo de mobilidade certo depende fortemente do cenário de aplicação. Implementaram o modelo sobre a plataforma MiXiM (KÖPKE *et al.*, 2008), atualmente incorporado ao Inet (VARGA, 2012). Na figura 2.2 pode-se ver, de uma forma

abstrata, a modularização do MoBAN.

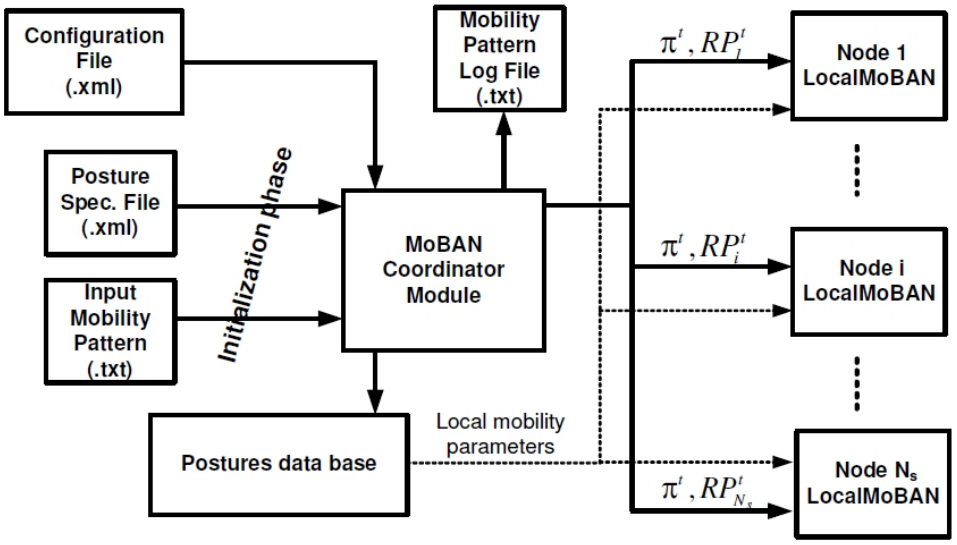

Figura 2.2: Diagrama de blocos da implementação do OMNeT ++ do modelo de mobilidade do MoBAN (NABI *et al.*, 2011).

O trabalho de Campos e Moraes (2003) propõe modelos de mobilidade individual para redes móveis *ad hoc* com objetivo de representar uma maior liberdade de movimentação dos dispositivos móveis. Através dos modelos propostos, permitem-se movimentos na mesma direção, em direções adjacentes, acelerações e intervalos de pausa no movimento, além de evitar mudanças bruscas de direção e paradas abruptas. Dessa forma, atingiu uma maior aproximação do movimento real dos usuários em um ambiente urbano e em rodovias. No artigo, os modelos são descritos analiticamente através de cadeias de Markov e são apresentadas comparações, feitas através de simulação, entre os modelos existentes e o Modelo Markoviano de Percurso Aleatório.

3 FUNDAMENTAÇÃO TEÓRICA

Para desenvolver este trabalho, foi realizada uma extensa revisão na literatura científica e foram revisados os principais conceitos envolvidos para identificar as possibilidades e limitações das tecnologias disponíveis. A Seção 3.1 descreve os principais fundamentos relacionados às redes de sensores. Na Seção 3.2, é apresentada os principais modelos de mobilidade. Na Seção 3.3, são apresentadas as características para o monitoramento do comportamento animal e na subseção 3.3.1, especificando um pouco mais, é apresentado um resumo sobre o estado da arte referente ao comportamento bovino, grupo animal escolhido para servir como parâmetro para o desenvolvimento do trabalho. Na Seção 3.4, são apresentados os conceitos teóricos referentes ao Estados Finitos da Cadeia de Markov e por fim, mas não menos importante, na Seção 3.5, é descrito os principais simuladores de redes utilizados pela comunidade científica e, logo após, é detalhado mais profundamente o simulador Castalia, por ser o escolhido para este trabalho.

3.1 REDES DE SENSORES SEM FIO

Os avanços tecnológicos nos sistemas eletro-mecânicos, na comunicação sem fio e no desenvolvimento de sensores, permitiram a evolução de um tipo especial de rede sem fio chamada de RSSF.

Uma RSSF pode ser vista como um tipo especial de MANET (KURKOWSKI *et al.*, 2005), em que os elementos da rede se comunicam diretamente por meio de enlace de comunicação sem fio. A diferença entre elas está relacionada às tarefas que cada nó executa. Enquanto na MANET os nós executam tarefas distintas, na RSSF eles tendem a executar uma tarefa conjunta, baseado no esforço colaborativo de um grande número de nós (AKYILDIZ *et al.*, 2002).

3.1.1 COMPONENTES RSSF

As RSSF são compostas basicamente pelos nós sensores, as interfaces de comunicação sem fio e os nós *gateway*. Cada um dos nós sensores possuem capacidade de sensoriamento multifuncional, processamento e comunicação. Os componentes básicos dele são: sensor,

processador, memória, rádio e bateria (LOUREIRO *et al.*, 2003) .

Uma RSSF pode ser formada por diferentes sensores. Cada sensor apresenta características físicas e de funcionamento próprias. Loureiro *et al.* (2003) indicam que muitos modelos de dispositivos sensores compartilham duas características: (i) habilidade de sensoriamento diminui quando a distância aumenta; e (ii) a habilidade de sensoriamento pode melhorar com o tempo de exposição, considerando os efeitos decrescentes dos ruídos nas medições.

Para os sensores que são posicionados longe do fenômeno analisado, são necessárias técnicas complexas para filtrar os ruídos. Em geral, os nós sensores são espalhados e se comunicam com um nó central, que armazena e processa os dados (AKYILDIZ *et al.*, 2002). A posição de cada nó não precisa ser planejada ou predeterminada, já que uma das características das RSSF é de se auto-organizar. Quando um sensor é adicionado ou removido, o próprio sistema deve reorganizar a sua topologia de comunicação.

Em alguns casos, a RSSF pode ter nós com a função de atuadores, que possuem a capacidade de alterar valores do ambiente para controlar o objeto monitorado. Quando um nó possui a função de monitoramento e atuação, ele é chamado de transdutor (LOUREIRO *et al.*, 2003).

Os nós sensores podem processar localmente os dados coletados e então enviá-los a uma estação base, através de um nó *gateway*. O nó *gateway* tem a função de permitir a comunicação da rede de sensores com outras redes para que os dados cheguem até o computador que roda a aplicação principal.

Os nós sensores, além de coletar dados do ambiente, podem atuar como roteadores dos dados gerados por outros nós da rede. Nesses casos, os nós são chamados de *sink node* ou nós sorvedouros.

3.1.2 APLICAÇÕES RSSF

Os nós de uma RSSF podem ser compostos por um ou mais sensores e coletar diferentes dados de um ambiente. Eles devem ser pequenos e com baixo consumo de energia, permitindo sua aplicação em diferentes áreas. Akyildiz *et al.* (2002) citam algumas das áreas de possível aplicação das RSSFs: aplicações militares (monitoramento de equipamentos e munições; vigilância do campo de batalha; reconhecimento de forças e terrenos opostos; avaliação de danos de batalha; detecção de ataques químicos e biológicos); aplicações

ambientais (rastreamento de animais; monitoramento das condições ambientais que afetam a agricultura e a pecuária; irrigação; agricultura de precisão); aplicações de saúde (rastreamento e monitoramento dos pacientes dentro de um hospital; administração de medicamentos em hospitais; tele-monitorização de dados fisiológicos humanos); aplicações domésticas (ambientes adaptativos ao usuário).

Dentre os diferentes cenários, destaca-se a aplicação no monitoramento de ambientes, que segundo Loureiro *et al.* (2003), pode ser aplicado tanto em ambientes internos quanto externos a fim de determinar os valores da temperatura, pressão atmosférica, quantidade de luz e umidade.

3.1.3 CARACTERÍSTICAS RSSF

As principais características de uma RSSF, comparada às redes de computadores tradicionais e as redes *ad hoc*, são: possuir um grande número de nós distribuídos; ter restrições de energia; ter limitações em potência, capacidade computacional e de memória nos nós e ser propensos 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 particulares das RSSFs variam em função da aplicação a qual a rede está atendendo. Algumas delas são apresentadas a seguir, baseadas em Loureiro *et al.* (2003) e Akyildiz *et al.* (2002):

- **Endereçamento dos nós sensores:** algumas aplicações exigem a identificação de cada nó individualmente, enquanto para outras apenas o resultado coletado interessa, independente da sua identificação.

- **Agregação dos dados:** dados coletados são sumarizados para posteriormente serem enviados ao nó central, reduzindo o número de mensagens transmitidas pela rede.

- **Mobilidade dos sensores:** nós podem ser fixos ou móveis dentro do ambiente. Nós para coletar a umidade ou temperatura de um ambiente podem ser fixos, enquanto para monitorar a posição de um animal dentro de uma área, devem ser móveis.

- **Quantidade de sensores:** as aplicações podem utilizar de dezenas a milhares de

sensores. Quanto maior a necessidade e interesse na escalabilidade do sistema, maior o número de sensores.

- **Custos de produção:** o custo global da rede está relacionado ao número e o custo dos nós existentes. O custo total não deve superar o custo da implantação de sensores tradicionais para que ela seja viável financeiramente.

- **Restrições de hardware:** um nó sensor é constituído por quatro componentes básicos: unidade de sensoriamento, unidade de processamento, unidade de comunicação e unidade de energia. Dependendo da aplicação podem existir componentes adicionais. Mesmo assim é fundamental que o sistema continue com baixo consumo de energia, baixo custo de produção, seja autônomo e adapte-se ao ambiente.

- **Tolerância a falhas:** os nós podem falhar, por danos físicos ou interferência do ambiente, ou ser bloqueados por falta de energia. A falha de um nó pode afetar a tarefa global do sistema. A rede deve possuir a capacidade de manter as suas funcionalidades sem qualquer interrupção quando algum nó falhar.

- **Limitação da energia disponível:** em grande parte das aplicações os sensores são colocados em áreas remotas, que não estão facilmente acessíveis para manutenção. O tempo de vida do nó é diretamente relacionado à quantidade de energia disponível.

3.1.4 LIMITAÇÃO DA ENERGIA DISPONÍVEL

Akyildiz *et al.* (2002) dividem o consumo de energia da rede em três domínios baseado nas principais funções dos nós sensores: sensoriamento, comunicação e processamento de dados.

As atividades de sensoriamento estão relacionadas com a percepção do ambiente e a coleta de dados. O consumo de energia varia em função da distância do alvo, ruídos do ambiente, tipo de dado coletado, volume de informação e frequência de amostragem (LOUREIRO *et al.*, 2003). Estas variações estão relacionadas ao tipo de aplicação: algumas necessitam de monitoramento constante dos eventos, enquanto outras não.

Os autores dividem o processamento na rede de sensores em duas categorias: processamento de suporte e processamento da informação. O primeiro está relacionado ao gerenciamento, comunicação e manutenção da rede. O segundo com a compressão, cor-

relação e criptografia dos dados, por exemplo. Os dados podem ou não ser processados pelo nó sensor, baseado na função da aplicação ou no envolvimento do nó com a rede.

O maior custo energético da rede está no domínio da comunicação. A energia consumida para comunicação, entre dois pontos, está diretamente relacionada à distância entre eles. O processamento de dados, antes da comunicação, é importante para reduzir o tamanho dos pacotes a serem transmitidos. O gasto energético em processamento de dados é muito menor do que com a comunicação de dados (AKYILDIZ *et al.*, 2002).

3.2 MODELOS DE MOBILIDADE

Conforme Campos e Moraes (2003), os Modelos de Mobilidade para RSSF buscam representar o comportamento de movimentação dos nós móveis em uma RSSF. Porém, algumas vezes, os modelos de mobilidade e os padrões de mobilidade são considerados e discutidos como sendo exatamente a mesma coisa. No entanto, é preciso distingui-los cuidadosamente. Segundo Schindelhauer (2006), o padrões de mobilidade podem ser obtido rastreando objetos móveis da realidade, enquanto os modelos de mobilidade tentam generalizar tais padrões formando um modelo matemático. Ainda, segundo Schindelhauer (2006), os padrões de mobilidade, ou seja, como as coisas e pessoas se movimentam, podem ser separados nos seguintes grupos: pedestres, veiculares, antena, meio dinâmico, robótico e movimento aeroespacial.

Esta dissertação foca em generalizar, ou descrever um modelo, que se aproxima do padrão de mobilidade pedestre, mais especificamente o padrão encontrado em animais livres em pasto. Em seu *survey*, Schindelhauer (2006) dedica um capítulo inteiro aos padrões de mobilidade e, segundo ele, a mobilidade pedestre descreve os padrões de caminhada de pessoas ou animais. As principais características são o uso pleno do plano bidimensional com obstáculos ocasionais e sua natureza caótica. Geralmente mas não necessariamente, possuem comportamento de grupo e é sempre limitada em velocidade, pois as pernas agem como pêndulos invertidos.

Um bom modelo de mobilidade é um pré-requisito essencial para avaliação do desempenho de protocolos para RSSF com mobilidade dos nós (NABI *et al.*, 2011). Os modelos de mobilidade tentam imitar o comportamento dos nós móveis na realidade em um cenário específico, caracterizando os padrões estocásticos do movimento do nó. Como esses modelos determinam as posições dos nós em qualquer instância de tempo durante

a simulação, consequentemente irão causar uma grande influência à topologia da rede e também às propriedades do link. Logo, a precisão do modelo tem um impacto na acurácia e performance alcançada da rede avaliada durante a simulação. Importante frisar que o modelo de mobilidade correto depende fortemente do cenário de aplicação, mais especificamente, qual padrão de mobilidade deseja alcançar.

A qualidade do canal e da conexão entre os nós dependem fortemente da posição relativa dos nós sensores. Isso torna a precisão do modelo de mobilidade para RSSF um fator extremamente crítico. Na verdade, simular um protocolo para estes tipos de redes sem utilizar um modelo de mobilidade apropriado não é confiável. Existem vários modelos de mobilidade apresentados na literatura para MANETs e RSSF. Uma pesquisa detalhada desses modelos pode ser encontrada em (CAMP *et al.*, 2002).

Em geral, pode-se classificar os modelos de mobilidade em duas classes principais denominadas modelos de mobilidade individual e modelos de mobilidade em grupo. Na primeira classe, não há correlação entre o movimento dos nós diferentes e os modelos de padrões de mobilidade de nó individuais independentemente da mobilidade dos outros nós na rede. O último leva em consideração um grupo de nós que potencialmente tem um relacionamento particular, que introduz a correlação entre suas posições. As atividades sociais dos seres humanos e animais em pasto ou em ambientes livres são exemplos de mobilidade em grupo.

3.2.1 MODELOS DE MOBILIDADE INDIVIDUAL

O Random Walk Mobility Model (RWMM) (ZONOOZI; DASSANAYAKE, 1997) é um modelo de mobilidade de nó individual comumente usado no qual um nó seleciona aleatoriamente uma direção e um valor de velocidade de um determinado intervalo. O nó então se move com um intervalo de tempo constante ou até uma distância constante percorrida. O movimento ocorre dentro de um determinado espaço retangular, que é a área de simulação. O nó então repete a seleção aleatória e o processo de movimento.

O Random Waypoint Mobility Model (RWPM) é uma versão adaptada do RWMM (HYYTIÄ; VIRTAMO, 2007) em que um tempo de pausa é inserido entre as mudanças de direção e velocidade. O tempo de pausa é selecionado aleatoriamente de um determinado intervalo. Na RWPM, um destino é selecionado aleatoriamente da área de simulação e um nó se move em direção a essa posição com uma velocidade escolhida aleatoriamente.

Uma relação específica entre o tempo de pausa e a velocidade pode ser aplicada com base em uma aplicação específica, para ajustar o modelo para uma rede mais estável ou uma rede com mudanças de topologia frequentes.

Tanto o RWMM, quanto o RWPM sofrem o problema da concentração do nó. A probabilidade de um nó estar no centro da área de simulação é maior e os *clusters* de nós se formam em torno do centro. O modelo de mobilidade de direção aleatória (RDMM) (ROYER *et al.*, 2001) tenta aliviar esse problema forçando os nós a encontrar uma borda em cada etapa de movimento. Além disso, uma versão probabilística de uma caminhada aleatória é proposta em Chiang e Gerla (1998) que usa uma matriz de probabilidade, que pode ser especifica para um determinado contexto ou aplicação, para determinar a posição alvo da próxima etapa. No modelo de mobilidade proposto por este trabalho também utilizamos matrizes de probabilidade, contextualizadas e específicas para o cenário que deseja-se simular.

Todos os modelos mencionados até agora não possuem memória. Isso significa que o passo de movimento completo não tem qualquer impacto na decisão sobre os parâmetros de movimento do próximo passo. Em Liang e Haas (1999), propõe-se o modelo Random Gauss-Markov Mobility (RGMM), no qual o valor de velocidade e direção é calculado usando uma distribuição Markoviana e Gaussiana combinada. Pela propriedade Markoviana, os valores de velocidade e direção no passo n^{th} passo são calculados de acordo com o seu valor na etapa $(n-1)^{th}$. Por outro lado, usando uma distribuição Gaussiana com os valores médios dados para a direção e velocidade, a aleatoriedade é inserida no processo de seleção. O impacto dessas duas partes é controlável configurando apenas um fator de sintonia.

3.2.2 MODELOS DE MOBILIDADE EM GRUPO

No domínio das redes sem fio, existem muitas situações em que os padrões de movimento de diferentes nós são dependentes entre si. Os padrões de mobilidade do gado mostram principalmente comportamentos de grupo (por exemplo, um indivíduo é tido como líder do grupo e os outros animais irão acompanhá-lo na maioria do tempo). Devido a isto, vários modelos de mobilidade foram propostos na literatura tentando modelar o padrão de movimento em grupo e, principalmente, dos seres humanos em cenários específicos. O modelo de mobilidade em coluna, o modelo de perseguição e o modelo de comunidade

nômade, todos apresentados em Sánchez e Manzoni (2001), são algumas abordagens iniciais para modelar esses padrões de mobilidade correlacionados. Seguindo o conceito de maior densidade de nó em locais mais populares temos, por exemplo, o modelo *Small World In Motion (SWIM)* (MEI; STEFA, 2009) que apresenta um modelo de mobilidade baseado no fato de que os humanos frequentam locais mais próximos de sua casa e locais onde podem encontrar muitas outras pessoas. Temos, também, o modelo *N-Body* apresentado em Zhao e Sichitiu (2010) que usa alguns traços reais de movimentos humanos e tenta capturar métricas de informações sociais a partir deles. O modelo, então, sintetiza essa informação para fazer traços de saída reproduzindo a heterogeneidade dos traços de entrada.

Entre todos os modelos de mobilidade de grupo, o modelo *Reference Point Group Mobility (RPGM)* (HONG *et al.*, 1999) é um modelo geral que pode ser utilizado para modelar muitos cenários. Na verdade, muitos modelos propostos recentes são de algum modo casos especiais do modelo RPGM. No modelo de mobilidade RPGM, um Centro Lógico (LC) é definido para o grupo de nós, cujo movimento define o movimento do grupo inteiro. Todo grupo i tem um vetor de movimento de grupo GM_i que determina o movimento do centro lógico do grupo (LC_i).

Observe que o RPGM não prescreve um padrão de mobilidade para mover o LC ou para o movimento individual dentro do grupo. Os padrões apropriados devem ser projetados de acordo com os requisitos do cenário exato.

3.3 COMPORTAMENTO ANIMAL

A etologia é a parte da ecologia que trata dos hábitos dos animais e da acomodação dos seres vivos às condições do ambiente. Chama-se de comportamento aquilo que percebe-se das reações de um animal ao ambiente que o cerca, segundo J.D.Carthy (1969). Para o estudo do comportamento é necessário observações dos movimentos do animal além de suas ações, a fim de compreender em que condições elas acontecem. A identificação de comportamentos pode servir para o gerenciamento dos animais e das áreas em que eles vivem. Além disso, segundo Scheibe e Gromann (2006) alguns desvios nos padrões normais de comportamento do animal podem ser resultados de doenças ou de perturbações exteriores, ou, até mesmo, uma oportunidade de otimização do rebanho, como por exemplo nos casos de percepção antecipada do cio.

Conforme proposto pelo *Farm Animal Welfare Council* na Grã Bretanha os animais necessitam estar livres e sem nenhuma influência externa para poderem expressar seu comportamento normal. Conforme relatado por Kilgour *et al.* (2012), uma abordagem geralmente utilizada para tentar alcançar o comportamento habitual de animais domésticos é estudar seus ancestrais selvagens em um ambiente sem a interferência dos seres humanos. Porém, ao se tratar especificamente do gado domesticado a pasto, não é viável essa abordagem, pois esse possui como ancestral o *Aurochs*, uma espécie extinta em 1627 segundo Clutton-Brock (1999), impossibilitando que seja observada.

Os primeiros estudos sobre comportamento animal (por volta de 1920) foram realizados utilizando a observação humana e registros manuscritos das atividades. Com a evolução do homem e o surgimento de novas tecnologias, novas abordagens foram adotadas, sendo que atualmente o uso de câmeras de vídeo e binóculos, com tecnologia de visão noturna, têm sido utilizadas como ferramentas de apoio (ANDERSON *et al.*, 2013).

Apesar destas serem ferramentas eficientes, existem problemas relacionados às limitações do homem, como a fadiga em trabalhos que exigem longos períodos de observação e a influência da presença do observador no habitat do animal. O uso de equipamentos eletrônicos, além de substituir a presença do homem, pode melhorar a qualidade dos dados coletados.

Diversos trabalhos desenvolvidos entre 1997 e 2002, utilizando dispositivos de GNSS para monitorar o comportamento do gado, são apresentados por Anderson *et al.* (2013). A tecnologia GNSS combinada com outros equipamentos eletrônicos é utilizada com frequência para monitorar os animais e seus comportamentos.

Watanabe *et al.* (2005) cita trabalhos que fazem uso de acelerômetros para medir os sinais do movimento e da postura do corpo em diferentes animais. O trabalho, que monitorou um gato doméstico, indica que os dados de aceleração podem ser úteis para indicar as atividades e o comportamento de animais terrestres. No caso de herbívoros, os dados poderiam ser utilizados para indicar a quantidade de grama ingerida pelo animal, por exemplo. Capturar os movimentos de animais livres requer pequenos sensores e o armazenamento de grandes conjuntos de dados. Comportamentos básicos normalmente envolvem movimentos de todo o corpo: postura do corpo, sentidos de movimento característicos e as frequências de movimentos distintos (SCHEIBE; GROMANN, 2006). A partir dos dados de aceleração deve ser possível identificar os diferentes comportamentos

por meio da análise do padrão de aceleração dos movimentos do animal.

As soluções comerciais disponíveis no mercado para rastrear animais, em especial da tecnologia GNSS, são caras. As opções de baixo custo normalmente são soluções criadas a partir de dispositivos comerciais, não projetadas para o monitoramento animal. Existem vários trabalhos que adaptaram essas soluções com sucesso, incluindo o trabalho de Jesus e Pires (2014) que utiliza um módulo GPS conectado à plataforma Arduino.

A utilização de dispositivos não comerciais pode apresentar outras vantagens, como a possibilidade de inclusão de outros recursos que possam avaliar novas variáveis relacionadas ao comportamento animal e o menor tempo de manutenção, em caso de falhas do equipamento, considerando que não há dependência do fabricante para a solução (ANDERSON *et al.*, 2013). Porém, há a necessidade de alguém localmente capacitado para que se ofereça suporte.

Em geral, as soluções para identificação do comportamento animal, utilizando recursos computacionais, são limitadas pelo consumo de energia e a capacidade da bateria. Para aplicações no Agronegócio, em especial de monitoramento do gado (HUIRCÁN *et al.*, 2010), é fundamental que o sistema possua recursos de baixo custo, em função do elevado número de elementos e uma gestão de energia eficiente, para manter o sistema o maior tempo disponível (considerando a dificuldade de acesso aos nós).

A tecnologia de RSSF é uma candidata natural para este tipo de aplicação (NADIMI *et al.*, 2008) pelo fato de incorporar em um único sistema a capacidade de sensoriamento, processamento e comunicação por uma rede sem fio, permitindo que os dados sejam coletados e encaminhados regularmente para um sistema central de controle e gestão.

Um dos métodos mais comum para equipar os animais é a utilização de colares. O colar não deve ficar apertado nem solto no pescoço do animal, pois apertado pode abrasonar a pele e solto pode enroscar em algum objeto do ambiente (galho, cerca, etc) (ANDERSON *et al.*, 2013). Além disso, alguns componentes precisam ficar em posição fixa, como antenas ou sensores, pois se o colar se movimentar no pescoço do animal a qualidade dos dados coletados será afetada.

Algumas alternativas citadas por Anderson *et al.* (2013), a fim de evitar que o colar deslize sobre o pescoço do animal, são: deixar os componentes mais pesados (por exemplo, as baterias) na parte inferior do colar, a fim de agirem como contrapeso; ou um colar composto por material elástico, que absorva o suor e se estenda pelo pescoço do animal

sem abrasoná-lo.

Um dos desafios, ao colocar equipamentos em animais livres na natureza, é garantir que os comportamentos do animal não sejam adversamente afetados. O design e o tamanho do dispositivo devem ser projetados de maneira que possua o menor peso e tamanho possíveis. Watanabe *et al.* (2005) indicam que o equipamento não deve ter mais que 5% do peso do animal enquanto Anderson *et al.* (2013) recomendam que o peso seja menor que 1% da massa corporal do animal. Após equipar o animal, deve-se aguardar um período para que o mesmo se adapte ao equipamento. Não há um tempo ideal de espera indicado, pois é variável conforme a espécie avaliada.

3.3.1 COMPORTAMENTO BOVINO

A identificação do comportamento animal é de suma importância quando se deseja construir um modelo de mobilidade que tente se aproximar da realidade. O intuito deste trabalho é elaborar um modelo de mobilidade para simulações de qualquer animal em pastagem ou ao ar livre (sem confinamento), ou seja, um modelo geral que possa ser configurado de acordo com a necessidade e contexto da simulação que for ser realizada. Todavia, se faz necessário criar uma instância especifica para validar o conceito e demonstrar o funcionamento do modelo. Então, escolheu-se o comportamento dos bovinos, mais especificamente o gado leiteiro. A estratégia foi especializar para alcançar um padrão e depois transformar para algo mais genérico sem perder as características que são importantes para o estudo de RSSF. A escolha dos bovinos foi feita devido a grande quantidade de trabalhos existentes nessa área, a importância desse animal para economia de nosso país e à proximidade e parceria existente entre a Universidade Federal de Juiz de Fora e a Embrapa Gado de Leite, como já citado anteriormente.

O *review* Kilgour (2012) publicado sobre o comportamento do gado pode ser considerado como o grande resumo sobre esse assunto e mostrou-se surpreso pela não existência de um artigo com esse fim e também com a baixa quantidade de trabalhos existentes sobre o assunto. Esse artigo analisou 22 trabalhos sobre o bem-estar animal, realizados entre 1927 e 2009, e apresentou 40 comportamentos identificáveis do gado, classificados em cinco grandes categorias: comportamentos de manutenção (por exemplo: pastando ou ruminando), auto expressão (por exemplo: raspando a pata no chão ou esfregando o pescoço em algum objeto), social (por exemplo: lambendo outro animal ou amamen-

tando), procriação (por exemplo: Inseminando ou parindo) e social mútuo (por exemplo: parado ou andando ao lado de outro animal). Apesar da diversidade, os comportamentos mais comuns observados foram: pastando, parado, ruminando, andando, bebendo água e lambendo algum tipo de suplemento. Além disso, os três primeiros ocupam entre 90% a 95% do dia do animal, tornando os outros comportamentos listados insignificantes, principalmente ao tratarmos de mobilidade. Outra informação para corroborar com o descarte de grande parte dos comportamentos listados é que a maioria dos trabalhos revisados, considera apenas os comportamentos pastando, parado e ruminando, ignorando outros comportamentos do animal. Apenas três dos 22 estudos revisados tinham informações detalhadas sobre os diversos comportamentos: Herbel e Nelson (1966), Zemo e Klemmedson (1970) e Hall (1989). Apesar dos dados sobre outros comportamentos do gado serem escassos na literatura, a maior parte dela indica que pastando é o comportamento mais comum.

Além das limitações dos trabalhos, quanto à restrição de comportamentos avaliados, outros problemas encontrados são: o pequeno tempo de observação, número baixo de animais no rebanho, áreas de pasto restritas e observação de um único animal do rebanho. Em outro trabalho, Kilgour *et al.* (2012) realizou a observação do comportamento utilizando seis rebanhos em cinco propriedades diferentes, em que todos os animais foram observados, em uma área de pastagem maior que 200 hectares, durante seis sessões nos períodos da manhã e da tarde, realizado em dois meses (agosto e setembro) de dois anos consecutivos.

No trabalho de Kilgour *et al.* (2012) foi utilizado o conceito de comportamentos primários e secundários, em que um comportamento secundário é aquele que acontece ao mesmo tempo que um comportamento primário. Os comportamentos primários, definidos pelos autores, foram: Em Pé e Deitado. Ao todo foram observados 18 comportamentos. Em nenhum momento, em qualquer um dos seis rebanhos, todos os animais estavam no mesmo comportamento simultaneamente. Isso reforça o fato de que é necessário considerar e compreender a variação de comportamento de um animal para outro, ou seja, a seleção do comportamento do animal, teoricamente, não é influenciado pelo grupo ao qual ele participa. Alguns trabalhos, avaliados por Kilgour *et al.* (2012), indicaram momentos em que todos os animais estavam pastando simultaneamente, porém em todos eles poucos animais foram observados. Por se tratar de um comportamento mais genérico e para

atender esse convívio em grupo, esse trabalho propõe na instância do modelo um Tipo de Área denominado "Pastagem". No qual os bovinos de um determinado grupo poderão permanecer e apresentar comportamentos diferentes (individuais). No trabalho Hancock (1954) também conclui-se que mesmo gados de um mesmo rebanho e de uma mesma cria apresentam diferenças comportamentais.

Em sua dissertação de mestrado, Lomba (2015) realiza uma observação importante sobre o trabalho de Kilgour *et al.* (2012) que, apesar de eliminar alguns dos problemas encontrados em trabalhos anteriores, a pesquisa não considerou o comportamento do animal no período noturno e em condições de chuva. Além disso, a atividade de ruminação foi considerada juntamente com a atividade de descanso (parado), já que os observadores não conseguiam ver a boca do animal (observações visuais realizadas sem uso de tecnologia avançada). Mesmo com estas restrições, confirmou-se os resultados de outros trabalhos quanto às principais atividades do gado: a atividade de pastagem representou 51% do tempo (6,1h das 12h observadas), seguido pelas atividades de descanso (em pé ou deitado) e andando. Dentre os seis rebanhos avaliados, a variação da proporção total destas atividades foi pequena (entre 95,4% e 97,7% do tempo total observado).

Nos trabalhos avaliados por Kilgour (2012) em seu *review*, que registraram e separaram os comportamentos ocorridos durante o dia e a noite (11 dos 22 trabalhos), a atividade pastando aconteceu com maior frequência durante a luz do dia, com poucos registros durante a noite. O pico da atividade pastando se dá no nascer e pôr do sol, informação confirmada em cinco dos seis rebanhos avaliados por Kilgour *et al.* (2012). Já a atividade de ruminação, registrada tanto quando o animal estava em pé quanto deitado, ocorre com maior frequência no período noturno, assim como a atividade de descanso.

Outro trabalho interessante analisado durante a revisão bibliográfica é o artigo Degasperi *et al.* (2003), no qual o autor descreve que parte significativa da vida de um bovino, como herbívoro, é consumida nos atos de ingestão e ruminação (55% a 60%). Descreve também que os bovinos geralmente procuram sombra, bebedouros, poças e cursos d'água para se refrescarem e se livrarem de insetos. Essas informações são bastante uteis ao formular a instância de configuração do modelo, pois os tipos de áreas que possuem algumas dessas nuances terão uma maior probabilidade de serem frequentados pelos animais. Outra informação importante é que o ato de coçar-se ou lamber-se (*grooming*), por si mesmo ou por companheiros de rebanho, são elementos essenciais na manutenção do corpo e

são importantes na estrutura social do rebanho. Demonstrando assim o comportamento como animal social e de convívio em grupo. Praticamente metade das horas que um bovino adulto encontra-se acordado é gasta descansando, normalmente deitados em decúbito ventral, com os pés e pernas embaixo do corpo. As vacas em lactação eram ordenhadas duas vezes ao dia, ocasião em que recebiam 4 kg de ração comercial com 16% de proteína. O teste de *Tukey* foi utilizado para analisar as médias dos períodos de tempo. Durante um período de 15 dias manteve-se o registro de 24 horas de cada atividade de cada animal, resultando nos seguintes valores de médias, desvio padrão e coeficiente de variação de cada estado ou atividade: deitado (4,97 ± 0,33; 6,63%), em pé (4,35 ± 0,15; 3,44%), ruminando (3,69 ± 0,12; 3,25%), andando (98 ± 0,12; 6,06%), defecando (4,32 ± 0,20; 4,62%), urinando (4,67 ± 0,23; 4,92%), bebendo água (2,75 ± 0,21; 7,63%), lambendo (2,22 ± 0,24; 10,81%) e comendo (2,93 ± 0,18; 6,14%).

Segundo estudo de González *et al.* (2015), que tem como principal objetivo desenvolver e avaliar uma metodologia para caracterizar a estrutura dos dados obtidos eletronicamente e classificar esses dados em atividades comportamentais ou estados, usando GPS e dados de sensores de movimento de coleiras em novilhos, os estados pertinentes à serem considerados são: alimentação, repouso, ruminação, viagens e outros comportamentos ativos. Inicialmente, pastando com a cabeça para baixo, navegando e a busca de alimentos foram registrados separadamente, porém pastagem ocupava mais de 95% de todos os comportamentos de alimentação. Então, decidiu-se unir as 3 atividades na atividade chamada "alimentação" (ou "forrageamento") por simplicidade. Após a recuperação do colar dos novilhos no final dos testes, os cartões de memória eram removidos e os dados eram baixados para serem processados. Ou seja, não foi utilizado uma RSSF para a coleta de informações. Não havia comunicação entre os nós, não formando redes.

Já no trabalho Martiskainen *et al.* (2009), os padrões de comportamento medidos incluíam estar de pé, deitado, ruminando, alimentando, caminhando normal e coxo, deitado e de pé. As medidas de sensibilidade e precisão foram utilizadas para avaliar o desempenho do modelo. Os modelos de classificação SVM alcançaram um reconhecimento razoável de permanência (80% de sensibilidade, 65% de precisão), deitado (80%, 83%), ruminar (75%, 86%), alimentação (75%, 81%), andar normalmente (79%, 79%) e caminhadas coxo (65%, 66%). Os resultados foram fracos para condição deitado (0%, 0%) e em pé (71%, 29%).

No trabalho (CECCHIN, 2012) os estados considerados são: deitada em ócio, em pé

em ócio, deitada ruminando, em pé ruminando, em pé se alimentado, em pé tomando água, sendo ordenhado e em movimento. Neste caso, percebe-se que estes estados podem ser associados a locais, criando a possibilidade de deduzir as transições entre os estados. Esse trabalho apresenta os percentuais dos comportamentos listados (A: andando; DO: deitado em ócio; DR: deitado ruminando; EO: em pé ócio sobre a cama; EOC: em pé ócio no corredor; ER: em pé ruminando sobre a cama; ERC: em pé ruminando no corredor; ES: em pé se alimentando) em cada período do dia, conforme mostra a imagem 3.1. Lembrando que este trabalho foi realizado para gado em confinamento no modelo *free-stall*.

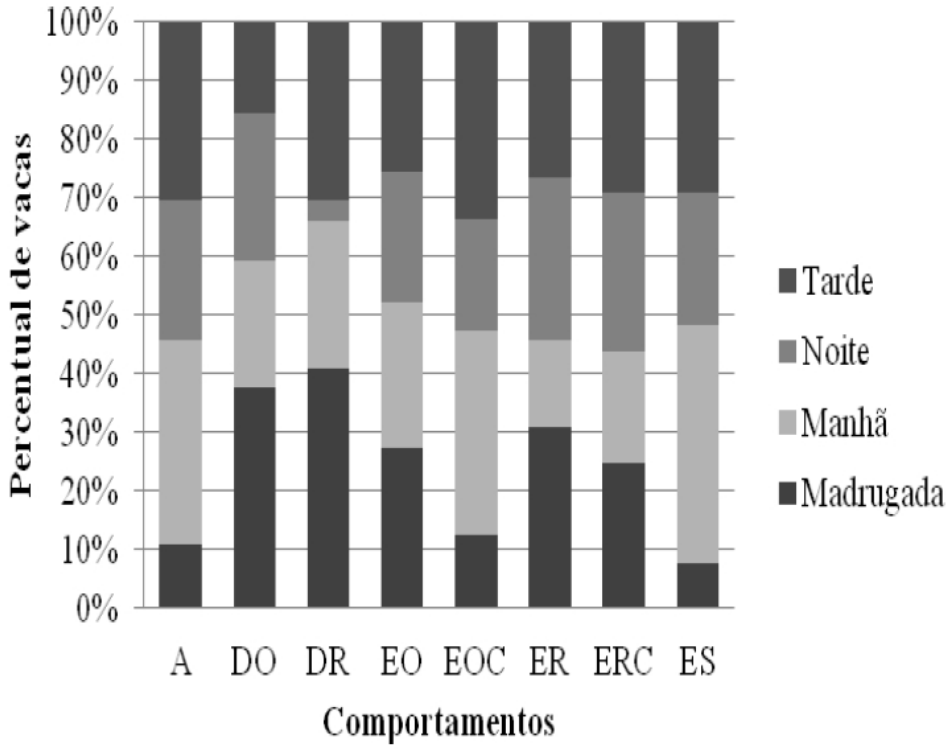

Figura 3.1: Distribuição em porcentagem dos comportamentos em cada período (CECCHIN, 2012).

O trabalho de Guo *et al.* (2006) desenvolveu uma RSSF para compreender a movimentação e outros comportamentos dos bovinos em pastagem, utilizando colares com sensores de GPS, acelerômetros e magnetômetros. O artigo apresenta a ideia do uso de colares com sensores FleckTM, ilustrados na imagem 3.2 (GPS, acelerômetros e magnetômetros) para analisar o comportamento do animal, principalmente nas áreas de permanência e de trânsito do animal. No estudo, 6 animais foram analisados individualmente. Depois que os dados foram coletados, tiveram que ser agrupados para a criação de um modelo do comportamento dos animais. A coleta dos dados foi feita durante 4 dias, nos 2 primeiros

dias os dados coletados foram usados apenas para criar um modelo padrão do comportamento dos animais e definir as principais áreas de permanência e de trânsito do animal. Os autores citam que o monitoramento de animais precisa ser capaz de lidar com a mobilidade e o movimento dos animais. Consequentemente, os links de comunicação precisam ser capazes de lidar com essa mobilidade e ser capazes de cobrir longas distâncias entre os nós móveis.

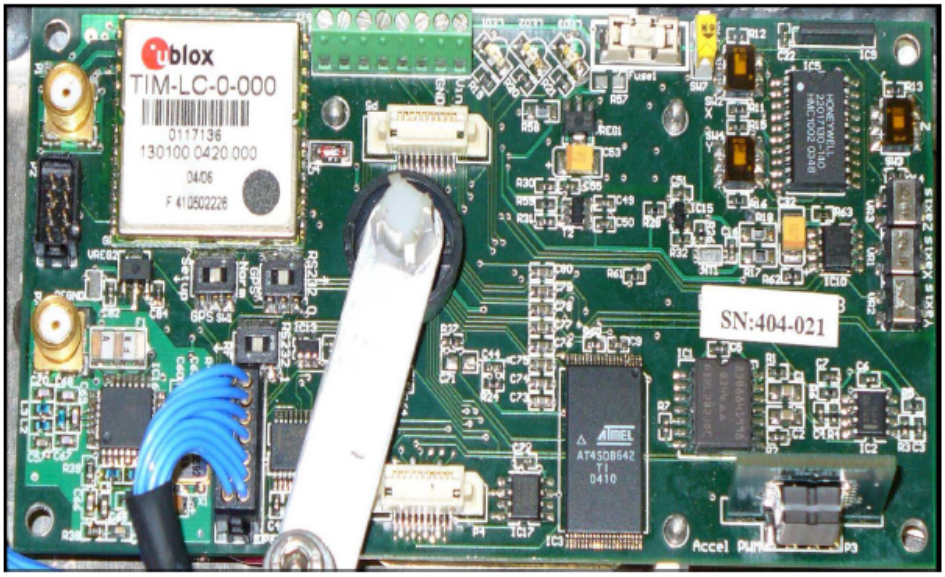

Figura 3.2: O Fleck2 (120mm × 60mm). Tem a bordo um sensor de temperatura, 3 acelerômetros, 3 magnetômetros e um receptor GPS (GUO *et al.*, 2006).

Guo *et al.* (2006) realizou uma classificação hierárquica dos comportamentos. A camada mais alta divide os comportamentos em estacionários e viajantes ou móveis. Logo após, separa os comportamentos estacionários: em sentado ou em pé, e os comportamentos móveis: em correndo e andando. E conforme demonstra a figura 3.3, realiza mais uma camada inferior de classificação dos comportamentos. Os autores ainda citam como um objetivo de trabalho futuro classificar as diversas atividades em uma estrutura mais completa. Esse trabalho também apresenta uma variação de velocidade de movimento dos bovinos de 0 a 4 m/s.

Já em (LOMBA, 2015), o modelo gerado foi criado para predizer quatro comportamentos primários: Pastando/Procurando, Andando, Em Pé e Deitado (figura 3.4). O esquema para identificação dos comportamentos primários utilizou dados dos sensores de movimentação (acelerômetro, giroscópio e magnetômetro) e os dados de posicionamento (GPS). Os comportamentos: em pé-parado e em pé-ruminando foram agrupados em uma

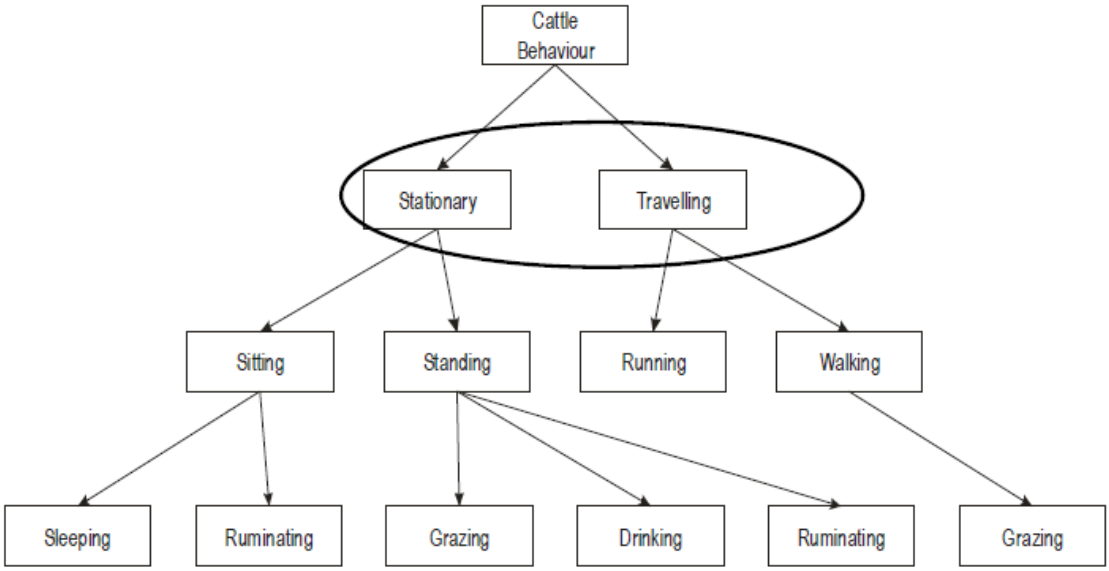

Figura 3.3: Uma maneira de classificar os comportamentos animais (GUO *et al.*, 2006).

única classe, definida como em pé; o mesmo foi realizado com as classes deitado-parado e deitado-ruminando, que foram agrupadas na classe Deitado.

Figura 3.4: Comportamentos primários identificados a partir dos dados de posição e movimentação (LOMBA, 2015).

Sabe-se que fatores ambientais relacionam-se diretamente com o comportamento animal. Oliveira (2013) elenca a temperatura, a radiação solar e a umidade relativa como os principais elementos do ambiente relacionados ao comportamento bovino. Segundo Schütz *et al.* (2009), o gado pode identificar pequenas alterações climáticas e buscar situações para contorná-las, como procurar áreas com sombra para reduzir a temperatura

corporal. No trabalho de Kilgour *et al.* (2012), identificou-se que 10,9% do tempo os animais estavam em áreas de sombra.

3.4 MATRIZ DE MARKOV

Uma matriz de Markov (em homenagem ao matemático russo Andrey Markov), também denominada matriz de transição ou matriz estocástica, é uma matriz quadrada que tem duas características: 1) todas as entradas são não-negativas e 2) todas as colunas ou linhas devem ter a soma de entradas igual a 1.

É utilizada para descrever as transições da cadeia de Markov.

Por exemplo, a matriz abaixo é uma matriz de Markov:

$$\begin{bmatrix} 0,1 & 0,0 & 0,7 & 0,2 \\ 0,3 & 0,3 & 0,0 & 0,4 \\ 0,0 & 0,0 & 0,8 & 0,2 \\ 0,6 & 0,1 & 0,2 & 0,1 \end{bmatrix}$$

3.4.1 PROCESSO ESTOCÁSTICO

Dentro da teoria das probabilidades, um processo estocástico é uma família de variáveis aleatórias representando a evolução de um sistema de valores com o tempo. É a contraparte probabilística de um processo determinístico. Ao invés de um processo que possui um único modo de evoluir, como nas soluções de equações diferenciais ordinárias, por exemplo, em um processo estocástico há uma indeterminação: mesmo que se conheça a condição inicial, existem várias, por vezes infinitas, direções nas quais o processo pode evoluir.

Em casos de tempo discreto, em oposição ao tempo contínuo, o processo estocástico é uma sequência de variáveis aleatórias, como por exemplo uma cadeia de Markov. As variáveis correspondentes aos diversos tempos podem ser completamente diferentes, o único requisito é que esses valores diferentes estejam todos no mesmo espaço, isto é, no contradomínio da função. Uma abordagem possível é modelar as variáveis aleatórias como funções aleatórias de um ou vários argumentos determinísticos, na maioria dos casos, em relação ao parâmetro do tempo. Apesar de os valores aleatórios de um processo estocástico em momentos diferentes parecerem variáveis aleatórias independentes, nas situações mais

comuns, eles exibem uma complexa dependência estatística.

Exemplo de processos estocásticos incluem flutuações nos mercados de ações e nas taxas de câmbio, dados médicos como temperatura, pressão sanguínea e variações nos potenciais elétricos do cérebro registrados em um eletroencefalograma, fluxo turbulento de um líquido ou gás, variações no campo magnético da Terra, mudanças aleatórias no nível de sinais de rádio sintonizados na presença de distúrbios meteorológicos, flutuação da corrente em um circuito elétrico na presença de ruído térmico, movimentos aleatórios como o movimento Browniano ou passeios aleatórios, entre outros.

Uma generalização de um processo estocástico, o campo aleatório é definido ao permitir que as variáveis sejam parametrizadas por membros de um espaço topológico ao invés do tempo. Exemplos de campos aleatórios incluem imagens de estática, topografia, ondas de superfície e variações na composição de um material heterogêneo.

Mais genericamente, qualquer tipo de evolução temporal, determinística ou essencialmente probabilística, que seja analisável em termos de probabilidade, pode ser chamada de processo estocástico.

3.5 SIMULADORES DE REDES

Com o avanço da tecnologia em redes de computadores, experimentos em ambientes reais têm se tornado cada vez mais complexos. Por isso, muitos pesquisadores optam por realizar seus experimentos em ambientes simulados computacionalmente. 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 de implementação, ambiente controlado, facilidade na coleta de dados, universalização da informação e compartilhamento de experimentos e resultados. Assim, diversos simuladores foram criados nas últimas décadas, desde os que são de propósito acadêmico até os que são de propósito comercial.

Várias realidades já podem ser implementadas e analisadas através dos simuladores atuais, como a tecnologia *ZigBee* (HAMMOODI *et al.*, 2009), ou até mesmo Redes Definidas por *Software* (mais conhecida pela sua sigla em inglês: SDN). Alguns destes simuladores possuem a funcionalidade de atribuir a um componente de rede o próprio *firmware* real. Dessa forma, há diversos trabalhos que fazem uso de simuladores de redes para os mais diversos fins. Entretanto, este uso intensivo de simuladores levanta uma série de questões sobre os trabalhos que os utilizam. Algumas dessas questões são: confiabilidade

e a validade dos dados apresentados, falta de reprodutibilidade dos cenários e falta de compartilhamento das implementações realizadas.

No ano de 2000, o pesquisador Lee Breslau, junto com sua equipe, realizou um trabalho no qual evidencia os avanços em simulações de rede na época (BRESLAU *et al.*, 2000). A principal ideia por trás desse trabalho é a criação de um simulador padrão adequado para estudos acadêmicos. A popularidade e aceitação deste simulador aumentariam a confiabilidade dos resultados, de acordo com o autor. Entre as contribuições de sua pesquisa está a apresentação e avaliação de um novo simulador, *NS (Network Simulator)*, contendo uma série de modelos e configurações para simulações de redes de computadores. Além do NS, vários outros simuladores foram criados para atender as redes ditas comuns ou tradicionais, com base na pilha de protocolo TCP-IP e cabeada. Como, pode-se citar: NS2, NS3, ONE, J-Sim, GloMoSim, Avrora, entre outros.

No entanto, apesar da grande quantidade de pesquisas realizadas em vários centros acadêmicos e industriais no que tange RSSF, essa área ainda se encontra aberta, com muitos aspectos relacionados a arquitetura, protocolos e camadas de software não muito bem definidas e padronizadas (MOTTOLA *et al.*, 2010). Tentar simular RSSF implica em preocupações inexistentes nas redes tradicionais. Segundo Minakov *et al.* (2016), além das questões de design inerentes às redes tradicionais, o desenvolvimento de RSSFs enfrenta um conjunto de objetivos específicos que antes eram considerados de importância secundária. Isso inclui, por exemplo, o gerenciamento dos recursos limitados de *hardware* presentes no nó da rede, a carga limitada da bateria e a infraestrutura de comunicação *Ad Hoc* personalizada.

Além disso, a grande variedade de protocolos MAC e de roteamento não padronizados disponíveis complica ainda mais o processo de *design*, dificultando a escolha da solução correta que melhor se adapta a um determinado sistema (LANGENDOEN; MEIER, 2010). É improvável que essa situação mude no futuro próximo devido à própria natureza específica do aplicativo dos sistemas RSSF. Como consequência, é essencial poder explorar e validar os vários aspectos desse design complexo (induzido por diferentes opções de projeto) antes de prosseguir para a implantação no mundo real.

Dentre os simuladores específicos ou ao menos otimizados para RSSF temos o Castalia (BOULIS, 2007), MiXiM (KÖPKE *et al.*, 2008), TOSSIM (LI; SERPEN, 2011) e o COOJA (ERIKSSON *et al.*, 2009). As simulações e implementação do modelo proposto

foram realizadas usando o Castalia devido aos motivos apresentados no próximo tópico.

Outro ponto crucial que deve ser analisado ao se trabalhar com simuladores é a credibilidade dos mesmos. Com o crescimento de trabalhos que utilizam simuladores de redes, a credibilidade dos resultados começou a ser questionada, gerando uma grande quantidade de trabalhos discutindo sobre essa vertente. Como, em seu trabalho de 2005, Stuart Kurkowski realizou uma análise dos artigos publicados no *International Symposium on Mobile Ad Hoc Networking and Computing*, entre os anos 2000-2005. Em seu estudo, Stuart relata uma série de falhas encontradas nos artigos, que diminuem drasticamente a credibilidade dos trabalhos (KURKOWSKI *et al.*, 2005). A figura 3.5 mostra parte do resultado deste trabalho.

Simulation Input Parameters		
Totals	**Percentage**	**Description**
109 of 114	95.6%	Conducted MANET protocol simulation studies.
62 of 109	56.9%	Stated the number of nodes used in the study.
58 of 109	53.2%	Stated the size of the simulation area.
62 of 109	56.9%	Stated the transmission range.
49 of 109	45.0%	Stated the simulation duration.
41 of 109	37.5%	Stated the traffic send rate.
31 of 109	28.4%	Stated the traffic type (e.g., CBR, etc.)
39 of 109	35.8%	Stated the number of simulation runs (iterations).
42 of 109	38.5%	Used mobility in the study.
34 of 42	81.0%	Stated the mean speed of the nodes.
26 of 42	61.9%	Stated the speed variance about the mean.
21 of 42	50.0%	Stated the mean pause time of the nodes.
16 of 42	38.1%	Stated the pause time variance about the mean.
38 of 42	90.5%	Stated which mobility model was used.
25 of 38	65.8%	Used the random waypoint mobility model [16].
2 of 25	8.0%	Used the steady-state version of the random waypoint mobility model [26].
2 of 38	5.3%	Used a group mobility model [14, 32].
4 of 38	10.5%	Used a grid/road mobility model (e.g., [7]).
5 of 38	13.2%	Used the random direction mobility model (e.g., [36]).

Figura 3.5: Tabela comparando resultados das pesquisas estudadas (KURKOWSKI *et al.*, 2005).

Pode-se observar que há uma série de parâmetros que não são incluídos nos trabalhos. Parâmetros básicos, como por exemplo, a duração da simulação foram incluídos em apenas 45% das pesquisas analisadas. Além disso, muitos falham no critério de reprodutibilidade, diminuindo assim a credibilidade dos resultados e do trabalho em si.

Considerando a data na qual o trabalho de Stuart Kurkowski foi publicado, mais de dez anos já se passaram, e os simuladores estão ainda mais populares nos dias de hoje.

Em Lessmann *et al.* (2008) existe um comparativo de uma série de simuladores com

base nos seguintes parâmetros: facilidade de instalação, implementação, documentação e interface com usuário (visualização e estatística). Este foi um estudo teórico, com base na experiência de 4 usuários. Também, em Köksal (2008) compara alguns simuladores de redes sem fio. Entre os parâmetros analisados, estão: Documentação, linguagem utilizada, modelos de simulação, interface e realismo.

Um ano após aos trabalhos citados no parágrafo acima, o trabalho Weingartner *et al.* (2009) propôs uma comparação entre simuladores utilizando os seguintes parâmetros: tempo de simulação e uso de memória. É importante observar que este é um dos únicos trabalhos analisados que realiza um teste de qualidade entre os diferentes simuladores. Em outras palavras, para que a avaliação e comparação possa ser feita, os resultados para o mesmo cenário devem ser próximos, como pode ser visto na figura 3.6. Além disso, as características dos simuladores, como tempo de simulação, são avaliadas de acordo com o tamanho da rede. Na figura 3.7, esta métrica pode ser observada.

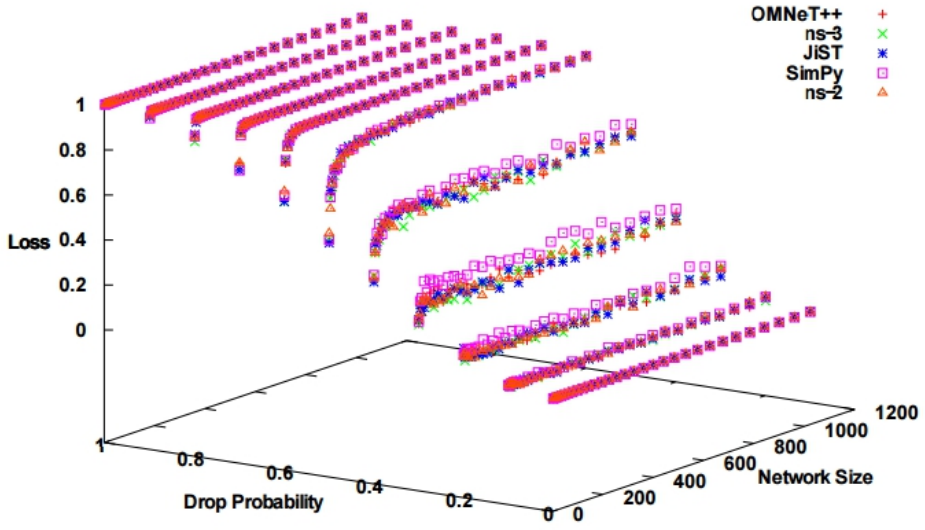

Figura 3.6: Comparação de resultados dos simuladores. (WEINGARTNER *et al.*, 2009).

Harsh Sundani publicou, em 2011, um trabalho no qual faz uma comparação entre simuladores de redes de sensores sem fio (SUNDANI *et al.*, 2011). Para analisar a performance de cada simulador escolhido, o autor usa como parâmetros as seguintes características: escalabilidade, nível de abstração, tempo de simulação e consumo de memória. Apesar de ter demonstrado que o Castalia não suporta simulação acima de 10000 nós sensores ao mesmo tempo e de citar como a maior limitação o fato de não ser especificamente uma plataforma de simulação de sensor. O autor concorda que o Castalia foi criado para

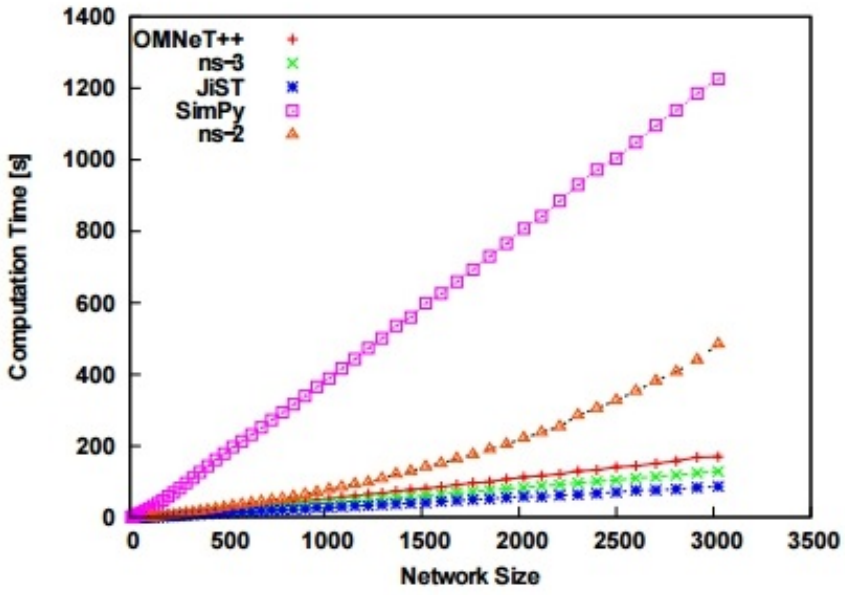

Figura 3.7: Comparação de performance dos simuladores. (WEINGARTNER *et al.*, 2009).

fornecer uma estrutura genérica confiável e realista para a validação de primeira ordem de um algoritmo antes de passar para a implementação de uma plataforma de algum sensor específico, e devido a este motivo é tão adotado em pesquisas, assim como nesse trabalho.

Em Bilalb *et al.* (2013), usa-se uma abordagem diferente do anterior. Nesse caso, são analisados parâmetros técnicos, como uso de memória, uso de CPU, escalabilidade (possibilidade de aumento de número de nós) e tempo de simulação. Em termos de credibilidade, parâmetros como estes, tendem a ser mais úteis, desde que o cenário seja muito bem especificado e repetido corretamente entre os diferentes simuladores. É importante observar que o autor, neste caso, explicou com detalhes qual era o cenário simulado, possibilitando a reprodutibilidade.

Já no trabalho de Nayyar e Singh (2015), compara-se os simuladores com base nos seguintes parâmetros: tipo de simulador (simulador genérico, simulador a nível de código e simulador a nível de *firmware*), tipo de licença, popularidade de uso e facilidade de programação. Considerando que este artigo trata de simuladores de redes de sensores sem fio, também é utilizado como parâmetro a presença de plataforma para simulação deste tipo específico de cenário.

3.5.1 SIMULADOR CASTALIA

O Castalia (BOULIS, 2009) é um simulador para RSSF e 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 e/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, pois é altamente paramétrico e pode simular uma ampla gama de plataformas.

O Castalia foi construído inteiramente sobre o OmNet++, importante *framework* para gerar eventos discretos e, devido a isso, ele é inteiramente modularizado e configurável, facilitando assim a importação e/ou implementação de protocolos.

Sua criação surgiu a partir de uma necessidade real de seus autores que precisavam testar padrões de comunicação em um projeto de RSSF. Eles desejavam modelos de canal e rádio precisos o suficiente para que os resultados das simulações fossem significativos e pudessem orientar a implementação real, só que os simuladores RSSF disponíveis estavam aquém da atual modelagem do estado da arte feita em redes de sensores. Especialmente na comunicação em que o impacto no resultado pode ser ainda mais significativo. Assim, usando modelos empíricos e dados reais, criaram o Castalia para atender estes anseios.

Como já dito anteriormente, a base do Castalia é OmNet++, para que o usuário consiga ter uma utilização plena e, principalmente, para que consiga realizar implementações de algoritmos e protocolos, é de extrema importância um conhecimento sobre o funcionamento do OmNet++. Uma maneira produtiva de conhecer mais sobre este *framework* é a realização dos tutoriais *hands-on* existentes no prórpio site do *framework*. A estrutura básica de módulos do Castalia é mostrada na figura 3.8.

Note que os nós não se conectam diretamente e sim através do canal sem-fio. As setas significam mensagens passando de um módulo para o outro. Quando um nó possui um pacote para enviar, ele vai para o canal sem-fio que irá decidir quais nós irão recebê-lo. Os nós também são vinculados por meio dos processos físicos que eles monitoram. Para cada processo físico, existe um módulo que contém a "verdade" da quantidade que o processo físico está representando. Os nós recuperam informações sobre o espaço e o tempo do processo físico (enviando uma mensagem ao módulo correspondente) para obter as leituras do sensor. Pode haver vários processos físicos representando os múltiplos

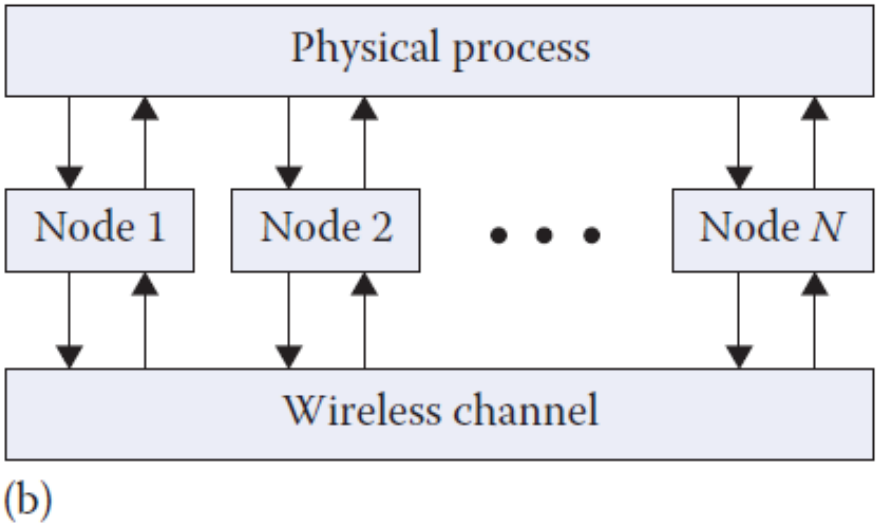

Figura 3.8: Estrutura básica dos módulos do Castalia (BOULIS, 2009).

dispositivos de detecção (múltiplas modalidades de detecção) que um nó possui.

O nó é composto por vários módulos. A próxima figura mostra a estrutura interna de um nó. As setas sólidas significam passagem de mensagem e as setas tracejadas significam chamadas de funções simples. Por exemplo, a maioria dos módulos chamam uma função do gerenciador de recursos para sinalizar que a energia foi consumida. O módulo Aplicativo é aquele que o usuário mais comumente 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 a mudança pelo usuário, geralmente, criando um novo módulo para implementar um novo protocolo ou um novo padrão de mobilidade, que é o grande foco desse trabalho. O Castalia oferece suporte para criar seus próprios protocolos ou aplicativos definindo classes abstratas apropriadas. Todos os módulos existentes são altamente sintonizáveis através de parâmetros.

Essa estrutura descrita acima é implementada em 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 (portas dentro e fora de portas) e uma possível estrutura de submódulo (se este for um módulo composto). Os arquivos com o sufixo .ned contêm 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

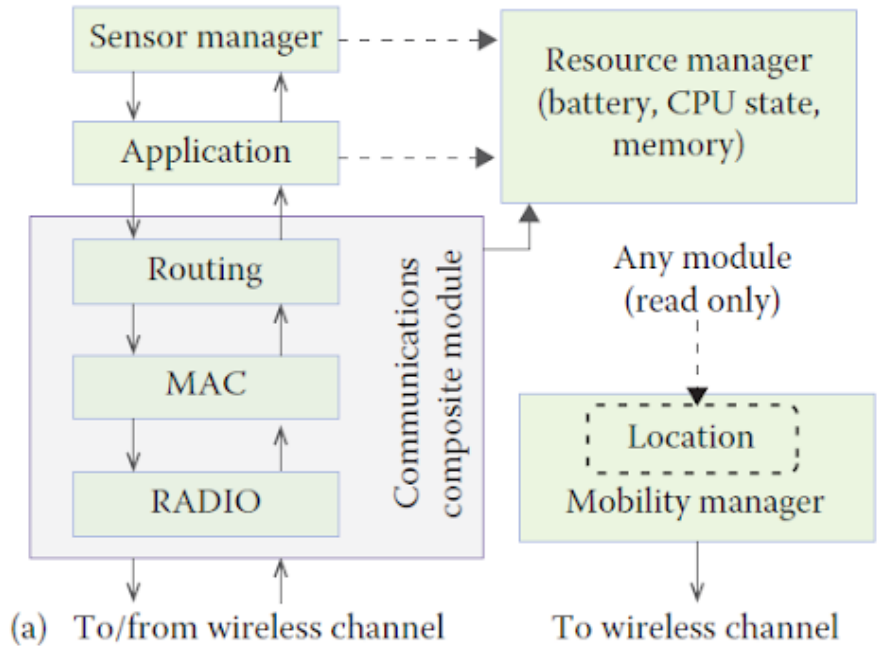

Figura 3.9: Módulos do Castalia (BOULIS, 2009).

composto, haverá subdiretórios para definir os submódulos. Se for um módulo simples, então há código C++ (arquivos .cc, .h) para definir seu comportamento. Essa hierarquia completa 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 sejam implementados).

Na pasta: /Castalia/bin encontra-se os executáveis para realizar as simulações, conforme descrito abaixo:

- **Castalia:** executa as simulações;
- **CastaliaResults:** interpreta os resultados provenientes do Castalia;
- **CastaliaPlot:** cria gráficos (utilizando o *GnuPlot*) utilizando as tabelas de saída resultantes do *CastaliaResults*. O manual oficial não menciona, mas é necessário a instalação do pacote *gnuplot* antes de utilizar o executável.

3.5.1.1 Módulo de Mobilidade dos Nós

Segundo Boulis (2009), o módulo de mobilidade, denominado *MobilityManager* especifica como os nós se movem pelo espaço. Ele mantém o estado de localização que outros módulos 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 encontrar que na maioria dos casos nada mudou). É muito melhor para o módulo de gerenciamento de mobilidade notificar o canal se algo mudou (por exemplo, ele poderia notificá-lo quando o nó acabou de alterar as células). No entanto, o cálculo de alterações de célula pode ser difícil e não precisa necessariamente fazer parte do módulo gerenciamento de mobilidade, portanto, para simplificar (mas ainda se manter eficiente) o módulo possibilita atualizações opcionais ou periódicas.

A princípio, sem nenhuma customização, esse módulo possui 3 parâmetros que devem ser derivados:

```
SN.node[*].MobilityManager.collectTraceInfo
SN.node[*].MobilityManager.updateInterval
SN.node[*].MobilityManagerName
```

Atualmente, o Castalia possui implementado apenas um módulo de padrão de mobilidade, o simples *LineMobilityManager*. O usuário apenas descreve o ponto de destino de um segmento de linha (o ponto de partida é a localização inicial do nó) e, portanto, define uma trajetória para o nó se mover para frente e para trás. O usuário também define a velocidade que o nó está se movendo.

Para implementar um novo modelo de mobilidade e manter a estrutura modular do Castalia deve-se seguir um padrão. A classe *VirtualMobilityManager* define uma base para qualquer módulo do gerenciador de mobilidade e fornece várias funções para ajudar nas operações do módulo. Todos os módulos do gerenciador de mobilidade devem ser derivados dessa classe base. No entanto, ao contrário das classes base de aplicativo de roteamento ou MAC, que já definem o método *handleMessage()* do OMNeT em um módulo do *MobilityManager*, o usuário deve definir esse método. Além disso, se for necessária uma inicialização ou finalização específica, as funções *initialize()* e *finishSpecific()* devem

ser redefinidas. Em outras palavras, aqui não há nenhuma camada extra de abstração com métodos como *startup()*, *fromYYlayer()* e *timerFiredCallback()*. No entanto, um método *initialize()* é definido na super classe que define operações comuns importantes, nesse caso, para definir inicialização própria, deve-se chamar o método da super classe também.

Sempre que o método *initialize()* é sobrescrito, deve-se chamar o método da superclasse primeiro, antes de entrar no código específico. Para finalizações específicas, apenas se faz necessário definir o método *finishSpecific()* (declarado como parte do *CastaliaModule*, que é a classe base do *VirtualMobilityManager*). Possivelmente, em versões futuras do Castalia, essa estrutura pode mudar e tornar o gerenciador de mobilidade virtual mais semelhante aos outros módulos virtuais.

Funções que são fornecidas a partir da classe base e podem ser chamadas a partir do código do módulo:

```
void notifyWirelessChannel()
```

Notifica o canal sem fio sobre a localização deste nó. Normalmente você precisa notificar periodicamente o canal sobre a posição do nó. Pode fazê-lo apenas chamando este método. Para criar o evento periódico, basta criar uma auto mensagem em *handleMessage()* que pode ser programada periodicamente ou outros padrões complexos podem ser criados, se necessário.

```
void setLocation(double x, double y, double z)
void setLocation(NodeLocationtype)
```

Esses métodos alteram a localização de um nó e notificam automaticamente o canal sem fio.

O local do nó atual pode ser acessado por meio da variável protegida *nodeLocation* do tipo *NodeLocationtype*.

4 MODELO PROPOSTO

Um modelo de mobilidade para redes móveis determina as posições do nó em qualquer instância do tempo de simulação e, portanto, influencia diretamente na topologia da rede e nas propriedades do canal e do *link*. Consequentemente, a precisão do modelo tem um grande impacto no desempenho da rede avaliada. Um modelo de mobilidade adequado para generalizar padrões de movimentação pedestre, mais especificamente animais em pasto, deve ser capaz de modelar estatisticamente os padrões de movimento corretos dos nós individuais, bem como o movimento que envolve o comportamento em grupo. Ao mesmo tempo, deve ser adaptável para vários cenários de aplicação nos quais os padrões de movimento, as atividades dos animais e o ambiente circundante podem ser diferentes. Primeiro será explicado a estrutura geral do modelo de mobilidade proposto, denominado *PastureMobilityManager*, e, em seguida, será apresentado os blocos constituintes em mais detalhes.

Conforme já foi citado, o objetivo principal é construir um modelo de mobilidade genérico e configurável para animais em pastagem que possa ser utilizado em simulações de RSSF. O modelo pode ser utilizado para simular a mobilidade de animais em pastagem, de acordo com o contexto e necessidade específica, sendo apenas necessário configurar o modelo conforme o que se deseja simular. E com o intuito de validar o conceito foi elaborado uma instância específica para gado leiteiro. Para criar essa instância específica, utilizou-se de informações coletadas na literatura científica e, assim, foi possível definir estados comportamentais que possam ser assumidas pelos animais. No decorrer desse capítulo, essa instância criada será utilizada com intuito de facilitar a explicação do modelo em voga.

4.1 ESTRUTURA DO MODELO

Durante a elaboração deste modelo, foi utilizado conceitos de alguns dos modelos de mobilidade considerados como estados da arte. O modelo RPGM (HONG *et al.*, 1999) é a base da estratégia de agrupamento dos animais. O RPGM foi estendido para atender as nuances específicas do cenário, introduzindo comportamentos que possuem parâmetros de mobilidade individuais. O *PastureMobilityManager* em si é constituído por duas unidades

básicas de controle, que são o seletor de comportamento e o módulo de transição entre as áreas de simulação. Por um lado, o processo do seletor de comportamento determina a velocidade do nó móvel, ou ausência da mesma (comportamentos estacionários), em qualquer momento. Já o módulo de transição irá determinar, através de matrizes markovianas, em qual área específica dentro do campo de simulação que um determinado grupo deve se encontrar. Ou seja, o processo de movimento global é responsável por controlar a mobilidade dos nós sensores de um determinado grupo como um todo. Conforme concluem Kwong *et al.* (2012), o rebanho pode se dividir em subgrupos independentes, cada um com sua própria dinâmica.

O padrão de mobilidade proposto começa carregando as informações contidas no arquivo *configFile.xml*, que será detalhado na seção 4.2, e constrói as matrizes markovianas de transição entre as áreas e entre os comportamentos para cada área. Após essa etapa de configuração das informações da simulação, é iniciado o processo de seleção da posição inicial do nó. A posição será selecionada utilizando uma distribuição uniforme randômica dentro de uma área pré determinada como entrada, simulando assim a entrada da pastagem.

Uma vez que a posição inicial foi informada ao Canal *Wireless*, o gerenciador de mobilidade proposto irá verificar se o nó em questão é um líder de grupo. Se sim, o gerenciador irá calcular através da matriz markoviana de transição de área se o grupo irá continuar na mesma área de simulação ou irá migrar para outra área. Se o nó não é um líder de grupo, o gerenciador verifica a qual grupo este nó pertence e qual Tipo de Área que o líder, consequentemente o grupo, esta frequentando no momento.

O próximo passo é verificar se irá ocorrer uma mudança de área (determinada pelo movimento global do grupo). Se sim, o nó sensor assume o comportamento "Andando"e calcula a velocidade de deslocamento, utilizando uma função normal randômica que recebe como entrada a velocidade mínima e a velocidade máxima do comportamento, e irá se mover até uma posição, também calculada randomicamente utilizando uma função de distribuição normal, dentro da nova área que será frequentada pelo grupo. Se não for ocorrer uma mudança de área, o nó irá verificar se irá ocorrer uma mudança de comportamento dentro da área. Nesse momento, a matriz markoviana de transição de comportamento irá determinar qual comportamento o nó deverá assumir.

Então, inicia-se uma nova fase. Se o comportamento definido for um comportamento

estacionário, deitado por exemplo, o gerenciador mantém a posição do nó sensor durante esse tempo de simulação em questão. No caso de uma postura móvel, andando por exemplo, o nó sensor irá se movimentar levando em consideração a estratégia de mobilidade escolhida e a velocidade relacionada ao comportamento específico. O comportamento da mobilidade, como o destino, a velocidade e o caminho para esse destino, depende da estratégia específica configurada para o cenário em execução. Foi implementado duas estratégias de mobilidade do nó individual: Gauss-Markov (LIANG; HAAS, 1999) e Random Waypoint Mobility Model (HYYTIÄ; VIRTAMO, 2007).

Quando o nó alcança o ponto de destino ou muda de um comportamento estacionário para um comportamento móvel, o gerenciador de mobilidade retorna para a fase de verificar se deve ocorrer uma mudança de Tipo 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 gerenciador de mobilidade disponibiliza um arquivo de *trace* para cada nó. Este arquivo, denominado trace_position_<node_index>.log no qual <*node_index*> é o identificador único do nó, conterá as posições, as coordenadas X e Y separados por um ";", do nó a cada *update_interval*, conforme demonstra a imagem 4.1. Ou seja, o arquivo irá disponibilizar cada passo do nó durante o tempo de simulação. Se os pesquisadores desejarem repetir o mesmo caminho dos nós na próxima simulação, deve-se apenas marcar o parâmetro $useTraceFiles$ como *true* no arquivo de configuração da simulação (omnetpp.ini). Este arquivo é extremamente importante para cumprir-se um dos objetivos desse projeto que é a padronização do caminho assumido pelos nós móveis possibilitando assim a comparação entre resultados de simulações diferentes. Por exemplo, comparar dois protocolos de roteamento diferentes. Outra função importante destes arquivos é gerar gráficos contendo o caminho dos nós. A imagem 4.2 mostra os arquivos disponibilizados ao final de uma simulação contendo 10 nós utilizando o gerenciador de mobilidade proposto.

Além dos arquivos de *trace* citados no parágrafo anterior, a simulação disponibiliza mais dois arquivos, conforme demonstra a imagem 4.2. Estes dois arquivos são disponibilizados para toda simulação realizada pelo Castalia. O primeiro arquivo, que recebe como nome a data e horário da simulação no seguinte formato YYMMDD-HHmmss.txt, é uma compilação final das informações coletadas durante a simulação. Alguns parâmetros importantes informados nesse arquivo são: quantidade de pacotes perdidos, energia consu-

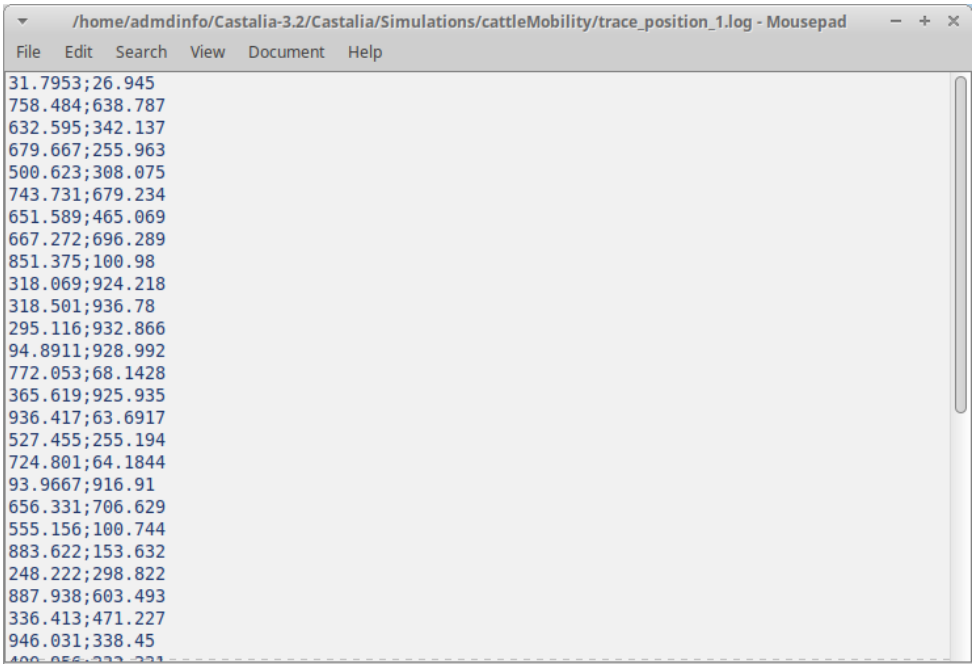

Figura 4.1: Exemplo de arquivo contendo o trace do nó 1.

mida, falhas com interfência, falhas sem interferência, quantidade de pacotes recebidos de cada nó e vários outros. Na imagem 4.2 pode-se ver dois exemplos deste tipo de arquivo: 180917-163352.txt e 180917-164305.txt. Já na figura 4.3 pode-se ver um exemplo deste aquivo e de como as informações coletadas durante a simulação ficam disponibilizadas. Conforme citado na sessão 3.5.1, o Castalia disponibiliza um programa para interpretar estes arquivos denominado *CastaliaResults*. A figura 4.4 exibe uma utilização simples deste programa.

O segundo arquivo, denominado Castalia-Trace.txt, disponibiliza informações sobre a execução da simulação. Este arquivo somente é gerado se o parâmetro global do gerenciamento de mobilidade genérico SN.node[*].MobilityManager.collectTraceInfo for setado igual a *true*. Este arquivo é bastante útil devido a grande dificuldade de realizar *debugger* durante a execução das simulações no Castalia. Foi essencial durante a fase de implementação do modelo proposto. Para escrever alguma entrada neste arquivo, basta utilizar a linha de código demonstrada na figura 4.5.

4.2 ARQUIVO DE CONFIGURAÇÃO

Conforme já foi mencionado na seção 3.5, toda simulação para ser executada no Castalia (ou até mesmo no OmNet++) deve possuir um arquivo de configuração denominado

Name	Size	Type	Date Modified
Trace Time 0.1	4,1 kB	folder	17/09/2018
180917-163352.txt	4,6 kB	plain text document	17/09/2018
180917-164305.txt	5,0 kB	plain text document	17/09/2018
Castalia-Trace.txt	496,2 kB	plain text document	17/09/2018
configCattleMobility.xml	3,0 kB	XML document	20/08/2018
omnetpp.ini	1,8 kB	plain text document	13/09/2018
trace_position_0.log	1,2 kB	application log	17/09/2018
trace_position_1.log	688 bytes	application log	17/09/2018
trace_position_2.log	906 bytes	application log	17/09/2018
trace_position_3.log	460 bytes	application log	17/09/2018
trace_position_4.log	509 bytes	application log	17/09/2018
trace_position_5.log	1,7 kB	application log	17/09/2018
trace_position_6.log	1,0 kB	application log	17/09/2018
trace_position_7.log	1,5 kB	application log	17/09/2018
trace_position_8.log	1,3 kB	application log	17/09/2018
trace_position_9.log	1,6 kB	application log	17/09/2018

Figura 4.2: Arquivos disponibilizados após simulação de 10 nós.

omnetpp.ini, no qual deve conter as configurações da simulação que o usuário deseja executar. Nesse arquivo deve-se configurar parâmetros como: tempo total de simulação, quantidade total de nós sensores, a aplicação que será simulada, os protocolos MAC, protocolos de roteamento e mais inúmeros parâmetros de todos os módulos existentes no Castalia. Um exemplo das possíveis configurações se encontra na Figura 5.8.

Com o intuito de tornar o modelo mais flexível e fácil no que tange configuração, os parâmetros específicos são preenchidos em um arquivo XML, dentro de um padrão pré estabelecido. O nome do arquivo é *configPastureMobility.xml* e ele deve ser carregado dentro do arquivo de configuração geral (omnetpp.ini) através do seguinte comando:

```
SN.node[0..9].MobilityManager.configFile =
    xmldoc("configPastureMobility.xml")
```

No comando acima, está sendo informado para o Castalia que os nós de 0 a 9 irão receber como parâmetro configFile no seu módulo de gerenciamento de mobilidade o XML *configPastureMobility.xml*. Perceba que se o usuário desejar configurar comportamentos diferentes para animais diferentes dentro de um mesmo rebanho, dentro de uma mesma simulação, é só criar um outro arquivo e especificar qual nó sensor irá ser configurado pelo outro arquivo. Isso pode ser extremamente útil para simular *outliers*, como por exemplo um animal doente ou que se encontre no cio.

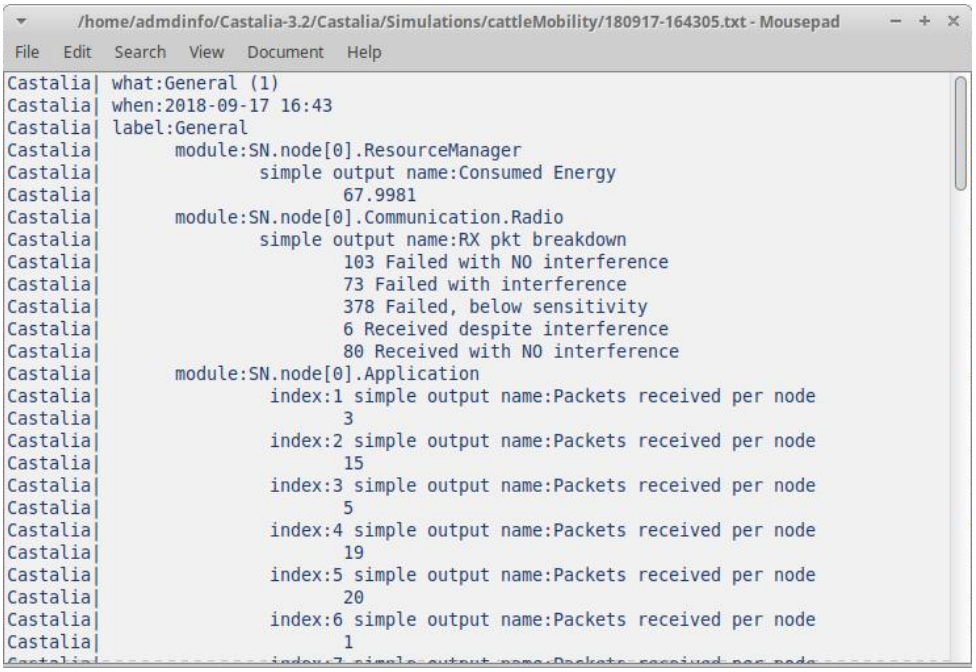

Figura 4.3: Arquivo 180917-164305.txt contendo o resultado da simulação.

Nos próximos parágrafos serão mostrados os parâmetros específicos que podem ser configurados no modelo proposto:

```
<initialParams areaTypeID="0" behaviorID="1"/>
```

A *tag inicialParams* possui dois parâmetros diferentes. O primeiro parâmetro, $areaTypeID$, informa ao gerenciador de mobilidade qual o tipo de área específica, dentro da área de simulação, que o nó sensor irá começar. Já o segundo, $behaviorID$ informa o comportamento inicial dentro da área do primeiro parâmetro. No exemplo acima, a simulação iniciaria na área com id igual a 0 e no comportamento igual a 1.

```
<leaderGroup percentValue="2"/>
```

Dentro do modelo proposto, a quantidade de líderes e, consequentemente, a quantidade de grupos existentes, pode ser definido de duas formas distintas. Pode ser informado um percentual de líderes que irão existir dentro do rebanho através da *tag leaderGroup* e do parâmetro $percentValue$, conforme exemplificado acima. Ao utilizar esta forma, o algoritmo do modelo irá realizar um calculo de quantos líderes existiram e randomicamente selecionar nós sensores para assumirem a liderança de seus grupos. No exemplo acima, apenas 2% dos nós sensores serão líderes, ou seja, se a quantidade total de nós for 100, irão existir 2 líderes e consequentemente 2 grupos de 50 nós. A segunda forma de indicar quantos grupos e quais nós são os líderes dos grupos é utilizando o arquivo de configuração

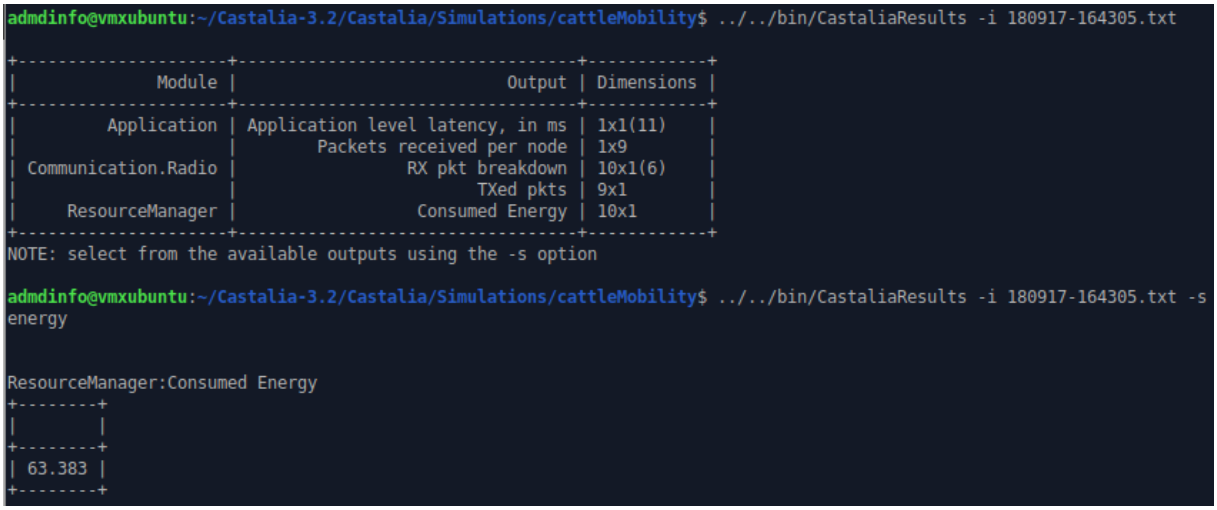

Figura 4.4: CastaliaResults interpretando o arquivo 180917-164305.txt.

```
trace() << "Localização alterada (x:y) para " << nodeLocation.x << ":" << nodeLocation.y;
```

Figura 4.5: Linha de código para escrever uma entrada no arquivo Castalia-Trace.txt.

geral da simulação (omnetpp.ini) através do parâmetro *groupLeaderIndex* do módulo de gerenciamento de mobilidade. No exemplo abaixo, os nós sensores de 0 a 4 irão ser liderados pelo nó 0 e os nós de 5 a 9 irão ser liderados pelo nó 5. Ou seja, dois grupos distintos existirão durante a simulação. Porém, vale salientar que o usuário pode configurar a quantidade de grupos que desejar e os grupos irão influenciar apenas no Tipo de Área que o indivíduo irá frequentar.

```
SN.node[0..4].MobilityManager.groupLeaderIndex = 0
SN.node[5..9].MobilityManager.groupLeaderIndex = 5
```

A primeira forma é mais aleatória e mais simples de configurar, mas será mais complexa se o usuário desejar reprodutibilidade da simulação. A segunda forma pode ser mais penosa para ser descrita se o usuário deseja um número grande de grupos, mas é completamente reprodutível após a primeira configuração realizada. Por este motivo, decidiu-se disponibilizar as duas maneiras de configuração.

```
<areaTransitionMatrix matrixID="0" walkingBehavior="1"/>
```

A *tag areaTransitionMatrix* determina, através do parâmetro *matrixID*, qual das matrizes markovianas (que também são descritas dentro deste arquivo) é a matriz de transição entre os tipos de áreas diferentes. Esta matriz que é utilizada pelos nós líderes para calcular as transições dentro do campo de simulação. Já o parâmetro *walkingBehavior* indica qual o *id* do comportamento "Andando", para ser utilizado quando o nó sensor

```
[General]
# ===========================================
# Always include the main Castalia.ini file
# ===========================================1
include ../Parameters/Castalia.ini

sim-time-limit = 1000s

SN.field_x = 1000   # meters
SN.field_y = 1000   # meters

SN.numNodes = 10

SN.wirelessChannel.onlyStaticNodes = false
SN.wirelessChannel.sigma = 0
SN.wirelessChannel.bidirectionalSigma = 0

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

SN.node[*].ApplicationName = "ThroughputTest"
SN.node[*].Application.packet_rate = 5
SN.node[*].Application.constantDataPayload = 2000

SN.node[*].MobilityManager.collectTraceInfo = true
```

Figura 4.6: Arquivo de configuração omnetpp.ini.

decide mudar de tipo de área dentro do campo de simulação. No exemplo acima, a matriz com o *id* igual a 0 é a matriz de transição entre as áreas.

```
<areaTypes>
        <areaType areaTypeID="3" name="Agua" xMin="0" yMin="900"
          zMin="0" xMax="500" yMax="1000" zMax="0" matrixID="1"/>
        <areaType areaTypeID="1" name="Cocho" xMin="500" yMin="0"
      zMin="0" xMax="1000" yMax="100" zMax="0" matrixID="2"/>
        <areaType areaTypeID="2" name="Sombra" xMin="500"
      yMin="250" zMin="0" xMax="1000" yMax="750" zMax="0" matrixID="3"/>
        <areaType areaTypeID="0" name="Pastagem" xMin="0" yMin="0"
      zMin="0" xMax="1000" yMax="1000" zMax="0" matrixID="4"/>
</areaTypes>
```

A *tag* composta *areaTypes* delimita cada tipo de área existente. Ela divide o campo de simulação em áreas específicas, nas quais os nós sensores terão comportamentos específicos. Para criar um Tipo de Área, utiliza-se a *tag areaType*, a qual possui os seguintes parâmetros: *areaTypeID* que indica o *id* ou identificador único que é utilizado para referenciar e correlacionar esse Tipo de Área durante a simulação; *name* que apenas serve

para nomear e para caracterizar as mudanças no arquivo de *trace* (pode ser utilizado também para rotular a área quando estiver utilizando simulações gráficas); os parâmetros de coordenadas mínimas $xMin$, $yMin$ e $zMin$ e os de coordenadas maximas $xMax$, $yMax$ e $zMax$ que delimitam o Tipo de Área dentro do campo de simulação; e por fim, mas não menos importante, o parâmetro $matrixID$ que indica qual é a matriz de comportamento correspondente à este Tipo de Área. Por exemplo, a primeira linha do exemplo acima delimita o tipo de área com *id* igual a 3 como espaço onde tem água na pastagem e informa que sua matriz de comportamento é de *id* igual a 1. O exemplo acima delimita o campo de simulação conforme representado na Figura 4.7.

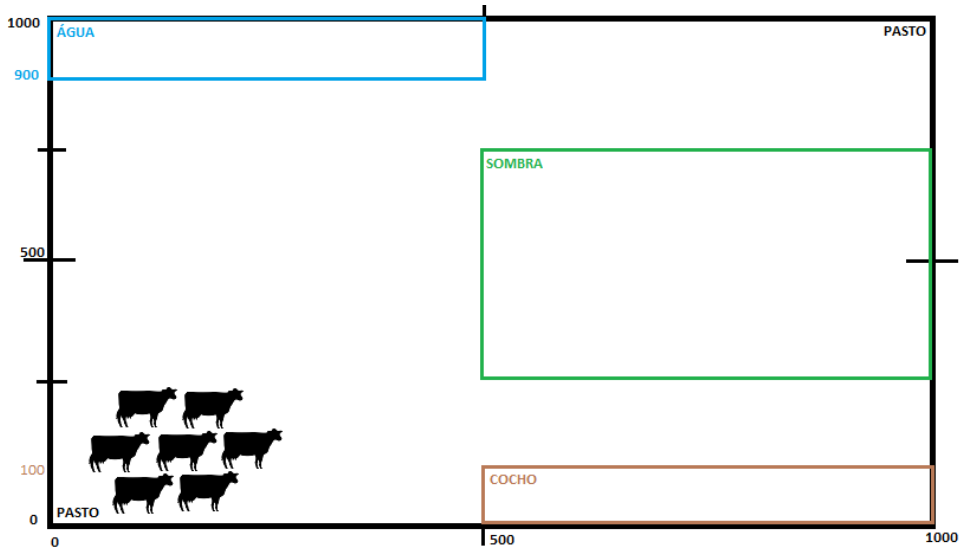

Figura 4.7: Divisão da pastagem em Tipos de Área do campo de simulação.

```
<behaviors>
    <behavior behaviorID="0" name="CORRENDO" minSpeed="5" maxSpeed="10"/>
    <behavior behaviorID="1" name="ANDANDO" minSpeed="2" maxSpeed="5"/>
    <behavior behaviorID="2" name="DEITADO" minSpeed="0" maxSpeed="0"/>
    <behavior behaviorID="3" name="ALIMENTANDO" minSpeed="1" maxSpeed="2"/>
</behaviors>
```

Outra *tag* composta que existe no modelo proposto é a *behaviors*. Ela lista os tipos de comportamentos existentes na simulação. Para configurar um comportamento, utiliza-se a *tag behavior* no interior da *tag* composta, contendo os parâmetros: $behaviorID$ que informa o *id* do comportamento e é utilizado dentro do algoritmo como valor de identificação; *name* que simplesmente nomeia o comportamento em questão; e os parâmetros *minSpeed* e *maxSpeed* que fornecem os valores da velocidade de deslocamento mínimo

e máximo, respectivamente. No exemplo acima, o usuário configurou na primeira linha um comportamento denominado Correndo e que possuí um *range* de velocidade de 5 a 10 metros por segundo. Apenas lembrando que os valores do exemplo são valores aproximados/deduzidos e foram selecionados para diferenciar os *ranges* de comportamento.

```
<markovMatrices>
        <markovMatrix matrixID="0">
            <row>
            <cell value="0.1" />
            <cell value="0.3" />
            <cell value="0.6" />
            <cell value="0.0" />
        </row>
            <row>
            <cell value="0.7" />
            <cell value="0.0" />
            <cell value="0.3" />
            <cell value="0.0" />
        </row>
            <row>
            <cell value="0.0" />
            <cell value="0.4" />
            <cell value="0.3" />
            <cell value="0.3" />
        </row>
            <row>
            <cell value="0.0" />
            <cell value="0.1" />
            <cell value="0.5" />
            <cell value="0.4" />
        </row>
        </markovMatrix>
</markovMatrices>
```

A *tag* que criam as matrizes markovianas ou matrizes de transição que serão utilizadas durante a simulação é *markovMatrices*. Essa *tag* é composta de *tags markovMatrix*

que possuem o parâmetro $matrixID$ que indentifica as matrizes e auxiliam quando é necessário refernciar e recuperar a matriz em questão. Por sua vez, essa *tag* é composta de um conjunto de *tags row* e *tags cell* que realmente montam as linhas e colunas, respectivamente, das matrizes. Lembrando que a matriz deve respeitar as características de uma matriz markoviana, conforme descrito na seção 3.4. O exemplo acima cria a seguinte matriz markoviana:

$$\begin{bmatrix} 0,1 & 0,3 & 0,6 & 0,0 \\ 0,7 & 0,0 & 0,3 & 0,0 \\ 0,0 & 0,4 & 0,3 & 0,3 \\ 0,0 & 0,1 & 0,5 & 0,4 \end{bmatrix}$$

5 RESULTADOS

Os resultados alcançados foram satisfatórios e dentro do esperado. A implementação do modelo na plataforma de simulação de RSSF denominada Castalia, representada de forma gráfica na figura 5.1, proporcionou uma mobilidade seguindo os princípios básicos de mobilidade e comportamento de animais em pastagem encontrado na literatura, conforme descrito no capítulo 2. Os nós se separam em grupos, de acordo com as configurações realizadas pelo usuário, se movimentando para o Tipo de Área escolhido pelo seu líder. Dentro destas áreas, cada indivíduo possui o próprio comportamento, mantendo assim a individualidade dos animais. Esse capítulo está dividido em 2 seções. A seção 5.1 descreve detalhes da implementação, dos principais métodos e parâmetros. Também apresenta alguns diagramas para clarificar um pouco mais o projeto. E finalizando, a seção 5.2 apresenta os resultados obtidos durante a validação realizada por meio de uma prova de conceito utilizando uma instância específica do modelo configurada para os bovinos.

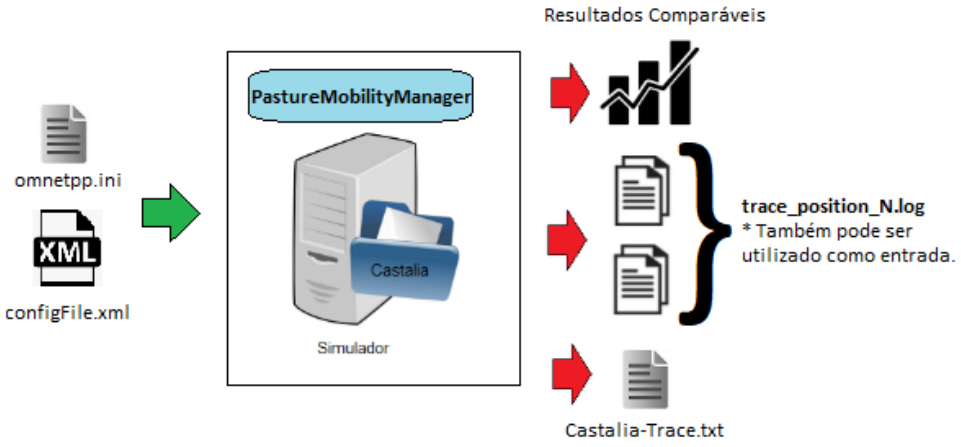

Figura 5.1: Implementação do modelo na plataforma de simulação Castalia.

5.1 IMPLEMENTAÇÃO

O módulo de gerenciamento de mobilidade foi implementado utilizando a linguagem C++ e a linguagem modularizada denominada NED utilizada pela plataforma OmNet++. A versão do Castalia escolhida, por se tratar de uma versão mais estável e frequentemente mais utilizada em experimentos científicos, foi a versão 3.2.

Foram criadas 4 classes durante a implementação, conforme pode-se visualizar no Diagrama de Classe apresentado na figura 5.2, sendo que *PastureMobilityManager* é a classe que realmente instancia o gerenciador de mobilidade dentro do Castalia. Para isso, deve-se setar o parâmetro SN.node[*].MobilityManagerName com o nome da classe, ou seja, *PastureMobilityManager*. A partir desse momento, o simulador irá entender que o tipo de mobilidade dos nós será a mobilidade de animais em pasto. As outras três classes foram criadas para auxiliar na recuperação e manipulação de informações durante a simulação, fazendo jus ao conceito de orientação a objetos. Foi criado a classe *Behavior*, que auxilia na recuperação e manipulação das informações sobre o comportamento dos animais, a classe *Transition*, que auxilia na recuperação e manipulação das informações referente às matrizes markovianas de transição e a classe *Coordinates* que auxilia na manipulação de coordenadas (posições dos nós móveis) no campo de simulação.

Tabela 5.1: Parâmetros Globais

Parâmetro	**Tpo**	**Valor Default**	**Obrigatório**
collectTraceInfo	boolean	false	NÃO
collectTraceRouteByNode	boolean	false	NÃO
readTraceRouteByNode	boolean	false	NÃO
updateInterval	double	0.1	SIM
areaMaxX	double	1000.0	SIM
areaMaxY	double	1000.0	SIM
margin	double	0.5	SIM
areaSelStrategy	int	0	SIM
groupLeaderIndex	int	0	NÃO
configFile	File	xmldoc(”configPastureMobility.xml”)	SIM
direction	double	1.0	NÃO
alpha	double	0.9	NÃO

A tabela 5.1 apresenta os parâmetros globais do modelo de mobilidade, ou seja, que devem ser configurados no arquivo de configuração global da simulação, o omnetpp.ini. Esses parâmetros são utilizados pela classe *PastureMobilityManager*. O parâmetro *collectTraceInfo* determina se a simulação irá coletar as informações e resultados referentes a execução da simulação. O *collectTraceRouteByNode* informa se a simulação irá gerar o arquivo de posicionamento individual de cada nó móvel ao final da simulação e o parâmetro *readTraceRouteByNode* indica que a simulação irá ler as posições dos arquivos de

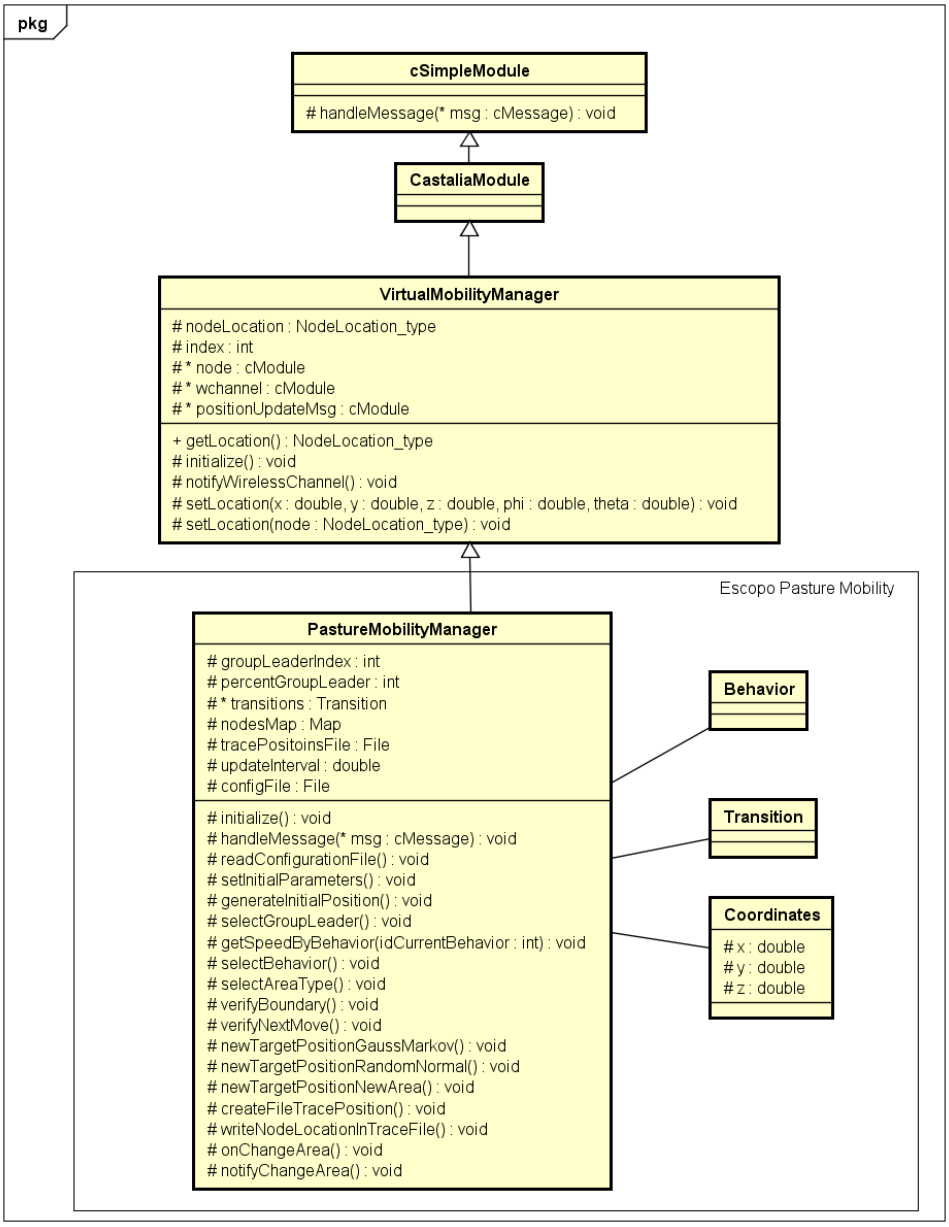

Figura 5.2: Diagrama de Classe

posicionamento individual do nó. O parâmetro *updateInterval* indica qual o intervalo de tempo que o módulo de mobilidade irá informar a posição do nó para o módulo do canal *wireless*. Os parâmetros *areaMaxX* e *areaMaxY* indicam, respectivamente, qual o valor máximo do eixo X e do eixo Y do campo de simulação. O parâmetro *margin* refere-se à margem de segurança existente para que um nó não ultrapasse os limites do campo de simulação. O parâmetro *areaSelStrategy* indica qual estratégia de seleção da posição dentro da área escolhida e é um *enumerator* que nessa primeira versão do gerenciador de mobilidade pode receber 2 valores: 0 e 1. Ao ser setado como 0, o gerenciador irá utilizar uma distribuição Normal randômica para escolha da nova localização dentro da

área e setar o valor 1 irá utilizar o modelo de mobilidade Gauss-Markov como estratégia de seleção. O parâmetro *groupLeaderIndex* indica qual o índice do nó líder do grupo ao qual o nó em questão pertence. O nó que possui o índice referente do parâmetro é quem irá decidir sobre as transições de áreas dentro do campo de simulação. Já o parâmetro *configFile* deve receber como valor um arquivo no formato XML conforme descrito na seção 4.2, considerado o parâmetro mais importante desse modelo de mobilidade devido a gama de configurações essenciais geradas por ele. Os últimos parâmetros listados, *direction* e *alpha* existem apenas para evitar erros de execução (*runtime*) da simulação e são necessários apenas se a estratégia de mobilidade interna do tipo de área escolhida for a Gauss-Markov.

Os principais métodos implementados na classe *PastureMobilityManager* se encontram listados na tabela 5.2 juntamente com uma descrição sucinta sobre o mesmo. Os métodos em **negrito** são métodos que devem ser sobrescritos toda vez que se implementa um módulo de mobilidade no Castalia. Alguns métodos existentes que apenas foram criados para organização do código foram desconsiderados na listagem referida.

Tabela 5.2: Métodos da classe *PastureMobilityManager*

Método	**Descrição**
initialize();	Executado apenas na instanciação da classe.
readConfigurationFile();	Lê o arquivo de configuração e instancia as classes auxiliares com os valores contidos no arquivo.
handleMessage(cMessage);	Executado a cade mensagem recebida pelo módulo de mobilidade. Onde ocorre todo o processamento.
setInitialParameters();	Parâmetros iniciais para a primeira movimentação.
getSpeedByBehavior(int);	Calcula a velocidade de acordo com o comportamento.
verifyBoundary();	Verifica se a próxima localização do nó não infringe os limites do campo de simulação.
verifyNextMove();	Realiza a verificação de como a próxima posição deve ser selecionada.
newTargetPositionGaussMarkov();	Calcula a próxima posição utilizando o modelo Gauss-Markov.
newTargetPositionNewArea();	Verifica se ocorrerá uma mudança de tipo de área.
newTargetPositionRNormal();	Calcula a próxima posição utilizando uma distribuição randômica normal.
notifyChangeArea(int);	O nó líder notifica os nós pertecentes ao seu grupo sobre a mudança do tipo de área.

Ao ser instanciada, a classe *PastureMobilityManager* irá executar o método *initialize()* e dentro desse método, toda configuração inicial deve ocorrer. Ou seja, os métodos *readConfigurationFile()* e *setInitialParameters()* serão executados. Após as configurações iniciais terem sido realizadas, o método irá chamar o método *scheduleAt()* que dispara uma mensagem para o canal sem fio e é recuperada pelo método *handleMessage(cMessage)*. Dentro deste método é onde toda a inteligência do gerenciador de mobilidade ocorre. Toda essa dinâmica citada pode ser melhor visualizada no diagrama de sequência exposto na figura 5.3. É nesse momento que o método *verifyNextMove()* é chamado sempre que necessário.

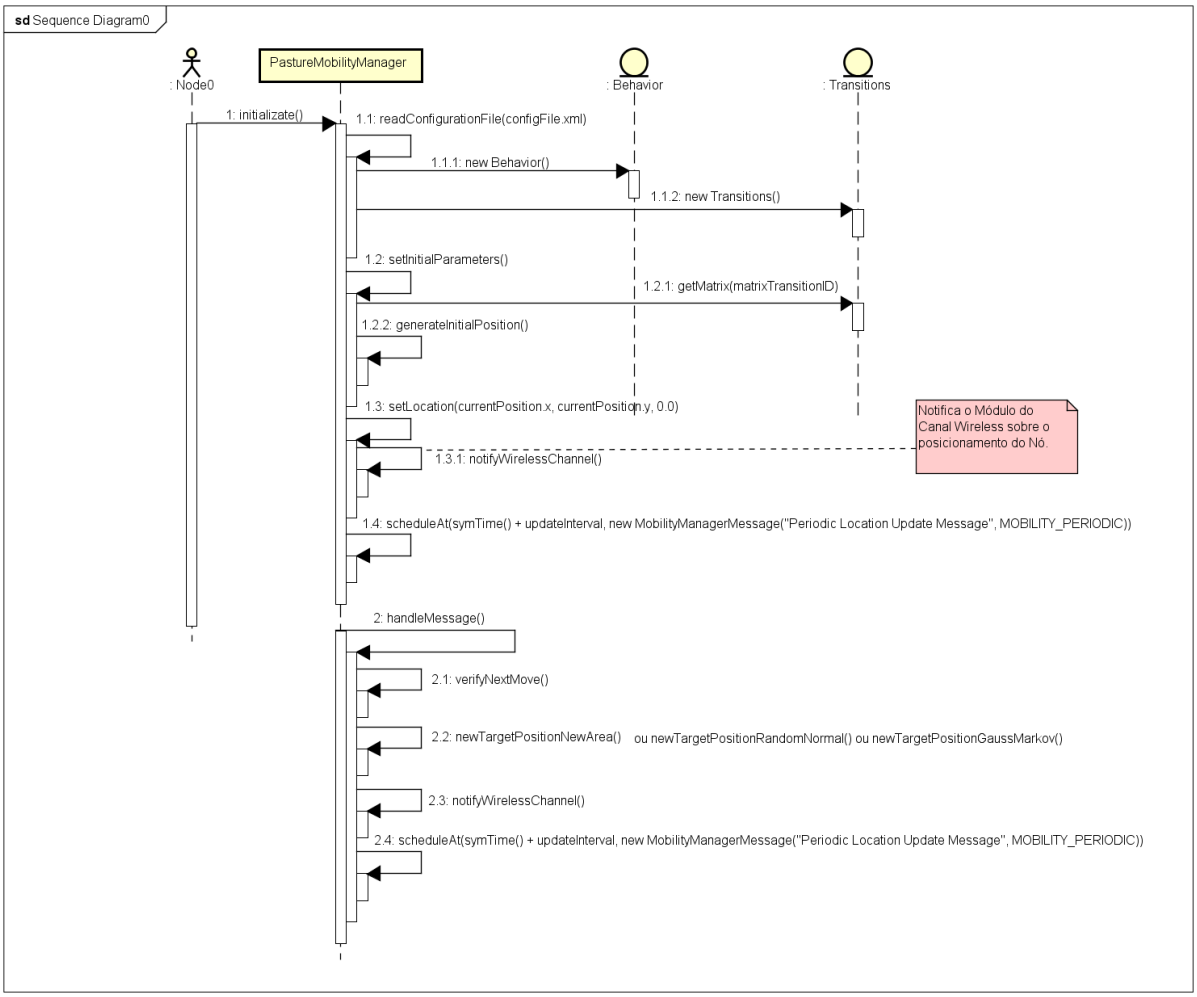

Figura 5.3: Diagrama de Sequência Macro

O método *verifyNextMove()*, como já dito, tem a função de calcular a nova posição do nó móvel. Verifica se o líder do grupo mudou de tipo de área. Se sim, irá chamar o método *newTargetPositionNewArea()* para que o nó possa realizar a mudança de área também para acompanhar o seu grupo. Se não, dependendo da estratégia escolhida pelo usuário,

irá chamar o método *newTargetPositionGaussMarkov()* ou *newTargetPositionRNormal()*.

Ao ser chamado, o método *newTargetPositionGaussMarkov()* irá calcular a nova posição do nó móvel conforme a figura 5.4. Toda vez que uma posição é encontrada, ocorre uma verificação se a mesma ultrapassa os limites do campo de simulação através de uma chamada do método *verifyBoundary()*. Este método utiliza a estratégia de refração para evitar que os nós móveis ultrapassem os limites estabelecidos.

```
422 /**
423  * Nova posição alvo segundo o modelo Gauss Markov de Movimentação
424  */
425 void PastureMobilityManager::newTargetPositionGaussMarkov()
426 {
427     trace() << "### Entrou no método newTargetPositionGaussMarkov() ###";
428
429     lastPosition.x = currentPosition.x;
430     lastPosition.y = currentPosition.y;
431
432     double speedOld = speed;
433     double directionOld = direction;
434
435     bool verifyBoundaryBool = true;
436
437     while (verifyBoundaryBool) {
438         //Equações para encontrar a nova velocidade e direção segundo o modelo Gauss-Markov
439         speed = alpha * speedOld
440                 + (1.0 - alpha) * speedMean
441                 + sqrt(1.0 - alpha * alpha) * normal(0.0, 1.0);
442         direction = alpha * directionOld
443                 + (1.0 - alpha) * directionMean
444                 + sqrt(1.0 - alpha * alpha) * normal(0.0, 1.0);
445
446         //Calculando a nova posição
447         currentPosition.x = lastPosition.x + speedOld * cos(directionOld);
448         currentPosition.y = lastPosition.y + speedOld * sin(directionOld);
449
450         verifyBoundaryBool = verifyBoundary();
451     }
452
453     //Calculando a distancia
454     distance = sqrt(pow(lastPosition.x - currentPosition.x , 2.0)
455             + pow(lastPosition.y - currentPosition.y, 2.0));
456
457     //Distancia dividido pela velocidade e dividido pelo intervalo de update
458     //para encontrar o quanto o nó anda em direção da nova localização
459     step = (distance / speedOld) / updateInterval;
460     distanceX = (currentPosition.x - lastPosition.x)/step;
461     distanceY = (currentPosition.y - lastPosition.y)/step;
462
463     //Primeiros passos
464     nodeLocation.x += distanceX;
465     nodeLocation.y += distanceY;
466
467     isNewDestination = false;
468     isWalkTime = true;
469 }
```

Figura 5.4: Método newTargetPositionGaussMarkov()

Já o método *newTargetPositionRNormal()* irá calcular a nova posição do nó móvel conforme a figura 5.5. Neste método, a estratégia para evitar coordenadas fora da área de simulação é simplesmente reexecutar o método até que retorne coordenadas dentro da área limite pré estabelecida.

```
384 /**
385  * Nova posição alvo segundo o modelo de Movimentação Normal Randômico
386  */
387 void PastureMobilityManager::newTargetPositionRNormal()
388 {
389     trace() << "### Entrou no método newTargetPositionRandomNormal() ###";
390
391     lastPosition.x = currentPosition.x;
392     lastPosition.y = currentPosition.y;
393
394     bool verifyBoundaryBool = true;
395
396     while (verifyBoundaryBool) {
397         //Calculando a nova posição randomicamente na área corrente
398         currentPosition = transitions->getRandomPosition(currentAreaTypeID, trace());
399
400         trace() << " Posição Randomica: X=" << currentPosition.x << ", Y=" << currentPosition.y;
401         //Verifica se a nova posição encontrada ultrapassou os limites
402         verifyBoundaryBool = !currentPosition.isInBoundary(limiteMin,limiteMax);
403     }
404     //Verifica se é para gravar o arquivo de log da rota dos nós
405     if (collectTraceRouteByNode) {
406         writeNodeLocationInTraceFile();
407     }
408
409     //Calculando a distancia
410     distance = sqrt(pow(lastPosition.x - currentPosition.x , 2)
411             + pow(lastPosition.y - currentPosition.y, 2));
412
413     //Distancia dividido pela velocidade e dividido pelo intervalo de update
414     //para encontrar o quanto o nó anda em direção da nova localização
415     step = (distance / speedMean) / updateInterval;
416
417     distanceX = (currentPosition.x - lastPosition.x)/step;
418     distanceY = (currentPosition.y - lastPosition.y)/step;
419 }
```

Figura 5.5: Método newTargetPositionRNormal()

5.2 INSTÂNCIA GADO BOVINO E PROVA DE CONCEITO

Para validação do modelo proposto e realização de alguns testes, uma intância específica foi criada como prova de conceito, utilizando valores aproximados ou induzidos atrelados aos conceitos de comportamento bovino coletados na literatura científica citados na seção 3.3.1. Como já mencionado anteriormente, a escolha desse grupo de animais deveu-se ao grande número de trabalhos e pesquisas existentes, principalmente relacionados a colares de monitoramento e rastreamento específicos para o controle dos bovinos, à importância econômica desse nicho no Brasil e à proximidade e parceria existente entre a Embrapa Gado de Leite, com sede em Juiz de Fora, e à Universidade Federal de Juiz de Fora.

Os valores da instância foram escolhidos ou estipulados de acordo com os valores e descrições encontrados nos vastos trabalhos científicos que abordam o assunto. Chegou-se às matrizes markovianas ou matrizes de transição com os valores expostos na figura 5.6.

A partir das informações encontradas no estado da arte na literatura científica sobre comportamento bovino em pastagem, gerou-se os valores para configurar o XML de configuração da instância para validação do modelo proposto. Utilizou-se principalmente as informações disponibilizadas pelos trabalhos (LOMBA, 2015) que apresentam como com-

ID = 3 COCHO

		0	1	2	3
ANDANDO	0	0,2	0,1	0,1	0,6
CORRENDO	1	0,6	0,4	0	0
DEITADO	2	0,2	0	0,2	0,6
ALIMENTANDO	3	0,2	0	0,2	0,6

ID = 1 PASTO

		0	1	2	3
ANDANDO	0	0,5	0,1	0,2	0,2
CORRENDO	1	0,6	0,4	0	0
DEITADO	2	0,2	0	0,5	0,3
ALIMENTANDO	3	0,2	0	0,3	0,5

ID = 4 SOMBRA

		0	1	2	3
ANDANDO	0	0,4	0,1	0,4	0,1
CORRENDO	1	0,6	0,4	0	0
DEITADO	2	0,2	0	0,6	0,2
ALIMENTANDO	3	0,2	0	0,5	0,3

ID = 2 ÁGUA

		0	1	2	3
ANDANDO	0	0,1	0,1	0,2	0,6
CORRENDO	1	0,6	0,4	0	0
DEITADO	2	0,2	0	0,4	0,4
ALIMENTANDO	3	0,1	0	0,3	0,6

Figura 5.6: Valores deduzidos para serem utilizados nas Matrizes de Transição do Comportamentos em cada Tipo de Área.

portamentos primários dos bovinos: Pastando/Procurando, Andando, Em Pé e Deitado; e do trabalho (GUO *et al.*, 2006) que realiza uma separação dos comportamentos em estacionário e móvel, além de citar o comportamento Correndo como um dos comportamentos móveis. Então, com intuito de gerar maior complexidade para os testes de uma RSSF e utilizando as informações encontradas nestes dois trabalhos, preferiu-se considerar os comportamentos: Andando, Deitado, Alimentando e Correndo.

Essa abstração de quatro comportamentos foi utilizada para que o modelo proposto possuísse 4 *ranges* de velocidade diferentes. Essas 4 possibilidades foram inferidas da seguinte forma. No trabalho (GUO *et al.*, 2006), o autor cita o comportamento Pastando (alimentando de grama) como sendo um comportamento estático e também um comportamento móvel, sendo colocado como comportamento secundário do comportamento em pé assim como do comportamento andando. Ele cita também os comportamentos secundários bebendo e ruminando (que pode ser dos dois tipos de comportamentos estacionários deitado ou em pé). No modelo proposto, decidiu-se agrupar todos esses comportamentos em um comportamento denominado alimentando que possui baixa mobilidade (menor do que o comportamento andando). E, para se manter coerente com o modelo disposto pelo autor, a probabilidade de ocorrer esse comportamento é maior nos tipos de área Água, Pastagem (suprindo assim a necessidade de um comportamento denominado pastando) e Cocho. Outro conceito interessante utilizado do trabalho (GUO *et al.*, 2006) foi a divisão

entre comportamentos estacionários e móveis. A imagem 5.7 demonstra a máquina de estado para as transições entre os Tipos de Área.

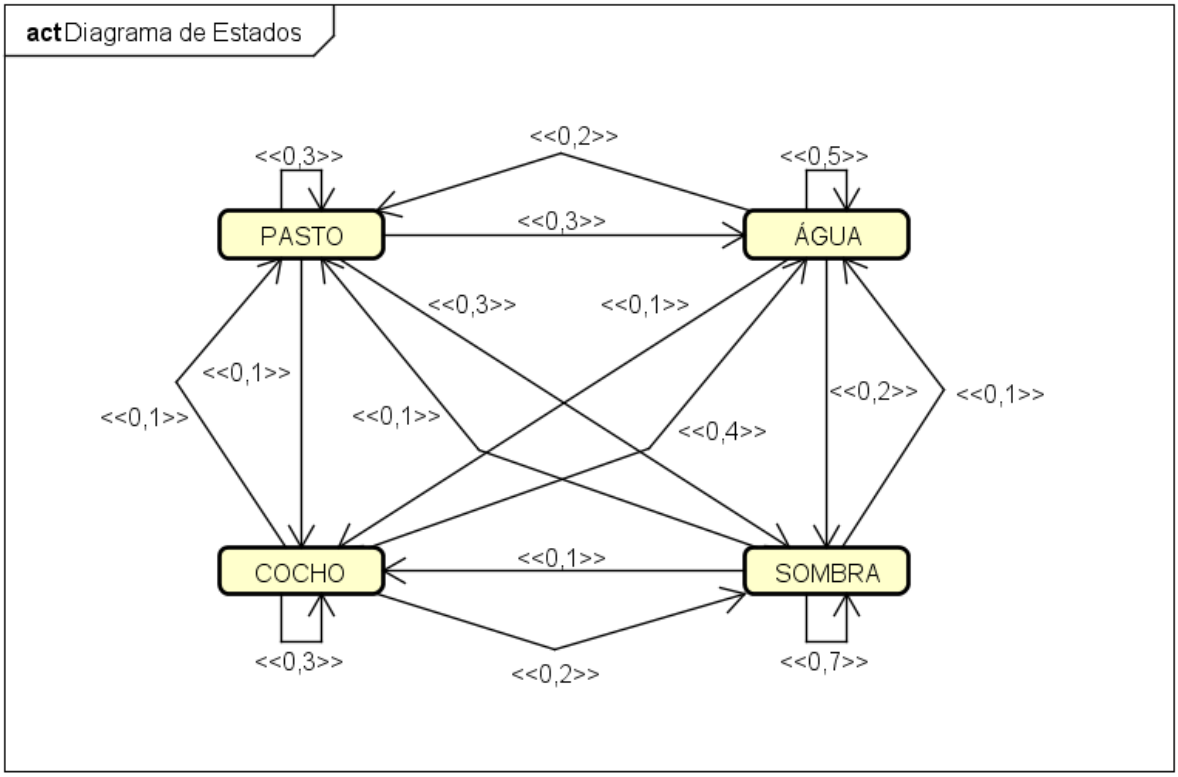

Figura 5.7: Máquina de estado para as transições entre os Tipos de Área.

O comportamento ”Andando”proposto em ambos trabalhos supra citados também foi considerado no modelo proposto. O comportamento ”Parado”foi utilizado para agrupar os comportamentos estacionários ”Deitado”e ”Em Pé”, pois em ambos tanto a velocidade mínima e máxima são iguais a zero. Seria possível criar ambos os comportamentos no modelo, mas não traria melhorias para o objetivo desse trabalho, visto que não criaria uma maior proximidade à realidade de mobilidade de animais a pasto. O comportamento ”Correndo”, utilizado no trabalho (GUO *et al.*, 2006) também foi somado ao modelo por acrescentar uma nuance diferente, possibilitando cenários com uma complexidade interessante para os testes. Além de ser um comportamento importante para refletir anomalias enfrentadas pelos animais. Por ter sido citado em poucos trabalhos, considerou-se um comportamento raro de ocorrer e então a probabilidade deste em todos os Tipos de Áreas é baixíssimo.

Dessa forma, chegou-se aos 4 comportamentos utilizados na prova de conceito para representar a mobilidade de gado bovino. Os valores de transições entre esses compor-

tamentos são diferentes em cada Tipo de Área existente dentro do campo de simulação. Por exemplo, conforme descrito nos trabalhos (OLIVEIRA, 2013) e (KILGOUR *et al.*, 2012), a probabilidade dos animais procurarem por sombra é bastante alta com o objetivo de reduzir a temperatura corporal e, ao estarem nesse ambiente, possuem uma alta probabilidade de comportamentos estáticos.

O comportamento alimentando, por exemplo, possui probabilidade alta nos Tipos de Áreas Água (0,6%) e Cocho (0,6%), pois esse comportamento é um agrupamento dos comportamentos Bebendo, Pastando, Ruminando e Lambendo Minerais (Sal por exemplo).

Assim como na transição de um comportamento para o outro, as transições entre os Tipos de Áreas também foram deduzidas de acordo com as informações encontradas na literatura. Ou seja, deve-se considerar o tipo de área Água com uma grande probabilidade de ser procurado pelos bovinos em pastagem, conforme sugere o trabalho (DEGASPERI *et al.*, 2003), no qual o autor descreve que durante parte significativa do tempo em que os bovinos estão em pastagem, geralmente, procuram sombra, bebedouros, poças e cursos d'água para se refrescarem e se livrarem de insetos. Então, por exemplo, foi considerada a probabilidade do gado quando se encontra na sombra permanecer na sombra de 0,7%, e do Cocho migrar para a Água de 0,4%, sendo essa maior probabilidade de todas, pois após lamber sal e se alimentar de ração o gado costuma procurar por água. Outro exemplo de dedução realizada para estipular os valores de transição entre os Tipos de Área foi que ao se encontrar na Água, o animal tende a permanecer nessa área, configurada com uma probabilidade de 0,5%.

Lembrando que por se tratar de uma plataforma flexível, pode-se acrescentar quantos comportamentos e tipos de áreas que forem necessários para se aproximar ao máximo do comportamento real do animal desejado. O XML com as configurações finais de comportamentos bovinos a pasto se encontra no Anexo 7.

5.2.1 EXECUÇÃO DA PROVA DE CONCEITO E RESULTADOS

Executou-se uma simulação contendo 10 nós móveis, sendo os 5 primeiros nós (índices de 0 a 4) liderados pelo nó de índice 0 e os 5 últimos (índices de 5 a 9) liderados pelo nó de índice igual a 5. O gerenciador de mobilidade de todos os nós é o *PastureMobilityManager*. O arquivo de configuração global utilizado foi o contido na imagem 5.8.

As configurações genéricas ficaram da seguinte forma: o tempo limite de simulação foi

```
[General]
# ============================================
# Always include the main Castalia.ini file
# ============================================1
include ../Parameters/Castalia.ini

sim-time-limit = 1000s

SN.field_x = 1000   # meters
SN.field_y = 1000   # meters

SN.numNodes = 10

SN.wirelessChannel.onlyStaticNodes = false
SN.wirelessChannel.sigma = 0
SN.wirelessChannel.bidirectionalSigma = 0

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

SN.node[*].ApplicationName = "ThroughputTest"
SN.node[*].Application.packet_rate = 5
SN.node[*].Application.constantDataPayload = 2000

SN.node[*].MobilityManager.collectTraceInfo = true
SN.node[*].MobilityManager.collectTraceRouteByNode = true
SN.node[*].MobilityManager.areaSelStrategy = 0

SN.node[0..9].MobilityManagerName = "PastureMobilityManager"
SN.node[0..9].MobilityManager.updateInterval = 100
SN.node[0..9].MobilityManager.configFile = xmldoc("configCattleMobility.xml")
SN.node[0..4].MobilityManager.groupLeaderIndex = 0
SN.node[5..9].MobilityManager.groupLeaderIndex = 5
```

Figura 5.8: Arquivo de configuração global da prova de conceito.

configurado com 10000 s, o campo de simulação foi de 10000 m^2 e a quantidade total de nós igual a 10. As configurações utilizadas no módulo do canal sem fio foram as padrões, ou seja: a *flag* que determina se a simulação terá apenas nós estáticos foi configurada como falsa, o valor de sigma e do sigma bidirecional foram setados como 0.

Como nessa prova de conceito o principal objetivo é verificar se os nós durante a simulação estão seguindo o modelo proposto, utilizou-se nos módulos de comunicação e aplicação a configuração padrão do *ThroughputTest*. Essa simulação vem como exemplo na instalação básica do Castalia na versão 3.2.

Já o módulo Gerenciamento de Mobilidade foi configurado da seguinte forma. Primeiramente, configuramos a simulação para utilizar o gerenciamento de mobilidade proposto por este trabalho. Essa configuração é realizada setando o parâmetro MobilityManagerName com o nome do gerenciador implementado, ou seja, igual a PastureMobilityManager. Logo após, deve-se dizer para aplicação quais arquivos de *log* e trace devem ser gerados. Nessa prova de conceito, desejou-se gravar tanto o arquivo de *tracer* padrão

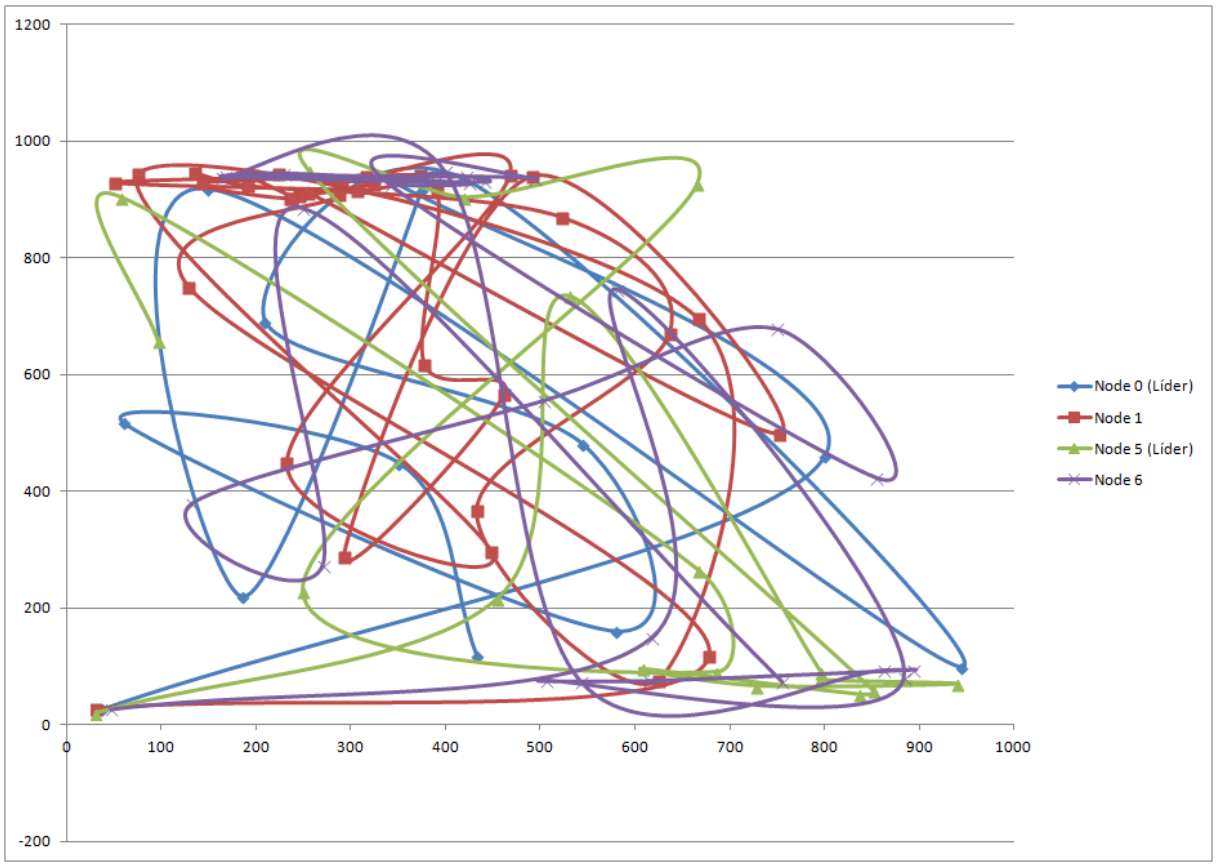

Figura 5.9: Gráfico contendo a trajetória dos nós da amostragem.

do simulador Castalia, para fins de informação sobre o que ocorreu durante a execução, como também os arquivos contendo os posicionamentos utilizados por cada nó. Para isso, deve-se setar o parâmetro *collectTraceInfo* igual a *true* e o parâmetro *collectTraceRouteByNode* igual a *true*, respectivamente. Escolheu-se utilizar a estratégia de seleção de comportamento igual a distribuição randômica normal, ou seja, o parâmetro *areaSelStrategy* recebeu o valor 0. Além disso, todos os nós receberam o mesmo valor para o parâmetro *updateInterval* igual 100 e o mesmo arquivo de configuração, contendo as informações comportamentais dos bovinos em pastagem, ao se setar o parâmetro *configFile* igual a xmldoc("configCattleMobility.xml"). Os nós de 0 a 4 foram atribuídos ao grupo do líder de índice 0 e os nós de 5 a 9 ao grupo do líder de índice 5, ao configurar-se o parâmetro *groupLeaderIndex*.

Como já dito, o intuito era verificar se o grupo se mantinha frequentando o mesmo tipo de área na maior parte do tempo de simulação, mantendo o individualismo no comportamento assumido dentro de cada tipo de área. Dentre os 10 nós móveis utilizados na simulação, para facilitar a visualização em gráficos, selecionou-se apenas dois nós móveis

de cada grupo. Os representantes do grupo 1 são os nós 0 e 1 e os representantes do grupo 2 são os nós 5 e 6. As coordenadas de posicionamento coletadas equivalem a cada posição diferente assumida pelo nó durante o tempo de simulação. Os passos, que equivalem ao tempo total de simulação dividido pelo intervalo de *update*, e também são os locais notificados para os módulos do Castalia não foram considerados devido a impossibilidade de demonstrar isso em gráficos e por não influenciar no resultado esperado. A figura 5.9 demonstra a trajetória completa dos 4 nós da amostragem selecionada.

Conforme podemos ver no gráfico 5.10, o nó móvel 1 segue o nó móvel 0, o líder de seu grupo durante as mudanças de tipo de área. Na figura 5.11, que mostra o gráfico de posicionamento de 2 nós (índices 5 e 6) do grupo 2, também percebe-se essa tendência de liderança.

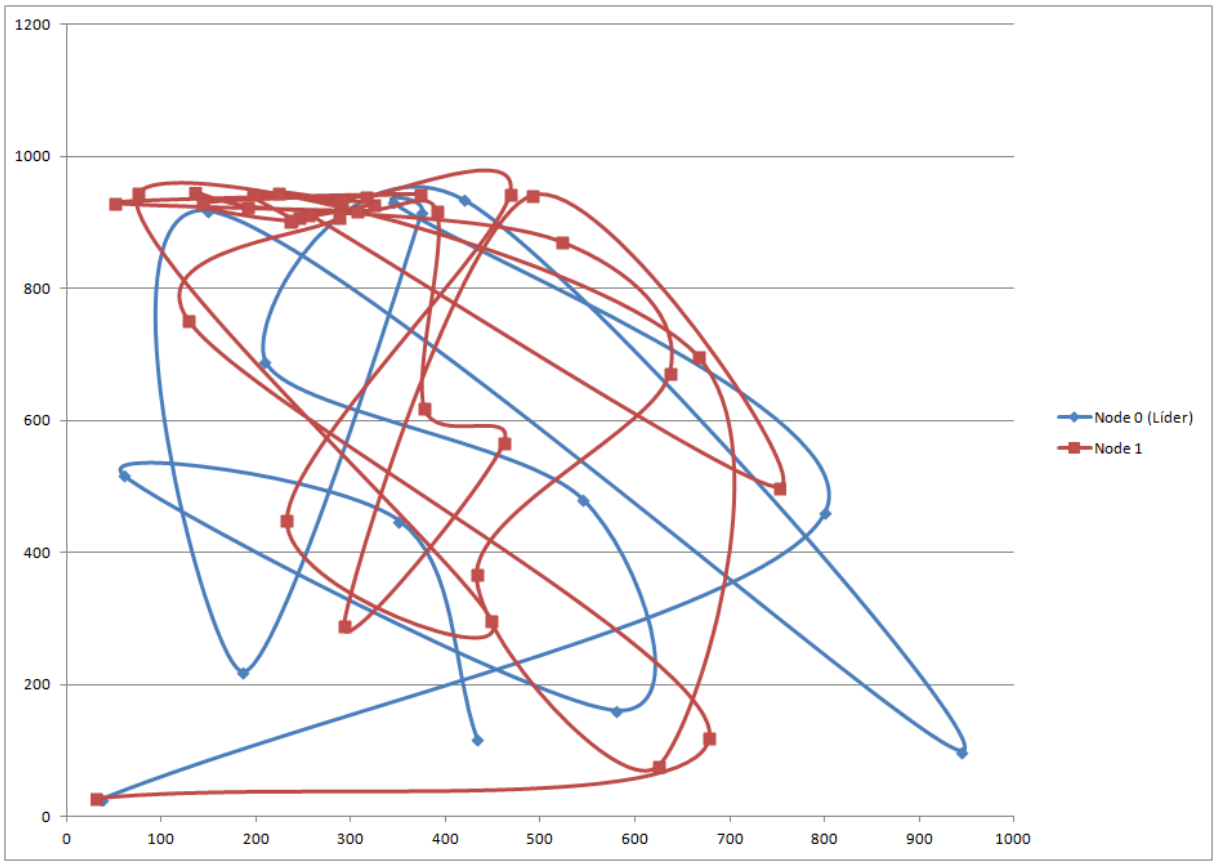

Figura 5.10: Gráfico de posicionamento do grupo 1.

Interessante notar também a alta concentração do posicionamento dos nós em tipo de áreas tidas como mais frequentadas, como a ÁGUA E COCHO. Isso se dá ao fato de termos configurado estas duas áreas como as mais prováveis, ou em outras palavras, com maior probabilidade de serem frequentadas pelos nós móveis, que neste contexto

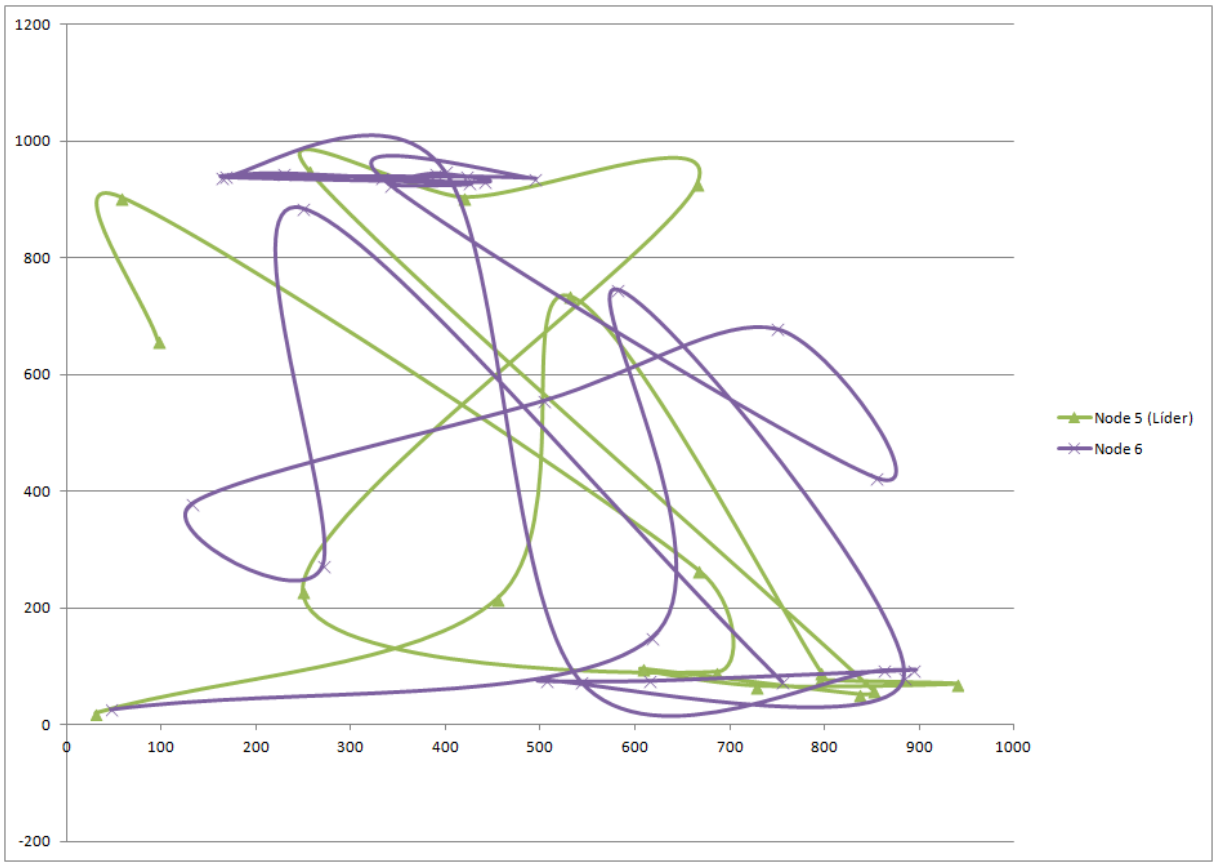

Figura 5.11: Gráfico de posicionamento do grupo 2.

estão representando bovinos em pastagem. Demarcou-se os tipos de áreas no gráficos apresentados acima para facilitar essa visualização.

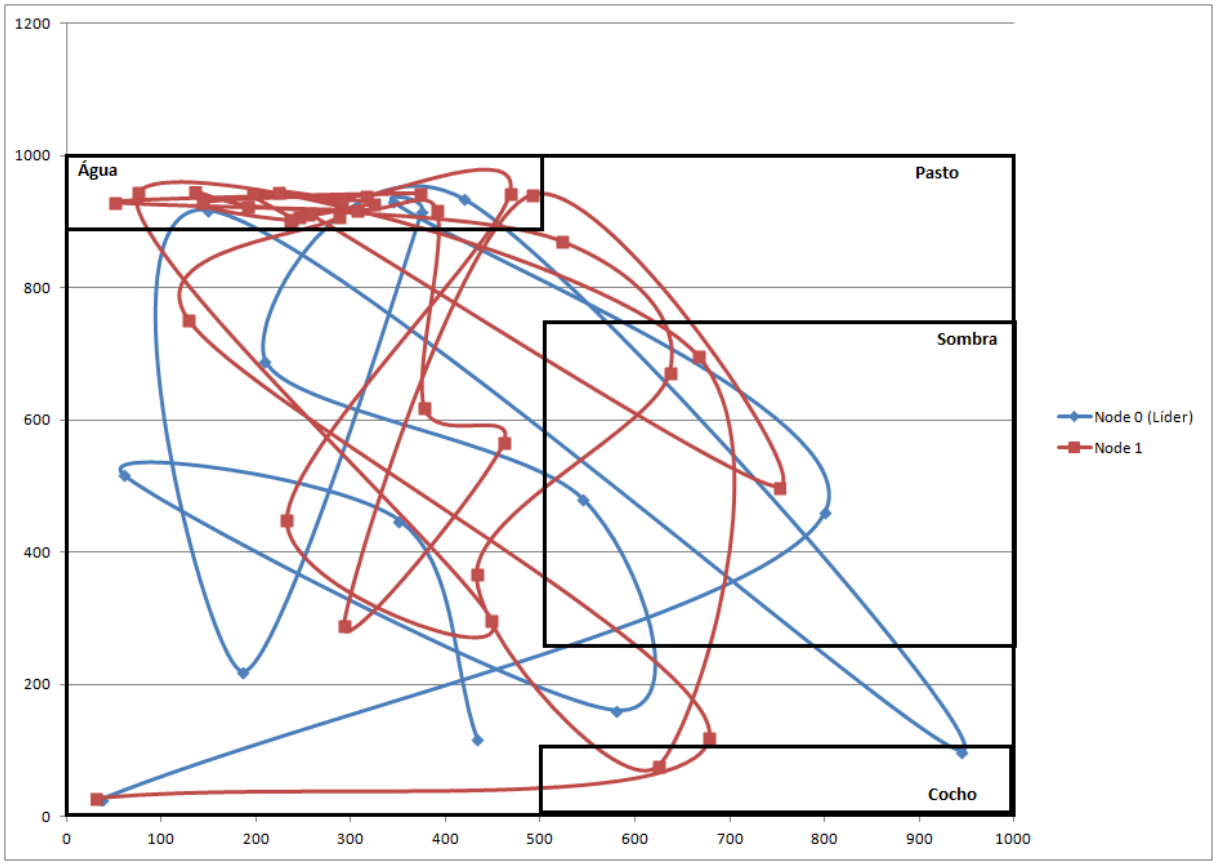

Figura 5.12: Gráfico de posicionamento do grupo 1 com Tipos de Áreas.

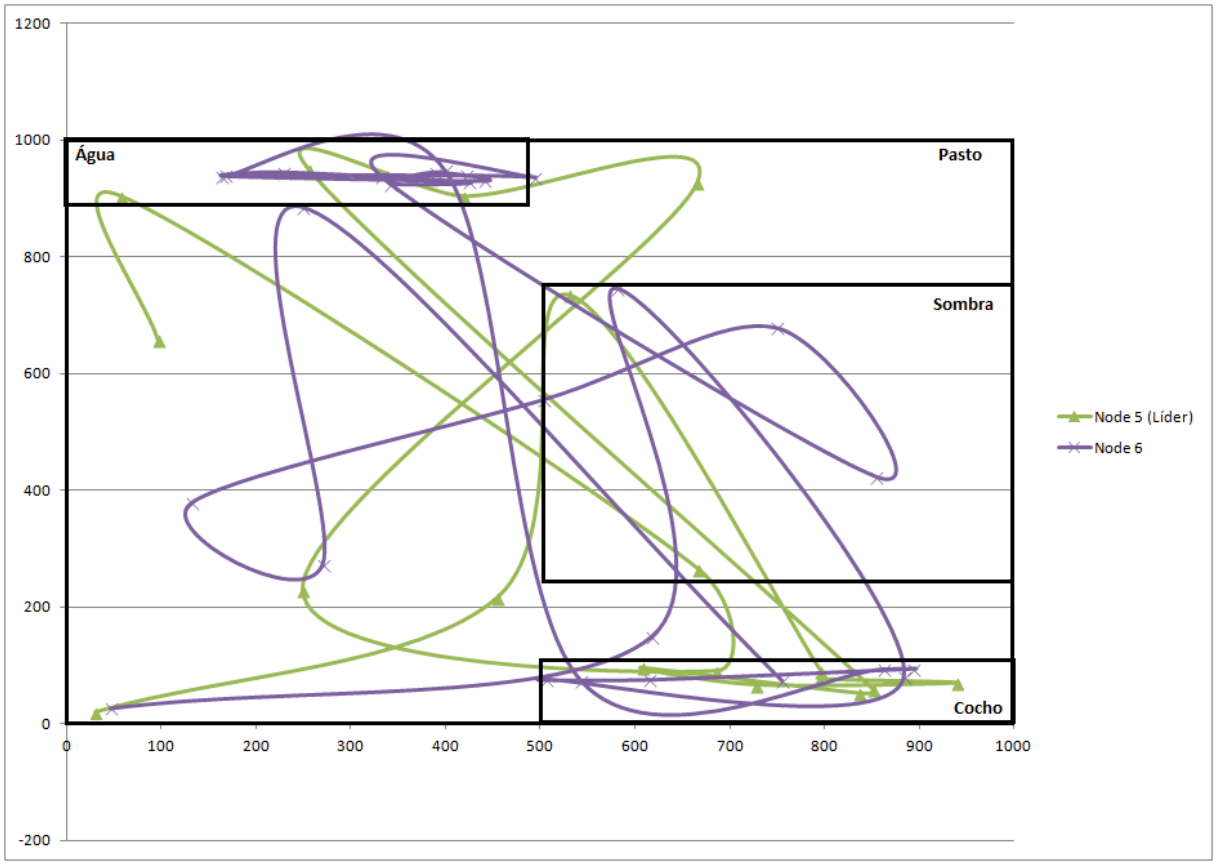

Figura 5.13: Gráfico de posicionamento do grupo 2 com Tipos de Áreas.

6 CONCLUSÕES

Este trabalho apresenta o modelo de mobilidade denominado *PastureManagerMobility* que possibilita a simulação de animais em pastagem, auxiliando assim projetos científicos que desejam trabalhar dentro deste contexto a se aproximar da realidade. Conforme já mencionado, simular novos protocolos ou até mesmo novos *hardware* utilizando modelos que se aproximam do mundo real antes de implementá-los, proporciona economia e eficiência aos projetos e aumentam a acurácia dos resultados das simulações, mitigando o risco de encontra erros e falsos resultados.

Desde o início, o intuito era proporcionar uma mobilidade aos nós durante as simulações próxima ao que acontece na realidade. Mobilidade essa contendo comportamentos em grupo e nuances individuais, sem paradas abruptas ou mudanças repentinas de direção e com memória para que o caminho fosse contínuo e realístico. E, desta forma, criando assim todo um ambiente de desafios que uma RSSF enfrenta ao ser implantada. Ao analisar os resultados obtidos pela prova de conceito, percebeu-se que todas essas características citadas foram atendidas pelo modelo de maneira satisfatória. Logo, pode-se concluir que o trabalho proposto conseguiu atender os objetivos estabelecidos.

Além do modelo de mobilidade, entende-se que o trabalho proporcionou outras contribuições relevantes. Uma delas é a implementação realizada para um dos simuladores mais populares no âmbito de RSSF, o Castalia versão 3.2. Esta implementação que se encontra disponibilizada no sistema de controle de versões distribuído *BitBucket* pode ser utilizada como um padrão de mobilidade dentro de pesquisas com finalidades semelhantes, possibilitando assim comparações entre os resultados destas pesquisas e a reprodutibilidade dos resultados alcançados.

Outra contribuição é a instância do modelo configurada para simulação específica de bovinos em pastagem, que é embasada em informações coletadas no estado da arte de artigos e trabalhos científicos referentes ao comportamento bovino, pode ser utilizada para facilitar simulações de RSSF para o monitoramento e rastreamento de bovinos. Os arquivos *trace* contendo os caminhos assumidos pelos nós móveis obtidos com a execução realizada como prova do conceito também podem contribuir e facilitar simulações que possuem objetivos semelhantes com a realizada por este trabalho. Estes arquivos podem

ser adotados como padrões.

Por se tratar de um assunto e uma área (RSSF) que ainda não havia sido abordada dentro do Programa de Pós-Graduação de Ciências da Computação da Universidade Federal de Juiz de Fora, pode-se considerar como uma contribuição secundária, um efeito colateral desse trabalho, a criação de um novo ramo de pesquisa dentro do Programa de Pós-Graduação de Ciências da Computação da Universidade Federal de Juiz de Fora, mais especificamente, dentro do LAPIC. Podendo assim, novos alunos darem continuidade ao trabalho, realizando evoluções ou novas pesquisas no novo ramo criado. Se pensar mais especificamente na área de monitoramento e rastreamento de animais, pode-se vislumbrar um estreitamento dos laços entre a Embrapa e a UFJF, podendo assim aumentar a contribuição entre as duas instituições com foco em pesquisas inovadoras e, até mesmo, em geração de novas patentes para o mercado.

6.1 TRABALHOS FUTUROS

O modelo proposto é passível de várias evoluções para atender outras necessidades de pesquisadores que desejam simular o monitoramento e rastreamento de animais em pastagem. Um exemplo dessas evoluções seria a possibilidade do gerenciador de mobilidade gerar *outliers*. Ou seja, simular nós sensores móveis que se desprendam dos grupos ou que comecem a se comportar com certas anomalias. Isso seria bastante útil para pesquisas que desejam simular padrões de doenças ou cio dos animais, para verificar como esses casos influenciam na topologia e nas características do link.

Outra evolução interessante é implementar uma correlação com o tempo, ou seja, uma correlação de três níveis (o nosso modelo possuí apenas dois níveis), que seria Área x Comportamento x Tempo. Essa evolução se mostra necessária pois em vários artigos, os pesquisadores informam a grande diferença de comportamento dos animais entre os períodos diurno e noturno. Isto é, pode-se implementar matrizes markovianas para representar cada período.

Pode-se citar também como trabalho futuro a possibilidade do modelo considerar a terceira dimensão (Z) no posicionamento comportamental dos nós sensores. Alguns comportamentos somente se diferem no eixo z, como por exemplo em pé parado ou deitado. Essa diferenciação pode ser bastante útil para simular alterações comportamentais que podem indicar doenças nos animais.

Outra possibilidade interessante seria criar aplicações específicas para simular os sensores utilizados em colares para monitoramento e rastreamento de animais em pastagem. Essas aplicações devem ser desenvolvidas em outro módulo do Castalia, o módulo de aplicações, 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. Ou, até mesmo, a elevação da temperatura influenciar na decisão do gerenciador de mobilidade para onde o nó sensor deve se deslocar. Esta parte dos sensores se mostra bastante interessante para aproximar ainda mais o modelo da realidade.

Algumas outras possibilidade de extensões que podem ser realizadas a partir deste trabalho são: implementar a possibilidade do Cocho ser móvel dentro da área de pastagem; fazer com que o modelo passe a considerar o relevo existente na pastagem; e passar a considerar o cansaço e gasto energético do animal (muito importante principalmente quando se simula rebanhos nos quais deseja-se otimizar o quanto de energia o animal gasta durante a pastagem).

E, por fim, a realização de testes comparativos entre o gerenciador de mobilidade proposto/implementado e outros gerenciadores com o intuito de comparar os parâmetros de redes disponibilizados pelo Castalia ao fim de suas simulações, como: consumo de energia, *throughput*, perda de pacotes, entre outros.

REFERÊNCIAS

AKYILDIZ, I. F.; SU, W.; SANKARASUBRAMANIAM, Y.; CAYIRCI, E. A survey on sensor networks. **IEEE communications magazine**, IEEE, v. 40, n. 8, p. 102–114, 2002.

ANDERSON, D. M.; ESTELL, R. E.; CIBILS, A. F. Spatiotemporal cattle data—a plea for protocol standardization. **Positioning**, Scientific Research Publishing, v. 4, n. 01, p. 115, 2013.

BILALB, S. M.; OTHMANA, M. *et al.* A performance comparison of network simulators for wireless networks. **arXiv preprint arXiv:1307.4129**, 2013.

BOULIS, A. Castalia: revealing pitfalls in designing distributed algorithms in wsn. In: ACM. **Proceedings of the 5th international conference on Embedded networked sensor systems**, 2007. p. 407–408.

BOULIS, A. Castalia user manual. **Online: http://castalia. npc. nicta. com. au/pdfs/Castalia-User Manual. pdf**, 2009.

BRESLAU, L.; ESTRIN, D.; FALL, K.; FLOYD, S.; HEIDEMANN, J.; HELMY, A.; HUANG, P.; MCCANNE, S.; VARADHAN, K.; XU, Y. *et al.* Advances in network simulation. **Computer**, IEEE, v. 33, n. 5, p. 59–67, 2000.

CAMP, T.; BOLENG, J.; DAVIES, V. A survey of mobility models for ad hoc network research. **Wireless communications and mobile computing**, Wiley Online Library, v. 2, n. 5, p. 483–502, 2002.

CAMPOS, C. A. V.; MORAES, L. F. M. de. Modelos markovianos de mobilidade individual para redes móveis ad hoc. **Anais do XXI Simpósio Brasileiro de Redes de Computadores (SBRC2003)**, p. 135–150, 2003.

CARVALHO, P. C. d. F.; TRINDADE, J. K. d.; MEZZALIRA, J. C.; POLI, C. H. E. C.; NABINGER, C.; GENRO, T. C. M.; GONDA, H. L. 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.

CECCHIN, D. **Comportamento de vacas leiteiras confinadas em free stall com camas de areia e borracha**. Dissertao (Mestrado), 2012.

CHIANG, C.-C.; GERLA, M. **Wireless network multicasting**. Tese (Doutorado) — UCLA, 1998.

CLUTTON-BROCK, J. **A natural history of domesticated mammals**, 1999.

DEGASPERI, S. A. R.; COIMBRA, C. H.; PIMPÃO, C. T. *et al.* Estudo do comportamento do gado holandês em sistema de semi-confinamento. **Revista Acadêmica: Ciência Animal**, v. 1, n. 4, p. 41–47, 2003.

ERIKSSON, J.; ÖSTERLIND, F.; FINNE, N.; TSIFTES, N.; DUNKELS, A.; VOIGT, T.; SAUTER, R.; MARRÓN, P. J. Cooja/mspsim: interoperability testing for wireless sensor networks. In: ICST (INSTITUTE FOR COMPUTER SCIENCES, SOCIAL-INFORMATICS AND TELECOMMUNICATIONS ENGINEERING). **Proceedings of the 2nd International Conference on Simulation Tools and Techniques**, 2009. p. 27.

GONZÁLEZ, L.; BISHOP-HURLEY, G.; HANDCOCK, R. N.; CROSSMAN, C. Behavioral classification of data from collars containing motion sensors in grazing cattle. **Computers and Electronics in Agriculture**, Elsevier, v. 110, p. 91–102, 2015.

GUO, Y.; CORKE, P.; POULTON, G.; WARK, T.; BISHOP-HURLEY, G.; SWAIN, D. Animal behaviour understanding using wireless sensor networks. In: IEEE. **Local Computer Networks, Proceedings 2006 31st IEEE Conference on**, 2006. p. 607–614.

HALL, S. J. Chillingham cattle: social and maintenance behaviour in an ungulate that breeds all year round. **Animal Behaviour**, Elsevier, v. 38, n. 2, p. 215–225, 1989.

HAMMOODI, I.; STEWART, B.; KOCIAN, A.; MCMEEKIN, S. A comprehensive performance study of opnet modeler for zigbee wireless sensor networks. In: IEEE. **2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies**, 2009. p. 357–362.

HANCOCK, J. Studies of grazing behaviour in relation to grassland management i. variations in grazing habits of dairy cattle. **The Journal of Agricultural Science**, Cambridge University Press, v. 44, n. 4, p. 420–433, 1954.

HENDERSON, T. R.; LACAGE, M.; RILEY, G. F.; DOWELL, C.; KOPENA, J. Network simulations with the ns-3 simulator. **SIGCOMM demonstration**, v. 14, n. 14, p. 527, 2008.

HERBEL, C. H.; NELSON, A. B. Activities of hereford and santa gertrudis cattle on a southern new mexico range. **Journal of Range Management**, JSTOR, p. 173–176, 1966.

HONG, X.; GERLA, M.; PEI, G.; CHIANG, C.-C. A group mobility model for ad hoc wireless networks. In: ACM. **Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems**, 1999. p. 53–60.

HUANG, Y.; WAN, Y.; REN, Y.; YAN, B.; LUAN, Z.; QIAN, D. Reducing communication overhead in threshold monitoring with arithmetic aggregation. In: IEEE. **Networking, Architecture, and Storage, 2009. NAS 2009. IEEE International Conference on**, 2009. p. 347–353.

HUIRCÁN, J. I.; MUÑOZ, C.; YOUNG, H.; DOSSOW, L. V.; BUSTOS, J.; VIVALLO, G.; TONEATTI, M. Zigbee-based wireless sensor network localization for cattle monitoring in grazing fields. **Computers and Electronics in Agriculture**, Elsevier, v. 74, n. 2, p. 258–264, 2010.

HYYTIÄ, E.; VIRTAMO, J. Random waypoint mobility model in cellular networks. **Wireless Networks**, Springer, v. 13, n. 2, p. 177–188, 2007.

IBGE, I. B. de Geografia e E. **Plano Tabular - Pesquisa da Pecuária Municipal - PPM**. 2018. Disponível em: <https://biblioteca.ibge.gov.br/visualizacao/periodicos/84/ppm$_2$017$_v$45$_b$r$_i$n$formativo.pdf$>.

J.D.CARTHY. **O Estudo do Comportamento**, 1969.

JESUS, L. d.; PIRES, P. Identificação do comportamento bovino por meio do monitoramento animal. In: IN: SIMPÓSIO BRASILEIRO DE PECUÁRIA DE PRESIÇÃO

APLICADA À BOVINOCUTURA DE CORTE, 1., 2014, CAMPO GRANDE. ANAIS... CAMPO GRANDE: EMBRAPA GADO DE CORTE, 2014. 1 P. **Embrapa Gado de Corte-Resumo em anais de congresso (ALICE)**, 2014.

KILGOUR, R. J. In pursuit of "normal": A review of the behaviour of cattle at pasture. **Applied Animal Behaviour Science**, Elsevier, v. 138, n. 1-2, p. 1–11, 2012.

KILGOUR, R. J.; UETAKE, K.; ISHIWATA, T.; MELVILLE, G. J. The behaviour of beef cattle at pasture. **Applied Animal Behaviour Science**, Elsevier, v. 138, n. 1-2, p. 12–17, 2012.

KÖKSAL, M. A survey of network simulators supporting wireless networks. **En línea: http://www. ceng. metu. edu. tr/˜ e1595354/A% 20Survey% 20of% 20Network% 20Simulators% 20Supporting% 20Wireless% 20Networks. pdf**, 2008.

KÖPKE, A.; SWIGULSKI, M.; WESSEL, K.; WILLKOMM, D.; HANEVELD, P.; PARKER, T. E.; VISSER, O. W.; LICHTE, H. S.; VALENTIN, S. Simulating wireless and mobile networks in omnet++ the mixim vision. In: ICST (INSTITUTE FOR COMPUTER SCIENCES, SOCIAL-INFORMATICS AND TELECOMMUNICATIONS ENGINEERING). **Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops**, 2008. p. 71.

KURKOWSKI, S.; CAMP, T.; COLAGROSSO, M. Manet simulation studies: the incredibles. **ACM SIGMOBILE Mobile Computing and Communications Review**, ACM, v. 9, n. 4, p. 50–61, 2005.

KWONG, K. H.; WU, T.-T.; GOH, H. G.; SASLOGLOU, K.; STEPHEN, B.; GLOVER, I.; SHEN, C.; DU, W.; MICHIE, C.; ANDONOVIC, I. Practical considerations for wireless sensor networks in cattle monitoring applications. **Computers and Electronics in Agriculture**, Elsevier, v. 81, p. 33–44, 2012.

LANGENDOEN, K.; MEIER, A. Analyzing mac protocols for low data-rate applications. **ACM Transactions on Sensor Networks (TOSN)**, ACM, v. 7, n. 2, p. 19, 2010.

LESSMANN, J.; JANACIK, P.; LACHEV, L.; ORFANUS, D. Comparative study of wireless network simulators. In: IEEE. **Networking, 2008. ICN 2008. Seventh International Conference on**, 2008. p. 517–523.

LI, J.; SERPEN, G. Tossim simulation of wireless sensor network serving as hardware platform for hopfield neural net configured for max independent set. **Procedia Computer Science**, Elsevier, v. 6, p. 408–412, 2011.

LIANG, B.; HAAS, Z. J. Predictive distance-based mobility management for pcs networks. In: IEEE. **INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE**, 1999. v. 3, p. 1377–1384.

LOMBA, L. F. D. **Identificação do comportamento bovino a partir dos dados de movimentação e do posicionamento do animal**. Dissertao (Mestrado), 2015.

LOUREIRO, A. A.; NOGUEIRA, J. M. S.; RUIZ, L. B.; NAKAMURA, E. F.; FIGUEIREDO, C. M. S. Redes de sensores sem fio. **Simpósio Brasileiro de Redes de Computadores (SBRC)**, p. 179–226, May 2003.

MARTISKAINEN, P.; JÄRVINEN, M.; SKÖN, J.-P.; TIIRIKAINEN, J.; KOLEHMAINEN, M.; MONONEN, J. Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines. **Applied animal behaviour science**, Elsevier, v. 119, n. 1-2, p. 32–38, 2009.

MEI, A.; STEFA, J. Swim: A simple model to generate small mobile worlds. In: IEEE. **INFOCOM 2009, IEEE**, 2009. p. 2106–2113.

MINAKOV, I.; PASSERONE, R.; RIZZARDI, A.; SICARI, S. A comparative study of recent wireless sensor network simulators. **ACM Transactions on Sensor Networks (TOSN)**, ACM, v. 12, n. 3, p. 20, 2016.

MOTTOLA, L.; PICCO, G. P.; CERIOTTI, M.; GUNă, Ş.; MURPHY, A. L. Not all wireless sensor networks are created equal: A comparative study on tunnels. **ACM Transactions on Sensor Networks (TOSN)**, ACM, v. 7, n. 2, p. 15, 2010.

NABI, M.; GEILEN, M.; BASTEN, T. Moban: A configurable mobility model for wireless body area networks. In: ICST (INSTITUTE FOR COMPUTER SCIENCES, SOCIAL-INFORMATICS AND TELECOMMUNICATIONS ENGINEERING). **Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques**, 2011. p. 168–177.

NACER, J. F. R. **Aspectos de Roteamento em Redes de Sensores sem Fio para Monitoramento Bovino**. Dissertao (Mestrado), 2017.

NADIMI, E. S.; SØGAARD, H.; BAK, T.; OUDSHOORN, F. W. Zigbee-based wireless sensor networks for monitoring animal presence and pasture time in a strip of new grass. **Computers and electronics in agriculture**, Elsevier, v. 61, n. 2, p. 79–87, 2008.

NAYYAR, A.; SINGH, R. A comprehensive review of simulation tools for wireless sensor networks (wsns). **Journal of Wireless Networking and Communications**, Scientific & Academic Publishing, v. 5, n. 1, p. 19–47, 2015.

NKWARI, P. K. M.; RIMER, S.; PAUL, B.; FERREIRA, H. Heterogeneous wireless network based on wi-fi and zigbee for cattle monitoring. In: IEEE. **IST-Africa Conference, 2015**, 2015. p. 1–9.

OLIVEIRA, M. T. P. d. **Análise comportamental de bovinos baseada em trajetórias semânticas aplicada à pecuária de precisão**. Dissertao (Mestrado), 2013.

ROYER, E. M.; MELLIAR-SMITH, P. M.; MOSER, L. E. An analysis of the optimum node density for ad hoc mobile networks. In: IEEE. **Communications, 2001. ICC 2001. IEEE International Conference on**, 2001. v. 3, p. 857–861.

SÁNCHEZ, M.; MANZONI, P. Anejos: a java based simulator for ad hoc networks. **Future generation computer systems**, Elsevier, v. 17, n. 5, p. 573–583, 2001.

SCHEIBE, K. M.; GROMANN, C. Application testing of a new three-dimensional acceleration measuring system with wireless data transfer (was) for behavior analysis. **Behavior research methods**, Springer, v. 38, n. 3, p. 427–433, 2006.

SCHINDELHAUER, C. Mobility in wireless networks. In: SPRINGER. **International Conference on Current Trends in Theory and Practice of Computer Science**, 2006. p. 100–116.

SCHÜTZ, K. E.; ROGERS, A. R.; COX, N. R.; TUCKER, C. B. Dairy cows prefer shade that offers greater protection against solar radiation in summer: Shade use, behaviour, and body temperature. **Applied Animal Behaviour Science**, Elsevier, v. 116, n. 1, p. 28–34, 2009.

SUNDANI, H.; LI, H.; DEVABHAKTUNI, V.; ALAM, M.; BHATTACHARYA, P. Wireless sensor network simulators a survey and comparisons. **International Journal of Computer Networks**, v. 2, n. 5, p. 249–265, 2011.

VARGA, A. **INET Framework for the OMNeT++ Discrete Event Simulator**. 2012.

WATANABE, S.; IZAWA, M.; KATO, A.; ROPERT-COUDERT, Y.; NAITO, Y. A new technique for monitoring the detailed behaviour of terrestrial animals: a case study with the domestic cat. **Applied Animal Behaviour Science**, Elsevier, v. 94, n. 1-2, p. 117–131, 2005.

WEINGARTNER, E.; LEHN, H. V.; WEHRLE, K. A performance comparison of recent network simulators. In: IEEE. **Communications, 2009. ICC'09. IEEE International Conference on**, 2009. p. 1–5.

WIETRZYK, B. **Practical mobile ad hoc networks for large scale cattle monitoring**. Tese (Doutorado) — University of Nottingham, 2008.

ZEMO, T.; KLEMMEDSON, J. O. Behavior of fistulated steers on a desert grassland. **Journal of Range Management**, JSTOR, p. 158–163, 1970.

ZHAO, C.; SICHITIU, M. L. N-body: Social based mobility model for wireless ad hoc network research. In: IEEE. **Sensor Mesh and Ad Hoc Communications and Networks (SECON), 2010 7th Annual IEEE Communications Society Conference on**, 2010. p. 1–9.

ZONOOZI, M. M.; DASSANAYAKE, P. User mobility modeling and characterization of mobility patterns. **IEEE Journal on selected areas in communications**, IEEE, v. 15, n. 7, p. 1239–1252, 1997.

7 ANEXO 1

Listing 7.1: Arquivo de configuração XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <initialParams areaTypeID="0" behaviorID="0"/>
    <leaderGroup percentValue="2"/>
    <areaTransitionMatrix matrixID="0"/>

    <areaTypes>
        <areaType areaTypeID="0" name="Pasto" xMin="0" yMin="0" zMin="0"
            xMax="1000" yMax="1000" zMax="0" matrixID="1"/>
        <areaType areaTypeID="1" name="Agua" xMin="0" yMin="900" zMin="0"
            xMax="500" yMax="1000" zMax="0" matrixID="2"/>
        <areaType areaTypeID="2" name="Cocho" xMin="500" yMin="0" zMin="0"
             xMax="1000" yMax="100" zMax="0" matrixID="3"/>
        <areaType areaTypeID="3" name="Sombra" xMin="500" yMin="250" zMin=
            "0" xMax="1000" yMax="750" zMax="0" matrixID="4"/>
    </areaTypes>

    <behaviors>
        <behavior behaviorID="0" name="ANDANDO" minSpeed="2" maxSpeed="5"
            />
        <behavior behaviorID="1" name="CORRENDO" minSpeed="5" maxSpeed="10
            " />
        <behavior behaviorID="2" name="DEITADO" minSpeed="0" maxSpeed="0"
            />
        <behavior behaviorID="3" name="ALIMENTANDO" minSpeed="1" maxSpeed=
            "2" />
    </behaviors>

    <markovMatrices>
        <markovMatrix matrixID="0" type="default">
            <row>
                <cell value="0.3" />
                <cell value="0.3" />
                <cell value="0.1" />
```

```
27                <cell value="0.3" />
28            </row>
29            <row>
30                <cell value="0.2" />
31                <cell value="0.5" />
32                <cell value="0.1" />
33                    <cell value="0.2" />
34            </row>
35            <row>
36                <cell value="0.1" />
37                <cell value="0.4" />
38                <cell value="0.3" />
39                    <cell value="0.2" />
40            </row>
41            <row>
42                <cell value="0.1" />
43                <cell value="0.1" />
44                <cell value="0.1" />
45                    <cell value="0.7" />
46            </row>
47        </markovMatrix>
48        <markovMatrix matrixID="1">
49            <row>
50                <cell value="0.5" />
51                <cell value="0.1" />
52                <cell value="0.2" />
53                    <cell value="0.2" />
54            </row>
55            <row>
56                <cell value="0.6" />
57                <cell value="0.4" />
58                <cell value="0.0" />
59                    <cell value="0.0" />
60            </row>
61            <row>
62                <cell value="0.2" />
63                <cell value="0.0" />
64                <cell value="0.5" />
65                    <cell value="0.3" />
```

```
66            </row>
67            <row>
68               <cell value="0.2" />
69               <cell value="0.0" />
70               <cell value="0.3" />
71                  <cell value="0.5" />
72            </row>
73         </markovMatrix>
74         <markovMatrix  matrixID="2">
75            <row>
76               <cell value="0.1" />
77               <cell value="0.1" />
78               <cell value="0.2" />
79                  <cell value="0.6" />
80            </row>
81            <row>
82               <cell value="0.6" />
83               <cell value="0.4" />
84               <cell value="0.0" />
85                  <cell value="0.0" />
86            </row>
87            <row>
88               <cell value="0.2" />
89               <cell value="0.0" />
90               <cell value="0.4" />
91                  <cell value="0.4" />
92            </row>
93            <row>
94               <cell value="0.1" />
95               <cell value="0.0" />
96               <cell value="0.3" />
97                  <cell value="0.6" />
98            </row>
99         </markovMatrix>
100
101        <markovMatrix  matrixID="3">
102           <row>
103              <cell value="0.2" />
104              <cell value="0.1" />
```

```
105             <cell value="0.1" />
106                 <cell value="0.6" />
107          </row>
108          <row>
109             <cell value="0.6" />
110             <cell value="0.4" />
111             <cell value="0.0" />
112                 <cell value="0.0" />
113          </row>
114          <row>
115             <cell value="0.2" />
116             <cell value="0.0" />
117             <cell value="0.2" />
118                 <cell value="0.6" />
119          </row>
120          <row>
121             <cell value="0.2" />
122             <cell value="0.0" />
123             <cell value="0.2" />
124                 <cell value="0.6" />
125          </row>
126       </markovMatrix>
127
128       <markovMatrix  matrixID="4">
129          <row>
130             <cell value="0.4" />
131             <cell value="0.1" />
132             <cell value="0.4" />
133                 <cell value="0.1" />
134          </row>
135          <row>
136             <cell value="0.6" />
137             <cell value="0.4" />
138             <cell value="0.0" />
139                 <cell value="0.0" />
140          </row>
141          <row>
142             <cell value="0.2" />
143             <cell value="0.0" />
```

```
144             <cell value="0.6" />
145                 <cell value="0.2" />
146           </row>
147           <row>
148             <cell value="0.2" />
149             <cell value="0.0" />
150             <cell value="0.5" />
151                 <cell value="0.3" />
152           </row>
153       </markovMatrix>
154   </markovMatrices>
155 </root>
```