COPPE/UFRJ

# UM ESQUEMA EM DOIS NÍVEIS PARA SUPORTE À MOBILIDADE EM REDES DE SENSORES SEM FIO

Filippe Coury Jabour Neto

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Aloysio de Castro Pinto Pedroza

Rio de Janeiro

Junho de 2009

UM ESQUEMA EM DOIS NÍVEIS PARA SUPORTE À MOBILIDADE EM REDES DE SENSORES SEM FIO

Filippe Coury Jabour Neto

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

---

Prof. Aloysio de Castro Pinto Pedroza, Dr.

---

Prof. Antonio Carneiro de Mesquita Filho, Dr. d'État

---

Prof. Antônio Jorge Gomes Abelém, D.Sc.

---

Prof. Célio Vinicius Neves de Albuquerque, Ph.D.

---

Prof. José Ferreira de Rezende, Dr.

---

Prof. Marcelo Gonçalves Rubinstein, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2009

Jabour Neto, Filippe Coury

Um esquema em dois níveis para suporte à mobilidade em redes de sensores sem fio/Filippe Coury Jabour Neto. – Rio de Janeiro: UFRJ/COPPE, 2009.

XVIII, 162 p.: il.; 29, 7cm.

Orientador: Aloysio de Castro Pinto Pedroza

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2009.

Referências Bibliográficas: p. 116 – 127.

1. Redes sem fio. 2. Redes de sensores. 3. Mobilidade em redes. 4. Agentes móveis. I. Pedroza, Aloysio de Castro Pinto. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

*À minha esposa Eugênia.*

*Aos meus filhos Teddy, Letícia,*

*Fábio e Débora.*

*Aos meus pais William e Denize.*

*Aos meus irmãos Bel, Lu e Zia.*

# Agradecimentos

Ao meu orientador Aloysio, pessoa de grande capacidade técnica e intelectual, e de postura ética e espiritual além do nosso tempo.

À Eugênia (Di), pela ajuda, incentivo e por traçar sempre as melhores estratégias!

Aos meus pais. Meus primeiros professores. Aos meus filhos. Meus atuais professores.

Ao meu sogro Giancoli e à minha sogra Edianez, pelo apoio e ajuda nesta e em todas as empreitadas.

A todos os meus professores e a todos aqueles que um dia se dispuseram a me ensinar alguma coisa.

À estrutura de ensino superior público do Brasil, em especial à UFJF, UFF e UFRJ, que me ofertaram a graduação, mestrado e doutorado.

À equipe do GTA, em especial ao Rezende, Leão e Otto.

Ao pessoal da secretaria, em especial à Daniele, Maurício e Solange.

Ao CEFET-MG.

A todos que contribuíram no desenvolvimento dos compiladores, sistema operacional, mecanismos de busca e demais programas que utilizei ao longo deste trabalho, todos livres.

A quem plantou, torrou, transportou, ensacou e vendeu o café que consumi durante minhas pesquisas. Eu mesmo cuava...

Novos conhecimentos e novas emoções sempre me deslumbraram. Os primeiros números, as primeiras palavras. As primeiras demostrações de teoremas, a geometria, a física. A cesta do Marcel no mundial de basquete de 1978. O lá menor com nona, a escala pentatônica menor, a de *blues*, a menor harmônica. Mais tarde, as equações de Maxwell, as integrais e transformadas, as séries, a física moderna sob a ótica das "Feynman Lectures on Physics". As vitórias do Botafogo ★. As músicas

do Queen, os solos do Blackmore, os Beatles, o Pink Floyd ...

São sempre fartas as inspirações e motivações. A todos que me inspiraram, motivaram e desafiaram, sou muito grato.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

# UM ESQUEMA EM DOIS NÍVEIS PARA SUPORTE À MOBILIDADE EM REDES DE SENSORES SEM FIO

Filippe Coury Jabour Neto

Junho/2009

Orientador: Aloysio de Castro Pinto Pedroza

Programa: Engenharia Elétrica

Este trabalho propõe um esquema em dois níveis para o suporte à mobilidade em redes de sensores sem fio. Baseia-se em interações locais entre sensores aleatoriamente móveis e em interações globais entre agentes (*software*) que se movem intencionalmente pela rede. Pretende-se tornar viáveis a coleta e entrega de dados em redes de sensores onde os recursos são escassos e onde pode haver alta mobilidade involuntária do evento de interesse, dos nós fonte (coletores), dos sensores intermediários e dos nós sorvedouros (destino dos dados). É proposto o protocolo 3M, um novo protocolo de enlace voltado à descoberta e manutenção de vizinhos. Adicionalmente, o 3M executa a alocação dinâmica de canais de freqüência. São propostos e analisados três algoritmos de decisão dos agentes móveis, destinados ao cumprimento das tarefas globais da rede. Foi desenvolvido um simulador de eventos discretos *multithread* que implementa todas as funcionalidades dos dois níveis da presente proposta. Sua construção, funcionamento e validação são apresentados nesta tese. Por fim, é proposto um novo esquema de autenticação de agentes móveis baseado em curvas elípticas. Este esquema se destina a proteger os sensores dos agentes desautorizados ou maliciosos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

A TWO-TIER APPROACH FOR MOBILITY SUPPORT IN WIRELESS SENSOR NETWORKS

Filippe Coury Jabour Neto

June/2009

Advisor: Aloysio de Castro Pinto Pedroza

Department: Electrical Engineering

This work proposes a two-tier approach for mobility support in wireless sensor networks. It is based on local interactions among randomly mobile sensors and on global interactions among software agents that move by the network. We intend to perform data collection and transport in sensor networks which lack resources and where sources, sinks and phenomena may have high degrees of involuntary mobility. It is proposed the 3M protocol, a new link layer protocol for neighbors discovery and maintenance. Additionally, 3M executes dynamic frequency channel allocation. Three algorithms for mobile agents decision are proposed and tested. They are intended to perform the global tasks. We developed a discrete event multithread simulator that implements all the features of the two levels of this proposal. Its construction, operation and validation are presented in this thesis. Finally, is presented a new authentication scheme for mobile agents, based on elliptic curves. This scheme is intended to protect the sensors from unauthorized or malicious agents.

# Sumário

**Lista de Figuras** **xii**

**Lista de Tabelas** **xiv**

**Lista de Abreviaturas** **xv**

**1 Introdução** **1**

1.1 Motivação . . . . . 2

1.2 Um esquema em dois níveis para suporte à mobilidade em RSSF . . . 4

1.3 Principais contribuições deste trabalho . . . . . 5

1.4 Estrutura do Documento . . . . . 5

**2 Redes de sensores sem fio e mobilidade** **7**

2.1 Categorias de mobilidade . . . . . 7

2.2 Formas de sensoriamento . . . . . 9

2.3 Localização e roteamento em RSSF . . . . . 10

2.3.1 Técnicas de localização . . . . . 10

2.3.2 Roteamento e entrega de mensagens . . . . . 14

2.4 Considerações finais . . . . . 19

**3 NIL - O Nível de Interações Locais** **20**

3.1 Coleta de dados . . . . . 20

3.2 Algoritmo de localização . . . . . 23

3.3 Previsão de localização do nó . . . . . 23

3.3.1 Trajetória estritamente crescente . . . . . 24

3.3.2 Trajetória do tipo *Manhattan* . . . . . 25

3.4 Manutenção do conjunto de vizinhos: revisão bibliográfica . . . . . 27

3.4.1 *Address Resolution Protocol* - ARP . . . . . 28
3.4.2 O Padrão IEEE 802.15.4 . . . . . 28
3.4.3 Protocolos SMACS e EAR . . . . . 30
3.4.4 O Padrão IEEE 802.11 . . . . . 34
3.4.5 O Padrão IEEE 802.15.1 - *Bluetooth* . . . . . 35
3.4.6 Disco - Descoberta assíncrona de vizinhos . . . . . 36
3.4.7 ENDP - Descoberta de vizinhos eficiente em RSSF móveis . . 40
3.4.8 MS-MAC - MAC para RSSF ciente da mobilidade . . . . . 40
3.4.9 Outras técnicas relacionadas ao controle de vizinhos . . . . . 42
3.5 3M: *Multichannel Mobile MAC* . . . . . 45
3.5.1 O canal de controle . . . . . 49
3.5.2 Mensagem de anúncio de presença . . . . . 49
3.5.3 O algoritmo de descoberta e manutenção de vizinhos . . . . . 52
3.5.4 Sintonia do canal de trabalho . . . . . 54
3.5.5 Avaliação de desempenho . . . . . 57
3.6 Considerações finais . . . . . 68

**4 NIG - O Nível de Interações Globais 69**
4.1 Introdução aos agentes móveis . . . . . 70
4.2 Aspectos de segurança . . . . . 73
4.2.1 Segurança das plataformas de agentes móveis . . . . . 74
4.2.2 Criptografia e assinatura digital . . . . . 74
4.2.3 Algoritmo de assinatura digital baseado em curvas elípticas (ECDSA) . . . . . 80
4.2.4 Esquema de segurança proposto . . . . . 82
4.2.5 Considerações finais da seção . . . . . 84
4.3 Aplicação dos agentes móveis ao NIG . . . . . 84
4.4 Uma visão geral dos níveis de interações . . . . . 87
4.5 Algoritmos de decisão dos agentes móveis . . . . . 87
4.5.1 Introdução . . . . . 87
4.5.2 Algoritmo somente-distância . . . . . 90
4.5.3 Algoritmo somente-coeficiente-angular . . . . . 91
4.5.4 Algoritmo distância-e-coeficiente-angular . . . . . 93

4.5.5 Considerações finais da seção . . . . . 93
4.6 Análise dos algoritmos . . . . . 94
4.6.1 O impacto da velocidade dos nós . . . . . 94
4.6.2 Análise do alcance de comunicação . . . . . 97
4.6.3 Passo do relógio: esforço computacional . . . . . 100
4.6.4 Considerações finais da seção . . . . . 105
4.7 Além das RSSF . . . . . 106

**5 Conclusão** **110**
5.1 Trabalhos futuros . . . . . 113
5.1.1 Parâmetros $\alpha$ e $\gamma$ nas previsões de localização do nó . . . . . 113
5.1.2 Adaptabilidade dos agentes móveis . . . . . 113
5.1.3 Evoluções no simulador . . . . . 114
5.1.4 Oscilações nas migrações . . . . . 115
5.1.5 Implementação real . . . . . 115
5.1.6 Análise do erro de localização . . . . . 115

**Referências Bibliográficas** **116**

**A O simulador desenvolvido** **128**
A.1 Outros simuladores . . . . . 128
A.2 Descrição do simulador . . . . . 130
A.2.1 O Núcleo do simulador . . . . . 131
A.2.2 A Interface de entrada de dados do simulador . . . . . 134
A.2.3 A Interface de saída de dados do simulador . . . . . 144
A.2.4 Os níveis de enlace e físico do simulador . . . . . 151
A.3 Validação e testes de sanidade . . . . . 152
A.3.1 Validação do relógio do simulador . . . . . 153
A.3.2 Validação dos algoritmos de decisão dos agentes móveis . . . . 154
A.3.3 Validação do mapeamento *setdest* X simulador . . . . . 157
A.4 Desempenho . . . . . 158
A.5 Considerações finais . . . . . 160

# Lista de Figuras


3.1 Predição de localização do nó - trajetória com menor variação . . . . 25
3.2 Previsão de localização para a trajetória *Manhattan* . . . . . 26
3.3 Rede hierarquizada em *clusters* . . . . . 30
3.4 Exemplo simplificado da descoberta de vizinhos pelo protocolo Disco 39
3.5 Mensagem de anúncio de presença . . . . . 50
3.6 Exemplos de vizinhos reais e conhecidos . . . . . 58
3.7 Falsos positivos do protocolo 3M para mobilidade com pausas longas 63
3.8 VRC e VCNR com variação da densidade de vizinhos . . . . . 64
3.9 Falsos positivos do protocolo 3M. Mobilidade praticamente sem pausas 65
3.10 VRC para diversos intervalos de *beacon* . . . . . 65
3.11 Percentual de colisões de quadros para diversos intervalos de *beacon* . 66
3.12 VR desconhecidos e VCNR para TTL menor que *beaconInterval* . . . 67

4.1 Criptografia assimétrica . . . . . 75
4.2 Assinatura digital . . . . . 76
4.3 Visão geral da proposta . . . . . 88
4.4 Os sensores e seus vizinhos no NIL . . . . . 88
4.5 Esquema de funcionamento do agente móvel . . . . . 89
4.6 Regiões destino e de retorno e a execução de um agente móvel . . . . 90
4.7 Tipos de movimento de um nó . . . . . 92
4.8 Migração do agente móvel com base somente no coeficiente angular . 93
4.9 Migração equivocada sem possibilidade de recuperação . . . . . 96
4.10 Impacto da velocidade (m/s) – Nº médio de sucessos por agente móvel 97
4.11 Impacto da velocidade (m/s) – Nº médio de migrações por agente móvel 98
4.12 Alcance da comunicação – Nº médio de sucessos por agente móvel . . 98

4.13 Alcance da comunicação – Nº médio de migrações por agente móvel . 99
4.14 Alcance da comunicação – Nº médio de migrações para cada sucesso . 100
4.15 Granularidade de tempo – Nº médio de sucessos por agente móvel . . 101
4.16 Granularidade de tempo – Nº médio de migrações por agente móvel . 102
4.17 Anomalia de um agente móvel migrando sucessivamente entre 2 nós . 103
4.18 Oscilação do agente móvel sem tratamento e com amortecimento . . . 107
4.19 Granularidade de tempo – Nº médio de migrações para cada sucesso . 108
4.20 Nº médio de sucessos por agente móvel . . . . . 108
4.21 Nº médio de migrações por agente móvel . . . . . 109

A.1 Funcionamento do relógio do simulador . . . . . 134
A.2 Detalhe de um arquivo de entrada do simulador . . . . . 137
A.3 Mapeamento do mobilidade do arquivo de entrada para o simulador . 139
A.4 Janela de configuração dos nós . . . . . 141
A.5 Janela de configuração dos agentes móveis . . . . . 142
A.6 Janela de configuração dos *logs* de saída . . . . . 143
A.7 Detalhe de um arquivo de saída do tipo Node_Nº.log . . . . . 145
A.8 Detalhe de um arquivo de saída do tipo MobileAgent_Nº.shortLog . . 146
A.9 Detalhe de um arquivo de saída do tipo MobileAgent_Nº.log . . . . . 147
A.10 Detalhe de um arquivo de saída do tipo MobileAgent_Nº.dist . . . . . 148
A.11 Distância do agente móvel à região alvo . . . . . 149
A.12 Detalhe de um arquivo de saída do tipo MobileAgent_Nº.origin / .target 149
A.13 Detalhe do arquivo com os dados gerais da simulação . . . . . 150
A.14 Resumo da simulação para um tipo de algoritmo de decisão . . . . . 150
A.15 Saída analítica do simulador, em vídeo . . . . . 151
A.16 Fuga radial de 16 nós . . . . . 155
A.17 Trajetória do agente móvel 0 . . . . . 156
A.18 Etapas de mapeamento da mobilidade para o padrão do simulador . . 158
A.19 Comparação do arquivo de entrada (*setdest*) com o *log* de saída . . . 162

# Lista de Tabelas


3.1 Estatística das previsões de localização do nó (metros) . . . . . 25
3.2 Alguns protocolos de enlace e de descoberta de vizinhos para RSSF . 44
3.3 Mensagens do canal de controle do protocolo 3M . . . . . 50

4.1 Tamanho dos parâmetros e do par de chaves . . . . . 79
4.2 Funcionamento do protocolo de autenticação proposto . . . . . 84
4.3 Dados preliminares de duas propostas de solução para a oscilação nas migrações dos agentes (valores médios) . . . . . 105
4.4 Dados preliminares de duas propostas de solução para a oscilação nas migrações dos agentes (valores arredondados) . . . . . 105

A.1 Acesso ao núcleo do simulador . . . . . 161

# Lista de Abreviaturas

| | |
|---|---|
| API | Application Programming Interface, p. 71 |
| AP | Access Point, p. 34 |
| ARP | Address Resolution Protocol, p. 28 |
| BI | Broadcast Invite, p. 33 |
| BSSID | Basic Service Set Identification, p. 35 |
| BSS | Basic Service Set, p. 34 |
| CDMA | Code Division Multiple Access, p. 47 |
| CRC | Cyclic Redundancy Check, p. 51 |
| CSMA/CA | Carrier Sense Multiple Access with Collision Avoidance, p. 42 |
| CTS | Clear To Send, p. 41 |
| DSA | Digital Signature Algorithm, p. 77 |
| DSSS | Direct-sequence spread spectrum, p. 47 |
| EAR | Eavesdrop-And-Register, p. 30 |
| ECC | Criptografia com Curvas Elípticas, p. 74 |
| ECDSA | Elliptic Curve Digital Signature Algorithm, p. 6 |
| ENDP | Energy-efficient neighbor discovery protocol for mobile wireless sensor networks, p. 40 |
| FAMA | Floor Acquisition Multiple Access, p. 42 |

| | |
|---|---|
| FCS | Frame Check Sequence, p. 51 |
| FFD | Full-function Device, p. 28 |
| FHSS | Frequency-hopping Spread Spectrum, p. 47 |
| FL | Frame Length, p. 51 |
| GPS | Global Positioning System, p. 10 |
| HDR | High Data Rate, p. 15 |
| IBSS | Independent Basic Service Set, p. 35 |
| IDE | Integrated Development Environment, p. 131 |
| IEEE | Institute of Electrical and Electronics Engineers, p. 27 |
| IMEP | Internet MANET Encapsulation Protocol, p. 42 |
| IP | Internet Protocol, p. 27 |
| ISM | Industrial, Scientific and Medical, p. 56 |
| ISO | International Organization for Standardization, p. 27 |
| JVM | Java Virtual Machine, p. 160 |
| LDR | Low Data Rate, p. 15 |
| MAC | Media Access Control, p. 27 |
| MANET | Mobile ad hoc Network, p. 42 |
| MD | Mobile Disconnect, p. 33 |
| MFR | MAC Footer, p. 51 |
| MHR | MAC Header, p. 51 |
| MIMO | Multiple-Input and Multiple-Output, p. 55 |
| MI | Mobile Invite, p. 33 |
| MR | Mobile Response, p. 33 |

| | |
|---|---|
| MS-MAC | Mobility-aware Sensor MAC, p. 41 |
| MT | Movement Type, p. 51 |
| MaRS | Maryland Routing Simulator, p. 129 |
| NIG | Nível de Interações Globais, p. 69 |
| NIL | Nível de Interações Locais, p. 20 |
| NSC | nonsynchronous scheduled communication, p. 31 |
| OSI | Open Systems Interconnection, p. 27 |
| PAN | Personal Area Network, p. 29 |
| PARSEC | Parallel Simulation Environment for Complex Systems, p. 130 |
| PHR | PHY Header, p. 51 |
| RFD | Reduced-function Device, p. 28 |
| RSSF | Redes de Sensores Sem Fio, p. 1 |
| RTS | Request To Send, p. 41 |
| SFD | Start-of-frame Delimiter, p. 51 |
| SHR | Synchronization Header, p. 51 |
| SMACS | Self-Organizing Medium Access Control for Sensor Networks, p. 30 |
| SNR | Signal-to-noise ratio, p. 33 |
| SP | Start of the Packet, p. 51 |
| TDMA | Time Division Multiple Access, p. 31 |
| TDM | Time-Division Multiplexing, p. 36 |
| TTL | Time to Live, p. 15 |
| URL | Uniform Resource Locator, p. 27 |

| | |
|---|---|
| VANET | Vehicular Ad hoc NETwork, p. 106 |
| VCNR | Vizinhos Reais Não Conhecidos, p. 59 |
| VC | Vizinhos Conhecidos, p. 59 |
| VNR | Vizinhos Não Reais, p. 59 |
| VRC | Vizinhos Reais Conhecidos, p. 59 |
| VRD | Vizinhos Reais Desconhecidos, p. 59 |
| VR | Vizinhos Reais, p. 59 |
| WPAN | Wireless Personal Area Network, p. 29 |

# Capítulo 1

# Introdução

Recentes avanços tecnológicos nas áreas de microprocessadores, comunicação sem fio e micro sistemas eletro-mecânicos têm possibilitado a utilização de redes de sensores sem fio (RSSF) [1]. Os sensores são dispositivos de dimensões reduzidas, com restrições computacionais e de comunicação, além de serem alimentados por baterias. Devido às suas dimensões e à impossibilidade de substituição da bateria, o recurso energético é extremamente escasso.

Os sensores têm a capacidade de efetuar medições no ambiente próximo a eles através de uma ou mais funções de sensoriamento existentes no dispositivo. Apesar das reduzidas capacidades citadas, ao se combinarem em grandes quantidades (centenas ou milhares), os sensores podem formar grandes redes e desempenhar importantes tarefas de aquisição de dados. Uma vez coletados, estes dados são transmitidos aos limites da rede, os chamados nós sorvedouros. Estes podem enviar os dados para fora da rede para que possam ser processados. As tarefas de sensoriamento podem se dar nos mais inóspitos e distantes ambientes.

Inúmeras aplicações emergem do uso das RSSF. Alguns exemplos são o monitoramento de *habitats*, pesquisas climáticas, acompanhamento médico, monitoramento de estruturas [2]; monitoramento de ambientes de difícil acesso como o fundo do oceano, vizinhanças de atividades vulcânicas, territórios inimigos, áreas de desastres e campos de atividade nuclear [3].

As redes formadas pelos sensores funcionam no modo *ad hoc*. Assim, os nós funcionam como roteadores, encaminhando os pacotes por múltiplos saltos até o destino final [4]. Este destino pode ser um nó sorvedouro ou outro sensor da rede.

Tipicamente, os sensores enviam os dados coletados aos nós sorvedouros, que são os pontos de saída da RSSF. Deste modo, em geral, é necessário que a rede utilize um protocolo de roteamento. Esses protocolos são definidos como sendo um conjunto de regras pré-estabelecidas, cujo objetivo é construir e gerenciar rotas para a realização do encaminhamento de dados entre os nós da rede [5].

As redes de sensores podem ser classificadas quanto à mobilidade em estáticas ou dinâmicas. Nas estáticas, não há qualquer tipo de mobilidade. Nas dinâmicas, pode-se ter mobilidade dos sensores, do observador (sorvedouro) ou do fenômeno, ou ainda de dois ou três destes ao mesmo tempo [6]. Uma aplicação freqüentemente encontrada, relacionada à mobilidade, é o rastreamento da movimentação de um agente externo, como um indivíduo ou veículo. A rede de sensores, móveis ou estáticos, através de coletas sucessivas, identifica a presença do objeto e estima a sua localização atual e futura.

Esta tese apresenta um esquema em dois níveis que possibilita a coleta e transporte de dados em RSSF com mobilidade aleatória dos nós sensores.

## 1.1 Motivação

A mobilidade relativa entre os nós, o sorvedouro e o fenômeno tem conseqüências importantes no funcionamento da RSSF. Diferente do que ocorre em redes *ad hoc* tradicionais, onde os nós móveis são dispositivos utilizados por pessoas, nas RSSF os sensores não estão interessados no fenômeno e se movem independentemente deste, levados por agentes externos como marés, correntes eólicas ou fluviais ou pela movimentação de seus hospedeiros (pessoas, veículos, etc). Neste caso, a RSSF deve manter o monitoramento do evento de interesse do observador, independente desta mobilidade [6]. Por outro lado, os protocolos de roteamento para redes *ad hoc* tradicionais consideram as restrições de banda e energia em patamares compatíveis com dispositivos de computação móvel tais como computadores portáteis (*notebooks*) ou computadores de mão (*palm tops*). Como já mencionado, as restrições computacionais e energéticas dos sensores são muito mais rigorosas. Deste modo, os mecanismos tradicionais de descoberta de rotas e recuperação daquelas perdidas devido à mobilidade consomem recursos de uma forma que pode ser excessivamente custosa para

os escassos recursos dos sensores.

A motivação para o estudo de mobilidade em redes de sensores está nos seguintes cenários: sensores usados na análise de movimento de tornados [6]; sensores conectados a automóveis circulando em metrópoles para estudar as condições do trânsito e planejamento de rotas [6]; sensores flutuando em correntes de rios [7]; projeto da NASA para exploração de Marte, com sensores movidos por ventos [8]; sensores introduzidos no corpo humano para monitorar condições físicas [9].

Existem, ainda, os cenários onde entidades são dotadas de mobilidade controlada, destinada ao cumprimento de uma determinada tarefa. Entretanto, se considerarmos uma tarefa secundária, diversa da principal, a mobilidade das primeiras entidades se torna aleatória e imprevisível para os atores envolvidos na segunda. Alguns exemplos são equipes de salvamento, soldados ou automóveis equipados com sensores. Além da atividade principal, a rede formada pode ser utilizada, de forma oportunista, para outras tarefas de sensoriamento. Assim, as técnicas de suporte à mobilidade apresentadas neste trabalho também podem ser utilizadas sobre estes ambientes.

A seguir um exemplo.

Dumiak [10], apresenta uma matéria sobre os "robocópteros". Trata-se de robôs autônomos voadores com capacidade de trabalho cooperativo. As aplicações sugeridas são auxílio a equipes de combate a incêndios e salvamento em inundações, acompanhamento ou localização de pessoas ou veículos e cobertura de eventos esportivos. Os robôs podem ainda ser usados para depositar os micro-sensores em uma área de interesse. Sendo a mobilidade dos robôs voltada à execução de tarefas específicas, esta mobilidade se torna aleatória para outra tarefa a ser executada sobre a mesma plataforma. Nota-se, então, que as mesmas técnicas de sensoriamento usadas em uma rede com nós dotados de mobilidade não controlada podem ser aplicadas neste caso, para executar uma tarefa secundária, diferente da que os robôs estejam executando.

Diante dos cenários expostos e baseado nas pesquisas realizadas, decidiu-se estudar as RSSF com nós móveis. O interesse está especialmente voltado para as redes com mobilidade involuntária e intensa dos nós sensores. A mobilidade involuntária se refere ao fato do sensor não possuir nenhum controle sobre seu movimento. Mobilidade intensa se caracteriza por movimentação constante ou muito freqüente, com

velocidades relativamente altas. As simulações feitas ao longo deste trabalho aplicaram deslocamentos da ordem de 10 a 90 $m/s$. O evento de interesse, sua localização e a região para onde os dados devem ser trazidos também podem ser móveis.

Conforme está descrito no Capítulo 2, existem diversas propostas que relacionam alguns níveis de mobilidade às RSSF. Em algumas, tem-se a mobilidade do nó sorvedouro [11, 12, 13], em outras, do fenômeno de interesse a ser monitorado [14]. Em um terceiro grupo, existe a mobilidade reduzida de alguns nós da rede [15, 16, 17, 18]. Entretanto, até este momento, não identificamos nenhuma que atenda de forma ampla à situação de alto grau de mobilidade aleatória, que envolva todas as entidades da rede: nós fontes, nós intermediários, nós sorvedouros e fenômeno. Manter o sensoriamento viável mesmo nestas condições é o objetivo deste trabalho.

## 1.2 Um esquema em dois níveis para suporte à mobilidade em RSSF

A presente proposta aplica técnicas de localização e coordenação aos sensores próximos entre si. Através de interações locais, busca-se produzir um conhecimento da localização dos sensores e dos membros do grupo local (vizinhos), além da alocação de canais exclusivos de comunicação e de previsões sobre o deslocamento futuro dos nós. Este nível atua através de troca de mensagens e é denominado Nível de Interações Locais (NIL), apresentado no Capítulo 3.

Em um nível superior, agentes móveis[1] utilizarão técnicas de mobilidade controlada, geralmente presentes em robôs autônomos. Deste modo, estes agentes executarão tarefas globais, transportando as requisições do sorvedouro à fonte e retornando com os dados da fonte ao sorvedouro.

As técnicas de mobilidade autônoma aplicadas a robôs necessitam de dados ambientais para parametrizar os algoritmos de localização, possibilitar uma cobertura otimizada de áreas e prover orientação a entidades externas. Os robôs se movem fisicamente no espaço contínuo. Na presente proposta, os agentes móveis se movem

[1]Em ciência da computação, um agente móvel é a composição de um programa de computador e dados que é capaz de migrar (mover-se) de um computador a outro de forma autônoma e continuar sua execução no computador de destino

no espaço discreto, sobre um substrato formado por sensores móveis.

De um modo geral, as requisições feitas a uma RSSF contêm as coordenadas da região onde o fenômeno ou evento deva ser sensoriado. Assim, nesta proposta, os agentes móveis executam suas tarefas tendo como parâmetro uma região alvo de destino e uma região de retorno. Em uma generalização, quando se deseja saber se um dado evento está ocorrendo em qualquer região coberta pela rede, basta injetar um número suficiente de agentes, endereçados aos diversos quadrantes (sub-regiões) da rede, para cumprir tal tarefa.

## 1.3 Principais contribuições deste trabalho

As principais contribuições desta tese são:

- Um esquema de baixa complexidade para a predição da localização e da direção do movimento dos nós;
- A proposta de um novo protocolo de enlace, descoberta e manutenção de vizinhos, apropriado às RSSF com alto grau de mobilidade aleatória dos nós;
- Um modelo de coleta e transporte de dados baseado em agentes móveis que explora três algoritmos distintos;
- O desenvolvimento de um simulador para os três itens citados anteriormente;
- Um esquema de segurança que protege os sensores contra agentes móveis não autorizados ou maliciosos.

## 1.4 Estrutura do Documento

No Capítulo 2 são apresentados alguns conceitos relativos à arquitetura das RSSF e os tipos de mobilidade que podem ocorrer nestas redes. São discutidas as categorias de mobilidade e formas de sensoriamento. São analisadas, adicionalmente, técnicas de suporte à mobilidade intencional e aleatória e de roteamento de mensagens. Ao longo do capítulo, são mencionadas as ligações dos modelos e técnicas estudados com as propostas da tese.

No Capítulo 3 é discutido o Nível de Interações Locais – NIL. Este nível inclui a coleta de dados, a execução de um algoritmo de localização, a previsão de localização futura e a manutenção de um conjunto de vizinhos. São analisados diversos protocolos de enlace e de descoberta de vizinhos. É proposto o protocolo 3M – *Multichannel Mobile MAC* – que trata da alocação de canais e da descoberta de vizinhos nos cenários de interesse. Por fim, é feita uma análise de desempenho simulada do 3M.

O Capítulo 4 apresenta o Nível de Interações Globais – NIG. Primeiramente, tem-se uma introdução aos agentes móveis e sua aplicação na presente proposta. Em seguida, é apresentada uma visão geral dos dois níveis de interações. O cerne do capítulo apresenta e discute três algoritmos de decisão propostos para os agentes móveis. Por fim, são apresentadas algumas simulações e os resultados são discutidos.

Segurança é um fator crítico em ambientes que suportam agentes móveis. Soluções de segurança não são o objetivo central deste trabalho. Entretanto, a Seção 4.2 discute questões de segurança relacionadas a agentes móveis e às RSSF. A referida seção apresenta os fundamentos da criptografia e assinatura digital baseadas em curvas elípticas. Por fim, apresenta-se um esquema de assinatura digital que utiliza o algoritmo ECDSA (*Elliptic Curve Digital Signature Algorithm*) e que tem por objetivo proteger os sensores (plataforma de agentes móveis) contra possíveis agentes maliciosos.

O Capítulo 5 contém as conclusões e trabalhos futuros.

O simulador desenvolvido é apresentado em detalhes no Anexo A, onde são expostos seu núcleo, as interfaces de entrada e saída e o nível de enlace. Vários testes de validação são mostrados e analisados. Em alguns momentos, a demonstração do funcionamento do simulador é utilizada para discussões adicionais de partes da proposta de suporte à mobilidade de que trata esta tese.

# Capítulo 2

# Redes de sensores sem fio e mobilidade

Este capítulo aborda alguns conceitos relativos à arquitetura de redes de sensores e os tipos de mobilidade que podem ocorrer nestas redes. São analisadas, adicionalmente, técnicas de suporte à mobilidade, auto-organização e descoberta de vizinhos.

Ao final, será feita a primeira apresentação dos cenários a serem tratados e a relação destes com os tópicos do capítulo.

## 2.1 Categorias de mobilidade

Será usada a seguinte terminologia para as entidades envolvidas [6]:

- **Nó sensor:** Dispositivo que mede o fenômeno, envia estas medições a outros sensores e, eventualmente, encaminha (roteia) mensagens de outros nós sensores. Pode ser referenciado apenas como nó ou apenas como sensor. Quando desempenha especificamente o papel de sensoriar (medir) o fenômeno e de injetar estas medidas na rede, é chamado de nó fonte (*source*);
- **Observador:** Usuário final interessado em obter as informações sobre o fenômeno disseminadas na rede. Pode ser referenciado como sorvedouro (*sink*);
- **Fenômeno:** Objeto de interesse a ser medido, disseminado e, eventualmente, analisado e filtrado pelos nós sensores.

Todas estas entidades podem existir em número arbitrário na rede. Tipicamente, existem alguns fenômenos e observadores e muitos nós sensores.

Nas redes estáticas, não há movimento relativo entre sensores, observadores e fenômenos. Um exemplo é um conjunto de sensores espalhados para medições de temperatura.

Havendo mobilidade, ela pode se dar em quaisquer das três entidades citadas anteriormente. Alguns exemplos e comentários:

- **Observador móvel** em relação aos sensores e ao fenômeno. Por exemplo, um avião sobrevoando um campo de sensores lançados em uma área inóspita. Periodicamente, o avião sobrevoa a área coletando as informações desejadas;

- **Sensores móveis** em relação ao observador e uns em relação aos outros. Por exemplo, sensores lançados em um rio, com estações de coleta de dados espaçadamente posicionadas às margens.
  Cabe aqui, ainda, outra subdivisão:

  - Mobilidade controlada: os sensores se movem voluntariamente;
  - Mobilidade involuntária: os sensores são movidos por agentes externos (mobilidade aleatória). Estão fixados a outros objetos móveis como automóveis, animais ou sujeitos ao fluxo de fluidos ou gases.

- **Fenômeno móvel:** Um exemplo típico é o uso de sensores para detecção da presença de animais.

Pode-se ter a situação em que os sensores possuem movimentação relativa entre si e, ainda, a rede como um todo se movimenta. Exemplos seriam sensores movimentados por ventos, correntes marinhas, rios, corrente sangüínea, esgotos, oleodutos, gasodutos, emissários submarinos, automóveis. Neste caso, pode ser que apenas em um instante futuro a rede tenha contato com o observador. Isto não impede que um ou mais sensores assumam o papel de sorvedouro nos instantes posteriores. As aplicações relacionadas aos exemplos dados são diversas: controle de poluição, detecção de rompimentos em dutos, auxílio na engenharia de trânsito, vazamentos de resíduos, mapeamento de correntes marinhas, acompanhamento da diluição de resíduos despejados em mares e rios, etc.

Considerando as aplicações citadas anteriormente e a relativa escassez de propostas que tratem a mobilidade dos nós sensores, dos observadores e dos fenômenos ao mesmo tempo, **o cenário de interesse deste trabalho é aquele onde pode haver mobilidade de qualquer uma das entidades da rede: sensor, fenômeno, observador, de mais de uma delas ou de todas elas.**

Em função da tecnologia atual dos micro-sensores, sendo estes incapazes de locomoção autônoma, **são tratados os cenários com mobilidade involuntária**.

## 2.2 Formas de sensoriamento

Segundo Ruiz *et al.* [1], as RSSF podem ser caracterizadas quanto à coleta de dados em:

1. **Coleta periódica:** Os nós coletam dados sobre os fenômenos em intervalos regulares de tempo. Um exemplo são aplicações que monitoram o canto dos pássaros. Os sensores farão a coleta durante o dia e permanecerão desligados durante a noite.

2. **Coleta contínua:** Os dados devem ser monitorados continuamente. Um exemplo é o monitoramento de incêndios em florestas.

3. **Coleta reativa:** Os nós sensores coletam dados quando ocorrem eventos de interesse. Exemplos são as aplicações que detectam a presença de objetos (pessoas, veículos, animais com certas características, etc.) na área monitorada.

4. **Coleta em tempo real:** Os nós devem coletar o máximo de informações no menor intervalo de tempo possível. Exemplos são aplicações que envolvem salvamentos em desastres.

Em todos os casos citados, pode-se determinar a região de interesse. Em casos onde se deseja seguir a movimentação de um objeto, ou se o fenômeno a ser monitorado estiver em movimento, a área de interesse também se move.

Neste trabalho, a identificação da localização dos nós sensores será necessária e utilizada como parte da solução para tratar a mobilidade das entidades da RSSF. Deste modo, será possível atender a requisições de sensoriamento sobre áreas determinadas. Assim, **esta é terceira característica dos cenários atendidos pela**

**proposta: tratar requisições que especifiquem, além do evento de interesse, uma região geográfica, através de coordenadas.**

## 2.3 Localização e roteamento em RSSF

Esta sessão apresenta um estudo de técnicas de localização e roteamento utilizadas em RSSF, além de modelos de outras áreas que podem ser aplicados à mobilidade em RSSF.

Existem dois pontos de interesse. Um deles diz respeito à localização dos nós. São técnicas que permitem ao nó identificar, com um certo grau de confiabilidade, a sua localização geográfica. O segundo ponto diz respeito à entrega de mensagens em um cenário com mobilidade, ou seja, o roteamento. Deste modo, a sessão se divide, a princípio, em duas partes: localização dos nós e entrega de mensagens. No caso da localização, as técnicas foram divididas em dois subgrupos: aquelas que tratam cenários com mobilidade intencional e as que abordam a mobilidade aleatória.

### 2.3.1 Técnicas de localização

Muitas aplicações de RSSF requerem que os nós tenham ciência de suas respectivas localizações. Em geral, é muito dispendiosa a inclusão de um sistema de localização (receptor GPS) nos sensores [7].

No trabalho de Intanagonwiwate *et al.* [19], são fornecidos exemplos de requisições que podem ser feitas a uma RSSF: "Quantos pedestres você observa na região geográfica X?" ou "Me diga em que direção e sentido aquele veículo na região Y está se movendo". Assim, observa-se a importância do conhecimento da localização do nó.

Quando existe a mobilidade dos nós, a localização tem que ser atualizada à medida que estes se deslocam.

As restrições de recursos dos nós sensores e uma menor precisão nas medições nos sinais de rádio tornam a localização em RSSF mais difícil do que a localização de robôs. Entretanto, a alta densidade de nós em uma rede de sensores pode possibilitar uma cooperação no compartilhamento de informações relativas à localização [7].

As RSSF geralmente operam no modo *ad hoc*. Os métodos de localização adequados às redes *ad hoc* são classificados em:

1. **Localização Centralizada:** Os sensores enviam informações a um ponto central, onde a computação é feita para determinar a localização dos nós.
   A mobilidade inviabiliza esta técnica em função dos altos custos de comunicação e dos atrasos intrínsecos [7]. Adicionalmente, não se pode garantir que a localização computada, ao chegar ao nó móvel, ainda é válida.

2. **Localização Distribuída:** Cada nó determina a sua localização através de poucas interações com seus vizinhos próximos. Pode ser de dois tipos:

   (a) **Baseada no sinal:** Se baseia no instante de chegada (ou variações nele), potência do sinal ou ângulo do sinal recebido. Nas técnicas propostas, os dispositivos necessários tendem a ser sofisticados e caros ou as considerações sobre propagação do sinal tendem a ser pouco realistas. Não se adequam aos escassos recursos de um nó sensor.

   (b) **Independente do sinal:** Se baseia apenas no conteúdo das mensagens. São mais apropriadas às RSSF devido às restrições de *hardware* e energia dos nós sensores.

      i. **Técnicas locais:** Depende de uma alta densidade de sementes (nós que conhecem e divulgam sua localização), de modo que cada nó possa escutar várias sementes. Os nós se comunicam e a localização pode ser estimada.

      ii. **Técnicas de contagem de saltos:** Quando a densidade de sementes é baixa, os nós estimam suas posições com base na localização das sementes e da contagem de saltos até elas. A contagem de saltos é extraída dos anúncios de localização que as sementes propagam pela rede.

Este trabalho considera disponível, em cada nó, a sua localização geográfica. As técnicas baseadas em sementes são as mais apropriadas aos cenários de interesse desta tese. A contagem de saltos é inadequada pois perde o sentido diante da mobilidade dos nós.

A seguir são relacionadas algumas propostas existentes de localização de nós móveis em uma rede. Os cenários para os quais foram propostas estas técnicas de localização são divididos em duas categorias. Em uma delas, a mobilidade é involuntária ou não controlada e no outra a mobilidade é intencional, ou seja, os sensores se movem deliberadamente e de forma controlada (como ocorre com robôs).

**Localização com mobilidade involuntária**

Hu e Evans [7] apresentam um esquema onde a posição inicial do nó não é conhecida. Um conjunto de posições é estimado, escolhido aleatoriamente dentre todas as posições possíveis. Dado um conjunto de possíveis posições atuais do sensor no passo $i$, a estimativa do conjunto de posições no passo $i+1$ é formado por escolhas aleatórias dentro dos círculos com centro em cada estimativa anterior e com raio $R = \pi \times {V_{max}}^2$. O nó não tem nenhuma informação sobre seu movimento, apenas a velocidade máxima possível ($V_{max}$). Assim, há um aumento na incerteza da localização do sensor. Esta é a situação mais geral, onde a movimentação pode se dar em qualquer direção e sentido e a velocidade varia uniformemente entre 0 e $V_{max}$. Se mais restrições ao movimento forem conhecidas, como sentido do movimento ou velocidade aproximada, estes valores poderão ser refletidos na função de probabilidade das previsões futuras.

A partir de localizações reais recebidas de sementes, diretamente ou através de anúncios por parte dos novos vizinhos, o conjunto de previsões é filtrado, de modo que localizações impossíveis são excluídas. A execução repetida do algoritmo mantém um conjunto de possíveis localizações do nó.

Esta técnica baseada em sementes é adequada aos cenários de interesse desta tese.

Barros e Nogueira [20] estudaram e classificaram alguns métodos de localização. A comparação dos intervalos de chegada de sinais eletromagnéticos com sinais de ultrassom, trocados com nós de referência (que conhecem sua localização), possibilita determinar a localização do nó. Os requisitos de *hardware* são complexos. Outra opção é medir o desvanecimento do sinal ou variações apresentadas por ele próximo à fonte emissora. A comparação de diversos atributos do sinal sendo recebido no momento, com padrões conhecidos, permite a estimativa da posição. Em um ambiente

com mobilidade aleatória, será muito difícil gerar estes padrões de referência.

O ângulo de chegada do sinal pode ser utilizado na localização. Entretanto, este método requer antenas muito complexas e não se adequa às RSSF.

As propostas encontradas em [20] não são apropriadas aos cenários de interesse desta tese, em função das restrições de *hardware* dos sensores.

### Localização com mobilidade controlada

Souza [21] propõe utilizar o método de decomposição celular para a discretização do espaço de estados de um robô. O trabalho mostra a implementação da solução baseada no algoritmo A-Estrela modificado, bem como um aplicativo planejador de trajetórias. A trajetória planejada é enviada ao robô móvel por radiofreqüência e este executa o trajeto com perfeição.

Howard *et al.* [22] propuseram um algoritmo de auto-distribuição incremental para RSSF. Estudaram cenários com mobilidade intencional (robôs). É preciso uma linha de mira entre os nós. Busca-se maximizar a cobertura da rede, sem isolar nós da rede. A complexidade do algoritmo proposto é $O(n^2)$. Um nó busca saber sua localização em função de um ou mais pares. O algoritmo é inteiramente distribuído, responde dinamicamente a mudanças ambientais e converge lentamente.

Os autores assumem que os nós possuem um sensor de alcance (*range sensor*); comunicação sem fio; todos se comunicam com uma estação base onde o algoritmo de disposição é executado; o ambiente permanece estático durante a disposição dos robôs. A localização dos nós é considerada sabida. O objetivo é a disposição subótima dos nós, satisfazendo os requisitos impostos.

Como estas, muitas propostas baseadas em mobilidade controlada foram encontradas na literatura. Por outro lado, não foram identificadas pesquisas especificamente voltadas para ambientes com mobilidade aleatória. Este fato motivou a utilização, nesta tese, destas técnicas de mobilidade controlada. As mesmas são adaptadas, através de agentes móveis, aos cenários de mobilidade aleatória.

Em outro estudo, Bachmayer e Leonard [23] apresentam uma técnica que comparam ao movimento de um cardume de peixes. São vários veículos "sentindo" um gradiente, trocando informações e caminhando juntos, coordenadamente, em direção a um destino.

Os veículos, ao seguirem os gradientes, respondem, ao mesmo tempo, a estímulos ambientais e ao comportamento dos seus vizinhos. Em outras palavras, a força que controla o movimento do nó é obtida da soma de aproximações do gradiente local com forças de controle inter-veículos, derivadas de potenciais artificiais considerados entre eles.

Simulações mostraram que veículos isolados não conseguiram atingir o ponto de potencial mínimo do campo (para o percurso descendente do gradiente), enquanto sistemas de múltiplos veículos o fizeram.

Como a presente tese tem foco nas redes com mobilidade involuntária, a técnica pode ser aplicada na coordenação do movimento de agentes móveis, migrando pelos nós móveis, em direção ao destino pretendido. Neste caso, os gradientes utilizados pelos agentes móveis são extraídos de informações obtidas dos diversos nós da rede.

Os estímulos ambientais a que os agentes móveis estão sujeitos são informações do estado dos nós sensores como localização, direção e sentido do movimento.

No Capítulo 4 é apresentado o funcionamento dos agentes móveis e a forma como as técnicas aqui estudadas foram utilizadas.

### 2.3.2 Roteamento e entrega de mensagens

#### Roteamento com mobilidade involuntária

Vahdat e Becker [24] focaram o roteamento em redes não totalmente conexas. A proposta se destina às redes *ad hoc* em geral e não especificamente às RSSF. O roteamento em redes sem fio sempre pressupõe um caminho conectado entre origem e destino. Isto pode não ser verdade em redes partidas (grafos não conexos). Através da mobilidade e de armazenamento (*bufferização*) de pacotes, consegue-se, através desta técnica, levar as mensagens até o destino.

Quando dois nós entram na faixa de comunicação um do outro, eles comparam as mensagens armazenadas e trocam entre si as mensagens novas para cada um deles. As mensagens têm identificadores unívocos gerados a partir de identificadores dos nós. Eventualmente, as mensagens chegarão aos seus destinos. O preço pago é um maior consumo de recursos da rede e uma latência alta, o que é aceitável se considerar que em protocolos de roteamento padrão para redes *ad hoc* a mensagem nem seria entregue ao destinatário.

Analogamente, os agentes móveis utilizados na proposta desta tese desempenham funções de armazenamento e intercâmbio de mensagens.

Choksi *et al.* [11] propõem um trabalho que oferece suporte à mobilidade em redes de sensores *ad hoc* baseadas em difusão. O referido artigo argumenta que os protocolos de difusão estão se mostrando adequados às redes de sensores. Entretanto, o impacto da mobilidade nestas classes de algoritmos permanece pouco explorado. Isto posto, o trabalho aplica técnicas de outras áreas e avalia as melhorias que as mesmas introduzem no tratamento da mobilidade no contexto da difusão de dados.

Foram propostas quatro modificações no *Directed Diffusion* [19]. A primeira reinicia a descoberta de caminhos quando um nó se move (difusão agressiva); a segunda usa um esquema de *handoff*; a terceira introduz *proxies* como âncoras de caminhos; e a quarta busca criar rotas antecipadamente, no sentido do movimento.

- **Difusão agressiva:** Em lugar de aguardar *timeouts* para iniciar a reconstrução de rotas, quando para de receber dados, o nó inicia este processo, supondo ter se movido.

  Em outras palavras, quando o sorvedouro não recebe pacotes de dados por determinado intervalo de tempo, inicia nova inundação do interesse, em lugar de esperar chegar um novo ciclo de dados em taxa reduzida (LDR - *Low Data Rate*) para reforçar (*reinforcement*) a nova rota;

- ***Handoff*:** Semelhante ao usado nas redes celulares. O nó tenta permanecer conectado buscando um novo nó entre seus vizinhos, que o mantenha na árvore de difusão. Quando o nó móvel troca o seu vizinho imediato por outro, o restante da rota, do segundo vizinho em diante, permanece a mesma. A modificação se dará então apenas no último salto de um caminho potencialmente grande.

  O nó sorvedouro mantém uma lista de vizinhos, que são aqueles de quem recebeu dados em taxa reduzida (LDR). Mantém uma lista de ativos, que são aqueles a enviar dados a altas taxas (HDR - *High Data Rate*). A lista de candidatos contém os vizinhos não ativos e é ordenada com base na variação da potência do sinal recebido. A idéia é identificar o afastamento do vizinho ativo através da variação na potência do sinal recebido. Constatado este afas-

tamento, um candidato é escolhido e promovido a ativo, através do envio de reforço (REINF). Neste instante, o sorvedouro passará a receber dados HDR de dois vizinhos. Com a continuidade da movimentação, é provável que o sinal vindo do vizinho ativo antigo continue se enfraquecendo. Quando este sinal baixar de um certo limite, este vizinho é rebaixado de ativo a simples vizinho, através do envio de reforço negativo (*negative reinforcement*).

Esta técnica é adequada quando o sorvedouro se move com pouca velocidade;

- ***Proxy*:** Os *proxies* são nós especiais que funcionam como âncoras dos caminhos. Ao se mover, e parar de receber os dados que vinha recebendo, um nó sorvedouro precisa encontrar uma nova rota apenas até um novo *proxy* e não ao longo de todo o caminho.

  Os sorvedouros enviam mensagens de interesse especiais (encapsulados), com número de saltos (TTL - *Time to Live*) crescente a partir de 2, até encontrar um *proxy*. O *proxy* por sua vez, ao receber um interesse, desencapsula e o envia à procura de uma fonte. Ao receber LDR da fonte, o *proxy* encapsula estes pacotes de dados e envia ao sorvedouro, criando o caminho fonte-sorvedouro. São dois domínios de difusão, ligados pelo *proxy*. A comunicação direta entre sorvedouro e fonte não é permitida. O *proxy* pode implementar agregação de interesses e pode responder imediatamente ao sorvedouro, caso já esteja recebendo dados relativos ao interesse, em LDR.

  Este esquema suporta maiores velocidades de deslocamento dos sorvedouros;

- **Antecipação de rotas:** Adequa-se às classes de aplicações relacionadas à coleta de dados dependentes da localização do nó sorvedouro, por exemplo: "Detectar eventos de áudio 100 m adiante da minha (sorvedouro) posição, no sentido do meu movimento". Nesta situação, quando um nó se move, o conjunto de nós que respondem aos seus interesses muda. A idéia é construir rotas potencialmente úteis no futuro, adiante, no sentido do movimento do nó, iniciando a descoberta (construção da árvore) na área em frente, antes do sorvedouro chegar lá.

Estas técnicas propostas por Choksi *et al.* [11] se aproximam bastante dos objetivos do presente trabalho. Entretanto, a avaliação feita via simulações limitou

os experimentos apenas à mobilidade dos sorvedouros. Todas as fontes e nós intermediários são estacionários. A mobilidade generalizada foi relacionada como trabalho futuro.

As aplicações do trabalho de Choksi *et al.* [11] diferem da presente proposta por não abordarem ambientes com mobilidade generalizada.

Shah *et al.* [12, 13] modelam e analisam uma arquitetura em três camadas para redes de sensores esparsas. A técnica utiliza entidades móveis (*Data MULEs*), externas à rede de sensores, que, ao se deslocarem pela área ocupada pela RSSF, coletam os dados sensoriados e os entregam ao ponto de acesso, por onde os dados podem fluir para fora da rede. O modelo de mobilidade utilizado nas simulações de avaliação do protocolo consideram deslocamento aleatório das *Data MULEs*. Os resultados obtidos mostram que os *buffers* dos sensores e *Data MULEs* são inversamente proporcionais às densidades de pontos de acesso e de *Data MULEs* na área da rede.

Enquanto a proposta dos Data MULEs conta com entidades externas para transportar os dados, a presente tese considera que tal tarefa é executada por agentes móveis.

Bertini *et al.* apresentam a replicação de dados em redes *ad hoc* para sistemas de apoio em situações de desastre [25]. O trabalho apresenta um mecanismo de replicação de dados em redes *ad hoc* que prolonga a disponibilidade dos mesmos e, conseqüentemente, da aplicação. Um nó móvel gera uma réplica de seus dados quando sua energia fica em um nível baixo ou se o nó móvel estiver na iminência de perder a conexão. A função de probabilidade de conectividade foi extraída do número de vizinhos e da potência do sinal recebido de cada um. Assim, a combinação dos estados discretos "nível de energia" e "grau de conectividade" é usada na decisão de replicação. Para a alocação de réplica, a teoria de decisão bayesiana é utilizada, porém com modelagem *fuzzy* dos estados e das observações feitas no sistema. O resultado é um método de alocação de réplicas adequado para um ambiente muito imprevisível como uma rede *ad hoc*.

A observação da conectividade foi feita com base no número de vizinhos e no enfraquecimento do sinal em função da distância destes vizinhos. Nas simulações, constatou-se que a conectividade é uma variável mais imprevisível que a energia.

O cálculo da probabilidade dos estados futuros de um nó é de complexidade sublinear em relação ao número de nós. Existe ainda a possibilidade de se regular o número de estados possíveis de conectividade e o número de níveis de discretização de cada estado. Isto permite o dimensionamento do processamento e da memória utilizada para a implementação da técnica a patamares compatíveis com os recursos típicos de sensores móveis.

A proposta de Bertini *et al.* poderia ser utilizada no presente trabalho. Um agente móvel, ao identificar que seu nó hospedeiro irá se desconectar, poderá optar por migrar, com o intuito de permanecer na região de interesse.

A proposta em estudo [25] utiliza uma tabela de estados e discretização relativa à conectividade dos nós. Esta tabela é gerada em tempo de projeto. Deste modo, não é necessária a construção da tabela dentro da própria rede. Entretanto, isto particulariza a utilização de uma instância da implementação àquele perfil de mobilidade. Esta questão deverá ser considerada e ainda está em aberto no momento. As alternativas são, de fato, considerar esta tabela disponível em tempo de projeto ou identificar o perfil de mobilidade da rede e injetar a tabela apropriada, ou modificações nesta, através dos agentes móveis. Esta pode ser uma tarefa secundária dos agentes, mas não foi explorada nesta tese.

Huang *et al.* [14] apresentam um esquema que permite enviar mensagens adiante, na direção e sentido do movimento de uma entidade externa que se mova ao longo da rede de sensores. Não há movimento dos sensores. Supõe-se disponíveis a localização dos nós e o conjunto de vizinhos. O esquema requer sincronização de relógios. A área de interesse se move ao longo da rede e as mensangens devem antecipar este movimento, estando à frente do movimento da entidade externa. As referidas mensagens podem desencadear a execução de tarefas de sensoriamento na região móvel de interesse. Uma aplicação citada como exemplo pelos autores é a detecção de intrusos em uma dada região e a conseqüente propagação deste evento para regiões à frente da trajetória, avisando aos interessados que o intruso segue naquela direção (um tanque inimigo, por exemplo). Outra aplicação descrita é levantar informações em regiões logo à frente da trajetória de um soldado, de modo a informá-lo sobre as condições desta região.

Aplicações deste tipo podem também ser atendidas pela proposta desta tese, se

baseando na localização dos sensores, na manutenção de um conjunto de vizinhos e em interesses direcionados a regiões conhecidas, possivelmente móveis. Entretanto, a presente proposta é operacional mesmo quando os nós sensores são aleatoriamente móveis, além de não requerer sincronização de relógios.

## 2.4 Considerações finais

Este capítulo apresenta técnicas relacionadas à mobilidade, localização, roteamento e entrega de mensagens em ambientes com mobilidade aleatória ou intencional.

Aspectos pontuais de várias das propostas aqui discutidas fazem parte da presente tese.

Os aspectos adaptados às interações globais, existentes nas tarefas dos agentes móveis, são apresentadas e analisadas no Capítulo 4.

O próximo capítulo aborda as interações locais. Nele são indicados caminhos para a localização dos nós, considerando a adoção de técnicas baseadas em sementes. É proposta uma técnica de previsão da localização e da direção do movimento dos nós. Por fim, é proposto um novo protocolo de descoberta e manutenção de vizinhos para RSSF móveis.

# Capítulo 3

# NIL - O Nível de Interações Locais

As interações locais são responsáveis pela coleta de dados, pela execução de um algoritmo de localização, pela previsão de localização futura e pela manutenção de um conjunto de vizinhos. Nesta proposta, considera-se que todos os nós da rede executam, no mínimo, estas quatro tarefas.

Todas as comunicações relacionadas às interações locais são feitas através de troca de mensagens.

Este capítulo começa descrevendo o mecanismo de coleta de dados utilizado no NIL. Na Seção 3.2, é descrito o algoritmo de localização já existente, requerido para o funcionamento do esquema proposto nesta tese. Na Seção 3.3, descreve-se o mecanismo proposto para a previsão da localização e da direção de deslocamento dos nós, sendo o mesmo avaliado por meio de simulações. A Seção 3.4 contém uma revisão bibliográfica sobre descoberta e manutenção do conjunto de vizinhos. Por fim, a Seção 3.5 apresenta detalhadamente o protocolo de enlace proposto por esta tese.

## 3.1 Coleta de dados

Como visto na Seção 2.2, a coleta de dados pode ser periódica, contínua, reativa ou em tempo real.

Quando os sensores são móveis, existem intervalos de tempo em que nenhuma leitura precisa ser feita, uma vez que o mesmo se encontra fora da região de interesse.

Conforme mencionado no Capítulo 1 e detalhado no Capítulo 4, a tarefa de

transporte dos dados será executada por agentes móveis. Os nós sensores possuem conhecimento de suas respectivas localizações e podem considerá-la como parâmetro para efetuarem ou não as medidas (sensoriamentos). Os agentes móveis possuem um conhecimento global da tarefa de sensoriamento em execução na rede e a relacionam com as coordenadas do seu hospedeiro atual e dos vizinhos deste hospedeiro. Deste modo, a presente proposta tem as seguintes características com relação ao início e fim dos períodos de amostragem de dados pelos sensores:

- **Amostragens sob demanda**: ao chegar à região de interesse, o agente móvel divulga para a lista de vizinhos do hospedeiro atual uma mensagem de início do período de amostragem, contendo um tempo de duração curto, adequado ao período em que permanecerá nas imediações deste grupo. Esta coleta é uma interação local, feita por troca de mensagens;

- **Amostragens contínuas**: os agentes móveis propagam pela rede mensagens de início do período de amostragem, contendo um longo tempo de duração e as coordenadas de interesse.

Em ambos os casos, o término dos períodos de amostragem se dará por temporização, em oposição a mensagens explícitas de fim de amostragem. Isto é mais apropriado, uma vez que a mobilidade pode gerar quebra repentina de conectividade entre os nós da rede, em função do afastamento ultrapassar o alcance do rádio de comunicação.

A amostragem sob demanda se aplica à situação em que o agente móvel tem a tarefa de verificar se um dado evento existe ou ocorreu em uma dada região, devendo retornar a outra região com esta informação. O referido evento pode ser a presença de um objeto (pessoa, animal, veículo, etc.), uma grandeza medida em um certo intervalo (temperatura, concentração de substâncias diluídas, pH [26], etc.). Neste caso, ao chegar à região alvo, o agente móvel divulga para a lista de vizinhos do hospedeiro atual uma mensagem de início do período de amostragem, contendo um tempo de duração curto, adequado ao período em que permanecerá nas imediações deste grupo. Em seguida, coleta os dados lidos e inicia a busca pela região de retorno, para onde deve levar os dados. Dentro de poucos instantes, o período de amostragem se esgotará e os sensores interromperão as leituras.

No caso da amostragem contínua, o interesse da aplicação está em saber se algum evento está ocorrendo em uma dada região ou se ocorreu em algum momento no passado. Não se trata de uma leitura atual, mas da ocorrência ou não do evento em um período longo de tempo. Entenda-se por período longo de amostragem, todo aquele que não se refere à uma medição instantânea. Pode se referir a alguns minutos, horas ou dias, dependendo da aplicação.

Relacionando com os exemplos da amostragem sob demanda, a amostragem contínua estaria interessada em saber se houve, em algum momento, a presença de um dado objeto (pessoa, animal, veículo, etc.). No caso de medida de grandezas escalares, deseja-se detectar se, em algum momento, o valor ultrapassou algum patamar pré-estabelecido (temperatura, concentração, pH, etc.). Neste último caso, a amostragem contínua pode também estar interessada em todos os valores, na forma da média ou de uma série de leituras.

Nas leituras contínuas, caberá ao nó sensor avaliar sua posição atual, comparar com as coordenadas da tarefa de sensoriamento recebida e efetuar ou não leituras.

Na amostragem sob demanda, o nó sensor não analisa as coordenadas geográficas onde se deseja monitorar o evento, nem precisa armazenar medidas efetuadas. Trata-se apenas de uma troca de mensagens momentânea entre um agente móvel e um grupo local de sensores. Já na amostragem contínua, deve haver uma programação dos sensores com a tarefa a ser executada. Esta programação deve conter as coordenadas de interesse, o período de amostragem e o perfil de armazenamento: detecção binária (ocorreu ou não), detecção binária de valor (ultrapassou ou não um patamar) ou acúmulo de leituras seqüenciais. Esta programação dos sensores pode ser feita pelos agentes móveis propostos neste trabalho. Os agentes migram pela rede disseminando a tarefa entre os nós sensores. A programação pode ser feita ainda por inundação. Técnicas de programação que envolvam mensagens enviadas diretamente a nós sensores específicos não são consideradas. Isto se deve ao fato da presente proposta, como será visto adiante, não necessitar e não considerar a execução de protocolos de roteamento na rede.

## 3.2 Algoritmo de localização

Todos os nós da rede deverão executar um método de localização.

Diante do exposto na Seção 2.3.1, recomenda-se a técnica de localização distribuída, independente do sinal, baseada em sementes que divulgam sua localização aos nós daquela região. Técnicas de estimativa de localização e correção (ajustes) [7] devem ser aplicadas. As sementes necessárias na etapa de ajustes das estimativas serão sensores capazes de identificar sua localização, tipicamente sensores dotados de GPS ou sensores que estejam recebendo esta informação de entidades externas à rede.

Neste trabalho considera-se que está em execução um algoritmo de localização e seu resultado está disponível em cada nó.

## 3.3 Previsão de localização do nó

Considerando os resultados do algoritmo de localização, esta tese propõe que os nós façam previsões (estimativas) de suas localizações futuras e da direção (coeficiente angular) do seu movimento. Trata-se de uma simplificação e adequação do filtro de Kalman [27] às RSSF.

Estas previsões da posição futura e da direção do movimento são utilizadas pelos agentes móveis na sua tomada de decisão, como apresentado no Capítulo 4.

A partir da computação da localização atual do nó, dois parâmetros são estimados: sua posição futura e a direção do movimento. Estes parâmetros serão a base do algoritmo de decisão dos agentes móveis (Capítulo 4).

Um algoritmo de localização computa periodicamente a posição atual $P$ do nó. Com duas destas medidas, $P_i(x_i, y_i)$ e $P_{i-1}(x_{i-1}, y_{i-1})$, o nó computa a inclinação $m$ da reta definida por estes pontos. Na Equação 3.1, $m_i$ representa a direção momentânea do deslocamento do nó; na Equação 3.2, $m_{pred_i}$ é a previsão da direção de deslocamento; $m_{pred_{i-1}}$ é a última previsão feita; e o parâmetro $\alpha$ é usado para dar maior ou menor peso ao último coeficiente angular calculado, $m_i$, em comparação ao histórico dos valores estimados, $m_{pred_{i-1}}$. Este algoritmo [28, 29] produz previsões

suficientes para nortear as tomadas de decisão dos agentes móveis.

$$m_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \tag{3.1}$$

$$m_{pred_i} = \alpha\, m_i + (1 - \alpha)\, m_{pred_{i-1}} \quad (0 \leq \alpha \leq 1) \tag{3.2}$$

Uma vez que a inclinação $m_i$ do movimento é estimada, a distância que o nó percorrerá no próximo passo é estimada da mesma maneira (Equação 3.3). O termo $d_{pred_{i-1}}$ representa a última previsão e $d_i$ é calculado em cada passo, baseado na posição "real" do nó. A localização "real" vem de um algoritmo de localização e incorpora um erro intrínseco. O termo $\gamma$ tem a mesma função que o $\alpha$.

$$d_{pred_i} = \gamma\, d_i + (1 - \gamma)\, d_{pred_{i-1}} \quad (0 \leq \gamma \leq 1) \tag{3.3}$$

Tanto a computação de $m_{pred_i}$ quanto a de $d_{pred_i}$ são de complexidade computacional constante, o que atende aos requisitos de economia de recursos das RSSF.

Foram realizadas simulações para validar o modelo de predição de localização proposto. O Anexo A apresenta a descrição completa do simulador desenvolvido. A seguir são detalhados os cenários, os testes e os resultados obtidos.

### 3.3.1 Trajetória estritamente crescente

A primeira etapa de testes buscou validar o algoritmo de predição de localização dos nós (Seção 3.3). O valor de $\alpha$ na Equação 3.2 foi fixado em $0,5$ , ou seja, a última medição e última estimativa (histórico) têm o mesmo peso. Foi gerada uma trajetória dita real, de um nó, baseada no seguinte padrão: o nó parte da origem e as coordenadas $x$ e $y$ são estritamente crescentes segundo as Equações 3.4.

$$\begin{gathered} x_{i+1} = x_i + \delta\ ,\ y_{i+1} = y_i + \delta' \\ \delta,\ \delta' \in \mathbb{R},\ 0 < \delta,\ \delta' \leq 10 \end{gathered} \tag{3.4}$$

Além da estimativa do coeficiente angular (direção) do deslocamento, foi estimada, da mesma forma, a velocidade de deslocamento. Ou seja, se estimou a direção e o módulo do vetor velocidade em cada passo da simulação.

Nesta situação, como a trajetória apresenta reduzida variação de direção e velocidade escalar, o previsor acompanha de forma satisfatória a trajetória real (Figura 3.1 e Tabela 3.1). Estas previsões, como argumento de decisão para os agentes móveis, irão produzir migrações corretas na busca da região alvo.

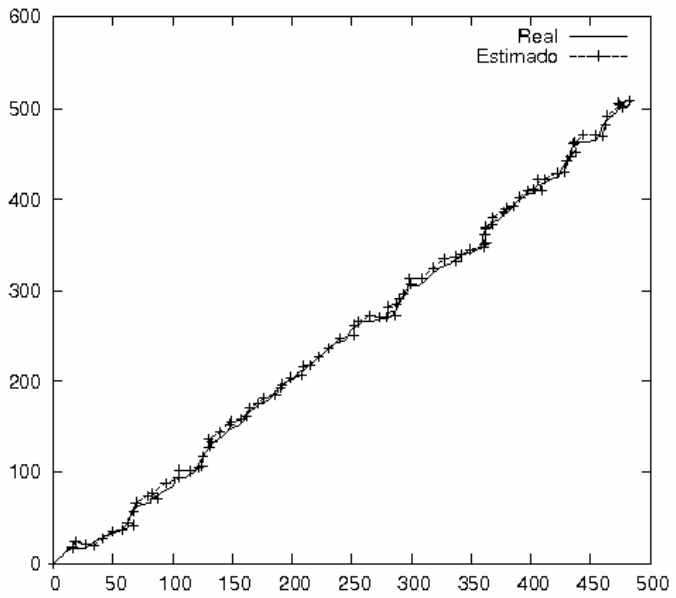



Figura 3.1: Predição de localização do nó - trajetória com menor variação

| | Trajetória regular | Trajetória Manhattan |
|---|---|---|
| Erro mínimo | $0,74$ | $0,62$ |
| Erro máximo | $11,13$ | $32,5$ |
| Erro médio | $4,67$ | $13,59$ |
| Desvio padrão | $2,35$ | $7,7$ |

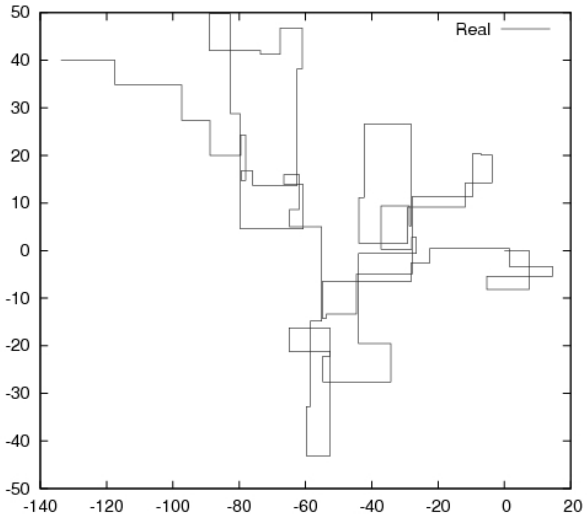
Tabela 3.1: Estatística das previsões de localização do nó (metros)

### 3.3.2 Trajetória do tipo *Manhattan*

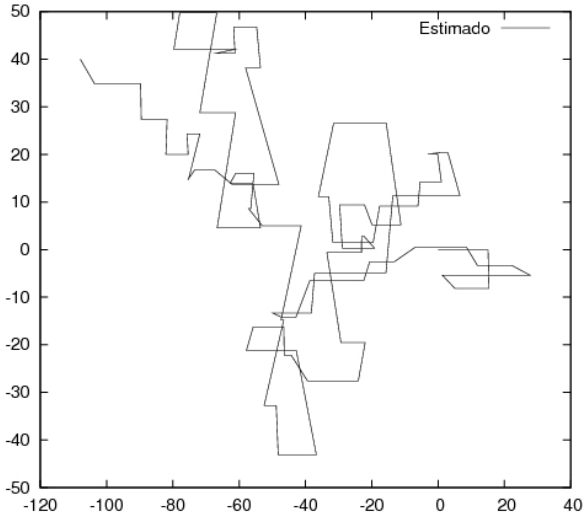
Neste caso, buscou-se avaliar o algoritmo de predição de localização dos nós em um segundo cenário, com mobilidade menos regular do que a usada na Seção 3.3.1. O parâmetro $\alpha$ foi mantido em $0,5$. A trajetória real segue o modelo *Manhattan* [30] [1].

[1] A trajetória *Manhattan* permite apenas deslocamentos na mesma direção dos eixos $X$ e $Y$. O nome faz alusão ao formato quadriculado da maior parte das ruas na ilha de *Manhattan*.

Usou-se o mesmo método para estimar direção e módulo da velocidade. Esta trajetória apresenta uma maior variação de direção, com mudanças de $90^o$. A cada movimento, a coordenada sorteada é modificada em uma faixa de $-20$ a $20m$ (2 vezes mais que no caso anterior). Observa-se na Figura 3.2(a) que a trajetória se desloca por toda a área do gráfico, apresentando mudanças completas de direção. O previsor obtém sucesso nos resultados gerados (Figura 3.2(b) e Tabela 3.1) e estes permitirão a migração acertada dos agentes móveis.




(a) Trajetória real do tipo *Manhattan*




(b) Trajetória prevista para o modelo *Manhattan*

Figura 3.2: Previsão de localização para a trajetória *Manhattan*

## 3.4 Manutenção do conjunto de vizinhos: revisão bibliográfica

Ao ser inicializado, no surgimento da rede como um todo, ou na adição posterior de novos sensores, o nó deve identificar a presença de vizinhos. Em um ambiente móvel, pode não haver nenhum vizinho em um instante inicial e, posteriormente, o nó pode entrar no raio de comunicação de um ou mais vizinhos. A lista de vizinhos precisa ser criada e mantida, já que os nós são móveis.

Quando executada na formação inicial da rede, esta etapa é chamada de auto-organização.

Em última instância, para um quadro ser enviado a um nó destinatário, o endereço deste nó precisa ser conhecido pela origem. No modelo de referência ISO-OSI [31, 32], o endereço de mais baixo nível é o endereço da camada de enlace (camada 2). É o chamado endereço MAC. A administração destes endereços é feita pelo IEEE (*Institute of Electrical and Electronics Engineers* [33]) e eles são supostamente unívocos em todo o mundo.

É comum a existência de um ou mais endereços nas camadas superiores, como endereços de rede (endereço IP [34] , por exemplo), portas de transporte, URLs e endereços eletrônicos nas aplicações. Na presente proposta, o único endereço necessário é o endereço MAC da interface de comunicação de cada nó.

Para o envio de um quadro da camada de enlace para todos os nós integrantes da rede local, basta usar o endereço de difusão (*broadcast*) da camada 2. Todos os nós ao alcance do emissor irão receber e processar este quadro como sendo endereçado a eles.

Para o envio de um quadro a um destinatário específico (*unicast*), é necessário informar no cabeçalho do quadro o endereço de destino. Deste modo, é preciso, antes de tudo, descobrir o endereço MAC do destinatário. Em redes sem fio com nós móveis, esta descoberta e a manutenção do conjunto de vizinhos adquire um maior grau de complexidade. Não se pode sobrecarregar o canal de comunicação com muitas mensagens de anúncio e descoberta e deve-se ainda considerar a alta volatilidade das informações obtidas, já que a mobilidade afasta vizinhos atuais e traz novos vizinhos a todo instante. Deve-se observar que o trabalho proposto por

esta tese visa atender inclusive a cenários com alto grau de mobilidade.

A seguir, prossegue-se com a revisão bibliográfica das propostas de protocolos de enlace e descoberta de vizinhos. Algumas são de uso geral e outras específicas para RSSF.

Após a revisão, na Seção 3.5, é proposto um protocolo específico para os cenários de interesse, uma vez que todos aqueles estudados apresentam alguma deficiência no atendimento à mobilidade.

### 3.4.1 *Address Resolution Protocol* - ARP

Em redes cabeadas, usa-se, por exemplo, o protocolo de resolução de endereços (*Address Resolution Protocol* - ARP [35]). De posse do endereço da camada superior (camada de rede) do destinatário, o nó origem envia um quadro de enlace em *broadcast*, indagando quem possui o referido endereço de rede. Este quadro contém o endereço MAC do emissor. O destinatário, proprietário do endereço de rede informado, responde em *unicast* ao nó emissor, informando o seu endereço MAC. Assim, o quadro pode ser transmitido. Esta solução não se adequa à presente proposta, uma vez que se baseia no conhecimento do endereço de destino em uma camada superior, no caso o endereço de rede. Esta informação não existe no presente modelo. Na verdade, como será visto mais adiante, o esquema proposto nesta tese necessita apenas das camadas 7, 2 e 1: aplicação, enlace e física.

### 3.4.2 O Padrão IEEE 802.15.4

O Padrão IEEE 802.15.4 [36, 37] tem sido muito adotado nas RSSF. Uma implementação muito difundida deste padrão é o ZigBee [38, 39]. Apesar desta grande aceitação, não são encontradas na literatura aplicações do referido padrão em ambientes de alta mobilidade.

O Padrão 802.15.4 especifica a existência de dois tipos distintos de dispositivos: dispositivos de plenas funcionalidades (*full-function device* - FFD) e dispositivos de funcionalidades reduzidas (*reduced-function device* - RFD). Um FFD pode atuar como coordenador de uma rede local pessoal (*personal area network* - PAN) ou como um dispositivo. Um FFD pode se comunicar com um RFD ou com outro FFD. Um RFD só pode se comunicar com um FFD. A questão relevante é que o

padrão especifica que uma PAN sem fio (*Wireless* PAN ou WPAN) deve incluir pelo menos um FFD operando como coordenador. Isto é inapropriado para redes com alto grau de mobilidade, já que associações e desassociações entre coordenadores e "coordenados" serão freqüentes e imprevisíveis.

Uma rede 802.15.4 pode operar na topologia estrela ou par-a-par (*peer-to-peer*). No primeiro modo de operação, todas as comunicações se dão através do coordenador, o que acentua a inadequação aos cenários móveis citada no parágrafo anterior.

Na topologia par-a-par, é possível a comunicação direta entre nós da rede, mas a existência do coordenador permanece. Ele mantém as funções de determinação do identificador único da PAN e de autorização para o ingresso de novos dispositivos na PAN. Apesar de extrapolar o escopo do padrão, o texto exemplifica para a definição do coordenador o fato de ter sido o primeiro dispositivo a comunicar no canal. Este exemplo é obviamente destinado a redes estáticas.

O padrão especifica o uso do *broadcast* de *beacons*[2] para a formação da PAN. No caso de dois ou mais FFDs tentarem se estabelecer como coordenadores, um mecanismo de contenção deve ser utilizado, ou seja, ao detectarem os múltiplos envios de *beacons*, os candidatos aguardarão intervalos de tempo aleatórios para efetuarem uma nova tentativa. Isto fará com que um deles retransmita o *beacon* antes dos demais, sendo este o futuro coordenador da PAN.

Um dispositivo órfão é aquele que perdeu o sincronismo com seu respectivo coordenador ou que perdeu o contato com o mesmo. Em ambos os casos, a reação a ser tomada envolve uma varredura (*orphan scan*) de um conjunto de canais, em busca de anúncios de coordenadores (*beacons*). Após a detecção de um coordenador, sucede-se um ciclo de associação do dispositivo ao coordenador. O surgimento de dispositivos órfãos, a necessidade de varreduras de canais e de novas associações serão freqüentes e proporcionais ao grau de dinamismo da rede. A manutenção do conjunto de vizinhos de um nó nestas situações dinâmicas, utilizando apenas o Padrão 802.15.4, será inviável.

Em uma rede hierarquizada em *clusters* (Figura 3.3 [37]), dispositivos devem conectar *clusters* vizinhos, de modo a manter a conectividade global da rede. Isto

[2]Um *beacon* é um pequeno quadro de sinalização utilizado em redes de computadores, sobretudo em redes sem fio. Este termo será muito utilizado ao longo deste trabalho. Será mantido o uso do termo em inglês.

aumenta ainda mais a complexidade da rede em caso de nós móveis.

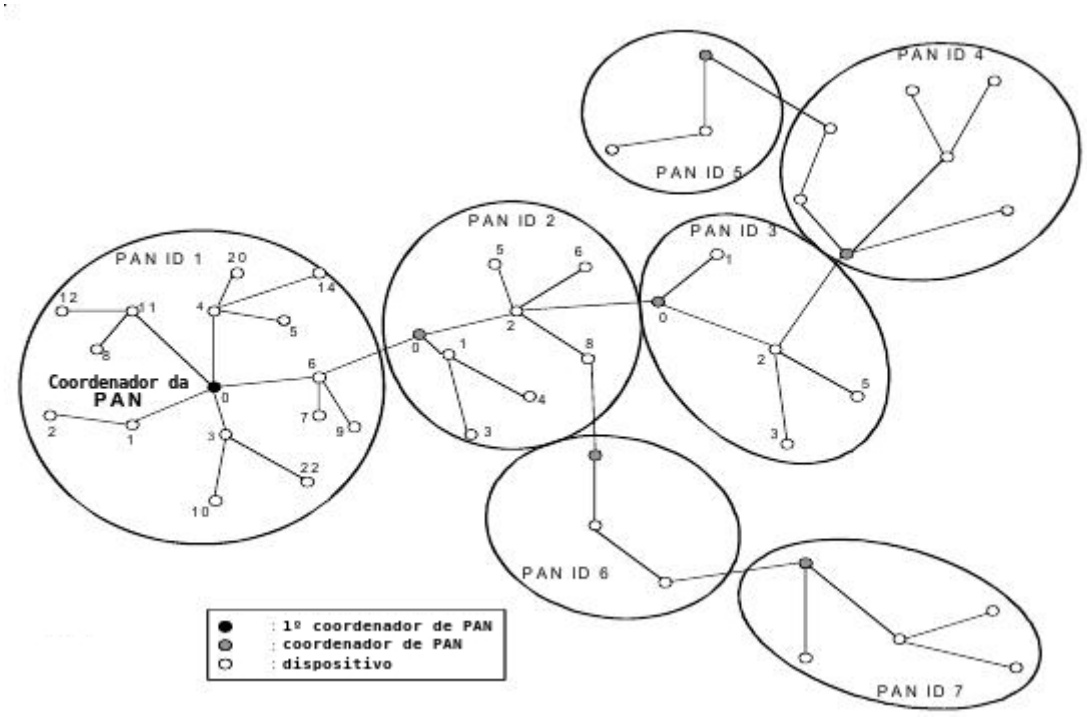



Figura 3.3: Rede hierarquizada em *clusters*

O padrão prevê a existência de coordenadores alternativos para uma PAN. Estes devem ser capazes de substituir o coordenador caso este deixe a rede por alguma razão. Apesar de aumentar as chances de manutenção da conectividade entre vizinhos, não elimina o problema inserido pela necessidade de se ter um coordenador ativo em cada PAN.

O desempenho das redes IEEE 802.15.4 cai rapidamente com aumento da mobilidade na rede [40].

### 3.4.3 Protocolos SMACS e EAR

Sohrabi *et al.* [15] apresentam dois algoritmos para auto-organização de RSSF: SMACS (*Self-Organizing Medium Access Control for Sensor Networks*) e EAR (*Eavesdrop-And-Register*). Os algoritmos destinam-se sobretudo a redes com nós estáticos ou com um subconjunto de nós de baixa mobilidade. Estes protocolos são analisados também por Loureiro *et al.* [9].

O protocolo SMACS executa a tarefa de descoberta de vizinhos. Inicializa e organiza de forma distribuída a camada de enlace, sem a necessidade de um mestre.

Não forma *clusters* ou topologias hierarquizadas. Ele aplica-se à auto-organização da rede estática.

Após a distribuição física, os nós sensores "despertam" (inicializam) aleatoriamente. Além da descoberta de vizinhos, o protocolo aloca um par de intervalos de tempo (*slots*) destinados à transmissão e recepção entre os diversos pares de nós vizinhos, em um esquema de multiplexação por divisão de tempo (TDMA). Estes *slots* são alocados dentro de intervalos maiores, os *superframes*, que possuem tamanho fixo para todos os nós. Entretanto, os diversos *superframes* não precisam estar alinhados (sincronizados) em toda a rede. Além da alocação dos *slots* de TDMA, os canais são alocados em freqüências escolhidas aleatoriamente dentre um amplo conjunto de opções, de modo a reduzir a probabilidade de colisões. De uma maneira mais geral, os autores propõem o uso de espectro espalhado (*spread spectrum*) em lugar de faixas de freqüência fixas. Cada enlace é alocado segundo um padrão distinto de saltos em freqüência.

O nó é capaz de desligar seu transceptor durante os *slots* não utilizados do seu *superframe*, economizando energia de forma significativa.

No protocolo SMACS, os *slots* são alocados de forma não síncrona (*nonsynchronous scheduled communication - NSC* [15]). Deste modo, enlaces de comunicação podem ser alocados de forma dinâmica e distribuída, durante o tempo de vida da rede.

O processo descrito anteriormente é iniciado com a descoberta de vizinhos. Após despertar, um nó escuta o meio por um tempo aleatório, em uma faixa de freqüência fixa e padronizada. Caso não receba nenhum convite, o nó decidirá por transmitir um. Os nós convidados enviam resposta à origem do convite. Uma terceira mensagem especifica qual dos convidados foi selecionado (escolhido) e uma quarta especifica os parâmetros que serão usados na comunicação. Este processo gera o que os autores chamam de subredes. Estas subredes são formadas por um subconjunto de nós que formam um grafo conexo e que compartilham as mesmas épocas de *superframes*. Estes *superframes* são intervalos de tempo que contém os *slots* de comunicação da subrede e que são apenas localmente sincronizados. Uma vez que existam nós que interconectem subredes adjacentes, a auto-organização da rede estará concluída. Deste modo, é automaticamente formada uma infraestrutura que

suportará transporte local e de longa distância na RSSF. Após a auto-organização de tal infraestrutura, pode-se utilizar a rede para as finalidades desejadas, aplicando, por exemplo, técnicas de transporte confiável [41, 42, 43, 44]; difusão de interesses e coleta de dados [19]; injeção de agentes móveis [2, 45]; etc.

O protocolo SMACS é escalável e eficiente em termos de consumo de energia [46].

A parte TDMA proposta para o SMACS busca a economia de energia, uma vez que os nós desligarão seus transceptores durante os *slots* não usados. A parte relacionada à freqüência (FDM ou FHSS) se destina a minimizar a probabilidade de colisões das transmissões. No caso de FHSS, há ainda uma redução da vulnerabilidade dos enlaces a ruídos acidentais ou maliciosos (*jamming*)[15]. Outro efeito desejável do FHSS é a introdução de uma codificação nas transmissões, que é a própria seqüência dos saltos, o que torna o canal mais seguro contra escutas não autorizadas.

A reserva de *slots* do esquema TDMA do SMACS vai gerando uma sincronização local dos *superframes* em uma subrede, ou seja, nós com a mesma "época" de *superframes*. O SMACS se destina à auto-organização inicial da rede, para um ambiente com nós estáticos. O custo de manutenção e escalonamento dos *slots* de tempo nos *superframes*, e nas correspondentes "épocas" locais, se justifica pela economia de energia alcançada. Por outro lado, uma vez concluída a etapa inicial de auto-organização da rede, as variações no *superframe* de um nó serão reduzidas. Elas ocorrerão apenas com a chegada ou partida de um nó móvel, segundo o protocolo EAR também estudado mais adiante.

Evitar colisões apenas no domínio da freqüência é uma consideração razoável. Assumindo-se que os rádios operam na faixa ISM de 902 a $928 MHz$ [47], com taxas de 10 *kbps* por canal, tem-se em torno de 2600 bandas de freqüência distintas [15]. Como o protocolo SMACS prevê a escolha aleatória de um canal para cada par de nós, a probabilidade de colisão é reduzida. Além disso, existe uma última etapa do protocolo que prevê a troca de um par de curtas mensagens de teste para validar a alocação do canal escolhido.

Uma vez que a rede estacionária está formada, nós móveis podem começar a interagir com ela. O objetivo do protocolo EAR [15] é prover a conectividade destes nós móveis com a rede estacionária formada e mantida pelo SMACS.

Como foi visto, o protocolo SMACS aloca *slots* de tempo para a comunicação par-a-par dos sensores. Para a introdução do EAR, os autores propõem a reserva de um *slot* privilegiado para a adesão de sensores móveis à subrede local.

A seguir o funcionamento do protocolo EAR.

Durante um *slot* pré-determinado do SMACS, o nó estacionário deve transmitir algum tipo de mensagem de convite à sua vizinhança, com o intuito de convidar novos nós estacionários a se juntarem à subrede local. Isto ainda é o funcionamento normal do SMACS. Com a introdução do EAR, cabe ao nó móvel simplesmente monitorar estas mensagens de convite do protocolo MAC estacionário, ou seja, do SMACS. Ao perceber a ocorrência de um convite, o nó móvel toma uma decisão com base no protocolo EAR. Uma das ações dos sensores móveis é a manutenção constante de uma lista de vizinhos. Cabe a eles criarem e finalizarem conexões com nós estacionários na sua vizinhança.

O algoritmo EAR faz uso das seguintes mensagens:

- ***Broadcast Invite* (BI):** o nó estacionário convida outros nós a se juntarem à subrede local (mensagem do SMACS);
- ***Mobile Invite* (MI):** segundo um critério de decisão (qualidade do canal (SNR - *Signal-to-noise ratio*), potência do sinal (relacionada à proximidade), nível de energia do vizinho, etc.), o nó móvel responde ao **BI**, requisitando uma conexão;
- ***Mobile Response* (MR):** havendo um par de *slots* disponíveis, o nó estacionário aceita o **MI** e os *slots* TDMA são alocados para a comunicação;
- ***Mobile Disconnect* (MD):** em algum momento futuro, o nó móvel informa ao estacionário da desconexão.

Observa-se que o papel do nó estacionário é mínimo e que o nó móvel pode adotar critérios de seleção para decidir a qual nó estacionário se conectar.

Para o suporte à um ambiente com mobilidade de todos os nós, uma combinação dos dois protocolos, SMACS e EAR, possibilita a manutenção do conjunto de vizinhos. Os protocolos permitem ainda a alocação e liberação de recursos de comunicação no caso de aproximação ou afastamento de nós móveis.

As características dos protocolos, segundo os próprios autores, faz com que os mesmos suportem apenas a baixa mobilidade de um subconjunto dos nós da rede. As simulações foram feitas com deslocamento de um nó, a $0,1\,m/s$, por um campo de 45 nós estáticos.

Nos cenários tratados pela presente tese, a mobilidade pode ser mais intensa e generalizada, o que torna inadequado o uso da proposta original do SMACS e EAR.

### 3.4.4 O Padrão IEEE 802.11

Segundo Kurose e Ross [48], o padrão mais utilizado para redes locais sem fio é o IEEE 802.11 [49, 50].

Uma rede local 802.11 é baseada em uma arquitetura celular, na qual o sistema se divide em células, cada uma chamada de conjunto de serviços básicos (*Basic Service Set - BSS*). Cada célula é controlada por uma estação base (*Base Station*), também chamada de ponto de acesso (*Access Point* - AP) [51]. Este modo de operação é o chamado modo infraestruturado.

Quando uma estação quer acessar uma BSS, ela precisa coletar informações de sincronização do AP. A estação pode coletar estas informações de duas maneiras:

- Varredura passiva (*passive scanning*): a estação espera receber um *beacon* do AP;
- Varredura ativa (*active scanning*): a estação envia uma requisição explícita (*Probe Request*) e aguarda a resposta (*Probe Response*) do AP.

No Padrão 802.11, tem-se a presença centralizadora do AP e as operações de associação, desassociação e reassociação. Este esquema é inapropriado às redes com alta mobilidade generalizada.

O IEEE 802.11 permite a operação no modo *ad hoc* (IBSS - *Independent* BSS). Neste caso, não existe a figura do AP e as estações se comunicam diretamente. Todos os nós dentro do raio de alcance, teoricamente, já fazem parte da rede. Mas isto implica dizer que um nó pode não conhecer todos os outros nós da rede, ou seja, não têm todos os endereços MAC dos demais. Para descobrir os endereços utiliza-se um procedimento de *broadcast*, onde o nó envia uma mensagem para todos os outros,

solicitando o endereço MAC. Desta forma, os nós que estão dentro do alcance do rádio respondem com seus endereços MAC, ao nó que solicitou.

Uma IBSS em uma rede 802.11 é simples e razoavelmente adequada aos cenários móveis. Existem, entretanto, algumas características não ideais.

Apesar da infraestrutura descentralizada, existe a formação da célula relacionada à IBSS. Uma primeira estação deve difundir *beacons* de sincronização e outras devem varrer os canais e aderir à célula. Deve ser gerado um identificador local da IBSS, o BSSID (*Basic Service Set Identification*). Alguma estação tem que responder às requisições de inclusão (*Probe Request* e *Response*, citadas anteriormente). Estes custos e atrasos não podem ser desconsiderados.

Outra questão se refere ao número de canais e às taxas de transmissão. O padrão prevê poucos canais, tipicamente menos de duas dezenas, dependendo da regulação do país. Há ainda a sobreposição de canais adjacentes, o que permite, na prática, o uso pleno de apenas 3 canais simultâneos. Isto se deve, em parte, pelas altas taxas de transmissão permitidas, de até $54Mbit/s$. Para as RSSF, é melhor a disponibilidade de mais canais, mesmo que com menores taxas de transmissão. Em geral, não existem fluxos de grandes volumes de dados nas RSSF.

### 3.4.5 O Padrão IEEE 802.15.1 - *Bluetooth*

O Padrão IEEE 802.15.1 [52] derivou-se das especificações do *Bluetooth*. O *Bluetooth* especifica um protocolo aberto de comunicação sem fio, destinado à troca de dados por curtas distâncias entre dispositivos fixos e móveis, criando as chamadas picorredes.

O *Bluetooth* usa multiplexação por divisão de tempo (*Time-Division Multiplexing* - TDM) com *slots* de 625 $\mu s$. A cada *slot*, um nó transmite em um dos 79 canais. São criadas picorredes de até 8 nós, sendo um deles o mestre e os demais escravos. Este mestre pode transmitir em todos os intervalos de tempo ímpares. Um escravo pode transmitir apenas imediatamente após o mestre ter se comunicado com ele. Além disso, o escravo só pode se comunicar com o mestre. Redes maiores podem ser criadas com base na existência de até 10 mestres.

O padrão usa dois procedimentos que permitem a conexão de novos nós à rede, o INQUIRY e o PAGE. O procedimento PAGE é usado quando o mestre da picorrede

conhece o endereço MAC do dispositivo. Já o procedimento INQUIRY, é utilizado quando não se conhece o endereço MAC. Para isto, é enviado um pacote especial, o INQUIRY PACKET, para todos os dispositivos dentro do alcance do mestre. Ambos os processos demandam algum tempo, mas o procedimento INQUIRY é mais demorado.

O Padrão *Bluetooth* tem uma vantagem sobre o IEEE 802.11 em relação a padronização do procedimento de busca. No *Bluetooth*, existe um forma definida de como é feita a busca, enquanto que no Padrão IEEE 802.11 não existe uma forma definida. Já a respeito da flexibilidade e rapidez, o Padrão IEEE 802.11 se sai melhor do que o *Bluetooth*. Isto porque no IEEE 802.11 basta o protocolo CSMA/CA emitir um pacote diretamente na rede. Já o *Bluetooth* tem que criar uma picorrede antes de enviar qualquer pacote. [53]

Em função do alcance de comunicação, da centralização em um mestre, da limitação do número máximo de nós (total e por picorrede), o *Bluetooth* não se adequa ao uso em RSSF, sobretudo com nós móveis.

### 3.4.6 Disco - Descoberta assíncrona de vizinhos

Dutta e Culler [40] apresentam o protocolo Disco. Trata-se de um mecanismo de descoberta assíncrona de vizinhos para aplicações móveis de sensoriamento.

O protocolo permite que os nós operem suas interfaces de comunicação em baixos ciclos de trabalho (1%, por exemplo) e ainda descubram e se comuniquem com vizinhos durante encontros infreqüentes e oportunistas, sem a necessidade de qualquer sincronismo prévio.

O protocolo Disco toma um par de números primos de modo que a soma de seus inversos seja igual ao ciclo de trabalho do rádio. Cada nó incrementa um contador local com um período fixo global. Se o contador local é divisível por ambos os primos, o nó liga o seu transceptor por um certo período. O protocolo garante que dois nós terão algumas sobreposições dos períodos de atividades dos seus transceptores, dentro de um tempo limitado, mesmo que escolham de forma independente os ciclos de trabalho do rádio.

A Figura 3.4 ilustra a forma simplificada de execução do protocolo. A primeira linha representa um tempo global, não utilizado pelos nós. Os pares de linhas se-

guintes representam números primos. A primeira de cada par mostra o incremento dos relógios locais. Observa-se que eles são iniciados de forma assíncrona. A segunda linha de cada par mostra o resto da divisão do contador local pelo respectivo número primo. Quando surge o valor zero nesta linha (células em cinza), o nó liga o transceptor. Pode-se observar que surgem instantes em que dois nós ligam suas interfaces de comunicação, possibilitando a descoberta recíproca e um eventual agendamento de comunicação (*Rendezvous*). Estes instantes de comunicação estão representados por retângulos com zeros nas duas extremidades.

Apenas no instante global 31 houve uma colisão, onde 3 nós estariam com suas interfaces de rádio em operação.

Temos, por exemplo, ciclos de trabalho de aproximadamente 33% para o número primo 3; 20% para 5 e assim por diante.

Os autores demonstram que não há necessidade de sincronismo entre os contadores. Ou seja, os contadores não precisam estar alinhados como estão na Figura 3.4. O protocolo é sensível à qualidade do par de primos escolhidos. A descoberta de pares de primos para se atingir um ciclo de trabalho específico não é trivial.

O protocolo privilegia a economia de energia, alternando ciclos de trabalho e de repouso das interfaces de rádio. Ele oferece limites superiores para a latência na descoberta de vizinhos, mas os valores apresentados são relativamente altos.

Foram levantados alguns valores significativos. Para atingir o conhecimento de 50% do vizinhos, foi observada uma latência de $4,44s$, com uma escolha favorável dos primos. Para descobrir 85%, verifica-se uma latência de $10s$.

O protocolo tem pouca escalabilidade. Resultados mostraram, para 5 vizinhos, latências entre $10s$ e $80s$ até a descoberta de todos eles. Estes valores, e mesmo os do parágrafo anterior, serão inviáveis para deslocamentos de $10m/s$, por exemplo. Para uma velocidade neste patamar, um raio de alcance de rádio de $30m$, sofrerá desconexão em $3s$, ou seja, todo o conjunto de vizinhos pode ter sido modificado antes da descoberta ter sido concluída.

O protocolo possibilita ajustes do período de incremento do relógio local e da duração do período de atividade, em cada ligação do transceptor. Durante os períodos de atividade, são enviados *beacons* para possibilitar a descoberta mútua. Incremento mais agressivo e períodos de atividade mais longos aumentarão o desem-

penho do algoritmo. Entretanto, tais situações não foram exploradas pelos autores, que não abordaram cenários móveis na implementação ou nas simulações. O protocolo destina-se principalmente a cenários estáticos ou ao eventual encontro de poucos nós e uma subseqüente posssibilidade de comunicação entre eles. Não se adapta a grandes variações no conjunto de vizinhos.

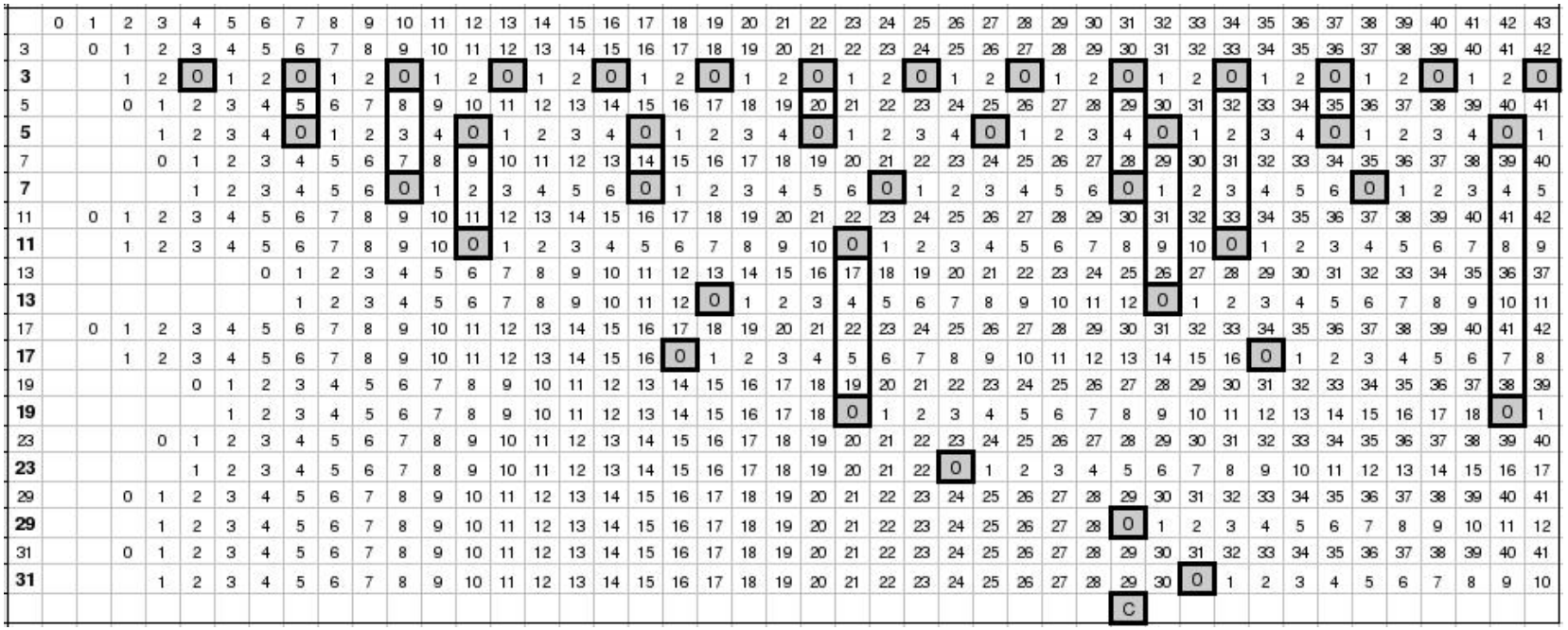



Figura 3.4: Exemplo simplificado da descoberta de vizinhos pelo protocolo Disco

### 3.4.7 ENDP - Descoberta de vizinhos eficiente em RSSF móveis

Kohvakka *et al.* [16] propõem a descoberta de vizinhos com eficiência energética em RSSF móveis (*Energy-efficient neighbor discovery protocol for mobile wireless sensor networks*).

O protocolo proposto deve ser executado sobre outro protocolo de enlace que produza sincronização local, como IEEE 802.15.4 ou S-MAC [54]. Utilizando estas informações de sincronização e os próprios *beacons* do protocolo de enlace em operação, é feito um escalonamento otimizado de escuta do canal, economizando energia. Através dos *beacons* já em uso na camada de enlace, cria-se um conhecimento de vizinhança de dois saltos, incluindo o canal em uso pelo vizinho. Utiliza-se um esquema de temporização para excluir informações supostamente obsoletas (não recentemente atualizadas).

A descoberta de vizinhos é feita através da varredura de todos os canais de freqüência em uso. Os autores recomendam o uso de um canal fixo em toda a rede para descoberta de vizinhos.

Apresenta-se um fator $k$ que indica quantos enlaces redundantes devem ser mantidos. Isto permite maior robustez diante do dinamismo oriundo da mobilidade dos nós.

As simulações foram feitas com apenas um nó móvel, em um cenário de sensores estáticos. A velocidade deste único nó móvel variou de $0,1$ a $10m/s$. Os experimentos reais foram feitos também com um único nó móvel e velocidades de 1 a $3m/s$.

### 3.4.8 MS-MAC - MAC para RSSF ciente da mobilidade

O protocolo MS-MAC (Mobility-aware Sensor MAC) [17, 18] ajusta o intervalo de varredura da rede de acordo com a mobilidade observada. Sem mobilidade, uma varredura de $10s$ é feita a cada 5 minutos. Quando o nó detecta mobilidade em função de variações na potência do sinal, o processo de descoberta de vizinhos é iniciado instantaneamente. Em redes dinâmicas a descoberta de vizinhos é feita no máximo duas vezes por minuto.

Os nós não precisam ter o mesmo escalonamento de atividade/inatividade (*wakeup/sleep*) em toda a rede, mas eles precisam estar divididos em *clusters* virtuais, nos quais todos os nós seguem o mesmo escalonamento (sincronismo) para estes períodos. Em cada *cluster* virtual, os nós despertam ao mesmo tempo para transmitir e receber pacotes. Se o nó não tem o que transmitir e se não há pacotes a ele endereçados, ele volta a "dormir". Transmissões de dados seguem o esquema de detecção de portadora e uso de RTS/CTS (Request to send / clear to send) proposto no Padrão IEEE 802.11.

Informações de sincronização do *cluster* precisam ser transmitidas em *broadcast* a cada 10 ciclos. O processo se repete a cada 5 minutos e pode ser repetido mais freqüentemente caso o nó não tenha nenhum vizinho. Em uma postura pró-ativa, a freqüência da ressincronização deve ser ajustada dinamicamente e deve ser inversamente proporcional ao número de vizinhos. Utiliza-se ainda a leitura das variações no sinal, citadas no primeiro parágrafo desta seção.

O esquema se mostrou superior ao S-MAC com o aumento da velocidade de deslocamento. Ele manteve maiores índices de conectividade, com um pequeno aumento no consumo. Entretanto, as simulações mantiveram 100 nós estáticos, enquanto apenas um nó se moveu pela área de sensoriamento.

Tem-se ainda a característica do uso de RTS/CTS, que é custosa em termos de consumo de banda, apesar de resolver o problema do terminal oculto[3]. O custo do RTS/CTS pode ser notado pelo fato das redes 802.11 permitirem que este esquema seja desabilitado para quadros menores que um patamar definido.

[3]Este problema ocorre quando tem-se três estações de comunicação sem fio ($A \Rightarrow B \Leftarrow C$) dispostas de modo que a estação central $B$ está ao alcance das outras duas ($A$ e $C$) e estas duas estão fora do alcance uma da outra. Caso $A$ e $C$ "escutem o canal", o identificarão como disponível e começarão a transmitir. Ocorrerá, entretanto, colisão na estação $B$, que receberá ambas as transmissões. Caso $A$ envie a $B$ um pedido para transmitir (RTS) e $B$ envie a autorização (CTS), $C$ também receberá a autorização e aguardará. Assim, não ocorrerá colisão.

### 3.4.9 Outras técnicas relacionadas ao controle de vizinhos

#### O protocolo IMEP

O protocolo IMEP (*Internet MANET Encapsulation Protocol*) [55] destina-se a dar suporte operacional a protocolos de roteamento, protocolos de controle ou outros quaisquer das camadas superiores. Ele incorpora mecanismos de suporte ao monitoramento de enlaces, conectividade de vizinhos, agregação e encapsulamento de pacotes de controle, difusão (*broadcast*), etc. Sempre para ambientes de comunicações sem fio [4].

O objetivo principal do protocolo IMEP é o de melhorar o desempenho global da rede, reduzindo o número de pacotes de controle do nível de rede. Isto é feito através do encapsulamento e agregação de múltiplos pacotes de controle (por exemplo, pacotes de roteamento, reconhecimentos, pacotes de monitoramento do estado dos enlaces, resolução de endereços de rede, etc) em mensagens IMEP maiores.

O uso do IMEP é desejável uma vez que o atraso de esquemas baseados em contenção, tais como CSMA/CA, IEEE 802.11, FAMA [56], é significativo e, por conseguinte, favorece a utilização de um menor número de mensagens maiores. Também pode ser útil em esquemas baseados em reserva de tempo (*time-slotted*), onde pequenos pacotes devem ser juntados (agregados) em pacotes melhor dimensionados à duração dos *slots* de tempo.

Trata-se, entretanto, de um protocolo da camada 3 (camada de rede), não dispensando um protocolo de enlace subjacente.

#### Eleição de líder e iminência de migração

Tarefas locais podem ser coordenadas por um líder local. Este líder ou coordenador pode ser definido como o primeiro a tentar assumir tal função, como visto nas Seções 3.4.2 e 3.4.4.

Para a eleição de líder, podem ser exploradas as técnicas descritas em [57], Seção 5.1, pp. 117-129. O identificador unívoco necessário como critério da eleição do líder poderá ser o nível de energia do nó ou a probabilidade deste deixar a região vizinha.

A técnica da decisão bayesiana com modelagem *fuzzy*, apresentada por Bertini *et al.* [25], pode ser aplicada para que os nós identifiquem a iminência de migração

da vizinhança e informem aos demais que irão, provavelmente, se desconectar.

No caso do líder, a iminência da migração ou a transposição do limite crítico de energia, deverá desencadear uma nova eleição de líder ou delegação direta da liderança a outro.

Devido aos altos graus de mobilidade a serem atendidos, nenhuma destas técnicas serão incorporadas ao esquema proposto. Será privilegiada a solução mais simples, de modo a reduzir a complexidade e o consumo de recursos energéticos e de banda de comunicação. Entretanto, por se tratar de uma área de pesquisa relativamente nova – RSSF com alto grau de mobilidade –, tais sofisticações podem ser exploradas no futuro.

| **Protocolo** | **Mobilidade** | **Centralizado / Distribuído** | **Sincronização?** | **Observações** |
|---|---|---|---|---|
| ARP | Sim | Distribuído | Não | Requer conhecimento do endereço da camada superior |
| IEEE 802.15.4 e ZigBee | Moderada | Localmente centralizado | Não | Exige um coordenador em cada PAN |
| SMACS | Não | Distribuído | Sim | Localmente sincronizado. Não requer alinhamento dos *slots* em toda a rede |
| EAR | Parcial | Distribuído (legado do SMACS) | Sim (legado do SMACS) | Testado com mobilidade parcial e muito reduzida |
| IEEE 802.11 infraestruturado | Sim | Celular (localmente centralizado) | Sim, em cada BSS | Exige um AP em cada BSS |
| IEEE 802.11 *ad hoc* | Sim | Distribuído | Apenas de *beacons* | Poucos canais com altas taxas de transmissão |
| IEEE 802.15.1 *Bluetooth* | Moderada | Centralizado | Sim | |
| Disco | Moderada | Distribuído | Não | Apenas a freqüência de incremento do contador e o tempo de serviço do rádio em cada intervalo de atividade são parâmetros globais. Os autores mencionam a mobilidade como motivação para a necessidade da descoberta de vizinhos, mas não a exploraram na implementação ou nas simulações |
| ENDP | Moderada | Distribuído | Localmente sincronizado | Utiliza um protocolo de enlace subjacente como IEEE 802.15.4 ou S-MAC. Avaliado com apenas um nó móvel em um campo de sensores estáticos. |
| MS-MAC | Parcial | Distribuído | localmente sincronizado | |
| IMEP | Sim | Não se aplica | Não se aplica | Protocolo da camada 3. Requer outro no nível de enlace. |
| 3M | Sim | Distribuído | Não | Apropriado a ambientes com alta mobilidade. |

Tabela 3.2: Alguns protocolos de enlace e de descoberta de vizinhos para RSSF

## 3.5 3M: *Multichannel Mobile MAC*

Conforme apresentado no Capítulo 1 e como será visto ao longo deste trabalho, a proposta desta tese depende do conhecimento do conjunto de vizinhos, por parte do nó, em um dado momento. É necessário também que um par de nós em uma vizinhança possa iniciar uma transação de migração de um agente móvel a qualquer momento.

Na Seção 3.4 é apresentado um estudo de alguns protocolos de enlace e descoberta de vizinhos para RSSF. Um resumo encontra-se na Tabela 3.2. Neste estudo, constata-se a inadequação dos mesmos aos ambientes de alta mobilidade de que trata esta tese.

Diante destes fatos, foi necessário desenvolver um novo protocolo de enlace para RSSF com alta mobilidade aleatória dos nós.

Trata-se de um protocolo que aloca e libera canais de freqüência sob demanda (***M**ultichannel*). Ele destina-se a ambientes com alto grau de mobilidade dos nós (***M**obile*). Como dito, trata-se de um protocolo de enlace, descoberta e manutenção de vizinhos (**M**AC). O protocolo foi denominado "3M": *Multichannel Mobile MAC* [58].

O funcionamento básico é apresentado a seguir.

Existe um canal de freqüência, pré-definido, chamado **canal de controle**, usado para anúncio de presença, anúncio de localização por parte das sementes e seleção de outros canais para comunicação em *unicast*. Todos os nós da rede enviam quadros de anúncio de presença periodicamente. Antes de enviar estes quadros, o nó "escuta" o canal. Estando ele disponível, o nó aguarda um período de tempo aleatório e envia seu anúncio de presença para todos os demais (*broadcast*). Todos os nós recebem e processam estes anúncios e incluem o emissor e mais alguns dados em sua tabela de vizinhos. Estas entradas na tabela de vizinhos são excluídas após um determinado período de tempo. Caso um novo anúncio de presença do mesmo emissor chegue, o contador de tempo para a exclusão do vizinho é reinicializado. Este canal de controle é utilizado em um esquema de melhor esforço. Não há garantia de entrega, reconhecimentos (ACKs) ou alocação prévia do canal.

Se o canal estiver ocupado, o nó espera um tempo aleatório e avalia novamente a disponibilidade do canal.

A necessidade de comunicação em *unicast* com um vizinho inicia um processo de alocação de um canal específico, chamado de **canal de trabalho**. O iniciador da comunicação envia, pelo canal de controle e em *unicast*, uma mensagem de alocação de canal. Devido ao fato do canal de controle funcionar com melhor esforço, pode-se optar por enviar mais de uma mensagem de alocação de canal, para uma maior probabilidade de recebimento. Imediatamente após receber esta mensagem, o destinatário ajusta seu transceptor para operar na nova freqüência. Após enviar sua última mensagem de alocação de canal, o iniciador também ajusta seu rádio para o mesmo canal. A partir daí, estes nós podem se comunicar com exclusividade nesta faixa de freqüência. Se o iniciador não obtiver resposta às suas mensagens no canal de trabalho, identificará que houve uma falha. A comunicação no canal exclusivo pode falhar por alguma destas razões: perda de todas as mensagens de alocação de canal; falha do nó destino; ou o nó destino saiu do alcance do emissor após a transação de ajuste de canal. Caso o iniciador identifique a impossibilidade de comunicação por uma destas razões, duas ações podem ser tomadas. Ele pode retornar ao canal de controle, consultar novamente a lista de vizinhos e decidir por enviar novas mensagens de alocação de canal, caso o destinatário ainda seja considerado vizinho. Outra opção é executar novamente o algoritmo do nível de aplicação que gerou a necessidade de comunicação no canal exclusivo.

Observa-se que o protocolo pode atender a diversas aplicações e caberá a elas definir o canal exclusivo a ser usado. Na presente proposta, o canal será usado para a migração de agentes móveis. Uma consequência importante neste caso é que o próprio agente pode ser injetado na rede já com o *seu* canal de serviço, eliminando a possibilidade de colisão no mesmo. Uma alternativa, por falta de canais disponíveis, é a injeção de mais de um agente para cada canal. Neste caso, haverá a possibilidade de colisão quando mais de um agente móvel estiver usando o mesmo canal de trabalho, na mesma região de cobertura. Trata-se de um compromisso entre a quantidade de canais e a quantidade de agentes móveis operando na rede. Um mecanismo secundário de acesso ao meio nos canais de trabalho pode ser implementado e está além dos objetivos atuais deste trabalho.

O protocolo pode ser facilmente estendido para canais de trabalho que operem com outras técnicas. Por exemplo, ajustando um código específico (*chipping*

*code*) para acesso múltiplo por divisão de código (CDMA - *Code Division Multiple Access*)[59, 60] ou uma seqüência de salto em freqüência específica (FHSS - *Frequency-hopping Spread Spectrum*[61] ou DSSS *Direct-sequence spread spectrum*[61]).

Nota-se que não há qualquer menção a *slots* de tempo, sincronização local ou global, ou à necessidade de qualquer recurso adicional de protocolos de outras camadas. Observa-se ainda que o protocolo é totalmente descentralizado, produzindo um conhecimento local de vizinhança e alocação de recursos para comunicação direta entre dois nós, sem a existência de qualquer entidade centralizadora ou coordenadora.

No estudo apresentado ao longo da Seção 3.4, foram identificadas características necessárias ao protocolo de enlace apropriado para ambientes com alto grau de mobilidade. Os mesmos são relacionados a seguir:

1. **Não deve depender de endereços das camadas superiores**. Como o ambiente altamente dinâmico impõe maior complexidade a processos em geral. Considerar informações disponíveis em outros níveis da pilha de protocolos, para simplificar ou viabilizar o nível de enlace, apenas desloca a complexidade, não solucionando-a;

2. **Deve ser totalmente descentralizado, mesmo no nível de uma vizinhança local (PAN, célula, etc)**. O ponto de acesso ou o coordenador local também são móveis. A presença destas entidades centralizadoras demandarão constantes transações de renovação e atualização na rede;

3. **Não deve considerar sincronismo local e, muito menos, global**. Diante das necessidades de execução de algoritmos de localização, da manutenção do conhecimento dos vizinhos e da tarefa de sensoriamento em si, manter uma rede inteira sincronizada será um peso ainda maior para o esquema como um todo. As dificuldades de sincronismo global são de tal forma relevantes que poucos protocolos de redes sem fio trabalham com esta premissa. O sincronismo local, na forma de alinhamento de *slots* de tempo na célula, na subrede local ou na PAN, sofre do mesmo problema imposto pela entidade centralizadora. Atualizações e ressincronizações serão necessárias freqüentemente, sobrecarregando a rede e impondo atrasos às demais comunicações ligadas aos

objetivos das RSSF;

4. **Deve permitir a descoberta e manutenção do conjunto de vizinhos**. Como este conjunto varia muito nas RSSF com alta mobilidade, esta torna-se uma tarefa muito importante. Ela deve ser implementada com baixa latência e baixo consumo de recursos de comunicação. Seus requisitos diferem dos das redes estáticas onde este conjunto, uma vez formado, sofre poucas variações futuras;

5. **Deve implementar mecanismos de acesso ao meio que minimizem as colisões e maximizem a vazão**;

6. **Deve maximizar os períodos de desligamento (inatividade) do transmissor de rádio, buscando reduzir o consumo de energia**. Isto equivale a reduzir o ciclo de trabalho (*duty cycle*) do rádio;

7. **Deve evitar o problema do terminal oculto**;

8. **Deve minimizar a escuta ociosa**[4] **do canal (*idle listenning*)**. Busca a redução do consumo de energia.

O protocolo 3M atende a todos os requisitos citados, à exceção dos dois últimos. O quinto é atendido indiretamente, pelo atendimento do sexto. Períodos de transmissão curtos e espaçados reduzem a probabilidade de colisões.

No caso do penúltimo, o problema existe apenas no canal de controle, inexistindo nos canais de trabalho. Os conceitos de canal de controle e de trabalho serão apresentados mais adiante.

Quanto à escuta ociosa, o protocolo 3M pressupõe escuta permanente do canal de controle. O custo energético da escuta ociosa é inferior ao da recepção de dados, que, por sua vez, é inferior ao da transmissão de dados. Entretanto, esta é uma característica negativa do protocolo e existem outras propostas que implementam a camada de enlace com períodos de inatividade também na escuta do canal.

Esta característica do protocolo 3M é conseqüência da mobilidade constante dos nós. Caso houvessem períodos de inatividade na escuta do canal, um certo

[4]Escuta ociosa ocorre sempre que o nó mantem seu receptor ativo, pronto a receber transmissões, mesmo quando nenhuma está em curso.

número de vizinhos não seria detectado, o que prejudicaria todo o funcionamento das aplicações e da rede como um todo.

Como proposto na Seção 3.2, o algoritmo de localização será baseado em sementes que farão a difusão de informações de localização, de modo que os nós possam estimar suas respectivas posições. Como será visto adiante, os anúncios feitos por estas sementes também utilizarão o canal de controle. Este fato é uma justificativa a mais para a escuta do canal de controle durante 100% do tempo.

A seguir é apresentado o funcionamento do protocolo.

### 3.5.1 O canal de controle

A mobilidade faz com que o conjunto de nós em uma dada região seja muito dinâmico. Um determinado nó pode se deslocar por toda a rede. Configurações como códigos CDMA, padrões de espalhamento de espectro, canal de freqüência a ser usado ou sincronismo de *slots* para TDMA não devem ser aplicados nestes casos.

No protocolo aqui proposto, todos os nós da rede utilizam um canal de controle pré-definido. Todos os nós da rede monitoram (escutam) este canal durante todo o tempo.

Todas as mensagens que trafegam no canal de controle são de tamanho muito reduzido (*beacons*).

Este canal é utilizado para o envio de 3 mensagens distintas, conforme a Tabela 3.3.

### 3.5.2 Mensagem de anúncio de presença

Todos os nós da rede devem emitir, em *broadcast* e com uma certa periodicidade, um *beacon* anunciando sua presença em uma dada vizinhança.

O formato da mensagem de anúncio de presença e o tamanho dos campos em *bytes* estão na Figura 3.5, onde tem-se:

- O SHR, PHR e PHY *payload* juntos formam o pacote físico. É tudo o que é transmitido. O tamanho total é de apenas 38 bytes;
- A carga do pacote físico, PHY *payload*, é formada pelo quadro MAC;

| Mensagem | Origem | Destino | Tipo | Descrição |
|---|---|---|---|---|
| Anúncio de presença | Todos os nós | Todos os nós | *broadcast* | Todos os nós enviam periodicamente um *beacon* anunciando sua presença |
| Localização | Sementes | Todos os nós | *broadcast* | Informações de localização difundidas pelas sementes |
| Sintonia de canal | Nó hospedeiro | Nó destino | *unicast* | Sintoniza um canal de freqüência para uso das camadas superiores (por exemplo, a migração de um agente móvel) |

Tabela 3.3: Mensagens do canal de controle do protocolo 3M

- O quadro MAC é formado pelo cabeçalho (MHR), carga útil (MAC *payload*) e rodapé (*footer*) (MFR).

| | | | PHY *payload* | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| SHR | | PHR | MHR | | | MAC *payload* | | | MFR |
| *Preamb.* | SFD | FL | C | S | D | SL | MT | L | FCS |
| 4 | 1 | 1 | 1 | 8 | 8 | 4 | 1 | 8 | 2 |

Figura 3.5: Mensagem de anúncio de presença

Tanto quanto possível, procurou-se manter a compatibilidade com o Padrão IEEE 802.15.4. O detalhamento dos campos é apresentado a seguir:

1. ***Preamble* (preâmbulo):** o campo preâmbulo é usado pelo transceptor para obter sincronismo com um quadro a ser recebido, conforme Cláusula 6.3.1 do Padrão IEEE 802.15.4;

2. **SFD (*Start-of-frame Delimiter*):** Indica o fim do SHR (*Synchronization Header*) e início do quadro de dados, conforme Cláusula 6.3.2 do Padrão IEEE 802.15.4;

3. **FL (*Frame Length*):** Contém o tamanho do quadro contido no restante dos dados a serem transmitidos, ou seja, o tamanho do PHY *payload*[5], conforme Cláusula 6.3.3 do padrão;

[5] *Payload*, ou carga útil, em protocolos de comunicação, refere-se ao dado real sendo transmitido.

4. **C:** O campo de controle não é utilizado. Foi mantido por coerência com o IEEE 802.1.5.4 e pode conter dados referentes aos dados do quadro, tipo de quadro, questões relativas a segurança e versão. No padrão, estas informações estão na Cláusula 7.2.1.1 e Tabela 79;

5. **S (*Source*):** Endereço MAC de origem do quadro, conforme Cláusula 7.2.1. Diferente do Padrão 802.15.4, que prevê endereços de 64 bits, ou endereços curtos de 16 bits. O protocolo 3M especifica o uso único do endereço MAC de 48 bits;

6. **D (*Destination*):** Endereço MAC de destino. Neste caso, o endereço de *broadcast* (FF-FF-FF-FF-FF-FF na base hexadecinal). A seguir estão os campos com os dados propriamente ditos;

7. **SL (*Slope*):** Contém o coeficiente angular da direção do deslocamento do nó. Este dado vem na forma de um número real, codificado segundo o Padrão IEEE 754-2008 [62];

8. **MT (*Movement type*):** Este campo informa o tipo de movimento do nó. Este dado corresponde a um caracter e é utilizado no processo de decisão dos agentes móveis que é apresentado no Capítulo 4;

9. **L (Localização):** Neste campo estão as duas coordenadas da localização bidimensional do nó. Os valores estão em ponto flutuante, segundo o mesmo padrão do campo SL;

10. **FCS (*Frame Check Sequence*):** A fim de detectar erros de bits, um mecanismo de verificação é aplicado a cada quadro. Trata-se da aplicação de um código polinomial de redundância cíclica (CRC - *Cyclic Redundancy Check*) de 16 bits. Mais detalhes podem ser encontrados na Cláusula 7.2.1.9 do padrão e em Tanenbaum 2003, Seção 3.2.2, p. 208 [47].

---

Ele é precedido por um cabeçalho que identifica o transmissor e o receptor (à exceção da camada física que não contém endereçamento) e, opcionalmente, outros campos de controle. Cada vez que um pacote é enviado para a camada superior do modelo ISO/OSI, este cabeçalho é retirado e apenas a carga útil é entregue. Este processo é chamado de encapsulamento e pode ocorrer várias vezes sobre a mesma carga útil.

### 3.5.3 O algoritmo de descoberta e manutenção de vizinhos

Os nós enviam periodicamente em *broadcast* um *beacon* de anúncio de presença. Não existe reconhecimento, tratamento de colisões e prevenção quanto a terminais ocultos.

Os anúncios recebidos são processados e aqueles irrecuperavelmente corrompidos, ou vítimas de colisão, são descartados.

#### Intervalo entre anúncios

Cada nó controla uma variável chamada ***beaconInterval***, que determina o intervalo com que um anúncio de presença deve ser enviado.

Este intervalo é relativamente grande, se comparado ao tempo em que o canal ficará ocupado com a mensagem enviada. Como será visto mais adiante, as simulações feitas com valores do ***beaconInterval*** entre $0,5$ e $0,1s$ produzem bons resultados.

Cada vez que o intervalo expira, o nó inicia o processo de envio do anúncio de presença. Somente quando esta transmissão estiver concluída, o intervalo de anúncio é restaurado ao valor original, para que nova espera seja iniciada.

Como visto na Seção 3.5.2, a mensagem de anúncio de presença tem 38 *bytes*. Operando a uma taxa de 250 *kbps*, uma das taxas recomendadas pelo Padrão IEEE 802.15.4, a ocupação do canal (*duty cycle*) varia de $0,24\%$ a $1,22\%$, para os intervalos de *beacon* citados anteriormente. Estes valores são eficientes em termos de consumo de banda e energia.

#### Acesso ao canal de controle

Uma vez expirado o intervalo de anúncio, o nó verifica se o canal está disponível, ou seja, o nó escuta o canal para verificar se há alguma transmissão em curso.

Caso o canal esteja livre, o nó inicia o decréscimo de outro temporizador, chamado *backoff*. Quando o *backoff* chega a zero, o canal inicia a transmissão. Este procedimento faz com que os diversos nós que estejam aguardando a disponibilidade do meio de transmissão não iniciem as transmissões todos juntos, logo que o meio se torne livre.

A variável *backoff* de cada nó é gerada de forma aleatória, dentro de uma faixa pré-estabelecida. Existe um parâmetro chamado *backoffBase*, sendo o *backoff* um

múltiplo aleatório deste parâmetro, segundo a Equação 3.5, onde $k$ é um número inteiro aleatório.

$$backoff = backoffBase \times k \quad (k \in \mathbb{N}^*) \tag{3.5}$$

Na linguagem Java [63, 64], na qual foi escrito o simulador, para $0 < k \leq 10$, temos:

**backoff = backoffBase * Util.geraInteiro(10);**

Caso o canal seja detectado como ocupado já na primeira escuta, o mesmo temporizador *backoff* citado nos parágrafos anteriores é disparado.

Durante o período de decréscimo da variável *backoff*, a cada decremento, o canal é verificado novamente (lembrando que o evento "canal livre" iniciou a fase de *backoff*). A qualquer momento em que o canal for detectado como ocupado, a variável *backoff* é gerada novamente, segundo a Equação 3.5.

Quando ocorre o acesso ao meio e a transmissão propriamente dita, um novo valor é atribuído ao *backoff* também pela Equação 3.5.

Existem protocolos que utilizam o algoritmo de *backoff* exponencial [50, 65]. Este algoritmo prevê que a cada detecção de colisão ou detecção de canal ocupado, o *backoff* deve aumentar com relação ao valor anterior, dentro de uma janela máxima. Continua havendo o comportamento aleatório, mas com intervalos de espera maiores a cada vez que não se consegue acesso ao meio ou que a transmissão falha.

No protocolo 3M, não há detecção de colisão nem detecção de perda por não reconhecimento (ACKs). Utiliza-se apenas um espalhamento no tempo, com uma aleatoriedade monótona, ou seja, a cada detecção de canal ocupado, um novo *backoff*, dentro da mesma faixa, é computado (podendo ser menor, igual ou maior do que o anterior).

O Algoritmo 1 mostra a lógica do envio de anúncios de presença.

### Manutenção do conjunto de vizinhos

Sempre que um anúncio de presença é recebido com êxito, ele é processado pelo nó. O endereço MAC é utilizado como identificador único dos nós (ver Seção 3.4). Caso o quadro venha de um nó até então desconhecido, este nó é inserido na lista

```
Algoritmo: processaBeacon
beaconIntervalCounter = 0
remainingSteps = numberOfStepsToTransmitABeacon //Número de ciclos do
                                    //simulador necessários para o envio de um beacon
backoff = backoffBase * k //k é um inteiro aleatório maior que zero
enquanto Durar a simulação faça
    se (beaconIntervalCounter == beaconInterval) então
        se Canal está disponível então
            se (backoff == 0) então
                Envia beacon
                se (remainingSteps == 1) então
                    beaconIntervalCounter = 0
                    remainingSteps = numberOfStepsToTransmitABeacon
                    backoff = backoffBase * k
                senão
                    remainingSteps - -
            senão
                backoff - -
        senão
            backoff = backoffBase * k
    senão
        beaconIntervalCounter++
```

**Algoritmo 1**: Anúncio de presença

de vizinhos conhecidos e um tempo de vida (TTL) é atribuído a esta entrada. Caso o vizinho já seja conhecido, ou seja, caso já conste da lista de vizinhos conhecidos, o TTL é restaurado ao valor padrão.

Periodicamente, o nó varre o seu conjunto de vizinhos, decrementando o TTL de todas as entradas e excluindo do conjunto aquelas com valor zero.

### 3.5.4 Sintonia do canal de trabalho

Segundo Kohvakka *et al.* [16], com altas taxas de transmissão a probabilidade de colisões entre *beacons* e quadros de dados é significativa. Deste modo, o uso de múltiplos canais na rede é factível e justificável. Os autores fizeram experimentos com os transceptores *Nordic Semiconductor* nRF24L01 e nRF2401A [66] operando

na faixa de $2,4\,GHz$ com 83 canais de 250 $kbps$ ou 1 $Mbps$. Nestes experimentos, foram observados 21 canais ortogonais (sem interferência). Os autores ressaltam que o rádio pode ser sintonizado muito rapidamente para uma nova freqüência.

Para Zhou *et al.* [67], uma vez que o sensores atuais utilizam de forma muito limitada uma única banda de freqüência, $19.2Kbps$ no MICA2 [68] e $250Kbps$ no MICAz [69] e Telos [70], é imperativo o projeto de protocolos da camada MAC que utilizem múltiplos canais. Estes protocolos poderão obter uma maior vazão através de comunicações paralelas. Além disso, o rádio CC2420 [71] utilizado no MICAz e nos sensores Telos, já prevê múltiplos canais físicos, abrindo caminho para o projeto desta família de protocolos.

Le *et al.* [72] desenvolveram e avaliaram um protocolo de alocação dinâmica de múltiplos canais. Segundo os autores, o uso de apenas uma banda de freqüência é fonte de ineficiência.

Padrões de comunicação sem fio, como o 802.11 e o 802.15.4, consideram que as estações podem migrar de um canal para outro, o que demonstra a capacidade dos dispositivos de suportarem variações na freqüência de operação.

Segundo Conti e Giordano [73], a baixa escalabilidade das redes *ad hoc* tem gerado muitas pesquisas do uso de múltiplas interfaces de rádio, antenas direcionais e MIMO[6], além de múltiplos canais, para um melhor aproveitamento do espectro e redução das interferências.

Sohrabi *et al.* consideram a alocação de um novo canal logo após o estabelecimento do enlace por um par de nós. Considera-se, no referido trabalho, que o novo canal é escolhido de forma aleatória, dentre um numeroso conjunto de opções. Os autores consideram a farta disponibilidade de canais segundo o seguinte exemplo: com rádios operando na faixa ISM (Industrial, Scientific, and Medical) de 902 a $928MHz$ e taxas de $10kbps$, tem-se cerca de 2600 bandas de freqüência distintas disponíveis.

O protocolo de enlace proposto nesta tese não é específico para as aplicações de agentes móveis que serão apresentadas no Capítulo 4. Entretanto, como será visto logo adiante, a aplicação de agentes móveis facilita muito a escolha do canal

[6] *Multiple-Input and Multiple-Output* é o uso de múltiplas antenas tanto no transmissor quanto no receptor, para melhorar o desempenho das comunicações.

de trabalho a ser usado.

Devido às razões apresentadas nesta seção e motivados também pelas tarefas já alocadas ao canal de controle, o projeto do protocolo 3M considera o uso de outros canais para as comunicações de trabalho da rede. A seguir é apresentado o funcionamento.

Como consequência de alguma decisão das camadas superiores, um nó pode decidir iniciar uma transação em *unicast* com um ou mais vizinhos.

Neste caso, o nó enviará a estes vizinhos uma ou mais mensagens de sintonia de canal. O envio de mais de uma mensagem se destina a aumentar a probabilidade de recebimento pelo(s) destinatário(s).

Esta mensagem disputa o canal de controle e tem o mesmo formato do *beacon* de anúncio de presença, à exceção do MAC *payload* que contém a identificação do canal a ser usado. Observa-se que esta mensagem será ainda menor que o anúncio de presença, já que o canal a ser usado pode ser identificado por um número inteiro, ocupando 2 *bytes* do quadro. Após o envio da mensagem de sintonia, o nó chaveia seu rádio para o canal escolhido e envia, neste canal, uma sonda (*probe*) para verificar se o correspondente já está apto a operar no canal de trabalho escolhido.

Não existe reconhecimento (ACK) para mensagens de sintonia. Caso um nó proponha uma comunicação com outro em um dado canal, se, ao chavear para este canal, detectar que o correspondente não está presente, caberá à aplicação tratar este fato. A ausência do correspondente no canal alocado pode se dar por três motivos: ***a)*** a mensagem de sintonia foi perdida (o canal de controle trabalha com melhor esforço); ***b)*** o correspondente saiu do raio de alcance do nó proponente (alto grau de mobilidade); ***c)*** falha do nó destino.

Uma alternativa é retornar ao canal de controle e enviar nova ou novas mensagens de sintonia. Outra opção é executar a rotina do nível de aplicação que deu origem à decisão de comunicação com o vizinho em questão. Pode ser que, em caso de desconexão, o novo resultado não mais envolva o mesmo vizinho.

O uso de reconhecimento (ACK) para a mensagem de sintonia pode ser considerado, apesar de não ter sido avaliado neste trabalho. Ele fará com que o chaveamento para o canal de trabalho só se dê com a concordância do correspondente. O custo é uma sobrecarga no canal de controle. No caso do uso do ACK, o mesmo deve

ter o menor tamanho possível e deve utilizar um *backoff* menor do que as demais mensagens para obter maior prioridade no acesso ao meio. Neste caso, ressalta-se que mensagens importantes de localização oriundas das sementes podem ser prejudicadas. Um fator a ser considerado nesta decisão de uso ou não do ACK é a freqüência e duração do uso dos canais de trabalho.

Caso outra técnica seja usada no canal de trabalho, mais informações podem ser enviadas no quadro e o campo controle (ver Figura 3.5) pode especificar a tecnologia a ser utilizada.

No caso da aplicação em execução na rede ser baseada em agentes móveis, surge uma possibilidade bastante interessante. O agente móvel, ao ser injetado na rede, já tem a si atribuído o "seu" canal de trabalho. Assim, ao decidir executar uma migração para um nó vizinho, uma requisição de envio de uma mensagem de sintonia de canal é feita ao protocolo de enlace. Nesta requisição, a aplicação, no caso o agente móvel, informa o seu canal de trabalho. Isto garante que este canal será único na rede e, conseqüentemente, não haverá colisão quando do seu uso.

Caso o número de canais disponíveis seja menor do que o número de agentes que se deseja injetar na rede, ainda é possível uma sobreposição. Neste caso, emergirá um problema probabilístico relacionado à localização dos agentes, necessidade de migração naquele momento e coincidência do canal de trabalho do agente. Dependendo destes parâmetros, um certo grau de sobreposição pode ser aceitável. Pode-se ter ainda a adaptabilidade do agente a um novo canal, em tempo de execução, diante da utilização de canais de trabalho encontrada na rede. Estas questões não serão analisadas com mais profundidade e ficam relacionadas como trabalhos futuros.

### 3.5.5 Avaliação de desempenho

#### Preliminares

Vizinhos reais (**VR**) são aqueles que, em um dado momento, estão no raio de cobertura do rádio de um nó. Este conjunto é fornecido pelo simulador, sempre que necessário. No exemplo da Figura 3.6, para o nó $X$, temos:

$$VR_X = \{A, B, C, D, E\}$$

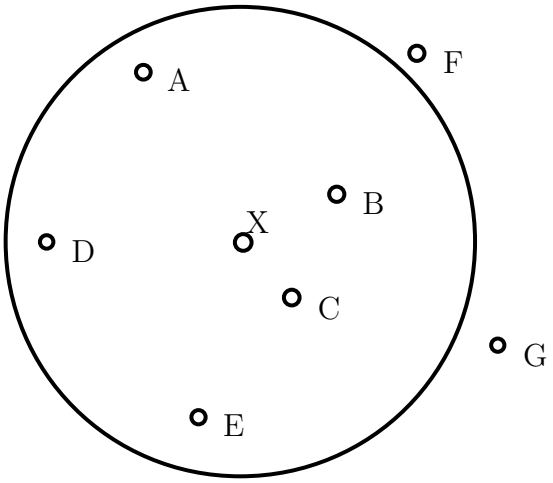



Figura 3.6: Exemplos de vizinhos reais e conhecidos

O conjunto de vizinhos conhecidos (**VC**) é computado pelo protocolo 3M em execução em todos os nós, durante toda a simulação. Na Figura 3.6, para o nó $X$, pode-se ter:

$$VC_X = \{B, C, D, E, F\}$$

Neste caso, o nó $F$ representa um falso positivo, ou seja, ele consta da lista de vizinhos conhecidos do nó $X$, mas, na verdade, está fora do alcance de rádio deste. Este conjunto dos falsos positivos é chamado de conjunto dos vizinhos conhecidos não reais (**VCNR**).

Por outro lado, o nó $A$ é um vizinho real, mas é desconhecido por $X$, fazendo parte do conjunto de vizinhos reais desconhecidos (**VRD**).

Estes dois conjuntos se constituem em erros do protocolo e devem ser minimizados. Na situação ideal tem-se:

$$VC_X \equiv VR_X$$

e

$$VCNR_X = VRD_X = \varnothing$$

Tem-se ainda, contido ou igual ao conjunto VR, o conjunto dos vizinhos reais conhecidos (**VRC**).

$$VRC_X = VR_X \cap VC_X$$

O chamado conjunto de vizinhos não reais (**VNR**), com o perdão do termo, contém todos os nós que estão fora do alcance de comunicação de um nó. No exemplo da Figura 3.6 tem-se então:

$$VNR_X = \{F, G\}$$
$$VCNR_X = VC_X - VRC_X = \{B, C, D, E, F\} - \{B, C, D, E\} = \{F\}$$
$$VRD_X = VR_X - VRC_X = \{A, B, C, D, E\} - \{B, C, D, E\} = \{A\}$$

**As métricas utilizadas para análise das simulações** foram a cardinalidade dos conjuntos VRC e VCNR, além do percentual de colisões entre mensagens de anúncio de presença.

Como a aplicação pode demandar o conjunto de vizinhos a qualquer momento, optou-se por amostrar o estado dos conjuntos de interesse no sistema a cada décimo de segundo.

Para se obter um índice de sucesso do protocolo, calculou-se, com base nos conjuntos VRC e VR, o percentual de elementos de VR que estão em VRC (Equação 3.6).

$$VRC(\%) = \frac{|VRC|}{|VR|} \times 100 \tag{3.6}$$

Para os falsos positivos, calculou-se o percentual de VCNR sobre a soma de elementos de VRC com VCNR, ou seja, a proporção dos erros do protocolo sobre o total de vizinhos conhecidos (Equação 3.7).

$$VCNR(\%) = \frac{|VCNR|}{|VCNR| + |VRC|} \times 100 = \frac{|VCNR|}{|VC|} \times 100 \tag{3.7}$$

Para o exemplo dado tem-se:

$$VRC(\%) = \frac{|VRC|}{|VR|} \times 100 = \frac{4}{5} \times 100 = 80\%$$
$$VCNR(\%) = \frac{|VCNR|}{|VC|} \times 100 = \frac{1}{5} \times 100 = 20\%$$

Os valores do exemplo dado não precisam totalizar 100%, como ocorreu. Bastaria existir mais um elemento em VCNR para que o resultado percentual VCNR(%) fosse 33%, sem afetar o valor de VRC(%).

### Os cenários e parâmetros das simulações

Para avaliar o protocolo proposto foi utilizado o simulador desenvolvido, descrito no Anexo A. O nível físico de transmissão foi considerado ideal, sem perdas, desvanecimento ou interferências.

Petrova *et al.* [74] mostram, para a camada física do Zigbee (802.15.4), que a taxa de erros para pequenas distâncias é muito reduzida, se mantendo abaixo de 1% para distâncias de até $20m$. Este patamar de 1% também é recomendado pelo Padrão IEEE 802.15.4.

De qualquer modo, já estando implementadas as camadas de aplicação e de enlace do simulador, a introdução de modelos mais reais na camada física é importante e estes serão desenvolvidos no futuro.

As simulações foram feitas com os seguintes cenários e parâmetros:

- Número de nós: 50;
- Duração: $100s$;
- Raio de alcance do rádio: $10m \Rightarrow Área \bigcirc \approx 314m^2$;
- Densidade de nós pretendida: 5 nós por área de cobertura $\Rightarrow$ $5\,nós/\,314m^2 \approx 1\,nó/63m^2 \approx 0,016\,nó/m^2$;
- Área total necessária: $50 \times 63 = 3.150m^2 \approx 56m\ X\ 56m$;
- Mobilidade: Os cenários de mobilidade foram criados com o aplicativo *setdest* que acompanha o simulador *ns-2* [75]. Para uma área, um número de nós e um tempo de duração pré-definidos, esta aplicação gera aleatoriamente uma coordenada inicial (passo 1), uma coordenada de destino (passo 2) e uma velocidade de deslocamento entre estas duas coordenadas (passo 3). Ao chegar à coordenada de destino, é gerado aleatoriamente um intervalo de repouso para o nó (passo 4). Em seguida, o processo se repete dos passos 2 a 4, até o término do período de duração definido. Este modelo de mobilidade é chamado *Random Waypoint*. Uma discussão do método e algumas referências estão na página *Random Waypoint Model* [76].

  - O número de nós, área e duração da simulação foram definidos como descrito anteriormente. Outras simulações foram feitas com densidades em torno de 11 e 18 nós por vizinhança (por área de cobertura);
  - Velocidade: distribuição normal entre 1 e 30 $m/s$;
  - Períodos de pausa: distribuição uniforme entre 0 e 20$s$, 0 e 4$s$, e entre 0 e $0,2s$, conforme o caso;
  - Velocidade média dos nós (em função dos períodos de pausa): $2,33\,m/s$, $6,41\,m/s$ e $10,99\,m/s$, respectivamente.

- Intervalo de envio de mensagens de anúncio de presença (*beaconInterval*): variou de $0,5$ a $0,1s$;
- Tamanho da mensagem de anúncio de presença (*beacon*): 38 *bytes*, de acordo com a especificação do protocolo;
- Taxa de transmissão do canal de controle: 250 *kbps*;
- Base de cálculo do *backoff*: $0,0001s$;
- Multiplicador aleatório para cálculo do *backoff*: um número inteiro escolhido aleatoriamente no intervalo $[1, 10]$ (segundo uma distribuição uniforme), ou seja, o *backoff* varia de $0,0001$ a $0,001\,s$;
- O relógio do simulador foi ajustado para 4 casas decimais (avança em intervalos de $0,0001s$). Este valor é suficiente em termos de precisão. Como são consumidos $0,001216\,s$ para transmissão de um *beacon*[7], serão consumidos 13 ciclos do simulador para transmitir cada *beacon*;
- TTL das entradas na tabela de vizinhos: variou de 50 a 160% do valor do *beaconInterval*.

Nos resultados descritos a seguir, apenas os parâmetros de entrada que podem assumir mais de um valor serão mencionados. Os demais permanecem com os valores citados na relação geral de parâmetros.

[7] O *beacon* tem 38 *bytes*. $38\,bytes = 38 \times 8 = 304\,bits$. A uma taxa de $250\,Kbps = 250.000\,bps$, o tempo gasto é dado por $\frac{304}{250.000} = 0,001216s$

**Resultados das simulações**

Uma primeira rodada de simulações foi executada com os parâmetros mencionados anteriormente (parâmetros padrão) e com as seguintes variações: período de pausa do nó distribuído entre 0 e 20$s$ (o que produz uma velocidade média de $2,33\,m/s$); TTL variando de $0,5$ a $0,75\,s$.

A média de vizinhos ao longo da simulação foi de $4,3$. Adicionando o próprio nó, o valor se aproxima da meta de densidade de nós pretendida.

Deve-se observar que a velocidade média leva em conta os períodos de pausa. Quando em movimento, as velocidades estarão entre a faixa de 0 a $30\,m/s$. Pode-se ter, por exemplo, um nó se movendo a $25\,m/s$, o que equivale a $90\,km/h$. Levando-se em conta ainda a mobilidade relativa entre dois nós, temos um cenário de alta mobilidade, o que exige muito do algoritmo de descoberta e manutenção de vizinhos.

O protocolo obteve uma média de $91,47\%$ de sucesso no conjunto VRC (Equação 3.6), com intervalo de confiança de $0,5\%$. A variação do TTL acima do valor do *beaconInterval* praticamente não afetou o conjunto VRC.

Para uma ocupação do canal da ordem de apenas $0,24\%$[8] e com escuta permanente do canal, obteve-se mais de 90% de sucesso no conjunto de vizinhos reais conhecidos.

Os resultados, para o mesmo cenário, com relação aos falsos positivos, são mostrados na Figura 3.7. Como esperado, com o aumento do TTL das entradas na tabela de vizinhos conhecidos, o erro aumenta.

Ao final deste lote de simulações, três constatações foram feitas. A primeira foi que longos períodos de pausa reduzem o grau de mobilidade e são favoráveis ao funcionamento do protocolo.

A segunda se refere ao TTL. Pelo funcionamento do 3M, não conhecer um vizinho real é menos danoso do que conhecer um vizinho "não real". Um nó pode iniciar uma transação de sintonia de canal com qualquer vizinho conhecido. Desconhecer um vizinho real apenas reduz o universo de escolha da aplicação. No caso de uma migração de um agente móvel, por exemplo, pode até ser que este vizinho desconhecido não fosse a melhor escolha, o que torna irrelevante a desinformação. Já o falso positivo, pode desencadear um processo custoso de envio de mensagem de

[8] Um *beacon* de $0,001216\,s$ a cada $0,5\,s$

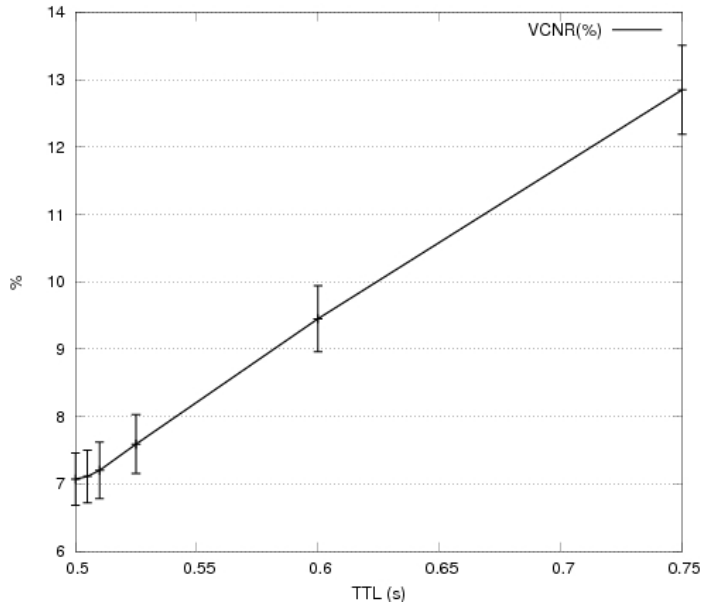



Figura 3.7: Falsos positivos do protocolo 3M para mobilidade com pausas longas

sintonia, ajuste do rádio, envio da sonda, temporização e eventual retorno ao canal de controle para reiniciar o processo.

A terceira se refere à densidade de nós relativamente baixa. Como a ocupação do canal requerida pelo protocolo é baixa, um cenário com maior densidade de nós deve ser simulado, para submeter o protocolo a uma maior ocupação do canal de controle.

Em função da primeira e terceira constatações mencionadas, novas simulações foram feitas com algumas variações.

Na primeira delas, os períodos de pausa foram reduzidos para a faixa de 0 a 4 $s$ (velocidade média de $6,41\,m/s$) e aumentou-se a densidade de nós para 11,2 e 18,6. Isto foi feito através do aumento do raio de comunicação dos nós para 15 e 20 $m$, respectivamente. O TTL foi mantido em $0,525\,s$.

Os resultados estão na Figura 3.8, onde se observa a robustez do protocolo diante do aumento da densidade de nós[9].

Levando em conta novamente a primeira constatação, novas simulações foram feitas com períodos de pausa praticamente nulos. As pausas ficaram entre 0 e $0,2\,s$, o que gera uma velocidade média de $10,99\,m/s$. Variou-se o TTL de $0,5$ a $0,8\,s$. O raio de comunicação foi de $10\,m$, o que retorna a densidade para a faixa inicial de

[9]Para $x = 4,3$ vizinhos, o valor refere-se ainda à pausa de 0 a 20 $s$

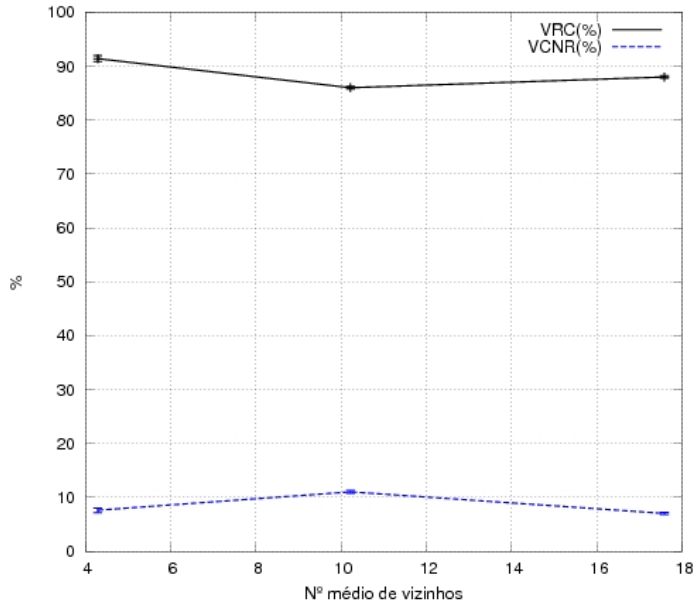



Figura 3.8: VRC e VCNR com variação da densidade de vizinhos

5 nós por área de cobertura. Na verdade, obervou-se que, com os nós se movendo praticamente o tempo todo, o valor médio de vizinhos nas simulações foi de $6,2$, o que leva à uma densidade de $7,2$ nós por vizinhança.

Não foi feita uma análise detalhada do código do programa *setdest*, mas análises dos cenários de mobilidade gerados levam à conclusão de que o *setdest* não explorou os limites extremos da área da simulação, próximos às bordas da área. Com uma maior mobilidade os nós se concentraram um pouco mais na área central do campo de simulação, levando à maior densidade nesta área.

O protocolo obteve uma média de $75,84\%$ de sucesso no conjunto VRC (Equação 3.6), com intervalo de confiança de $0,46\%$. Mais uma vez, valores do TTL maiores que o intervalo de *beacon* praticamente não afetaram o conjunto VRC.

Neste cenário de maior mobilidade, o desempenho do protocolo cai consideravelmente, deixando de identificar um quarto dos vizinhos. Neste caso, já se torna necessário diminuir o intervalo entre anúncios de presença, como será visto mais adiante.

A Figura 3.9 mostra que os falsos positivos já partem de um valor elevado para TTL de $0,5\ s$ e aumentam ainda mais com valores maiores de TTL. Além de mais anúncios de presença, valores menores de TTL são necessários, conforme a segunda constatação feita anteriormente.

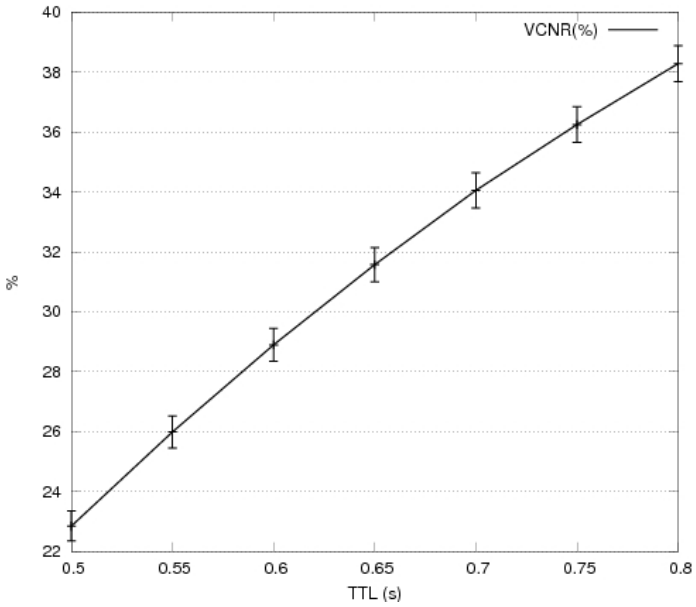



Figura 3.9: Falsos positivos do protocolo 3M. Mobilidade praticamente sem pausas

A Figura 3.10 mostra o desempenho do protocolo com o aumento da freqüência de anúncios de presença. Pode-se observar que com este aumento, mesmo com o alto grau de dinamismo dos nós, o protocolo obteve bons resultados. Existe, entretanto, um teto para o cenário simulado, em torno de $83,64\%$, para $beaconInterval = 0,2\,s$. Acima deste valor, como observa-se na Figura 3.11, as colisões aumentam, reduzindo o desempenho.

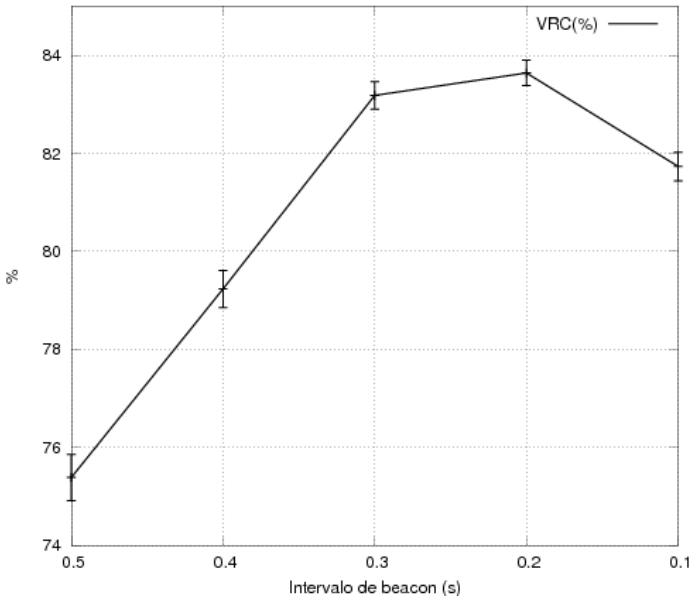



Figura 3.10: VRC para diversos intervalos de *beacon*

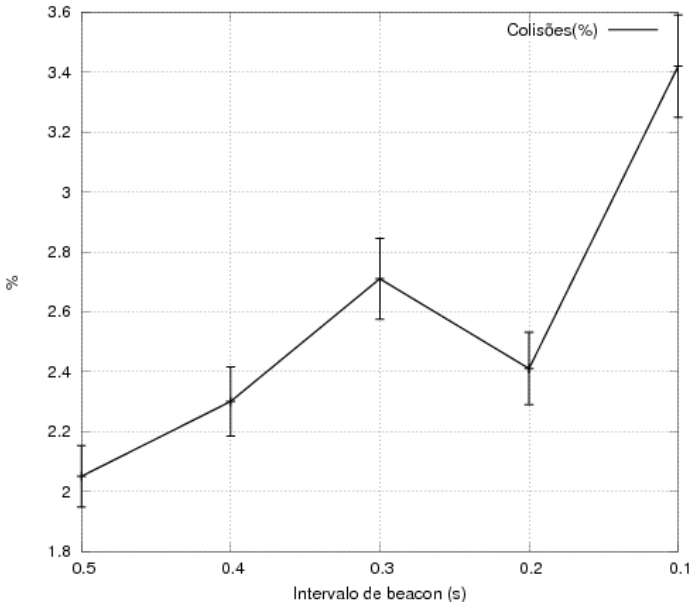



Figura 3.11: Percentual de colisões de quadros para diversos intervalos de *beacon*

Os últimos resultados a serem apresentados neste capítulo se referem ao TTL das entradas nas tabelas de vizinhos.

Pelos eventuais custos impostos pelos falsos positivos, já apresentados, foram feitos testes com valores de TTL menores que o intervalo de *beacon*. Trata-se de uma ação conservadora. Considera-se melhor excluir prematuramente uma entrada da tabela de vizinhos e receber um novo anúncio logo em seguida, do que mantê-la, criando um período de incerteza para as aplicações em execução. A Figura 3.12 mostra resultados de simulações feitas com $beaconInterval = 0,4\ s$ e $TTL < beaconInterval$ ($0,39\ s$ (98%), $0,3\ s$ (75%) e $0,2\ s$ (50%)). Neste gráfico, optou-se por representar o conjunto de vizinhos reais **desconhecidos** em lugar de VRC.

De fato, valores de TTL menores que o *beaconInterval* reduzem o número de falsos positivos de quase 20% para menos de 10%. Observa-se, entretanto, que o número de vizinhos reais desconhecidos aumentou muito.

Torna-se necessário, então, balancear estes dois parâmetros de modo a obter os melhores resultados. O valor de VRC na Figura 3.10, para $beaconInterval = 0,2\ s$, apresenta um bom resultado para o conjunto VRC: $83,64\%$. Nas mesmas simulações, com $TTL = 0,95 \times beaconInterval = 0,19\ s$, (95%), obtem-se $VCNR = 8,96\%$, também um bom desempenho.

Como visto também na Figura 3.10, com $beaconInterval = 0,1\ s$, tem-se apenas

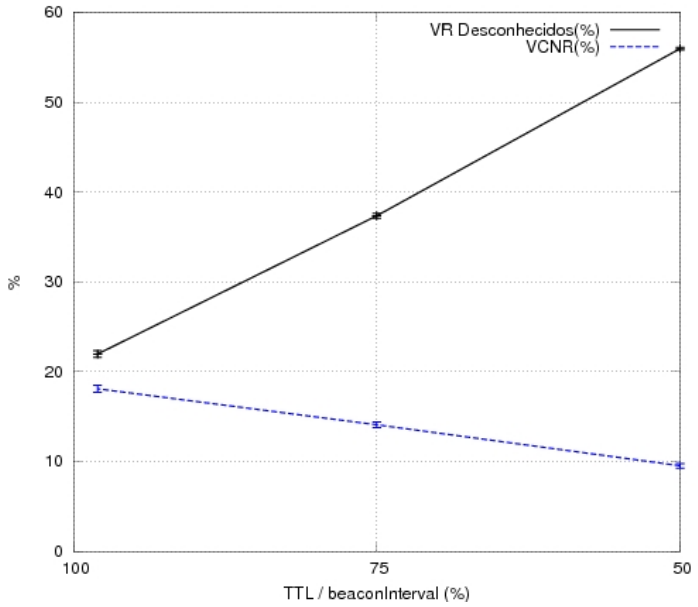



Figura 3.12: VR desconhecidos e VCNR para TTL menor que *beaconInterval*

uma pequena queda no VRC. Entretanto, para esta mesma situação, com $TTL = 0,9 \times beaconInterval = 0,09\ s$, (90%), obteve-se o melhor resultado para VCNR, $4,24\%$.

O caminho, então, é ajustar o valor do *beaconInterval* com base no conjunto VRC e no percentual de colisões. Este ajuste poderá ser mais ou menos "agressivo" em função de maior ou menor mobilidade generalizada na rede. Em seguida, ajusta-se o TTL, como uma fração entre 90 e 98% do intervalo entre anúncios de presença, para minimizar o conjunto VCNR.

Deve-se levar em conta que os resultados foram obtidos com um meio físico de comunicação considerado ideal. Com implementações reais ou simulações com modelos de erros e perdas na transmissão, os resultados serão inferiores.

Para altas densidades de nós, o canal de controle apresentará congestionamento e altos índices de colisões. Nas aplicações desta tese, este fato não se constitui em grande prejuízo. Como poderá ser observado no próximo capítulo, um conhecimento parcial da vizinhança se constitui em informação suficiente para a tomada de decisão dos agentes móveis. Em redes muito densas, o intervalo de anúncio de presença poderá ser reduzido, de modo a minimizar o congestionamento do canal de controle. Para os agentes móveis, pode ser melhor conhecer um percentual menor de um amplo número de vizinhos, do que ter um alto grau de conhecimento dentre um conjunto

pequeno de vizinhos. Apesar de serem prejudiciais no NIL, redes muito densas são favoráveis à execução das tarefas globais que serão discutidas no Capítulo 4.

Para aplicações que requeiram o conhecimento de 100% dos vizinhos, são necessárias técnicas que vão além da proposta do protocolo 3M. Para redes densas e com nós muito dinâmicos, esta tarefa é dispendiosa e de difícil execução.

## 3.6 Considerações finais

Este capítulo apresentou questões sobre coleta de dados e algoritmos de localização relacionadas ao NIL, Nível de Interações Locais, da proposta da tese.

Foi apresentado um estudo de protocolos de enlace e descoberta de vizinhos para RSSF. Identificou-se a inadequação das diversas propostas aos cenários de alta mobilidade que são alvo deste trabalho. Em função disso, desenvolveu-se o protocolo 3M, que atende aos requisitos de mobilidade e permite uma alocação eficiente de canais de freqüência para atender às aplicações em execução.

Por fim, foi feita uma análise de desempenho do protocolo proposto e foram apresentadas métricas e discutidos mecanismos de ajuste dos parâmetros do protocolo.

No próximo capítulo é apresentado, discutido e avaliado o NIG, Nível de Interações Globais da proposta.

# Capítulo 4

# NIG - O Nível de Interações Globais

Este capítulo trata as tarefas segundo um enfoque global da rede.

Uma vez que as tarefas do NIL estão em execução, cada nó sensor conhece a sua localização, a previsão de localização futura e o conjunto de vizinhos.

Sob o ponto de vista de alguma aplicação em execução, a rede deve ser vista como um todo. Requisições devem ser submetidas à RSSF e esta deve responder com as informações solicitadas.

Uma rede de sensores é, em geral, uma rede densa. A mobilidade aleatória impõe dificuldades adicionais ao funcionamento dela. A organização em duas camadas, com tarefas locais e globais tratadas separadamente, via troca de mensagens e agentes móveis, se constitui na proposta de tratamento dos desafios impostos pela mobilidade.

Uma vez que o NIL consiga coletar dados e torná-los disponíveis nos domínios locais, torna-se necessário um tratamento do roteamento destes até os sorvedouros, que também podem se mover em relação ao fenômeno.

Diante desta situação, esta tese propõe a existência de um nível de interações globais - NIG. Neste nível, é tratado o transporte troncalizado dos dados agregados da origem ao destino. Ao passo que o NIL tem foco em uma localidade, relacionada ao raio de cobertura de rádio dos nós sensores, o NIG busca uma visão de toda a rede e das tarefas a serem executadas na mesma. O NIL busca produzir um conhecimento mútuo entre os nós de uma pequena região e faz com que os nós

tenham estimativas relacionadas aos seus movimentos. O NIG busca cumprir uma tarefa global, por exemplo, buscar dados nos sensores em um limite da rede e trazer estes dados às proximidades de outra região. O NIG é executado sobre o NIL e utiliza as informações e os nós físicos deste nível inferior para executar suas tarefas.

Tong *et al.* [77] argumentam que, diante das restrições atuais dos sensores, é muito difícil, senão impossível, confiar apenas aos sensores as tarefas de acesso ao meio, criação e manutenção de rotas, armazenamento e encaminhamento de pacotes, etc. É proposto então que existam alguns dispositivos poderosos que responderão por tarefas críticas da rede. Como em cenários dotados de mobilidade não controlada não se pode garantir que estes "dispositivos poderosos" estarão no lugar certo e na hora certa, esta tese considera que tais tarefas especiais devam ser desenvolvidas através da cooperação de agentes móveis, utilizando recursos agregados de sensores homogêneos em uma dada região.

## 4.1 Introdução aos agentes móveis

A definição do que vem a ser exatamente um agente ainda não representa um consenso. Contudo, uma definição que está sendo considerada como adequada para muitos pesquisadores é a seguinte:

"Um agente é um sistema computacional encapsulado, que é situado em algum ambiente e é capaz de ações flexíveis e autônomas no ambiente, de forma a alcançar seus objetivos" [78].

Ser um sistema computacional encapsulado quer dizer que um agente deve ser uma entidade facilmente identificável e com interfaces bem definidas. Ele está imerso em um ambiente particular sobre o qual possui controle parcial e capacidade de observação. O agente recebe informações em relação a estados do ambiente e pode agir, ou não, de acordo com as informações recebidas, a fim de cumprir determinado objetivo pré-estabelecido (ações flexíveis). Em relação a ações autônomas, pode-se dizer que o agente deve possuir controle tanto sobre seu estado interno quanto sobre seus atos. Um agente deve ser reativo, ou seja, ter a capacidade de responder em tempo hábil às mudanças que ocorrem no ambiente, sempre em busca de seus objetivos. Por fim, um agente deve ser também proativo, adotando oportunamente

novos objetivos e tomando assim novas iniciativas.

Se além das características citadas anteriormente, o agente puder migrar de uma plataforma para outra, ele é chamado de agente móvel. Agente móvel é um tipo especial de *software* que pode executar autonomamente. Uma vez despachado, ele pode migrar de nó em nó, executando seu processamento por conta própria [79]. Algumas vantagens do uso de agentes móveis são a escalabilidade da RSSF, economia de banda, facilidade de reprogramação e agregação de dados [80]. Muitas aplicações são listadas por Fok *et al.* [2], onde a implementação de um *middleware* (plataforma) de agentes móveis para o sensor MICA2 e o desenvolvimento de algumas aplicações são descritos e avaliados. O referido trabalho implementou a plataforma de agentes para sensores, chamada Agilla. Foram implementadas e executadas aplicações reais, de complexidade semelhante àquelas que esta tese delega aos agentes móveis.

Agentes móveis são programas que não estão ligados ao sistema inicial (*home platform*), estando livres para migrar entre os nós da rede. São tipicamente construídos através de componentes procedurais ou de classes de objetos e carregam consigo seu código e estado. Estado são valores de atributos que vão ajudá-lo a determinar o que fazer e quando fazer.

O sistema inicial, citado no parágrafo anterior, pode ser composto por um ambiente que contém uma ou mais aplicações que vão interagir com um servidor remoto. Estas aplicações podem incluir, por exemplo, busca e recuperação de informações.

Este ambiente de aplicações deve estar interligado com um ambiente de execução de agentes. Através de APIs – *Application Programming Interface* (ou Interface de Programação de Aplicativos), a aplicação pode passar parâmetros para várias classes de agentes e, igualmente, estes agentes podem retornar parâmetros para o ambiente de execução.

O ambiente de execução dos agentes, também chamado de plataforma de agentes, deve permitir a execução de diferentes tipos de programas agentes. A plataforma provê diferentes tipos de serviços às aplicações clientes e pode estar interligada com várias funções do sistema operacional, tais como gerenciador de memória, relógio, sistema de arquivos e, principalmente, ao serviço de transporte de mensagens. Este serviço de mensagens é utilizado para enviar e receber agentes móveis e as próprias mensagens, através da infra-estrutura de comunicação. O ambiente de execução

pode também fazer uso de alguma máquina virtual que torne transparente o sistema operacional local. Um exemplo é a máquina virtual Java [63].

O agente deve estar apto a se mover para outro nó se o serviço necessário não estiver disponível ou for insatisfatório, ou ainda, o agente deve estar apto a determinar se ele deve visitar outro nó baseado nos dados que ele recebeu do nó corrente.

A mobilidade dos agentes lhes garante acesso aos mais diferenciados tipos de sistemas e, conseqüentemente, interação com estes ambientes. Logo, é necessário salientar que os agentes móveis podem ser ferramentas poderosas, mas também podem ser destrutivos se algumas precauções não forem tomadas.

O mau uso da tecnologia de agentes pode causar danos tanto aos agentes quanto aos nós hospedeiros. O grande desafio que se tem é como prover, simultaneamente, segurança efetiva para os agentes móveis e para os nós onde serão executados. Na Seção 4.2 é proposto um esquema de segurança destinado a proteger a plataforma (no caso o nó sensor) de agentes maliciosos ou não autorizados.

Na presente proposta, os principais estímulos que atuam sobre os agentes são as informações de localização obtidas dos nós sensores e as tarefas de sensoriamento em execução na rede. A plataforma que suporta os agentes móveis está em execução nos sensores. Enquanto os nós sensores são dotados de movimento real no espaço, os agentes móveis são dotados de mobilidade virtual sobre o campo de sensores. Os agentes móveis ficam limitados à granularidade com que os sensores estão dispostos no campo e à localização atual deles. Se considerarmos que um programa em execução "***está***" na plataforma onde suas instruções estão sendo processadas, então podemos dizer que o agente móvel também se move "***fisicamente***". De fato, para as aplicações a serem tratadas, esta mobilidade existe e é eficaz.

Os nós sensores têm interações locais que ocorrem na forma de troca de mensagens. As informações globais referentes às tarefas de coleta de dados em execução na rede como um todo são executadas e propagadas pelos agentes móveis.

Qi e Iyengar [79] apresentam uma rede de sensores que utiliza o paradigma de agentes móveis. Nesta arquitetura, em lugar de movimentar grandes volumes de dados pela rede, os dados ficam nos sensores enquanto os agentes (códigos + dados tratados (agregados)) se movem de nó em nó. Como discutido pelos autores, os benefícios da arquitetura são: transmitir agentes no lugar de dados reduz a banda

requerida pela rede, aumento da escalabilidade, aumento da vida útil da rede e possibilidade de reprogramação da rede.

Lange e Oshima [81] apontam que mobilidade é uma propriedade desejável de agentes de *software*, dependendo das tarefas a serem executadas, do volume dos dados a serem manipulado e das características das redes, em termos de desempenho, qualidade de serviços (QoS) e topologia. Na presente proposta, a mobilidade passa a ser fundamental para que as técnicas de localização e cobertura da área de interesse baseadas em mobilidade intencional sejam aplicáveis.

## 4.2 Aspectos de segurança

Um sistema baseado em agentes consiste de uma coleção de componentes distribuídos entre vários computadores conectados através de uma rede. Esses componentes precisam interagir entre si, a fim de trocar dados ou acessar os serviços uns dos outros. A utilização de agentes móveis implica a necessidade de serviços extras nos sensores onde estes códigos serão executados. Cada sensor necessita de um ambiente computacional, normalmente identificado como plataforma de agentes, cujo propósito é dar suporte às aplicações para realocar dinamicamente seus componentes de *software* sob diferentes sítios. Esta plataforma é usada para criar, interpretar, transferir e retomar agentes.

Tipicamente, no paradigma cliente X servidor, os dados são transmitidos até uma estação base, para que neste momento eles possam ser processados. No paradigma baseado em agentes móveis, uma possibilidade é que os dados não se movam, mas sim os agentes, que passam de nó em nó, efetuando o processamento dos dados armazenados. Os dados carregados pelos agentes podem ser apenas os resultados do processamento, podem ser dados locais agregados ou fundidos, ou podem ser os dados puros coletados.

Contudo, a aceitação do paradigma baseado em agentes ainda está comprometida pelos insuficientes mecanismos de segurança. O código móvel (agente móvel) apresenta novos desafios de segurança e realça problemas já conhecidos, particularmente em função de políticas inadequadas de segurança existentes na área da mobilidade [82].

Devido às características do paradigma de agentes móveis e às ameaças a que este está exposto, os mecanismos de segurança possuem três finalidades: i) proteção do suporte de comunicação, ii) da plataforma de agentes ou iii) dos próprios agentes. Esta seção se concentra no problema da segurança das plataformas de agentes, considerando redes de sensores móveis e em larga escala. Sua capacidade extremamente limitada de computação e de comunicação torna desafiadora a implementação de quaisquer mecanismos de segurança.

### 4.2.1 Segurança das plataformas de agentes móveis

Diante dos possíveis ataques oriundos dos agentes móveis, os recursos e funcionalidades das plataformas devem ser protegidos. Uma plataforma, para se proteger de um agente, depende não só da verificação da autenticidade do proprietário do agente, mas também do grau de confiança das plataformas já visitadas pelo agente, já que um agente móvel pode se tornar malicioso em virtude de ter sido corrompido por plataformas visitadas anteriormente.

Uma plataforma de agentes móveis pode ser exposta a, basicamente, três tipos de ataques: personificação, quando um agente assume a identidade de outro, a fim de obter acesso a recursos e serviços não permitidos a ele; a negação de serviços, quando agentes iniciam ataques que exploram vulnerabilidades da plataforma gerando um consumo excessivo de recursos; e o acesso não autorizado, quando um agente obtém acesso a um recurso sem possuir autorização [83].

### 4.2.2 Criptografia e assinatura digital

A criptografia com curvas elípticas (ECC) está emergindo como um atrativo criptossistema de chaves públicas para dispositivos móveis e sem fio. Comparado à criptossistemas tradicionais, como RSA [84], o ECC oferece segurança equivalente com chaves menores, o que resulta em uma computação mais rápida, menor consumo de energia, além da economia de memória e banda. Tais características são extremamente úteis para sensores que possuem grandes limitações em termos de CPU, energia e conectividade.

Criptografia é a combinação de uma chave com um algoritmo matemático baseado em uma função unidirecional. Este algoritmo é aplicado aos dados, juntamente

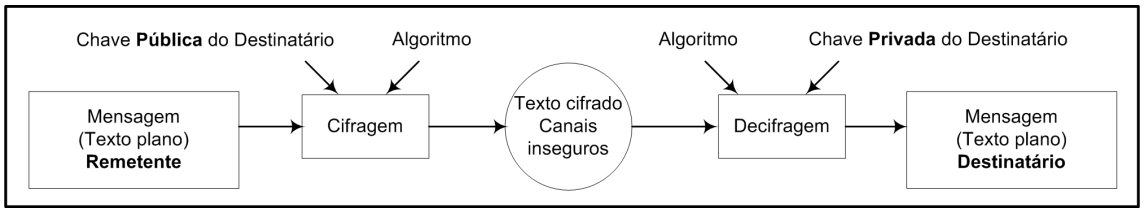



Figura 4.1: Criptografia assimétrica

com a chave, de modo a torná-los indecifráveis para qualquer um que os veja. O modo como isso é feito garante que a obtenção dos dados originais só é possível através da obtenção do algoritmo e chave usados inicialmente. Mantendo-se um destes dois componentes secretos (no caso, a chave), faz-se com que a visualização dos dados por terceiros se torne extremamente difícil.

Existem dois tipos básicos de algoritmos criptográficos que podem ser utilizados tanto sozinhos como em combinação. Estes algoritmos, de Chave Única e Chave Pública e Privada, são usados para diferentes aplicações de deve-se analisar qual é o melhor para cada caso.

O primeiro tipo de algoritmo que surgiu foi o de chave única, também chamados de algoritmos de chave simétrica. Neste, o sistema usa a mesma chave tanto para cifrar como para a decifrar os dados, e esta deve ser mantida em segredo. Estes algoritmos têm a vantagem de serem muito mais rápidos na cifragem e decifragem do que os algoritmos de chave pública e privada, além de poderem ser implementados em *hardware*. A sua desvantagem, talvez a única, é a distribuição de chaves, a qual deve ser feita por meio de um canal seguro ou com a utilização de protocolos de distribuição apropriados.

Sistemas de criptografia com chave pública (sistemas assimétricos) foram inicialmente propostos por Whitfield Diffie e Martin Hellman em 1976 [85]. Esses sistemas trabalham com duas chaves diferentes, independentes e não facilmente deriváveis: A chave pública, utilizada na codificação de uma mensagem cifrada, e a chave privada, utilizada na sua decodificação. A Figura 4.1 ilustra o esquema descrito.

Opcionalmente, quando se deseja assinar digitalmente uma mensagem, veja Figura 4.2, o emprego da chave pública e privada se inverte, ou seja, o remetente "assina" a mensagem através de sua chave privada e envia para o destinatário a mensagem junto com a assinatura. O destinatário consegue, através da chave pública

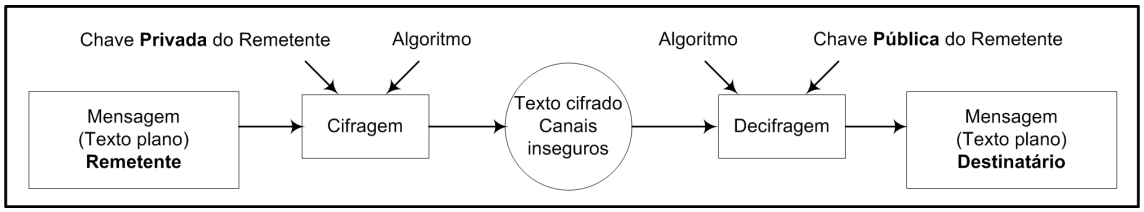



Figura 4.2: Assinatura digital

do remetente, validar a assinatura recebida e garantir que a mensagem assinada foi enviada pelo remetente. Não há criptografia. A segurança está em poder armazenar a chave privada em segurança e ser computacionalmente muito difícil obter essa chave, a partir da mensagem cifrada e da chave pública correspondente.

Quando a assinatura digital é gerada, ela passa a ser transmitida para o destino junto com o agente. Caso este tenha sofrido quaisquer alterações no caminho, o receptor, quando calcular a assinatura digital do agente recebido e compará-la com a assinatura recebida, irá perceber que as duas são diferentes e concluirá que o código do agente foi alterado. Tal procedimento garante a integridade do agente que chega à plataforma.

A autenticação através da assinatura digital é uma operação bastante rápida quando comparada à criptografia, porém ela sozinha não consegue impedir que os dados sejam lidos. Ela deve ser usada apenas nos casos onde existe a necessidade de confiabilidade e não de sigilo. Caso se necessite de ambos, usa-se a assinatura digital em conjunto com a criptografia.

Nesta proposta, utiliza-se apenas a autenticação do código do agente. O objetivo é garantir à plataforma que a ligação existente entre cada agente e seu autor é autêntica.

### Classes de sistemas de criptografia assimétrica

A criptografia é a combinação de uma chave com um algoritmo matemático baseado em uma função unidirecional, isto é, um problema matemático de difícil solução. Dentro deste contexto, os sistemas existentes apoiam-se no fato de que o algoritmo mais eficiente levará um longo período de tempo até encontrar a solução.

Segundo Vieira [86], tem-se observado que, ao se projetar sistemas criptográficos de chave pública, é necessário haver um compromisso entre o nível de segurança e o

tempo de resposta que se deseja obter. Nesse aspecto, quanto mais desenvolvidos forem as ferramentas e algoritmos utilizados para violação dos sistemas de criptografia existentes, maiores têm que ser os parâmetros (chaves) e, conseqüentemente, maior o esforço no trabalho de codificação e decodificação dos textos cifrados.

Atualmente, existem somente três tipos de sistemas de criptografia com chave pública considerados seguros e eficientes. Esses sistemas estão classificados de acordo com o problema matemático em que eles se baseiam [87]:

**Fatoração de inteiros**

**Definição:** Dado um número $n$ que é o produto de dois valores primos grandes $p$ e $q$ (ou seja, $n = pq$), determinar $p$ e $q$.

**Base:** Enquanto encontrar números primos grandes é uma tarefa relativamente fácil, o problema de fatorar o produto desses valores é considerada uma tarefa computacionalmente intratável.

**Logaritmo discreto**

Vários esquemas de criptografia de chave pública são baseados nos logaritmos discretos, como, por exemplo, o DSA (Digital Signature Algorithm) [84], os vários esquemas de ElGamal [84], o esquema de assinatura digital de Schnorr [84], o esquema de assinatura digital de Nyberg_Rueppel [84], entre outros.

Para todos os esquemas baseados nos logaritmos discretos existe um análogo utilizando curvas elípticas. O problema dos logaritmos discretos consiste na dificuldade de se encontrar algum método computacionalmente viável de se calcular logaritmos num dado grupo G.

Antes de se apresentar a definição de logaritmos discretos, se faz necessário apresentar o conceito de elemento gerador de um grupo.

**Elemento gerador:** um elemento $\alpha$ é dito gerador ou primitivo de um grupo G se todos os elementos $\beta$ não nulos deste grupo puderem ser escritos sob a forma $\beta = \alpha^k$, onde $k$ é um número inteiro.

**Definição:** Considere G um grupo multiplicativo finito de ordem $n$ e $\alpha$ o gerador de G. O logaritmo discreto do elemento $\beta$ de G na base $\alpha$, denotado por $\log_\alpha \beta \bmod n$, é o único inteiro $x$, $0 \leq x < n$, tal que $\beta = \alpha^x \bmod n$.

**Base:** Um algoritmo óbvio para se calcular o logaritmo discreto consiste em computar potências sucessivas $\alpha^k \bmod n$ até que $\beta$ seja encontrado (método da

força bruta). Este método é computacionalmente inviável para $n$ muito grande.

**Logaritmo discreto em curvas elípticas**

**Definição:** Dada uma curva elíptica $E$, definida por um conjunto finito de pontos de natureza $F_q$ ($q$ é o número elementos do conjunto) – $E(F_q)$ – e os pontos $P$, $Q \in E(Fq)$, determinar o inteiro $l(0 \leq l \leq q-1)$ tal que $Q = lP$.

**Base:** Ao passo que é relativamente fácil determinar o ponto $Q = lP$ (é visto ao se definir as operações em curvas elípticas), determinar $l$ dados os pontos $P$, $Q \in E(F_q)$ é bem mais difícil .

### Aplicação de curvas elípticas em criptografia

Ao se comparar os três sistemas de criptografia citados anteriormente, devemos considerar dois aspectos básicos: segurança e eficiência. A questão da segurança se baseia no fato de que o problema dos logaritmos discretos sobre curvas elípticas é considerado mais intratável que o dos logaritmos discretos sobre corpos finitos. Entretanto, nem todas as curvas elípticas podem ser consideradas seguras, assim como nem todas as curvas elípticas são eficientes do ponto de vista da aritmética elíptica.

A discussão acerca da eficiência de cada um dos sistemas de criptografia leva em consideração os seguintes fatores: carga computacional, tamanho de chave e tamanho de banda. O resultado desta comparação pode ser encontrado detalhadamente em [87] e resumido a seguir:

**Carga Computacional:** Mede a eficiência com que os algoritmos podem implementar as transformações com as chaves públicas e privadas (sistema em operação). As melhores implementações de cada um dos sistemas indicam que o ECC executa, aproximadamente, 10 vezes mais rápido que o RSA ou DSA.

**Tamanho de Chave:** A Tabela 4.1 compara o tamanho dos parâmetros de sistema e das chaves para três diferentes sistemas.

**Tamanho de Banda:** Corresponde a quantos bits (a mais) devem ser transmitidos após criptografar ou assinar uma mensagem, em relação a mensagem original. Nesse quesito, todas as três opções apresentam valores parecidos, com o ECC se destacando principalmente nos casos em que são processadas mensagens pequenas, de aproximadamente 100 bits [87].

| | Parâmetros do Sistema | Chave Pública | Chave Privada |
|---|---|---|---|
| RSA | n/a | 1088 | 2048 |
| DAS | 2208 | 1024 | 160 |
| ECC | 481 | 161 | 160 |

Tabela 4.1: Tamanho dos parâmetros e do par de chaves

Neste contexto, pode-se concluir que o uso da criptografia com chave pública baseada em curvas elípticas é a opção mais adequada às redes de sensores, não somente em termos de segurança, como também na eficiência de sua execução.

Segundo Almeida Jr. [88], são muitas as escolhas que devem ser feitas para se desenvolver um criptossistema baseado em curvas elípticas (ECC). Elas vão desde o tipo de curva elíptica, passando por sua representação, e chegando na escolha de algoritmos a serem aplicados. A seguir são apresentados os passos para a implementação de criptossistemas baseados em curvas elípticas. O detalhamento de cada uma destas opções pode ser encontrado no referido trabalho [88].

- Definir a natureza de seu campo finito $F_q$ ($F_P$ ou $F_{2m}$);
- Selecionar a representação para os elementos de $F_q$ (polinomial, base ótima, subcampos, etc);
- Implementar aritmética e operações em $F_q$;
- Selecionar uma curva apropriada em $F_q$ (quais parâmetros utilizar para a curva);
- Definir um ponto gerador em $E(F_q)$;
- Definir o mapeamento da mensagem original em pontos de uma curva ("*embedding*").

Além da definição da própria curva $E(F_q)$ como parte dos parâmetros globais que devem ser mantidos públicos, existe um ponto denominado ponto gerador ou ponto base ($g \in E(F_q)$). Esse ponto é uma referência que permitirá a realização da criptografia. O ponto $g$ é obtido a partir da escolha de um valor $n$ primo grande tal que $ng = \infty$ (ponto no infinito).

O último passo para a implementação das curvas elípticas é o *message embedding*, que, segundo Vieira [86], é um procedimento essencial ao uso deste tipo de criptografia. Este passo consiste em uma forma de mapear a mensagem original (texto puro) em pontos de uma curva elíptica. Isso corresponde a colocar a mensagem original sobre a curva elíptica definida. Esses pontos, depois de sofrerem operações parametrizadas pelas chaves individuais de um usuário, dão origem a um outro conjunto de pontos, que representam os pontos originais cifrados. Aqui observa-se uma clara diferença entre os métodos de criptografia tradicional e a técnica baseada em curvas elípticas: ao invés de texto cifrado, são transmitidos um conjunto de pontos cifrados. Esses pontos cifrados, ao serem recebidos pelo destinatário, são convertidos, através da chave correspondente, nos pontos originais. Nesse momento, ao aplicar a rotina de mapeamento invertida, recupera-se texto original e todo o ciclo criptográfico se fecha.

Uma vez estabelecidos todos os parâmetros e características gerais nas quais o sistema de criptografia com curvas elípticas deve se basear, basta que cada usuário determine seus parâmetros individuais, ou seja, seu par de chaves pública/privada, e os demais parâmetros locais necessários ao resto da implementação.

### 4.2.3 Algoritmo de assinatura digital baseado em curvas elípticas (ECDSA)

ECDSA (Elliptic Curve Digital Signature Algorithm) é um algoritmo específico para curvas elípticas análogo ao DSA (Digital Signature Algorithm). O DSA é uma variante dos algoritmos ElGamal que foram padronizados em 1991. O ECDSA opera sobre curvas elípticas $E(F_q)$ enquanto o DSA opera sobre um grupo primo e multiplicativo $(Z_p^*)$.

A seguir, é descrito o algoritmo ECDSA, utilizado para geração de chave, assinatura digital e verificação da assinatura digital.

#### Geração das chaves

1. Selecione uma curva elíptica $E(F_q)$ de maneira que o número de pontos contidos nela seja divisível por um número primo grande $n$;

2. Selecione um ponto $P \in E(F_q)$ de ordem $n$;

3. Selecione um número inteiro $d$, randômico, no intervalo [1,n - 1];

4. Computar $Q = dP$;

5. A chave pública é $(E, P, n, Q)$. A chave privada correspondente é $d$, a curva elíptica utilizada é $E$, $P$ é o ponto escolhido na curva e $Q$ é o ponto que obtem-se multiplicando $P$ pela chave privada $d$;

**Assinando uma mensagem**

Para assinar uma mensagem $m$ devemos aplicar o seguinte algoritmo:

1. Selecionar um número randômico $k$ no intervalo [1, n-1];

2. Computar $kP = (x_1, y_1)$ e $r = x_1 \, mod \, n$;

3. Computar $k^{-1}$ mod $n$;

4. Computar $s = k^{-1}(h(m) + dr)$ mod $n$, onde $h$ é o *Secure Hash Algorithm* (SHA-1);

5. Se $s = 0$ então volte para o passo 1. (Se $s = 0$ então $s^{-1} \, mod \, n$ não existe);

6. A assinatura da mensagem $m$ é o par de inteiros $(r, s)$.

**Verificação da assinatura**

1. Obter a chave pública do remetente $(E, P, n, Q)$ seguramente. Verificar se o $r$ e o $s$ estão no intervalo[1,n - 1];

2. Computar $w = s^{-1}$ mod $n$ e $h(m)$;

3. Computar $u_1 = h(m)w$ mod $n$ e $u_2 = rw$ mod $n$;

4. Computar $u_1P + u_2Q = (x_0, y_0)$ e $v = x_0$ mod $n$;

5. Aceitar a assinatura se $v = r$;

No ECDSA, as entidades participantes do processo de criptografia não precisam gerar sua própria curva elíptica. Pode-se escolher um ponto e uma curva padrão. No cinco passos descritos, a geração da curva e dos pontos são mostrados para manter uma visão completa e coerente dos algoritmos. Pode-se assumir, sem medo de errar, que os parâmetros $E, P$ e $n$ são conhecidos por todas as partes envolvidas.

### 4.2.4 Esquema de segurança proposto

Enquanto o ECC tem a vantagem de possibilitar o uso de pequenas chaves, os requisitos de implementação continuam excessivos. Logo, implementações eficientes e otimizadas tornam-se imprescindíveis para plataformas particularmente restritivas, como aquelas usadas em sensores. Um dos objetivos desta proposta é apresentar o projeto da implementação de um protocolo de autenticação, baseado no algoritmo ECDSA, e parametrizado de maneira a atender os requisitos das redes de sensores. É importante ressaltar que o ECDSA é um algoritmo de assinatura digital e não um protocolo. Além disso, sua implementação está vinculada à correta escolha de parâmetros, como por exemplo, a curva que será utilizada. Tais parâmetros influenciam grandemente na ocupação de memória e gerenciamento de energia dos sensores.

Nas redes de sensores, os nós e os agentes normalmente são oriundos de uma mesma entidade administrativa o que possibilita um "encontro" prévio entre os milhares de sensores e os agentes que irão habitá-los. Esta é a maior diferença entre o ambiente de redes de sensores e as redes tradicionais. O protocolo proposto aqui torna-se muito mais eficiente e econômico partindo deste pressuposto. Nas redes de sensores, os nós são produzidos e depositados pela mesma entidade administrativa que irá injetar os agentes no futuro. Isto gera uma potencial relação de confiança, antes mesmo que os nós sejam depositados no ambiente de monitoração (*deployed*). Tal particularidade aumenta o desempenho do protocolo de autenticação proposto, uma vez que não são necessárias entidades certificadoras para garantir a relação de confiança entre a chave pública e o autor do agente. Assim, cada sensor carregará consigo somente uma única chave pública: a do autor dos agentes que irão visitá-lo.

Durante o processo de criação de um agente móvel, o proprietário, na qualidade da autoridade que o agente representa, assina o código do agente e o despacha pela

rede de sensores para que ele recolha as medições efetuadas e as traga de volta para a região de retorno. Portanto, pode-se afirmar, que na estação base são executadas somente as duas primeiras fases do algoritmo ECDSA: geração das chaves e da assinatura digital. A terceira fase, validação da assinatura digital, é executada por cada plataforma (sensor) que receberá o agente.

Uma plataforma, ao receber um agente móvel, deve primeiramente, através da verificação da assinatura do código do agente, comprovar que este agente permanece íntegro e confirmar a sua associação ao seu proprietário. Assim, quando a assinatura digital é gerada, ela passa a ser transmitida para o destino junto com o agente. Caso este tenha sofrido quaisquer alterações no caminho, o receptor, quando calcular a assinatura digital do agente recebido e compará-la com a assinatura recebida, irá perceber que as duas são diferentes e concluir que o código foi alterado.

Os agentes móveis passam pelo protocolo de autenticação junto à plataforma destino para que esta tenha certeza de sua origem e da não alteração de seu código durante o caminho percorrido, já que os agentes têm como premissa itinerários livres e múltiplos saltos (*multi-hop*).

Desta forma, modificações feitas por plataformas maliciosas podem ser facilmente detectadas. Conseqüentemente, quando a plataforma destino detectar uma modificação no código do agente, esta deve não só impedir sua execução como também sua propagação pelo restante da rede. Da mesma forma, um agente externo, não assinado pela entidade administrativa, será imediatamente idenficado e eliminado. A Tabela 4.2 ilustra o funcionamento do protocolo de autenticação proposto:

É importante ressaltar que no protocolo de autenticação proposto não existe a necessidade de uma entidade certificadora, uma vez que os sensores serão pré-configurados pela mesma entidade administrativa. Tal solução justifica a especificação de funções descritas na Tabela 4.2.

Através do processo de autenticação descrito, é possível garantir a origem dos agentes em uma comunicação entre duas plataformas. Entretanto, é preciso ressaltar que, a assinatura digital garante que um determinado agente não foi alterado durante seu percurso, mas não impede que o código deste agente seja lido.

| | |
|---|---|
| Estação Base | Sensor |
| Gera chave pública e privada | |
| | Sensores são depositados c/ a chave pública |
| Gera Assinatura Digital | |
| Envia Agente e sua correspondente assinatura digital (credenciais) | |
| | Recebe o Agente e suas credenciais |
| | Valida assinatura digital recebida |

Tabela 4.2: Funcionamento do protocolo de autenticação proposto

### 4.2.5 Considerações finais da seção

Nesta seção foi apresentado um novo esquema de segurança para proteção das plataformas de agentes móveis em redes de sensores. Este esquema baseia-se principalmente no algoritmo ECDSA e se propõe a validar o código de todo agente que chega ao sensor. Desta maneira, consegue-se evitar que agentes maliciosos ou oriundos de outras entidades administrativas tenham acesso aos sensores e conseqüentemente à plataforma.

O importante neste esquema de segurança é adaptar esta grande variedade de mecanismos de segurança às severas restrições impostas pelas redes de sensores, impondo um baixo consumo de energia e memória. Para este fim, uma técnica com menores custos computacionais e de comunicação é aplicada: criptografia de curvas elípticas.

## 4.3 Aplicação dos agentes móveis ao NIG

Algumas das propostas estudadas no Capítulo 2 foram inseridas no padrão de funcionamento dos agentes móveis proposto neste trabalho.

A proposta de Vahdat e Becker [24], Seção 2.3.2, permite a entrega de mensagens apesar da rede não ser totalmente conexa. Adaptando o que prevê a proposta, os agentes móveis armazenam as mensagens até que "encontrem" outros agentes a quem devam encaminhá-las ou até que cheguem à região de entrega. Esta região se

situa no raio de alcance de rádio do nó sorvedouro ou da estação base que retira os dados de interesse da RSSF.

Choksi *et al.* [11], Seção 2.3.2, propõem um trabalho que oferece suporte à mobilidade em redes de sensores *ad hoc* baseadas em difusão. O tópico relativo à antecipação de rotas foi inserido nos algoritmos de decisão dos agentes móveis (Seção 4.5). Os agentes móveis, de posse das previsões de localização dos nós, executam ou não migrações objetivando alcançar uma região geográfica desejada.

Shah *et al.* [12, 13], Seção 2.3.2, propõem entidades reais, as *Data MULEs*, que coletam dados pela rede de sensores. A aplicação na proposta desta tese é direta, diferindo apenas pelo fato da coleta ser feita por agentes móveis. Na proposta original, as *Data Mulles* possuem mobilidade aleatória. Na presente proposta, existe a vantagem da mobilidade dos agentes ser voluntária e voltada para a tarefa em execução.

A proposta de Vahdat e Becker [24] (Seção 2.3.2) considera o armazenamento (*bufferização*) de dados até o nó em questão encontrar outro que não tenha estes dados. Ao encontrar, os dados são repassados também a este nó. Deste modo, consegue-se a multiplicação dos dados pela rede e a entrega destes, mesmo com particionamentos (desconexões) momentâneos na rede. Na proposta desta tese, um agente móvel, ao permanecer em um determinado nó de um subgrafo de uma rede não totalmente conexa, pode, no futuro, obter conexão com outro subgrafo, atingindo uma região de destino apropriada.

Já que os agentes móveis estarão cobrindo toda a RSSF, poderão se fundir ou se clonar, e podem assumir tarefas adicionais e adjacentes ao seu movimento. Diante da necessidade de difusão de dados reais de posicionamento para o processo de ajuste nos algoritmos de localização, os agentes móveis podem auxiliar nesta tarefa. Isto pode ser feito através da coleta de dados de localização dos nós que tiverem tal capacidade e a divulgação destes aos nós sensores visitados. A tabela de estados e discretização relativa à conectividade utilizada na proposta de detecção de iminência de desconexão [25], Seção 2.3.2, poderá ser divulgada ou modificada em tempo de execução pelos agentes móveis.

Estas possibilidades são identificadas como potencialidades do esquema aqui proposto, mas não foram modeladas e implementadas. Foi priorizada a modelagem e

implementação das ações mais fundamentais de coleta e transporte de dados.

De um modo geral, todas as técnicas de localização estudadas, que dependem de mobilidade intencional ou controlada, podem ser aplicadas na medida em que os agentes móveis terão como se mover autonomamente. As duas limitações são a acurácia do processo de localização e a densidade de sensores. Os agentes móveis perceberão uma discretização do plano, já que só podem se mover de sensor a sensor.

A técnica de localização com estimativas e posteriores correções de [7], Seção 2.3.1, pode ser aplicada tanto aos sensores no NIL quanto aos agentes móveis no NIG. Ao ser aplicado no NIL, os sensores podem recorrer aos dados das sementes tanto no próprio NIL, quando tiverem contato direto com elas, quanto ao receber dados de localização propagados por agentes móveis que visitem suas vizinhanças.

Como visto na Seção 2.3.1, Bachmayer e Leonard [23] apresentam a proposta baseada em gradientes ambientais e interações entre pares, para orientar o mobilidade intencional. A aplicação nesta tese se dá em mais de um ponto. As interações locais produzem os vetores de movimento dos sensores, de modo que estes criem uma informação distribuída, ou seja, o próprio campo de gradientes que norteará o movimento dos agentes móveis. Adicionalmente, poderão refinar a estimativa de localização dos sensores. Isto porque os agentes móveis poderão utilizar os estímulos ambientais retirados do substrato inferior dos sensores por onde se movem e estímulos adjacentes de outros agentes móveis, de modo a tomar as decisões mais corretas na execução de sua tarefa.

Huang *et al.* [14], Seção 2.3.2, apresentam um esquema que permite definir um caminho pelo qual mensagens devem seguir, por um campo de sensores estáticos e com relógios sincronizados. A proposta desta tese produz uma rota de movimentação do agente, ao longo da RSSF randomicamente móvel, sem necessidade de sincronismo de relógios.

Li e Hus [89] apresentam um protocolo de navegação em redes de sensores que define obstáculos e áreas de perigo. O protocolo sugere, a uma entidade externa móvel, um caminho a seguir que evite estes perigos e obstáculos. Nesta tese, o caminho ideal é definido pelo agente móvel. A analogia a perigos e obstáculos, regiões pelas quais o agente não deve passar, são áreas que não atendem aos limites de coordenadas dos eventos de interesse, bem como os sensores que se movem em

direções desfavoráveis ao atingimento da região de interesse.

De um modo geral, as técnicas apresentadas se relacionam à proposta desta tese. Algumas de forma direta, influenciando na forma como atuam os agentes móveis. Outras influenciaram de forma indireta, justificando o uso de migração autônoma e de influências ambientais no funcionamento dos agentes.

## 4.4 Uma visão geral dos níveis de interações

Na Figura 4.3 vemos uma representação esquemática da arquitetura. Os eventos estão no retângulo 1. Admite-se mobilidade dos eventos dentro desta área e do retângulo 1 em relação ao 2. Neste caso, áreas de interesse podem deixar de ser cobertas pela rede temporariamente ou definitivamente.

Observa-se que os níveis 2 e 3 estão exatamente sobrepostos. O escopo de 3 não pode ser maior que o de 2, já que os agentes móveis se hospedam em sensores, ou seja, sobre um espaço virtual discreto que é um subconjunto do retângulo 2. O retângulo 2 pode ser maior que o 3, na medida em que existam sensores sem agentes. A intenção é buscar uma cobertura sub-ótima e econômica do campo de sensores. Os retângulos 2 e 3 podem sofrer translações globais entre si, mas a intenção é que isto não ocorra. Ao identificar o perfil de mobilidade da rede, espera-se que a cobertura dela se mantenha e que não tenhamos agentes portadores de dados "se suicidando" na extremidade por não ter para onde ir.

Na Figura 4.4 tem-se um detalhamento apenas ilustrativo do NIL (retângulo 2). Os sensores formam vizinhanças locais, com conhecimento mútuo. Os sensores podem migrar entre estas vizinhanças e podem estar momentaneamente sem vizinhos conhecidos. Os sensores e suas funções são homogêneos. Não há líderes ou coordenadores.

## 4.5 Algoritmos de decisão dos agentes móveis

### 4.5.1 Introdução

Como já mencionado, alguns cenários de interesse são sensores usados para estudo do movimento de tornados [6]; sensores conectados a veículos, em grandes cidades,

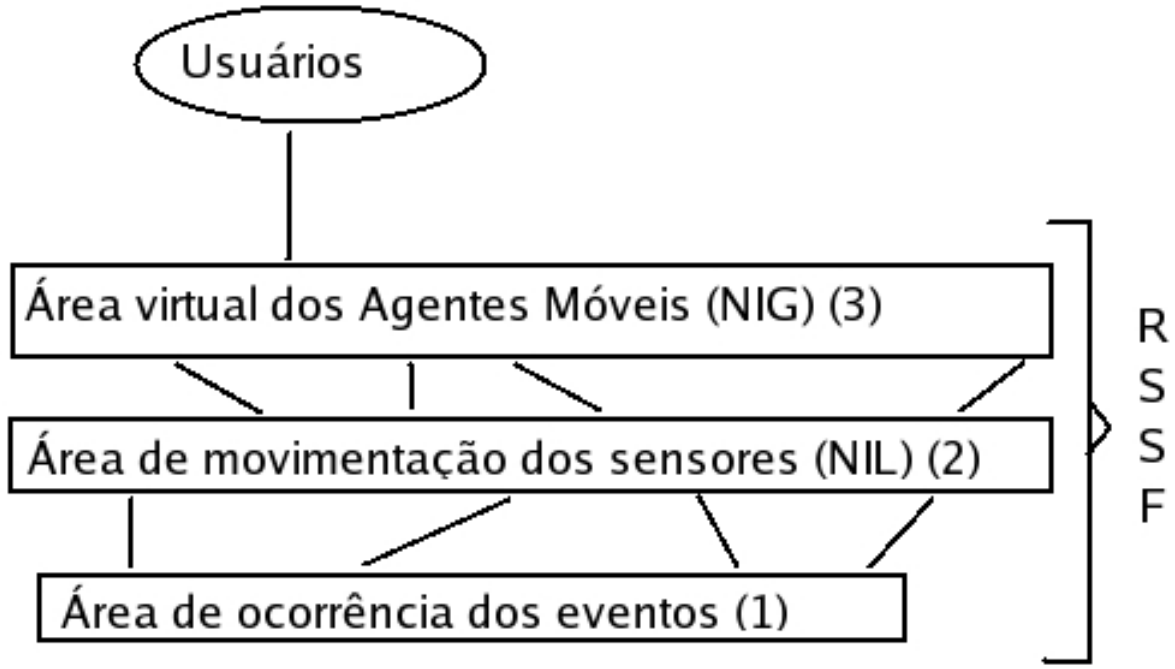



Figura 4.3: Visão geral da proposta

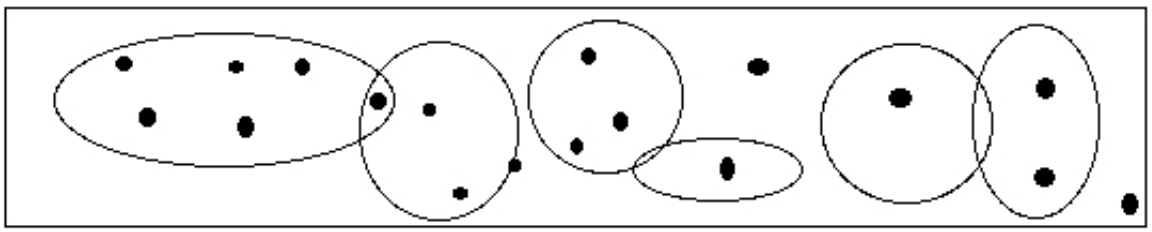


Figura 4.4: Os sensores e seus vizinhos no NIL

para estudo das condições de tráfego e planejamento de rotas [6]; sensores flutuando em correntes fluviais [7]; projeto da NASA para exploração de Marte, com sensores dirigidos por correntes eólicas [8]; etc. Nestes cenários, os nós se movem aleatoriamente pelo campo de sensoriamento. Considera-se que os nós executam um protocolo de localização que lhes forneça suas coordenadas em um dado momento, com uma certa precisão. Os nós estimam repetidamente sua posição futura e a direção de seu movimento (Seção 3.3).

Executando o protocolo 3M (Seção 3.5), todos os nós mantém uma lista com seus vizinhos conhecidos em um dado instante. Para um dado nó $i$, este conjunto será chamado de $V_i$.

No presente trabalho, os agentes móveis têm tarefas de sensoriamento a serem executadas. Estas tarefas contém a especificação do fenômeno de interesse e da forma de coleta (Seção 3.1). Além disso, possuem as coordenadas da região **onde** os dados devem ser coletados (região destino) e as coordenadas da região **para onde** os dados devem ser trazidos (região de retorno). O destino momentâneo de um agente móvel é chamado de **região alvo**. A região alvo alterna entre região destino e a de retorno. Estas regiões são definidas pelas coordenadas dos vértices opostos de um retângulo. Em outras palavras, o agente viaja até a região destino (seu alvo atual); chegando lá, coleta os dados ou executa suas tarefas; muda seu alvo para a região de retorno; retorna até esta região; entrega os dados ou resultados e fica apto a repetir o processo ou executar outra tarefa.

Quando injetado na rede, o agente móvel inicia uma instância do processo de migração, em busca do destino, na direção da região alvo. Periodicamente, o agente móvel consulta a posição atual estimada por seu hospedeiro. Se o hospedeiro não estiver na região alvo, o agente móvel executa o algoritmo de decisão. Se os hospedeiro estiver dentro da região alvo, o agente coleta ou entrega os dados de interesse, ou executa a computação programada. Em seguida, inicia uma nova instância do processo de migração, em direção à região de retorno, que agora passa a ser a região alvo. A Figura 4.5 ilustra este esquema.

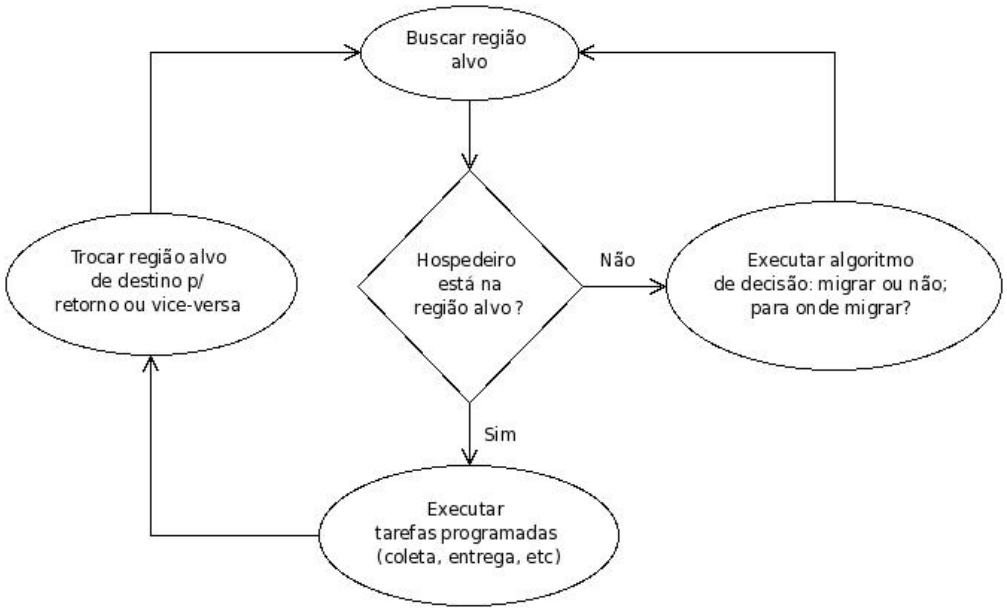



Figura 4.5: Esquema de funcionamento do agente móvel

A Figura 4.6 foi extraída de uma das simulações feitas e é utilizada aqui como um exemplo inicial. Análises mais detalhadas são feitas mais adiante.

A região destino é definida pelos pontos $A(700, 900)$ e $B(900, 700)$. A região de retorno é definida pelos pontos $C(100, 300)$ e $D(300, 100)$. O ponto $I$ corresponde às coordenadas do nó hospedeiro do agente móvel no início da simulação e o ponto $F$ às do fim da simulação. A área da simulação vai da origem ao ponto $(1000, 1000)$.

Cada vez que atinge uma das regiões alvo (destino ou retorno), a outra passa a ser o alvo e o agente tenta atingi-la. Na parte superior esquerda, quando o agente móvel se afastou do alvo, pode-se deduzir que não havia melhor alternativa do que permanecer no hospedeiro atual até que surgisse uma melhor oportunidade.

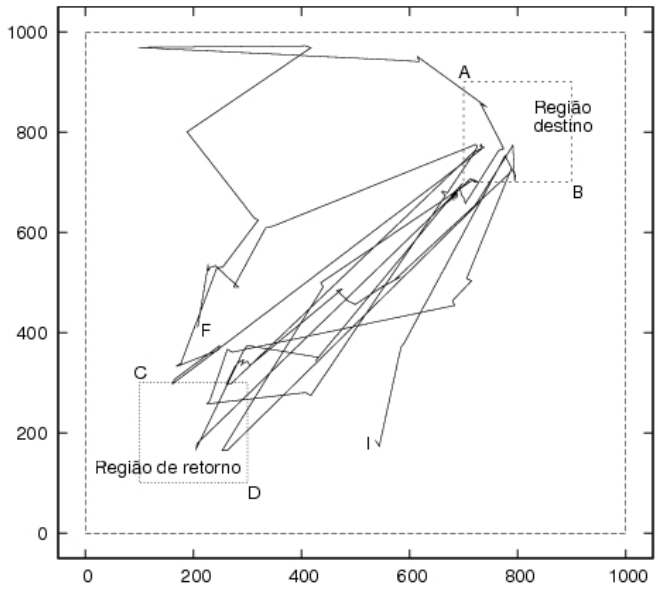



Figura 4.6: Regiões destino e de retorno e a execução de um agente móvel

Para a decisão de migração dos agentes móveis são propostos três algoritmos que são apresentados a seguir [90].

### 4.5.2 Algoritmo somente-distância

O agente móvel identifica no conjunto de vizinhos $V_i$ (incluindo o hospedeiro atual), qual deles está mais próximo do centro da região alvo. Se for o hospedeiro atual, nenhuma ação é tomada. Se for um dos vizinhos, o agente móvel aloca um canal de trabalho e inicia o processo de migração para este vizinho.

O conjunto $V_i$ contém, além da identificação de cada vizinho, a sua respectiva

estimativa de posição atual. Como somente o agente móvel detém as coordenadas da região alvo, cabe a ele computar a distância de cada vizinho até o centro desta região e decidir.

### 4.5.3 Algoritmo somente-coeficiente-angular

Enquanto executa o algoritmo de localização (Seção 3.3), o nó calcula, a cada iteração, um parâmetro chamado **tipo de movimento**. Este parâmetro assume os valores $S$(parado), $U$(para cima), $D$(para baixo), $R$(para a direita) ou $L$(para a esquerda).

Os tipos $U$ e $D$ são direções paralelas ao eixo Y, ou muito próximas. Assim, sempre que a coordenada $x$ permanece constante de um passo para outro da execução do algoritmo de previsão de localização e direção do movimento (Seção 3.3), tem-se um dos dois casos, $U$ ou $D$. Nestas situações, o coeficiente angular da direção de deslocamento foi fixado em 6.000.000 (correspondendo a ângulos não mais que $0,00001°$ afastados de 90° ou de 270°). Estes casos são isolados para tratamento dos infinitos da tangente, para evitar erros de precisão no cálculo de operações de divisão e para reduzir o consumo de ciclos de CPU também no cálculo destas operações. A rigor, a probabilidade do tipo de movimento cair nesta faixa é muito reduzida quando as simulações operam sobre mobilidade aleatória. Entretanto, para trajetórias do tipo *Manhattan* (Seção 3.3.2), esta situação ocorre muito freqüentemente.

Os tipos $R$ e $L$ estão em todo o restante do primeiro e quarto quadrantes ($R$), e segundo e terceiro quadrantes ($L$). A Figura 4.7 ilustra os tipos de movimento de um nó (os exemplos de deslocamentos $U$ e $D$, não exatamente sobre o eixo $Y$, foram exagerados para melhor visualização).

O agente móvel computa o parâmetro **tipo de movimento** relacionado à reta definida pela localização de seu hospedeiro e o centro da região alvo, isto é, a inclinação da **trajetória ótima**. Este valor é chamado de **tipo de movimento ótimo**. A inclinação da trajetória ótima é chamada de **coeficiente angular ótimo**. Se houver, dentre os vizinhos e o hospedeiro (conjunto $V_i$), um ou mais nós com o mesmo tipo de movimento que o da trajetória ótima, o agente móvel identifica aquele cuja inclinação da trajetória esteja mais próxima do coeficiente angular ótimo. Se for o seu hospedeiro, nenhuma ação é tomada. Se for um dos vizinhos, o agente

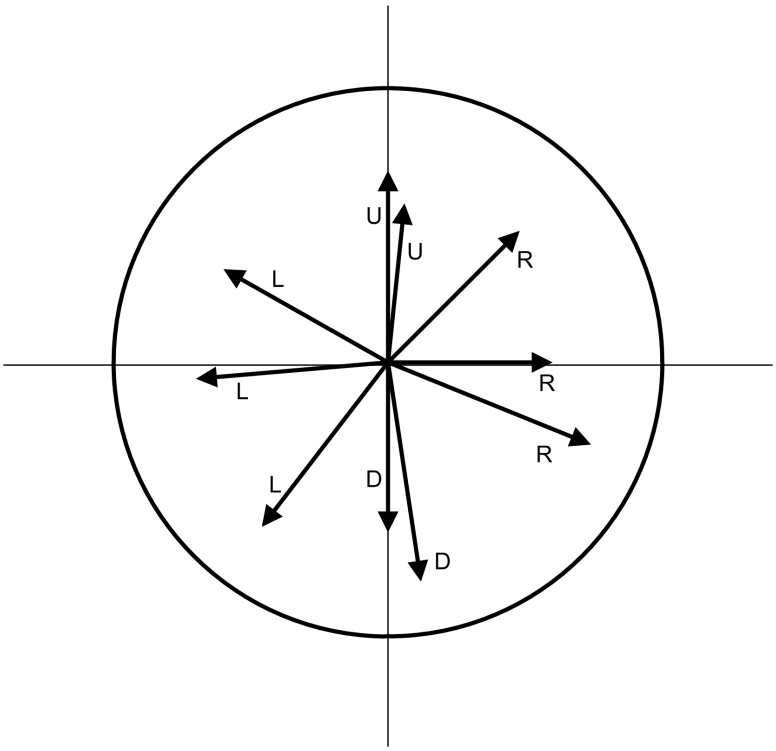



Figura 4.7: Tipos de movimento de um nó

móvel inicia o processo de migração para este nó. Se não houver nenhum vizinho com tipo de movimento igual ao da trajetória ótima, nenhum ação é tomada.

A Figura 4.8 apresenta um exemplo. Os nós $K$ e $M$ se movem quase verticalmente. Seus tipos de movimento são $U$ (para cima) e $D$ (para baixo), respectivamente. $H, Y, Z$ e $X$ se movem para a direita (tipo de movimento $R$). O nó $P$ se move para a esquerda ($L$).

O agente móvel está situado no nó $H$. O agente conhece as coordenadas $A$ e $B$ da região alvo e pode calcular as coordenadas do ponto $C$. O nó $H$ tem uma estimativa de sua localização atual. Com base nestes dados, o agente móvel pode computar o coeficiente angular $\theta\,ótimo$ e o tipo de movimento ótimo, no caso $R$. O conjunto de vizinhos $V_i$, computado pelo algoritmo 3M, contém os nós $X, Z, Y, K, P$ e $M$. Os *beacons* enviados por cada nó carregam as localizações estimadas e tipos de movimento de cada um. Com base nestes dados, o agente despreza os vizinhos $K, P$ e $M$, em função do tipo de movimento. Dentre os vizinhos com tipo de movimento $R$, a decisão de migração terá como alvo o nó $Y$, já que:

$$|\theta ótimo\ -\ \theta y| \text{ é mínimo } \forall\, \theta_i \in V_i - \{K, P, M\}.$$

No algoritmo **somente-distância**, o nó $Z$ seria o destino da migração do agente.

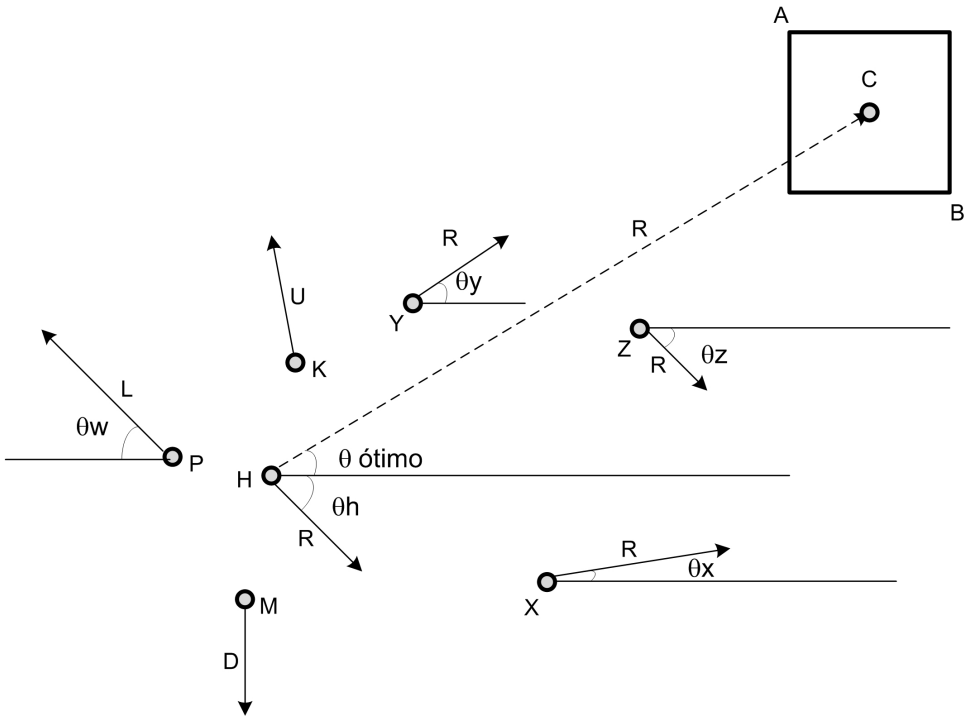



Figura 4.8: Migração do agente móvel com base somente no coeficiente angular

### 4.5.4 Algoritmo distância-e-coeficiente-angular

Se houver, no conjunto $V_i$, um ou mais nós com o mesmo tipo de movimento que o da trajetória ótima, o agente móvel executa o algoritmo **somente-distância** (Seção 4.5.2) com base apenas neste subconjunto de $V_i$. Se todos os nós em $V_i$ tiverem tipos de movimento diferentes do ótimo, o algoritmo **somente-distância** é usado sobre todo o conjunto $V_i$.

No exemplo da Figura 4.8, pelo algoritmo **distância-e-coeficiente-angular**, o nó $Z$ seria o destino da migração do agente.

### 4.5.5 Considerações finais da seção

O nó hospedeiro sempre entra no conjunto $V_i$ inicial, sobre o qual o agente móvel decide.

Nenhum algoritmo de roteamento é necessário e nenhum estado precisa ser armazenado na rede. Decisões instantâneas e baseadas em valores não cumulativos existentes nos nós permitem aos agentes móveis executar as tarefas necessárias. Os

três algoritmos são de complexidade linear *O(k)*, onde $k$ é o número de vizinhos e $k \leq n$ ($n$ é o número total de nós da rede).

O algoritmo **somente-coeficiente-angular** não pode ser aplicado a redes estáticas. Neste caso, tanto o algoritmo **somente-distância** quanto o híbrido são plenamente eficazes.

Para um deslocamento contrário ao fluxo do movimento dos sensores, um dos dois algoritmos que consideram a distância são necessários. Seja um fluxo de sensores em um sentido de uma via de tráfego de veículos, de uma correnteza ou escoamento de um fluido. Se os agentes possuem tarefas situadas no sentido contrário, não existirão trajetórias do mesmo tipo de movimento da trajetória ótima. Somente migrando contra o fluxo, com base na distância à região alvo, será possível obter sucesso.

## 4.6 Análise dos algoritmos

Para avaliação, comparação e classificação dos algoritmos, foram selecionados três parâmetros principais de análise: velocidade, alcance de comunicação de rádio e o esforço computacional envolvido. Como métricas, adotou-se o número de sucessos, que ocorrem quando o agente móvel atinge a região alvo, e o número de migrações, relacionado sobretudo com o consumo de energia. Ao final de cada simulação, calculou-se a média destes dois valores, para cada agente móvel.

Nas simulações descritas a seguir [91], o conjunto de vizinhos $V_i$ utilizado como base de decisão dos agentes móveis foi o ideal, computado pelo simulador através do método **public synchronized Vector getNeighboursSet(int nodeID)**. Este método retorna todos os vizinhos que estão ao alcance de rádio de um nó. Não se usou o conjunto de vizinhos conhecidos, computado pelo método **public Vector getNeighborsKnown()**, computado pela execução do protocolo 3M, uma vez que as referidas simulações foram executadas antes da implementação do protocolo de enlace proposto nesta tese.

### 4.6.1 O impacto da velocidade dos nós

O aumento da velocidade de deslocamento dos nós tem impacto positivo sobre os dois algoritmos que se baseiam na distância até a região alvo: **somente-distância**

e **distância-e-coeficiente-angular**. O maior dinamismo dos nós faz com que eles passem mais vezes pelas regiões alvo. É interessante observar que um alto grau de mobilidade pode ser prejudicial a muitos algoritmos de roteamento, na medida em que rotas construídas se tornarão inválidas nos instantes seguintes. Na presente proposta, e para os cenários explorados, um maior grau de mobilidade melhora o desempenho ou na pior das hipóteses não o reduz.

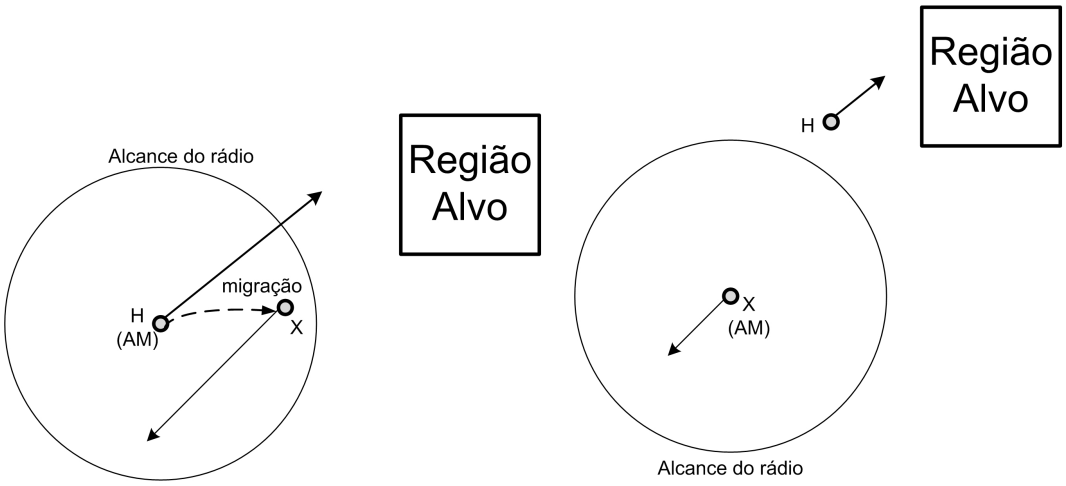
O algoritmo que considera somente o coeficiente angular não desfruta dos benefícios do maior dinamismo e chega até a ser prejudicado por ele. Como a migração ocorre para o vizinho se movendo na direção mais próxima à do coeficiente angular ótimo, conclui-se que as opções descartadas, para o cenário simulado, eram mais interessantes do que a escolhida.

Existe, entretanto, um efeito colateral da velocidade sobre o algoritmo **somente-distância**. Este algoritmo desconsidera a direção de deslocamento do nó destino, quando da decisão do agente móvel. Assim, o agente móvel pode decidir migrar para um vizinho mais próximo da região alvo, mesmo que ele esteja se afastando dela e o seu hospedeiro atual esteja indo em direção a ela. Esta decisão, combinada com a granularidade de tempo de execução (ver precisão do relógio na Seção A.2) pode representar uma migração errada e irrecuperável. Isto ocorre devido ao fato da migração de volta ao hospedeiro original não ser mais possível. Quanto menor o alcance do rádio e quanto maior a velocidade dos nós, maior a probabilidade disso acontecer.

A Figura 4.9 mostra a situação descrita. O agente móvel migra do hospedeiro $H$ para o nó $X$ com base na menor distância à região alvo (Figura 4.9(a)). No passo seguinte (Figura 4.9(b)), o ex-hospedeiro $H$ já está fora do alcance do rádio do nó $X$, impedindo a recuperação do algoritmo. Pelo algoritmo **somente-coeficiente-angular** o tipo de movimento ótimo seria $R$ e o tipo de movimento do nó $X$ seria $L$. A migração equivocada não teria acontecido.

Durante o desenvolvimento deste trabalho foi construído um simulador dos níveis de interações locais e globais. O Anexo A descreve o simulador e suas funcionalidades.

Para esta análise do impacto da velocidade, as simulações foram feitas com os seguintes parâmetros: 300 nós distribuídos e movimentados aleatoriamente, segundo




(a) Migração equivocada (b) Impossibilidade de recuperação

Figura 4.9: Migração equivocada sem possibilidade de recuperação

o modelo *Random Waypoint*; velocidade dos nós: distribuição normal entre 1 e os valores do eixo $X$ das Figuras 4.10 e 4.11; intervalo de repouso: distribuição uniforme entre 0 and $20s$; área: $500m$ x $500m$ (delimitada pelos pontos (0, 0) e (500, 500); tempo de simulação: $1.000s$; precisão do relógio (passo): $1s$; alcance do rádio: $20m$; erro do algoritmo de localização: 0%; região de destino: $A_D(350, 450), B_D(450, 350)$, onde $A_D$ é o vértice superior esquerdo e $B_D$ é o vértice inferior direito; região de retorno: $A_R(50, 150), B_R(150, 50)$ (mesmos vértices descritos anteriormente); três agentes móveis são injetados em cada nó (um executando cada tipo de algoritmo) e começam a processar no instante $5s$, executando até o fim da simulação.

As simulações deste capítulo foram feitas com um grande número de nós. Isto possibilitou a injeção de muitos agentes. Com a execução de múltiplas rodadas, os intervalos de confiança se mostraram reduzidos e não serão representados.

A Figura 4.10 mostra o ganho dos dois algoritmos baseados em distância no número médio de sucessos. O aumento mais significativo ocorre até a velocidade máxima de $30m/s$. Após este valor, observa-se uma leve queda do algoritmo **somente-distância**. Isto se deve ao problema discutido na Figura 4.9. Pela mesma razão, após $30m/s$, o algoritmo **somente-coeficiente-angular** apresenta um leve ganho de desempenho. O algoritmo **distância-e-coeficiente-angular**, por ser híbrido, combina a perda e o ganho mencionados e se mantém estável a partir dos $30m/s$. Não há perda significativa de desempenho em nenhum dos casos.

A Figura 4.11 mostra o número de migrações para cada um dos algoritmos. Este

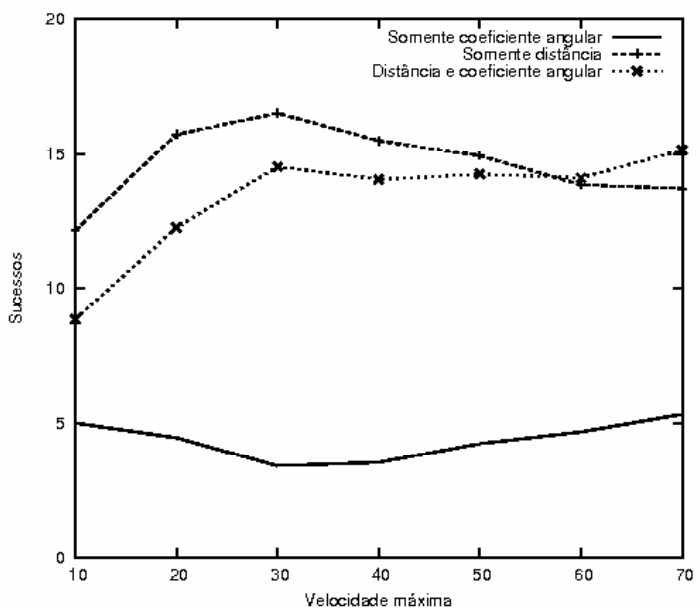



Figura 4.10: Impacto da velocidade (m/s) – Nº médio de sucessos por agente móvel

parâmetro refere-se ao consumo de energia, já que o gasto com comunicação é o mais relevante em RSSF. O algoritmo **somente-distância**, por ser "guloso", efetua muito mais migrações, devido às correções que tem que fazer quando o agente móvel migra equivocadamente. O **somente-coeficiente-angular** faz menos migrações, mas presta menos serviço (menos sucessos). O melhor custo benefício fica com o algoritmo híbrido, já que o número médio de sucessos é próximo ao do algoritmo **somente-distância** e o consumo de energia devido às migrações é bem inferior.

### 4.6.2 Análise do alcance de comunicação

O alcance de comunicação dos sensores se reflete diretamente no desempenho da rede. Como visto na Seção 4.6.1, o algoritmo **somente-distância** depende da execução de mais de uma iteração (mais de um passo) para um mesmo conjunto de vizinhos (ou um conjunto não muito diferente do anterior) para poder se recuperar a tempo das migrações equivocadas. Deste modo, quanto maior o raio de alcance do rádio de comunicação, maiores as chances de recuperação.

De um modo geral, quanto maior o raio de comunicação, maior o número de vizinhos no conjunto $V_i$ e mais acuradas as decisões dos agentes móveis. Em contrapartida, mais mensagens são trocadas e maior é o esforço computacional nos laços de decisões dos algoritmos.

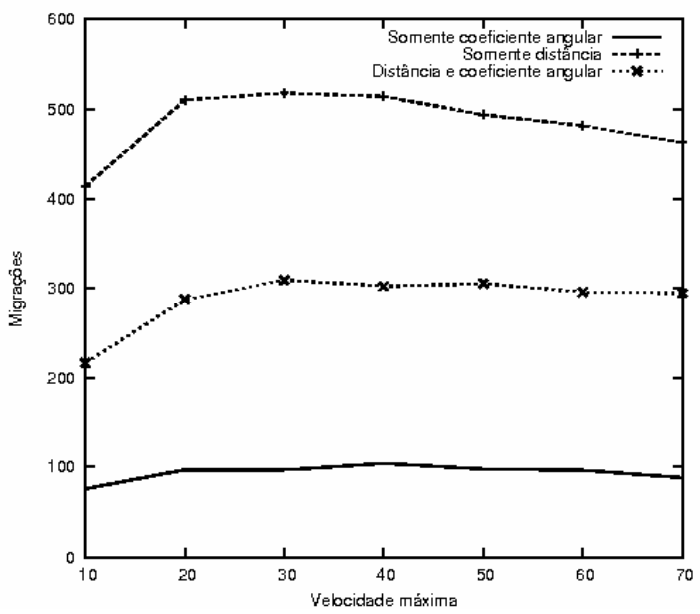



Figura 4.11: Impacto da velocidade (m/s) – Nº médio de migrações por agente móvel

Para esta análise, usou-se o mesmo cenário da Seção 4.6.1, com a velocidade máxima fixa em $30m/s$ e a faixa de comunicação variando de 10 a $90m$ de raio.

Na Figura 4.12, observa-se uma grande superioridade dos algoritmos que consideram a distância ao destino na tomada de decisão de migração, sendo o **somente-distância** o de melhor desempenho.

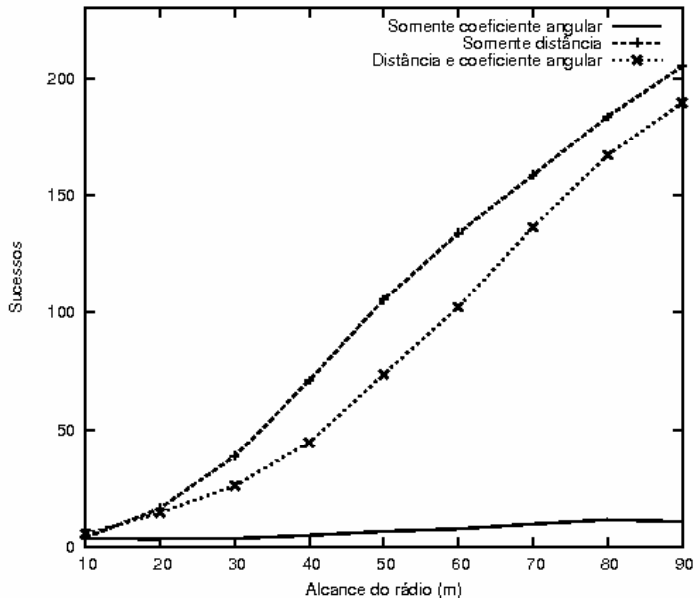



Figura 4.12: Alcance da comunicação – Nº médio de sucessos por agente móvel

O algoritmo **somente-coeficiente-angular** não atende a este cenário. O

número de sucessos não se beneficiou do aumento do raio de cobertura do rádio. Os dois algoritmos que consideram a distância se valem da possibilidade de migrações que cubram distâncias próximas a 100% do alcance do rádio. Isto permitiu a escalabilidade observada.

É interessante observar na Figura 4.13 que o algoritmo **somente-coeficiente-angular** aumentou razoavelmente o número de migrações, mesmo sem conseguir um bom número de sucessos. Já os algoritmos **somente-distância** e **distância-e-coeficiente-angular** aumentaram rapidamente o número de migrações, se valendo do maior alcance do rádio para obter mais sucessos. Entretanto, por volta de 50 e $80m$, respectivamente, os algoritmos estabilizaram as migrações e, conseqüentemente, parte do consumo de energia. Isto se deve ao fato da maior densidade de nós vizinhos já não se refletir em ganho representativo de distância percorrida. Este é um bom comportamento destas soluções.

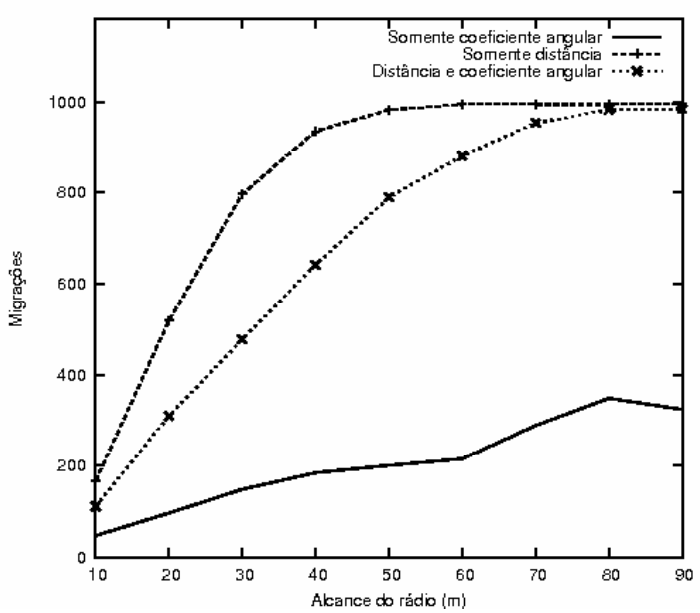



Figura 4.13: Alcance da comunicação – Nº médio de migrações por agente móvel

Devido à paridade de resultados entre os dois algoritmos de distância, o puro e o híbrido, foi extraído das simulações um fator de migrações por sucesso. Ele incorpora o custo energético, de tempo e computacional para se obter um sucesso na tarefa do agente móvel. Este fator está na Figura 4.14. Observa-se que os poucos sucessos do algoritmo **somente-coeficiente-angular** custaram mais caro que os muitos dos outros dois algoritmos, sobretudo a partir de $20m$ de raio de alcance.

Nota-se ainda que o algoritmo só de distância começa com um custo mais elevado que o híbrido, só se equiparando a ele a partir de $40m$ de raio de cobertura de rádio.

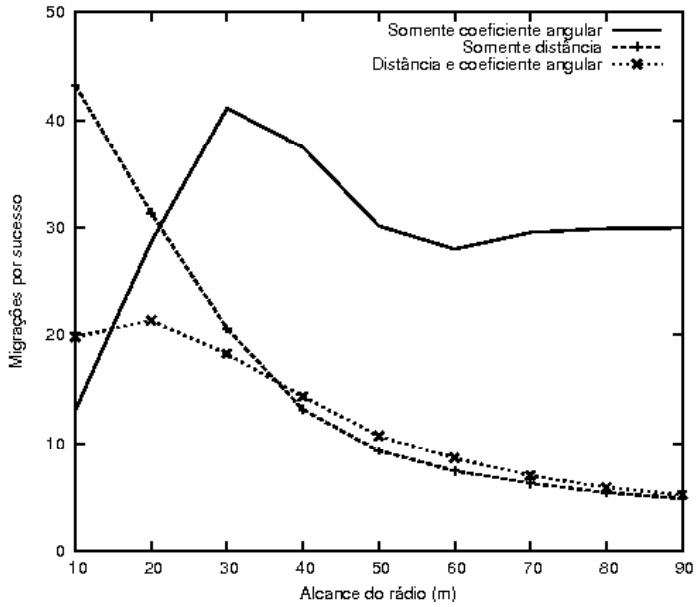



Figura 4.14: Alcance da comunicação – Nº médio de migrações para cada sucesso

### 4.6.3 Passo do relógio: esforço computacional

Os agentes móveis podem executar seus respectivos algoritmos de decisão com uma granularidade de tempo menor (mais passos por segundo).

O algoritmo de localização subjacente, as previsões de localização e direção executadas nos nós e a descoberta e manutenção de vizinhos podem ser executadas com outra granularidade.

Como visto na Seção 3.5, a manutenção de vizinhos requer uma granularidade de tempo menor para obter um bom percentual de vizinhos conhecidos e um baixo número de falsos positivos.

A estimativa de localização futura, da direção e tipo de movimento (Seção 3.3) não necessitam de alta freqüência de computação, já que as variações do movimento em frações de segundos são menos relevantes. Entretanto, o passo de execução desta interação local deve ser no mínimo igual ao passo de execução do algoritmo de decisão do agente móvel. Caso não seja, o agente móvel decidirá duas ou mais vezes com base no mesmo estado do sistema, o que é improdutivo.

As simulações foram feitas em um cenário com 700 nós distribuídos e movimentados aleatoriamente; velocidade dos nós: distribuição normal entre 1 e $30m/s$; intervalo de repouso: distribuição uniforme entre 0 and $20s$; área de $1.000m$ x $1.000m$; tempo de simulação de $1.000s$; precisão do relógio (passo): 0, 1 e 2 casas decimais (passos $1s$, $0,1s$ e $0,01s$, respectivamente); alcance do rádio de $10m$; erro do algoritmo de localização: 0%; região de destino: $A_D(700, 900), B_D(900, 700)$. $A_D$ é o vértice superior esquerdo e $B_D$ é o vértice inferior direito; região de retorno: $A_R(100, 300), B_R(300, 100)$ (mesmos vértices descritos anteriormente); 300 agentes móveis são injetados nos 300 primeiros nós, começam a processar no instante $5s$ e executam até o fim da simulação.

A Figura 4.15 mostra que a redução do passo de $1s$ para $0,1s$ gerou um ganho considerável de rendimento para os três algoritmos, sobretudo para o **somente-distância**. Vale lembrar, entretanto que este aumento significa multiplicar por 10 todas as tarefas relativas à computação de localização, direção e tipo de movimento. Já uma nova redução de uma casa decimal, para $0,01s$, produziu um ganho menos significativo.

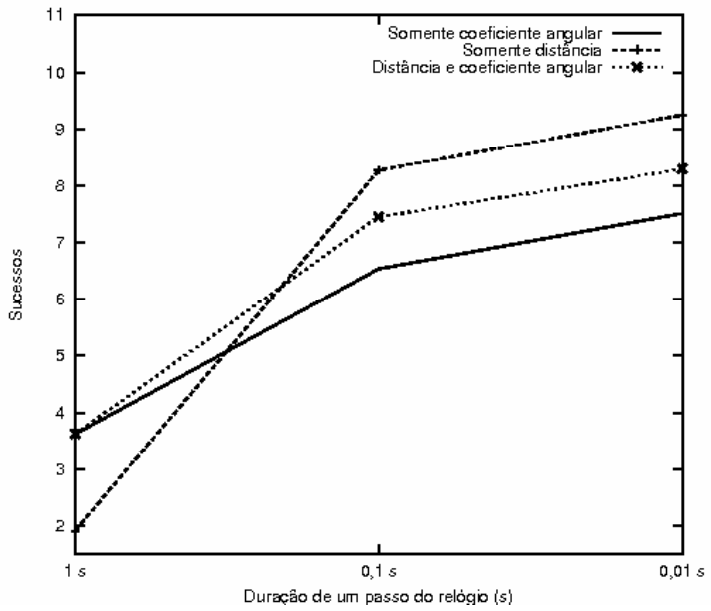



Figura 4.15: Granularidade de tempo – Nº médio de sucessos por agente móvel

Na primeira redução, o aumento do número de migrações foi moderado para o algoritmo **somente-coeficiente-angular**, médio para o algoritmo híbrido e elevado para o **somente-distância** (Figura 4.16). Este último se valeu das iterações

adicionais para corrigir erros de migração. Na segunda redução, os algoritmos de distância pouco aumentaram ou até diminuíram as migrações, apesar do aumento de sucessos.

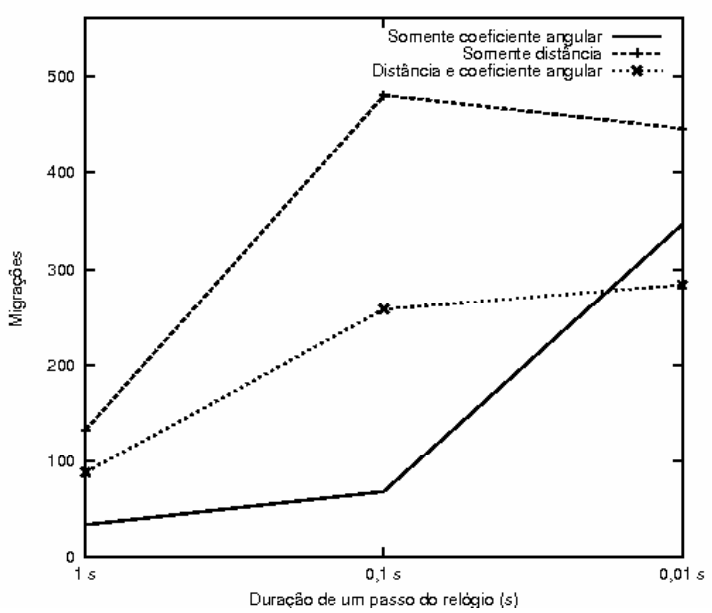



Figura 4.16: Granularidade de tempo – Nº médio de migrações por agente móvel

O algoritmo do coeficiente angular puro teve um aumento exagerado no número de migrações. A causa foi identificada. Trata-se de uma oscilação no algoritmo. O agente móvel fica migrando sucessivamente entre dois nós vizinhos, até que eles saiam do alcance um do outro ou até que um deles mude a inclinação do seu deslocamento. Esta oscilação ocorre quando a trajetória ótima está entre as trajetórias de dois nós móveis. Após cada migração, quando um novo valor do coeficiente angular ótimo é calculado, a trajetória do outro nó se torna mais vantajosa sob o ponto de vista do algoritmo. Ao migrar de volta para o antigo hospedeiro, a situação se inverte, e assim por diante. Um trecho da trajetória de um agente móvel com o comportamento descrito está na Figura 4.17. No caso, em um intervalo de $2,79s$, o agente efetuou 279 migrações entre os mesmos dois nós. Para a granularidade de tempo ajustada para $0,1s$, as migrações caem 10 vezes. Seriam 27 migrações, o que ainda é indesejável. Para granularidade em $1s$, como foram feitas as simulações das duas seções anteriores, o comportamento não causa prejuízo.

Está em estudo uma solução para esta situação. Ela precisa controlar a migração para o último hospedeiro, impedindo-a ou permitindo apenas um número fixo delas.

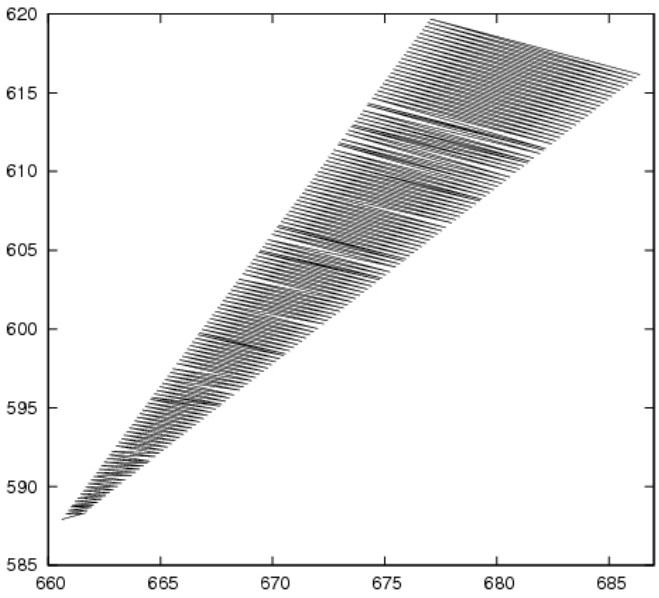



Figura 4.17: Anomalia de um agente móvel migrando sucessivamente entre 2 nós

Isto implicará mais um estado armazenado no sistema (nos agentes) e mais uma ramificação na árvore de decisões do algoritmo.

Testes preliminares foram efetuados em um pequeno conjunto de agentes e em apenas um cenário de simulação. Os testes foram feitos com o algoritmo **somente-coeficiente-angular**. A técnica utilizada foi a de não impedir nenhuma oscilação, a menos que o ganho seja significativo. Assim, após uma migração, o agente móvel armazena o hospedeiro anterior e só migra de volta para ele se o ganho de direção estiver acima de um patamar.

Foram selecionados 11 agentes. Alguns que apresentaram a oscilação, outros que não apresentaram. No segundo grupo, alguns com muitos sucessos e poucas migrações e outros com rendimento mediano. A simulação efetuada permitiu retorno ao mesmo hospedeiro apenas para um ganho de direção maior do que aproximadamente[1] 5°. Isto equivale a dizer que o coeficiente angular do deslocamento do destinatário atual está mais próximo do ótimo em mais de um décimo (0,1) do que o último estava.

As simulações mostraram redução do número de migrações na maioria dos casos, sobretudo com os agentes com muitas migrações, que são aqueles que apresentaram

[1] A relação da tangente com os ângulos varia à medida que os ângulos aumentam. Aumentar 0,1 na tangente tem mais impacto em ângulos próximos de zero do que naquelas próximos de 90°

a oscilação. Por outro lado, houve redução do número de sucessos em alguns casos e aumento em outros. A média de sucessos variou muito pouco. Os dados estão nas Tabelas 4.3 e 4.4. Apesar das simulações não totalmente conclusivas, parece ser uma proposta promissora. Muitas oscilações permaneceram, em função do ganho de direção de $0,1$ ser pequeno para ângulos acima de $70°$. As oscilações permaneceram em rotas muito verticais.

Outra proposta foi sondada. Quando se detecta a oscilação, é inserido um amortecimento de alguns ciclos de relógio. Assim, sempre o agente decide migrar para o nó de onde veio, a migração não é efetuada e o agente bloqueia as migrações para este nó durante um período. Migrações para outros nós são permitidas a qualquer momento.

As simulações foram feitas no mesmo cenário mencionado no primeiro teste. O período de amortecimento foi configurado para 10 ciclos de relógio. A proposta reduziu ainda mais o número de migrações e a média de sucessos chegou até a aumentar. Este aumento, entretanto, não é conclusivo, já que o universo de testes é muito reduzido. De fato, não há nenhum motivo para a melhora no número de sucessos. A técnica de redução da oscilação não tem nenhuma relação com aumento do número de sucessos. Ao modificar o critério de decisão, os agentes tomam caminhos diferentes e toda a sua trajetória futura é modificada de maneira imprevisível. A Figura 4.18 mostra o efeito do amortecimento e, ao mesmo tempo, mostra como o resultado final é uma rota totalmente diferente daquela seguida no caso do algoritmo original. No caso analisado, a região alvo está acima dos nós, nas coordenadas $A(700, 900), B(900, 700)$ e os dois nós da oscilação estão se deslocando para baixo, um para a esquerda e outro para a direita.

As Tabelas 4.3 e 4.4 mostram os dados das simulações.

Na Tabela 4.3 são listadas as médias de sucessos e migrações dos 11 agentes móveis testados, nos três casos: algoritmo **somente-coeficiente-angular** original; primeira proposta, que evita a oscilação e considera o ganho de direção; e segunda proposta, que introduz um amortecimento na oscilação. Vale mencionar que alguns agentes testados não haviam apresentado as referidas oscilações.

Um estudo detalhado e a proposta de uma solução definitiva estão relacionadas como trabalhos futuros.

| | Nº de sucessos | Nº de migrações | Migrações por sucesso |
|---|---|---|---|
| Algoritmo original | 8, 18 | 489 | 74 |
| Primeira proposta | 8, 09 | 156 | 20 |
| Segunda proposta | 8, 64 | 96 | 12 |

Tabela 4.3: Dados preliminares de duas propostas de solução para a oscilação nas migrações dos agentes (valores médios)

| | Algoritmo original | | | Proposta 1 | | | Proposta 2 | | |
|---|---|---|---|---|---|---|---|---|---|
| Ag. | Suc. | Migr. | Migr./suc. | Suc. | Migr. | Migr./suc. | Suc. | Migr. | Migr./suc. |
| 27 | 7 | 1236 | 177 | 7 | 268 | 38 | 6 | 151 | 25 |
| 34 | 11 | 349 | 32 | 7 | 173 | 25 | 9 | 79 | 9 |
| 46 | 9 | 68 | 8 | 9 | 102 | 11 | 9 | 80 | 9 |
| 50 | 7 | 323 | 46 | 9 | 76 | 8 | 9 | 64 | 7 |
| 58 | 11 | 263 | 24 | 11 | 290 | 26 | 11 | 81 | 7 |
| 66 | 7 | 1230 | 176 | 7 | 266 | 38 | 9 | 165 | 18 |
| 100 | 7 | 320 | 46 | 6 | 46 | 8 | 6 | 46 | 8 |
| 118 | 11 | 321 | 29 | 8 | 55 | 7 | 11 | 73 | 7 |
| 130 | 4 | 879 | 220 | 7 | 265 | 38 | 9 | 166 | 18 |
| 150 | 7 | 320 | 46 | 9 | 76 | 8 | 7 | 72 | 10 |
| 158 | 9 | 68 | 8 | 9 | 102 | 11 | 9 | 80 | 9 |

Tabela 4.4: Dados preliminares de duas propostas de solução para a oscilação nas migrações dos agentes (valores arredondados)

Na Figura 4.19, confirma-se, através do custo de cada sucesso, o bom desempenho do algoritmo **somente-coeficiente-angular** nestes cenários, para os passos $1s$ e $0,1s$. Para o passo $0,01s$, a oscilação mencionada se refletiu nas migrações deste algoritmo, mas ainda assim ele permaneceu mais econômico que o **somente-distância**.

### 4.6.4 Considerações finais da seção

O algoritmo **somente-distância** tende a ser mais agressivo que os demais e na maioria das vezes atinge um maior número de sucessos. Entretanto, tem maior tendência a efetuar um maior número de migrações. O **somente-coeficiente-angular** se com-

porta de maneira mais econômica em termos de consumo de energia (migrações), mas não escala em algumas situações. O algoritmo híbrido consegue incorporar as vantagens dos outros dois, mas carrega os problemas de ambos também.

Em ambientes sem mobilidade, apenas a localização (distância) será eficaz, não tendo sentido o conceito de direção do movimento.

Não há como generalizar um único algoritmo para todas combinações dos parâmetros aqui analisados. Cada um deles terá melhor desempenho em determinadas situações.

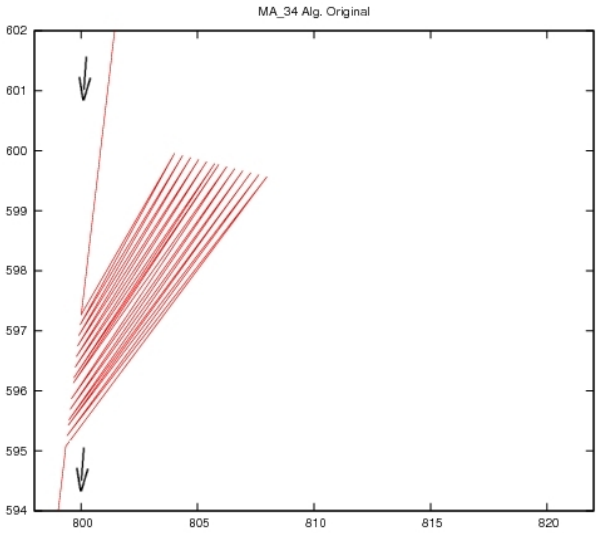
As Figuras 4.20 e 4.21 ilustram os conclusões recém mencionadas. Elas resumem simulações feitas com os mesmos cenários da Seção 4.6.3, com o número de nós variando de 10 a 800. Em linhas gerais, os algoritmos baseados em distância e as simulações com passo $0,1s$ apresentam mais sucessos e mais migrações.

Ainda na Figura 4.20, observa-se o algoritmo **somente-distância** sem nenhuma reação ao aumento do número de nós. A combinação de velocidades variando entre 1 e $30m/s$ com o alcance de rádio em $10m$ tornou este algoritmo inviável.

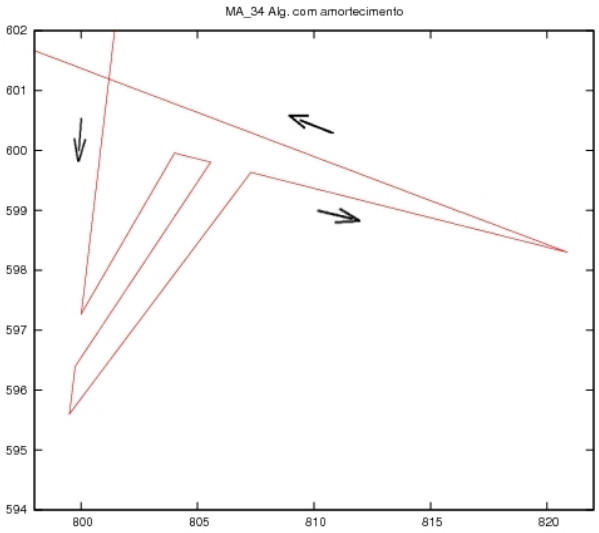
## 4.7 Além das RSSF

As redes veiculares [92, 93, 94] (*Vehicular Ad hoc NETwork* – VANET), área de pesquisa emergente, apresentam diversas similaridades à proposta desta tese. Nestas redes, a localização do nó será menos custosa e mais precisa; os recursos computacionais e energéticos serão mais abundantes e as velocidades e deslocamentos muito intensos. Por estas razões, as técnicas aqui apresentadas mostram grande potencial de uso nas redes veiculares.

As aplicações do presente trabalho às VANETs se voltam mais para rápidas interações do que para fluxos de dados. Estas rápidas interações podem ser exemplificadas como envio de informações de segurança de/para os veículos, informações de condições de tráfego, anúncios de acidentes e interrupções na via [93]. Adequam-se, ainda, à descoberta de vizinhos e comunicação entre veículos. Técnicas de uso de múltiplos canais, como a proposta nesta tese, vêm sendo muito relacionadas às redes veiculares [95].




(a) Algoritmo original




(b) Algoritmo com amortecimento

Figura 4.18: Oscilação do agente móvel sem tratamento e com amortecimento

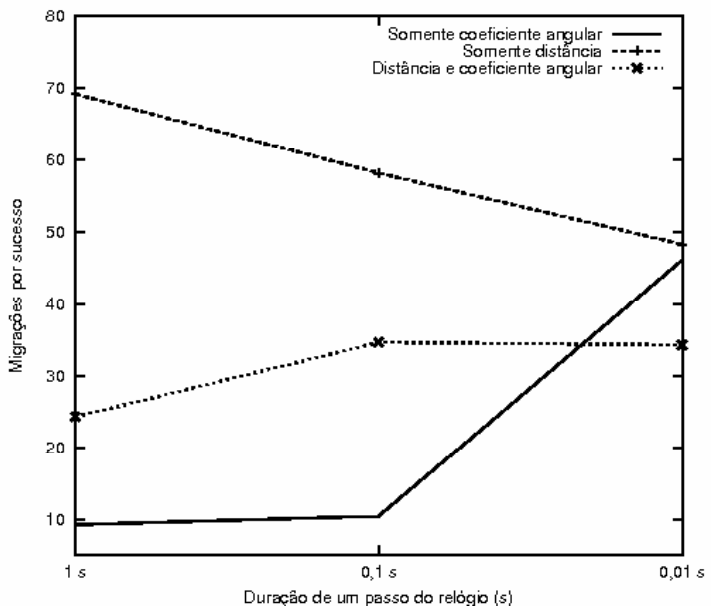



Figura 4.19: Granularidade de tempo – Nº médio de migrações para cada sucesso

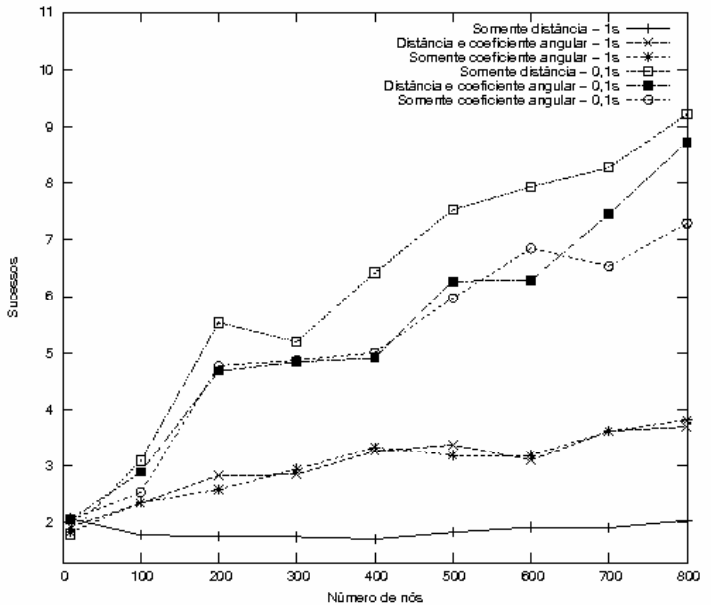



Figura 4.20: Nº médio de sucessos por agente móvel

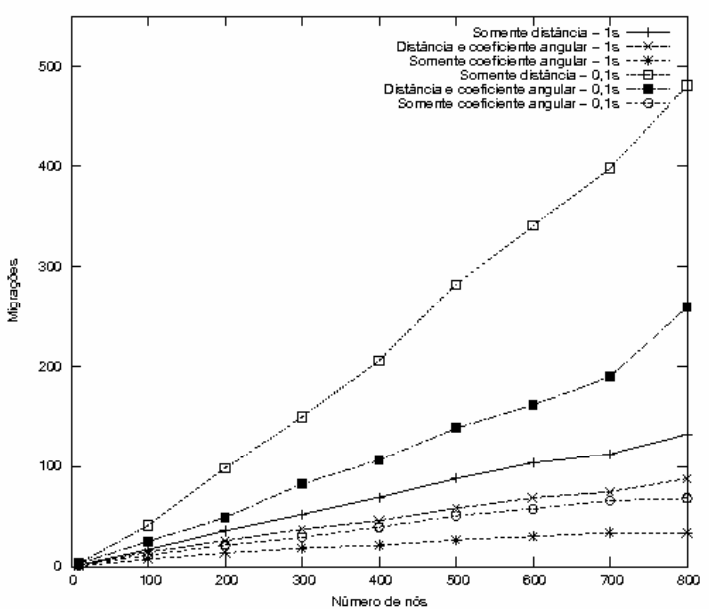



Figura 4.21: Nº médio de migrações por agente móvel

# Capítulo 5

# Conclusão

Este trabalho apresenta um esquema em dois níveis para suporte à mobilidade em RSSF. A proposta é leve e bem dimensionada aos escassos recursos dos sensores atuais. O projeto tem dois níveis, sendo estes mapeados em apenas três camadas do modelo de referência ISO-OSI: física, enlace e aplicação. O esquema se mostra eficiente e torna possível a adoção de técnicas distintas no NIL e no NIG [96].

Os estudos realizados mostram o grande interesse por pesquisas em cenários dotados de mobilidade. Por outro lado, observou-se que os ambientes com mobilidade generalizada e intensa são pouco explorados. Tanto nos protocolos de enlace estudados, quanto nas técnicas de suporte à mobilidade e roteamento em ambientes móveis, sempre se deparou com simulações, implementações e modelos voltados para mobilidade reduzida e parcial. Esta tese sempre tratou e testou a mobilidade em condições superiores àquelas encontradas na literatura.

A presente proposta tem uma dependência fundamental em um algoritmo de localização subjacente. A revisão bibliográfica demonstrou que existem muitas técnicas de localização disponíveis e muitos protocolos que utilizam e pressupõem localização dos nós da rede de sensores.

Foi demonstrado que as técnicas de predição de localização e da direção do movimento, propostas por esta tese, apresentam bons resultados. Observou-se que o modelo proposto, de complexidade sub-linear, produz resultados suficientes para os processos de tomada de decisão dos níveis superiores da proposta.

O protocolo 3M, proposto para o nível de enlace e para a manutenção do conjunto de vizinhos, se mostrou eficiente em ambientes de mobilidade até então pouco

explorados para as redes de sensores. Com baixa ocupação do canal, obtem-se um alto índice de sucesso na descoberta de vizinhos e pequenos índices de falsos positivos. A alocação dinâmica de canais de freqüência de baixa taxa de transmissão é adequada às RSSF. Este esquema de múltiplos canais se torna ainda mais eficaz quando acoplado aos agentes móveis do NIG.

O protocolo 3M é totalmente distribuído, sem coordenador, mestre ou pontos de acesso. Não há necessidade de nenhum nível de sincronismo (local ou global), nem o uso de *slots* de tempo para transmissão. Estes fatores, aliados à análise conceitual, comparativa e simulada que foi feita, mostram que o mesmo é apropriado aos cenários amplamente dinâmicos de que trata esta tese. O sucesso do protocolo se baseia na sua simplicidade, especificidade e tem como custo principal a escuta permanente do canal de controle.

Conclui-se para os cenários explorados, através das simulações, que o uso conservador de valores do TTL dos vizinhos, menor do que o intervalo de anúncio de presença, é adequado aos ambientes de alta mobilidade. Ao custo de uma menor qualidade no conhecimento dos vizinhos, evita-se o risco de iniciar comunicações de rádio sem chances de sucesso.

O NIG deve estar sobreposto a uma rede com nós cientes de sua localização e nós que executam o protocolo 3M. Partindo deste pressuposto, as interações globais terão êxito em cenários com nós dotados de mobilidade generalizada e não controlada. Constatou-se que é viável a aplicação de técnicas de mobilidade controlada sobre ambientes de mobilidade aleatória. Os agentes conseguem se valer da mobilidade casual dos nós e cumprem suas tarefas. Enquanto o aumento da mobilidade se constitui em custo adicional, gera mais falhas ou até inviabiliza muitos protocolos de roteamento, observa-se que o esquema em dois níveis proposto se vale da mobilidade, obtendo, em geral, melhor desempenho.

As análises feitas nos três algoritmos de decisão dos agentes móveis mostraram que tanto a posição quanto o direção do movimento devem ser consideradas. De um modo geral, os algoritmos baseados na distância até a região alvo são mais eficientes. Entretanto, existem pontos de ruptura, baseados na combinação da velocidade dos nós e no raio de cobertura do rádio, que tornam o quesito direção e tipo de movimento fundamentais na tomada de decisão.

Observa-se que, como no NIL, o NIG é totalmente distribuído, descentralizado e assíncrono. Conclui-se, ao longo deste trabalho, que estas características são fundamentais nas soluções que pretendam atender às redes totalmente móveis.

Como na previsão de localização e no protocolo 3M, nos agentes móveis os algoritmos são de custo computacional reduzido e proporcional ao número de vizinhos. Todas as propostas desta tese são escaláveis, na medida em que não são usadas informações globais que dependam do número de nós da rede, do comprimento das rotas ou de outros estados globais do sistema. Todas as interações e decisões são locais e baseadas no conjunto momentâneo de vizinhos. Não há iterações computacionais que devam convergir, a computação distribuída produz o conhecimento e os argumentos às decisões que produzirão o cumprimento das tarefas globais.

Foi possível concluir, ao longo deste trabalho, que a construção de um simulador complexo e voltado para um ambiente de interesse específico é viável. Existe uma tendência atual de uso da linguagem Java para este fim. Esta linguagem se mostrou muito ágil na fase de desenvolvimento e muito eficiente em tempo de execução. O modelo de múltiplas linhas de execução (*multithread*), com *threads* mapeando entidas reais para o interior do simulador, foi fator de simplificação no desenvolvimento, compreensão, expansão e manutenção do sistema. O núcleo de controle do relógio do sistema se mostra um item de projeto eficiente e versátil. Conclui-se, através de testes detalhados, que o simulador reproduz com fidelidade os experimentos desejados.

Por fim, observa-se que é muito severa a tarefa de suportar mobilidade generalizada através de sensores tão simples. Entretanto, este trabalho mostra a viabilidade de fazê-lo e deixa muitos direcionamentos para aperfeiçoamentos e modificações nos esquemas propostos.

A seguir serão relacionados os trabalhos futuros que se pretende levar adiante nesta pouco explorada área de pesquisa.

## 5.1 Trabalhos futuros

### 5.1.1 Parâmetros $\alpha$ e $\gamma$ nas previsões de localização do nó

Conforme exposto na Seção 3.3, os parâmetros $\alpha$ e $\gamma$ são utilizados para dar maior ou menor peso ao último valor calculado, em comparação ao histórico dos valores estimados ao longo do tempo. As simulações foram feitas com o valor $0,5$ para ambos. Valores apropriados para cada perfil de deslocamento, velocidade e trajetória serão investigados no futuro. Os agentes móveis podem difundir os melhores valores ou modificar os valores em uso.

### 5.1.2 Adaptabilidade dos agentes móveis

Em muitas situações, pode ser interessante a adaptação dos agentes móveis às condições da rede. Uma etapa, já em estudo, que pretende-se implementar é a adaptabilidade dos agentes móveis em tempo de execução. Dentro deste contexto, quatro ações dos agentes serão exploradas:

1. **Fusão**: agentes com tarefas iguais ou sobrepostas podem se fundir em um só e prosseguir na execução.

2. **Adaptação**: diante de estímulos ambientais, o agente muda sua forma de agir.

3. **Plágio**: observando outros agentes de maior sucesso, o agente copia parâmetros e comportamentos, e passa a agir como eles.

4. **Reprodução**: agentes se combinam e geram descendentes. Neste caso, deve haver um mecanismo de seleção natural [97] para manter o número de agentes controlado e para privilegiar os mais adaptados. Está sendo modelada a adaptação de técnicas de algoritmos genéticos para um "modelo distribuído de algoritmos genéticos leves". Neste caso podem ocorrer mutações aleatórias na busca de maior eficiência.

Alguns quesitos podem formar o "genoma" dos agentes para a adoção de uma das quatro ações relacionadas.

- Muitos sucessos e poucas migrações: boa característica;
- Usar um canal de trabalho com colisão: característica ruim.

Comportamentos que podem ser modificados, copiados ou passados aos descendentes:

- Mudar de tipo de algoritmo em uso;
- Ajustar o ângulo de visada (horizonte de eventos) do agente nas decisões de migração. Atualmente, os quatro tipos de movimento consideram apenas dois hemisférios (180°) (**L** e **R**), além de parado (**S**), para cima (**U**) a para baixo (**D**). Mais opções de tipo de movimento podem ser investigadas;
- Granularidade de tempo nas decisões. Um agente decidindo de 1 em 1 $s$ pode estar se saindo melhor que outro decidindo de $0,5$ em $0,5$ $s$, por exemplo;
- Parâmetros relacionados à futura solução para a oscilação nas migrações (Seções 4.6.3 e 5.1.4).

### 5.1.3 Evoluções no simulador

Com relação ao simulador, pretende-se implementar alguns modelos de propagação para a camada física.

O uso de interface gráfica pode ser um obstáculo à execução remota e muitas vezes torna exaustiva a execução de muitas rodadas de simulações, com a inserção repetida dos mesmos dados ou dados com pequenas variações. Uma alternativa é passar um parâmetro que iniba a execução das telas gráficas. A leitura dos parâmetros seria feita a partir de um arquivo de entrada.

Outra evolução pretendida para o simulador é a implementação de *threads* no nível do núcleo do sistema operacional. O objetivo é desfrutar do paralelismo real (mais de um núcleo de processamento) e poder utilizar plenamente os recursos de ambientes de computação paralela, como *clusters*. O paralelismo já foi identificado em uma plataforma, como discutido na Seção A.4, mas necessita ser explorado de forma mais geral.

### 5.1.4 Oscilações nas migrações

Como visto na seção 4.6.3, pretende-se fazer um estudo detalhado da oscilação nas migrações dos agentes móveis. O objetivo é propor e testar uma ou mais soluções.

### 5.1.5 Implementação real

Está em fase inicial um projeto de especificação de uma plataforma real para a implementação do esquema proposto.

### 5.1.6 Análise do erro de localização

Todo algoritmo de localização produz resultados com alguma margem de erro. O simulador desenvolvido para esta tese incorpora o parâmetro relacionado ao erro de localização em todas as rotinas ligadas à localização e movimentação dos nós. As simulações realizadas não consideraram erro de localização e este parâmetro foi configurado com o valor zero.

Para os algoritmos baseados na distância à região alvo, o erro de localização não apresenta impacto na lógica dos algoritmos. Basta a execução de simulações para avaliar seu impacto.

No caso dos algoritmos que consideram a direção do deslocamento, um erro de localização pode fazer com que o algoritmo identifique uma direção de movimento equivocada. É necessário introduzir, nos algoritmos que usam o coeficiente angular, mecanismos que considerem erros de localização e mantenham seu funcionamento correto. Os parâmetros $\alpha$ e $\gamma$ tem relação com possíveis soluções para este problema.

# Referências Bibliográficas


[1] RUIZ, L. B., CORREIA, L. H. A., VIEIRA, L. F. M., et al., "Arquiteturas para Redes de Sensores Sem Fio", *XXII SBRC*, Maio 2004, Brasil.

[2] FOK, C.-L., ROMAN, G.-C., LU, C., "Rapid Development and Flexible Deployment of Adaptive Wireless Sensor Network Applications", *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pp. 653–662, 2005.

[3] PEREIRA, M. R., DE AMORIM, C. L., DE CASTRO, M. C. S., "Tutorial sobre Redes de Sensores", 2001, Disponível em: http://magnum.ime.uerj.br/cadernos/cadinf/vol14/3-clicia.pdf.

[4] CORSON, S., MACKER, J., "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", *Request for Comments: 2501*, Janeiro 1999.

[5] CAMPOS, C. A. V., DE MORAES, L. F. M., "Caracterizando a mobilidade de usuários e avaliando seu impacto no roteamento das redes móveis *ad hoc*". In: *WCSF*, p. 127, 2004.

[6] TILAK, S., ABU-GHAZALEH, N. B., HEINZELMAN, W., "A taxonomy of wireless micro-sensor network models", *ACM SIGMOBILE Mobile Computing and Communications Review*, v. 6, n. 2, pp. 28–36, Abril 2002.

[7] HU, L., EVANS, D., "Localization for Mobile Sensor Networks", *MobiCom '04*, pp. 45–57, 2004, ACM.

[8] ANTOL, J., CALHOUN, P., FLICK, J., et al., "Low Cost Mars Surface Exploration: The Mars Tumbleweed", Agosto 2003, NASA Langley Research Center. NASA/TM-2003-212411.

[9] LOUREIRO, A. A. F., NOGUEIRA, J. M. S., RUIZ, L. B., et al., "Redes de Sensores Sem Fio", *XXI SBRC*, pp. 179 – 226, 2003, Brasil.

[10] DUMIAK, M., "Robocopters Unite!" *IEEE Spectrum*, v. 46, n. 2, pp. 10, Fevereiro 2009.

[11] CHOKSI, A., MARTIN, R. P., NATH, B., et al., *Mobility Support for Diffusion-Based Ad-Hoc Sensor Networks*, Technical report, Department of Computer Science, Rutgers University, Abril 2002.

[12] SHAH, R. C., ROY, S., JAIN, S., et al., "Data MULEs: Modeling a Three-tier Architecture for Sparce Sensor Networks", Maio 2003, IEEE SNPA Workshop.

[13] SHAH, R. C., ROY, S., JAIN, S., et al., "Data MULEs: Modeling and Analysis of a Three-tier Architecture for Sparse Sensor Networks", *Ad Hoc Networks*, v. 1, n. 2-3, pp. 215–233, Setembro 2003.

[14] HUANG, Q., BHATTACHARYA, S., LU, C., et al., "FAR: Face-aware routing for mobicast in large-scale sensor networks", *ACM Trans. Sen. Netw.*, v. 1, n. 2, pp. 240–271, 2005.

[15] SOHRABI, K., GAO, J., AILAWADHI, V., et al., "Protocols for self-organization of a wireless sensor network", *Personal Communications, IEEE*, v. 7, n. 5, pp. 16–27, 2000.

[16] KOHVAKKA, M., SUHONEN, J., KUORILEHTO, M., et al., "Energy-efficient neighbor discovery protocol for mobile wireless sensor networks", *Ad Hoc Netw.*, v. 7, n. 1, pp. 24–41, Novembro 2007.

[17] PHAM, H., JHA, S., "Addressing mobility in wireless sensor media access protocol". In: *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, pp. 113–118, 2004.

[18] PHAM, H., JHA, S., "An adaptive mobility-aware MAC protocol for sensor networks (MS-MAC)". pp. 558–560, Oct. 2004.

[19] INTANAGONWIWAT, C., GOVINDAN, R., ESTRIN, D., "Directed diffusion: a scalable and robust communication paradigm for sensor networks". In: *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 56–67, ACM Press: New York, NY, USA, 2000.

[20] BARROS, R. Q., NOGUEIRA, J. M., "Estudo de Métodos de Localização de Nós em Redes de Sensores Sem Fio", *VII semana da pós-graduação DCC - UFMG*, 2003.

[21] DE SOUZA, S. C. B., *Planejamento de trajetória para um robô móvel com duas rodas utilizando um algoritmo A-Estrela modificado*, Master's Thesis, Programa de Pós-graduação em Engenharia Elétrica, COPPE/UFRJ, Dezembro 2008.

[22] HOWARD, A., MATARIC, M. J., SUKHATME, G. S., "An Incremental Self-Deployment Algorithm for Mobile Sensor Networks", 2002, Kluwer Academic Publishers.

[23] BACHMAYER, R., LEONARD, N. E., "Vehicle Networks for Gradient Descent in a Sampled Environment", *Proc. 41st IEEE Conf. Decision and Control*, v. 6, n. 2, 2002.

[24] VAHDAT, A., BECKER, D., "Epidemic Routing for Partially-Connected Ad Hoc Networks", Abril 2000, Technical Report CS-200006, Duke University, April 2000.

[25] BERTINI, L., LOQUES, O., LEITE, J., "Replicação de Dados em Redes Ad Hoc para Sistemas de Apoio em Situações de Desastre", *XXIII Simpósio Brasileiro de Redes de Computadores*, Maio 2005, Fortaleza, Brasil.

[26] WIKIPÉDIA, "pH", http://pt.wikipedia.org/wiki/PH, acesso em Março de 2009.

[27] KALMAN, R. E., "A New Approach to Linear Filtering and Prediction Problems", *Transactions of the ASME–Journal of Basic Engineering*, v. 82, n. Series D, pp. 35–45, 1960.

[28] JABOUR, F. C., JABOUR, E. G., PEDROZA, A. C. P., "Um esquema em duas camadas para suporte a mobilidade em redes de sensores sem fio", *XXV Simpósio Brasileiro de Telecomunicações - SBrT*, 2007.

[29] JABOUR, F. C., JABOUR, E. G., PEDROZA, A. C. P., "Mobility support for wireless sensor networks", *3rd IEEE European Conference on Smart Sensing and Context (EuroSSC)*, Outubro 2008, publicado como pôster.

[30] "Selection Procedures for the Choice of Radio Transmission Technologies of the UMTS (TS30.03 v3.2.0)", TS 30.03 3GPP, Abril 1998.

[31] DAY, J. D., ZIMMERMANN, H., "The OSI Reference Model". In: *Proc. of the IEEE*, v. 71, pp. 1334–1340, IEEE, Dezembro 1983.

[32] "ISO/IEC 7498-1:1994, Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model", 1994.

[33] "Institute of Electrical and Electronics Engineers", http://www.ieee.org/, acesso em Abril de 2009.

[34] POSTEL, J., "Internet Protocol", RFC 791 (Standard), Setembro 1981, Updated by RFC 1349.

[35] PLUMMER, D., "Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", RFC 826 (Standard), Novembro 1982, Updated by RFCs 5227, 5494.

[36] "IEEE Standard 802, part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)", Outubro 2003, IEEE Computer Society.

[37] "IEEE Standard 802, part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)", Setembro 2006, IEEE Computer Society.

[38] MYERS, S., "ZigBee/IEEE 802.15.4", 2007, Electrical and Computer Engineering University of Wisconsin Madison.

[39] "Getting Started with ZigBee and IEEE 802.15.4", Fevereiro 2008, Daintree Networks.

[40] DUTTA, P., CULLER, D., "Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications". In: *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 71–84, ACM: New York, NY, USA, 2008.

[41] JABOUR, E. G., JABOUR, F., PEDROZA, A., "CTCP: Analisando a Entrega, Latência e Consumo de Energia". In: *7th International Information and Telecommunication Technologies Symposium - I2TS*, Foz do Iguaçu, Brasil, Dezembro 2008.

[42] JABOUR, E. G., JABOUR, F. C., PEDROZA, A., "CTCP: Reliable Transport Control Protocol for Sensor Networks". In: *2008 Fourth International Conference on Intelligent Sensors, Sensor Nertworks and Information Processing*, Sydney, Australia, Dezembro 2008.

[43] JABOUR, E. G., JABOUR, F. C., PEDROZA, A., "Collaborative Transport Control Protocol". In: *Proceedings of IEEE International Conference on Computer and Electrical Engineering – ICCEE*, Phuket Island, Thailand, Dezembro 2008.

[44] JABOUR, E. G., JABOUR, F. C., PEDROZA, A., "Protocolo de Transporte Colaborativo para Redes de Sensores sem Fio". In: *IX Workshop de Testes e Tolerância a Falhas do Simpósio Brasileiro de Redes de Computadores - SBRC´2008*, Rio de Janeiro, Brasil, Maio 2008.

[45] FOK, C.-L., ROMAN, G.-C., LU., C., "Agilla - A Mobile Agent Middleware for Wireless Sensor Networks", http://mobilab.wustl.edu/projects/agilla/, acesso em março de 2009.

[46] SOHRABI, K., POTTIE, G., "Performance of a novel self-organization protocol for wireless ad-hoc sensor networks", *Vehicular Technology Conference, 1999. VTC 1999 - Fall. IEEE VTS 50th*, v. 2, pp. 1222–1226 vol.2, 1999.

[47] TANENBAUM, A. S., *Redes de computadores*. Campus - Elsevier: Rio de Janeiro - Brasil, 2003.

[48] KUROSE, J. F., ROSS, K. W., *Redes de Computadores e a Internet : uma abordagem top-down*. 3rd ed. Pearson Addison Wesley: São Paulo, 2006.

[49] "IEEE Standard 802, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", Junho 2007, IEEE Computer Society.

[50] VASSIS, D., KORMENTZAS, G., ROUSKAS, A. N., et al., "The IEEE 802.11g standard fo high data rate WLANs", *IEEE Network*, v. 19, n. 3, pp. 21–26, 2005.

[51] BRENNER, P., "A Technical Tutorial on the IEEE 802.11 Standard", Julho 1996, Breezecom Wireless Communications.

[52] "IEEE Standard 802, part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications Wireless Personal Area Networks (WPANs)", Junho 2005, IEEE Computer Society.

[53] FILHO, L. J. H., LIMA, M. M. A. E., "Um estudo sobre redes Bluetooth e IEEE 802.11 e de sua interoperabilidade com o protocolo TCP", 2003, Laboratório de Redes de Computadores - Universidade Estadual do Oeste do Paraná.

[54] YE, W., HEIDEMANN, J., ESTRIN, D., "An Energy-efficient MAC Protocol for Wireless Sensor Networks". In: *21st Conference of the IEEE Computer and Communications Societies (INFOCOM)*, v. 3, pp. 1567–1576, Junho 2002.

[55] CORSON, M. S., PAPADEMETRIOU, S., PAPADOPOULOS, P., et al., "An Internet MANET Encapsulation Protocol (IMEP) Specification", *INTERNET-DRAFT*, Agosto 1999, IETF MANET Working Group.

[56] FULLMER, C. L., GARCIA-LUNA-ACEVES, J. J., "Floor acquisition multiple access (FAMA) for packet-radio networks", *SIGCOMM Comput. Commun. Rev.*, v. 25, n. 4, pp. 262–273, 1995.

[57] BARBOSA, V. C., *An introduction to distributed algorithms*. MIT Press: Cambridge, MA, USA, 1996.

[58] JABOUR, F. C., JABOUR, E. G., PEDROZA, A. C. P., "3M: Um protocolo de enlace multicanal para redes de sensores com alto grau de mobilidade", Maio 2009, Relatório técnico submetido em Maio de 2009 à Conferência Latino-Americana de Informática - CLEI 2009.

[59] VITERBI, A. J., *CDMA: Principles Of Spread Spectrum Communication*. 1st ed. Addison-Wesley Wireless Communications, 1995.

[60] "TelecomSpace - Telecom Tutorials and Forum - CDMA", http://www.telecomspace.com/cdma.html, acesso em Maio de 2009.

[61] PETAR POPOVSKI, H. Y., PRASAD, R., "Strategies for adaptive frequency hopping in the unlicensed bands", *IEEE Wireless Communication*, pp. 60–67, Dezembro 2006.

[62] "IEEE Standard for Floating-Point Arithmetic", *IEEE Std 754-2008*, pp. 1–58, 29 2008.

[63] "Java", http://java.sun.com, acesso em Abril de 2009.

[64] "Java Platform, Standard Edition 6 API Specification", http://java.sun.com/javase/6/docs/api/, Acesso em Maio de 2009.

[65] "IEEE Standard 802.3, Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications", Dezembro 2005, IEEE Computer Society.

[66] "Nordic Semiconductor ASA", http://www.nordicsemi.com/, acesso em Maio de 2009.

[67] ZHOU, G., STANKOVIC, J. A., SON, S. H., "Crowded Spectrum in Wireless Sensor Networks", *Proceedings of the 3rd Workshop on Embedded Networked Sensors (Em- Nets)*, Maio 2006.

[68] HILL, J., SZEWCZYK, R., WOO, A., et al., "System Architecture Directions for Networked Sensors". In: *Architectural Support for Programming Languages and Operating Systems*, pp. 93–104, 2000.

[69] "Crossbow Wireless Sensor Networks", http://http://www.xbow.com/, acesso em Maio de 2009.

[70] POLASTRE, J., SZEWCZYK, R., CULLER, D., "Telos: enabling ultra-low power wireless research". In: *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, IEEE Press: Piscataway, NJ, USA, 2005.

[71] "CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver", http://www.chipcon.com/ , acesso em Maio de 2009.

[72] LE, H. K., HENRIKSSON, D., ABDELZAHER, T., "A Practical Multi-channel Media Access Control Protocol for Wireless Sensor Networks". In: *IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks*, pp. 70–81, IEEE Computer Society: Washington, DC, USA, 2008.

[73] CONTI, M., GIORDANO, S., "Multihop Ad Hoc Networking: The Reality", *Communications Magazine, IEEE*, v. 45, n. 4, pp. 88–95, 2007.

[74] PETROVA, M., RIIHIJARVI, J., MAHONEN, P., et al., "Performance study of IEEE 802.15.4 using measurements and simulations". v. 1, pp. 487–492, April 2006.

[75] "ns-2 Network Simulator", http://www.isi.edu/nsnam/ns, 1998, acesso em Maio de 2009.

[76] "Random Waypoint Model", `http://www.netlab.tkk.fi/~esa/java/rwp/rwp-model.shtml`, acesso em Junho de 2009.

[77] TONG, L., ZHAO, Q., ADIREDDY, S., "Sensor Networks with mobile agents". In: *in Proc. 2003 Military Communications Intl Symp*, pp. 688–693, Outubro 2003, Boston, MA., http://acsp.ece.cornell.edu/.

[78] UTO, N., *Segurança de Sistemas de Agentes Móveis*, Master's Thesis, Instituto de Computação, Universidade Estadual de Campinas, 2003.

[79] QI, H., IYENGAR, S. S., CHAKRABARTY, K., "Multi-Resolution Data Integration Using Mobile Agents in Distributed Sensor Networks". In: *IEEE-Systems Man Cybernetics*, v. 31, n. 3, pp. 383–390, Agosto 2001.

[80] CHEN, M., KWON, T., YUAN, Y., et al., "Mobile agent-based directed diffusion in wireless sensor networks", *EURASIP J. Appl. Signal Process.*, v. 2007, n. 1, pp. 219–219, 2007.

[81] LANGE, D. B., OSHIMA, M., "Seven good reasons for mobile agents". In: *Communications of the ACM*, v. 42, n. 3, pp. 88–89, Março 1999.

[82] ZACHARY, J., "Protecting Mobile Code in the Wild", *IEEE Internet Computing*, v. 7, n. 2, pp. 78–82, Mar 2003.

[83] JANSEN, W., KARYGIANNIS, T., "NIST Special Publication 800-19-Mobile Agent Security", National Institute of Standards and Technology - USA.

[84] TERADA, R., *Segurança de Dados Criptografia em Redes de Computador*. 1st ed. Edgard Blücher LTDA, 2000.

[85] GREENBERG, M. S., BYINGTON, J. C., "Mobile Agents and Security", *IEEE Communications Magazine*, v. 7, n. 36, pp. 76–85, 1998.

[86] VIEIRA, A. C. C., *Projeto de Circuitos Criptográficos para Aplicação em Segurança*, Ph.D. Thesis, COPPE / UFRJ, 2005.

[87] "Current Public-Key Cryptographic Systems - A Certicom Whitepaper", www.certicom.com, Abril 1997.

[88] JÚNIOR, A., *Criptossistemas baseados em curvas elípticas: Estudo de Casos e Implementação em processador de sinais digitais*, Ph.D. Thesis, Departa-

mento de Engenharia de Computação e Automação Industrial, Universidade Estadual de Campinas, 2002.

[89] LI, Q., RUS, D., "Navigation protocols in sensor networks", *ACM Trans. Sen. Netw.*, v. 1, n. 1, pp. 3–35, 2005.

[90] JABOUR, F. C., JABOUR, E. G., PEDROZA, A. C. P., "Mobility support for wireless sensor networks". In: *Proceedings of IEEE International Conference on Computer and Electrical Engineering – ICCEE*, Phuket Island, Thailand, Dezembro 2008.

[91] JABOUR, F. C., JABOUR, E. G., PEDROZA, A. C. P., "Redes de sensores móveis: análise da velocidade, comunicação e esforço computacional". In: *7th International Information and Telecommunication Technologies Symposium - I2TS*, Foz do Iguaçu, Brasil, Dezembro 2008.

[92] ALVES, R. S. A., CAMPBELL, I. V., COUTO, R. S., et al., "Redes Veiculares: Princípios, Aplicações e Desafios", Minicursos do Simpósio Brasileiro de Redes de Computadores, SBRC'2009, Maio 2009.

[93] KOSCH, T., KULP, I., BECHLER, M., et al., "Communication Architecture for Cooperative Systems in Europe", *Communications Magazine, IEEE*, v. 47, n. 5, pp. 116–125, Maio 2009.

[94] LIU, B., BEHROOZ, DU, H., et al., "VGSim: An Integrated Networking and Microscopic Vehicular Mobility Simulation Platform", *Communications Magazine, IEEE*, v. 47, n. 5, pp. 134–141, Maio 2009.

[95] UZCÁTEGUI, R. A., ACOSTA-MARUM, G., "WAVE: A tutorial", *Communications Magazine, IEEE*, v. 47, n. 5, pp. 126–133, Maio 2009.

[96] JABOUR, F. C., JABOUR, E. G., PEDROZA, A. C. P., "Mobile wireless sensor networks: neighbors discovery, channel allocation and mobile agents applications", Relatório técnico submetido à IEEE Communications Magazine, Maio 2009.

[97] DARWIN, C., *Origem das espécies*. Ed. Itatiaia: Belo Horizonte, 2002, Trad. Eugênio Amado.

[98] DEITEL, H. M., DEITEL, P. J., *Java - Como programar*. PEARSON, 2005, 6. ed.

[99] HORSTMANN, C. S., CORNELL, G., *Core Java 2 - Fundamentos*. Alta Books, 2005, 7. ed.

[100] HORSTMANN, C. S., CORNELL, G., *Core Java - Advanced Features*. Prentice-Hall, 8. ed.

[101] LEVIS, P., "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications", Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), November 2003.

[102] "Simulating TinyOS Networks", Novembro 2003, http://www.cs.berkeley.edu/~ pal/research/tossim.html, acesso em Junho de 2007.

[103] "TinyOS", http://www.tinyos.net/, acesso em Junho de 2007.

[104] "J-Sim", http://www.j-sim.org/ . Futuro endereço: http://sites.google.com/site/jsimofficial/, acesso em Maio de 2009.

[105] "Maryland Routing Simulator", http://www.cs.umd.edu/projects/netcalliper/software.html, acesso em Maio de 2009.

[106] "NetSim", http://sourceforge.net/projects/netsim/, acesso em Maio de 2009.

[107] KESHAV, S., "The REAL Network Simulator", http://minnie.tuhs.org/REAL/, acesso em Maio de 2009.

[108] "Global Mobile Information Systems Simulation Library", http://pcl.cs.ucla.edu/projects/glomosim/, acesso em Maio de 2009.

[109] AT UCLA, P. C. L., "Parallel Simulation Environment for Complex Systems", http://pcl.cs.ucla.edu/projects/parsec/, acesso em Maio de 2009.

[110] "The TIOBE Programming Community index – TIOBE Software", http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, acesso em Maio de 2009.

[111] "Netbeans", http://www.netbeans.org/, Acesso em Maio de 2009.

[112] FALL, K., VARADHAN, K., "The ns Manual", Fevereiro 2007, A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC. Fall e Varadhan: editores.

[113] WILLIAMS, T., KELLEY, C., "gnuplot - An Interactive Plotting Program", http://www.gnuplot.info/documentation.html, Março 2007, Version 4.2 organized by Hans-Bernhard Bröker, Ethan A Merritt, and others. Manual prepared by Dick Crawford.

[114] TANENBAUM, A. S., *Sistemas Operacionais Modernos*. 2nd ed. Prentice Hall: São Paulo - Brasil, 2003.

[115] SILBERSCHATZ, A., GALVIN, P., GAGNE, G., *Sistemas Operacionais – Conceitos e aplicações*. trad. da 1. ed. ed. Elsevier: Rio de Janeiro, 2000.

[116] PRESSMAN, R., *Engenharia de Software*. McGraw-Hill Interamericana do Brasil, 2005.

[117] VIJAY, "software testing help", http://www.softwaretestinghelp.com/, Acesso em Maio de 2009.

[118] BLAAR, H., LEGELER, M., RAUBER, T., "Efficiency of Thread-parallel Java Programs from Scientific Computing", *Parallel and Distributed Processing Symposium, International*, v. 2, pp. 0115b, 2002.

[119] PHILIPPSEN, M., ZENGER, M., "JavaParty: Transparent Remote Objects in Java", *Concurrency: Practice and Experience*, v. 9, n. 11, pp. 1225–1242, November 1997.

[120] YELICK, K., SEMENZATO, L., PIKE, G., et al., "Titanium: A High-Performance Java Dialect". In: *In ACM*, pp. 10–11, 1998, Workshop on Java for High-Performance Network Computing, Stanford, 1998.

# Apêndice A

# O simulador desenvolvido

Este anexo descreve o simulador desenvolvido como ferramenta de avaliação e evolução dos protocolos propostos. São apresentados também resultados dos testes de sanidade e confiabilidade feitos no mesmo.

Durante a evolução desta tese, foi identificada a indisponibilidade de uma ferramenta que agregasse as funcionalidades ligadas à mobilidade e a plataformas de agentes móveis, ao mesmo tempo. Devido à complexidade dos simuladores já existentes, considerou-se mais otimizado desenvolver um específico, em lugar de agregar as funcionalidades desejadas a outros.

O simulador foi inteiramente desenvolvido na linguagem Java [63, 64, 98, 99, 100] e a totalidade do código fonte encontra-se disponível em `http://www.gta.ufrj.br/~jabour/`.

## A.1 Outros simuladores

O simulador *ns-2* é amplamente utilizado em simulações de redes de computadores, inclusive redes sem fio, redes *ad hoc* e redes de sensores. Possui a vantagem de ter muitas aplicações e protocolos desenvolvidos. Devido à sua estrutura complexa, é não-trivial e às vezes difícil incluir outros protocolos e algoritmos, ou acomodar outras arquiteturas no *ns-2*. Além disso, sua arquitetura não tão estruturada e a mistura de classes compiladas e interpretadas tornam difícil a compreensão e validação de códigos. Todas estas questões também têm impacto no aumento do tempo de desenvolvimento. Para o presente trabalho, não se mostra apropriado por não

incorporar uma plataforma de agentes móveis.

Atualmente, o TinyOS *Simulator* (TOSSIM) [101] [102] é o mais amplamente utilizado para simulações de redes de sensores. Trata-se de um simulador escalável, totalmente compatível com o sistema operacional para sensores TinyOS [103]. Permite o teste e implementação de algoritmos. O TOSSIM implementa a pilha de protocolos do TinyOS até o nível de bits, permitindo simulações de baixo nível, bem como de sistemas com aplicações de alto nível. Pode ser integrado a uma ferramenta de visualização, o TinyViz e já conta com um modelo de energia validado.

Fok *et al.* desenvolveram a plataforma (*middleware*) Agilla [2] [45] que provê um ambiente de programação de agentes móveis para redes de sensores sem fio. A plataforma é executada sobre o sistema operacional TinyOS e pode ser executada sobre o simulador TOSSIM. Agilla permite interações através de acessos a um espaço de tuplas onde dados podem ser consultados e inseridos, os agentes podem se clonar, executar migrações forte e fraca, disponibilizando uma série de instruções para a programação de agentes. Mesmo as interações locais podem se dar através de agentes estacionários na plataforma. A plataforma não prevê mobilidade dos nós, sendo as coordenadas de localização utilizadas como identificador do nó. Esta questão fundamental de projeto fez com que a mesma fosse abandonada para as simulações necessárias.

J-Sim[104] é um ambiente de simulação baseados em componentes, escrito integralmente em Java. Contém pacotes para diversos ambientes, incluindo um para RSSF. A última versão é relativamente recente, tendo sido disponibilizada há três anos. Possui suporte à mobilidade, mas também não incorpora a plataforma de agentes móveis.

*Maryland Routing Simulator* (MaRS)[105] é um pacote de simulação para redes baseado em um simulador anterior chamado NetSim [106]. Como o próprio nome diz, é principalmente usado para o estudo de diferentes algoritmos de roteamento. Já o NetSim, iria requerer, além da compreensão do funcionamento, a construção da extensão para a plataforma de agentes móveis.

*REalistic And Large (REAL) network simulator* [107] é voltado especificamente para o estudo de diferentes mecanismos de controle de fluxo e congestionamento em redes TCP/IP.

O simulador GloMoSim [108] possui funcionalidades relativas à mobilidade. A última versão é do ano de 2000. Não foram encontrados muitos exemplos de aplicações, sobretudo para redes de sensores. O simulador utiliza outro simulador paralelo de eventos discretos, o Parsec [109], e não possui suporte à agentes móveis. Estes fatores desencorajaram a sua adoção.

A necessidade de se ter suporte a simulações de agentes móveis e a possibilidade de simular diversos ambientes com mobilidade levaram ao desenvolvimento de um simulador específico.

Para definir a linguagem de programação, buscou-se uma linguagem orientada a objetos, que possibilite rapidez no desenvolvimento e que seja muito utilizada para o propósito em questão. Observou-se que a grande maioria dos simuladores desenvolvidos recentemente são escritos em Java. Quanto ao uso geral, esta linguagem tem sido a mais usada nos últimos dez anos, tendo hoje uma utilização em torno de 20% dentre todas as demais [110].

Outra questão considerada foi a facilidade de implementação da programação concorrente, através de múltiplas linhas de execução (*threads*).

A linguagem Java reune todas as características citadas anteriormente. Java apresenta outras características que, apesar de não fundamentais, auxiliaram muito durante o desenvolvimento. É independente de plataforma, podendo ser executada em diversas arquiteturas e diferentes sistemas operacionais, sem nenhuma modificação ou necessidade de recompilação do código. Possui uma documentação bastante completa na Internet, além de inúmeros fóruns de discussão, grupos de usuários e páginas com exemplos e soluções.

O Simulator usado neste trabalho foi integralmente desenvolvido em Java e a sua descrição está na próxima seção.

## A.2 Descrição do simulador

Foi desenvolvido um simulador de eventos discretos usado para validar e comparar os algoritmos propostos. Trata-se de uma aplicação Java que lê um arquivo de entrada no padrão do arquivo Tcl do simulador *ns-2* [75]. Este arquivo contém as instruções de movimentação dos nós da RSSF. Todos os nós são configurados com o

mesmo alcance de comunicação de rádio (em metros) e o mesmo erro percentual do algoritmo de localização. O simulador é configurado com uma precisão de relógio que vai de zero a doze casas decimais para cada passo da simulação. Cada nó, cada agente móvel e o relógio global da simulação são executados como linhas de execução independentes e concorrentes (*threads*). Somente quando todas as *threads* terminam a computação de um passo, o relógio avança para o próximo (segundo a precisão configurada para o relógio). Alguns agentes móveis são injetados na rede com alguns parâmetros: hospedeiro inicial, instante inicial de processamento, coordenadas da região de destino, coordenadas da região de retorno e tipo de algoritmo (Seções 4.5.2, 4.5.3 e 4.5.4) a ser usado.

A versão do Java utilizada na última compilação do simulador é a 1.6.0_04. O ambiente de desenvolvimento ou IDE (*Integrated Development Environment*) usado para desenvolvimento foi o *Netbeans* [111], versão 6.0.1 (Build 200801291616).

O protocolo 3M (Seção 3.5) é executado em todos os nós da rede.

Considera-se que um algoritmo de localização está em execução e que uma nova estimativa de localização é gerada para cada nó, em cada passo do relógio da simulação. Um parâmetro $\epsilon$ que representa o erro do algoritmo de localização é informado como dado de entrada. Deste modo, o simulador mantém sempre dois conjuntos de coordenadas para cada nó, a real (*realPosition*) e a medida pelo algoritmo de localização (*mesuredPosition*), que leva em conta o erro $\epsilon$. O conjunto de vizinhos reais $V_i$ é fornecido pelo simulador quando demandado. O conjunto de vizinhos conhecidos é mantido por cada nó, como resultado da execução do protocolo 3M. Quando o agente móvel decide migrar do seu hospedeiro para algum vizinho, isto é feito em um passo do relógio, com 100% de probabilidade de sucesso.

As classes, campos e métodos do simulador possuem seus nomes em inglês para facilitar a divulgação do simulador e intercâmbio de informações com outros desenvolvedores e programadores.

### A.2.1 O Núcleo do simulador

As simulações são naturalmente repletas de eventos em execução simultânea. A modelagem do simulador leva em conta esta característica, o que facilitou o projeto, documentação e compreensão da ferramenta como um todo. Deste modo, ao se

identificar uma nova entidade (ou classe) no sistema e ao se definir o escopo de suas ações no tempo, basta criar uma nova *thread* que represente esta entidade. Esta *thread* deve acomodar os atributos ("o que ela é") e métodos ("o que ela faz") dentro da classe e colocá-la em execução concorrente com as demais.

Como foi visto, a criação de uma nova classe de *threads* com uma função específica na simulação tem impacto mínimo do simulador como um todo. Na versão atual do simulador (5.3.0), são executadas três *threads* distintas: **NodeThread**, **MobileAgent** e **Clock**. Existe uma instância de **NodeThread** (uma *thread*) para cada nó da rede; uma instância de **MobileAgent** para cada agente móvel; e uma única instância da classe **Clock** para toda a simulação.

A seguir serão listadas as funções das classes **NodeThread** e **MobileAgent**.

**NodeThread**

- Ler o arquivo de entrada com a mobilidade, gerando uma estrutura de dados na memória principal com todos os deslocamentos e pausas previstos (uma única vez);
- Ajustar a mobilidade informada ao passo de relógio em uso (a cada passo);
- Mover o nó (a cada passo);
- Executar o protocolo 3M (a cada passo);
- Liberar o passo do relógio (após cada passo);
- Escrever arquivos de saída (*logs*) (a cada passo, com alguns resumos finais).

**MobileAgent**

- Executar um dos algoritmos de decisão do agente móvel (a cada passo);
- Liberar o passo do relógio (após cada passo);
- Escrever arquivos de saída (*logs*) (a cada passo, com alguns resumos finais).

Cada vez que um agente móvel atinge a região alvo, é contado um **sucesso**. Então, a região alvo muda da região destino para a região de retorno (ou vice versa).

O simulador conta ainda o número de migrações dos agentes e registra o instante de todos os eventos de interesse. Todos estes dados são escritos em arquivos individuais de cada nó e cada agente móvel, como será visto mais adiante.

Dentro de um ciclo de relógio, os eventos são executados segundo uma ordenação não determinística, pelas diversas *threads* em execução. Cada *thread* pode ou não ter alguma tarefa a executar dentro do ciclo atual. Caso tenha, ela executa esta tarefa e envia uma mensagem ao relógio global informando que terminou suas tarefas para este ciclo. Caso não tenha, libera o ciclo de relógio imediatamente, através do mesmo método.

A liberação é feita através do acesso a um método da classe **Clock**:

**clockInterface.nodeStepOk(nodeThreadID);**
**clockInterface.mobileAgentStepOk(mobileAgentID);**

Este métodos são acessados com exclusão mútua, através do modificador **synchronized** do Java. Isto garante que apenas uma *thread* esteja executando o método em um dado instante.

Um exemplo típico desta situação está nos agentes móveis. Cada um deles é executado por uma *thread*. Em geral, nas simulações, o instante inicial de execução do agente é um pouco posterior ao início da simulação, como se ele houvesse sido injetado na rede um pouco depois. Entretanto, todas as *threads* são criadas no início da execução do programa. Assim, até que chegue o momento inicial de execução do agente móvel, sua *thread* fica apenas liberando os ciclos de relógio do simulador. Durante este intervalo, outras *threads*, como as dos nós, por exemplo, já possuem instruções a serem executadas dentro dos ciclos de relógio.

O simulador permite a ocorrência de eventos com intervalos de tempo que são múltiplos do relógio do núcleo. Assim, nos passos intermediários, o detentor deste evento apenas libera o relógio da simulação. Assim, se a classe precisar executar a tarefa a cada $0,1\,s$ e o relógio estiver ajustado para avançar em períodos de $0,001\,s$, dentro da sua lógica interna, ela deverá liberar o relógio 99 vezes, executar a referida tarefa e liberar novamente o relógio. Em seguida o ciclo se repete.

O núcleo do simulador é executado pela classe **Simulator**. Esta classe cria uma instância da classe **Clock** que, como descrito anteriormente, controla o relógio global da simulação. O ponto fundamental deste controle está nas mensagens de liberação

do passo vindas das diversas *threads* em execução. A comunicação entre as diversas *threads* e a classe **Clock** se dá através da interface do relógio (**ClockInterface**) que é passada, pela classe **Simulator**, como parâmetro, a cada **thread** criada (Figura A.1). As *threads* podem enviar dados e consultar a *thread* **Clock**, através desta interface. Um exemplo é a obtenção do instante atual (**currentInstant = clockInterface.getCurrentInstant()**). Isto é feito para saber se o relógio já avançou para o próximo passo, ou seja, se todas as *threads* já liberaram o passo atual. Deste modo, a *thread* em questão decide se já pode iniciar a execução do próximo passo. A decisão é tomada segundo o Algoritmo 2.

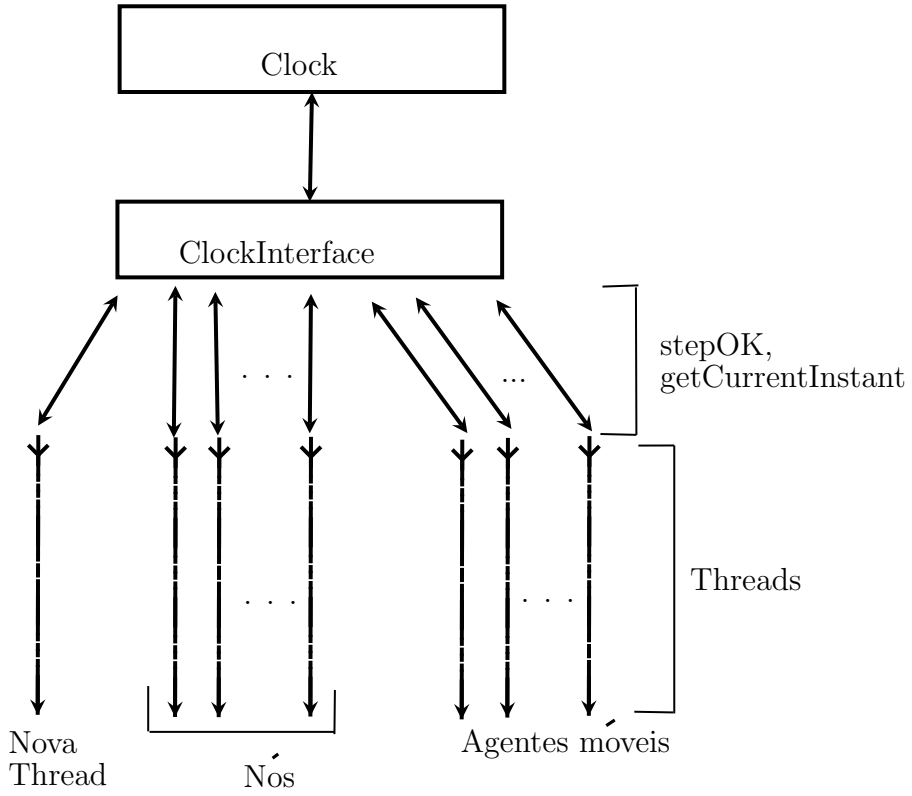



Figura A.1: Funcionamento do relógio do simulador

A classe **Simulator** também disponibilizada uma interface (**SimulatorInterface**), para prover acesso das diversas *threads* aos recursos do núcleo. Na Tabela A.1, ao final deste apêndice, estão as descrições dos recursos e das utilizações.

### A.2.2 A Interface de entrada de dados do simulador

O simulador recebe dados de entrada de quatro fontes distintas.

```
Algoritmo: Avanço do relógio local
currentInstant = clockInterface.getCurrentInstant() //passo corrente
//maxInstant é extraído do arquivo de movimentação (parâmetro de entrada)
lastInstant = 0 //passo anterior
enquanto currentInstant ≤ maxInstant faça
    currentInstant = clockInterface.getCurrentInstant()
    se currentInstant ≠ lastInstant então
        lastInstant = currentInstant
        //Executa processamento do passo atual
        clockInterface.nodeStepOk(Identificador da Thread)
    senão
        Libera o processador //Chance p/ as outras Threads cumprirem o passo
```

**Algoritmo 2**: Avanço do relógio local

### O arquivo de entrada com o cenário de mobilidade

A primeira fonte de dados é o arquivo de entrada de dados de mobilidade. Este arquivo deve seguir o formato do simulador *ns-2*. Foram tratados dois comandos básicos, o comando *set* e o *setdest* [112]. O primeiro especifica, para um dado nó da rede, o valor de uma das três coordenadas $X, Y$ ou $Z$. Um exemplo:

**$node_(0) set X_ 48.870191435562** : atribui à coordenada $X$, do nó identificado como **$node_(0)**, o valor 48,870191435562 (o *ns-2* trabalha com ponto decimal).

O segundo, especifica um deslocamento no plano:

**$ns at 2.064061274129 "$node_(1) setdest 9.422354510019 34.296218655448 23.871009329373"** : indica que, no instante 2,064061274129 $s$ da simulação, o nó identificado como **$node_(1)** deve se mover até as coordenadas (9,422354510019, 34,296218655448), a uma velocidade constante de 23,871009329373 $m/s$.

Qualquer linha no arquivo de entrada que fuja deste padrão é ignorada pelo simulador. Os métodos que tratam este arquivo de entrada geram uma estrutura de dados na memória principal com os dados correspondentes. Trata-se de um objeto da classe **Vector**. Cada nó lê o mesmo arquivo de entrada e carrega o "seu" **Vector** com as linhas que comecem **$ns_ at** e que, após o instante, contenham

**$node_(i)**, onde $i$ é o identificador do próprio nó. Em outras palavras, cada nó ignora as informações de todos os outros. Com isso, tem-se três vantagens: objetos na memória são acessados muito rapidamente, melhorando o desempenho; cada nó terá estruturas de dados menores para tratar; qualquer outro modelo de mobilidade, uma vez mapeado para um **Vector** nos padrões do simulador, pode ser usado. A Figura A.18 mostra a seqüência de execução desde o *setdest* até a movimentação durante a simulação.

O simulador *ns-2* é acompanhado de um programa chamado *setdest*, mesmo nome do comando descrito anteriormente. Ele gera cenários de mobilidade, implementando o modelo *Random Waypoint*. O *setdest* recebe parâmetros de configuração e gera arquivos de movimentação conforme o padrão do *ns-2* mostrado anteriormente. Para uma área, um número de nós e um tempo de duração pré-definidos, esta aplicação gera aleatoriamente uma coordenada inicial (passo 1), uma coordenada de destino (passo 2) e uma velocidade de deslocamento entre estas duas coordenadas (passo 3). Ao chegar à coordenada de destino, é gerado aleatoriamente um intervalo de repouso para o nó (passo 4). Em seguida, o processo se repete dos passos 2 a 4, até o término do período de duração definido.

Neste trabalho, os deslocamentos do tipo *Manhattan* foram gerados através de programas criados em Java e todos os demais deslocamentos randômicos foram gerados com o *setdest*.

O *setdest* gera também informações de distância entre pares de nós, o que torna sua execução muito mais lenta para cenários grandes e simulações de longa duração. Para reduzir o tempo de geração dos cenários, o código fonte foi modificado de modo a gerar apenas a posição inicial e os deslocamentos subseqüentes dos nós.

A Figura A.2 mostra um detalhe de um arquivo gerado pelo *setdest*. O cabeçalho do arquivo traz um resumo dos parâmetros usados para a geração do cenário de mobilidade pelo aplicativo *setdest*. Mais detalhes sobre estes parâmetros são apresentados nas descrições das simulações feitas ao longo deste trabalho. Entretanto, não é necessário o uso do *setdest*. Qualquer arquivo que especifique as coordenadas iniciais e os deslocamentos, dentro dos padrões da referida figura, são aceitos pelo simulador.

Ainda na Figura A.2, em seguida, são mostradas as linhas que especificam a

```
#
# nodes: 50, speed type: 2, min speed: 1.00, max speed: 30.00
# avg speed: 2.33, pause type: 2, pause: 10.00, max x: 56.00, max y: 56.00
#
$node_(0) set X_ 48.870191435562
$node_(0) set Y_ 43.247065511390
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 38.509682219944
$node_(1) set Y_ 41.751533123739
$node_(1) set Z_ 0.000000000000
. . .
$node_(49) set X_ 0.150664936597
$node_(49) set Y_ 52.459035979463
$node_(49) set Z_ 0.000000000000
. . .
$ns_ at 2.064061274129 "$node_(1) setdest 9.422354510019 34.296218655448 23.871009329373"
. . .
$ns_ at 3.321970371258 "$node_(1) setdest 9.422354510019 34.296218655448 0.000000000000"
. . .
$ns_ at 13.868663232932 "$node_(0) setdest 55.095188080286 10.547316320669 19.946323254340"
. . .
$ns_ at 15.537491956003 "$node_(0) setdest 55.095188080286 10.547316320669 0.000000000000"
. . .
$ns_ at 17.658554656627 "$node_(1) setdest 32.884772757076 49.586146986559 15.161678221922"
. . .
$ns_ at 87.147261394673 "$node_(0) setdest 28.549893547403 38.089674477152 0.000000000000"
```

Figura A.2: Detalhe de um arquivo de entrada do simulador

posição inicial dos nós. As reticências representam trechos retirados do arquivo. São geradas coordenadas iniciais para todos os nós, identificados de 0 a 49. Por fim são mostradas algumas linhas de movimentação e pausa dos nós.

Com valores truncados, podemos ver que o nó 1 foi posicionado nas coordenadas $(38, 41)$; aos 2 $s$ iniciou movimentação para $(9, 34)$ com velocidade de 23 $m/s$; aos 3,3$s$, já no destino previsto, permaneceu em repouso (velocidade zero) até a próxima instrução de movimentação, aos 17 $s$. Este ciclo de mobilidade e pausa se repete até o fim da simulação, para todos os nós. As movimentações se dão com velocidade constante e em linha reta.

Naturalmente, se tomarmos o instante em que o nó 1 se move pela primeira vez (2,064061274129 $s$), a distância entre sua posição inicial (38,509682219944 , 41,751533123739) e seu primeiro destino (9,422354510019 , 34,296218655448), que vale 30 $m$ e a velocidade especificada (23,871009329373 $m/s$), tem-se um tempo de deslocamento de 1,257909097129 $s$. Este valor, somado ao instante inicial do deslo-

camento, se iguala ao instante do próximo evento associado ao nó 1: 2,064061274129 + 1,257909097129 = 3,321970371258.

No simulador, foi necessário fazer uma conversão deste padrão de mobilidade para a granularidade de tempo em uso. Como já mencionado, o simulador é executado com uma resolução de tempo de 0 a 12 casas decimais. A cada passo do relógio, o nó deve se mover segundo uma fração do deslocamento total existente entre dois comandos do tipo *setdest* descritos anteriormente. A conversão foi feita da seguinte maneira:

A cada passo do relógio:

- É computada uma variável **time**, conforme a expressão:
  $time = currentInstant - lastInstant$;
- É executado o método **move(destino, velocidade, time)**. Os parâmetros **destino** e **velocidade** são retirados do objeto **Vector** que contém os movimentos do nó. São equivalentes a dois comandos *setdest* consecutivos para um mesmo nó, como na Figura A.2;
- Calcula-se a distância percorrida: $coveredDistance = speed \times time$;
- Calcula-se a distância total percorrida entre dois comandos *setdest*, com base na posição atual **realPosition** e no argumento **destino**:[1]
  $distanceFromRealPositionToDestiny =$
  $= \sqrt{(destino.X - realPosition.X)^2 + (destino.Y - realPosition.Y)^2}$ ;
- Por fim, por proporção (ou semelhança de triângulos), são calculadas as novas coordenadas do nó:

$$X = realPosition.X + \frac{\big(coveredDistance \times (destino.X - realPosition.X)\big)}{distanceFromRealPositionToDestiny}$$

$$Y = realPosition.Y + \frac{\big(coveredDistance \times (destino.Y - realPosition.Y)\big)}{distanceFromRealPositionToDestiny}$$

Assim, o nó se moveu apenas a fração do deslocamento total definido pelo arquivo de entrada, correspondente ao tempo de um passo do relógio, qualquer que ele seja.

A mobilidade se dá sobre as coordenadas reais do nó.

[1] Considere Ponto_Qualquer.X como a abscissa e Ponto_Qualquer.Y como a ordenada do ponto.

A Figura A.3[2] mostra o deslocamento do nó 1 entre as coordenadas iniciais da Figura A.2 e o primeiro destino do nó. O deslocamento deve ocorrer entre os instantes 2,06 $s$ e 3,32 $s$. O exemplo considera uma granularidade de tempo de $0,1\,s$, ou seja, um passo do simulador a cada $0,1\,s$.

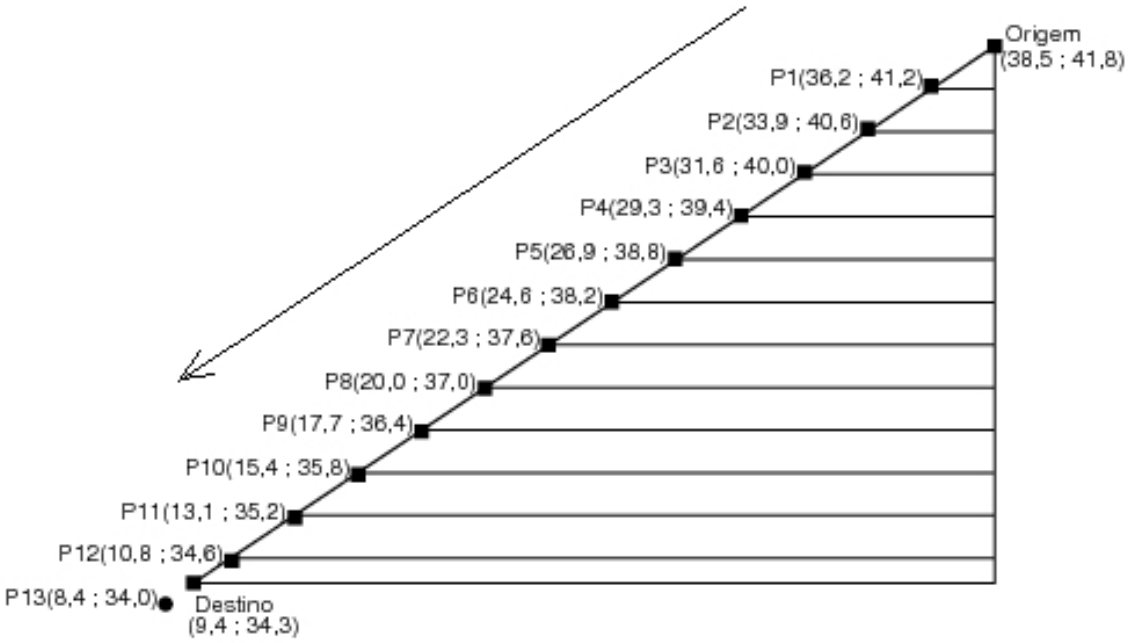



Figura A.3: Mapeamento do mobilidade do arquivo de entrada para o simulador

Na Figura A.3, tem-se os pontos inicial (origem) e final (destino) da trajetória prevista. Os pontos $P1, P2, P3, \cdots, P13$ mostram as posições que o nó assumirá durante cada passo do relógio.

Observa-se que, somente após atingir o ponto $P13$, o nó 1 passa a trabalhar com o próximo trecho de deslocamento. Existe uma resolução mínima que o método criado consegue atingir, em função do *clock* usado na simulação. Nas simulações com mobilidade aleatória utilizadas nesta tese, estas imprecisões nos pontos de inflexão das trajetórias são irrelevantes. Se for necessário reproduzir com maior acurácia o deslocamento existente no arquivo de entrada, os passos do relógio devem ser reduzidos (mais casas decimais).

[2]A escala do eixo Y está multiplicada pelo fator 3 para melhor visualização

### A configuração dos nós

A segunda interface de entrada é a janela de configuração dos nós da rede (Figura A.4). Nela são informados:

- ***Simulation ID*** – **Identificador da simulação:** O simulador permite que mais de uma simulação seja executada ao mesmo tempo. Para cada uma delas, todos os parâmetros precisam ser especificados;
- ***Number of nodes*** – **Número de nós:** Número total de nós da rede. Tem que ser o mesmo do arquivo de entrada;
- ***Clock Precision*** – **Passo de relógio da simulação:** de 0 a 12 casas decimais;
- ***Input data file*** – **Arquivo de entrada:** Permite localizar o arquivo de entrada descrito anteriormente;
- ***Radio range*** – **Raio de cobertura do rádio:** Valor em metros. O mesmo para todos os nós;
- ***Beacon interval, Backoff base, Beacon size, Data rate*** e ***Neighbor TTL*** – **Intervalo de *beacon*, base do *backoff*, taxa de transmissão para o canal de controle e TTL dos vizinhos conhecidos:** Todos os campos conforme descritos na Seção 3.5. Com relação às unidades, tem-se o *Beacon size* em *bytes* e todos os demais referenciados à precisão do relógio (*Clock Precision*).

Para os campos da Figura A.4 que são relativos ao passo do relógio, deve-se fazer as conversões apropriadas. Vejamos através de exemplos. Caso se esteja trabalhando com precisão de 4 casas decimais, tem-se o relógio avançando um passo a cada $0,0001\,s$ (um passo a cada $100\,\mu s$). Caso se deseje configurar o intervalo de anúncios de presença (*Beacon interval*) para meio segundo, o campo deve ser preenchido com o valor 5000, já que 5000 ciclos do relógio correspondem a $5000 \times 0,0001 = 0,5\,s$. O mesmo serve para *Backoff base* e *Neighbor TTL*.

O campo *Data rate* deve ser informado em *bytes / ciclos*. Assim, para $250\,kbps$ e 4 casas decimais de precisão do relógio, deve-se ter:

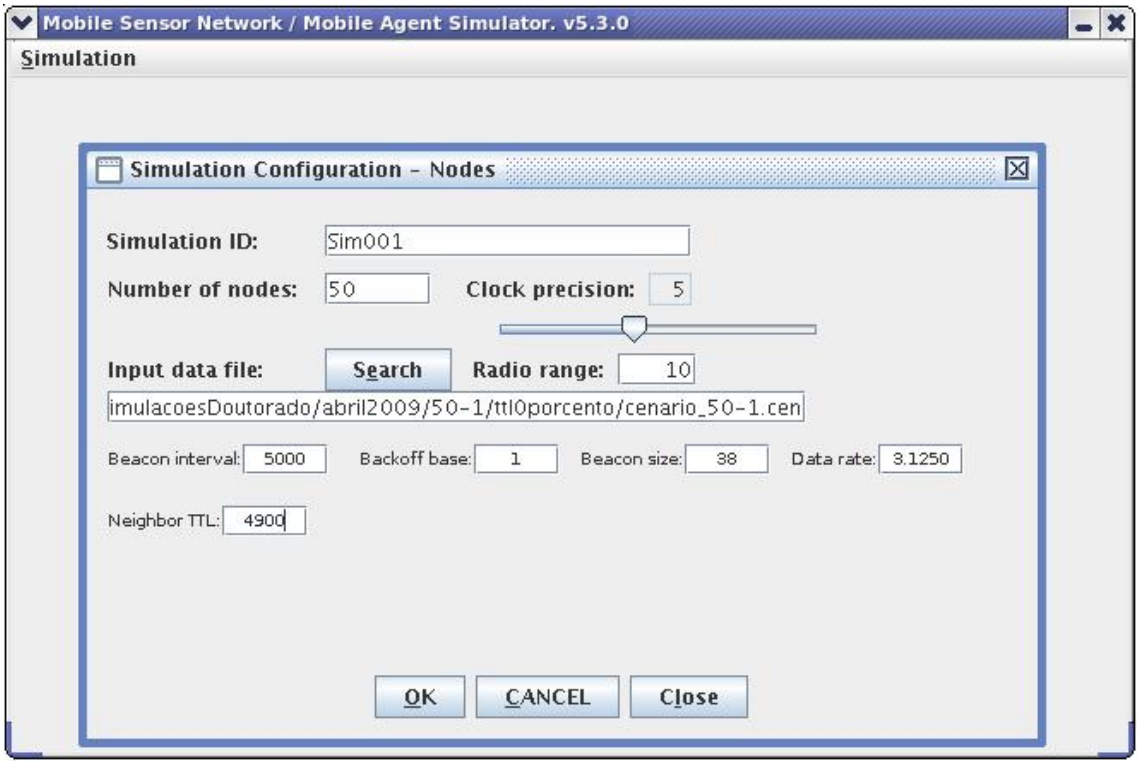



Figura A.4: Janela de configuração dos nós

$$250\ kbps = 250.000\ bps = \frac{250.000}{8}\ \frac{bytes}{s} = 31.250\ \frac{bytes}{s} =$$
$$= 31.250\ \frac{bytes}{s} \times 0,0001\ \frac{s}{ciclo} = 3,125\ \frac{bytes}{ciclo}$$

### Criação e configuração dos agentes móveis

A terceira interface de entrada é a janela de criação e configuração dos agentes móveis (Figura A.5). Nela são informados:

- ***Simulation*** – **Identificador da simulação:** Deve-se selecionar uma simulação já criada na janela de configuração dos nós;
- ***Mobile agent ID*** – **Identificador do agente móvel:** Criado automaticamente;
- ***To be created at*** – **Instante de criação:** O agente móvel começa a executar a partir deste instante;
- ***Initial node ID*** – **Hospedeiro inicial:** O agente começa a executar neste nó;

- ***Coordinates*** **– Coordenadas:** Coordenadas da região alvo e da região de retorno. Vértices superior esquerdo e inferior direito. Não há limite para o número de agentes a serem criados;

- **Tipo de algoritmo:** São os quatro botões na base da janela. Especificam o tipo de algoritmo a ser executado pelo agente móvel. É individual para cada agente criado. Não precisa ser o mesmo para todos. O terceiro foi abandonado e os demais atuam conforme descrito na Seção 4.5. Na versão 6.2.0 do simulador, atualmente em teste, existem mais 3 subtipos que previnem as oscilações nas migrações dos agentes.

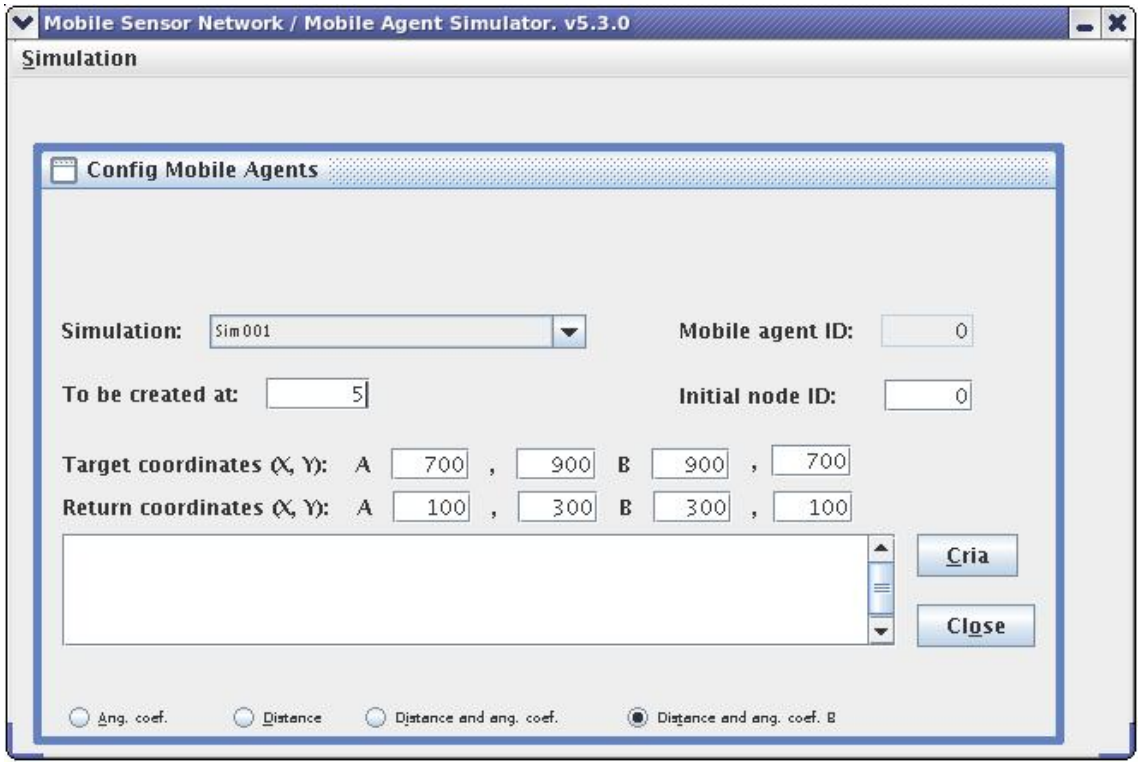



Figura A.5: Janela de configuração dos agentes móveis

### Configuração dos *logs* de saída

A quarta interface de entrada é a janela de configuração dos *logs* de saída (Figura A.6). A seguir são relacionadas as opções de configuração. Detalhes dos arquivos de saída e saídas para tela estão na Seção A.2.3.

- ***Nodes*****:** Habilita ou desabilita a geração do *log* de mobilidade individual de cada nó, arquivo **Node_Nº.log**;

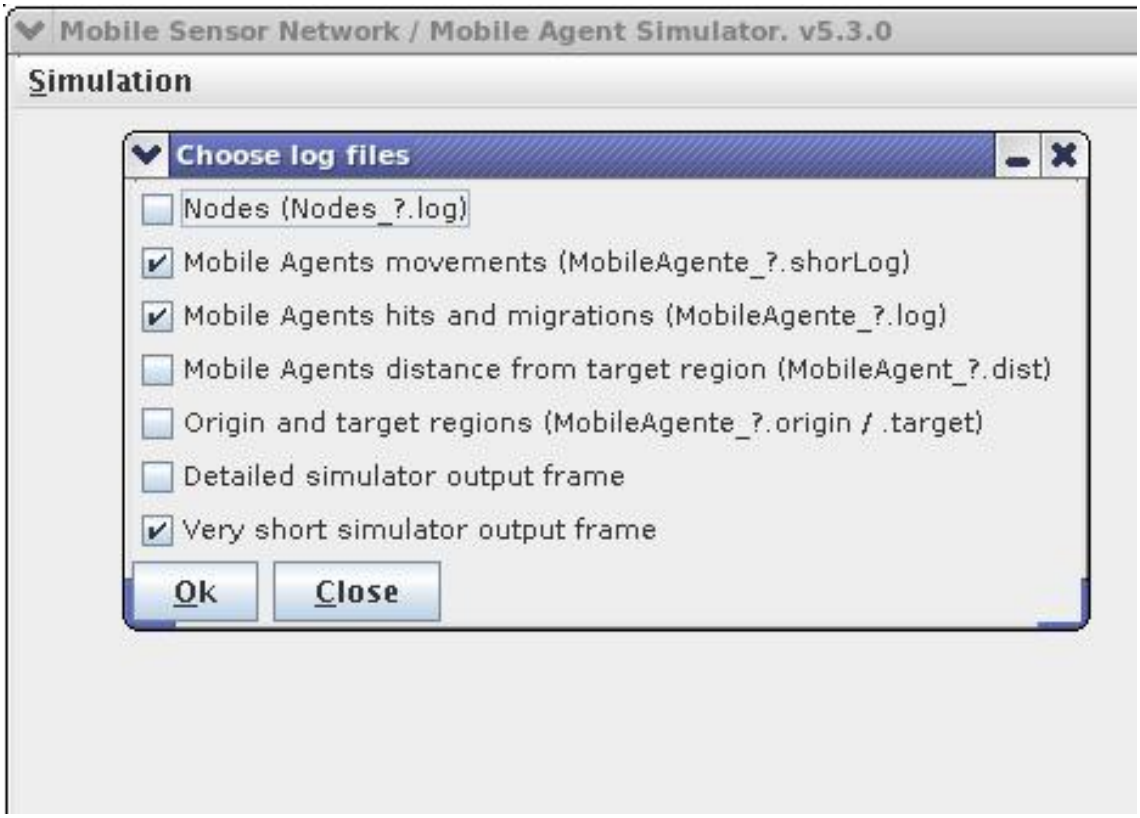



Figura A.6: Janela de configuração dos *logs* de saída

- ***Mobile Agents movements*:** Habilita ou desabilita a geração do arquivo de *log* da mobilidade individual de cada agente móvel, arquivo **MobileAgent_Nº.shortLog**. Corresponde à posição do nó hospedeiro do agente em cada passo da simulação;

- ***Mobile Agents Hits and Migrations*:** Habilita ou desabilita a geração do arquivo de *log* com o histórico individual de cada agente móvel, contendo as migrações efetuadas e os sucessos (atingimento da região alvo), arquivo **MobileAgent_Nº.log**;

- ***Mobile Agents distance from target region*:** Habilita ou desabilita a geração do arquivo de *log* individual com a distância do agente móvel, ou seja, de seu hospedeiro, até o centro da região alvo, arquivo **MobileAgent_Nº.dist**;

- ***Origin and target regions*:** Gera um arquivo com as coordenadas da região alvo e da região de retorno, para cada agente móvel, arquivos **MobileAgent_Nº.origin / .target**. As coordenadas são suficientes para o aplicativo *Gnuplot* [113] desenhar a área quadrada referente às duas regiões;

- **Detailed simulator output frame:** Configura o simulador para emitir o máximo de mensagens na janela de saída durante a simulação. É útil na fase de desenvolvimento e manutenção do simulador ou em simulações novas, durante a calibragem de parâmetros. Como gera muitas chamadas de sistema [114] e muita concorrência das *threads* pela "posse" do direito de escrita na janela [115], tende a reduzir o desempenho do simulador;

- ***Very short simulator output frame*:** Reduz ao máximo a impressão de mensagens na janela de saída em tela. Caso esta opção e a anterior estejam desmarcadas, existe um volume intermediário de mensagens na tela.

Uma vez configurados os dados dos nós, dos agentes móveis e do perfil dos *logs* de saída, com a opção ***Simulation*** – ***New***, o simulação é iniciada.

Ao final, a janela de saída em tela pode ser salva como arquivo texto e é gerado também um arquivo com o nome da simulação e a terminação ".data".

A seguir é apresentada a interface de saída do simulador.

### A.2.3 A Interface de saída de dados do simulador

Durante a execução da simulação, os arquivos de *log* especificados da janela da Figura A.6 vão sendo gerados em paralelo. Também as saídas para vídeo vão sendo emitidas durante a execução da simulação.

As análises e estudos das simulações devem ser feitas através de um pós-processamento dos arquivos de saída (arquivos de *log*) e da tela de saída.

A seguir serão apresentadas e descritas estas saídas.

#### Arquivos Node_Nº.log

Estes arquivos registram todo o deslocamento de cada nó ao longo de toda a simulação, conforme a granularidade de tempo especificada. É útil para analisar o perfil da mobilidade na rede e validar as decisões dos agentes móveis. O arquivo contém também as coordenadas estimadas pelo algoritmo de localização. Nesta tese, o algoritmo de localização não foi implementado e seus resultados são considerados disponíveis. Entretanto, o simulador permite inserir o erro deste algoritmo. Por

fim, tem-se o coeficiente angular da reta do movimento e o tipo de movimento (ver Seção 4.5.3).

A Figura A.7 mostra um detalhe de um arquivo de *log* do nó. A primeira coluna contém o instante. Observa-se que a simulação foi executada com 4 casas decimais de período de *clock*. Em seguida, estão as coordenadas $X$ e $Y$ reais e, logo em seguida, as estimadas pelo algoritmo de localização. Os valores são iguais já que a simulação rodou com erro de localização zero. O penúltimo valor corresponde ao coeficiente angular do movimento e o último é o tipo de movimento. Quando o nó está parado (**s**) (primeira linha), o coeficiente angular é configurado para um valor muito elevado. Isto faz com que este nó nunca seja escolhido para migração em função da direção do movimento, devido ao fato de estar parado. O segundo bloco mostra um deslocamento para a direita (**r**) e o último um deslocamento para a esquerda (**l**).

```
. . .
10.6414 14.686613 26.982832 14.686613 26.982832 6000000.0 s
. . .
10.6461 14.7403555 27.00587 14.7403555 27.00587 0.4286906 r
10.6462 14.741499 27.006361 14.741499 27.006361 0.4286906 r
10.6463 14.742642 27.006851 14.742642 27.006851 0.4286906 r
. . .
14.0305 36.896843 36.501465 36.896843 36.501465 0.25523013 l
14.0306 36.89502 36.501 36.89502 36.501 0.25523013 l
14.0307 36.893196 36.500534 36.893196 36.500534 0.25523013 l
14.0308 36.891373 36.50007 36.891373 36.50007 0.25523013 l
. . .
```

Figura A.7: Detalhe de um arquivo de saída do tipo Node_Nº.log

### Arquivo MobileAgent_Nº.shortLog

Contém os instantes e as coordenadas do agente móvel, ou seja, do seu nó hospedeiro naquele instante (Figura A.8). Os registros começam no instante de início da execução do agente, no caso, 5 $s$. No exemplo da Figura, o hospedeiro estava em repouso no início da execução.

### Arquivos MobileAgent_Nº.log

Este arquivo registra a "vida" do agente móvel. Começa com o instante em que a *thread* foi criada, que é sempre zero. Em seguida, grava o período de execução do

```
5.0000 33.652046 29.156422
5.0001 33.652046 29.156422
5.0002 33.652046 29.156422
. . .
25.8635 1.3542259 12.763707
25.8636 1.3529353 12.764414
25.8637 1.3516448 12.765121
. . .
```

Figura A.8: Detalhe de um arquivo de saída do tipo MobileAgent_Nº.shortLog

agente. Todas as linhas seguintes (ver Figura A.9) indicam apenas a espera pelo tempo de início da execução, enquanto a *thread* está apenas liberando o relógio para os demais processamentos. O início da execução é informado, juntamente com as coordenadas da região alvo e o hospedeiro inicial. Após esta linha, são escritas no arquivo todas as migrações do agente. Nestas linhas constam o nó destino, o número acumulado de migrações, o coeficiente angular e tipo de movimento ótimos (Seção 4.5) e os dados de direção de todos os vizinhos do hospedeiro do agente, no formato "nó / coeficiente angular / tipo de movimento". Este tipo de registro se refere aos agentes que decidem com base apenas no coeficiente angular. A referida figura contém linhas de detalhe para os 3 tipos de agentes, ou seja, para os 3 tipos de algoritmos de decisão. Na prática, cada arquivo é formatado conforme o tipo de agente que o gerou.

Pode-se notar no instante 8.8659 da Figura A.9, que a região alvo está à direita ($Optimum : 0.5121642/r$), o hospedeiro está indo para a esquerda ($0/0.114889115/l$) e o nó para onde o agente migrou segue para a direita ($2/-12.967741/r$). Nota-se também que o desvio angular do movimento do nó 2 com relação à trajetória ótima é muito grande. O nó 2 se move para baixo, em um ângulo de aproximadamente 85°[3], enquanto a trajetória ótima tem 27° para cima[4]. Entretanto, 6 décimos de segundo adiante, na migração 2, instante 9.4899, o algoritmo efetua uma correção, reduzindo o desvio para 48°[5].

Pesquisas com ajustes mais finos nos tipos de movimento, considerando ângulos de visada menores na decisão dos agentes, estão relacionadas como trabalhos futuros.

Cada vez que o agente atinge a região alvo, o evento é registrado, como na linha

[3] $\tan -12,967741 = -85,59°$

[4] $\tan 0,5121642 = 27,12°$

[5] $\tan 2{,}136364 - \tan 0{,}29978576 = 64{,}92° - 16{,}69° = 48{,}23°$

```
0.0000 thread started
0.0000 will run from 5 to 49.591238770895
0.0001 is waiting its initial instant
0.0002 is waiting its initial instant
. . .
5.0000 is waiting its initial instant
5.0000 started. Searching target (37.5,37.5). Current host is node 0
8.8659 decided to migrate to node 2 (migration number 1). Optimum:0.5121642/r ;
    0/0.114889115/l ; 2/-12.967741/r ; 4/6000000.0/s
9.4899 decided to migrate to node 3 (migration number 2). Optimum:0.29978576/r ;
    0/6000000.0/s; 2/-12.967741/r; 3/2.136364/r; 4/6000000.0/s; 5/6000000.0/s; 9/6000000.0/s
. . .
32.6593 decided to migrate to node 32 (migration number 6681). ; 21/20.128973 ; 23/19.678461 ;
    28/24.756798 ; 32/11.608799 ; 33/21.388311 ; 34/19.078926 ; 41/26.471378 ; 43/17.453058
. . .
50.5006 decided to migrate to node 27 (migration number 19122). Optimum:-18.17685/r ;
    8/11.706913/7.2241373/l ; 10/25.74771/6000000.0/s ; 18/5.771166/6000000.0/s ;
    27/3.9311726/6000000.0/s ;29/17.623825/6000000.0/s ; 36/24.832983/0.6324973/l ;
    46/7.848933/2.9343562/l ; B
. . .
99.8993 HIT 4150 34360 migrations
. . .
99.8994 decided to migrate to node 23 (migration number 34362). Optimum:0.77035373/r ;
    1/5.0332127/6000000.0/s ; 2/11.671522/-0.5501729/l ; 4/4.385944/1.2166842/l ;
    6/7.7691007/0.9938272/l ; 7/1.7391492/6000000.0/s ; 12/6.9621253/6000000.0/s ;
    13/13.427994/-0.06673525/l ; 16/24.59749/0.6646217/l ; 19/12.797454/6000000.0/s ;
    21/7.352835/2.6226416/r ; 22/6.04108/0.4649831/l ; 23/3.9025831/-1.261828/r ;
    24/6.1438823/-1.1626794/l ; 25/22.121964/1.2160804/r ; 26/20.436808/0.6443089/r ;
    27/9.939766/0.11163895/l ; 28/12.306578/3.9192543/l ; 36/20.677156/6000000.0/s ;
    41/20.3532/6000000.0/s ; 42/9.447186/0.8675393/l ; 46/12.837417/-5.89933/l ;
    47/12.344824/0.1378026/l
. . .
99.9500 MA 0 4150 hits and 34362 migrations
```

Figura A.9: Detalhe de um arquivo de saída do tipo MobileAgent_Nº.log

referente ao instante 99.8993.

No bloco referente ao instante 99.8994 da Figura A.9, tem-se o formato da linha de migração quando o algoritmo de decisão híbrido é utilizado (Seção 4.5.4). Por isso, a linha contém, para cada nó vizinho, seu identificador, a distância ao centro da região alvo e o coeficiente angular da direção de deslocamento, acompanhado do tipo de movimento. No caso, apenas os vizinhos com tipo de movimento **r** foram considerados. Dentre eles, o escolhido para migração foi o nó 23, cuja distância à região alvo é a menor. Quando não há nenhum vizinho com o mesmo tipo de movimento e a decisão é tomada com base em todo o conjunto de vizinhos, o caracter **B** é inserido ao final da linha, como mostrado no instante 50.5006.

Para agentes móveis executando o algoritmo **somente-distância** (Seção 4.5.2),

como no instante 32.6593, apenas as distâncias são exibidas.

A última linha do arquivo contém o instante final, o identificador do agente móvel, o total de sucessos (*hits*) e de migrações do agente na simulação.

### Arquivos MobileAgent_Nº.dist

Neste arquivo estão os instantes da simulação e a distância do agente à região alvo. O objetivo é que esta distância diminua até ocorrer um *hit*. Neste instante, saltará para um valor máximo, já que a região alvo muda de destino para retorno, ou vice-versa (Figura A.10).

```
. . .
735 711.78345
736 707.515
737 703.3407
738 699.2621
739 695.2809
740 691.39886
741 691.39886
742 687.6176
743 683.9389
744 680.3642
745 676.8953
746 673.5338
747 670.28143
748 667.1396
749 664.11005
750 661.19415
751 658.3935
752 655.70953
753 650.69763
. . .
```

Figura A.10: Detalhe de um arquivo de saída do tipo MobileAgent_Nº.dist

Na figura A.11 vemos um gráfico extraído de um arquivo de distância. Observa-se a eficiência do agente em reduzir a distância à região alvo. Não são valores extraídos de muitas simulações. Trata-se de uma única instância de um agente. A densidade de nós é de apenas $0,001\ \frac{nó}{m^2}$ (800 nós em uma área de $1000m$ X $1000m$) e o raio de cobertura do rádio de $10\,m$. O relógio avança de 1 em $1\,s$. O passo de $1\,s$, em conjunto com a baixa densidade e a velocidade baixa com relação à área (1 a $30\,m/s$, pausas de 1 a $20\,s$, velocidade média de $9,37\,m/s$) justificam a alta latência. A latência pode ser observada entre dois mínimos próximos da ordenada 100, no gráfico. A distância não chega a zero devido ao fato das regiões destino e de

retorno serem quadrados de $200m$ X $200m$. Assim que o agente chega a esta região, ele retorna à outra, ao passo que a distância é medida até o centro das respectivas regiões. Ao atingir a região alvo, a distância sobe para aproximadamente 800 $m$, que é a distância dos limites de uma região ao centro da outra.

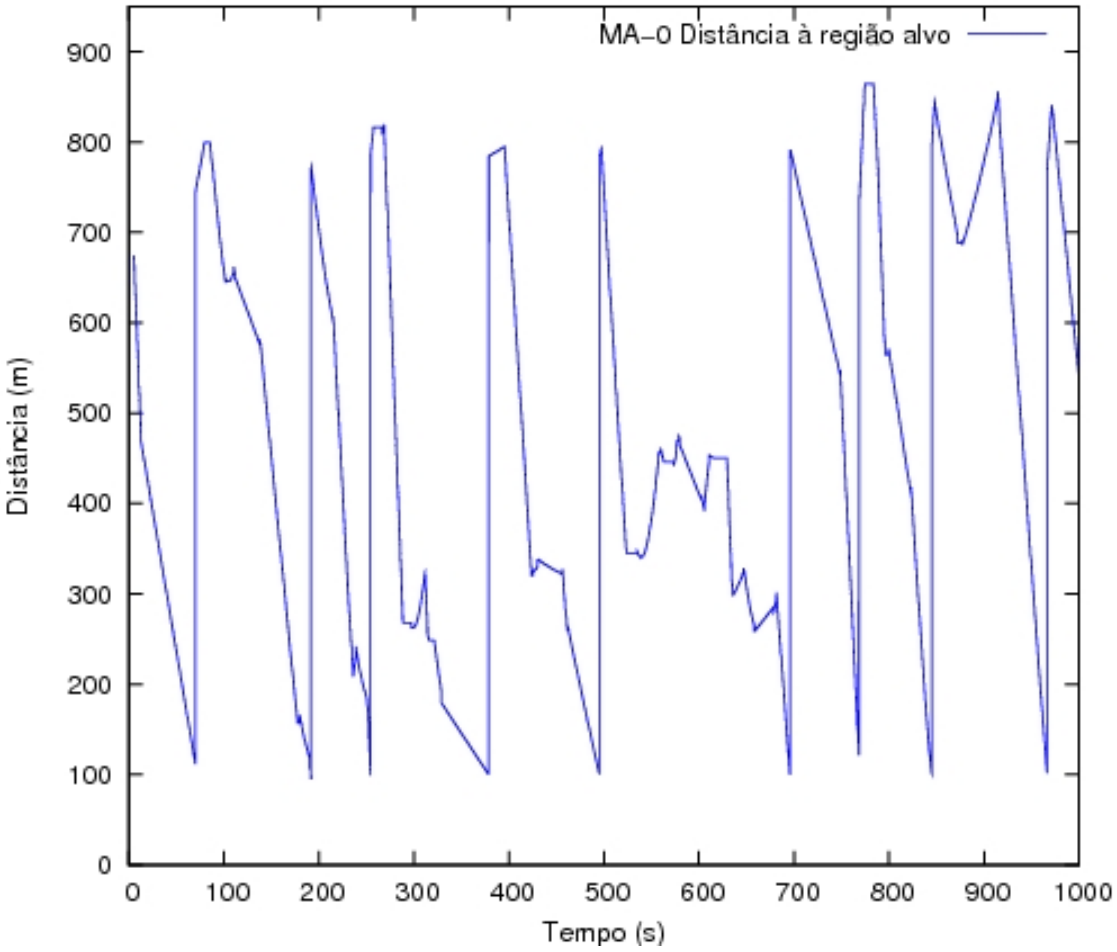



Figura A.11: Distância do agente móvel à região alvo

### Arquivos MobileAgent_Nº.origin / .target

Apenas para gerar graficamente as regiões alvo e de retorno (Figura A.12).

```
900.0 1000.0
1000.0 1000.0
1000.0 900.0
900.0 900.0
900.0 1000.0
```

Figura A.12: Detalhe de um arquivo de saída do tipo MobileAgent_Nº.origin / .target

### Arquivo com os dados gerais da simulação

É criado um arquivo com todos os dados de configuração da simulação (Figura A.13)

```
Thu May 14 20:08:40 BRT 2009
Simulator version.: 5.3.0
Simulation.....: escritaDaTese
Number of nodes...: 10
Input data file...: /home/jabour/simulacoesDoutorado/2009-05/10-1/cenario_10-1.cen
Radio range.....: 10
Clock precision...: 4
Beacon Interval...: 5000
Backoff Base.....: 1
Beacon Size.....: 38
Data rate.....: 3.125
Neighbor TTL.....: 4900
Logs.....: true, true, true, true, true, true, false
Mobile agents data:
   escritaDaTese;0;5;0;35.0;40.0;40.0;35.0;0.0;5.0;5.0;0.0;0.
   escritaDaTese;1;5;1;35.0;40.0;40.0;35.0;0.0;5.0;5.0;0.0;1.
   escritaDaTese;2;5;2;35.0;40.0;40.0;35.0;0.0;5.0;5.0;0.0;3.
   escritaDaTese;3;5;3;35.0;40.0;40.0;35.0;0.0;5.0;5.0;0.0;0.
   escritaDaTese;4;5;4;35.0;40.0;40.0;35.0;0.0;5.0;5.0;0.0;1.
   escritaDaTese;5;5;5;35.0;40.0;40.0;35.0;0.0;5.0;5.0;0.0;3.
   escritaDaTese;6;5;6;35.0;40.0;40.0;35.0;0.0;5.0;5.0;0.0;0.
   escritaDaTese;7;5;7;35.0;40.0;40.0;35.0;0.0;5.0;5.0;0.0;1.
   escritaDaTese;8;5;8;35.0;40.0;40.0;35.0;0.0;5.0;5.0;0.0;3.
```

Figura A.13: Detalhe do arquivo com os dados gerais da simulação

### Arquivo com o resumo de cada algoritmo

Para cada algoritmo de decisão, o último agente móvel a terminar sua execução é responsável por imprimir o resumo do algoritmo correspondente, conforme mostrado na Figura A.14

| Algor. | Hits | | Migrations | | # MA |
|---|---|---|---|---|---|
| type | Total | Average | Total | Average | |
| 3 | 56952 | 189,84 | 295374 | 984,58 | 300 |

Figura A.14: Resumo da simulação para um tipo de algoritmo de decisão

### Saídas para vídeo

O simulador inicia, para cada simulação em execução, uma janela de saída. São emitidas mensagens periódicas durante a simulação e ao final é possível salvar tudo

para um arquivo texto. Conforme visto na Seção A.2.2, esta saída pode ser resumida (apenas avanço do relógio, de um em um segundo) ou analítica (Figura A.15).

```
Thread Clock initiated with 10 nodes; 10 mobile agents; clock running with 0 decimal
    places (step = 1)
Simulation is at instant 0
0 Thread_0 started
0 Thread_0 colecting movement trace from cenario_10-1.cen
0 Thread_0 opened file cenario_10-1.cen
. . .
0 Thread_0 closed file cenario_10-1.cen
0 Thread_0 terminate colecting movement trace from cenario_10-1.cen
0 Thread_0 instantiating its Node Object
0 Node_0 will run (move) until 9994.642316488360 seconds
. . .
0 MobileAgent_0 thread started
0 MobileAgent_0 log file opened, thread processing
0 MobileAgent_0 will run from 5 to 9994.642316488360
. . .
Simulation has advanced to instant 6
6 MobileAgent_1 started. Searching target 950.0 , 950.0
. . .
12 Node_1 will now move to 75.0, 100.0 at 5.0 m/s
12 Node_3 will now move to 100.0, 75.0 at 5.0 m/s
12 Node_5 will now move to 100.0, 25.0 at 5.0 m/s
. . .
Simulation has advanced to instant 482
482 MobileAgent_7 HIT 1 2 migrations
. . .
Simulation has advanced to instant 9995
9995 Node_0 stopped
. . .
9995 MobileAgent_0 log file closed
9995 MobileAgent_0 compact log file closed
9995 MobileAgent_0 stopped
. . .
```

Figura A.15: Saída analítica do simulador, em vídeo

### A.2.4 Os níveis de enlace e físico do simulador

O nível de enlace do simulador implementa o protocolo 3M, conforme descrito na Seção 3.5. Os parâmetros de entrada do simulador utilizados pelo protocolo estão descritos na Seção A.2.2.

A *thread* de cada nó envia mensagens de anúncio de presença (Algoritmo 1, Seção 3.5.3).

Foi criada a classe **Channel** que modela um canal de comunicação. Cada nó pode instanciar quantos canais forem necessários. Estes canais são instâncias da

classe **Channel** e representam os canais em uso pelo nó. Todos os nós criam pelo menos o canal de controle.

Os canais controlam o estado do meio através da variável lógica *busy*. Sempre que um nó envia dados ao meio, os canais de mesma freqüência de todos os nós na sua área de cobertura vão para o estado **busy = true**. O canal volta a estar disponível (*idle* ou **busy = false**) quando o nó recebe o fim do quadro.

Um nó "escutar" o meio corresponde a ler o estado do campo *busy* da sua instância de canal para tal freqüência.

O nó detecta o fim do quadro através da informação de tamanho do quadro (campo **FL** − ***Frame Length***, Seção 3.5.2) contida no cabeçalho do pacote físico. Ao receber o fim do quadro, o nível físico envia os dados à camada 2 através do método **sendToMACLayer(String phyBuffer)**.

Durante a recepção do quadro, a interface física do nó pode ser sensibilizada com outros fragmentos de quadros. Deste modo, o método citado invoca outro método (**validate(phyBuffer)**), que verifica se houve ou não colisão. Caso não tenha havido, o quadro é processado. Caso contrário ele é descartado. Além disso, são feitos registros de anúncios bem sucedidos e colisões, para análises posteriores.

Como apresentado na Seção 3.5.5, no item que descreve o ajuste do relógio para as simulações e na nota de rodapé correspondente, o simulador calcula quantos ciclos de relógio são necessários para enviar o quadro. Durante o número de ciclos calculado, o processo de envio descrito anteriormente é executado.

Em função da diferença de ordem de grandeza entre mobilidade (segundos) e camada de enlace (microssegundos), não se considera variações de topologia durante a transmissão de um quadro (*beacon*), ou seja, não são tratadas variações no conjunto de vizinhos reais. Para quadros maiores e mesmo para uma maior consistência deste módulo do simulador, estas rotinas devem ser revistas. Nas simulações efetuadas não foram observados impactos negativos no comportamento do simulador.

## A.3 Validação e testes de sanidade

Em ciência da computação, os testes de sanidade representam uma breve análise da funcionalidade de um *software*, sistema, ou cálculo, com a finalidade de garantir que

o sistema ou metodologia funcione como esperado. É freqüentemente usado antes de uma rodada de testes mais exaustiva [116].

Vijay [117] iniciou, em 2004, um *blog* específico sobre testes de *softwares* e em 2006 migrou o conteúdo para um domínio próprio. Na referida página da Internet constata-se a grande importância dos testes em programas. Trata-se de uma fonte de informações sobre técnicas e ferramentas de testes, muitas delas aplicadas neste trabalho, como testes de sanidade, de fumaça e testes manuais.

É fundamental que o simulador tenha o comportamento esperado, de forma determinística, em cada uma de suas funções e rotinas. Por este motivo, o simulador foi exaustivamente testado e validado a cada passo da sua evolução. Apesar da complexidade que o simulador atingiu, a validação de cada uma das funções mais importantes é viável e eficaz.

Um exemplo deste fato está na precisão das operações matemáticas. Por coerência com o simulador *ns-2*, que foi usado como padrão do arquivo de mobilidade, o simulador manipulou as coordenadas também com 12 casas decimais. Surgiram, então, erros mínimos nas últimas casas decimais. Isto fez com que fosse adotada a classe **BigDecimal**. Trata-se de uma classe apropriada à manipulação de números reais grandes, além de manipular com grande precisão as casas depois da vírgula. Deste modo, o simulador passou a trabalhar, até onde foi verificado, com erro de precisão nulo até a décima segunda casa, para todas as operações usadas.

### A.3.1 Validação do relógio do simulador

Um comportamento que mereceu especial atenção foi a consistência do relógio da simulação. Um erro possível seria o relógio avançar quando ainda houvesse alguma *thread* em execução no passo atual. Isto levaria a uma falsa simultaneidade e invalidaria o simulador.

A primeira ação foi uma análise do código fonte. O mesmo é simples, na medida em que cada *thread* implementada tem que enviar uma mensagem de "passo ok" (*stepOk*) para a *thread Clock*. Esta, por sua vez, mantém estruturas de dados de controle do passo atual de sua exclusiva manipulação (*private*) e só libera o próximo passo quando todas as *threads* "filhas" o tiverem liberado.

Em seguida, foram feitas observações nas saídas para vídeo, com a impressão do

relógio e de delimitadores (linhas tracejadas) a cada passo avançado. Os eventos relacionados aos nós e agentes móveis também são rotulados com o passo atual na *thread*. A validação se deu uma vez que nenhum registro com marcas de tempo divergentes (atrasadas ou adiantadas) foi observado dentro dos delimitadores.

Por fim, foi criada uma *thread* especial que não libera o relógio em um certo instante. Observou-se que todas as demais pararam no mesmo passo. Em outras palavras, toda a simulação parou, esperando a *thread* especial.

### A.3.2 Validação dos algoritmos de decisão dos agentes móveis

Para verificar se os agentes estavam decidindo com base nos dados corretos e se estavam tomando as decisões conforme especificavam os algoritmos, foram feitos testes manuais.

Para cada tipo de agente, ou seja, para cada tipo de algoritmo de decisão, tomou-se uma amostragem de instantes de migração. Em seguida, levantou-se, junto aos *logs* dos nós, os parâmetros de interesse. Estes são a localização, a direção e o tipo de movimento. Com base nestes dados e no alcance do rádio, verificou-se se o conjunto de vizinhos estava correto. Para cada nó do conjunto, verificou-se se os dados estavam de acordo com a mobilidade prevista pelos arquivos de mobilidade, antes de passarem pelo filtro de entrada (arquivos originais gerados pelo *setdest*). Com base nesta massa de dados, resolveu-se o algoritmo manualmente. Os resultados foram confrontados com as decisões tomadas pelos agentes no simulador. Os resultados não apresentaram erros.

Muitas verificações foram feitas, como aquelas apresentadas na Seção A.2.3 que descreve a interface de saída. Ou seja, tomando uma linha do arquivo com uma decisão do agente, observou-se no conjunto de vizinhos, se os dados estavam corretos (tangentes e tipo de movimento). Em seguida, se o alvo de migração adotado correspondia à escolha correta.

Foram avaliados instantes sem migrações para verificar se a não migração foi a opção correta no momento.

Neste e em muitos testes, mensagens impressas na tela eram criadas temporariamente para acompanhar a computação e verificar se a evolução dos dados cal-

culados apresentavam resultados consistentes com o esperado. Da mesma maneira, muitas variáveis e o estado de muitas *threads* foram monitorados ao longo de muitas instâncias de execução, através de ferramentas apropriadas do Netbeans.

Outro teste realizado foi chamado de Fuga Radial. Neste teste, foi criado um cenário cujo resultado é de fácil dedução. Assim, basta ver se o comportamento do simulador é o esperado.

São criados 16 nós. Todos são posicionados em um mesmo ponto $(50, 50)$. Ao iniciarem seus deslocamentos, todos se afastam do ponto inicial, com velocidades iguais, conforme mostrado na Figura A.16.

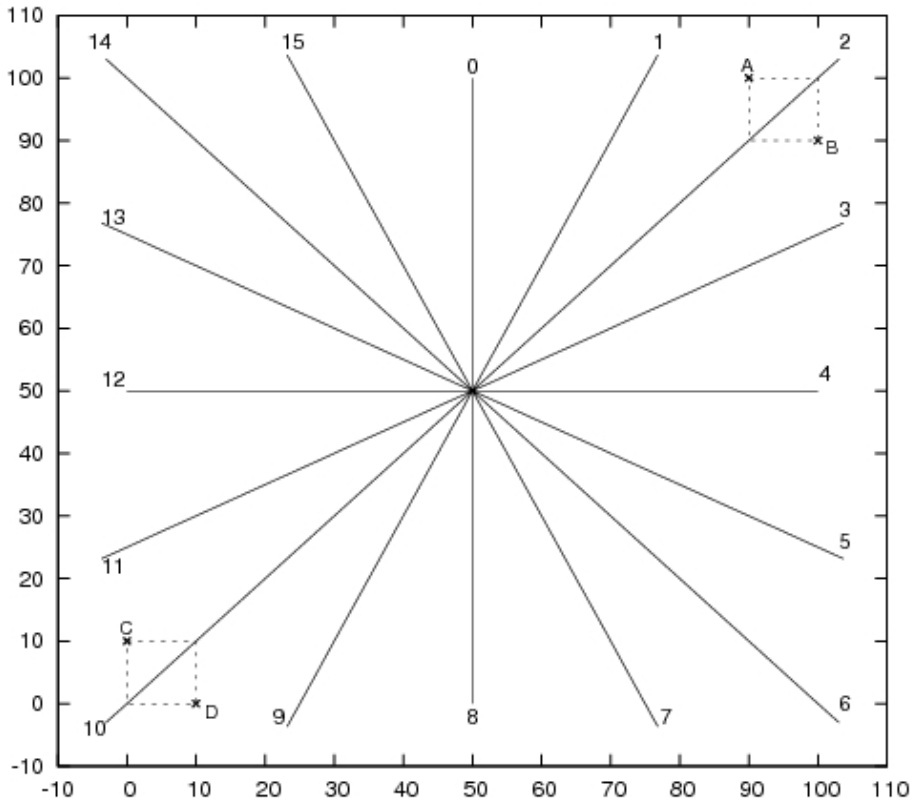



Figura A.16: Fuga radial de 16 nós

É injetado um agente em cada um dos nós. Todos têm como destino a região delimitada pelos pontos $A(90, 100)$ e $B(100, 90)$. A região de retorno é delimitada pelos pontos $C(0, 10)$ e $D(10, 0)$.

O identificador de cada nó está no fim da reta de deslocamento de cada um deles. O nó que passa pela região destino é o nó 2.

A faixa de cobertura do rádio, de $30m$, é suficiente para que todos os nós estejam no raio de cobertura uns dos outros durante um período de tempo suficiente para

ocorrerem migrações (à exceção do nó 12). Deste modo, pelo algoritmo **somente-coeficiente-angular**, espera-se que todos os nós migrem para o nó 2.

Cada agente móvel é iniciado em um nó com o mesmo identificador, ou seja, o agente 0 inicia no nó 0, o agente 1 no nó 1 e assim por diante. O passo do relógio é de um segundo e os agentes começam a executar aos 3 $s$. Todos os nós se movem a uma velocidade de 5 $m/s$.

A seguir as decisões dos agentes e os comentários.

O agente 0 migrou para o nó 3 e em seguida para o nó 2 (notação a ser usada: $MA0 : 0 \rightarrow 3 \rightarrow 2$). Como o nó 0 começa com um trajetória para cima, após 4 $s$ já se deslocou 20 $m$ e a trajetória ótima, entre seu nó hospedeiro 0 e o centro da região alvo (ponto médio entre $A$ e $B$) está mais próxima da trajetória do nó 3 do que do 2. No passo seguinte a migração já é para o nó 2 e não há mais nenhuma migração. Assim, a trajetória e as migrações do agente móvel 0 estão de acordo com o esperado e podem ser verificadas na Figura A.17

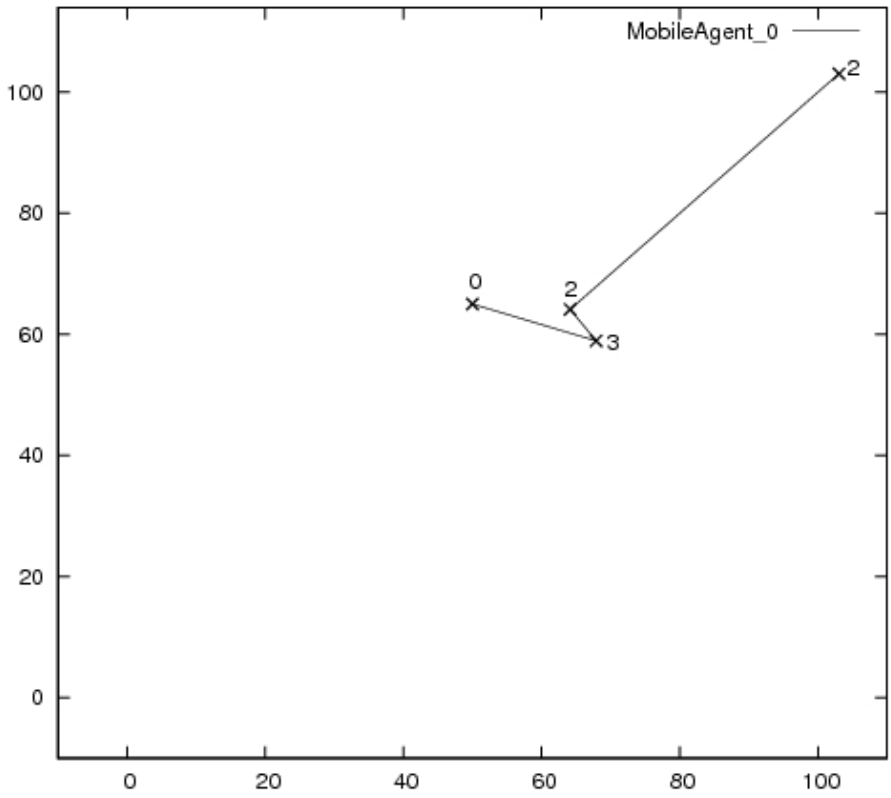



Figura A.17: Trajetória do agente móvel 0

$MA1 : 1 \rightarrow 2$: validado.

$MA2$ : não migrou: validado.

*MA3* : 3 → 2: validado.

*MA4* : 4 → 1 → 2: validado. Como o nó 4 já havia se deslocado para a direita, a trajetória do nó 1 é melhor. Logo em seguida ocorre a correção, indo para o nó 2.

*MA5* : 5 → 1 → 2: validado.

*MA6* : 6 → 2: validado.

*MA7* : 7 → 3 → 1 → 3 → 1 → 3 → 1 → 2: validado.

*MA8* : 8 → 4 → 1 → 3 → 1 → 3 → 1 → 2: validado.

*MA9* : 9 → 5 → 2: validado. A trajetória do nó 5 não é muito boa, mas é melhor do que a do nó 9. Os nós de 0 a 4 já estavam fora de alcance na primeira decisão de migração. Do 5, houve a convergência para o nó 2.

*MA10* : 10 → 6 → 3 → 1 → 3 → 1 → 3 → 2: validado.

*MA11* : 11 → 7 → 4 → 2: validado.

*MA12* :: Não migrou e não atingiu a região alvo: validado.

*MA13* : 13 → 3 → 2: validado.

*MA14* : 14 → 3 → 2: validado.

*MA15* : 15 → 3 → 2: validado.

Ao final da primeira parte da simulação de validação, todos os agentes que atingiram a região alvo permaneceram no nó 2, como era esperado. Todos estes agentes tiveram suas regiões alvo mudadas para a região delimitada pelos pontos $C$ e $D$, como prevê o algoritmo de execução do NIG.

A última etapa desta simulação fez com que um novo nó passasse pela diagonal situada entre as localizações dos nós 2 e 10. Como esperado, todos os agentes hospedados no nó 2 migraram para este novo nó, atingindo a região de retorno no canto inferior esquerdo da Figura A.16.

### A.3.3 Validação do mapeamento *setdest* X simulador

Como mostrado na Figura A.18, são várias etapas até que a movimentação do nó ocorra durante as simulações. O *setdest* gera o arquivo de mobilidade. Cada nó processa e filtra este arquivo, gerando seu próprio padrão de mobilidade na memória principal. Os métodos de mobilidade de cada nó atualizam a posição do nó, gerando a mobilidade, conforme descrito na Seção A.2.2. Por fim, cada nó gera um arquivo de saída com a sua movimentação durante a simulação.

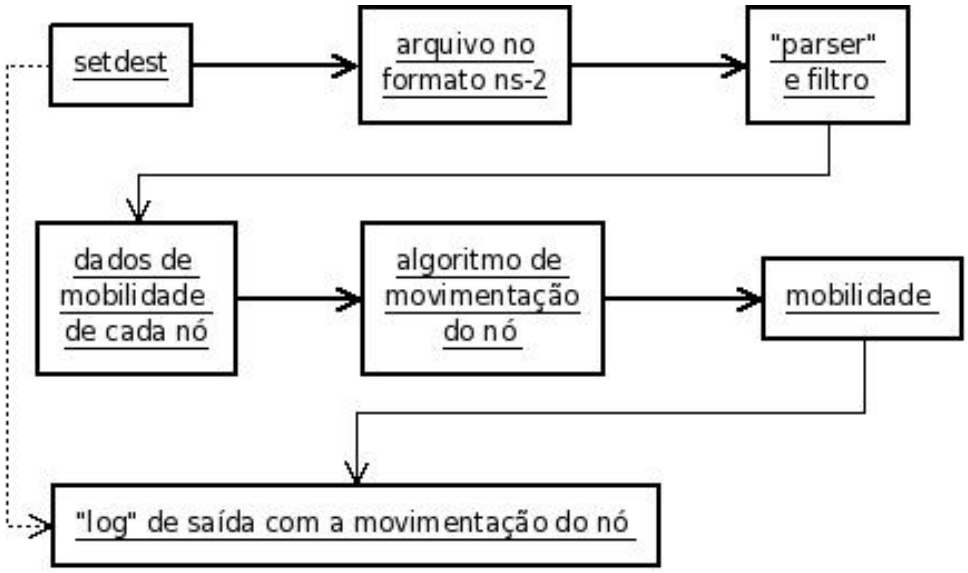



Figura A.18: Etapas de mapeamento da mobilidade para o padrão do simulador

O primeiro passo da validação foi a análise gráfica e analítica dos métodos e dos resultados, conforme Seção A.2.2 e Figura A.3.

Outro teste feito foi comparar graficamente a entrada e a saída, conforme linha pontilhada da Figura A.18. Deste modo, foi gerado o gráfico de mobilidade para alguns nós a partir do arquivo de entrada e a partir do *log* de saída. Deste modo foi possível verificar que nenhum erro foi inserido durante as etapas intermediárias. Os resultados de um dos testes estão na Figura A.19.

## A.4 Desempenho

Não foi feita uma análise acurada do desempenho do simulador quanto ao consumo de recursos do sistema. Apenas algumas considerações superficiais serão apresentadas.

Foram executadas simulações com até 1000 nós e 1000 agentes móveis, o que atingiu um total de mais 2000 *threads* em execução concorrente. Este cenário exigente foi executado em um sistema com um processador Intel Pentium 4, com 384 MB de memória RAM e sistema operacional Linux. Nestes cenário maiores, o desempenho do sistema para outros processos em execução ficou comprometido, mas não inviável.

Cada *thread* dos nós e dos agentes móveis pode manipular mais de um arquivo.

Nas simulações descritas anteriormente, quando o número de arquivos abertos superou a casa dos 4000, ocorreram alguns problemas de gerenciamento. O sistema operacional não disponibilizou ponteiros (*handlers*) para todos eles. O simulador é executado dentro de uma única instância da máquina virtual Java (JVM - Java Virtual Machine). Para o sistema operacional, trata-se de um único processo a manipular este número de arquivos. As *threads* utilizadas são implementadas no espaço de usuário e não no nível de *kernel*. Pesquisas preliminares mostraram que existem configurações de ambiente que podem solucionar o problema. Entretanto, estas não foram aplicadas com êxito. O que se fez foi utilizar a configuração seletiva de *logs* de saída e reduzir o número de agentes móveis por rodada para não atingir os limites críticos.

Uma das evoluções pretendidas para o simulador é a implementação de *threads* no nível do núcleo do sistema operacional, para poder desfrutar do paralelismo real (mais de um núcleo de processamento) ou para utilização em ambientes de computação paralela como *clusters* [118, 119, 120]. As *threads* Java são implementadas no espaço de usuário e cabe à JVM o seu gerenciamento.

A despeito das questões apresentadas no parágrafo anterior, um nível de paralelismo foi identificado na prática. Em uma plataforma executando o sistema operacional *Microsoft Windows XP Professional – Versão 2002 – Service Pack 2*, com processador *Intel Core2 Quad CPU – Q9400 – 2.66GHz*, observa-se que o simulador ocupa paralelamente os 4 núcleos do processador. Este fato é altamente vantajoso para o desempenho do simulador, reduzindo o tempo de execução devido aos ciclos de CPU a aproximadamente um quarto. O simulador TOSSIM, no mesmo sistema, ocupa apenas um dos núcleos.

Uma simulação com 700 nós e 10 agentes apresenta um uso que varia de 50 a $82MB$ de memória principal.

Em muitas situações, várias instâncias da JVM (até oito), com uma simulação em cada uma, foram colocadas em execução concorrente no sistema descrito anteriormente, sem falhas. Algumas execuções foram feitas em plataformas semelhantes, mas com o sistema operacional Windows, da Microsoft. Também não foram observadas falhas e não foi necessária nenhuma modificação ou recompilação, o que é característica da linguagem Java.

De um modo geral, o simulador apresentou bom desempenho. Pelas experiências com o uso de outros simuladores como o TOSSIM e *ns-2*, observou-se um desempenho um pouco inferior. De um modo geral, a linguagem C obtém maior desempenho do que Java.

## A.5 Considerações finais

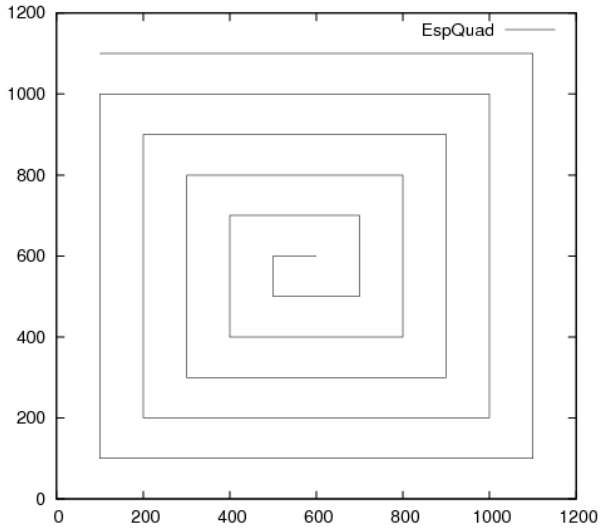
Este anexo apresentou o simulador desenvolvido para testes, evolução e validação das propostas desta tese.

O funcionamento da ferramenta, suas interfaces e núcleo foram descritos. Foi demonstrado o funcionamento seguro e correto do simulador, através de testes de sanidade.

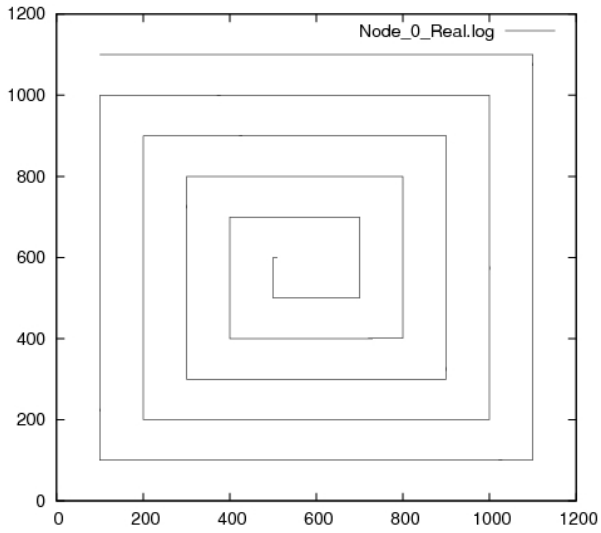
Pela sua arquitetura, observou-se que o simulador pode ser estendido para outras aplicações e pode receber a implementação de outras camadas de protocolo.

| **Método:** | **getMaxInstant(int nodeID)** |
|---|---|
| **Descrição:** | Acesso ao instante final da simulação |
| **Método:** | **getNodeMesuredPosition(int nodeID)** |
| **Descrição:** | Acesso à posição medida do nó pelo algoritmo de localização |
| **Uso:** | Utilizada por **MobileAgent**. Na realidade, o agente móvel tem esta informação no próprio nó onde está. No simulador, a comunicação se dá através do núcleo. |
| **Método:** | **getNeighboursSet(int nodeID)** |
| **Descrição:** | Retorna todos os vizinhos reais de um nó. |
| **Uso:** | Usado como escopo de decisão dos agentes móveis, antes da conclusão da implementação do protocolo 3M. |
| **Método:** | **getNodeRealPosition(int nodeID)** |
| **Descrição:** | Retorna a posição real do nó. |
| **Uso:** | Usada pela *thread* **MobileAgent** apenas para enviar aos arquivos de *logs* de saída. Nenhuma decisão é tomada com base nestas coordenadas já que elas não são conhecidas. Apenas a posição medida (estimada pelo algoritmo de localização) é conhecida. |
| **Método:** | **broadcastBeacon(int nodeID, int remainingSteps)** |
| **Descrição:** | Enviar a fração de um *beacon*. O simulador, com base no tamanho do quadro, na taxa de transmissão e na precisão do relógio, calcula quantos ciclos são necessários para enviar cada quadro. |
| **Uso:** | Usado pelos nós para enviar *beacons*, na execução do protocolo 3M. |
| **Método:** | **getNumberOfNodes()** |
| **Descrição:** | Retorna o número de nós na rede. |
| **Uso:** | Uso interno do simulador. Cada nó cria uma estrutura de dados homogênea (vetor ou *array*) para controlar o TTL dos vizinhos conhecidos. A rigor, o nó não tem este dado e na prática deve ser usado um valor arbitrário, suficientemente grande ou alocação dinâmica de memória. |

Tabela A.1: Acesso ao núcleo do simulador




(a) Mobilidade oriunda do arquivo de entrada




(b) Mobilidade oriunda do *log* de saída

Figura A.19: Comparação do arquivo de entrada (*setdest*) com o *log* de saída