MARCOS YUZURU DE OLIVEIRA CAMADA

MASIM: UMA FERRAMENTA PARA SIMULAÇÃO DE AGENTES MÓVEIS EM REDES DE SENSORES SEM FIO

FLORIANÓPOLIS
2009

UNIVERSIDADE FEDERAL DE SANTA CATARINA

CURSO DE PÓS-GRADUAÇÃO
EM ENGENHARIA DE AUTOMAÇÃO E SISTEMA

MASIM: UMA FERRAMENTA PARA SIMULAÇÃO DE AGENTES MÓVEIS EM REDES DE SENSORES SEM FIO

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia de Automação e Sistemas.

MARCOS YUZURU DE OLIVEIRA CAMADA

Florianópolis, Agosto de 2009.

MASIM: UMA FERRAMENTA PARA SIMULAÇÃO DE AGENTES MÓVEIS EM REDE DE SENSORES SEM FIO

Marcos Yuzuru de Oliveira Camada

'Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia de Automação e Sistemas, Área de Concentração em *Controle, Automação e Sistemas*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina.'

Carlos Barros Montez, Dr.
Orientador

Prof. Eugenio de Bona Castelan Neto
Coordenador do Programa de Pós-Graduação em Automação e Sistema

Banca Examinadora:

Carlos Barros Montez, Dr.
Presidente

Flávio Morais de Assis Silva, Dr.
Co-orientador

Michelle Silva Wangham, Dr[a].

Rômulo Silva de Oliveira, Dr.

Jomi Fred Hübner, Dr.

A minha família e aos meus verdadeiros amigos que sempre me apoiaram...

AGRADECIMENTOS

Antes de tudo, eu agradeço eternamente aos meus queridos pais, Ernesto e Solange Camada, os quais são meus exemplos de vida e que sempre me apoiaram incondicionalmente, foram afetuosos e desejaram o melhor para mim. Além disto, agradeço também a meus estimados irmãos, Diogo e Ilza Camada, também pelo apoio e carinho. Com certeza, esta minha vitória é também uma vitória para minha amada família.

A meu orientador Carlos Montez e co-orientador Flávio Assis pelo aprendizado, paciência e sabedoria em me direcionar para o desenvolvimento deste trabalho.

Aos grandes e valiosos amigos de Salvador-BA, em especial, Flavinha, Romildo, Danuza e Pedrita, os quais, sempre que possível, procuraram me apoiar e mesmo com distancia demonstraram seu afeto.

Aos meus grandes amigos do "Sindicato do LCMI"pelo apoio, diversão e vitórias durante esta jornada.

Ao amigo Benedito Junior pela ajuda e conselhos sobre MATLAB e Simulink que foram fundamentais para o desenvolvimento deste meu trabalho.

Aos meus grandes amigos da "República Pé Vermeio"pelas diversões, companheirismo e que, praticamente, foram meus irmãos em Florianópolis.

Aos "Vizinhos de Cima", Tia Ivete, Aline e família, pela amizade e carinho.

Aos meus amigos de escalada, Freitas, Elton, Gabriel, Marquinhos, Warody, Talita e as demais pessoas do PIB Adventure pelas aventuras e amizade.

Aos professores do Departamento de Automação e Sistema da UFSC pelo aprendizado.

Aos funcionários responsáveis pela manutenção e limpeza do laboratório pelos valiosos serviços.

Aos meus amigos da empresa Pixeon pela amizade e profissionalismo. Agradecimentos especiais a Iomani Engelmann e Fernando Peixoto pela compreensão em flexibilizar do meu horário de trabalho durante a fase crítica do desenvolvimento deste meu trabalho.

Aos meus amigos do curso de Bombeiro Comunitário Voluntário pela garra, comprometimento pelas causas nobres e profissionalismo.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) pela disponibilização da bolsa de pesquisa.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia de Automação e Sistemas.

MASIM: UMA FERRAMENTA PARA SIMULAÇÃO DE AGENTES MÓVEIS EM REDES DE SENSORES SEM FIO

Marcos Yuzuru de Oliveira Camada

Agosto/2009

Orientador: Carlos Barros Montez
Co-Orientador: Flávio de Assis Silva
Área de Concentração: Controle, Automação e Informática Industrial
Palavras-chave: Rede de Sensores Sem Fio, Agentes Móveis, Simulador
Número de Páginas: xxiii + 86

Agentes móveis são componentes de *softwares* autônomos capazes de se mover carregando consigo, além do seu código, o seu estado de execução. Recentemente, vem sendo proposto o uso deste tipo de agentes em Redes de Sensores Sem Fio (RSSF). Além da facilidade da reprogramação dinâmica do código que executa nos nodos, o uso de agentes pode, em determinados cenários, economizar energia e prolongar o tempo de vida da rede. Devido à carência de ferramentas próprias para simulação de aplicações baseadas em agentes móveis em RSSF, este trabalho propõe uma ferramenta com este objetivo denominada de MASiM (*Mobile Agents Simulator in Wireless Sensor Network*). A ferramenta proposta foi desenvolvida usando linguagem de programação C++ MEX e estende os *toolboxes* Simulink e TrueTime do MATLAB. A utilização da ferramenta é demonstrada no trabalho através do desenvolvimento de diversos cenários. Além disso, usando essa ferramenta, é realizada uma breve comparação de protocolos baseados em agentes com protocolos baseados em difusão de mensagens.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

MASIM: A SIMULATION TOOL FOR MOBILE AGENTS IN WIRELESS SENSOR NETWORKS

Marcos Yuzuru de Oliveira Camada

August/2009

Advisor: Carlos Barros Montez
Co-Advisor: Flávio de Assis Silva
Area of Concentration: Control, Automation and Industrial Computing
Key words: Wireless Sensor Network, Mobile Agents, Simulator
Number of Pages: xxiii + 86

Mobile Agents are autonomous software components able to move carrying their code and execution state. Recently, it has been proposed the use of such agents in Wireless Sensor Networks (WSN). Besides the ease of dynamic reprogramming in the code that runs in the nodes, the use of agents may, in certain scenarios, save energy and prolong the network lifetime. Due to the lack of suitable tools for applications simulation based on mobile agents in WSN, this master dissertation introduces a tool for this purpose called MASiM (Mobile Agents Simulator in Wireless Sensor Network). This tool was developed using the C ++ MEX programming language and it extends the toolboxes of MATLAB, Simulink and TrueTime. The use of the tool is presented in this dissertation by developing several scenarios. Furthermore, it was made a brief comparison between protocols based on agents and those based on broadcast.

Sumário

Lista de Abreviaturas

ACA	Arquitetura de Componentes Autônomos (*Autonomous Component Architecture*)
ADC	Conversor de Analógico-Digital (*Analog-Digital Converter*)
API	Interfaces de Programação de Aplicação (*Application Programming Interfaces*)
APS	Sub-camada de Suporte de Aplicação (*Application Support Sub-Layer*)
ATEMU	*ATmel EMUlator*
AVR	*Advanced Virtual RISC*
BLAS	Subprogramas de Álgebra Linear Básica (*Basic Linear Algebra Subprograms*)
CAN	*Controller Area Network*
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
DCF	Função de Coordenação Distribuída (*Distributed Coordination Function*)
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
E/S	Entrada e Saída
FDMA	Acesso Múltiplo de Divisão de Frequência (*Frequency Division Multiple Access*)
FFD	Dispositivos de Funções Completas (*Full Function Device*)
FIPA	*Foundation for Intelligent Physical Agents*
FORTRAN	*FORmula TRANslation*
FTP	Protocolo de Transferência de Arquivo (*File Transfer Protocol*)
GPS	Sistema de Posicionamento Global (*Global Positioning System*)
IEEE	*Institute of Electrical and Electronics Engineers*
J-SIM	*Java SIMulator*
LAPACK	*Linear Algebra PACKage*
MAC	*Media Access Control*
MADD	*Mobile Agent Based Wireless Sensor Network*
MANET	*Mobile Ad hoc Network*
MASIF	*Mobile Agent System Interoperability Facility*
MASIM	*Mobile Agents SIMulator in Wireless Sensor Network*

MATLAB	*MATrix LABoratory*
MAWSN	*Mobile Agent based Wireless Sensor Network*
MDL	Linguagem de Descrição de Modelo (*Model Description Language*)
MEX	*Matlab EXecutable*
NAM	*Network AniMator*
NS-2	*Network Simulator*
OMG	*Object Management Group*
OTCL	*Object oriented TCL*
RSSF	Rede de Sensores Sem Fio
RSF	Dispositivos de Funções Reduzidas (*Reduced Function Device*)
SAM	Sistemas de Agentes Móveis
SAP	Ponto de Acesso a Serviço (*Service Access Point*)
SGA	Sistema de Gerenciamento de Agentes
STM	Serviço de Transporte de Mensagem
SENMA	*Sensor Network with Mobile Agents*
SENSE	*SEnsor Network Simulator and Emulator*
SGA	Sistema de Gerenciamento de Agente
SO	Sistema Operacional
SRM	*Scalable Reliable Multicast*
STM	Serviço de Transporte de Mensagem
TCL	Linguagem de Comandos de Ferramentas (*Tool Command Language*)
TCP	Procolo de Controle de Transmissão (*Transmission Control Protocol*)
TDMA	Acesso Múltiplo de Divisão Tempo (*Time Division Multiple Access*)
TOSSIM	*TinyOS SIMulator*
UDP	Protocolo de Datagrama de Usuário (*User Datagram Protocol*)
UML	Linguagem de Modelagem Unificada (*Unified Modeling Language*)
ZC	Coordenador Zigbee (*Zigbee Coordenator*)
ZDO	Objecto de Dispositivo Zigbee (*Zigbee Device Object*)
ZED	Dispositivo Final Zigbee (*Zigbee End Device*)
ZR	Roteador Zigbee (*Zigbee Router*)
WLAN	Rede de Área Local Sem Fio (*Wireless Local Area Network*)

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

O monitoramento de regiões abrangentes e de difícil acesso ou que apresente algum risco ao ser humano pode ser uma necessidade nas áreas civil e militar. Uma abordagem possível de baixo custo para estas demandas é a utilização de Redes de Sensores Sem Fio (RSSF) (Yu *et al.* [2004]). Na área civil, pode ser aplicada em ambientes de manufatura para monitoramento e controle dos equipamentos fabris. Além disto, a RSSF também pode ser utilizada para sensoriamento de poluição em afluentes, na agricultura e em residências domésticas. Na área militar, a RSSF pode ser aplicada para monitorar deslocamento inimigo ou a presença de substâncias nocivas ao ser humano (Cayirci *et al.* [2002]).

Uma RSSF é um tipo de rede sem fio constituída por um conjunto de nodos com sensores que podem monitorar determinados fenômenos tais como, temperatura, luminosidade, movimentos, umidade, presença de fogo ou algum produto químico em certa região (Loureiro *et al.* [2003]). As RSSF podem ser constituídas por dezenas de nodos. Estes nodos podem ser lançados de maneira aleatória no ambiente e são capazes de se auto-organizarem (Malik e Shakshuki [2007]). As RSSF devem ser capazes de operar em condições dinâmicas, pois fatores internos (ex. falha na comunicação, exaustão energética dos nodos, ou inclusão de novos nodos na rede) ou externos (ex. mudanças topológicas, inferência de rádio no canal de comunicação) podem ocorrer (Sohrabi *et al.* [2000]).

Uma grande vantagem que a RSSF apresenta em relação às redes cabeadas e outros tipos de redes sem fio deve-se ao pequeno tamanho dos nodos que a constituem, o que permite o sistema tornar-se pervasivo. Além disto, em alguns tipos de nodos apresentam baixo custo de produção. Desta forma, pode-se aplicar uma quantidade de nodos capaz de monitorar uma área abrangente. Estes nodos são pequenos dispositivos computacionais autônomos formados por um processador de baixa potência de processamento, memória e bateria de baixa capacidade. Eventualmente, estes nodos

podem se mover através de um equipamento específico para isto (Ye *et al.* [2001]).

Devido ao fato de apresentar baixo consumo energético, os nodos deste tipo de rede devem se comportar de maneira econômica energeticamente, para assim, aumentar o tempo de vida da rede como um todo. Desta forma, é necessário adotar medidas que visem à economia de energia. Estas medidas podem ser realizadas em *hardware* ou em *software*. As soluções de *hardware* estão relacionadas com a utilização de dispositivos mais eficientes no consumo de energia no nodo. As soluções em *software* consistem na utilização de algoritmos mais otimizados e que evitem a utilização do canal sem fio.

Uma proposta de solução de *software* é a utilização de aplicações baseadas em agentes móveis. O agente móvel é um sistema de *software* autônomo que apresenta, como principal característica, a capacidade de se transportar de uma estação (nodo) para outra utilizando, para isto, recursos da rede de comunicação. Quando este agente se move, ele carrega consigo seu código e estado de execução. Desta forma, quando ele chega à estação destino, ele é capaz de voltar ao seu estado de execução antes de se mover (Macêdo e Silva [2005]).

A utilização de agentes móveis pode apresentar algumas vantagens, tais como, redução de carga da rede, abstração de protocolo das camadas de rede inferiores, execução assíncrona e autônoma, capacidade de se adaptar às mudanças do sistema, natureza heterogênea (diferentes *hardwares* e sistemas operacionais), robustez e tolerância a falhas. Devido a estas vantagens, os agentes móveis são utilizados para diversas aplicações de redes cabeadas tradicionais e redes *ad hoc* sem fio. Alguns tipos de aplicações para estas redes são comércio eletrônico (*e-commerce*), assistência pessoal através de execução local, segurança, encaminhamento de informações distribuídas, serviços de redes de telecomunicação, aplicações de grupo ou *Workflow*, monitoramento e notificação, disseminação de informação e processamento paralelo (Lange e Oshima [1999]).

A utilização de agentes móveis em RSSF não é adequada em todas as situações. Atualmente, os sistemas baseados em agentes móveis apresentam grandes desafios principalmente relacionados com segurança e desempenho. Além disto, as abordagens simples baseadas em trocas de mensagens podem ser vantajosas dependendo da configuração da rede e da aplicação que executa sobre esta (Jansen *et al.* [1999]). No entanto, para se avaliar a adequação de uma determinada abordagem, é necessário efetuar uma avaliação de desempenho no sistema. A escolha da técnica de avaliação de desempenho é uma etapa importante na fase de elaboração de um determinado projeto. Pode-se considerar que existem três técnicas: modelagem analítica, simulação e mensuração. Estas técnicas podem ser usadas em conjunto, embora na aplicação delas devem-se considerar algumas condições, tais como, estágio do ciclo de vida e tempo disponível para avaliação do sistema, ferramentas disponíveis, custo alocado

ao projeto e nível de precisão requerida (Jain [1991]).

1.1 Motivação

Atualmente, existem ferramentas que permitem realizar simulações de RSSF. Contudo, uma limitação frequentemente encontrada nestes simuladores está relacionada à ausência ou limitação do modelo energético adotado. Os modelos energéticos, quando existentes, são geralmente simples e pouco realistas. Além disto, nenhum dos simuladores para RSSF pesquisados disponibilizam um suporte de recursos voltado para a simulação de aplicações baseadas em agentes móveis.

1.2 Objetivos

Devido à carência de ferramentas próprias para simulação de aplicações baseadas em agentes móveis em RSSF, este trabalho tem como objetivo geral elaborar uma ferramenta que atenda esta necessidade. Além disto, objetivos específicos deste trabalho são:

- Desenvolver uma ferramenta que implemente a maioria dos estados de um agente móvel definido pela FIPA (*Foundation for Intelligent Physical Agents*) (FIPA [2004]) e OMG (*Object Management Group*) (OMG [1997]), tais como, mobilidade, clonagem, serialização e carregamento ou criação do agente. A ferramenta deve seguir o modelo energético adotado neste trabalho é baseado nas especificações de *hardware* do *Mica2* (Crossbow [2009]). A programação das simulações é baseada na especificação de missão dos agentes, o que determina ciclo de vida de um ou conjunto de agentes de maneira simples e flexível;
- Realizar uma comparação entre protocolos baseados em agentes móveis e difusão, descrevendo possíveis pontos positivos e negativos entre eles.

1.3 Metodologia

A metodologia aplicada para elaboração deste trabalho consistiu, inicialmente, em realizar um estudo das ferramentas de simulação de redes sem fio, tais como, *JSim* (JSim [2005]), *NS-2* (Xue *et al.* [2007]), *ATEMU* (Polley *et al.* [2004]), *TOSSIM* (Levis *et al.* [2003]), *SENSE* (Chen *et al.* [2004]) e o *MATLAB* (MathWorks [2009]). Assim, foi analisado o modelo de consumo energético, a linguagem

de especificação de cenários de simulação, a plataforma de Sistema Operacional, a possibilidade dos nodos moverem-se e qual a especificação das camadas de enlace e física destas ferramentas. Além disto, foram analisados quais destas ferramentas possibilitam realizar simulação de uma RSSF convencional, seguindo a especificação *IEEE 802.15.4* para camada de enlace e física.

A etapa seguinte foi o estudo de ferramentas relacionadas à simulação de agentes móveis em RSSF. Nestes estudos, verificou-se, que até o momento, não existem ferramentas com este objetivo específico.

Na etapa seguinte, os requisitos que a ferramenta deve atender foram definidos. Através destes requisitos, foi definido que a ferramenta *MATLAB* (com os *toolboxes Simulink* e *TrueTime*) seria utilizada como plataforma base para elaboração deste trabalho por suportar a simulação de RSSF.

A etapa de modelagem da ferramenta se definiu as principais entidades da ferramenta e como estas se relacionam. Além disto, foram definidos o ciclo de vida do agente, dos protocolos de comunicação de mobilidade e de clonagem destes agentes, além dos modelos de RSSF, tarefas e falhas. Esta modelagem baseou-se nos modelos de agentes das entidades FIPA (FIPA [2004]) e a OMG (OMG [1997]).

Na etapa de codificação, utilizou-se a linguagem de programação *C++ MEX* no ambiente *MATLAB*. Nesta etapa, também foram estendidos os blocos disponibilizados pelo *Simulink* e *TrueTime* para definição da topologia da RSSF.

Os testes realizados na ferramenta verificaram se o consumo de energia e comunicação dos nodos obedece as especificações do Mica2 (Crossbow [2009]). Estes testes foram realizados utilizando o *MATLAB 7.0.4* sobre o sistema operacional *Linux Ubuntu 8.10 - Intrepid Ibexs*.

1.4 Organização do Texto

Este trabalho está estruturado da seguinte forma: no Capítulo 2 é realizada uma descrição geral de RSSF, caracterizando-a e abordando sua organização. A descrição dos agentes móveis, tal como, exemplo de aplicações em RSSF é feita no Capítulo 3. O ambiente de simulação base utilizado neste trabalho (*MATLAB*, *Simulink* e *TrueTime*) como também outros ambientes de simulação de RSSF populares são apresentados no Capítulo 4. Os modelos da RSSF e dos agentes móveis utilizados neste trabalho são descritos no Capítulo 5. A aplicação da ferramenta proposta está presente no Capítulo 6. A conclusão e sugestões sobre trabalhos futuros são descritas no Capítulo 7.

Além destes Capítulos, alguns materiais foram anexados a este trabalho. No Apêndice A, são apresentados os casos de usos descritivos dos requisitos definidos neste trabalho. Um exemplo de arquivo de configuração inicial da ferramenta está presente no Apêndice B. A amostra de um exemplo de topologia de rede definida através de blocos do *MASiM*, *TrueTime* e *Simulink* é apresentada no Apêndice C. No Apêndice D, é definido o arquivo de configuração da organização de um cenário de simulação. Um exemplo de definição de missão de agente está presente no Apêndice E. No Apêndice F, são apresentados alguns gráficos com resultados estatísticos disponibilizados pela ferramenta proposta. Nos Apêndices G e H, são apresentados os resultados estatísticos gerais das simulações de protocolos baseados em agentes móveis e difusão nos cenários de 50 e 100 nodos, respectivamente.

Capítulo 2

Rede de Sensores Sem Fio

Este Capítulo faz uma caracterização das Redes de Sensores Sem Fio de um modo geral, possibilitando, assim, entendimento da aplicação da ferramenta proposta neste trabalho. Portanto, este Capítulo está organizado da seguinte maneira: na Seção 2.1, é apresentada uma definição de RSSF e a arquitetura dos nodos sensores. A classificação das RSSF é realizada na Seção 2.2. Na Seção 2.3, são apresentados exemplos de alguns tipos de aplicações possíveis de RSSF. A organização do protocolo utilizado nas RSSF, *Zigbee* (*IEEE 802.15.4*), é abordada em 2.4. Na Seção 2.5, é realizada a descrição de como a RSSF estar organizada. A Seção 2.6 apresenta as considerações finais deste Capítulo.

2.1 Caracterização

As Redes de Sensores Sem Fio (RSSF) são consideradas como um sistema que pode ser composto por centenas de nodos autônomos, que, normalmente, trabalham em conjunto com o objetivo de realizar a missão de todo o sistema. Este tipo rede pode não apresentar uma infraestrutura pré-definida, ou seja, os nodos podem ser distribuídos de maneira aleatória em um determinado ambiente (Karl e Willig [2003]; Loureiro *et al.* [2003]).

Os nodos sensores são caracterizados principalmente por apresentarem sensores que conseguem monitorar determinados fenômenos ambientais. Estes nodos possuem dimensões reduzidas, baixa capacidade computacional (processador e memória) e energética (Karl e Willig [2003]). Além disto, eventualmente, estes nodos também podem se comportar como nodos roteadores de mensagens (Heo e Varshney [2003]).

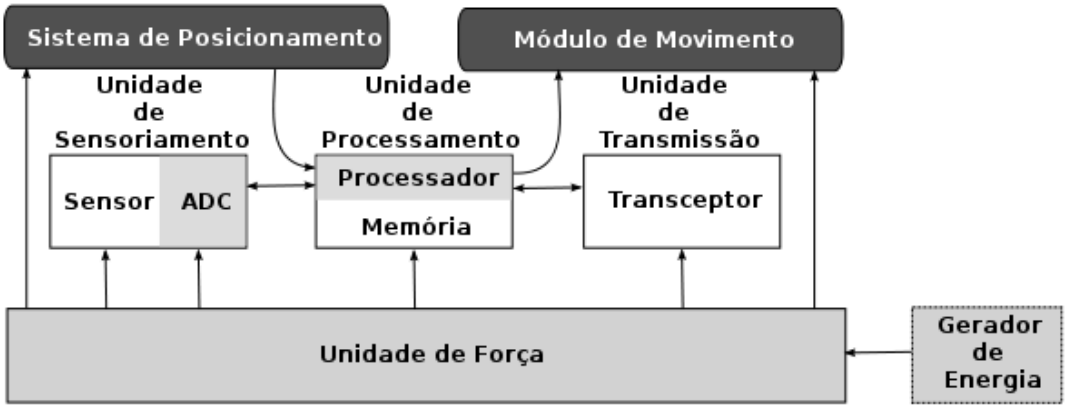

Figura 2.1: Modelo da arquitetura de um nodo sensor (Cayirci *et al.* [2002]).

A arquitetura de um nodo sensor está representada pela Figura 2.1. Desta maneira, este nodo é composto por uma **Unidade de Processamento** (**Memória** e **Processador**), uma **Unidade de Transmissão e Recepção de Mensagens** (**Transceptor**), **Unidade de Sensoriamento** (**Sensor**, **Conversor Analógico/Digital**) e uma **Unidade de Força**. Em alguns projetos de RSSF, os nodos também podem apresentar um **Sistema de Posicionamento** (ex.: *Global Positioning System - GPS*), um **Módulo de Movimento**, e um **Gerador de Energia** (ex.: células fotoelétricas) (Cayirci *et al.* [2002]; Loureiro *et al.* [2003]).

A **Unidade de Processamento** é responsável por gerenciar e executar as tarefas relacionadas aos sensores, ao envio e recebimento de mensagens e Sistema Operacional. A **Unidade de Transmissão** é um dispositivo capaz de estabelecer comunicação com outros nodos através de sinais de rádio. A **Unidade de Sensoriamento** se resume, basicamente, ao sensor em que os dados analógicos do fenômeno monitorado devem ser digitalizados. A **Unidade de Força** é responsável pela alimentação energética do sistema do nodo e é constituída, basicamente, por baterias (Cayirci *et al.* [2002]).

Além dos dispositivos obrigatórios citados anteriormente, eventualmente, um nodo sensor também pode apresentar uma **Unidade de Posicionamento**. Neste caso, o dispositivo mais referenciado é o *GPS*. O *GPS* é um equipamento que fornece a localização através das informações disponibilizadas por satélites. Um **Módulo de Movimento** também pode ser parte dos acessórios de um nodo sensor. Este equipamento permite os nodos sensores manipulem o ambiente, como, por exemplo, realizar deslocamento sobre uma superfície. Uma RSSF pode ser projetada para atuar em uma região inóspita que pode ocasionar risco a vida humana, desta maneira, a manutenção energética dos nodos sensores via intervenção humana (troca ou recarga das baterias) pode ser substituída por um **Gerador de Energia** para aumentar o tempo de vida da rede (Cayirci *et al.* [2002]).

Em um projeto de RSSF, alguns requisitos são importantes, tais como: capacidade de se adaptar na ocorrência de falhas na comunicação ou dos nodos (tolerância a falhas) e realizar auto-gerenciamento (configuração, manutenção e organização) sem a intervenção direta de seres humanos (Koushanfar *et al.* [2002]; Sohrabi *et al.* [2000]).

Além dos nodos sensores, uma RSSF é constituída por um ou mais nodos sorvedouros denominados de estações base. Uma justificativa para a presença de mais de uma estação base pode ser para que uma RSSF seja tolerante a falhas (Karlof e Wagner [2003]). A Estação Base é considerada um nodo especial que possui maior capacidade de processamento, memória e energia, podendo ser recarregável continuamente. Além do mais, estes nodos podem apresentar um alcance de rádio muito maior se comparado com os nodos sensores normais.

Diferentemente como ocorre em redes tradicionais, uma RSSF, normalmente, não é de propósito geral, ou seja, uma RSSF não é utilizada para vários tipos de aplicações. Assim, esta rede é projetada para uma aplicação específica. Desta maneira, dependendo do tipo de fenômeno a ser monitorado, condições da topologia local e precisão deste monitoramento, diferentes arquiteturas e restrições deverão ser consideradas para a RSSF, tais como: pilha de protocolo a ser utilizada na comunicação, configuração de *software* e *hardware* (tipo específico de sensor, presença ou não de atuadores ou de nodos heterogêneos) e tipo de fonte energética a ser utilizada pelos nodos (Loureiro *et al.* [2003]).

Em um projeto de RSSF, o consumo de energia é um aspecto importante a ser considerado. Uma RSSF pode ser aplicada em regiões de difícil acesso, logo, a manutenção energética (troca ou recarga da bateria) através da intervenção humana pode não ser possível. O tempo de vida de uma RSSF é diretamente relacionado à disponibilidade energética dos seus nodos. Portanto, medidas que garantam uma maior economia de energia são necessárias. Esta economia pode ser alcançada via *hardware* e *software*. A economia via *hardware* pode ser bem sucedida através do uso de processadores de menor potência e antenas mais eficientes na utilização do canal de transmissão. O maior consumo de energia se dá na comunicação dos nodos (envio e recebimento de mensagens), assim, a economia através de solução em *software* pode ser obtida, principalmente, no tipo do protocolo de comunicação utilizado entre os nodos, visando à diminuição de mensagens trocadas entre os nodos (Heinzelman *et al.* [2000]; Rodoplu e Meng [1999]).

Os protocolos de comunicação dos nodos podem adotar diferentes estratégias para minimizar a quantidade de mensagens trocadas na rede. Uma delas pode ser o pré-processamento dos dados do fenômeno monitorado antes de serem transmitidos (Lindsey e Raghavendra [2002]; Manjeshwar e Agrawaly [2002]). Além disto, os dados podem ser enviados diretamente (Heinzelman *et al.* [2000])

para o nodo sorvedouro ou serem transmitidos em múltiplos saltos (Manjeshwar e Agrawal [2003]), ou seja, os dados são transmitidos para os nodos vizinhos mais próximos que retransmitem para outros nodos, até chegar ao nodo sorvedouro. Neste caso, os nodos retransmissores se comportam como nodos roteadores. Uma abordagem alternativa é a utilização de agentes móveis. Neste caso, os agentes se movem de um nodo a outro, carregando consigo o seu estado de execução e os dados do fenômeno monitorado, podendo fazer um pré-processamento sobre estes. Esta última abordagem é a considerada neste trabalho.

2.2 Classificações das RSSF

Uma RSSF é direcionada para um determinado tipo de aplicação. Assim, a arquitetura (*hardware* e *software*) dos nodos, as pilhas de protocolo a ser utilizadas e a quantidade de nodos da rede irão depender do ambiente onde será aplicado a rede e o tipo de fenômeno que será monitorado. Da mesma forma, as RSSF podem ser classificadas em relação a diferentes critérios. Na Tabela 2.1, são apresentadas algumas classificações de RSSF (Ilyas e Mahgoub [2005]).

Fatores	**Grupos Distintos**
Distancia da estação base	Single-hop *vs* Multi-hop
Dependência de dados	Não agregante *vs* Agregante
Distribuição dos nodos sensores	Determinístico *vs* Dinâmico
Esquema de controle	Auto-configurável *vs* Não auto-configurável
Modo de ativação dos nodos sensores	Pró-ativa *vs* Reativa

Tabela 2.1: Classificação das RSSF de acordo com diferentes critérios (Ilyas e Mahgoub [2005]; Manjeshwar e Agrawal [2001]).

Um critério de classificação que pode ser utilizado é em relação a distancia dos nodos sensores com a estação base. Desta forma, de acordo com este critério, as RSSF podem ser classificadas como (Ilyas e Mahgoub [2005]):

- **Salto simples (*single-hope*):** todos os nodos sensores conseguem se comunicar diretamente com a estação base. Neste caso, a infraestrutura da rede é mais simples e a região alvo de monitoramento não deve ser extensa;
- **Múltiplos saltos (*multi-hope*):** alguns nodos necessitam entregar os dados para a estação base através de nodos intermediários. Estes nodos intermediários comportam-se como roteadores. Estes nodos podem aplicar algum processamento de agregação (ou fusão) nestes dados. A infraestrutura deste tipo de RSSF é mais complexa e deve ser aplicada em regiões extensas.

As RSSF também podem ser classificadas em relação à densidade de nodos e dependência dos dados, que podem ser (Ilyas e Mahgoub [2005]):

- **Agregado:** em uma RSSF densamente distribuída, as informações do fenômeno monitorado pelos nodos sensores e seus vizinhos podem ser co-relacionadas devido à proximidade destes nodos. Deste modo, os nodos intermediários podem aplicar operações de agregação ou fusão nos dados recebidos dos seus vizinhos, removendo, assim, possíveis redundâncias presentes nestas informações. A principal vantagem deve-se à significativa redução de tráfego de mensagens na rede, da mesma forma, o consumo de energia na comunicação das mensagens também é diminuído. Entretanto, há o aumento no consumo de energia e de memória dos nodos intermediários, devido ao processamento necessário para realizar a agregação ou fusão dos dados. Este caso deve ser aplicado em uma RSSF densamente distribuída em que o interesse do Usuário seja nas informações coletivas com uma precisão moderada;
- **Não Agregado:** os dados de cada nodo são enviados para seu destino sem nenhum processamento de agregação ou fusão sobre eles. A vantagem nesta abordagem é que os nodos intermediários não gastam tanta energia de processamento no tratamento das mensagens (recebidas ou enviadas). Além disto, é possível obter uma alta precisão das informações do monitoramento. Contudo, a grande desvantagem é o aumento rápido do tráfego de mensagens, podendo elevar a quantidade de colisões, congestionamentos e latência dos enlaces. Além disto, esta abordagem pode ocasionar o aumento do consumo de energia com trocas de mensagens. Assim, este caso deve ser aplicado em redes menos densa em que o Usuário necessite de alta precisão nas informações monitoradas.

Em relação à arquitetura de *hardware* dos nodos sensores atuais, geralmente, é considerado que o maior consumo de energia está relacionado com a transmissão de dados. Isto se deve ao envolvimento de processamento, ativação e manutenção do rádio. A energia consumida nestas atividades é, consideravelmente, maior em relação ao que é gasto no processamento de dados apenas (Ilyas e Mahgoub [2005]).

A forma que os nodos são dispostos no ambiente também pode ser usada como critério de classificação das RSSF. Desta forma, as RSSF podem ser classificadas como (Ilyas e Mahgoub [2005]):

- **Determinística:** o posicionamento dos nodos são fixos ou pré-planejados. Neste caso, o controle e implementação do sistema são simples. Contudo, este esquema pode ser aplicado em poucos tipos de sistemas, onde a informação do posicionamento do nodo sensor pode ser obtida previamente;

- **Dinâmica:** o posicionamento dos nodos não é determinado previamente e existe um alto grau de imprevisibilidade de como estes nodos irão se organizar quando são lançados no ambiente. Este caso é utilizado na maioria das aplicações de RSSF e necessita de algoritmos de controle mais complexos.

O esquema de controle dos nodos também pode ser utilizado para a classificação das RSSF. Neste caso, pode ser de dois tipos(Ilyas e Mahgoub [2005]):

- **Auto-configurável:** os nodos sensores são capazes de estabelecer e manter a conectividade entre eles, com o objetivo de colaborar entre si e cumprir uma tarefa de sensoriamento e de controle. Este esquema é mais aplicável em um sistema de larga escalar em que o cumprimento das tarefas de monitoramento, coleta e disseminação de informação são mais complexos;

- **Não Auto-configurável:** os nodos sensores não apresentam um mecanismo em que permita-os organizar por eles próprios, dependendo de um controle central. Este esquema pode ser utilizado em pequena escala.

Embora o sistema **Auto-configurável** seja mais complexo que o sistema **Não Auto-configurável**, eles são mais práticos para serem utilizados no mundo real, especialmente quando o tamanho da rede torna-se muito grande.

A última forma de classificação de RSSF considerada neste trabalho é em relação ao modo de ativação dos nodos sensores. Desta forma, usando este critério, uma RSSF pode ser classificada como (Manjeshwar e Agrawal [2001]):

- **Pró-ativa:** são caracterizadas por enviar mensagens periodicamente sobre o fenômeno monitorado para outros nodos;

- **Reativa:** os nodos enviam mensagem para outros nodos sobre o fenômeno monitorado, quando o valor desse nodo atinge uma zona alvo.

As classificações abordadas nesta seção podem sobrepor entre elas, pois uma RSSF pode apresentar características de diferentes domínios. Assim, em uma região ampla, é recomendável que uma RSSF seja auto-configurável, determinística, agregada e multi-hop (Ilyas e Mahgoub [2005]).

2.3 Tipos de Aplicações de RSSF

Atualmente, existe uma ampla possibilidade de aplicações reais e em potenciais de RSSF. Isto se deve as características deste tipo de rede, tais como, confiabilidade, auto-organização, ser extensível e fácil implantação. Assim, a utilização de RSSF é recomendável, principalmente, em situações na qual a utilização de redes convencionais não é possível como, ambientes inóspitos ou de difícil acesso. Alguns exemplos possíveis aplicações de RSSF são (Arampatzis *et al.* [2005]; Ilyas e Mahgoub [2005]; Loureiro *et al.* [2003]; Mainwaring *et al.* [2002]):

- **Militar:** neste tipo de aplicação de RSSF, o principal objetivo é utilizar a rede para a comunicação e controle inteligente, permitindo traçar estratégias de ataque e defesa sem o envolvimento direto de ser humano na prospecção de informações de áreas inimigas. A utilização de RSSF neste caso pode estar relacionada à obtenção de informações sobre a força ou posicionamento de um agrupamento inimigo. Além disto, é também possível realizar monitoramento de armamentos nucleares, biológicos, químicos ou detecção de potenciais ações terroristas. As principais características atrativas da RSSF para esta aplicação são, fácil instalação no ambiente, auto-organização, tolerância a falha (redundância de nodos) e detecção coletiva sem o suporte humano para manutenção;

- **Monitoramento de ambientes abertos:** a utilização de RSSF neste tipo de aplicação visa monitorar ou controlar ambientes (ecossistemas) vastos, complexos ou selvagens. Alguns exemplos são detecção de inundações, monitoramento do ar ou esgoto, monitoramento climático local, detecção do solo para uma agricultura mais precisa e eficaz, exploração de reservas minerais e estudos geofísicos. As principais características importantes da RSSF para este tipo de aplicação é a possibilidade de utilização de uma grande quantidade de nodos baratos e a capacidade de auto-configuração;

- **Prevenção de desastre e resgate:** outro tipo de aplicação de RSSF é para prevenção de desastre em construções com a aplicação de nodos em locais estratégicos da construção para detectar risco de desabamentos. Estes nodos devem ser capazes de monitorar o estresse sofrido em uma determinada construção. Em outra aplicação de prevenção está relacionada à utilização de RSSF para monitorar vazamentos em dutos de gases, combustíveis ou substancias tóxicas. As RSSF podem também ser úteis para auxiliar em operações de resgates de vítimas através de sensores sensíveis a presença humana;

- **Cuidados médicos:** em ambientes médicos, tais como, clínicas, hospitais e laboratórios, as RSSF podem ser aplicadas em diferentes cenários. Uma deles é a utilização de RSSF para

detecção de vírus ou outras doenças contagiosas, como a Malária. Além disto, em ambientes hospitalares, os nodos sensores podem ser aplicados para monitoramento de pacientes;

- **Ambientes inteligentes:** as RSSF podem ter uma aplicação em ambientes inteligentes como, por exemplo, na aferição remota do consumo de água, gás ou eletricidade, sendo estes dados transmitidos em conexões sem fio. Outro exemplo prático para esta situação é a presença de nodos sensíveis a temperatura informando a um computador central para ajudar a refrigeração ou ventilação adequada de uma determinada dependência de uma casa;

- **Exploração científica:** as RSSF podem auxiliar em pesquisas científicas em ambientes de difícil acesso ou hostil ao ser humano, tais como profundidade de oceanos ou em cavernas. Estes nodos sensores podem utilizados para monitorar abalos sísmicos, temperatura ou presença de gases;

- **Interatividade ambiental:** a utilização de RSSF para interação com o meio ambiente é uma abordagem possível e empregada, por exemplo, em alguns museus. Neste exemplo, é possível a interação das pessoas com os objetos presentes no ambiente através de sensores sensíveis a presença humana.

2.4 Protocolo de Comunicação

Atualmente, existe uma grande variedade de padrões propostos para comunicação para redes sem fio. Os mais conhecidos são *Wi-fi* (*IEEE 802.11b*) (IEEE [2007]), *Bluetooth* (*IEEE 802.15.1*) (IEEE [2005]) e *Zigbee* (*IEEE 802.15.4*) (IEEE [2006]). Na Tabela 2.2, é realizado uma comparação entre estes três protocolos. Devido à característica de cada um destes protocolos, a utilização deles dependerá do tipo de aplicação e característica de *hardware*.

Padrões	Zigbee (IEEE 802.15.4)	Wi-Fi (IEEE 802.11b)	Bluetooth (IEEE 802.15.1)
Alcance da transmissão (em metros)	1 – 100	1 – 100	1 – 10
Tempo de vida de baterias (dias)	100 – 1000	0.5 – 5	1 – 7
Número de nodos	> 64000	32	7
Tamanho da pilha de protocolo (KB)	4 – 32	1000	250

Tabela 2.2: Comparação entre os padrões *Zigbee*, *Wi-Fi* e *Bluetooth* (Digi [2009]).

O *Zigbee* é um conjunto de especificações para comunicação entre dispositivos eletrônicos sem fios de baixo custo e baixa potência mantidos pela *Zigbee Alliance*. Assim, devido às restrições de *hardware* dos nodos em uma RSSF, o protocolo normalmente utilizado segue a especificação do padrão *Zigbee*. Algumas vantagens que esta especificação oferece são (Alliance [2008]):

- Baixo ciclo processador (permitindo um tempo de vida maior da bateria);
- Baixa latência;
- Suporte a múltiplas topologias de rede: estática, dinâmica, estrela e malha;
- Suporte a mais de 65 mil nodos em uma rede;
- Provê conexões seguras entre equipamentos;
- Mecanismo para evitar colisões;
- Indicação de qualidade do enlace;
- Avaliação do canal aberto.

Em uma rede sem fio do tipo *Zigbee*, existem nodos heterogêneos que podem se diferenciar pelos seus recursos de *hardware* disponível e pelo papel que eles executam na rede. Desta forma, em relação aos recursos de *hardware*, um dispositivo pode ser classficado como (Alliance [2008]):

- **Dispositivos de Funções Completas (*Full Function Device - FFD*):** São os dispositivos que apresentam maiores recursos computacionais (processador e memória). Assim, estes consomem maior quantidade de energia. Contudo, normalmente, estes dispositivos também apresentam recursos energéticos suficientes para suprir as necessidades dos *hardwares*. Nesta topologia, o **FFD** pode assumir o papel de **Coordenador**, **Roteador** ou mesmo um **Dispositivo Final**;
- **Dispositivos de Funções Reduzidas (*Reduced Function Device - RSF*):** São dispositivos que apresentam recursos computacionais (processador e memória) e energéticos mais limitados e podem se comunicar com os **FFDs** (**Coordenador** e **Roteador**). Na topologia da rede, estes assumem o papel de **Dispositivo Final**.

Em relação ao papel que estes dispositivos podem exercer na rede *Zigbee*, estes dispositivos podem ser classificados como (Alliance [2008]):

- **Coordenador *Zigbee* (*Zigbee Coordenator - ZC*):** São os dispositivos com função de coordenador da rede. Este é responsável por inicializar a rede, distribuir os endereços para os nodos, realizar a manutenção da rede e armazenas informações sobre os nodos. Este dispositivo também pode funcionar como um *Gateway* para alguma outra rede. Este papel é complexo e necessita de grandes recursos computacionais, assim, ele é implementado por dispositivos **FFD**. Em uma RSSF, este dispositivo representa a estação base;

- **Roteador *Zigbee* (*Zigbee Router - ZR*):** São os dispositivos intermediários e têm o papel de encaminhar as mensagens de outros dispositivos para o **ZC**. Através disto, uma rede pode ser expandida em uma região extensa. Este papel é implementado pelos dispositivos **FFD**, pois exige elevados recursos computacionais. Estes dispositivos são representados pelos nodos roteadores numa RSSF;

- **Dispositivo Final *Zigbee* (*Zigbee End Device - ZED*):** São dispositivos que apresentam um *hardware* simples e específico, por isto, consomem pouca energia. Em uma RSSF, os **ZED** são representados pelos nodos sensores.

Em uma rede *Zigbee*, estes dispositivos podem se organizar em diferentes topologias. Os tipos de topologias previstas no *Zigbee* estão presentes na Figura 2.2 e são os seguintes (Alliance [2008]):

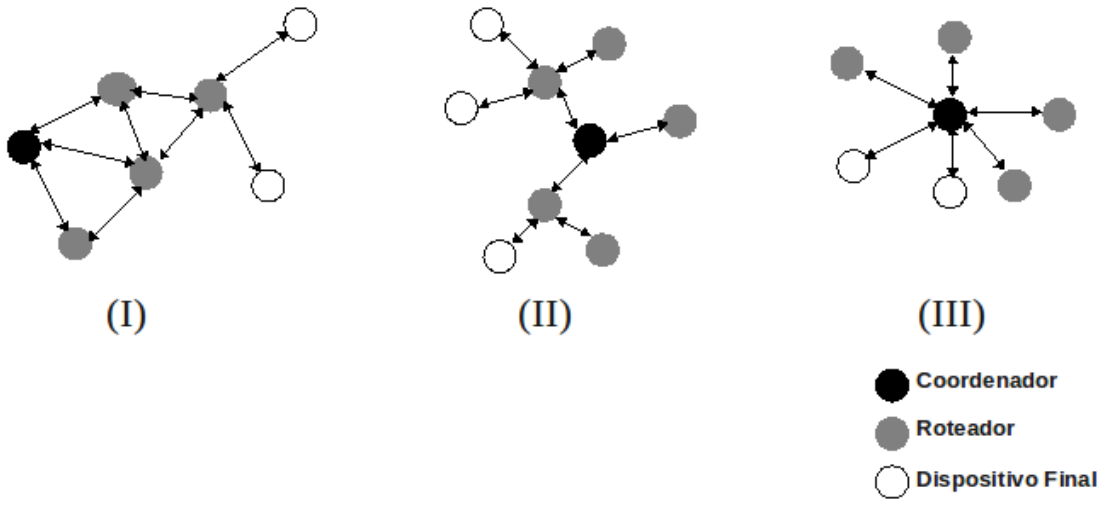

Figura 2.2: Topologias de Redes do *Zigbee*. (I) Malha, (II) Árvore, (III) Estrela (Alliance [2008]).

- **Ponto-a-ponto ou Malha (*Mesh*):** neste tipo de topologia, o papel de roteador de mensagens é descentralizado e envolve o processamento coletivo. Devido a isto, diversos caminhos para o

roteamento de mensagem podem ser realizados. Este tipo de topologia é normalmente aplicado em uma região extensa e a rede tem capacidade de se auto-organizar para otimizar o tráfego dos dados;

- **Árvore (*Cluster Tree*):** este tipo de topologia é considerado um tipo de topologia de **Malha** com maior hierarquia entre os nodos participantes. Neste caso, os **Dispositivos Finais** só podem se comunicar com os **Roteadores**. Estes **Roteadores** podem comunicar entre si e encaminhar a mensagem para um **Coordenador**;

- **Estrela (*Star*):** este tipo de topologia organiza-se e de maneira mais simples. Na topologia **estrela**, existe um **Coordenador** responsável por comunicar com diversos **Dispositivos Finais**. Assim, a rede é altamente dependente deste **Coordenador**. Desta forma, esta topologia é recomendada para ambientes com poucos obstáculos e não muito extensos.

A especificação da arquitetura do *Zigbee* é organizada em uma estrutura de pilhas de blocos denominados de camadas. Cada camada realiza um conjunto de serviços específicos para a camada superior. A interface disponibilizada por uma camada para a camada superior é denominada de Ponto de Acesso de Serviço (*Service Access Point - SAP*). Assim, a Figura 2.3 representa a organização destas pilhas de camada da especificação *Zigbee* (Alliance [2008]).

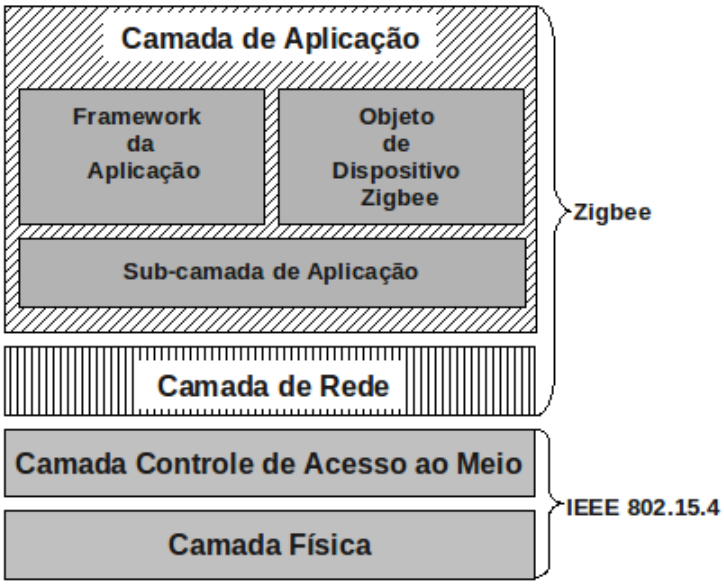

Figura 2.3: Organização das camadas do protocolo de uma rede *Zigbee* (Alliance [2008]).

O *Zigbee* define as camadas de **Aplicação** e de **Rede**. As Camadas de **Controle de Acesso ao Meio** (*Medium Access Control - MAC*) e **Física** (*Physical - PHY*) são definidas pela especificação

IEEE 802.15.4 (Alliance [2008]; IEEE [2006]).

A **Camada de Aplicação** é a camada superior disponibilizada pela especificação *Zigbee*. Esta é constituída pelo ***Framework***, **Objeto de Dispositivo *Zigbee*** e pela **Subcamada de Aplicação**.

O ***Framework*** é o ambiente de execução no qual os objetos da aplicação nos dispositivos *Zigbee* são armazenados. Este ***Framework*** contém bibliotecas com própria especificação e são organizadas em grupos (*clusters*) de atributos e operações. Estes grupos são definidos para cada **Perfil de Aplicação** (*Application Profile*) objetivando a diminuição do tempo de desenvolvimento do projeto, além de permitir desenvolver aplicações distribuídas e interoperáveis (Alliance [2008]).

Os **Objetos de Dispositivo *Zigbee*** (*Zigbee Device Object - ZDO*) representam a classe básica de funcionalidades que disponibilizam uma interface entre os objetos de aplicação, o perfil do dispositivo e a **Subcamada de Aplicação**. Este monta a informação de configuração de aplicações, gerenciamento de rede e de conexões. Estes objetos são definidos pelos fabricantes dos dispositivos (Alliance [2008]).

A **Subcamada de Aplicação** (*Application Support Sub-Layer - APS*) disponibiliza uma interface entre a **Camada de Rede** e a **Camada de Aplicação**. Assim, esta atribui um conjunto de serviços que podem ser usados pelo *ZDO* e objetos definidos pelo fabricante (Alliance [2008]).

A **Camada de Rede** é responsável por gerenciar os dados relacionados ao roteamento e segurança das mensagens. Esta camada suporta três tipos de topologias: malha, árvore e estrela (Alliance [2008]).

A **Camada de Acesso ao Meio** (*Medium Access Control - MAC*) segue a especificação do *IEEE 802.15.4* (IEEE [2006]). Esta camada foi projetada para possibilitar o crescimento da rede sem a necessidade de equipamentos de transmissão muito potentes. Ela é responsável por criar e sincronizar os *beacons* da rede (caso o dispositivo seja um Coordenador), suportar a segurança de comunicação dos dispositivos. Esta camada utiliza o mecanismo *CSMA/CA* (*Carrier Sense Multiple Access with Collision Avoidance*) para evitar as colisões dos pacotes e promover a sincronização dos quadros. A **Camada MAC** pode utilizar ou não o *beacon*. Quando se utiliza (*CSMA/CA* com *slot*), os nodos roteadores periodicamente enviam mensagens de sinalização para tentar confirmar a sua presença para outros nodos roteadores. A vantagem da utilização do *beacon* é que os outros nodos só precisam estar ativos no momento da sinalização, assim, podendo permanecer a maior parte do tempo inativo. Desta forma, neste modo, os nodos economizam mais a energia. A **Camada MAC** também pode não implementar o *beacon* (*CSMA/CA* sem *slot*). Neste modo, como não existe a sinalização do uso do meio, os nodos precisam estar sempre ativos consumindo, desta forma, mais energia (Alliance

[2008]).

A **Camada Física** também é especificada pelo padrão *IEEE 802.15.4* (IEEE [2006]). Este padrão foi desenvolvido para oferecer uma interface de baixo custo de implementação e consumo de energia. Esta camada é responsável por ativar e desativar a antena do rádio, indicar a qualidade do enlace para o recebimento de pacotes, selecionar a frequência do canal e transmitir e receber dados. A **Camada Física** opera na faixa de frequência sem controle governamental (*Industrial Scientific and Medical - ISM*) de 2.4*GHZ* (com 16 canais *ZigBee*, cada canal requer 5*MHz* da largura de faixa), de 915*MHZ* e de 868*Mhz*.

2.5 Organização das RSSF

As RSSF, do ponto de vista organizacional, podem ser consideradas como um tipo especial de *MANETs* (*Mobile Ad hoc Network*). As *MANETs* são um tipo de rede caracterizado por serem constituídas por um conjunto de nodos móveis que são capazes de se comunicar entre si através de canais sem fio, assim, não necessitam de uma infra-estrutura fixa e pré-definida (Sadagopan *et al.* [1999]). Além disto, os enlaces entre os nodos podem ser desfeitos de maneira arbitrária devido à mobilidade dos nodos. Por causa disto, é necessária permanente re-configuração das rotas. Da mesma forma que pode ocorrer com as RSSF, nas *MANETs* os nodos também podem se comportar como roteador de mensagens (Ilyas e Mahgoub [2005]; Loureiro *et al.* [2003]).

Apesar das semelhanças existentes entre RSSF e *MANETs*, existem aspectos que diferenciam uma da outra, tais como (Ilyas e Mahgoub [2005]; Loureiro *et al.* [2003]; Sadagopan *et al.* [1999]):

- Os nodos constituintes de uma *MANET* apresentam recursos computacionais e energéticos mais abundantes se comparados com os nodos de uma RSSF. Assim, os nodos da MANET podem realizar processamento mais complexo, tais como, execução de algoritmos de roteamento e técnicas de segurança mais sofisticados, que necessitem de processamento e memória. Além disto, ele pode ter aplicação de propósito geral. Contudo, as RSSF, por apresentar restrições mais severas de seus recursos (energético e computacional), abordagens que requerem alto grau de processamento e memória podem ser impedidas. As RSSF também são projetadas para uma aplicação específica para um determinado tipo de fenômeno;

- Devido aos recursos energéticos limitados, as RSSF são muito susceptíveis a falhas se comparadas à *MANETs*. Além disto, os tipos de aplicação de RSSF apresentam mais riscos à inte-

gridade físicas dos nodos, devido ao ambiente hostil onde ela pode ser aplicada, por exemplo, aplicações militares;

- Em algumas aplicações de RSSF, pode-se considerar que os nodos sensores apresentam baixo custo de produção e alto custo de manutenção devido à possível complexidade de acesso ao meio onde a rede foi aplicada. Desta maneira, os nodos sensores podem ser considerados descartáveis. Entretanto, nas *MANETs*, normalmente, os nodos são aplicados em regiões de fácil acesso, assim, manutenções podem ser realizadas individualmente em cada nodo.

O ciclo de vida de uma RSSF é formado por cinco etapas, e a qualquer momento é possível ir de uma etapa a outra. As etapas deste ciclo que são: **Configuração**, **Manutenção**, **Sensoriamento**, **Processamento** e **Comunicação**.

O ciclo de vida de uma RSSF começa com a etapa de **Configuração** da rede. As RSSF, normalmente, não apresentam uma infra-estrutura pré-definida, pois os nodos podem ser lançados em uma região de maneira aleatória. Assim, esta rede deve ser capaz de se auto-organizar (*self-organizing*). Nesta etapa, os nodos se despertam para o estabelecimento da rede. A tarefa de localização, se existir, também é realizada nesta etapa. Dependendo dos recursos disponíveis na rede, nesta fase também os nodos podem se agrupar formando grupos (*clusters*).

A etapa de **Manutenção** tem como objetivo prolongar a vida da rede, diminuindo a imprevisibilidade e atender requisitos de aplicação, como estabelecimentos de novas rotas. Isto se torna necessário caso algum nodo falhe por conta do esgotamento de energia ou pela topologia dinâmica onde ele se encontra. Além disto, novas rotas podem ser restabelecidas para realizar uma alternância de nodos roteadores, para assim, distribuir uniformemente o gasto energético entre todos os nodos da rede.

A etapa de **Sensoriamento** consiste em realizar o monitoramento do ambiente e coletar informações do mesmo através de sensores. Assim, o tipo de sensor irá depender do fenômeno que se deseja monitorar, tais como, luminosidade, som, calor, pressão, umidade ou movimentos.

A etapa de **Processamento** pode ser de dois tipos: processamento de suporte ou de informação. O primeiro está relacionado com o processamento funcional dos nodos sensores, como o processamento envolvido no gerenciamento, comunicação e manutenção da rede. O segundo está envolvido com o processamento de coleta de informações do meio, como os dados serão armazenados e transmitidos.

A etapa de **Comunicação** está relacionada com a troca de informações e cooperação de informações entre os nodos sensores. A topologia da RSSF é dinâmica, assim, os enlaces entre os nodos

podem ser desfeitos e refeitos a qualquer momento. Além disto, devido à natureza da comunicação sem fio, esta possui limitações na presença de obstáculos e faixa de alcance. A presença de ruídos ambientais ou interferência de outros nodos sensores próximos também pode afetar a comunicação entre os nodos. Devido a isto, pode ser necessário realizar a retransmissão de mensagens, ocasionando assim, um gasto de energia extra.

2.6 Considerações Finais

Este Capítulo introduziu uma caracterização geral sobre RSSF. Assim, foi descrita a arquitetura básica de uma RSSF, descrevendo os componentes de *hardware* chaves deste tipo de rede, que são: nodos sensores e estação base. Para facilitar o entendimento deste tipo de rede, também foi realizada uma classificação das características desta rede.

Com o objetivo de contextualizar as RSSF no mundo real e analisar algumas aplicações potenciais desta rede, também foram descritas algumas aplicações deste tipo de rede. A especificação do protocolo de comunicação utilizado nas RSSF, de um modo geral, *Zigbee* e *IEEE 802.15.4* foram detalhadas também neste Capítulo. Além disto, foram informadas as diferenças principais entre rede com fio e *MANETs*. Da mesma forma, foram abordadas as etapas do ciclo de vida de uma RSSF.

Capítulo 3

Agentes Móveis

A utilização de agentes móveis em redes cabeadas e redes *ad hoc* sem fio justifica-se em diversos tipos de aplicações. Contudo, sua utilização em RSSF é relativamente nova. Assim, este capítulo tem como objetivo descrever agentes e sua aplicação em RSSF. Em 3.1, é realizada a caracterização de agentes móveis, as vantagens e desvantagens da sua utilização. Além disto, é descrito o modelo de arquitetura de sistemas de agentes e dos próprios agentes. Uma descrição de algumas soluções baseadas em agentes móveis em RSSF é apresentada em 3.2. As considerações finais deste capítulo são realizadas em 3.3.

3.1 Caracterização

Um agente é um programa que age de maneira autônoma. Portanto, este programa pode executar tarefas por iniciativa própria sem a interferência obrigatória de usuários ou de outros programas. Estes agentes podem ser classificados como Estacionários ou Móveis (FIPA [2004]; OMG [1997]).

Os Agentes Estacionários são caracterizados por sempre executarem no local onde eles iniciaram. Assim, caso um agente deste tipo precise de informações de outro sistema ou necessidade de interagir com outro agente, ele usa, normalmente, algum mecanismo de comunicação através de troca de mensagens (OMG [1997]).

Os Agentes Móveis são conhecidos também como Agentes Transportadores, são capazes de se transportar de um sistema a outro através da rede. Esta característica permite-o mover para outro ambiente de agente acessando recursos deste ambiente (OMG [1997]). Este ambiente é denominado neste trabalho como Agência (Silva [1999]).

Os Agentes Estacionários e Móveis apresentam um estado de execução em um determinado momento. Contudo, o agente móvel deve ser capaz de carregar seu código e o seu estado de execução quando este se move de um nodo para outro. Desta forma, este agente é capaz de voltar à operação que realizava no sistema anterior (FIPA [2004]; OMG [1997]).

Os agentes móveis apresentam algumas características que podem ser vantajosas em aplicações que necessitem soluções distribuídas. Algumas destas características são (Lange e Oshima [1999]):

1. **Redução de carga na rede**. Algumas aplicações distribuídas podem necessitar da participação de diferentes estações para cumprir um determinado objetivo. Para isto, pode ser necessário um grande fluxo de trocas de mensagens entre estas estações. Assim, em algumas aplicações, os agentes móveis podem ser usados para reduzir este tráfego realizando um pré-processamento nestes dados, removendo informações redundantes e empacotá-los e encaminhá-los diretamente para a estação destino;

2. **Encapsulamento de protocolos e operação em ambientes heterogêneos**. Nos Sistemas Distribuídos, pode existir a participação de diferentes protocolos de comunicação. Estes protocolos podem apresentar características particulares, por exemplo, de eficiência, segurança ou tipo de licença (proprietária ou não). Além disto, este mesmo sistema pode atuar em diferentes ambientes tanto de *hardware* como de *software*. Assim, a implementação de aplicações nestes casos pode ocasionar em um desprendimento de trabalho e recursos consideráveis para se adaptar a esta heterogeneidade de protocolo e ambiente de execução. Então, a utilização de uma aplicação baseada em agentes móveis pode ser uma abordagem conveniente, pois, nestes agentes, podem ser implementados diferentes protocolos. Além disto, como a execução do agente móvel é dependente apenas de um ambiente próprio (agência), ele pode executar em uma plataforma de *software* intermediária;

3. **Execução assíncrona e autônoma**. Os dispositivos móveis podem ficar, em alguns momentos, desconectados da rede. Os agentes móveis podem ser uma solução atraente para este tipo de situação, pois eles podem embutir determinadas tarefas e operar de maneira assíncrona e autônoma em diferentes nodos. Assim, quando este nodo voltar a se conectar a rede, este agente pode se mover para outro nodo;

4. **Robustez e tolerância a falhas**. Um Sistema baseado em agentes móveis pode tolerar falhas através da duplicação (operação de clonagem) dos agentes em diferentes nodos da rede.

Estas características vantajosas dependem do tipo de aplicação de RSSF e da sua configuração, tanto de *software* como de *hardware*. A redução de carga que pode ser obtida na utilização de agentes

móveis em uma RSSF vai depender de como este agente é implementado. Os agentes móveis também apresentam algumas características desfavoráveis à sua utilização em alguns tipos de aplicações (Jansen *et al.* [1999]).

Devido a estas características desvantajosas, a utilização dos agentes móveis nestes tipos de aplicação pode ser impedida ou dificultada. Desta forma, algumas destas características negativas que os sistemas baseados em agentes móveis podem apresentar são (Jansen *et al.* [1999]):

1. **Segurança**. Em um sistema baseado em agentes móveis, diversas ameaças à segurança podem existir. Estas ameaças podem ser classificadas como: agente-para-agente, agente-para-plataforma, plataforma-para-agente, outro-para-agente. A categoria de ameça agente-para-agente ocorre em situações em que um determinado agente malicioso visa a explorar e atacar as fraquezas de segurança de outro agente. A ameaça agente-para-plataforma ocorre quando um agente malicioso explora as vulnerabilidades da plataforma e aproveita disto para atacá-la. A ameaça plataforma-para-agente age de maneira oposta à ameaça anterior, ocorrendo de plataformas maliciosas em relação ao agente. A ameaça outro-para-agente ocorre quando entidades externas do sistema de agentes (pode ser outro sistema de agente) realizam ataques para uma determinada plataforma de agente. Algumas diferentes estratégias de segurança podem ser implementadas numa plataforma de agentes móveis, tais como: restrição de acesso de nodos, atribuição e controle de privilégios de entidades na rede, assinatura digital de agentes e criptografia das mensagens;

2. **Desempenho e Tamanho do Código**. O desempenho de um sistema baseado em agentes móveis irá depender dos recursos computacionais dos nodos e largura de banda da rede existentes. Além disto, algumas soluções de segurança podem degradar o desempenho da rede ou então não ser aplicável em redes que apresentam recursos computacionais limitados. Também deve existir a preocupação com crescimento do tamanho do agente quando ele trafega de um nodo para outro coletando informações.

Atualmente, o paradigma de sistemas de *software* baseados em agentes tem sido aplicado como uma abordagem em ambientes de Sistemas Distribuídos. Assim, surge a necessidade de padronizar as interfaces para permitir a interoperabilidade destas aplicações e das funcionalidades que caracterizam um sistema de agentes móveis. Portanto, as duas principais instituições que propõem isto são *Object Management Group* (OMG) (OMG [1997]) e *Foundation for Intelligent Physical Agents* (FIPA) (FIPA [2004]).

A OMG propõe uma especificação de padronização para interoperabilidade entre plataformas de agentes móveis de diferentes fornecedores denominada de *Mobile Agent System Interoperability Facility - MASIF*. Esta especificação não define as operações do agente local, tais como interpretação, serialização, deserialização e execução do agente. Contudo, esta especificação visa a definir parâmetros no perfil do agente para especificar requisitos para que um sistema de agentes receba, envie ou execute um agente. Assim, esta especificação visa a padronizar o gerenciamento, transferência, nomes de sistemas de agentes e dos agentes, tipos de sistemas de agentes e sintaxe de localização (OMG [1997]).

A padronização de gerenciamento de agentes visa a estabelecer um modelo de criação de agente e suspensão, retorno e finalização do estado do agente. Assim, através desta padronização das operações de gerenciamento, permite a um administrador gerenciar diferentes sistemas de agentes. Além disto, em uma aplicação de agentes, é desejável que o agente possa mover-se livremente entre sistemas de diferentes tipos. Desta forma, resulta em uma infraestrutura disponível comum para os agentes. Contudo, para alcançar a integração de infraestrutura de diferentes tipos, é necessário que exista um padrão de sintaxe e semântica de nomes de agentes e sistemas de agentes. Isto permite que os sistemas de agentes e agentes se identifiquem, como também os clientes identifiquem agentes e sistemas de agentes. A transferência de um agente só pode ocorrer caso o sistema de agente suporte o agente. A padronização da sintaxe de localização é importante para permitir que os sistemas de agentes possam localizar outros sistemas (OMG [1997]).

A arquitetura de sistemas de agentes móveis proposta pela OMG está representada na Figura 3.1. O **Sistema de Agente** é executado sobre um determinado **Sistema Operacional**. O **Sistema de Agente** deverá realizar o gerenciamento dos recursos que o **Agente** poderá manipular e consumir, como, por exemplo, o acesso aos recursos de comunicação através de uma **Infraestrutura de Comunicação**. O **Ambiente** é a interface entre o **Agente** e o **Sistema de Agente** (OMG [1997]).

A FIPA é uma organização internacional que se dedica a promover a utilização de agentes inteligentes através da especificação de desenvolvimento aberto suportando a interoperabilidade entre agentes e aplicações baseadas em agentes. Além disto, a FIPA define os requisitos mínimos para a mobilidade dos agentes. Assim, a FIPA especifica os serviços e ontologia de gerenciamento dos agentes e, transporte de mensagem da plataforma do agente. O modelo de referência do sistema de gerenciamento de agente é representado na Figura 3.2 (FIPA [2004]).

No modelo de agentes da FIPA, o **Agente** pode comunicar-se com uma **Aplicação** (**Software**). Este **Agente** deve ter um identificador único (*Agent Identifier*) em todo o sistema. Além disto, existe um componente opcional denominado de **Facilitador de Diretório** (***Directory Facilitator***). Este com-

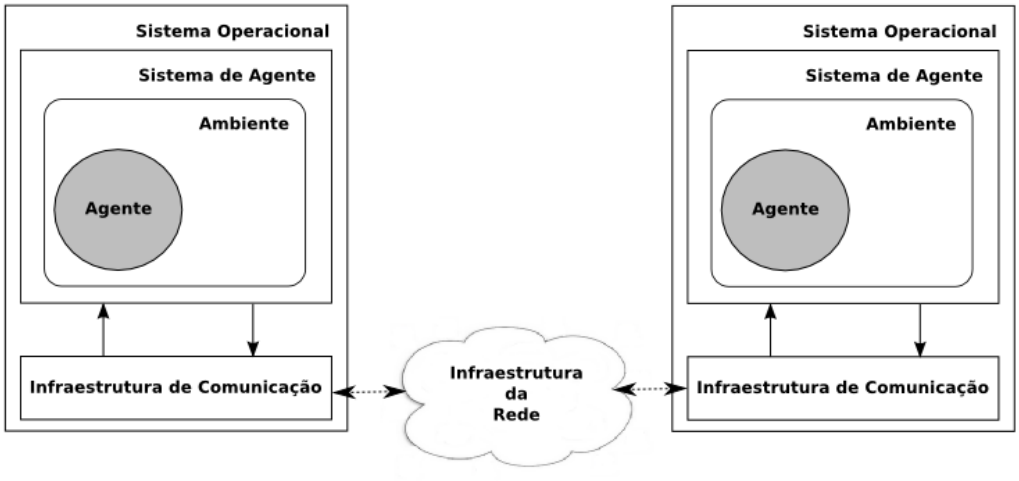

Figura 3.1: Modelo de Sistemas de Agentes (OMG [1997]).

ponente permite aos agentes registrarem seus serviços e consultar outros serviços que são oferecidos por outros agentes. O **Sistema de Gerenciamento de Agente** (SGA) é um componente obrigatório que tem o papel de controlador supervisório sobre o acesso e uso da **Plataforma de Agentes**. O **Serviço de Transporte de Mensagem** (STM) é o componente de comunicação padrão entre agentes em diferentes plataformas de agentes. A **Plataforma de Agentes** provê uma infraetrutura na qual os agentes podem ser empregados. Assim, a **Plataforma de Agentes** é constituída pelas máquinas, sistemas operacionais, aplicação de suporte de agente e os componentes de gerenciamento de agentes do FIPA (FIPA [2004]).

A FIPA também propõe um modelo de estados possíveis de agente. Este modelo é representado na Figura 3.3.

No estado **Ativo**, o **Serviço de Transporte de Mensagem** (**STM**) entrega a mensagem para o agente. Nos estados de **Iniciado**, **Esperando**, **Transiente** e **Suspenso**, o **STM** armazena as mensagens até que o agente retorne ao estado **Ativo** ou encaminhe a mensagem para uma nova localização. Contudo, apenas os agentes móveis podem entrar no estado de **Transiente**, assim, é assegurado que o agente estacionário execute todas suas instruções no nodo em que este foi criado. No estado **Desconhecido**, o **STM** armazena ou rejeita as mensagens, dependendo da política do STM e dos requisitos de transporte das mensagens.

As transições de estado **Cria** e **Invoca** significam, respectivamente, a criação (ou instalação) de um novo agente e invocação do agente. A transição **Destroi** é iniciada pelo Sistema de Agente Móvel e força a finalização do agente de maneira abrupta. A transição **Sair** consiste na finalização normal do agente. O agente entra no estado de **Suspenso** através da transição **Suspende**. Esta transição também

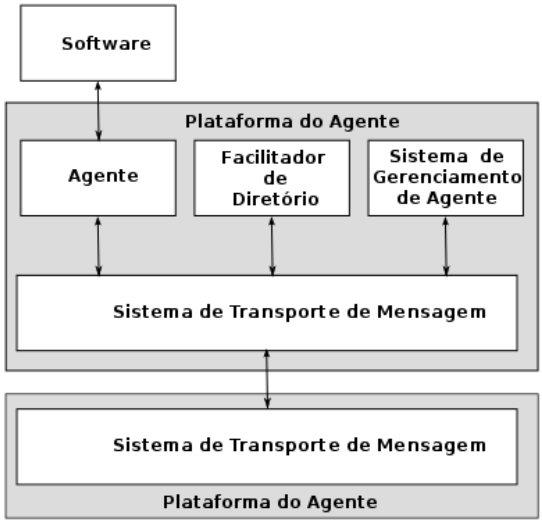

Figura 3.2: Modelo do sistema de gerenciamento de agente da FIPA (FIPA [2004]).

só pode ser iniciada pelo Sistema de Agentes Móveis (SAM). A transição de **Espera** põe o agente no estado de **Esperando**. A transição de **Executa** é executada pelo SAM e põe apenas o agente móvel no estado **Transitório**.

3.2 Aplicações de Agentes Móveis em RSSF

Atualmente, existem alguns sistemas de agentes desenvolvidos ou em desenvolvimento para RSSF. Um destes sistemas é o ***Agilla*** (Agilla [2009]). O ***Agilla*** é um sistema que facilita o desenvolvimento de aplicações baseadas em agentes móveis para RSSF. Os agentes no ***Agilla*** são pró-ativos e são capazes de mover-se e clonar-se para outros nodos. O ***Agilla*** é executado sobre o sistema operacional TinyOS (Berkeley [2004]) e permite que vários agentes executem ao mesmo tempo em cada nodo, dependendo apenas da quantidade de memória disponível.

O ***Agilla*** disponibiliza uma lista de vizinhanças e uma memória lógica compartilhada para comunicação entre agentes no mesmo nodo. A lista de vizinhança disponibiliza os endereços dos nodos vizinhos. Esta informação é importante, pois pode ser usada para auxiliar a decisão do agente nas operações de mover ou de clonar. A memória lógica compartilhada é estruturada como um espaço de tuplas. Esta abordagem permite o desacoplamento da comunicação entre o agente remetente e o agente destinatário. Assim, eles não precisam conhecer a existência do outro para se comunicarem (Agilla [2009]).

Na literatura, existem algumas propostas de protocolos para RSSF baseados em agentes móveis.

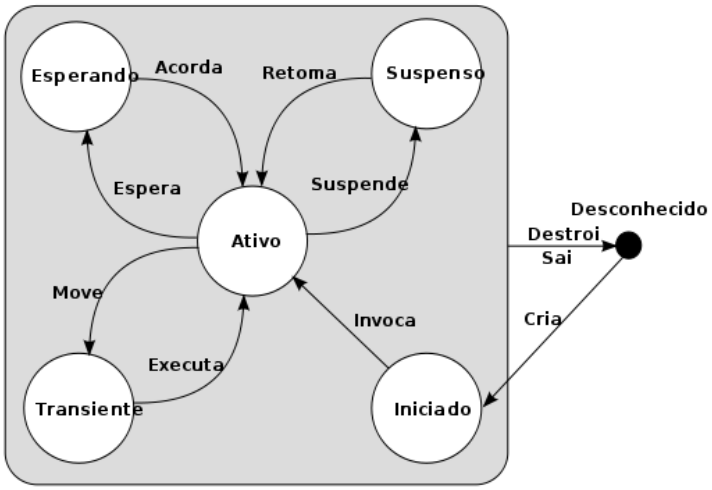

Figura 3.3: Modelo de estado do agente pela FIPA(FIPA [2004]).

A arquitetura *Mobile Agent Based Wireless Sensor Network* (MAWSN) é proposta por (Chen *et al.* [2006]) para a utilização de agentes móveis para remover a redundância de informações sobre um fenômeno em uma determinada região, garantindo um desempenho de energia melhor que o modelo de cliente/servidor. Contudo, esta proposta apresenta alta latência fim-a-fim. Outra proposta de utilização de agentes móveis em RSSF é apresentada em Chen *et al.* [2007] através do MADD (*Mobile Agent-based Direction Diffusion*) (Intanagonwiwat *et al.* [2003]). Neste trabalho, é utilizada a difusão direcionada para determinar um caminho de melhor eficiência (menos saltos) até o sensor alvo. A arquitetura *Sensor Network with Mobile Agents* (SENMA) é proposta por (Tong *et al.* [2003]) para diminuir a redundância de informações em RSSF movendo a complexidade de processamento nos nodos sensores para os agentes móveis. A utilização de agentes móveis para protocolos de localização de caminhos de objetos móveis, seguindo em volta destes objetos é proposta por (Tseng *et al.* [2002]). Em Massaguer [2005], propõe a utilização de agentes móveis e algoritmo genético para traçar planos do caminho deste agente na RSSF para coletar informações monitoradas dentro de um limite de tempo, além de visar, a minimização do gasto energético. Em Qi *et al.* [2003], é proposto a utilização de agentes móveis em RSSF para o processamento de informações colaborativas de múltiplos sensores. Esta proposta visa ser tolerante a falhas e ações maliciosas.

3.3 Considerações Finais

Este Capítulo teve objetivo de realizar uma caracterização de agentes móveis. Para isto, foram definidos os agentes e o ambiente no qual ele executa (agência). Além disto, foram descritos algumas

vantagens e desvantagens na utilização de agentes móveis. Estas vantagens irão depender do tipo de aplicação, pois nem todos os tipos de aplicações estes benefícios estão presentes devido a limite de *hardware* ou restrições do sistema. Algumas destas vantagens também estão relacionadas aos agentes estacionários tais como, encapsulamento de protocolos, possibilidade de operar em ambientes heterogêneos e tolerância a falhas. Contudo, isto não diminui a importância da utilização de agentes móveis, pois a redução de carga que estes tipos de agentes podem realizar é importante em aplicações que apresentam altas restrições de banda de comunicação ou que o custo de comunicação pode ser bastante elevado.

Neste Capitulo, também foram abordados propostas de padronização das interfaces de sistemas de agentes e dos estados dos agentes que são realizados pela OMG e FIPA. Estes padrões visam à interoperabilidade de comunicação entre agentes e integração de diferentes sistemas baseados em agentes móveis. Além disto, foi descrita um sistema de agentes móveis para RSSF denominado de **Agilla**. Também foram descritas algumas propostas de protocolos para RSSF utilizando agentes móveis como foram referenciadas em (Chen *et al.* [2006]), (Chen *et al.* [2007]), (Tong *et al.* [2003]), (Tseng *et al.* [2002]), (Massaguer [2005]) e (Qi *et al.* [2003]).

Capítulo 4

Trabalhos Relacionados

Neste capítulo são descritos alguns ambientes de simulação de redes pesquisados para a elaboração deste trabalho. Para isto, este capítulo está estruturado da seguinte maneira: a descrição do ambiente utilizado como base para elaboração da ferramenta proposta neste trabalho como o *MATLAB*, *Simulink* e o *Truetime* é realizado na seção 4.1. Uma descrição mais sucinta de outras ferramentas de simulação de redes utilizadas pela comunidade científica, tais como *J-Sim*, *NS-2*, *ATEMU*, *TOSSIM* e o *SENSE* e uma comparação entre estas estão presentes na seção 4.2. Por fim, as considerações finais sobre as ferramentas descritas neste capítulo são descritas na seção 4.3.

4.1 Ambiente de Simulação Utilizado

A ferramenta proposta neste trabalho, MASiM (*Mobile Agents SIMulator in wireless sensor network*), foi elaborada sobre um ambiente que disponibiliza um conjunto de funcionalidades. Estas funcionalidades agregam facilidades para o desenvolvimento desta ferramenta. O ambiente base utilizado é denominado de *MATLAB* (*Matrix LABoratory*).

O *MATLAB* foi concebido pelos projetos *LINPACK* (Dongarra *et al.* [1984]) e *EISPACK* (Meyering *et al.* [2009]) e que depois incorporou as bibliotecas BLAS (*Basic Linear Algebra Subprograms*) (Lawson *et al.* [2009]) e LAPACK (*Linear Algebra PACKage*) (Anderson *et al.* [1990]). O *MATLAB* é patenteado pela *The Math Works, Inc*. Este ambiente foi projetado, originalmente, para ser um *software* que permite a manipulação e o acesso de matrizes de maneira simples. Atualmente, este *sofware* também é utilizado para simulação de projetos nas áreas da engenharia, computação e matemática.

O elemento básico de dados da linguagem do *MATLAB* é o vetor. Esta estrutura de dados não

precisa ter uma dimensão limitada, o que permite a resolução de vários problemas de computação técnica, principalmente, operações sobre matrizes e vetores, de maneira mais simples, quando comparado com a utilização de outras linguagens, tais como, *Java*, *C* ou *Fortran* (MathWorks [2009]).

O *MATLAB* é constituído de uma linguagem e de um ambiente de programação, visualização de gráficos e animações. Desta forma, o *MATLAB* permite a representação de problemas computacionais em notação matemática. Atualmente, o *MATLAB* é utilizado para desenvolver algoritmos, aquisição de dados através de simulações, além de permitir realizar modelagem, prototipagem, análise de dados e desenvolvimento de aplicações com interface gráfica para o usuário (MathWorks [2009]).

O ambiente *MATLAB* é constituído, principalmente, das seguintes partes (MathWorks [2009]):

- **Ferramenta de área de trabalho e ambiente de desenvolvimento**. Conjunto de ferramentas que torna mais fácil e produtiva a utilização de funções do *MATLAB* e manipulação de arquivos. A maioria destas ferramentas apresenta uma interface gráfica com o usuário tais como: área de trabalho, janela de comando, editor, depurador e analisador de código, navegador que facilita a manipulação de variáveis de *workspace* e arquivos;

- **Bibliotecas com funções matemáticas**. Contém uma grande quantidade de algoritmos estendendo funções elementares da aritmética, geometria, operações com matriz, entre outras;

- **A Linguagem**. Linguagem de alto nível para manipulação de matriz/vetores com estrutura de controle de fluxo, funções, estrutura de dados, entrada/saída e características de Programação Orientada a Objeto. Portanto, isto permite desenvolver tanto programas pequenos rapidamente sem a preocupação com o reuso, como também, programas de grande porte para aplicações mais complexas que necessitam de um planejamento para seu reuso;

- **Gráficos**. O *MATLAB* permite mostrar vetores e matrizes como gráficos, além de anotações com comentários sobre um determinado gráfico. Além disto, também inclui funções de alto nível para visualização de dados de duas ou três dimensões, processamento de imagens, animações e gráficos de apresentação. Como também funções de baixo nível que permitem personalizar completamente os gráficos e construir interfaces gráficas completas para o usuário nas aplicações do *MATLAB*;

- **Interfaces externas**. A biblioteca de interface externa permite elaborar programas em *C++* e *Fortran* para interagir com o *MATLAB*. Isto inclui rotinas do *MATLAB* (ligação dinâmica) para interação e manipulação (leitura e escrita) de arquivos *M-Files*.

O ambiente *MATLAB* apresenta um conjunto de bibliotecas de soluções para aplicações específicas denominadas de *toolbox*, que podem ser desenvolvidas em linguagem do *MATLAB* ou em *C++ MEX* (*Matlab EXecutable*). O *C++ MEX* é uma linguagem baseada no *C++ ANSI* (Ansi [1995]) que estende funcionalidades especificas (manipulação de matrizes) disponibilizadas pelo ambiente *MATLAB*. Além das bibliotecas nativas do ambiente, o *MATLAB* permite a instalação e utilização de outros *toolboxes*. As bibliotecas utilizas para a elaboração deste trabalho foram *Simulink* (Simulink [2009]) e o *TrueTime* (Ohlin *et al.* [2007]).

O *Simulink* é um pacote que permite modelar, simular e analisar aplicações denominadas de Sistemas Dinâmicos. Os Sistemas Dinâmicos podem ser considerados simplesmente como sistemas nos quais a saída muda com o tempo. Desta forma, o *Simulink* pode ser usado para explorar o comportamento de Sistemas Dinâmicos tais como, circuitos e sistemas elétricos, mecânicos e termodinâmicos, absorção de choques e entre outros. Além disto, permite simular protocolos de comunicação, controle, processamento de sinal, processamento de vídeo e imagem (Simulink [2009]).

As principais características do *Simulink* são (Simulink [2009]):

- biblioteca com blocos pré-definidos extensíveis e expansíveis;
- interatividade com um editor gráfico que permite montar e gerenciar projetos através dos diagramas de blocos;
- capacidade de gerenciar projetos complexos através de modelos organizados em hierarquia de componentes de projeto;
- possibilidade de navegar, criar, configurar e procurar em modelos os sinais, parâmetros, propriedades e códigos gerados e associados com o modelo desenvolvido;
- interfaces de Programação de Aplicação (*Application Programming Interfaces - API*) permitem conectar com outros programas de simulação. Isto é possível através da incorporação dos códigos escritos em outras linguagens ou com a própria linguagem do *MATLAB*;
- modelo de simulação normal, acelerado e super acelerado para executar simulações de código interpretado ou em código *C* com passo variável ou passo fixo;
- depurador gráfico para examinar os resultados das simulações e diagnosticar o desempenho e comportamento não esperado do projeto, assegurando a consistência do modelo;
- acesso completo ao *MATLAB* para analisar e visualizar resultados, personalizar modelagens ambientais, definição de sinais, parâmetros e testes de dados.

O *Simulink* é um *software* de múltiplos domínios de modelagem. Devido a isto e suas características citadas anteriormente, existem outras bibliotecas que expandem o *Simulink*. Uma destas bibliotecas utilizadas neste trabalho é o *TrueTime* (Ohlin *et al.* [2007]).

O *TrueTime* é um simulador de sistemas de controle em tempo real. O *TrueTime* facilita a co-simulação de controladores de execução de tarefa e *kernel* de tempo real, transmissão de rede, dinâmicas plantas contínuas (Ohlin *et al.* [2007]).

As principais características do simulador são (Ohlin *et al.* [2007]):

- escrito em *C++ MEX* (*Matlab EXecutable*);
- simulação baseada em evento;
- interrupção externa;
- possibilidade de escrever tarefas em funções de *M-Files* ou *C++*. Isto também possibilita a chamada de diagramas de blocos dentro de código das funções;
- bloco de rede com fio tais como, *Ethernet*, *CAN* (*Controller Area Network*), *TDMA* (*Time Division Multiple Access*), *FDMA* (*Frequency Division Multiple Access*), *Round Robin* e rede *switched ethernet*;
- blocos de rede sem fio tais como, *802.11b WLAN (Wireless Local Area Network)* e *Zigbee (802.15.4 sem beacon)*;
- modelo de carga e consumo de energia, escala de voltagem dinâmica e relógios locais;
- blocos de interface de redes *stand-alone*.

O *toolbox TrueTime* disponibiliza um conjunto de funções e blocos para programação das simulações. Os blocos do *TrueTime* são mostrados na Figura 4.1 os quais são um bloco de *kernel* (*TrueTime Kernel*), quatro blocos de redes (*TrueTime Network*, *Wireless Network* e *ttGetMsg/ttSendMsg*) e um bloco de bateria (*Truetime Battery*) (Ohlin *et al.* [2007]).

O bloco de *kernel* representa um *kernel* de tempo real genérico com entradas e saídas Analógico/Digital – Digital/Analógico e interfaces de rede. Este executa as tarefas definidas pelo programador e manipuladores de interrupção e também suporta várias políticas de escalonamento e as primitivas de tempo real (Ohlin *et al.* [2007]).

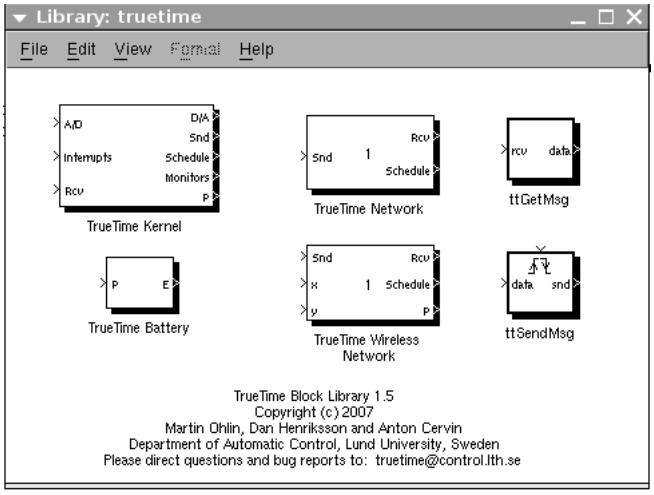

Figura 4.1: Blocos do *TrueTime* (Ohlin *et al.* [2007]).

O *TrueTime* permite programar de diferentes maneiras a mesma aplicação de tempo real, contudo, cada tarefa ou manipulador de interrupção em uma aplicação deve ser implementado em uma função de código. Cada função de código é chamada repetidamente pelo *kernel* durante a simulação. O tempo de execução simulado de cada código é retornado pela função do código. Há três opções de programação, estas são citadas na ordem de melhor desempenho: *C++ MEX*, código *MATLAB* e diagrama de blocos do *Simulink*. O código é dividido dentro de um ou mais segmentos. O código de cada segmento é executado de maneira não preemptável. Os segmentos múltiplos são usados para simular: espera de entrada e saída, auto-suspensão, espera (por eventos, semáforos, monitores e *mailboxes*), laços (*loops*) e condicionais (*branches*). A Figura 4.2 representa a execução dos seguimentos do código (Ohlin *et al.* [2007]).

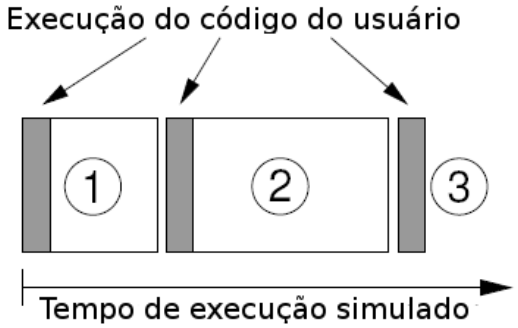

Figura 4.2: Segmentos de sequência do código do usuário (Ohlin *et al.* [2007]).

A execução de uma tarefa e de manipuladores de interrupções é definida por funções de código no *TrueTime*. Estas funções de código são divididas dentro de segmentos de código como são repre-

sentadas na Figura 4.2. Desta maneira, toda a execução do código do usuário é realizada no inicio de cada segmento de código. O tempo de execução de cada segmento deve ser retornado pela função de código (Ohlin *et al.* [2007]).

No *TrueTime*, o modelo da bateria, ilustrado na Figura 4.1, é um simples integrador que pode ser carregado ou descarregado. Os consumidores de energia podem ser computação, transmissão de rádio, a utilização de sensores e atuadores (Ohlin *et al.* [2007]).

4.2 Ambientes de Simulação Relacionados

O *J-Sim* (*JavaSim*) (Sobeih *et al.* [2005]) é uma ferramenta de simulação de RSSF com uma arquitetura de *software* baseada em Componentes Autônomos (*Autonomous Component Architecture - ACA*) desenvolvido em linguagem *Java*. No ACA, as entidades básicas são os componentes. Tais componentes comunicam entre si através da troca de dados (mensagens) pelas portas de comunicação dos componentes. O comportamento da troca destes dados deve obedecer a regras, denominadas de contratos, para integração entre os componentes envolvidos (JSim [2005]). O *J-Sim* disponibiliza uma arquitetura de componentes fracamente acoplados, ou seja, os componentes podem ser projetados, implementados e testados independentemente um do outro. Além disto, devido à separação de *software* e *hardware*, o *J-Sim* permite que novos componentes sejam inclusos de maneira *plug-and-play*, desta maneira, novos componentes podem ser adicionados em tempo de execução.

Esse simulador possui componentes de redes sem fio, cabeadas e redes de sensores sem fio. O *J-Sim* também apresenta uma interface para inserir comandos que integram diferentes linguagens tais como, *Perl*, *Tcl* e *Pyton*. Além disto, este apresenta modelo de bateria, processador, rádio, gerador de fenômenos e o protocolo da camada de enlace/MAC da rede sem fio é o *IEEE 802.11*.

O *NS-2* (Fall e Varadhan [2008]; Xue *et al.* [2007]) é um simulador de eventos discretos aplicado em pesquisa de rede. O *NS-2* dispõe de um suporte considerável para simulação de protocolos TCP, roteamento e *multicast* sobre redes com fio e sem fio (local ou satélites). O *NS-2* iniciou-se da variante do projeto *REAL network* (Keshav [1997]) em 1989. Através do *NS-2*, é possível realizar pesquisas sobre protocolos de comunicação e estudos sobre tráfegos de mensagens. Este é um projeto *open-source* que permite acesso a todos os níveis de detalhamento da ferramenta. A programação da simulação de mais alto-nível, tal como definição da topologia e organização da rede, é realizada através da linguagem *OTcl* (*frontend*). Entretanto, a programação de mais baixo-nível, como programação das aplicações em cada estação e dos protocolos de comunicação, é realizada com a linguagem *C++*. Além disto, *NS-2* pode vir em pacotes separados ou em um único pacote (*all-at-one*). Entre

outros pacotes, o NAM (*Network Animator*) (Estrin *et al.* [2000]) permite gerar visualização da execução da simulação. O *NS-2* possui modelos de tráfego e aplicações (*Web*, *File Transfer Protocol* - FTP, *Telnet*) e protocolos de transporte *unicast* (*Transmission Control Protocol - TCP*, *User Datagram Protocol - UDP*) e *multicast* (*Scalable Reliable Multicast - SRM*).

O *NS-2* não é um projeto finalizado e existem contribuições em andamento de pesquisadores e desenvolvedores. Portanto, falhas no *software* ainda são descobertas e corrigidas. Devido a isto, os usuários do *NS-2* são responsáveis por verificar se suas simulações não sofrem interferência e nem são invalidadas por estas falhas (Keshav [1997]).

O *ATEMU* (*ATmel EMUlator*) (Polley *et al.* [2004]) é um simulador de redes de sensores de alta precisão e de plataforma escalável, que pode ser utilizado antes da aplicação da pesquisa em uma RSSF real. Além disto, o *ATEMU* também disponibiliza uma interface gráfica para o usuário que facilita a programação e depuração das aplicações de RSSF. Esta interface é denominada de *XATDB*. O *XATDB* permite também monitorar operações de um nodo sensor individual instrução por instrução. A alta precisão do *ATEMU* deve-se à simulação das operações dos nodos sensores individualmente e a transmissão de cada nodo com outro. O *ATEMU* adota a arquitetura padrão do *MICA2* (Crossbow [2009]) e microprocessador AVR (*Advanced Virtual RISC*), mas também é capaz de simular redes de sensores heterogêneas. Este também é capaz de simular as operações de baixo-nível de cada nodo sensor de maneira individual. A simulação é feita com os componentes do nodo tais como: processador, relógio e interface de rádio. Além disso, este simulador apresenta um modelo extensível do "ar"para simular operações sem fio.

O *TOSSIM* (*TinyOS SIMulator*) (Levis *et al.* [2003]) é um simulador de eventos discretos de alta precisão e escalável para centenas de nodos para RSSF e sistema operacional *TinyOS*[1]. A principal característica deste simulador é que aplicações programadas nele não necessitam sofrer nenhuma alteração no código para ser executado no sistema operacional *TinyOS*. Assim, o *TOSSIM* pode simular centenas de nodos executando aplicações completas. Contudo, este não garante que tais aplicações nos nodos reais irão executar de maneira idêntica quando foram simulados. Os erros na rede podem ser simulados através da definição da probabilidade de eles acontecerem.

A arquitetura do *TOSSIM* é constituída por: suporte de compilação em grafos de componentes *TinyOS* dentro da infraestrutura de simulação, filas de eventos discretos, um conjunto de componentes do *TinyOS* reimplementado, mecanismo para estender o modelo de rádio e do Conversor Analógico Digital (*Analog-Digital Converter - ADC*) e serviços de comunicação para programas externos para interação com o simulador. A fila de eventos faz parte do núcleo deste simulador. As interrupções

[1] http://www.tinyos.net

são modeladas através de eventos diferentemente de como ocorre no *TinyOS* realmente. Um evento dispara um manipulador de interrupção do componente de abstração de *hardware*. Além disto, o *TOSSIM* simula o comportamento no nível de *hardware* tais como: o ADC, relógio, transmissor de potência variável, memória EEPROM, componentes de sequência de inicialização (*boot*) e diversos componentes da pilha de protocolos da comunicação sem fio (Levis *et al.* [2003]).

Essa ferramenta também disponibiliza mecanismos que permitem ao desenvolvedor definir o nível de complexidade do modelo de rádio necessário para uma determinada simulação. Contudo, a versão atual deste simulador não modela o consumo energético (Levis *et al.* [2003]).

O *TOSSIM* também disponibiliza uma interface gráfica. A rede é organizada em um grafo direcionado, sendo que os nodos são os vértices e cada aresta é o enlace entre dois nodos para a qual pode ser definida uma probabilidade de erro. Além disto, o *TOSSIM* disponibiliza uma interface de *socket* TCP que permite monitorar ou atuar na simulação remotamente.

O *SENSE* (*SEnsor Network Simulator and Emulator*) (Chen *et al.* [2004]) é um simulador de eventos discretos para rede de sensores que visa à eficiência, facilidade e poder de simulação. As principais características do projeto de arquitetura deste simulador são: extensibilidade, reusabilidade e escalabilidade. A extensibilidade é alcançada através de sua arquitetura baseada no modelo de *Component-Port*. No modelo *Component-Port*, um componente se comunica com outro através de portas de entrada (*inports*) e de saída (*outports*). A implementação dos *inports* pode ser considerada de maneira semelhante aos parâmetros de entrada de uma função. Contudo, os *outports* são considerados uma abstração de um ponteiro para função, definindo, assim, qual a funcionalidade esperada por outro componente. Assim, através deste modelo, componentes antigos podem ser trocados de maneira simples por componentes novos, caso estes apresentem interfaces compatíveis. Portanto, usuários avançados podem desenvolver módulos de simulação para necessidades específicas (Chen *et al.* [2004]).

Devido à independência dos componentes do simulador *SENSE*, a reusabilidade é alcançada, estendendo funcionalidades de módulos já disponíveis. A escalabilidade é obtida através da paralelização da simulação entre componentes compatíveis (Chen *et al.* [2004]).

O simulador *SENSE* apresenta como camada de enlace/MAC da rede sem fio o padrão *IEEE 802.11* com *DCF* (*Distributed Coordination Function*). Além disto, este simulador implementa um modelo simples de bateria, que consiste de um consumo linear da mesma. Outro modelo de bateria é o componente mais realístico, em que a carga torna-se dependente da corrente (Chen *et al.* [2004]).

A Tabela em 4.1 apresenta uma comparação resumida das características das ferramentas de

simulação descritas neste Capítulo. Assim, alguns critérios de comparação foram: modelo de consumo de energia, tipo de protocolo suportado na camada de enlace/MAC, arquitetura do *software*, possibilidade de nodos móveis, linguagens de especificação das simulações, plataformas de Sistema Operacional para execução e tipo de licença do *software*. Estas ferramentas não apresentam nenhum recurso específico para a simulação de agentes móveis. Contudo, a simulação de agentes móveis é possível através da elaboração de uma extensão ou biblioteca com este objetivo nestas ferramentas.

Ferramenta	**MATLAB**[1]	**JSim**	**NS2**	**ATEMU**	**TOSSIM**	**SENSE**
Modelo de consumo energético	Sim	Sim	Sim[2]	Não	Não	Sim
Camada MAC e física da rede sem fio	IEEE 802.11 e 802.15.4	IEEE 802.11	IEEE 802.11 e 802.15.4[2]	IEEE 802.15.4	IEEE 802.11	IEEE 802.11
Mobilidade dos Nodos	Sim	Sim	Sim	Sim	Não	Sim
Linguagem de especificação	MATLAB e C++ MEX	Perl, Tcl, Python	OTcl (*frontend*) e C++ (*backend*)	C	Pyton e C++	C++
Plataforma de SO	Windows e Linux	Windows e Linux	Windows e Linux	Linux[3]	Linux	Windows e Linux
Tipo de licença do *software*	*Copyright*	Livre para uso acadêmico	GPL[4]	BSD[5]	*Copyright*	Livre para uso acadêmico

Tabela 4.1: Comparação das ferramentas de simulação.

4.3 Considerações Finais

Este Capítulo realizou uma descrição de ambientes de simulação. Assim, foram descritos as características e recursos disponíveis nas ferramentas *MATLAB*, *Simulink* e *TrueTime*. Através destas informações é possível justificar a utilização destas ferramentas como ambiente base para o desenvolvimento da ferramenta proposta neste trabalho, o *MASiM*.

[1] MATLAB, Simulink e Truetime juntos.
[2] Existe bibliotecas não oficiais que estendem estas funcionalidades.
[3] Linux Redhat.
[4] GNU General Public License.
[5] Berkeley Software Distribution.

Os requisitos para um ambiente base para o *MASiM* são: possibilidade de suporte à camada enlace/MAC do *IEEE 802.15.4*, um modelo de consumo de energia e possibilidade de nodos móveis. Estes requisitos são muito importantes para realizar simulações de uma RSSF mais realista. O *JSim* não suporta o protocolo *IEEE 802.15.4*. A desvantagem do *ATEMU* é que ele não possui um modelo de consumo de energia. A ferramenta *TOSSIM* não apresenta o modelo de consumo energético e de mobilidade dos nodos, além de não suportar o protocolo *IEEE 802.15.4*. O simulador *SENSE* só suporta o protocolo *IEEE 802.11*. Assim, entre as ferramentas analisadas, apenas o *MATLAB* e o *NS-2* satisfizeram requisitos necessários.

O *NS-2* é, provavelmente, uma das ferramentas de simulação de redes mais utilizadas no meio de pesquisa sobre redes. Contudo, a tarefa da sua utilização tornou-se árdua e demasiadamente complexa, pois a integração e configuração de algumas bibliotecas de terceiros para permitir o suporte das características de RSSF não foi possível integralmente. Além disto, os componentes de software que permitem a utilização do protocolo *IEEE 802.15.4* não foram inteiramente homologados e erros ainda ocorrem. O *MATLAB* atende todos os requisitos esperados para ser utilizado como ambiente base da ferramenta proposta, *MASiM*. Por causa disto, o *MATLAB* com os *toolboxes Simulink* e *TrueTime* foram escolhidos para a elaboração deste trabalho.

Capítulo 5

Ferramenta para Simulação de Agentes Móveis

Este Capítulo descreve a ferramenta proposta neste trabalho para simular agentes móveis em uma RSSF denominada de *MASiM*. Esta ferramenta apresenta um conjunto de recursos com o objetivo de possibilitar a especificação e execução destes cenários de simulação.

Assim, a definição do modelo considerado na ferramenta, tais como o modelo de nodos, RSSF, de tarefas e de falhas é apresentada em 5.1. A especificação do projeto da ferramenta *MASiM*, que inclui o levantamento de requisitos, a representação dos componentes de *software* envolvidos, a definição dos protocolos de mobilidade e clonagem, como também a representação das principais classes envolvidas no projeto, estão presente em 5.2. A organização das tarefas, que são executadas nos nodos para simular a operação de agentes móveis em RSSF, é descrita na seção 5.3. A especificação dos procedimentos de configuração para realizar a simulação de um determinado cenário é descrita em 5.4. As considerações finais deste Capítulo estão presentes em 5.5.

5.1 Modelo Considerado na Ferramenta

Esta seção define o modelo considerado para elaboração da ferramenta proposta neste trabalho. Desta forma, em 5.1.1, é descrito o modelo dos nodos, classificando os nodos e o comportamento deles. Em 5.1.2, é definido o modelo de RSSF, informando o protocolo da camada de enlace e como esta rede é organizada. O modelo de tarefas é especificado em 5.1.3 e os tipos de falhas considerados são descritos em 5.1.4.

5.1.1 Modelo dos Nodos

O modelo de nodos considerado neste trabalho assume que todos os nodos da rede são capazes de monitorar variáveis físicas do ambiente e de transmitir os valores destas variáveis através de um enlace sem fio. Também é considerado que cada nodo tem a capacidade de obter informações sobre seu próprio nível energético (Raghunathan *et al.* [2002]; Yi e Chakrabarty [2003]), sua localização geográfica e valores de tempo, os quais estão sincronizados com os valores de outros nodos. Além disto, os valores do tempo também são monotônicos crescentes, ou seja, seu valor sempre cresce de maneira linear (Hadzilacos e Toueg [1994]).

A capacidade de obter seu valor energético está presente na maioria dos nodos sensores atuais. No entanto, neste trabalho, não são tratados os meios de se obter o sincronismo de relógios e localização dos nodos (uma possibilidade é a presença de nodos com GPS que são comercializados atualmente).

O modelo considera dois tipos de nodos: nodos sensores e estação base. Há diferenças na arquitetura de *hardware* e *software* entre estes dois tipos de nodos. O alcance do sinal de rádio é maior das antenas dos nodos estação base comparado com o alcance dos nodos sensores. Em alguns casos, o sinal de rádio do nodo, estação base, pode alcançar todos os nodos da rede em certa área. Os nodos estação base também possuem poder de processamento e de armazenamento comparável a de um computador pessoal ou *laptop*. Diferentemente dos nodos sensores, os nodos estação base não apresentam restrições consideráveis da capacidade energética. Além disto, é considerada a possibilidade da existência de um ou mais nodos estação base na mesma rede (Alsalih *et al.* [2007]).

Neste modelo, também se assume que tanto os nodos sensores como a estação base podem mover-se no ambiente, seja por decisão própria ou por ação do meio externo, onde estes estão presentes (Alsalih *et al.* [2007]; Hu e Evans [2004]). Devido a isto, os enlaces dos nodos podem ser desfeitos ou refeitos dinamicamente.

5.1.2 Modelo da RSSF

No modelo de rede proposto, os nodos são organizados em uma rede em malha (*mesh*) como definido pelo padrão da Zigbee[1]. A camada física e a subcamada MAC estão em conformidade com a especificação do padrão *IEEE 802.15.4* sem *beacon* (IEEE [2006]). Portanto, não há formação de *superframes*, e os nodos utilizam o protocolo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) quando desejam transmitir suas mensagens.

[1] http://www.zigbee.org

Além disto, a topologia da RSSF é dinâmica (Cayirci *et al.* [2002]) pois os canais de comunicação entre os nodos podem ser desfeitos permanentemente ou refeitos de maneira não pré-determinada. Este dinamismo busca modelar situações onde ocorre a mobilidade dos nodos, falhas no canal de comunicação (possivelmente devido a colisões) ou falhas em nodos (provavelmente, devido a esgotamento da energia do nodo).

O modelo de rede considerado neste trabalho consiste em uma rede sem fio constituída por nodos homogêneos, com exceção da estação base. Estes nodos apresentam a mesma configuração de *hardware* de comunicação e de *software* básico. Os enlaces formandos entre os nodos são considerados simétricos, ou seja, o enlace $A \rightarrow B$, ligando o nodo A ao nodo B, é considerado exatamente igual ao enlace $B \rightarrow A$. Contudo, os enlaces entre os nodos sensores e estações bases são considerados assimétricos, pois é considerado que a estação base alcança todos os nodos em uma área, mas os nodos sensores podem não alcançar a estação base diretamente.

Os nodos da rede podem ser cuidadosamente colocados em suas posições, ou podem ser dispostos no ambiente de maneira aleatória. Desta forma, não há, necessariamente, uma infraestrutura pré-determinada. Em ambos os casos, os nodos podem ser dispostos no ambiente de tal forma que o nodo estação base possa não ser alcançado por um único salto. Devido a isto, os nodos sensores podem não alcançar em único salto todos nodos em uma região, mas apenas os que estão em sua volta. Estes nodos alcançáveis em único salto são denominados de nodos vizinhos. Assim, para a maioria dos nodos sensores, a comunicação direta com a estação base ocorre em um único sentido, ou seja, a estação base consegue se comunicar com os nodos sensores, mas estes não conseguem se comunicar diretamente com a estação base. A comunicação direta com o nodo estação base em ambos os sentidos ocorre apenas entre os nodos sensores mais próximos à estação base.

Também se assume no modelo adotado que cada nodo é identificado unicamente (endereço) em toda a rede (Cayirci *et al.* [2002]) e é ciente de quais nodos ativos são seus nodos vizinhos. Este conhecimento é possível, pois os nodos difundem periodicamente a sua localização para todos os nodos vizinhos. Além da sua localização, os nodos sensores podem também enviar o seu nível energético nesta mesma mensagem.

5.1.3 Modelo das Tarefas

O sistema de *software* dos nodos é constituído por um conjunto de tarefas que executam diferentes papéis, tais como receber ou enviar mensagens e processar um agente. Uma tarefa é formada por um conjunto de unidades de trabalho, denominadas de *jobs*, que são executadas e escalonadas

no sistema para acessar recursos. Estes recursos podem ser, por exemplo, a rede ou a memória (Liu [2000]). Assim, como o acesso a estes recursos pode ocorrer de maneira concorrente, as tarefas são escalonadas por um processador (é considerada a existência de apenas um único processador) para ter o acesso a um determinado recurso.

O escalonamento das tarefas é orientado a prioridade. Um algoritmo de prioridade pode ser de dois tipos: *fixo* ou *dinâmico*. Desta maneira, um algoritmo de *prioridade fixa* atribui para todos os *jobs* em cada tarefa a mesma prioridade. Contudo, o algoritmo de *prioridade dinâmica* atribui diferentes prioridades para os *jobs* individuais em cada tarefa (Liu [2000]).

As tarefas neste trabalho podem ser de dois tipos: *periódicas* ou *aperiódicas*. As tarefas periódicas são executadas de maneira regular e o período e o tempo de execução destas tarefas são conhecidos pelo sistema (Liu [2000]). As tarefas aperiódicas são executadas através de eventos internos (eventos de outras tarefas) ou externos (interrupção de rede, E/S) que podem ocorrer em intervalos aleatórios (Liu [2000]).

Além disto, as tarefas do sistema são preemptáveis. As tarefas preemptáveis são aquelas que podem ter a sua execução do *job* interrompida por *jobs* de outras tarefas mais urgentes (maior prioridade). Assim, quando o *job* da tarefa mais urgente for completado, o escalonador de tarefas retorna o processo para a execução do *job* da tarefa anterior a partir do ponto de suspensão (Liu [2000]).

O modelo não considera o *overhead* gerado pela troca de contexto entre as tarefas. Esse valor de tempo se for significativo, deverá ser adicionado aos tempos de execução de cada tarefa. As tarefas podem compartilhar informações entre si através de trocas de mensagens em memória compartilhada. Além disto, estas tarefas apresentam uma validade temporal para serem executadas (*deadline*) e tempo de execução.

5.1.4 Modelo de Falhas

Em uma rede de computadores, de um modo geral, processos ou canais de comunicação podem falhar. Neste trabalho, os processos podem ser considerados como os nodos que compõem a RSSF. Uma falha pode ser definida como um desvio de um comportamento correto ou desejável (Hadzilacos e Toueg [1994]). Em uma RSSF, falhas podem ocorrer tanto nos nodos como nos canais de comunicação. Os fatores externos, ou ambientais, como também fatores internos da rede podem ser responsáveis pela ocorrência de falhas no sistema. Os nodos sensores podem falhar devido a uma programação errônea acidental ou maliciosa de um dos nodos.

Além disto, um determinado nodo pode sofrer falhas devido ao esgotamento de sua bateria, ocasionando falhas intermitentes na comunicação, ou falhas permanentes quando este esgotamento do seu nível energético for completo. Também falhas nos canais de comunicação podem surgir devido a colisões de mensagens enviadas ao mesmo tempo pelos nodos da rede, ou devido a ruídos ambientais que podem interferir na comunicação. Estes canais também podem ser desfeitos quando nodos movem-se no ambiente, saindo, assim, do raio de alcance do sinal de rádio dos seus nodos vizinhos. Desta forma, as falhas podem ser classificadas como *falhas por omissão*, *falhas de temporização* e *falhas arbitrárias* (Hadzilacos e Toueg [1994]).

A *falha de omissão* ocorre quando o nodo ou canal de comunicação deixa de executar uma ou mais ações quando se esperaria que os fizesse. Quando este tipo de falha ocorre com um nodo, é chamado de *parada por falha*, também conhecido como *crash*, e este nodo deixa de executar repentinamente e permanece neste estado. Neste caso, em sistemas síncronos, outros nodos podem detectar a ocorrência desta falha através de mecanismo de *timeout*. Este mecanismo consiste em determinar um tempo limite para que uma determinada ação ocorra. A falha de omissão ocorre no canal de comunicação quando mensagens que deveriam ser entregues são perdidas neste canal. Isto pode acontecer no momento de envio (*falhas por omissão de envio*), no momento de recebimento (*falhas por omissão de recepção*) da mensagem ou no meio de comunicação (*falhas por omissão no canal*) (Hadzilacos e Toueg [1994]).

Já as *falhas arbitrárias* ou também conhecidas como *falhas bizantinas* também podem ocorrer tanto nos processos como também nos canais de comunicação. Desta forma, um processo sofre falha bizantina quando este omite de maneira arbitrária passos esperados do processamento ou efetua algum processamento indesejável. Esta falha ocorre nos canais de comunicação quando, por exemplo, o conteúdo de uma mensagem é corrompido, mensagens inexistentes são enviadas ou mensagens existentes são entregues mais de uma vez (Coulouris *et al.* [2007]; Hadzilacos e Toueg [1994]).

As falhas de omissão e arbitrárias podem ocorrer nos canais de comunicação e nos nodos. Os nodos apresentam relógios sincronizados entre si, desta forma, as falhas de sincronização só podem existir nos canais de comunicação. Assim, as falhas de sincronização nos canais de comunicação se devem a colisões de mensagens enviadas pelo canal sem fio. Uma possível solução para tolerar este tipo de falha é a implementação de um mecanismo de *timeout*. Através deste mecanismo, é determinado um tempo de espera para o recebimento de uma mensagem. Caso este tempo seja ultrapassado, é considerado que houve uma falha de recebimento da mensagem. Neste trabalho, quando uma agência não consegue enviar uma determinada mensagem, esta irá realizar uma quantidade de tentativas pré-definidas. Caso ela não receba uma mensagem de resposta da agência destino, é considerado que

houve alguma falha no canal ou no nodo destino.

5.2 Especificação da Ferramenta

Nesta seção, são descritos os detalhes de projeto da ferramenta *MASiM*. Desta forma, são apresentados os requisitos, os componentes de *softwares* envolvidos, bem como suas interfaces. Além disto, é detalhada a modelagem do comportamento dos protocolos de mobilidade e de clonagem do agente e é feita uma breve descrição das principais classes do projeto.

5.2.1 Requisitos da Ferramenta

A especificação e definição de quais requisitos um *software* deve atender são importantes para determinar o seu escopo. Neste trabalho, os requisitos são classificados como: funcionais (característica, capacidade e segurança do sistema) e não funcionais (usabilidade, ser confiável, desempenho, facilidade de suporte, implementação, interface, operações, empacotamento, questões legais) (Larman [2001]). Os requisitos funcionais considerados no *MASiM* são:

1. O sistema deve permitir ao usuário especificar a missão dos agentes;
2. O envio de mensagens entre nodos deve ser permitido;
3. O envio de mensagens entre agentes deve ser permitido;
4. O sistema deve disponibilizar uma operação para aquisição do valor do nível energético do nodo;
5. O sistema deve disponibilizar uma operação para aquisição do valor da quantidade de memória disponível do nodo;
6. O sistema deve disponibilizar uma operação para aquisição do valor do relógio local do nodo;
7. O sistema deve disponibilizar uma operação para aquisição dos valores do sensor do nodo;
8. Os agentes devem poder se clonar de um nodo para outro;
9. Os agentes devem poder se mover entre nodos;
10. O sistema deve permitir a definição e manipulação de variáveis estatísticas;

11. O sistema deve permitir configuração das características da rede sem fio;
12. Devem existir operações para a alteração do tamanho do agente;
13. O sistema deve permitir a especificação de topologia de rede de um determinado cenário de simulação;
14. O sistema deve informar sobre o tempo de simulação;
15. O sistema deve permitir a especificação do posicionamento dos nodos;
16. O sistema deve permitir nodos móveis.

Os Requisitos Funcionais podem ser agrupados em Casos de Uso. Os Casos de Uso considerados neste trabalho são: **Definir topologia da rede**, **Definir Parâmetros e Variáveis de Simulação**, **Especificar Missão do Agente** e **Obter Dados da Simulação**. Estes Casos de Uso são representados na Figura 5.1 e são detalhados nos Casos de Uso Descritivos no Apêndice A, respectivamente, por: UC01, UC02, UC03 e UC04.

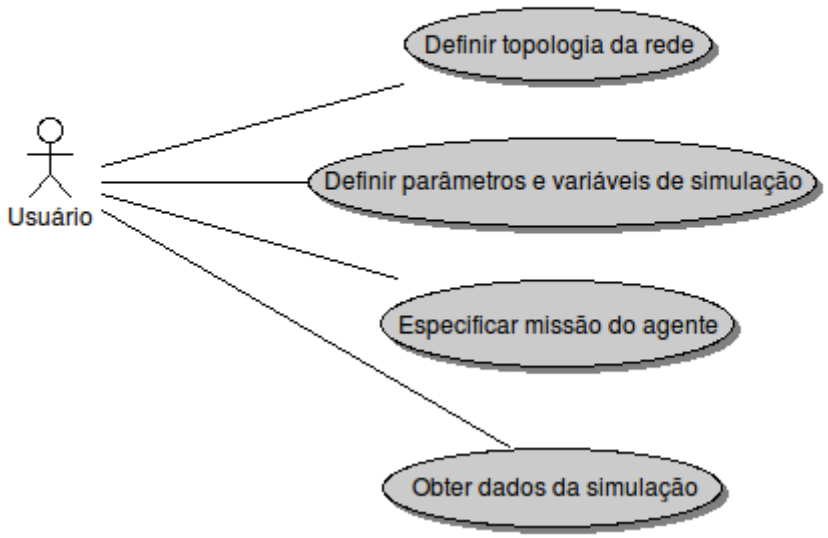

Figura 5.1: Diagrama de Caso de Uso dos Requisitos Funcionais do *MASiM*.

Na Tabela 5.1, é descrito o rastreamento dos Requisitos Funcionais com os Casos de Uso. Assim, é possível verificar a quais casos de uso um determinado requisito funcional pertence.

Além dos Requisitos Funcionais, o *MASiM* deve atender alguns Requisitos Não Funcionais, tais como:

1. Diferentes tipos de fenômenos (evento ambiental monitorado) devem ser permitidos;
2. A unidade de medida do tempo de simulação deve ser segundos;
3. Cada mensagem apresenta uma identificação única, cujo valor é formado pelo identificador do nodo remetente, concatenado com um número único. O nodo remetente é responsável por realizar o controle da identificação das mensagens;
4. Os agentes devem ser identificados de maneira única em toda a rede;
5. Um agente apresenta apenas uma missão em todo seu ciclo de vida;
6. A ferramenta deve permitir a simulação de sistemas multi-agentes;
7. Os agentes devem acessar recursos dos nodos através de uma interface própria;
8. Os agentes só executam em nodos que apresentam um ambiente próprio para sua execução;
9. A Rede Sem Fio a ser utilizada é baseada no protocolo *IEEE 802.15.4*.

Requisitos Funcionais	Casos de Uso			
	UC1	UC2	UC3	UC4
1			X	
2			X	
3			X	
4			X	X
5			X	X
6			X	
7			X	
8			X	
9			X	
10		X	X	X
11	X	X		
12		X	X	
13	X			
14		X		
15	X	X		
16	X	X		

Tabela 5.1: Tabela de rastreamento dos requisitos funcionais nos Casos de Uso.

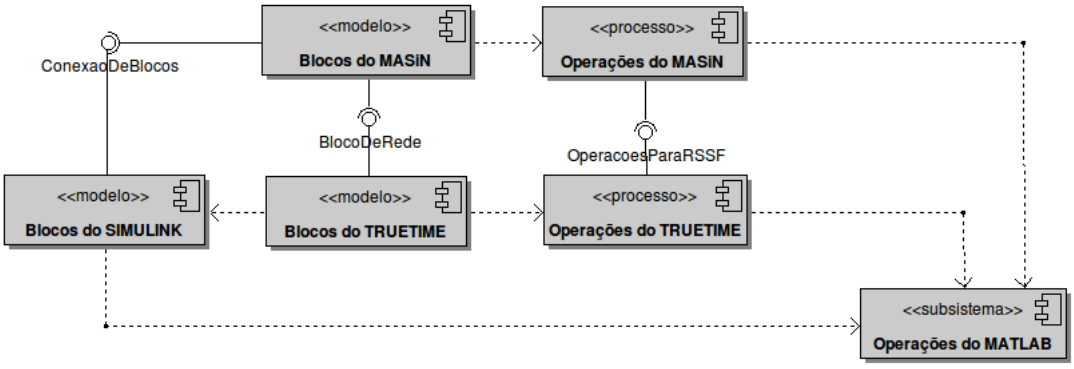

Figura 5.2: Diagrama de Componentes do *MASiM*

5.2.2 Componentes de Software Participantes

Na elaboração do projeto do *MASiM*, é importante descrever quais são os recursos a serem utilizados. A importância disto está relacionada aos requisitos de configuração do sistema e o planejamento da utilização de recursos que a ferramenta depende para seu funcionamento. Desta forma, na Figura 5.2, estão representados os componentes utilizados para a elaboração desta ferramenta, com suas respectivas interfaces.

O *MASiM* deve ser instalado e configurado em um ambiente que disponibiliza um conjunto de facilidades para a codificação da ferramenta. O ambiente utilizado representado como componente na Figura 5.2 é o *MATLAB*. Os principais recursos a serem utilizados neste ambiente são a manipulação de variáveis globais (variáveis de *workspace*), funções pré-definidas para manipulação de matrizes e animações das simulações.

Outro componente do qual a ferramenta proposta neste trabalho depende é o *Simulink*. O *Simulink* é um *toolbox* para o *MATLAB* que disponibiliza um conjunto de blocos básicos pré-programados, que são utilizados pelo *TrueTime* e pela ferramenta *MASiM* na definição da topologia da rede, representação de gráficos de variáveis estatísticas.

O *TrueTime* é mais um componente do qual o *MASiM* depende. Este componente irá disponibilizar conjuntos de blocos e operações que facilitam a programação da ferramenta *MASiM*, principalmente de redes (com fio ou sem fio) e operações de programação de tarefas de tempo real. Os blocos do *TrueTime* encapsulam os blocos do *Simulink*. Da mesma forma, os blocos disponibilizados pelo *MASiM* encapsulam os blocos do *TrueTime* realizando uma pré-configuração sobre os mesmos para se comportarem como uma RSSF. As operações disponibilizadas pelo *TrueTime* serão utilizados

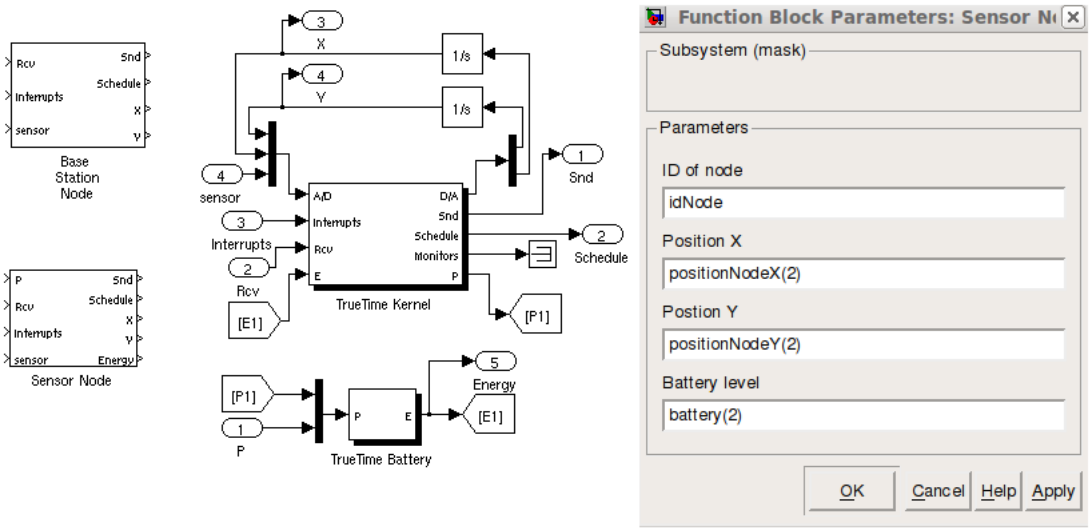

Figura 5.3: Bloco de Nodo Estação Base e Nodo Sensor do *MASiM*.

na programação do *MASiM* como também auxiliam na especificação das missões dos agentes.

Na Figura 5.3, são representados os blocos de Nodo Estação Base (*Base Station Node*) e *Nodo Sensor* (*Sensor Node*) disponibilizados pelo *MASiM*. A única diferença entre o bloco *Nodo Estação Base* e *Nodo Sensor* é que no primeiro não é considerado o consumo de energia.

Estes dois blocos definem portas de comunicação com outros blocos, que podem ser de entrada ou saída do bloco. As portas de entrada são: recebimento de mensagem (*Rcv*), entrada de interrupção (*interrupts*) e entrada de dados monitorados (*sensor*). Além das portas de entrada, existem também as portas de saída, que são: envio de mensagem (*Snd*), informação sobre a execução das tarefas dos nodos (*Schedule*) e posicionamento do nodo (*X* e *Y*).

Na Figura 5.3, são representados os blocos do *TrueTime* (*TrueTime Kernel* e *TrueTime Battery*) e *Simulink* para construção dos blocos de nodos do *MASiM*. Além disto, está presente na mesma figura uma tela do sistema com a configuração do nodo, com os seguintes campos: identificador do nodo (*ID of node*), posição inicial (*Position X* e *Position Y*) e nível da bateria (*Bettery Level*).

O bloco de Rede de Sensores Sem Fio (*Wireless Sensor Network*) disponibilizado pelo *MASiM* é representado na Figura 5.4. Este bloco apresenta como portas de entrada: canal de mensagens enviadas na rede (*Snd*) e posicionamento dos nodos (*X* e *Y*). Como portas de saída, o bloco de rede possui: canal de mensagens recebidas na rede (*Rcv*), informação sobre o escalonamento das mensagens dos nodos trafegadas no canal (*Schedule*) e potência da energia de toda a rede (*P*).

O bloco do *TrueTime* (*TrueTime Wireless Network*) e os blocos do *Simulink* utilizados para

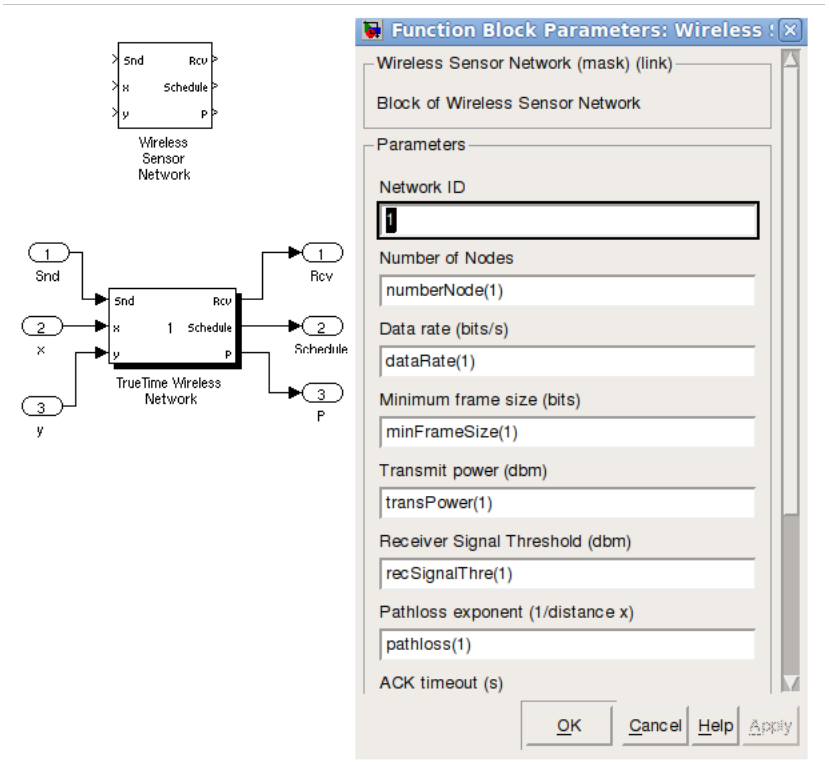

Figura 5.4: Bloco de Rede do *MASiM*.

compor o bloco de RSSF também estão presentes na Figura 5.4. Além disto, nesta mesma figura há a tela do sistema com parâmetros de configuração deste bloco de rede, com os seguintes campos: identificador da rede (*Network ID*), número de nodos (*Number of Nodes*), taxa de transmissão de dados (*Data Rate*), tamanho mínimo da janela de dados trafegados (*Minimum Frame Size*), potência de transmissão (*Transmit Power*), limiar do sinal recebido (*Receiver Signal Threshold*), taxa de perda (*Pathloss Exponent*), prazo para o recebimento do ACK (*ACK Timeout*), limite de repetições (*Retry Limit*) e limiar de codificação de erro (*Erro Coding Threshold*).

Uma descrição mais detalhada dos componentes *MATLAB*, *Simulink* e *TrueTime*, informando os principais recursos disponibilizados por estes, está presente no Capítulo 4 (Ambientes de Simulação).

5.2.3 Ciclo de Vida do Agente

A execução da missão do agente representa ciclo de vida deste agente. Este ciclo de vida na ferramenta *MASiM* é definido como o modelo do FIPA (*Foundation for Intelligent Physical Agents*), detalhado na Figura 3.3 do Capítulo 3 (Agentes Móveis). Assim, no estado **Iniciado**, o agente é

instanciado e é atribuída a sua missão. Além disto, o agente também recebe sua identificação única e global. O usuário é responsável por garantir a exclusividade deste identificador ao agente. O agente é ativado em um nodo este inicia a execução da sua missão. Neste momento, ele passa a estar no estado **Ativo**.

No estado **Ativo**, o agente executa operações definidas na sua missão. Duas operações principais são mobilidade e clonagem. A execução destas duas operações envolve um protocolo específico de mobilidade ou de clonagem, que é negociada entre as agências. Em ambas as operações, o agente fica no estado de **Suspenso** até a finalização do protocolo, que ocorre através de uma chamada de sistema da agência. No estado de **Suspenso**, o estado de execução do agente é interrompido até que a agência reative o agente. O protocolo de mobilidade é definido no diagrama de atividade representado na Figura 5.5.

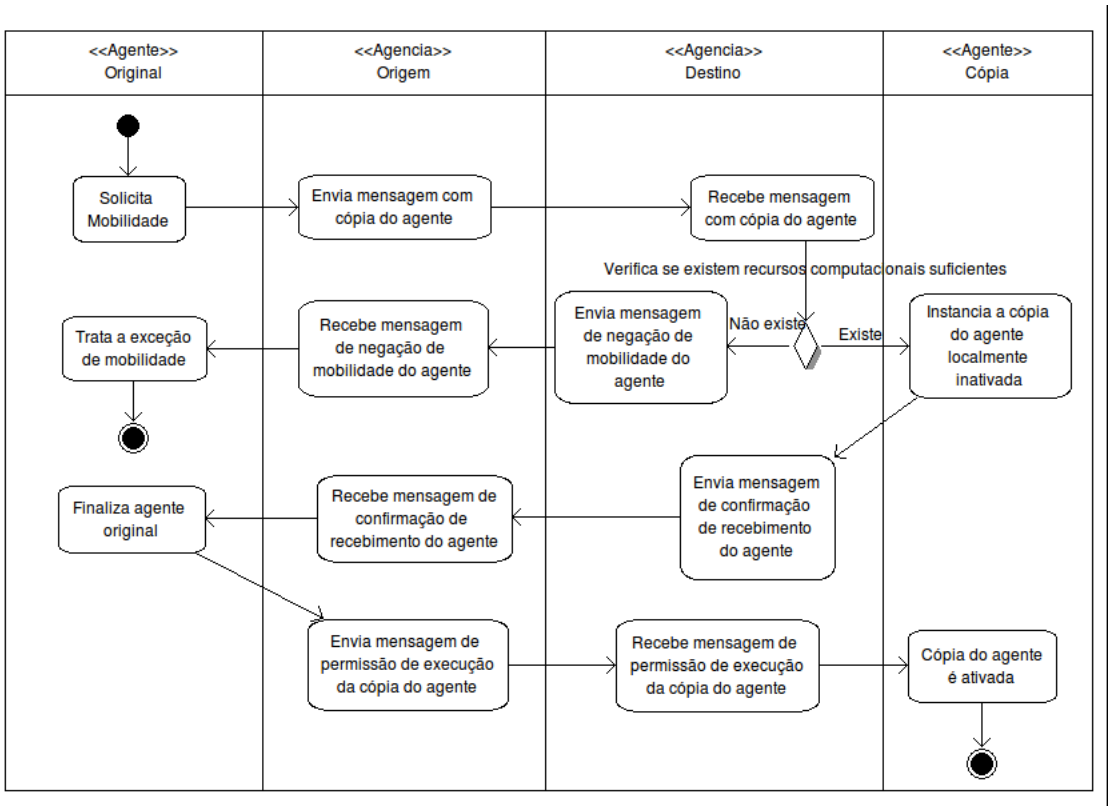

Figura 5.5: Diagrama de Atividades do protocolo de mobilidade do agente.

O protocolo de mobilidade inicia-se a partir da solicitação do agente original de mover-se de um nodo para outro. Neste momento, o agente fica no estado de **Suspenso** e a agência origem envia uma mensagem que contém uma cópia do agente original para a agência destino. A agência destino irá verificar se existem recursos suficientes para a execução deste agente. Caso os recursos sejam

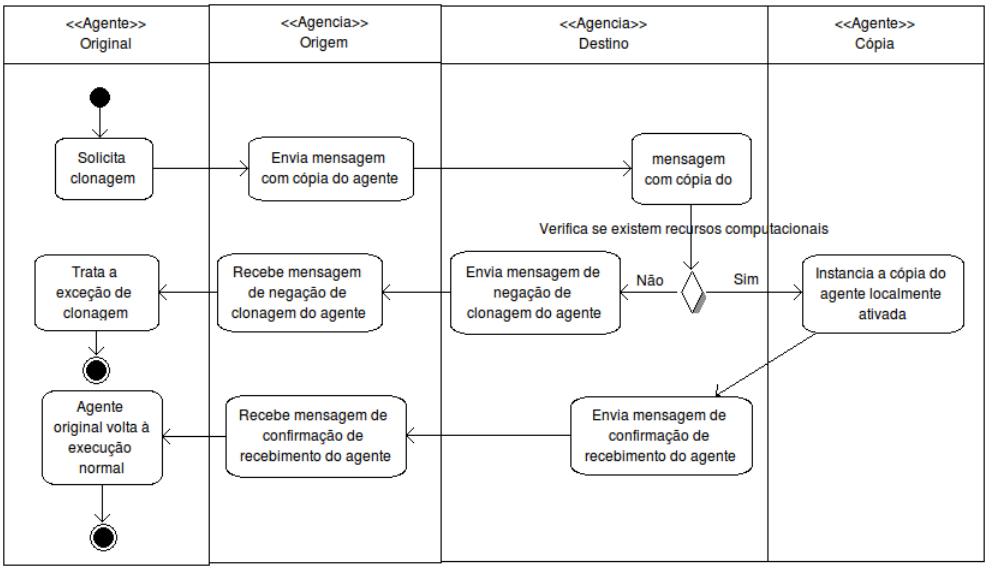

Figura 5.6: Diagrama de Atividades do protocolo de clonagem do agente.

suficientes, esta agência instancia o agente original. Contudo, ela deixa este agente no estado de **Suspenso**. Logo depois, a agência envia uma mensagem de confirmação do recebimento do agente para a agência origem. A agência origem, ao receber esta mensagem, finaliza a sua instância local do agente. Assim, a agência envia uma mensagem de permissão de execução da cópia do agente na agência destino. A agência destino, ao receber esta mensagem, ativa a execução do agente. Desta forma, o agente irá voltar a executar no próximo estado. Caso a agência destino não tenha recursos suficientes para execução do agente, ela envia uma mensagem para a agência origem negando a mobilidade deste agente.

O protocolo de clonagem é definido no diagrama de atividades representado pela Figura 5.6. O protocolo de clonagem inicia-se de maneira análoga ao protocolo de mobilidade. O agente original solicita à agência origem a sua clonagem e este agente fica no estado de **Suspenso**. A agência origem envia uma mensagem para a agência destino com uma cópia do agente original. A agência destino, ao receber esta mensagem, verifica se existem recursos suficientes para a execução do agente. Caso existam, esta agência cria uma instância deste agente. Diferentemente, como ocorre no protocolo de mobilidade, esta cópia do agente fica no estado de **Ativo** logo após a agência destino enviar a mensagem de confirmação de recebimento do agente. A agência origem, ao receber esta mensagem, põe o agente no estado de **Ativo**. Caso a agência destino não tenha recursos suficientes para a execução do agente, ela envia uma mensagem para a agência origem negando a clonagem do agente.

Quando um agente se move ou se clona, seu estado de execução e suas variáveis internas são

transformados em um fluxo de *bytes* (Serialização) que serão trafegados na rede do nodo origem para um determinado nodo destino. Desta forma, quando todo este fluxo de *bytes* chega ao nodo destino, a agência destino deverá reconstruir o agente através deste fluxo recebido (Desserialização) (Günes *et al.* [2003]). Todo este processo é realizado pelo *toolbox TrueTime*, e a forma como tais operações são realizadas não fazem parte do escopo deste trabalho. Além disto, o tamanho do agente pode variar durante a execução do seu ciclo de vida. Contudo, neste caso, o usuário é o responsável em determinar como se dará a variação do tamanho do agente.

Um agente pode enviar ou esperar uma determinada mensagem. Quando o agente espera por uma mensagem, este agente fica no estado de **Esperando**. Neste estado, existe a instância do agente, contudo, ele só passa para o próximo estado especificado em sua missão quando chega alguma mensagem para ele. Caso um agente envie uma mensagem para outro agente que não está mais presente na agência, esta mensagem será descartada pela agência.

5.2.4 Principais Entidades da Ferramenta

A representação da organização das entidades do sistema pode ser realizada através do diagrama de classe. As principais classes do *MASiM* são: *MensagemNodo*, *MensagemAgencia*, *InformacaoMensagemAgencia*, *MensagemAgente*, *InformacaoAgencia*, *Fenomeno*, *Agente*, *Missao*, *Estado*, *Nodo* e *Agencia*. Estas classes são representadas na Figura 5.7. É considerado que existem métodos de acesso e escrita para todos os atributos das classes, deste modo, estes métodos não estão presentes nestes diagramas de classe.

As principais entidades de simulação, nodo (*Nodo*), agência (*Agencia*) e agente *Agente*, realizam a comunicação entre si através de troca de mensagens. Estas mensagens são trocadas entre estas entidades de maneira assíncrona. Caso uma entidade envie uma mensagem para outra entidade inexistente, esta mensagem é descartada. Estas mensagens são modeladas em classe e obedecem a uma hierarquia. Assim, pode-se afirmar que existem três classes de mensagens: mensagem do nodo (*MensagemNodo*), da agência (*MensagemAgencia*) e do agente (*MensagemAgente*). Cada mensagem carrega os dados que contêm informações a serem envidas.

A classe *MensagemNodo* representa a entidade mensagem que é trocada entre nodos. Os principais atributos desta classe são: *remetente*, *destinatario* e *identificador*. O remetente refere-se ao nodo que envia a mensagem. O atributo *destinatario* é o nodo a que a mensagem se destina. O *identificador* refere-se à identificação única da mensagem. A mensagem do nodo tem como conteúdo a informação principal que é enviada para o nodo destino, que consiste na mensagem da agência (*MensagemAgencia*).

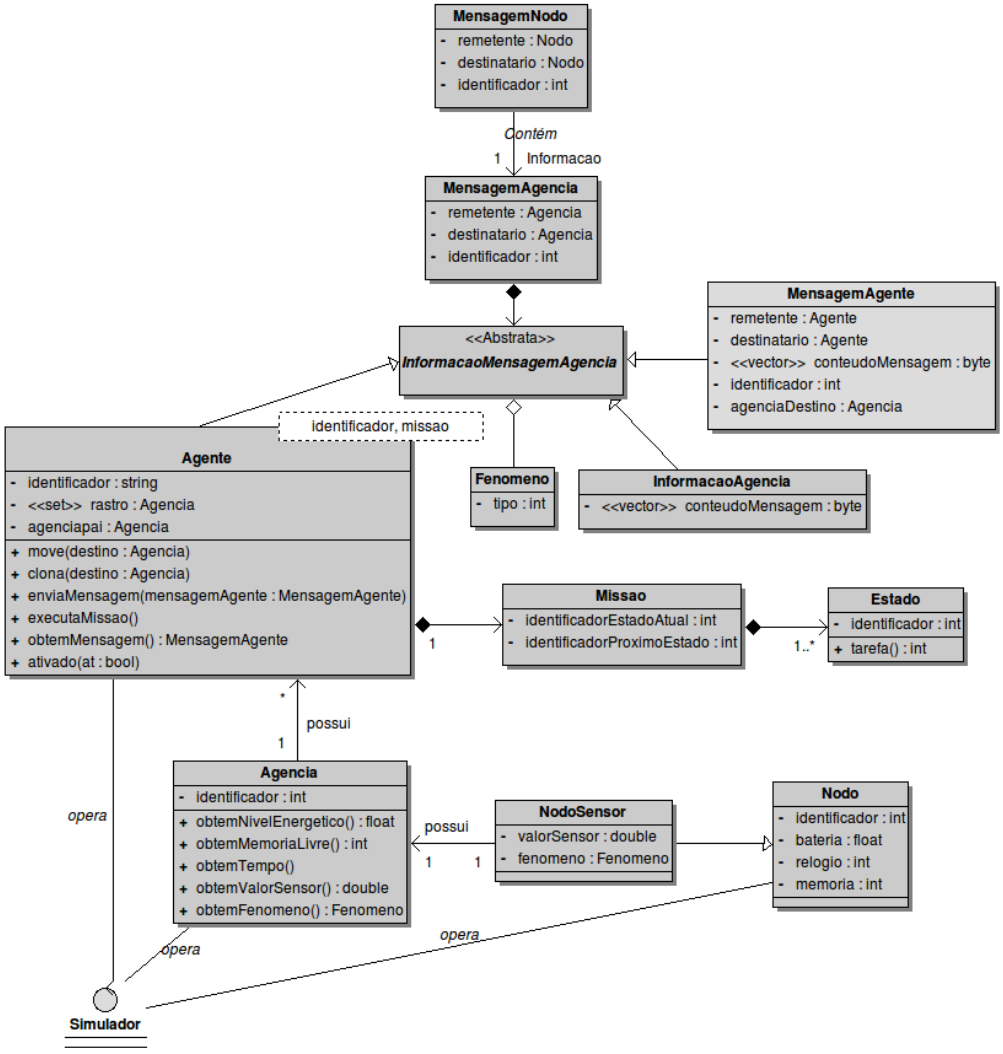

Figura 5.7: Diagrama de Classe das mensagens.

A mensagem da agência (*MensagemAgencia*), da mesma forma que ocorre com a classe *MensagemNodo*, apresenta como atributos a agência que enviou a mensagem (*remetente*), a agência a que se destina a mensagem (*destinatario*) e o identificador único da mensagem (*identificador*).

A classe *InformacaoMensagemAgencia* é uma classe abstrata que pode ser estendida pelas classes *Agente*, *InformacacaoAgencia* e *MensagemAgente*. A classe *InformacaoMensagemAgencia* contém o fenômeno da agência remetente. A classe *Fenomeno* representa o fenômeno que um determinado nodo monitora. Esta classe tem como atributo o tipo do fenômeno (*tipo*).

A classe *Agente* representa o agente propriamente dito. Esta classe tem como atributo um identificador único (*identificador*), um conjunto de agências que este agente percorreu (*rastro*) e a

referência para a agência pai (*agenciapai*), caso este agente seja um clone de outro agente. Contudo, esta referência pode ficar obsoleta se este agente pai se mover e não avisar aos seus clones sobre sua nova localização. No momento em que o agente é instanciado, além do identificador, o agente também recebe uma missão (*missao*). Esta missão não pode ser alterada enquanto este agente existir.

Além dos seus atributos, a classe *Agente* apresenta também os métodos *move* (determina a agência para a qual o agente irá se mover), *clone* (determina a agência em que será criada uma cópia do agente) e *enviaMensagem* (solicita o envio de uma mensagem para um determinado agente). Quando um determinado agente não é especificado, a mensagem é enviada para todos os agentes em uma determinada agência. Além destes métodos, há um método para obter uma determinada mensagem (*obtemMensagem*). O método *ativado* indica que o agente estar no estado de *Suspenso*. Os estados da missão do agente são executados pelo método *executaMissao*.

A missão do agente é representada pela classe *Missao*. Esta entidade contém o identificador do estado atual de execução (*identificadorEstadoAtual*) e o identificador do próximo estado a ser executado (*identificadorProximoEstado*) do agente. Através desta informação, é possível determinar o estado em que o agente deve executar após ele mover ou se clonar. A classe *Missao* pode ter um conjunto de estados (*Estado*). Estes estados contêm as operações que o agente deve realizar no método *tarefa*. Estas operações são definidas pelo usuário. Além disto, cada estado tem um identificador (*identificador*).

A classe *MensagemAgente* representa a mensagem que um agente envia para outro agente em uma determinada agência. O atributo *remetente* consiste no agente que envia a mensagem e o atributo *destinatario* representa o agente que deve receber a mensagem. A agência onde se encontra o agente destinatário é representado pelo atributo *agenciaDestino*. O atributo *conteudoMensagem* é a informação contida na mensagem enviada de um agente para outro. A agência destino, quando recebe uma mensagem para um determinado agente sob seu domínio, põe esta mensagem numa região de memória para que o agente possa obtê-la. Caso um agente envie uma mensagem para um agente que não se encontra na agência, esta mensagem é descartada pela agência. Não há nenhuma confirmação da existência do agente destino para o agente remetente por parte das agências.

A classe *InformacaoAgencia* representa a informação que *MensagemAgencia* pode conter durante a negociação nos protocolos de mobilidade e de clonagem do agente entre as agências.

A classe *Nodo* representa os nodos da rede. Cada nodo é identificado unicamente através do atributo *identificador*. O nível energético da bateria do nodo é representado pelo atributo *bateria*. O valor do relógio local do nodo é representado com atributo *relogio*. O atributo *memoria* determina

a quantidade de memória disponível no nodo. Um nodo em uma RSSF pode ser de dois tipos: estação base e nodo sensor. A estação base é representada pela classe *Nodo*. A classe *NodoSensor* representa o nodo sensor. Esta classe estende as características da classe *Nodo* e apresenta o atributo adicional *valorSensor*, que consiste no valor do sensor. O tipo de fenômeno monitorado pelo nodo é representado pelo atributo *fenomeno*.

A agência é um ambiente que permite a execução do agente em um nodo, funcionando como uma interface entre o agente e nodo, provendo e gerenciando recursos disponíveis no nodo para o agente. A agência é representada pela classe *Agencia*. Esta classe é identificada unicamente pelo atributo *identificador*. O valor deste atributo irá corresponder ao mesmo valor do atributo *identificador* do nodo na qual ela está contida. Os métodos que ela disponibiliza são utilizados pelos agentes para acessar informações sobre os recursos do nodo. Assim, para obter informação sobre o nível energético do nodo, é invocado o método *obtemNivelEnergetico*. Através do método *obtemMemoriaLivre*, é possível descobrir o quanto de memória disponível tem o nodo. O método *obtemValorSensor* retorna o valor do sensor do nodo. O tipo do fenômeno monitorado no nodo sensor é obtido através do método *obtemFenomeno*.

O papel da classe controladora *Simulador* é realizado pelo ambiente *MATLAB* junto com os *toolboxes TrueTime* e *Simulink*. O *Simulador* é responsável por realizar a execução da simulação, realizando o controle do tempo e das variáveis de simulação, como o consumo da bateria e operação do relógio dos nodos da rede, cálculo da mobilidade dos nodos, execução das operações dos nodos e da rede.

5.3 Programação das Tarefas

A ferramenta *MASiM* foi programada através linguagem de programação *C++ MEX*. Esta é uma linguagem baseada na sintaxe do *C++ ANSII*, estendendo recursos específicos do ambiente *MATLAB*. Além disto, a programação foi baseada nos recursos disponibilizados pelo *toolbox TrueTime*.

A utilização do *TrueTime* exige que a programação siga um modelo baseado em tarefas. Estas tarefas podem ser periódicas ou aperiódicas. Uma tarefa também pode se comunicar com outras através de regiões de memórias compartilhadas denominadas de *Mailboxes*. Para seguir este modelo de programação, as tarefas do *MASiM* foram organizadas em dois grupos: Tarefas do Nodo e Tarefas da Agência. As Tarefas do Nodo são aquelas estritamente relacionadas às operações que todos os nodos apresentam. As Tarefas da Agência são as tarefas presentes apenas nos nodos que executam agência. Assim, as Tarefas do Nodo são:

- **Controlador de Mensagens Recebidas pelo Nodo:** Tarefa aperiódica disparada por uma interrupção gerada quando uma determinada mensagem chega. Esta tarefa tem o papel de receber e encaminhar a mensagem para a tarefa responsável por ela;

- **Manipulador de Mensagens Sobre a Vizinhança:** Tarefa aperiódica disparada pela tarefa **Controlador de Mensagens Recebidas pelo Nodo**. Esta tarefa tem o papel de processar as mensagens com informação sobre os nodos vizinhos. Assim, estas informações sobre a vizinhança são armazenadas em memória para a utilização dos agentes;

- **Manipulador de Mensagens Simples:** Tarefa aperiódica disparada pela tarefa **Controlador de Mensagens Recebidas pelo Nodo**. Esta tarefa tem o papel de tratar as mensagens definidas pelo usuário. A forma que esta mensagem deve ser tratada é especificada pelo usuário;

- **Controlador de Mensagens Enviadas pelo Nodo:** Tarefa aperiódica responsável por enviar uma determinada mensagem para outro nodo. Uma determinada tarefa põe uma informação no *Mailbox* que deve ser enviada e dispara um evento para ativar o **Controlador de Mensagens Enviadas pelo Nodo**. Esta tarefa obtém a informação e embute numa mensagem que será enviada para um nodo;

- **Notificação à Vizinhança do Nodo:** Tarefa periódica que tem como papel enviar mensagem para os nodos vizinhos com informação sobre o nodo corrente. As informações que são enviadas são: o nível energético, identificador e coordenada do nodo. Esta tarefa põe as informações do nodo no *Mailbox* e dispara um evento que ativa a tarefa **Controlador de Mensagens Enviadas pelo Nodo**. Esta última tarefa será responsável por enviar mensagem sobre o nodo para todos os nodos vizinhos.

As tarefas que têm o papel de executar operações relacionadas à agência são as seguintes:

- **Controlador de Mensagens Recebidas pela Agência:** Tarefa aperiódica disparada pela tarefa **Controlador de Mensagens Recebidas pelo Nodo**. Esta tarefa tem como papel receber e processar uma determinada mensagem para ela presente no *Mailbox*. Caso esta mensagem seja para a agência, esta tarefa irá processar esta mensagem. Isto ocorre quando é uma mensagem do protocolo de mobilidade ou de clonagem. Entretanto, caso esta mensagem contenha um agente, ela encaminha para a tarefa **Criador do Agente**, a qual é responsável por instanciar o agente no nodo local. A mensagem também pode ser para um agente local. Assim, esta mensagem é encaminhada para a tarefa **Controlador de Execução do Agente**, responsável por entregar a mensagem ao agente, caso este exista. Caso contrário, esta mensagem é descartada;

- **Criador do Agente:** Tarefa aperiódica disparada pela tarefa **Controlador de Mensagens Recebidas pela Agência**. Esta tarefa tem como papel instanciar um determinado agente que tenha sido recebido durante a mobilidade ou clonagem para o nodo atual;
- **Controlador de Execução do Agente:** Tarefa periódica que tem como papel executar os agentes seguindo uma fila de execução. Esta tarefa executa e controla o ciclo da missão de cada agente.

As tarefas do nodo e da agência apresentam uma hierarquia de prioridade de execução pelo processador. A prioridade de uma tarefa pode ser classificada como, alta, média ou baixa. Assim, se em um determinado instante houver concorrência entre duas ou mais tarefas pela execução do processador, a tarefa de maior prioridade terá o privilégio de ser executada pelo processador, podendo preemptar a tarefa de menor prioridade. No gráfico da Figura 5.8, é representada esta hierarquia de prioridade das tarefas.

As tarefas aperiódicas *Controlador de Mensagens Recebidas pelo Nodo* (**A**) e *Controlador de Mensagens Enviadas pelo Nodo* (**C**) possuem a maior prioridade no sistema. Assim, em um nodo, as operações de receber e enviar mensagem apresenta a mesma prioridade. As tarefas *Manipulador de Mensagens Simples* (**B**) e *Controlador de Mensagens Recebidas pela Agência* (**F**) apresentam prioridade média no sistema. Finalmente, as tarefas *Manipulador de Mensagens sobre a Vizinhança* (**D**), *Notifica a Vizinhança do Nodo* (**E**), *Criador de Agente* (**G**) e *Manipulador de Execução do Agente* (**H**) apresentam prioridades mais baixas no sistema.

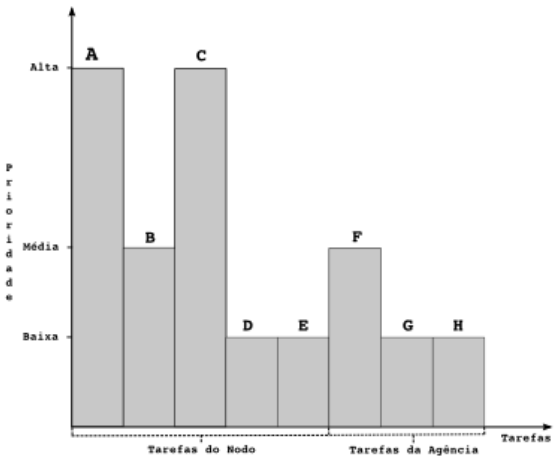

Figura 5.8: Hierarquia de prioridade de execução das tarefas.

Estas tarefas são pré-configuradas pela ferramenta. Contudo, os atributos das tarefas podem ser alterados no arquivo principal de configuração, tais com, o período e o *offset* das tarefas periódicas,

como também o *deadline* das tarefas aperiódicas. Além disto, também neste arquivo, a prioridade, o tempo de execução e a política de escalonamento das tarefas também podem ser redefinidos para cada nodo da rede.

5.4 Configuração dos Cenários de Simulação

A elaboração de cenários de simulação de agentes em uma RSSF faz uso de componentes das bibliotecas próprias especificadas neste trabalho, além dos componentes disponíveis do *Simulink* e *TrueTime* sobre o ambiente *MATLAB*. Assim, para realizar a simulação de um determinado cenário, há a necessidade de se realizar um conjunto de atividades, como é representado no Diagrama de Atividade da configuração do *MASiM* na Figura 5.9.

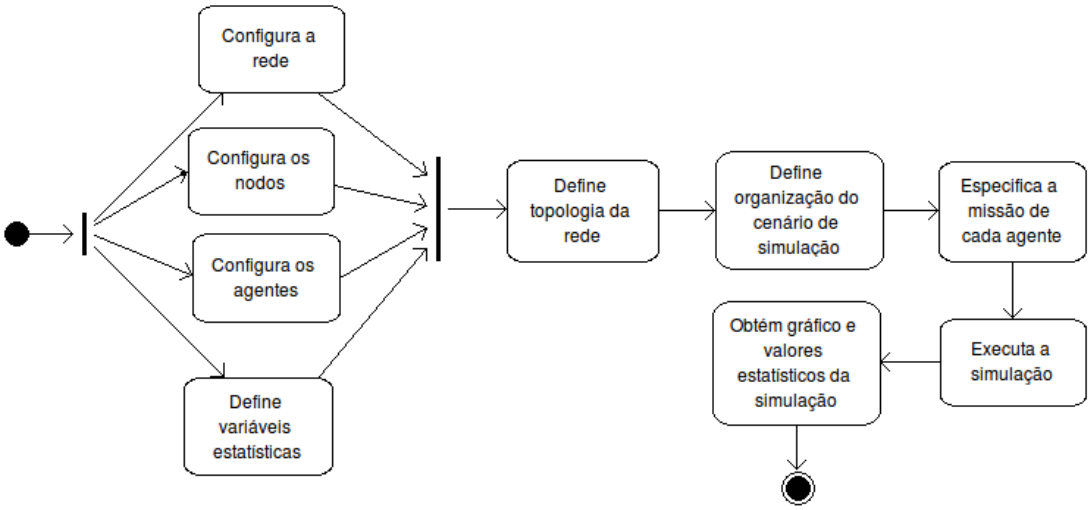

Figura 5.9: Diagrama de Atividade da configuração do *MASiM*.

Inicialmente, é necessário determinar a configuração da ferramenta para realizar a simulação. As seguintes atividades podem ser realizadas de maneira independente uma da outra: **Configura a rede**, **Configura os nodos**, **Configura os agentes** e **Define as variáveis estatísticas**. Contudo, o próximo passo a ser executado depende de todas as atividades citadas estarem completas. Estas atividades devem ser realizadas em um arquivo de configuração, que é carregado no início da simulação. Um exemplo deste arquivo está presente no Apêndice B.

Na atividade **Configura a rede**, as variáveis de configuração da rede são especificadas. Estas variáveis são: número de nodos, a taxa de transferência de dados, tamanho mínimo da janela das

mensagens, potência da antena de rádio, e demais variáveis relacionadas à transmissão de mensagens na rede tais como, quantidade de tentativas de envio de mensagem e limiar do sinal de recebimento.

Na atividade **Configura os Nodos** são atribuídas valores às variáveis relacionadas à configuração dos nodos, tais como valor inicial da bateria, tipo de fenômeno, capacidade da memória e posição dos nodos. Além disto, é possível especificar a ocorrência de falhas dos tipos *parada por falha* (*faultNodes*), *falha arbitrária* no envio de mensagem e *falhas de temporização* dos relógios locais dos nodos (*clockDriff*).

Na atividade de **Configura os Agentes** atribuem-se os valores das variáveis dos agentes móveis, tais como o tamanho inicial em *bytes* do agente e as variáveis que o agente deve manipular durante a execução de sua missão.

A topologia da rede é definida através da elaboração do diagrama de blocos. Estes blocos são disponíveis através das ferramentas *MASiM*, *TrueTime* e *Simulink*. Através destes blocos, o tipo de rede a ser utilizada e o mapeamento dos blocos dos nodos com as operações que eles devem realizar são especificados. Esta atividade é realizada em *Define Topologia da Rede* e é representada na Figura C.1 do Apêndice C.

Na atividade **Define Organização do Cenário de Simulação** especifica como são iniciados os cenários de simulação. Esta atividade é representada no Apêndice D através do Algoritmo D.1.

Na atividade **Especifica a Missão de Cada Agente** são definidos os estados e o fluxo de execução destes estados de cada agente. Cada estado terá um conjunto de operações especificadas pelo usuário. Um exemplo desta atividade está presente na Figura E.1 do Apêndice E. A especificação da missão dos agentes pode necessitar de algumas funções que são disponibilizadas pelo *MASiM*. Estas funções estão presentes na Tabela 5.2.

Após definidos os arquivos de configuração, topologia da rede e a organização do cenário, é realizada a tarefa **Executa a Simulação**. Durante a execução da simulação, as informações sobre a execução do agente são armazenadas em um arquivo e as variáveis estatísticas são carregadas. Além disto, podem existir animações da simulação. Contudo, o bloco de *animação de simulação* utilizado no *Simulink* torna a execução mais lenta. Em qualquer momento, a simulação pode ser parada e valores momentâneos podem ser obtidos.

A atividade *Obtém Gráfico e Valores Estatísticos da Simulação* é realizada utilizando-se recursos do *Simulink* (*scope*) e do *MATLAB* (*plot*) para a geração de gráficos das variáveis. Nas Figuras F.1 e F.2 do Apêndice F, são exemplos de gráficos gerados de escalonamento das mensagens e consumo de energia de um nodo usando o *scope*.

Função	Descrição
maGetEnergyLevel	Obtém o nível energético da agência onde está sendo executado o agente.
maGetTime	Obtém o valor do relógio do nodo onde está sendo executado o agente.
maMoveAgent	Operação de mobilidade do agente de uma agência para outra. Deve-se especificar o agente que irá mover-se, o nodo destino, identificador da rede, o próximo passo a ser executado pelo agente quando ele se mover e o estado a ser executado quando ocorrer alguma exceção na operação.
maCloneAgent	Operação de clogem do agente. Deve-se especificar o agente a ser clonado, identificador da agência destino, identificador da rede, o próximo estado a ser executado pelo agente original, próximo estado a ser executado a ser executado pelo agente clonado e o passo de execução de exceção.
maSendAgentMessage	Envia uma mensagem para outro agente. Deve-se especificar o agente remetente, o agente destinatário, a agência destinatário, identificador da rede, o tipo de fenômeno monitorado, a informação da mensagem e o próximo estado a ser executado pelo agente remetente após o envio da mensagem.
maGetAgentMessageReceived	Espera por uma mensagem do agente.
maGetAgencyID	Obtém o identificador da agência atual.
maGoToNextStep	Determina qual é o próximo estado que o agente deverá executar.
maAgentFinalState	Marca o estado final do agente.
maGetAllNeighbors	Operação que retorna uma lista de identificadores dos nodos vizinhos. Se não houver vizinho retorna ***null***
maGetLengthAllNeighbors	Obtém a quantidade de nodos vizinhos do nodo atual. Se não houver nodos vizinhos, retorna 0
maGetMostEnergyLevelNeighbor	Operação que informa qual o nodo vizinho que apresenta maior nível energetico.
maGetSensorData	Operação que retorna o dado sensoriado pelo nodo sensor.
maGetCurrentTime	Operação que retorna o valor do relógio do nodo.
maExistClonedAgent	Verfica se o agente já se clonou em uma determinada agência.

Tabela 5.2: Lista de funções que podem ser utilizadas no escopo dos estados do agente.

5.5 Considerações Finais

Este Capítulo definiu os modelos considerados pela ferramenta: modelo de nodos, RSSF, das tarefas e de falhas. Além disto, foram também descritos os requisitos que o *MASiM* deve contemplar e a sua arquitetura.

A descrição da arquitetura abordou os principais componentes de *software* utilizados, tais como

o *MATALAB*, *Simulink* e *TrueTime*. Além disto, foi descrito o ciclo de vida do agente e o comportamento dos protocolos de mobilidade e de clonagem. As entidades principais que constituem a ferramenta também foram descritas abordando seus atributos, métodos e a interação entre elas.

Os detalhes sobre a implementação foram abordados neste Capítulo descrevendo a programação das tarefas do *TrueTime*. As atividades de configuração para a execução da simulação e de aquisição dos valores estatísticos também foram descritos através do diagrama de atividades.

Nesta versão da ferramenta, existem algumas limitações que deveram ser resolvidas em versões futuras desta ferramenta. Algumas destas limitações são:

1. Permitir o usuário definir quais são as informações a serem enviadas nas mensagens de notificação da vizinhança;
2. Possibilidade do usuário especificar outros protocolos de mobilidade e clonagem de agente além do protocolo disponibilizado neste trabalho;
3. Possibilidade da especificação das missões dos agentes utilizando a linguagem do *MATLAB*
4. Necessidade de sistema com interface gráfica que permita definir a topologia de maneira mais automática.

Capítulo 6

Avaliação do Toolbox MASiM Através de Cenários de Uso

Neste Capítulo, são propostos determinados cenários em que demonstram a utilização e aplicação prática da ferramenta *MASiM*. Através destes cenários, é possível verificar as funcionalidades disponíveis e como é possível realizar a especificação das missões dos agentes móveis em uma Rede de Sensores Sem Fio.

A descrição e a especificação das características topológicas da RSSF dos cenários propostos são expostos na seção 6.1. Na seção 6.2, é proposta uma abordagem baseada em agentes móveis para estes cenários. Nesta abordagem, os agentes devem mover-se para os nodos sensores que conseguem alcançar diretamente a estação base. Outra abordagem também baseada em agentes é descrita em 6.3. Nesta abordagem, de forma contrária à abordagem anterior, os agentes podem se mover para os nodos que não alcançam diretamente a estação base. Além disto, duas outras abordagens baseadas na difusão de mensagens em RSSF são simuladas neste trabalho para comparar com as abordagens baseadas em agentes em relação a eficiência entre elas. Assim, a primeira abordagem deste tipo é a difusão simples que é descrita na seção 6.4. A segunda abordagem, denominada neste trabalho como difusão seletiva, é definida na seção 6.5. A comparação de eficiência entre todas as abordagens definidas anteriormente é realizada na seção 6.6. As considerações finais deste capítulo estão presentes na seção 6.7.

6.1 Especificação da RSSF utilizada nos cenários

Os cenários propostos para representar a utilização de agentes móveis em RSSF é uma adaptação do cenário o descrito por (Krishnamurthy e Zeid [2004]) baseado em um ambiente de man-

ufatura. Um ambiente de manufatura apresenta um conjunto de necessidades para o controle dos processos. Uma dessas necessidades é o monitoramento da operação de equipamentos de produção. Como resultado deste monitoramento, alarmes sobre o mau funcionamento destes equipamentos precisam ser emitidos e direcionados para o departamento de engenharia competente. Além dessas mensagens de alarmes, a aquisição de informações estatísticas sobre o ambiente fabril também precisam ser coletadas. Contudo, um desafio neste tipo de ambiente é a frequência da alteração das informações de interesse. Assim, essas informações possuem validades, as quais são dependentes do tempo (*deadlines*) em que estas informações são adquiridas. Portanto, a adoção de uma estratégia para se obter tais informações torna-se uma questão chave para a execução dos processos de maneira mais eficiente.

Em um ambiente de manufatura, podem existir alguns equipamentos críticos que são muitos sensíveis a alta temperatura. Assim, há a necessidade de um ambiente inteligente para ajustar a temperatura de um determinado ambiente para evitar danos nestes equipamentos além de evitar o desperdício de energia com excesso de resfriamento em um determinado ambiente sem necessidade. A utilização de RSSF pode auxiliar a construção deste ambiente inteligente. Nesta situação, um conjunto de nodos sensores distribuídos em alguns ambientes da fábrica podem monitorar um determinado ambiente e enviar informações da central. Esta central pode ser responsável em controlar o resfriamento e ventilação dos ambientes da fábrica.

A topologia da RSSF proposta neste cenário é modelada em um arquivo MDL no ambiente do *MATLAB*. Esta modelagem utiliza-se os diagramas de blocos elaborados neste trabalho, além dos especificados nas bibliotecas do *Simulink* e *TrueTime*.

A topologia dos cenários é mostrada parcialmente na Figura 6.1. Neste trabalho, são disponibilizados os diagramas de bloco de estação base (**A**), nodo sensor (**B)** e Rede de Sensores Sem Fio (**C**). Estes blocos são conectados entre si através de blocos auxiliares para conexão disponibilizados pelo *Simulink* tais como: *Source Block* (**F**), *Sink Block* (**E**), *Mux* (**G**) e *Demux* (**H**).

Os dois cenários utilizados neste trabalho diferenciam entre si apenas na quantidade de nodos sensores. O primeiro e segundo cenário de simulação é constituído, respectivamente, com 50 e 100 nodos. Estes nodos foram dispersos em um ambiente sem obstáculos consideráveis e totalmente planos de $300m^2$, inicialmente, de maneira aleatória. Além disso, é considerado que o posicionamento destes nodos não varia com o decorrer do tempo. Considera-se também, que apenas 10% dos nodos monitoram um determinado fenômeno alvo em cada cenário de simulação. Na Figura 6.1, estes nodos, que efetuam o monitoramento, são representados através dos blocos em **D**. Neste exemplo, os fenômenos monitorados apresentam valor 1 ou 0. O fenômeno de interesse considerado apresenta o

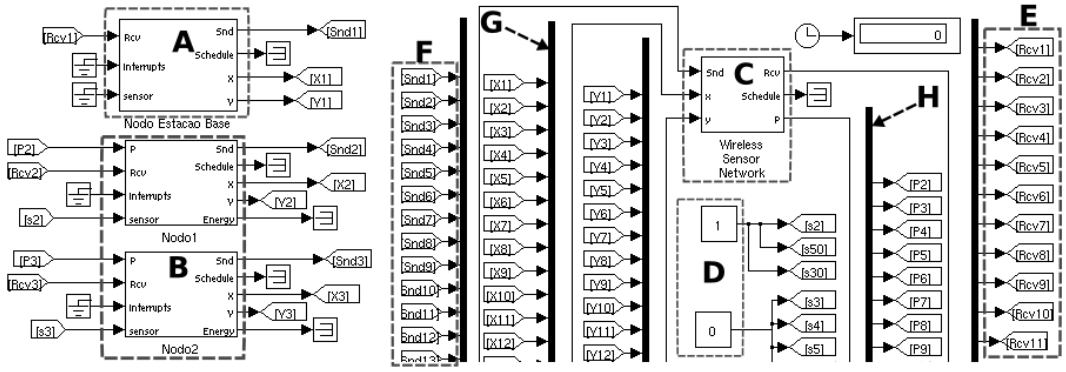

Figura 6.1: Diagrama de blocos da topologia da RSSF utilizada nos cenários.

valor igual a 1.

Os nodos utilizados nas simulações são baseados na configuração de *hardware* do Mica2 (Polastre *et al.* [2004]). Assim, a fonte de energia dos nodos sensores são duas baterias que somam uma carga energética de $27000J$ e voltagem de $3V$. O sinal de rádio da antena tem um raio de alcance de aproximadamente $99m$. Assim, há possibilidade de existir nodos desconexos na rede. Os consumidores de energia do nodo são processamento de dados e operações de envio e recebimento de mensagens durante a simulação. Neste trabalho, não é considerado o consumo de energia nas operações de gerenciamento do sensor ou de possíveis atuadores.

6.2 Abordagem baseada em agentes móveis que não alcançam a Estação Base

Com o objetivo de simplificar a referência desta abordagem, esta é denominada como **S1**. Este protocolo baseado em agentes visa ser simples para que haja uma redução na quantidade de mensagens trocadas, assim, reduzindo o consumo de energia durante a execução deste protocolo. Neste cenário, os nodos sensores são distribuídos em uma região de maneira aleatória. Os nodos sensores conseguem se comunicar diretamente com a estação base até uma determinada distancia (*r* na Figura 6.2). Nesta abordagem, um agente na estação base (**item A** da Figura 6.2) deve clonar-se para seus nodos vizinhos. Estes agentes clones representados pelos **itens A1** e **A2** da mesma figura, devem se mover para os nodos que apresentem maior nível energético ainda não visitados até encontrar algum nodo que tenha monitorado um determinado fenômeno alvo. Quando isto acontece, o agente deve enviar uma mensagem para o agente na estação base. Contudo, caso algum agente clone se mova para um nodo sensor que não alcance diretamente a estação base, este agente deverá

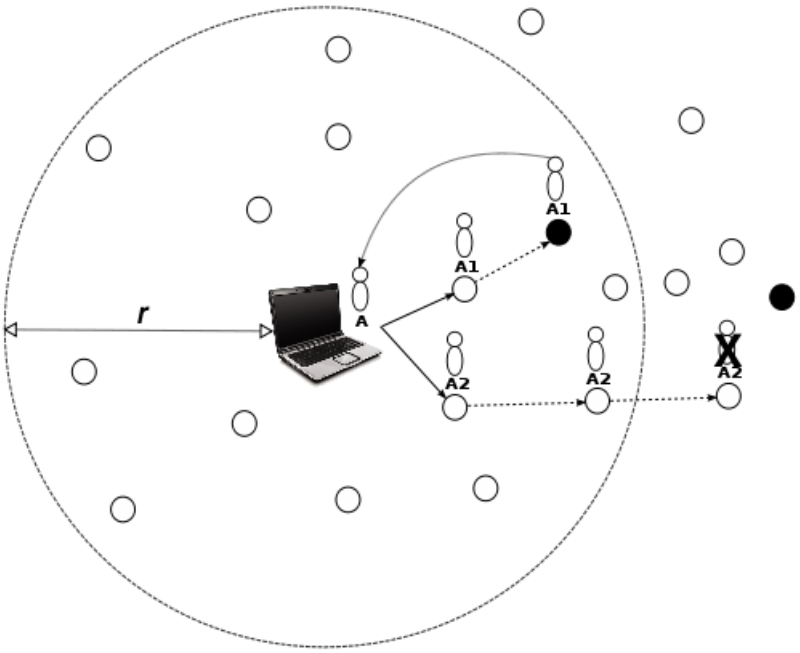

Figura 6.2: Representação do protocolo baseado em agentes da abordagem S1.

morrer, pois ele não irá conseguir enviar mensagem diretamente para o agente da estação base, caso encontre um fenômeno alvo monitorado neste nodo.

A Figura 6.3 representa a máquina de estados da missão do agente móvel neste cenário. Os estados **0**, **2** e **3** são referentes ao agente da estação base e os estados **1**, **4** e **5** são dos agentes clones. Os estados **2**, **3**, **4** e **5** são finais, ou seja, nestes estados, a missão do agente podem ser finalizados. O Algoritmo 6.1 especifica a missão do agente na estação base e dos seus clones.

Assim, a missão do agente neste cenário pode ser dividida nos seguintes estados da máquina de estados da Figura 6.3:

- **Estado 0:** este estado é especificado no Algoritmo 6.1 entre as linhas **5** e **18**. Neste estado, o agente na estação base se clona (**linha 10**) para todos os nodos vizinhos. Quando o agente recebe a mensagem com a notificação da ocorrência de algum evento alvo, este vai para o estado final representado pelo **Estado 2**. Caso, durante o processo de clonagem, alguma exceção seja lançada, o estado do agente irá para o **Estado 3**. Desta forma, os **Estados 2 e 3** lidam com as exceções de clonagem e mobilidade, respectivamente;

- **Estado 1:** este estado é especificado no Algoritmo 6.1 entre as linhas **19** e **42**. Neste estado, o agente verifica se o nodo corrente alcança a estação base (linhas **24 – 27**). Caso não seja

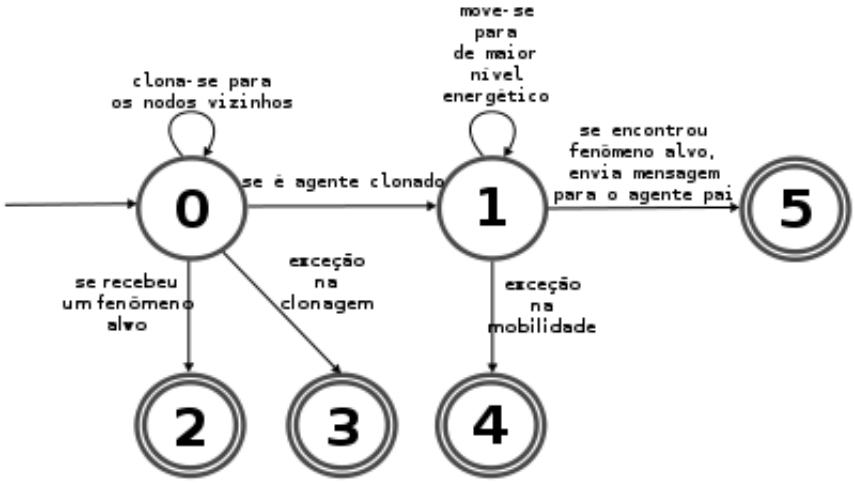

Figura 6.3: Modelo da especificação da missão dos agentes da abordagem S1.

alcançável, o agente clone irá morrer (**Estado 4**); Na situação oposta, o agente clone obtém a informação de interesse do nodo corrente. Assim, se o nodo corrente monitorou um fenômeno alvo, este então envia uma mensagem para o agente na estação base (**linha 32**). O agente irá mover-se para o nodo vizinho com maiores recursos energético (**linha 34**) caso o nodo corrente não tenha monitorado algum fenômeno alvo;

- **Estados 2, 3, 4 e 5:** estes estados são finais e são especificados no Algoritmo 6.1 entre as linhas **43** e **45**. Os **Estados 3 e 4** devem tratar as exceções que podem ocorrer na clonagem e mobilidade, respectivamente. Os **Estados 2 e 5** são estados finais quando as operações de receber fenômeno alvo e encontrar fenômenos são realizadas sem erro. Para simplificar o entendimento deste algoritmo, estes estados apenas realizam a operação de finalizar a missão do agente.

```
0  void executaMissaoAg1(Agent* agent, int nextStep) {
1    // Definicao das variaveis agentID, agentPaiID, allNeighbors, neighborNodesList,
         nodoEscolhido, neighborNodesListLength e reachFather
2
3    (*agent).getID(agentID);
4    switch (nextStep) {
5      case 0:
6        if (maGetAgentMessageReceived(agent) == NULL) {
7          bool noCloned = false;
8          allNeighbors = maGetAllNeighbors((*agent).getCurrentAgencyID());
9          int pos = 0;
10         while ( (!noCloned) && (pos < maGetLengthAllNeighbors((*agent).getCurrentAgencyID()
               )) ) {
11           if ( (!maExistClonedAgent(agent, allNeighbors[pos])) && (allNeighbors[pos] != (*
                 agent).selectLastVisitedNode()) ) {
12         noCloned = true;
```

```
13      } else {pos++;}
14          }
15          if (noCloned) {maCloneAgent(agent, allNeighbors[pos], 1, 0, 1, 3);}
16          else {maGoToNextStep(agent, 0);}
17        } else {maGoToNextStep(agent, 2);}
18        break;
19      case 1:
20        neighborNodesList = (*agent).getNeighborNodesList();
21        neighborNodesListLength = (*agent).getNeighborNodesListLength();
22        reachFather = false;
23        posList = 0;
24        while ( (!reachFather) && (posList < neighborNodesListLength) ) {
25          if (neighborNodesList[posList] == (*agent).getFatherNodeID()) {reachFather = true;}
26          posList++;
27        }
28        if (reachFather) {
29          if (maGetSensorData() == 1) {
30            (*agent).getCurrentAgencyID(), agentPaiID, (*agent).getFatherNodeID());
31            (*agent).getFatherAgentID(agentPaiID);
32            maSendAgentMessage(agent, agentPaiID, (*agent).getFatherNodeID(), 1, (*agent).
                  getPhenomenonType(), agent, 4);
33          }else{
34              nodoEscolhido = maGetMostEnergyLevelNeighbor((*agent).getCurrentAgencyID(), (*
                    agent).getListVisitedNodeID(), (*agent).lengthVisitedNode());
35              if (nodoEscolhido != -1) {
36                (*agent).storeVisitedNode((*agent).getCurrentAgencyID());
37                maMoveAgent(agent, nodoEscolhido, 1, 1, 5);
38                (*agent).resetAttemptsToExecuteProtocol();
39              } else {maGoToNextStep(agent, 1);}
40          }
41        }else{maGoToNextStep(agent, 4);}
42        break;
43      case 2,3,4,5:
44        maAgentFinalState(agent);
45        break;
46    }
47  }
```

Algoritmo 6.1: Missão do agente de S1.

6.3 Abordagem baseada em agentes que se movem além do alcance da Estação Base

O protocolo baseado em agentes proposto nesta abordagem tenta realizar o balanceamento entre a economia de energia e a cobertura da rede. Este protocolo é representado pela Figura 6.4.

Com o objetivo de facilitar a referência para esta abordagem, ela é denominada de **S2**.

Neste cenário, da mesma maneira que ocorre no cenário anterior, os nodos sensores conseguem se comunicar diretamente com a estação base até uma determinada distancia (r na Figura 6.4). Nesta abordagem existe um agente na estação base (**item A**) que se clona para os seus nodos vizinhos. Os agentes clones (**itens A1** e **A2**) movem-se para os seus nodos vizinhos de maior nível energético ainda não visitado até encontrar um determinado nodo que tenha monitorado um fenômeno alvo. Quando isto acontece, este agente deve enviar uma mensagem para o agente na estação base. Entretanto, existe a possibilidade de o agente clone estar em um nodo que não alcança diretamente o agente na estação base. Neste caso, o agente se move de volta para um ou mais nodos sensores anteriores até chegar a um nodo já visitado que consegue alcançar diretamente a estação base, podendo enviar a mensagem com o fenômeno alvo para o agente na estação base.

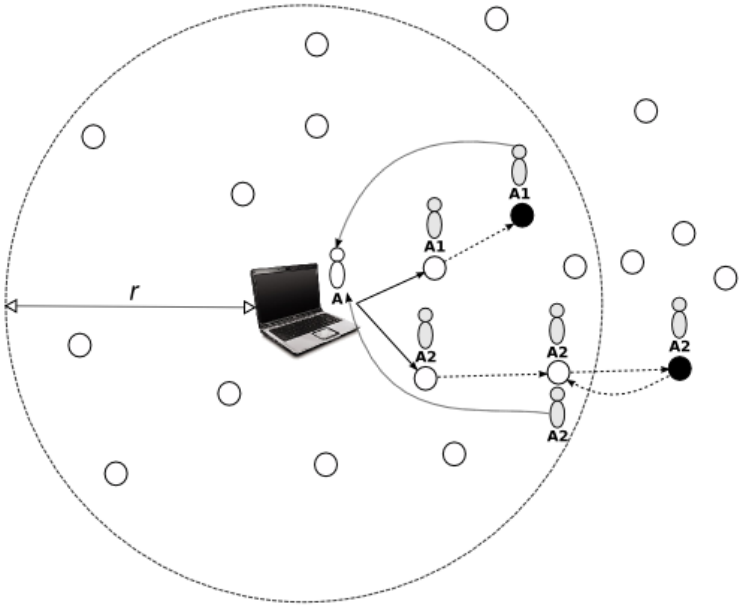

Figura 6.4: Representação do protocolo baseado em agentes da abordagem S2.

A missão dos agentes desta abordagem é representada pela Figura 6.5 e especificada pelo Algoritmo 6.2. Na Figura 6.5, os estados **0**, **3** e **4** são referentes aos estados do agente na estação base. Os estados dos agentes clones são representados por **1**, **2**, **5** e **6**.

Desta forma, a missão do agente nesta abordagem pode ser divida nos seguintes estados da máquina de estado:

- **Estado 0:** este estado é especificado pelo Algoritmo 6.2 entre as linhas **5** e **21**. Neste estado, o agente na estação base deve se clonar para todos os nodos vizinhos (linhas **11 – 15**). Quando

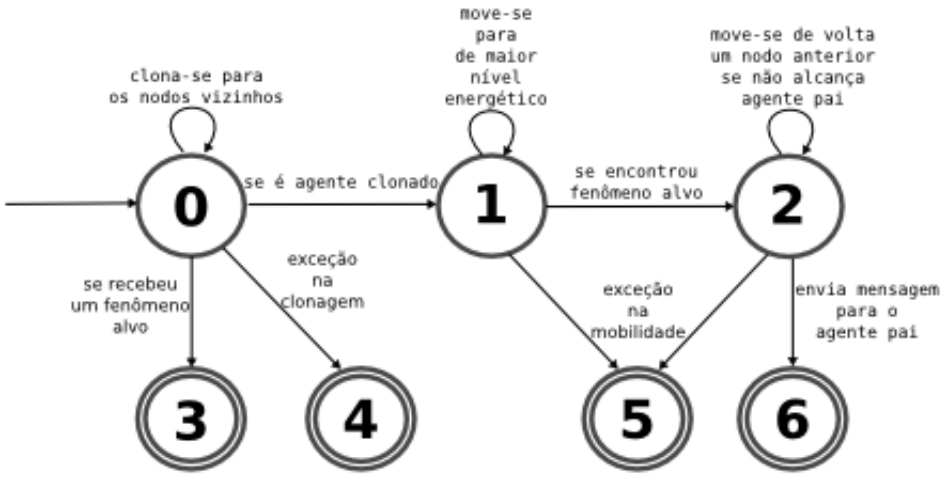

Figura 6.5: Modelo de especificação da missão dos agentes da abordagem S2.

ele recebe a mensagem com a notificação da ocorrência de algum evento alvo, ele vai para o estado final representado pelo **Estado 3**. Durante o processo de clonagem, alguma exceção pode ser lançada e esta deverá ser tratada no **Estado 4**;

- **Estado 1:** este estado é especificado pelo Algoritmo 6.2 entre as linhas **22** e **32**. Neste estado, o agente clone verifica se o nodo corrente monitorou o fenômeno alvo (linha **23**). O agente irá continuar a mover-se para o nodo vizinho não visitado com maior nível energético (linhas **38 – 43**) se o nodo corrente não tenha monitorado tal fenômeno. O próximo estado do agente é o **Estado 2**, caso ele encontre um nodo que tenha monitorado o evento alvo. Durante a mobilidade do agente pode surgir alguma exceção. Nesta situação, esta exceção deverá ser tratada no **Estado 5**;

- **Estado 2:** este estado é especificado pelo Algoritmo 6.2 nas linhas **33 – 53**. Neste estado, o agente verifica se o nodo corrente alcança diretamente a estação base (linhas **38 – 43**). Caso não alcance, o agente repetirá este processo até chegar a um nodo que isto seja verdade. Neste nodo, o agente enviará a mensagem de notificação para o agente na estação base (linha **46**) e finaliza sua missão no **Estado 6**. As exceções que podem ocorrer durante o processo de mobilidade do agente clone são tratadas no **Estado 5**;

- **Estado 3, 4, 5 e 6:** estes estados são finais e são especificados no Algoritmo 6.2 entre as linhas **54** e **56**. Os **Estados 4 e 5** devem tratar as exceções que podem ocorrer na clonagem e mobilidade, respectivamente. Quando as operações de receber fenômeno alvo e encontrar fenômenos são realizadas com sucesso, os estados finais **3 e 6** são invocados. Para simplificar

o entendimento deste algoritmo, estes estados apenas realizam a operação de finalizar a missão do agente.

```
0  void executaMissaoAg1(Agent* agent, int nextStep) {
1    // Definicao das variaveis agentID, agentPaiID, allNeighbor, neighborNodesList,
         nodoEscolhido, neighborNodesListLength, posList, reachFather
2
3    (*agent).getID(agentID);
4    switch (nextStep) {
5      case 0:
6         if (maGetAgentMessageReceived(agent) == NULL) {
7          bool noCloned = false;
8          allNeighbors = maGetAllNeighbors((*agent).getCurrentAgencyID());
9          int pos = 0;
10
11         while ( (!noCloned) && (pos < maGetLengthAllNeighbors((*agent).getCurrentAgencyID()
               )) ) {
12           if ( (!maExistClonedAgent(agent, allNeighbors[pos])) && (allNeighbors[pos] != (*
                 agent).selectLastVisitedNode()) ) {
13         noCloned = true;
14     } else {pos++;}
15         }
16
17         if (noCloned) {
18     maCloneAgent(agent, allNeighbors[pos], 1, 0, 1, 4);
19         } else {maGoToNextStep(agent, 0);}
20       } else {maGoToNextStep(agent, 3);}
21       break;
22     case 1:
23       if (maGetSensorData() == 1) {
24         maGoToNextStep(agent, 2);
25       } else {
26           nodoEscolhido = maGetMostEnergyLevelNeighbor((*agent).getCurrentAgencyID(), (*
                 agent).getListVisitedNodeID(), (*agent).lengthVisitedNode());
27           if (nodoEscolhido != -1) {
28             (*agent).storeVisitedNode((*agent).getCurrentAgencyID());
29             maMoveAgent(agent, nodoEscolhido, 1, 1, 6);
30           } else {maGoToNextStep(agent, 1);}
31       }
32       break;
33     case 2:
34       neighborNodesList = (*agent).getNeighborNodesList();
35       neighborNodesListLength = (*agent).getNeighborNodesListLength();
36       reachFather = false;
37       posList = 0;
38       while ( (!reachFather) && (posList < neighborNodesListLength) ) {
39         if (neighborNodesList[posList] == (*agent).getFatherNodeID()) {
40           reachFather = true;
41         }
```

```
42            posList++;
43          }
44          if (reachFather) {
45            (*agent).getFatherAgentID(agentPaiID);
46            maSendAgentMessage(agent, agentPaiID, (*agent).getFatherNodeID(), 1, (*agent).
                  getPhenomenonType(), agent, 5);
47          } else {
48             nodoEscolhido = (*agent).getLastVisitedNode();
49             if (nodoEscolhido != -1){
50               maMoveAgent(agent, nodoEscolhido, 1, 2, 5);
51             } else {maGoToNextStep(agent, 5);}
52          }
53          break;
54        case 3,4,5,6:
55          maAgentFinalState(agent);
56          break;
57      }
58  }
```

Algoritmo 6.2: Missão do agente da abordagem S2.

6.4 Simulação do protocolo de difusão simples

A abordagem baseada na difusão simples é representada pela Figura 6.6. Com o objetivo de facilitar a referência desta abordagem, ela é denominada **S3**. Nesta forma de difusão, cada nodo sensor deverá enviar uma mensagem para todos os seus vizinhos. A estação base irá analisar as mensagens que recebeu e verificará a ocorrência, ou não, de um fenômeno alvo.

Neste cenário, os nodos conseguem comunicar diretamente com a estação base, se o nodo estiver em uma determinada distancia desta estação base (r na Figura 6.6). Assim, existe a possibilidade de existirem nodos desconexos na rede.

Caso um nodo receba uma mensagem com evento monitorado de outro nodo, ele anexa a sua mensagem e encaminha para seus vizinhos. Assim, quando este fenômeno chega à estação base, a missão desta abordagem é concluída.

6.5 Simulação do protocolo de difusão seletiva

Este protocolo baseado em difusão de mensagem tem como objetivo ser mais eficiente do que o protocolo de difusão proposto na seção anterior com relação à economia de mensagens trocadas.

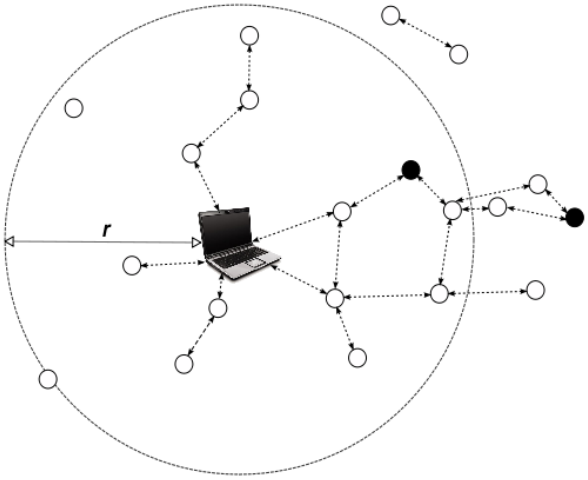

Figura 6.6: Representação do protocolo baseado em difusão de mensagens da abordagem S3.

Com o objetivo de facilitar a referência desta abordagem, ela é denominada **S4**. Nesta abordagem, representado pela Figura 6.7, apenas os nodos sensores que monitoraram ou que receberam alguma mensagem com fenômeno alvo devem fazer um *broadcast* de mensagem, para todos os seus vizinhos, sobre a ocorrência do fenômeno alvo.

Da mesma forma como ocorre no cenário da seção anterior, neste cenário, caso o nodo sensor esteja em uma determinada distancia desta estação base (r na Figura 6.7), este nodo consegue se comunicar diretamente com a estação base. Desta forma, nodos desconexos podem existir.

Nesta abordagem, apenas os nodos que monitoram um evento alvo ou que receberam mensagem com o evento alvo envia mensagem para a vizinhança. Assim, quando este fenômeno chega à estação base, a missão desta abordagem é concluída.

6.6 Comparação entre as abordagens baseadas em agentes e difusão de mensagens

Esta Seção realiza uma análise comparativa do desempenho em relação aos protocolos abordados nas seções anteriores deste Capítulo, os baseados em agentes e os que são baseados em difusão de mensagens. Foram realizadas 20 simulações em cada cenário (50 e 100 nodos). O tempo de cada uma das simulações foi de $100s$. O consumo de energia no nodo é realizado durante o processamento de informações e no envio/recebimento de mensagens no nodo. Assim, o recebimento e o envio de uma mensagem e processamento realizado pelo processador são gastos $16,62x10^{-4}J$, $9,6x10^{-4}J$ e

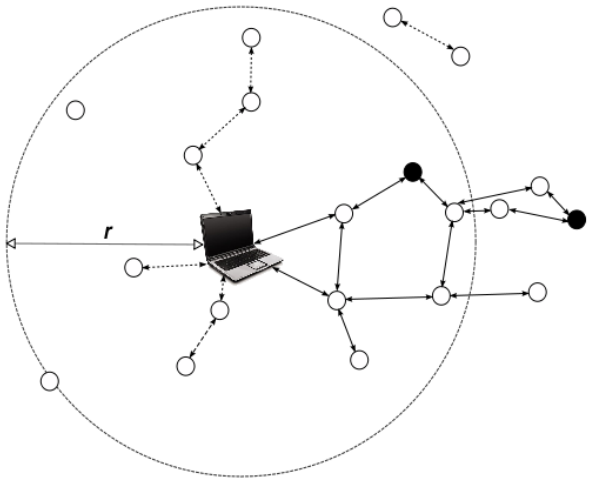

Figura 6.7: Representação do protocolo baseado em difusão de mensagens da abordagem S4.

$4,8x10^{-4}J$, respectivamente. Estes valores foram calculados seguindo as especificações do modelo do Mica2 (Polastre *et al.* [2004]).

A economia de energia em uma RSSF é um requisito muito importante que deve ser considerado quando se propõe algum protocolo de roteamento de mensagens. Assim, uma comparação do consumo médio de energia entre as quatro abordagens descritas anteriormente é necessária e está representada na Figura 6.8. Os protocolos baseados em agentes descritos nas seções 6.2 e 6.3 apresentam valores médios de consumo de energia representado no gráfico pelas colunas **S1** e **S2** respectivamente. Os protocolos baseados em difusão de mensagens descritos nas seções 6.4 e 6.5 são representados, respectivamente, no gráfico pelas colunas **S3** e **S4**. As informações mais detalhadas sobre os resultados destas simulações estão nos Apêndices G e H.

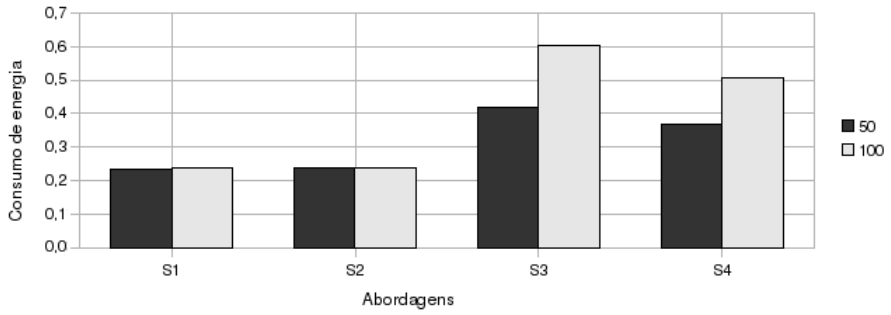

Figura 6.8: Comparativo de consumo de energia entre as abordagens.

As abordagens baseadas em agentes foram mais econômicas energeticamente comparadas à abordagem de difusão de mensagem. A simulação do primeiro cenário com 50 nodos na abordagem

de agentes **S1** apresentou gasto médio de energia por nodo sensor durante toda a simulação próximo de $0,236J$. Na simulação do segundo cenário com 100 nodos, a mesma abordagem teve um gasto médio de energia de aproximadamente de $0,240J$. Na segunda abordagem baseado em agentes **S2**, o consumo médio aproximado de energia por nodo sensor na simulação do primeiro e do segundo cenário foram, respectivamente, de $0,238J$ e $0,239J$. Assim, o consumo médio de energia dos nodos nas abordagens baseadas em agentes variou pouco mesmo com a variação da quantidade de nodos. Nota-se, desta forma, uma tendência de independência do consumo de energia dos nodos nesta abordagem mesmo variando a quantidade de nodos.

Na abordagem baseada em difusão de mensagens **S3**, o primeiro e segundo cenários apresentam gastos médios de, respectivamente, $0,418J$ e $0,601J$. A abordagem **S4** apresentou no primeiro e segundo cenário de simulação os gastos de energia de $0,367J$ e $0,508J$, respectivamente. Desta forma, pode-se afirmar que houve um aumento considerável do consumo energético pelos nodos nas abordagens baseadas em difusão. A motivação disto é que, com o aumento na quantidade de nodos, aumenta-se também a quantidade de mensagens recebidas para cada nodo.

As simulações realizadas demonstram que as abordagens de agentes são mais econômicas no consumo de energia se comparadas às abordagens de difusão de mensagens apresentadas neste trabalho. Através dos dados presentes nos resultados da simulação, observa-se que a abordagem de difusão é mais eficiente na cobertura da rede. Nas abordagens de difusão, a cobertura foi completa, ou seja, no final da simulação, todos os nodos foram alcançados pelo protocolo em todos os cenários. Entretanto, na abordagem baseada em agentes **S1** a cobertura média para o primeiro e segundo cenário foi de 14% e 12% respectivamente. Na abordagem baseada em agentes **S2**, a cobertura média para o primeiro e segundo cenário foi de 21% e 12%, respectivamente. Estes dados podem variar com mudanças na modelagem das missões dos agentes.

Além disto, nas simulações realizadas neste trabalho apresentaram também que a abordagem de agentes não foi tão eficiente em relação à difusão de mensagens comparado com o tempo de chegada da primeira mensagem com notificação da ocorrência do fenômeno alvo. Isto se deve o protocolo de difusão é mais simples, assim, tem menos troca de mensagens para obter uma determinada informação. Nos protocolos baseados em difusão, as mensagens sempre chegaram à média no tempo mínimo ($2s$). Contudo, a abordagem dos agentes **S1**, no primeiro e segundo cenário o tempo médio de chegada foi, respectivamente, $15s$ e $19s$. Além disto, houve simulações desta abordagem em que o agente da estação base não recebeu a mensagem sobre o fenômeno dentro do tempo da simulação. O tempo de recebimento da mensagem com fenômeno alvo na abordagem dos agentes **S2** referente ao primeiro e segundo cenário ambos de $15s$.

6.7 Considerações Finais

Este Capítulo teve como objetivo demonstrar a utilização da ferramenta proposta neste trabalho, o *MASiM*, através da sua aplicação em cenários de uso. O *MASiM* permite a especificação da topologia da rede através de um arquivo MDL para o ambiente do *MATLAB* composto por blocos do *Simulink*, *TrueTime* e blocos próprios da ferramenta. Assim, o *MASiM* permite a especificação e extensão de uma ampla variedade de cenários de simulação de agentes móveis em RSSF. As missões dos agentes podem ser especificadas e modeladas através de uma máquina de estados usando sintaxe da linguagem *C++ MEX*.

Além da demonstração da utilização da ferramenta proposta, outro objetivo deste Capítulo foi realizar uma análise comparativa entre dois protocolos baseado agentes móveis em relação a dois protocolos baseados em difusão. Desta maneira, através desta análise, foi verificada a eficiência dos protocolos baseados em agentes em relação à economia no consumo de energia. Contudo, também se verificou que estes protocolos não são eficientes no tempo de resposta da detecção de um fenômeno alvo e cobertura da rede com comparação aos protocolos baseados em difusão propostos neste trabalho.

Capítulo 7

Conclusão

Este trabalho teve como objetivo desenvolver uma ferramenta de simulação de agentes para RSSF denominado *MASiM* (*Mobile Agents SIMulator in wireless sensor network*). Através desta ferramenta é possível especificar cenários de simulação de protocolos baseados em agentes estáticos e móveis em RSSF de maneira flexível. Esta flexibilidade é alcançada através da disponibilização de funções e diagramas de blocos que facilitam a programação de cenários e da missão dos agentes, como também o modelo de estados dos agentes (móveis ou estáticos), que permite determinar diferentes comportamentos deste agente durante seu ciclo de vida.

Além disto, esta ferramenta também permite estender os blocos de definição de topologia da rede. Assim, é possível especificar, agrupar e criar novos blocos baseados nos blocos já existentes disponibilizados por esta ferramenta, permitindo, assim, modelar cenários de forma mais prática. Esta característica é possível, pois o MASiM é baseada nos diagramas de blocos do *Simulink* e *TrueTime* do *MATLAB*.

A interface de programação do *MASiM* teve como objetivo possibilitar a utilização de funções e estruturas de dados que permitem ao usuário programar os modelos de simulação. Assim, a topologia da rede é definida através dos diagramas de blocos disponíveis e a programação dos nodos e definição da missão dos agentes são realizadas através da linguagem *C++ MEX*. A missão de um determinado agente é modelada como uma máquina de estados. Os blocos disponibilizados são referentes à RSSF, nodo sensor e estação base.

Neste trabalho, considera-se que o agente é capaz de se mover entre nodos que apresentam um ambiente especial para execução do agente denominado agência. A agência irá disponibilizar e escalonar recursos que podem ser acessados pelo agente móvel. Algumas informações disponibilizadas por este ambiente para o agente móvel são: relógio local, espaço de memória compartilhada

entre os agentes no mesmo nodo, informações sobre a vizinhança, valores dos sensores, obtenção do nível energético do nodo corrente, envio e recebimento de mensagens entre os agentes.

Uma contribuição também importante realizada neste trabalho foi os resultados obtidos nas simulações dos cenários baseados em agentes móveis e difusão de mensagens em uma RSSF. Nestas simulações, verificaram-se as possíveis vantagens e desvantagens em utilizar a abordagem de agentes móveis ou de difusão. A economia de energia obtida na simulação da abordagem de agentes móveis foi à principal vantagem em relação à abordagem baseada na difusão de mensagens. Contudo, a principal desvantagem na utilização de agentes móveis em relação à difusão nestas simulações foi a curta cobertura da rede.

7.1 Perspectiva de Trabalhos Futuros

Alguns aperfeiçoamentos deverão ser realizados nas próximas versões desta ferramenta. A primeira delas é a flexibilidade das informações sobre a vizinhança. Atualmente, a mensagem que é enviada periodicamente pelo nodo para os seus vizinhos contém apenas informações sobre o nível energético do nodo. Portanto, na próxima versão desta ferramenta, pretende-se permitir que informações personalizadas pelo usuário sejam enviadas pelo nodo. Além disso, é esperado que o protocolo de negociação de mobilidade do agente possa ser definido também pelo programador. Algumas necessidades mais imediatas deverão ser desenvolvidas, como a possibilidade de realizar a programação, utilizando também a própria linguagem do *MATLAB*. Existem também a necessidade de se desenvolver uma interface gráfica que permita a geração dos blocos da topologia de maneira mais prática e automática de como é realizado atualmente, em que a criação e a ligação dos blocos é feita manualmente.

Referências Bibliográficas

Agilla (2009). Agilla - a mobile agent middleware for wireless sensor networks. "http://www.cs.wustl.edu/mobilab/projects/agilla/". Acessado em 28 de Junho de 2009.

Alliance, Z. (2008). *Zigbee Specification*. ZigBee Standards Organization, 2400 Camino Ramon, Suite 375, San Ramon,.

Alsalih, W., Akl, S., e Hassanein, H. (2007). Placement of multiple mobile base stations in wireless sensor networks. *Proceedings of the IEEE Symposium on Signal Processing and Information Technology(ISSPIT)*, Vol. 15pp. 229 – 233.

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., e Sorensen, D. (1990). Lapack: A portable linear algebra library for high-performance computers. "http://www.netlib.org/lapack". Acessado em 17 de Junho de 2009.

Ansi (1995). Ansi/iso c++ draft standards. "http://www.csci.csusb.edu/dick/c++std/". Acessado em 19 de Novembro de 2009.

Arampatzis, T., Lygeros, J., e Manesis, S. (2005). A survey of applications of wireless sensors and wireless sensor networks. pp. 719 – 724. Intelligent Control, 2005. Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation.

Berkeley (2004). Tinyos. "http://www.tinyos.net". Acessado em 12 de março de 2009.

Cayirci, E., Akyildiz, I. F., Su, W., e Sankarasubramaniam, Y. (2002). A survey on sensor networks. *IEEE Communications Magazine*, Vol. 40pp. 102– 114.

Chen, G., Branch, J., Pflug, M. J., Zhu, L., e Szymanski, K. (2004). Sense: A sensor network simulator. http://www.cs.rpi.edu/ szymansk/papers/wpcn.04.pdf. Acessado em 19 de Agosto de 2009.

Chen, M., Kwon, T., Yuan, Y., Choi, Y., e Leung, V. C. M. (2007). Mobile agent-based directed diffusion inwireless sensor networks. *EURASIP Journal on Advances in Signal Processing*, Vol. 2007pp. 13.

Chen, M., Kwon, T., Yuan, Y., e Leung, V. C. (2006). Mobile agent based wireless sensor networks. *Journal of Computers*, Vol. 1.

Coulouris, G., Dollimore, J., e Kindberg, T. (2007). *Sistemas DIstribuídos - Conceitos e Projeto*. 4º edição.

Crossbow (2009). Mica2 - wireless measurement system. "`http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`". Acessado em 18 de Junho de 2009.

Digi (2009). http://www.digi.com/technology/rf-articles/wireless-zigbee.jsp. Acessado em Novembro de 2009.

Dongarra, J., Bunch, J., Moler, C., e Stewart, P. (1984). Linpack. "`http://www.netlib.org/linpack`". Acessado em 17 de Junho de 2009.

Estrin, D., Handley, M., Heidemann, J., McCanne, S., Xu, Y., e Yu, H. (2000). Network visualization with nam, the vint network animator. *Computer*, Vol. 33pp. 63 – 68.

Fall, K. e Varadhan, K. (2008). *The ns Manual*. UC Berkeley and LBL and USC/ISI and Xerox PARC.

FIPA (2004). Fipa agent management specification. "`http://www.netlib.org/lapack`". Acessado em 29 de Junho de 2009.

Günes, M. H., Türsem, M. E., Yildiz, M., e Kuru, S. (2003). Performance analysis of mobile agents using simulation. Proc. *Proceedings of the Advanced Engineering Design Conference (AED2003)*, Praga, República Tcheca.

Hadzilacos, V. e Toueg, S. (1994). A modular approach to fault-tolerant broadcasts and related problems. Relatório técnico, Cornell University.

Heinzelman, W. R., Chandrakasan, A., e Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. *Proceedings of the Hawaii International Conference on System Science*, Vol. 2pp. 10.

Heo, N. e Varshney, P. K. (2003). An intelligent deployment and clustering algorithm for a distributed mobile sensor network. *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 5pp. 4576 – 4581.

Hu, L. e Evans, D. (2004). Localization for mobile sensor networks. Proc. *Proceedings of the 10th annual international conference on Mobile computing and networking*, pp. 45 – 57, Philadelphia, PA, EUA. ACM.

IEEE (2005). Part 15.1:wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (wpans). Relatório técnico, IEEE Computer Society.

IEEE (2006). Part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (wpans). Relatório técnico, IEEE Computer Society.

IEEE (2007). Part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. Relatório técnico, IEEE Computer Society.

Ilyas, M. e Mahgoub, I. (2005). *Handbook of Sensor Networks: Compact Wireless and Wired Sensing System*. CRC Press LLC.

Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J., e Silva, F. (2003). Directed diffusion forwireless sensor networking. Proc. *IEEE/ACM Transactions on Networking (TON)*, Vol. 11, pp. 2 – 16, Piscataway, NJ, EUA. IEEE Press.

Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley & Sons, Wiley- Interscience, New York, NY.

Jansen, W., Mell, P., Karygiannis, T., e Marks, D. (1999). Applying mobile agents to intrusion detection and response. Relatório técnico, National Institute of Standards and Technology Computer Security Division, Washington, D.C, EUA.

JSim (2005). The autonomous component architecture. "`http://www.j-sim.org/whitepapers/aca.html`". Acessado em 17 de Junho de 2009.

Karl, H. e Willig, A. (2003). A short survey of wireless sensor networks. Relatório técnico, Technical University Berlin, Telecommunication Networks Group.

Karlof, C. e Wagner, D. (2003). Secure routing in wireless sensor networks: attacks and countermeasures. Proc. *IEEE International Workshop on Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE*, pp. 113 – 127.

Keshav, S. (1997). Real 5.0 overview. "http://www.cs.cornell.edu/skeshav/real/overview.html". Acessado em 17 de Junho de 2009.

Koushanfar, F., Potkonjak, M., e Sangiovanni-Vincentelli, A. (2002). Fault tolerance techniques for wireless ad hoc sensor networks. *Proceedings of IEEE Sensors*, Vol. 2pp. 1491 – 1496.

Krishnamurthy, S. e Zeid, I. (2004). Distributed and intelligent information access in manufacturing enterprises through mobile devices. Proc. *Journal of Intelligent Manufacturing*, pp. 175 – 186. Kluwer Academic.

Lange, D. B. e Oshima, M. (1999). Seven good reasons for mobile agents. *Commun. ACM*, Vol. 42, No. 3, pp. 88–89.

Larman, C. (2001). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Prentice Hall PTR, 2º edição.

Lawson, C. L., Hanson, R. J., Kincaid, D. R., e Krogh, F. T. (2009). Blas (basic linear algebra subprograms). "http://www.netlib.org/blas". Acessado em 17 de Junho de 2009.

Levis, P., Lee, N., Welsh, M., e Culler, D. (2003). Tossim: accurate and scalable simulation of entire tinyos applications. Proc. *Proceedings of the 1st international conference on Embedded networked sensor systems*, pp. 126 – 137, Los Angeles, California, EUA. ACM.

Lindsey, S. e Raghavendra, C. S. (2002). Pegasis: Power-efficient gathering in sensor information systems. *Aerospace Conference Proceedings*, Vol. 3pp. 1125 – 1130.

Liu, J. W. S. (2000). *Real-Time Systems*. Prentice Hall, Upper Saddle RIver, New Jersey 07458.

Loureiro, A. A., Nogueira, J. M. S., Ruiz, L. B., de Freitas Mini, R. A., Nakamura, E. F., e Figueiredo, C. M. S. (2003). Redes de sensores sem fio. *Simpósio Brasileiro de Rede de Computadores (SBRC)*, Vol.

Macêdo, R. J. A. e Silva, F. M. A. (2005). The mobile groups approach for the coordination of mobile agents. Proc. *Journal of Parallel and Distributed Computing*, Vol. 65, pp. 275 – 288, Orlando, FL, EUA.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., e Anderson, J. (2002). Wireless sensor networks for habitat monitoring. Proc. *International Workshop on Wireless Sensor Networks and Applications*, pp. 88 – 97, Atlanta, Georgia, EUA. Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications.

Malik, H. e Shakshuki, E. (2007). Data dissemination in wireless sensor networks using software agents. Proc. *Annual International Symposium on High Performance Computing Systems and Applications*, pp. 28, Saskatoon, Saskatchewan, Canadá. IEEE Computer Society.

Manjeshwar, A. e Agrawal, D. P. (2001). Teen: a routing protocol for enhanced efficiency in wireless sensor networks. Proc. *Proceedings 15th International Parallel and Distributed Processing Symposium*, pp. 2009 – 2015, San Francisco, California, EUA.

Manjeshwar, A. e Agrawal, D. P. (2003). The acquire mechanism for efficient querying in sensor networks. Proc. *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, pp. 149 – 155, Los Angeles, CA, EUA.

Manjeshwar, A. e Agrawaly, D. P. (2002). Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. Proc. *Parallel and Distributed Processing Symposium*, pp. 195 – 202, Fort Lauderdale, Florida, EUA.

Massaguer, D. (2005). Multi mobile agent deployment in wireless sensor networks. Dissertação de Mestrado, University of California.

MathWorks, T. (2009). *MATLAB Getting Started Guide*. The MathWorks, Inc.

Meyering, C., Meyering, J., e Dubrulle, A. (2009). Eispack. "`http://www.netlib.org/eispack`". Acessado em 17 de Junho de 2009.

Ohlin, M., Henriksson, D., e Cervin, A. (2007). *TrueTime 1.5 - Reference Manual*. Department of Automatic Control, Lund University.

OMG (1997). Mobile agent system interoperability facilities specification. "`http://www.omg.org`". Acessado em 24 de Junho de 2009.

Polastre, J., Hill, J., e Culler, D. (2004). Versatile low power media access for wireless sensor networks. Proc. *Conference On Embedded Networked Sensor Systems and Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 95 – 107, Baltimore, MD, EUA. ACM.

Polley, J., Blazakis, D., McGee, J., Rusk, D., e Baras, J. (2004). Atemu: a fine-grained sensor network simulator. Proc. *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004*, pp. 145 – 152.

Qi, H., Xu, Y., e Wang, X. (2003). Mobile-agent-based collaborative signal and information processing in sensor networks. Proc. *Proceedings of the IEEE*, Vol. 91, pp. 1172 – 1183.

Raghunathan, V., Schurgers, C., Park, S., e Srivastava, M. B. (2002). Energy aware wireless sensor networks. *IEEE Signal Processing*, Vol. 19pp. 40 – 50.

Rodoplu, V. e Meng, T. H. (1999). Minimum energy mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, Vol. 17.

Sadagopan, N., Krishnamachari, B., e Hel, A. (1999). Mobile ad hoc networking and the ietf. *ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 1pp. 11 – 13.

Silva, F. A. (1999). Agentes móveis. Proc. *Escola de Informática da SBC - Edição Nordeste (EINÉ99)*, Salvador, BA.

Simulink (2009). *Simulink 7 - Simulation and Model-Based Design*. The MathWorks. Acessado em 17 de Junho de 2009.

Sobeih, A., Chen, W.-P., Hou, J. C., Kung, L.-C., Li, N., Lim, H., ying Tyan, H., e Zhang, H. (2005). J-sim: A simulation and emulation environment for wireless sensor networks. *IEEE Wireless Communications magazine*, Vol. 13pp. 2006.

Sohrabi, K., Gao, J., Ailawadhi, V., e Pottie, G. J. (2000). Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, Vol. 7pp. 16 – 27.

Tong, L., Zhao, Q., e Adireddy, S. (2003). Sensor networks with mobile agents. Proc. *2003 Military Communications Intl Symp in Proc*, pp. 688 – 693.

Tseng, Y.-C., Kuo, S.-P., Lee, H.-W., e Huang, C.-F. (2002). A mobile-agent approach for location tracking in a wireless sensor network. Proc. *Proc. International Computer Symposium*.

Xue, Y., Lee, H. S., Yang, M., Kumarawadu, P., Ghenniwa, H., e Shen, W. (2007). Performance evaluation of ns-2 simulator for wireless sensor networks. Proc. *Canadian Conference on Electrical and Computer Engineering, 2007. CCECE 2007*, pp. 1372 – 1375.

Ye, W., Heidemann, J., e Estrin, D. (2001). An energy-efficient mac protocol for wireless sensor networks. pp. 1567–1576.

Yi, Z. e Chakrabarty, K. (2003). Energy-aware target localization in wireless sensor networks. Proc. *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pp. 60 – 67.

Yu, Y., Krishnamachari, B., e Prasanna, V. K. (2004). Issues in designing middleware for wireless sensor networks. *IEEE Network*, Vol. 18pp. 15 – 21.

Apêndice A

Casos de Uso do MASiM

Código:	UC01
Nome:	Definir topologia da rede
Atores:	Usuário
Pré-condição:	Arquivo com especificação das variáveis de simulação definido
Pós-condição:	Topologia definida
Fluxo-base:	1. O Usuário seleciona os blocos de rede e dos nodos; 2. O Usuário define a quantidade de nodos; 3. O Usuário define os blocos de fenômenos a serem monitorados; 4. O Usuário conecta os blocos; 5. O Usuário configura parâmetros dos blocos de rede e dos nodos.
Requisitos associados:	11, 13, 15 e 16

Código:	UC02
Nome:	Definir parâmetros e variáveis de simulação
Atores:	Usuário
Pré-condição:	-
Pós-condição:	Definidas os parâmetros e variáveis de simulação
Fluxo-base:	1. O Usuário inicializa os parâmetros do sistema; 2. O Usuário define as variáveis globais para a simulação; 3. O Usuário inicializa valores das variáveis globais para a simulação; 4. O Usuário define as variáveis estatísticas; 5. O Usuário inicializa as variáveis estatísticas.
Requisitos associados:	10, 11, 12, 14, 15 e 16

Código:	UC03
Nome:	Especificar missão do agente
Atores:	Usuário
Pré-condição:	-
Pós-condição:	Variáveis de simulação definidas
Fluxo-base:	1. O Usuário especifica os estados da missão do agente; 2. O Usuário define as operações em cada estado; 3. O Usuário define o fluxo de execução dos estados do agente.
Requisitos associados:	1, 2, 3, 4, 5, 6, 7, 8, 9, 10 e 12

Código:	UC04
Nome:	Obter dados da simulação
Atores:	Usuário
Pré-condição:	Simulação de um determinado cenário executado
Pós-condição:	Apresenta gráficos e tabelas com dados estatísticos definidos
Fluxo-base:	1. O Usuário seleciona as variáveis estatísticas; 2. O Usuário escolhe modo de visualização das variáveis; 2.1. Modo impressão dos valores brutos escolhido; 2.2. Modo impressão dos valores em tabela escolhido; 2.3. Modo de apresentação em gráfico escolhido.
Requisitos associados:	4, 5, 10 e 12

Apêndice B

Arquivo de Configuração Inicial do MASiM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% File of start setup of Wirless Sensor Network %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Setup Wireless                       %
%                                                        %
% numberNoder:  Node number of WSN                       %
% dataRate: Data rate (bits/s)                           %
% minFrameSize: Minimum frame size (bits)                %
% transPower: Transmit power (dbm)                       %
% recSignalThre: Receiver Signal Threshold (dbm)%
% pathLoss: Pathloss exponent (1/distance x)             %
% ackTimeout: ACK Timeout (s)                            %
% retryLimit: Retry Limit                                %
% errorCodingThre: Error Coding Threshold                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;

global numberNode;
global dataRate;
global minFrameSize;
global transPower;
global recSignalThre;
global pathloss;
global ackTimeout;
global retryLimit;
global errorCodingThre;
global positionNodeX;
```

```
31  global positionNodeY;
32
33  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
34  %                        Define a WSN                                    %
35  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36  numberNode(1) = 100;
37  dataRate(1) = 250000;
38  minFrameSize(1) = 248;
39  transPower(1) = -20;%
40  recSignalThre(1) = -90;%
41  pathloss(1) = 3.5;
42  ackTimeout(1) = 0.000864;
43  retryLimit(1) = 3;
44  errorCodingThre(1) = 0.01;
45
46  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47  %                     Setup Sensor Nodes                                 %
48  %                                                                        %
49  % clockDriff: Clock driff                                               %
50  % clockOffset: Clock offset                                             %
51  % typePhenomenon:  Type of phenomenon                                   %
52  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
53  global clockDriff;
54  global clockOffset;
55  global typePhenomenon;
56  global batteryNode;
57  global faultNodes;
58
59
60  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61  %                  Define Nodes of WSN # 1                               %
62  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63
64  for id=1:numberNode(1)
65      clockDriff(1,id) = 0;
66      clockOffset(1,id) = 0;
67      typePhenomenon(1,id) = 1;
68      batteryNode(id) = 2.5;
69      memorySize(1,id) = 16;
70
71      % Define the fault nodes (#10)
72      %switch (id)
73      %  case {5, 10, 15, 20, 25, 30, 35, 40, 45, 50}
74      %      faultNodes(id) = 1;
75      %      transientFaultNodes(id) = 25;
76      %  otherwise
77      %      faultNodes(id) = 0;
78      %      transientFaultNodes(id) = 0;
79      %end;
```

```
end;

load('posX.mat');
load('posY.mat');

positionNodeX = posX;
positionNodeY = posY;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Setup of The Tasks                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%   Task of node   %%%%%%%%%%%%%%%%%%%%%%%%%%

global rcvMsgNodeTaskEnergy;
rcvMsgNodeTaskEnergy = 0.00048;

global sentMsgNodeTaskEnergy;
sentMsgNodeTaskEnergy = 0.3684;

global handleSimpleMsgTaskEnergy;
handleSimpleMsgTaskEnergy = 0.00048;

global notifySentMsgNodeTaskEnergy;
notifySentMsgNodeTaskEnergy = 0.00048;

global handleNotifySentMsgNodeTaskEnergy;
handleNotifySentMsgNodeTaskEnergy = 0.00048;

%%%%%%%%%%%%%%%%  Task of agency  %%%%%%%%%%%%%%%%%%%%%%%%%%

global rcvMsgAgyTaskEnergy;
rcvMsgAgyTaskEnergy = 0.00048;

global executeAgentTaskEnergy;
executeAgentTaskEnergy = 0.00048;

global createAgentTaskEnergy;
createAgentTaskEnergy = 0.00048;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Setup of The Agents                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global initialAgentSize;
global arriveTime;

initialAgentSize = 232;
arriveTime = 0;

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Variaveis estatiscas                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global QuantFaultNodeFound;
QuantFaultNodeFound = 0;

global QuantVisitedNodes;
QuantVisitedNodes = 0;
```

Algoritmo B.1: Arquivo de configuração do simulador.

Apêndice C

Amostra da Topologia da Rede

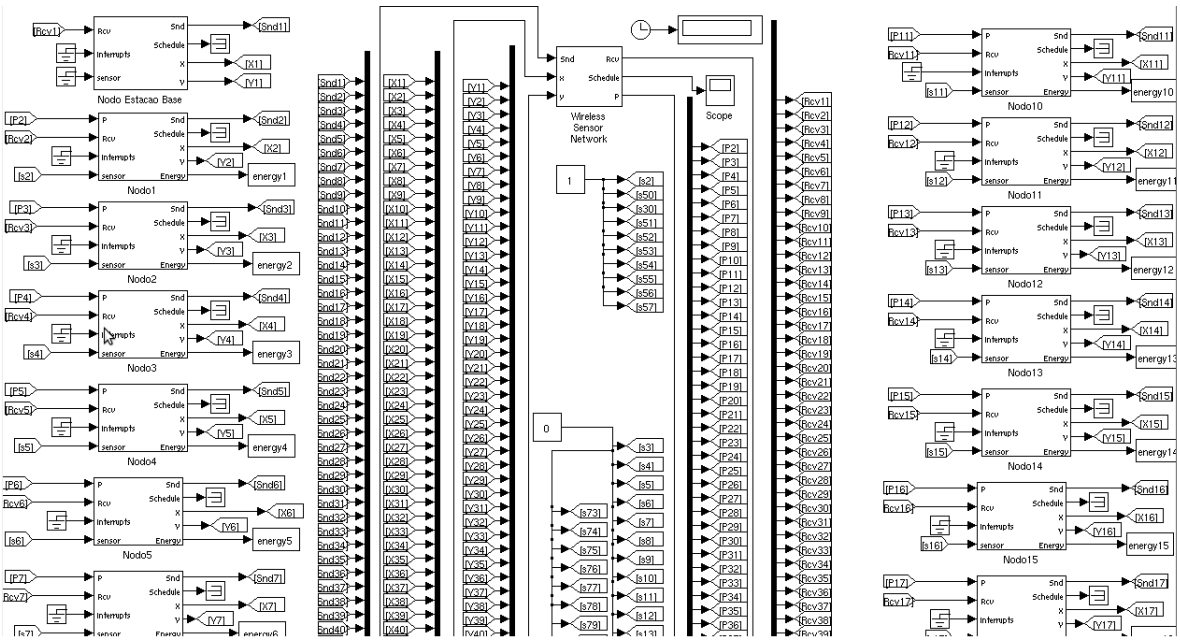

Figura C.1: Amostra de topologia da rede.

Apêndice D

Configuração da Organização do Cenário

```
#define S_FUNCTION_NAME nodes_init

#include "class/defconstants.h"

#include "createagency.cpp"
#include "createagent.cpp"
#include "definenodecontroller.cpp"
#include "getnodeid.cpp"
#include "putconfigurationvariable.cpp"

#include "class/agent.cpp"
#include "class/simulationconfigure.cpp"

#include "missaoag1.cpp"


#ifndef _MEXLIB
#include "mex.h"
#include "ttkernel.cpp"
#define _MEXLIB 1
#endif

SimulationConfigure* sc = NULL;

void init() {
  sc = new SimulationConfigure("base");

  rtsys->energyConsumption = 0.00042;

  maDefineNodeController(1, sc);

  int id = maGetNodeID();

```

```
34    maCreateAgency(id, PHENO_TYPE_SOUND, sc);
35
36    if (id == 1) {
37      maCreateAgent(id, PHENO_TYPE_SOUND, "Agent1", 0, missaoAg1);
38    }
39
40  }
41
42  void cleanup(){
43  }
```

Algoritmo D.1: Configuração da organização do cenário de simulação.

Apêndice E

Missão do Agente

```
1  #include "gotonextstep.cpp"
2  #include "moveagent.cpp"
3  #include "cloneagent.cpp"
4  #include "agentfinalstate.cpp"
5  #include "getagentmessagereceived.cpp"
6  #include "getmostenergylevelneighbor.cpp"
7  #include "getsensordata.cpp"
8  #include "getallneighbors.cpp"
9  #include "getlengthallneighbors.cpp"
10 #include "existclonedagent.cpp"
11 #include "sendagentmessage.cpp"
12
13 #include "getconfigurationvariable.cpp"
14 #include "putconfigurationvariable.cpp"
15
16 #ifndef _MEXLIB
17 #include "mex.h"
18 #include "ttkernel.cpp"
19 #define _MEXLIB 1
20 #endif
21
22 void missaoAg1(Agent* agent, int nextStep) {
23   // Idenficador do agente
24   char agentID[150];
25   char agentPaiID[150];
26   int* allNeighbors = NULL;
27   int* neighborNodesList = NULL;
28   int nodoEscolhido, neighborNodesListLength, posList;
29   bool reachFather;
30
31   (*agent).getID(agentID);
32   switch (nextStep) {
33     case 0:
```

```
34        ssPrintf("[%f] Agente %s verifica se recebeu mensagem\n", ttCurrentTime(), agentID);
35        if (maGetAgentMessageReceived(agent) == NULL) {
36
37          ssPrintf("[%f] Agente %s verifica se pode realizar clonagem \n", ttCurrentTime(),
                agentID);
38          bool noCloned = false;
39          allNeighbors = maGetAllNeighbors((*agent).getCurrentAgencyID());
40          int pos = 0;
41
42          while ( (!noCloned) && (pos < maGetLengthAllNeighbors((*agent).getCurrentAgencyID()
                )) ) {
43            if ( (!maExistClonedAgent(agent, allNeighbors[pos])) && (allNeighbors[pos] != (*
                  agent).selectLastVisitedNode()) ) {
44              noCloned = true;
45        } else {
46            pos++;
47        }
48          }
49
50          if (noCloned) {
51        ssPrintf("[%f] Agent %s clona de %i para o nodo %i \n", ttCurrentTime(), agentID, (*
              agent).getCurrentAgencyID(), allNeighbors[pos]);
52        maCloneAgent(agent, allNeighbors[pos], 1, 0, 1, 4);
53          } else {
54          ssPrintf("[%f] Agent %s espera por mensagens de vizinhanÃ§a em %i\n", ttCurrentTime()
              , agentID, (*agent).getCurrentAgencyID());
55          maGoToNextStep(agent, 0);
56          }
57
58        } else {
59            ssPrintf("%f :: Agent %s recebeu uma mensagem e vai para o estado final\n",
                  ttCurrentTime(), agentID);
60            maGoToNextStep(agent, 3);
61        }
62
63        break;
64      case 1:
65        ssPrintf("[DEBUG] %f missao1 :: Agente clone %s verifica se o nodo %i tem o fenomeno
              alvo\n", ttCurrentTime(), agentID, (*agent).getCurrentAgencyID());
66        if (maGetSensorData() == 1) {
67          ssPrintf("[DEBUG] %f missao1 :: Agente clone %s encontra o fenomeno alvo no nodo %i
                \n", ttCurrentTime(), agentID, (*agent).getCurrentAgencyID());
68          // Tenta enviar mensagem
69          maGoToNextStep(agent, 2);
70        } else {
71            ssPrintf("[DEBUG] %f missao1 :: Agente clone %s nÃ£o encontra o fenÃ´meno alvo em
                  %i e tenta mover-se para outro nodo\n", ttCurrentTime(), agentID, (*agent).
                  getCurrentAgencyID());
72            // Move-se
```

```
73            nodoEscolhido = maGetMostEnergyLevelNeighbor((*agent).getCurrentAgencyID(), (*
                  agent).getListVisitedNodeID(), (*agent).lengthVisitedNode());
74            if (nodoEscolhido != -1) {
75              ssPrintf("[DEBUG] %f missao1 :: Agente clone %s em %i ira se mover para %i\n",
                    ttCurrentTime(), agentID, (*agent).getCurrentAgencyID(), nodoEscolhido);
76              (*agent).storeVisitedNode((*agent).getCurrentAgencyID());
77              maMoveAgent(agent, nodoEscolhido, 1, 1, 6);
78
79            } else {
80                // Tenta de novo
81                ssPrintf("[DEBUG] %f missao1 :: Agente clone %s em %i e tenta mover-se de
                      novo para outro nodo\n", ttCurrentTime(), agentID, (*agent).
                      getCurrentAgencyID());
82                maGoToNextStep(agent, 1);
83            }
84
85        }
86
87        break;
88      case 2:
89        // Se for alcanÃ§Ã¡vel, envia mensagem para a estaÃ§Ã£o base
90        // NÃ£o Ã©, volta um nodo (outro estado)
91        neighborNodesList = (*agent).getNeighborNodesList();
92        neighborNodesListLength = (*agent).getNeighborNodesListLength();
93        reachFather = false;
94        posList = 0;
95
96        ssPrintf("[DEBUG] %f missao1 :: Agente clone %s em %i verifica se Ã© alcancavel a EB\
              n", ttCurrentTime(), agentID, (*agent).getCurrentAgencyID());
97        while ( (!reachFather) && (posList < neighborNodesListLength) ) {
98          if (neighborNodesList[posList] == (*agent).getFatherNodeID()) {
99            reachFather = true;
100         }
101
102         posList++;
103       }
104
105       if (reachFather) {
106         ssPrintf("[DEBUG] %f missao1 :: Agente clone %s em %i Ã© alcancavel a EB e envia
                  mensagem em %i\n", ttCurrentTime(), agentID, (*agent).getCurrentAgencyID(), (*
                  agent).getFatherNodeID());
107         (*agent).getFatherAgentID(agentPaiID);
108         maSendAgentMessage(agent, agentPaiID, (*agent).getFatherNodeID(), 1, (*agent).
                  getPhenomenonType(), agent, 5);
109       } else {
110          ssPrintf("[DEBUG] %f missao1 :: Agente clone %s em %i nao Ã© alcancavel a EB\n",
                   ttCurrentTime(), agentID, (*agent).getCurrentAgencyID());
111          // Move de volta
112          nodoEscolhido = (*agent).getLastVisitedNode();
```

```
113             if (nodoEscolhido != -1){
114               ssPrintf("[DEBUG] %f missao1 :: Agente clone %s em %i move-se de volta para %i\n
                      ", ttCurrentTime(), agentID, (*agent).getCurrentAgencyID(), nodoEscolhido);
115               maMoveAgent(agent, nodoEscolhido, 1, 2, 5);
116             } else {
117                 // Chegou ao nodo origem, Agente vai morrer
118                 ssPrintf("[DEBUG] %f missao1 :: Agente clone %s em %i irÃ¡ morrer\n",
                        ttCurrentTime(), agentID, (*agent).getCurrentAgencyID());
119                 maGoToNextStep(agent, 5);
120             }
121         }
122
123         break;
124       case 3:
125         ssPrintf("[%f] Agente %s morre\n", ttCurrentTime(), agentID);
126         maAgentFinalState(agent);
127
128         break;
129       case 4:
130         ssPrintf("[%f] Agente %s exceÃ§Ã£o\n", ttCurrentTime(), agentID);
131         maAgentFinalState(agent);
132
133         break;
134       case 5:
135         ssPrintf("[%f] Agente clone %s morre\n", ttCurrentTime(), agentID);
136         maAgentFinalState(agent);
137
138         break;
139       case 6:
140         ssPrintf("[%f] Agente clone %s exceÃ§Ã£o\n", ttCurrentTime(), agentID);
141         maAgentFinalState(agent);
142
143         break;
144     }
145 }
```

Algoritmo E.1: Exemplo de missão do agente.

Apêndice F

Gráficos do MASiM

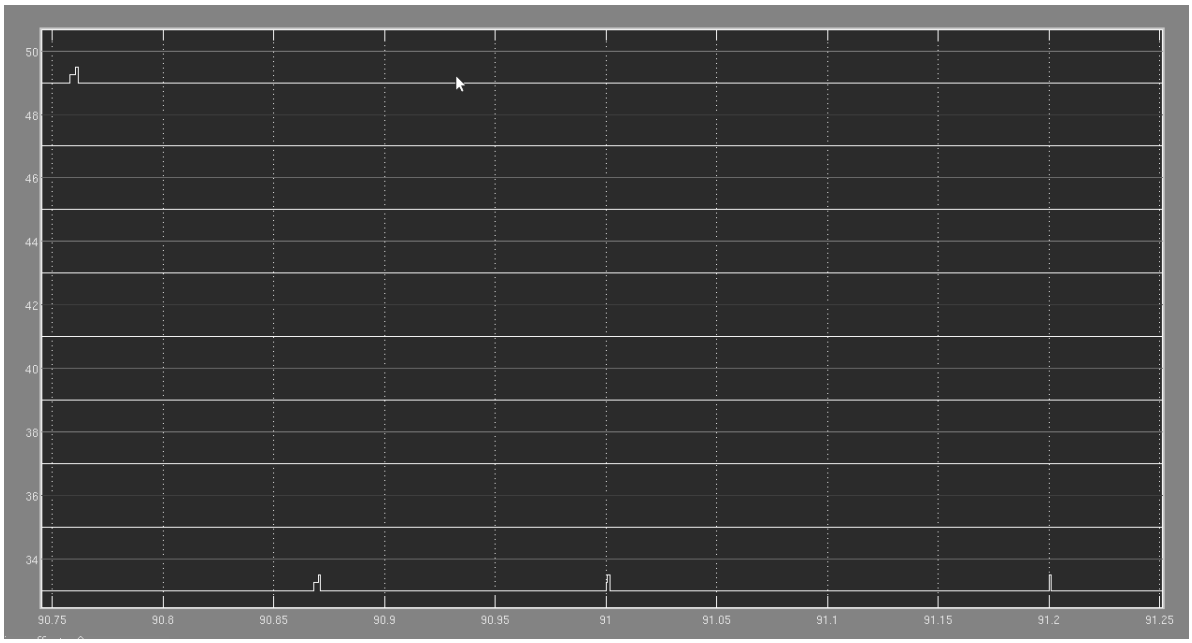

Figura F.1: Gráfico com o escalonamento de mensagem na rede.

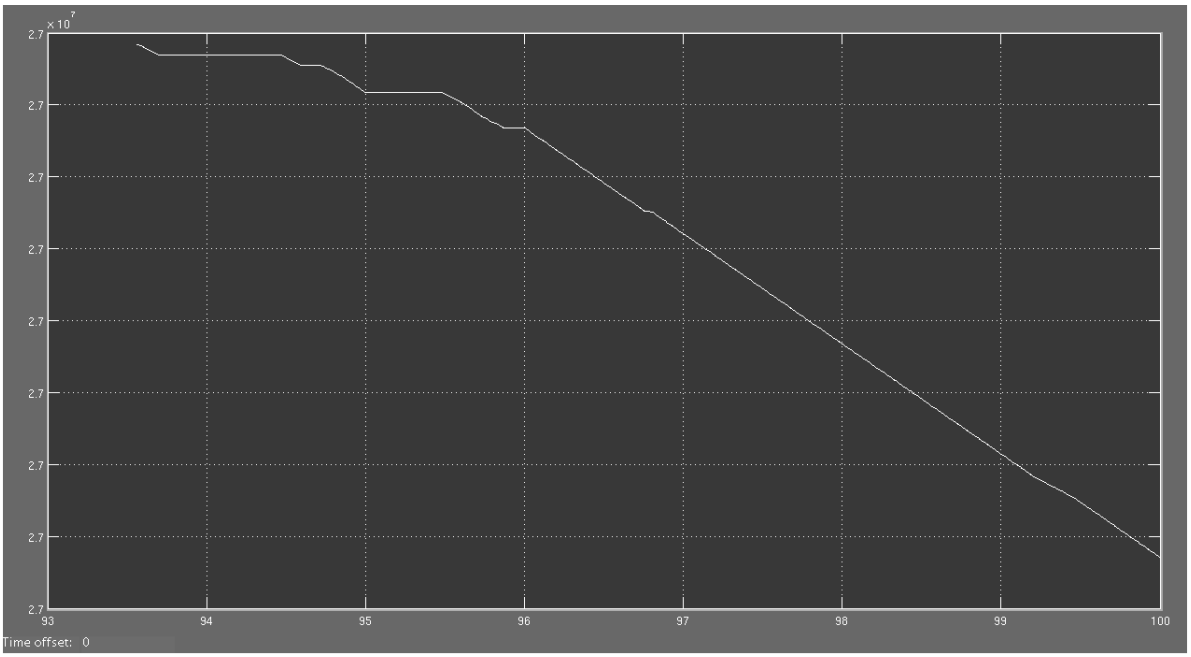

Figura F.2: Gráfico com o consumo de energia de um nodo.

Apêndice G

Resultados da Simulação com 50 Nodos

# Simulação	Energia Consumida (J)				Cobertura da Rede (%)				Tempo de Chegada do Evento (s)				Quant. de Msg. Enviadas			
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
1	0,239952	0,239952	0,435851	0,379909	26	34	100	100	27,00	28	2,00	2,00	275	317	981	981
2	0,239958	0,239958	0,414592	0,364649	6	6	100	100	5,80	6,00	2,00	2,00	203	203	981	981
3	0,239959	0,239958	0,414590	0,364649	14	6	100	100	15,80	6,00	2,00	2,00	219	203	981	981
4	0,239952	0,239952	0,435849	0,379909	26	34	100	100	27,00	28,00	2,00	2,00	275	317	981	981
5	0,239959	0,216924	0,414590	0,364649	14	6	100	100	11,00	6,00	2,00	2,00	214	203	981	981
6	0,222403	0,225976	0,414591	0,364649	12	42	100	100	15,80	26,00	2,00	2,00	216	463	981	981
7	0,230642	0,239956	0,435849	0,379909	16	8	100	100	16,80	6,60	2,00	2,00	231	206	981	981
8	0,221330	0,239952	0,435850	0,379909	10	42	100	100	11,00	16,20	2,00	2,00	215	248	981	981
9	0,225991	0,225988	0,386222	0,344462	20	6	100	100	20,80	6,00	2,00	2,00	227	203	981	981
10	0,239958	0,225988	0,414592	0,364649	6	8	100	100	5,80	6,60	2,00	2,00	203	206	981	981
11	0,240413	0,239952	0,435849	0,379909	10	34	100	100	11,80	28,00	2,00	2,00	211	317	981	981
12	0,239964	0,262483	0,386222	0,344462	6	46	100	100	5,80	21,60	2,00	2,00	203	303	981	981
13	0,239958	0,239958	0,414591	0,364649	6	6	100	100	5,80	6,00	2,00	2,00	203	203	981	981
14	0,257970	0,239954	0,435849	0,379909	12	22	100	100	13,20	18,60	2,00	2,00	224	261	981	981
15	0,254810	0,239963	0,386222	0,344462	10	10	100	100	11,80	7,20	2,00	2,00	210	209	981	981
16	0,222220	0,239957	0,414590	0,364649	6	34	100	100	5,80	21,00	2,00	2,00	203	300	981	981
17	0,239955	0,239952	0,435849	0,379909	26	34	100	100	27,20	28,00	2,00	2,00	258	317	981	981
18	0,235299	0,240197	0,414590	0,364649	16	6	100	100	16,40	6,00	2,00	2,00	222	203	981	981
19	0,221332	0,236215	0,414591	0,364649	22	30	100	100	30,80	21,00	2,00	2,00	264	269	981	981
20	0,225985	0,239958	0,414591	0,364649	10	6	100	100	10,80	6,00	2,00	2,00	208	203	981	981
Min	0,225560	0,228680	0,400957	0,354941	7	6	100	100	7	5	2	2	200	189	981	981
Max	0,246241	0,246640	0,434595	0,378983	21	36	100	100	23	24	2	2	248	326	981	981
Média	0,235901	0,237660	0,417776	0,366962	14	21	100	100	15	15	2	2	224	258	981	981

Figura G.1: Primeira parte do resultado da simulação do cenário com 50 nodos.

# Simulação	Quant. de Msg. Enviadas				Quant. De Msg Recebidas				Quant. De Clones	
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2
1	275	317	981	981	2374	2392	13540	12953	5	5
2	203	203	981	981	2067	2089	12020	11495	1	1
3	219	203	981	981	2052	2079	12020	11495	3	1
4	275	317	981	981	2374	2392	12540	12953	5	5
5	214	203	981	981	2029	2082	12020	11495	2	1
6	216	463	981	981	2086	2353	12020	11495	3	5
7	231	206	981	981	2277	2200	13540	12953	3	1
8	215	248	981	981	2365	2368	13540	12953	2	3
9	227	203	981	981	1715	1791	10120	9688	4	1
10	203	206	981	981	2067	2089	12020	11495	1	1
11	211	317	981	981	2261	2392	13540	12953	1	5
12	203	303	981	981	1809	1899	10120	9688	1	4
13	203	203	981	981	2067	2089	12020	11495	1	1
14	224	261	981	981	2176	2277	13540	12953	2	3
15	210	209	981	981	1795	1828	10120	9688	1	1
16	203	300	981	981	2026	2165	12020	11495	1	4
17	258	317	981	981	2250	2392	13540	12953	5	5
18	222	203	981	981	2156	2014	12020	11495	3	1
19	264	269	981	981	2101	2081	12020	11495	6	3
20	208	203	981	981	2047	2089	12020	11495	2	1
Min	200	189	981	981	1919	1963	11089	10621	1	1
Max	248	326	981	981	2290	2343	13345	12847	4	4
Média	224	258	981	981	2105	2153	12217	11734	3	3

Figura G.2: Segunda parte do resultado da simulação do cenário com 50 nodos.

Apêndice H

Resultados da Simulação com 100 Nodos

# Simulação	Energia Consumida (J)				Cobertura da Rede (%)				Tempo de Chegada do Evento (s)			
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
1	0,239933	0,239931	0,601053	0,512811	25	14	100	100	-	17,6	2,00	2,00
2	0,239926	0,239929	0,636771	0,512811	14	12	100	100	22,60	12,80	2,00	2,00
3	0,243231	0,233051	0,642685	0,541170	7	48	100	100	12,60	46,00	2,00	2,00
4	0,239930	0,241554	0,642685	0,541171	7	5	100	100	12,60	7,80	2,00	2,00
5	0,242164	0,239931	0,605597	0,512811	27	16	100	100	-	16,00	2,00	2,00
6	0,242033	0,239929	0,642685	0,541172	11	12	100	100	21,40	12,80	2,00	2,00
7	0,237628	0,239933	0,557700	0,476800	19	10	100	100	30,80	14,60	2,00	2,00
8	0,239927	0,239928	0,604127	0,511538	12	5	100	100	16,80	12,60	2,00	2,00
9	0,239925	0,239926	0,642685	0,511539	9	13	100	100	17,60	22,60	2,00	2,00
10	0,239932	0,234946	0,605597	0,541171	9	3	100	100	24,00	7,20	2,00	2,00
11	0,239920	0,239921	0,573117	0,489340	22	26	100	100	-	32,60	2,00	2,00
12	0,239932	0,239930	0,603259	0,511515	8	3	100	100	24,00	6,60	2,00	2,00
13	0,242047	0,237622	0,604127	0,511539	11	3	100	100	26,60	7,20	2,00	2,00
14	0,240607	0,239932	0,564106	0,481772	16	4	100	100	24,40	7,80	2,00	2,00
15	0,239928	0,237343	0,576958	0,491193	5	2	100	100	11,40	6,00	2,00	2,00
16	0,241742	0,239425	0,603261	0,510614	7	3	100	100	15,80	6,40	2,00	2,00
17	0,239926	0,239927	0,569943	0,486339	3	3	100	100	6,40	6,60	2,00	2,00
18	0,237429	0,234562	0,576087	0,491498	14	9	100	100	12,60	13,60	2,00	2,00
19	0,239936	0,239939	0,588635	0,498317	3	3	100	100	6,40	6,60		2,00
20	0,239929	0,237040	0,577242	0,492043	15	40	100	100	36,40	28,80	2,00	2,00
Min	0,238899	0,236491	0,572635	0,487958	5	-1	100	100	11	4	2	2
Max	0,241703	0,240979	0,629197	0,528758	19	24	100	100	27	25	2	2
Média	0,240301	0,238735	0,600916	0,508358	12	12	100	100	19	15	2	2

Figura H.1: Primeira parte do resultado da simulação do cenário com 100 nodos.

# Simulação	ia Consum	Quant. de Msg. Enviadas				Quant. De Msg Recebidas				Quant. De Clones	
	S1	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2
1	0,239933	587	488	1981	1981	6683	6921	50440	50237	14	3
2	0,239926	445	439	1981	1981	7326	7085	55860	50237	4	2
3	0,243231	420	797	1981	1981	6997	7438	55860	55747	2	9
4	0,239930	420	412	1981	1981	6997	6712	55860	55747	2	1
5	0,242164	603	490	1981	1981	7200	6864	50440	50237	13	3
6	0,242033	439	439	1981	1981	7082	7085	55860	55747	4	2
7	0,237628	499	430	1981	1981	6480	6673	44000	43781	5	2
8	0,239927	451	428	1981	1981	7285	7149	50160	49925	2	2
9	0,239925	431	445	1981	1981	7525	7326	55860	49925	3	4
10	0,239932	447	409	1981	1981	6799	6581	50440	55747	4	1
11	0,239920	559	609	1981	1981	7997	7862	45980	45919	12	6
12	0,239932	447	406	1981	1981	6799	7009	50220	70299	4	1
13	0,242047	489	409	1981	1981	7677	7277	50160	49925	5	1
14	0,240607	463	412	1981	1981	6554	6726	44700	44510	4	1
15	0,239928	411	403	1981	1981	7195	7402	46740	46512	2	1
16	0,241742	416	406	1981	1981	6953	6969	44700	50090	3	1
17	0,239926	406	406	1981	1981	7428	7299	45720	45554	1	1
18	0,237429	441	445	1981	1981	6683	6655	46540	46471	2	2
19	0,239936	406	406	1981	1981	6332	6114	48400	47897	1	1
20	0,239929	556	622	1981	1981	7100	7175	46580	46463	7	4
Min	0,238899	405	365	1981	1981	6636	6632	45531	44544	1	0
Max	0,241703	529	565	1981	1981	7473	7400	53921	56553	9	4
Média	0,240301	467	465	1981	1981	7055	7016	49726	50549	5	2

Figura H.2: Segunda parte do resultado da simulação do cenário com 100 nodos.