**Universidade de Brasília**
**Instituto de Ciências Exatas**
**Departamento de Ciência da Computação**

# Avaliação do uso de Diferentes Protocolos para Localização com Abordagem de Sistema Multiagente e Rede Neural

Humphrey Corrêa da Fonseca

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientadora
Prof.ª Dr.ª Célia Ghedini Ralha

Brasília
2011

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenador: Prof. Dr. Maurício Ayala Rincón

Banca examinadora composta por:

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora) — CIC/UnB
Prof. Dr. André Costa Drummond — CIC/UnB
Prof.ª Dr.ª Flávia Maria Santoro — Unirio

**CIP — Catalogação Internacional na Publicação**

Fonseca, Humphrey Corrêa da.

Avaliação do uso de Diferentes Protocolos para Localização com Abordagem de Sistema Multiagente e Rede Neural / Humphrey Corrêa da Fonseca. Brasília : UnB, 2011.
116 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2011.

1. RNA, 2. SMA, 3. QoS, 4. wlans

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

**Universidade de Brasília**
**Instituto de Ciências Exatas**
**Departamento de Ciência da Computação**

# Avaliação do uso de Diferentes Protocolos para Localização com Abordagem de Sistema Multiagente e Rede Neural

Humphrey Corrêa da Fonseca

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof.ª Dr.ª Célia Ghedini Ralha (Orientadora)
CIC/UnB

Prof. Dr. André Costa Drummond
CIC/UnB

Prof.ª Dr.ª Flávia Maria Santoro
Unirio

Prof. Dr. Maurício Ayala Rincón
Coordenador do Mestrado em Informática

Brasília, 15 de dezembro de 2011

# Dedicatória

Em memória de Sônia, Francisco e Alta. Vocês sempre estiveram presentes e sempre serão lembrados.

# Agradecimentos

Muitas pessoas me apoiaram nessa etapa da minha, que não poderia ter sido concluída sem esse apoio. Entre elas destacam-se meus pais, Fernando José Corrêa da Fonseca e Sirlene Gonçalves dos Santos, pela presença constante e ajuda incondicional em todas as situações; e minha noiva, Fernanda Pereira Ibaldo, pelo apoio de sempre, pelos comentários que se mostraram decisivos para o desenvolvimento do trabalho e pelas artes criadas para esse trabalho.

Meus agradecimentos a minha avó Maria por ter cedido a casa nos primeiros experimentos e pela constante preocupação com a minha alimentação. Agradeço também ao meu irmão Andrew pela ajuda nos experimentos.

Meus agradecimentos aos meus amigos Bruno, Leonardo, Pedro, Vinícius e Wesdarley que me deram apoio, entenderam as minhas ausências e pelos momentos de diversão.

Não poderia deixar de agradecer a doutoranda Ana Régia de M. Neves, pela proposta de pesquisa e pelo apoio nos momentos mais importantes desde que este trabalho foi iniciado.

Agradeço também à minha orientadora, Prof.ª Dr.ª Célia Ghedini Ralha (CIC/UnB), pelas valiosas orientações, pelo exemplo de competência, profissionalismo e por todo o suporte durante o mestrado.

# Abstract

Context-oriented systems have received greather interest in the computing community, since they need a better user integration with the environment. However, the first task when defining a context sensitive application is the user location, which must be done dynamically and intelligently. The location problem appears more complex when it checks for various electronic devices, transmitting signals simultaneously on the same radio frequency, which can be used to determine position of an electronic device, but in deferent patterns, such as Bluetooth (IEEE 802.15.1), ZigBee (IEEE 802.15.4) and Wi-Fi (IEEE 802.11). Given the above, this work addresses the problem of locating users indoors, through the use of diferent protocols with neural network approach with multi-agent systems and applied aspects of quality of service.

**Keywords:** RNA, SMA, QoS, wlans

# Resumo

Sistemas orientados a contexto têm apresentado um crescente interesse na comunidade de computação, uma vez que necessitam uma melhor integração do usuário com o ambiente onde ele se insere. No entanto, a primeira tarefa durante a definição de uma aplicação sensível ao contexto é a localização do usuário, a qual deve ser feita de forma dinâmica e inteligente. O problema de localização se mostra mais complexo quando se verifica a existência de diversos dispositivos móveis, transmitindo simultaneamente sinais na mesma radiofreqüência, que podem ser usados para determinar posição do usuário através de dispositivos eletrônicos, com padrões diferentes, como o Bluetooth (IEEE 802.15.1), Zigbee (IEEE 802.15.4) e Wi-Fi (IEEE 802.11). Diante do exposto, esse trabalho aborda o problema de localização de usuários em ambientes fechados, através do uso de diferentes protocolos com abordagem de rede neural e sistemas multiagente tendo aplicado aspectos de qualidade de serviço.

**Palavras-chave:** RNA, SMA, QoS, wlans

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas e Siglas

**ACC** *Agent Communication Channel*. 31, 37

**AMS** *Agent Management System*. 31, 32, 34

**ANOVA** *Analysis of Variance Kernel*. 50

**AoA** *Angle of Arrival*. xii, 45, 46

**API** Application Programming Interface. 54, 66

**APS** *Application Support Sublayer*. 14, 15

**BD-ADDR** *Bluetooth Device Address*. 11

**BSSID** *Basic Service Set Identifier*. 67

**CAC** *Channel Access Code*. 11

**CDMA** *Code Division Multiple Access*. 20

**CIS** *Context Information Service*. 55

**CPU** *Central Processing Unit*. 36, 55

**CR** *Conflict Resolution*. 72, 73

**CS** *Configuration Service*. 55

**CSMA-CA** *Carrier Sense Multiple Access with Collision Avoidance*. 13

**DAC** *Device Access Code*. 11

**DF** *Directory Facilitator*. 31–33

**DIAC** *Dedicated Inquiry Access Code*. 11

**DLL** *Data Link Layer*. 12

**DS** *Discovery Service*. 55

**DSSS** *Direct Sequence Spread Spectrum*. 15

**FFD** *Full Function Device*. 16

**FIPA** *Foundation for Intelligent Physical Agents*. ix, xi, 23, 30, 31, 33, 34, 36–38, 64

**FIPA-ACL** *Foundation for Intelligent Physical Agents - Agent Communication Language*. ix, 27, 28, 31, 37

**GIAC** *General Inquiry Access Code*. 11

**GPS** *Global Positioning System*. xi, 1, 45, 46

**GUI** *Graphical User Interface*. 31

**GUID** *Globally Unique Identifier*. 32

**HEC** *Header-Error-Check*. 11

**IA** *Inteligência Artificial*. 1–3, 22, 24, 27, 39

**IAC** *Inquiry Access Code*. 11

**IEEE** *Institute of Electrical and Electronics Engineers*. 6

**IIOP** *Internet Inter-ORB Protocoll*. 37

**ISM** *Industrial, Scientific and Medical*. 7

**J2SE** *Java 2 Platform, Standard Edition*. 66

**JADE** *Java Agent DEvelopment*. ix, xi, 30–38, 64, 94

**JRE** *Java Run-time Enviroment*. 32

**JSR** *Java Specification Request*. 66, 67

**JVM** *Java Virtual Machine*. 31, 32

**KB** *Knowledge Base*. 71

**KQML** *Knowledge Query and Manipulation Language*. ix, xi, 27, 28

**KSE** *Knowledge Sharing Effort*. 27

**L2CAP** *Logical Link Control and Adaptation Layer*. 8

**LAP** *Lower Address Part*. 11

**LBS** *Location Based Services*. ix, xiv, 3, 4, 51, 57, 59, 60, 94

**LEA** *Location Estimation Algorithms*. ix, 47, 48, 57, 62–64, 73, 76, 94, 95

**LEAP** *Lightweight Extensible Agent Plataform*3. 30

**LGPL** *Lesser General Public License*. 30

**LIS** *Location Inference Service*. xii, 55, 56

**LLC** *Logical Link Control*. 12, 13, 19

**LME** *Layer Management Entity*. 16, 19, 20

**LMP** *Link Management Protocol*. 8

**MAC** *Media Acess Control*. viii, 12–14, 16, 18–20, 55, 67, 69

**MCPL** *MAC Commom Part Layer*. 13

**MCPS** *MAC Common Part Sublayer*. 13

**MF** Mapa de *Fingerprinting*. 47, 48, 64, 77, 78, 95

**MIMO-OFDM** *Multiple-Input and Multiple-Output Orthogonal Frequency-Division Multiplexing*. 21

**MLME** *Sub Layer Managment Entity*. 13

**MLP** *Multi Layer Perceptron*. xi, 39–41

**MNN** *Multiple Nearest Neighbour*. 51, 55, 95

**MoCA** *Mobile Collaboration Architecture*. xiv, 54, 55

**MSSID** *Mutiple Service Set Identifier*. 67

**NAP** *Non-significant Part*. 11

**NWK** *Network Layer*. 14

**OBEX** *Object Exchange Protocol*. 8

**OBS** *Observation*. 72

**OFDM** *Orthogonal Frequency-Division Multiplexing*. 21

**OSI** *Open System Interconnection*. 8, 13, 16

**OSS** *One Step-Secant*. 49

**PA** Ponto de Acesso. xii, 16, 18, 19, 21, 48, 52–55, 57, 58, 60, 61, 63, 66, 67, 69, 73, 77, 95

**PAN** *Personal Area Networks*. 7, 13

**PD-SAP** *Physical Data Service Acess Point*. 13

**PHY** *Camada Física*. viii, 12, 14–16, 18–20

**PIC** *Programable Integrated Circuit*. 67

**PLCP** *Physical Layer Convergence Protocol*. 18

**PLME** *Physical Layer Managment Entity*. 13

**PMD** *Physical Medium Dependent*. 18

**QoS** *Quality of Service*. ix, 3, 4, 59, 60, 62–64, 71, 72, 76, 79, 90, 94, 95

**RBF** *Radial Basis Functions*. 50

**RF** *radiofreqüência*. 1, 7, 15–18, 47, 55

**RFD** *Reduce Function Device*. 16

**RM** *Knowledge Managed-based*. 73

**RMI** *Remote Method Invocation*. 31, 32, 37, 95

**RNA** *Rede Neural Artificial*. xii, xiv, 1–5, 21, 22, 39, 40, 43, 44, 48, 49, 57, 59–64, 71–73, 75–77, 79, 81, 82, 85–95

**RSS** *Received Signal Strength*. 1, 47–50

**RSSI** *Received Signal Strength Indicator*. 2, 3, 6, 17, 19, 20, 47, 48, 52, 55, 59, 61–64, 66, 67, 69, 71–73, 76–79, 85, 90, 94

**RTLS** *Ekahau Real-Time Location System*. 54

**SAP** *Service Access Point*. 14

**SMA** *Sistema Multiagente*. 1, 3–5, 21–25, 28–31, 35, 39, 59, 64, 76, 90, 94, 95

**SNR** Sinal-Ruído. 47

**SRM** *Structural Risk Minimization*. 50

**SRM1** *Symbolic Region Manager*. 55

**SSID** *Service Set Identifier*. 67

**SVM** *Support Vector Machines*. 48, 50, 57, 95

**TDD** *Time Division Duplexing*. 10

**TDoA** *Time Difference of Arrival*. xii, 45, 46

**TRCK** *Tracking*. 73

**UAP** *Upper Address Part*. 11

**UDP** *User Datagram Protocol*. 8

**USB** *Universal Serial Bus*. 67

**WAP** *Wireless Application Protocol*. 8

**WLAN** *Wireless Local Area Network*. 1, 6

**WMAN** *Wireless Metropolitan Area Networks*. 6

**WPAN** *Wireless Personal Area Networks*. 6, 7, 12

**WWAN** *Wireless Wide area Network*. 6

**ZC** Coordenador *Zigbee*. 16

**ZDO** *Zigbee Device Object*. 14, 15

**ZED** *Zigbee End Device*. 16

**ZR** Roteador *Zigbee*. 16

# Capítulo 1

# Introdução

Os dispositivos móveis permitem a mobilidade de usuários, dando acesso a variados recursos computacionais enquanto se deslocam de um ponto para outro. Com a mobilidade dos usuários várias técnicas e modelos são necessários para dinamicamente adequar as aplicações conforme diferentes contextos e ambientes. Neste sentido, cresce o interesse por aplicações sensíveis ao contexto, as quais permitem a interação de usuários com o ambiente onde se inserem, podendo ser feita de forma dinâmica e inteligente (Niessink & Vliet, 1998; Dey et al., 2001).

Dey (2001) define o contexto como qualquer informação que possa ser usado para caracterizar a situação de uma entidade (pessoa, lugar ou objeto). Esta informação é considerada relevante para a interação entre usuário e aplicativo. Nesse sentido, muitos atributos podem ser usados para descrever um contexto.

Uma das informações de contexto mais importante é a localização do usuário (Abowd et al., 1999; Abowd & Mynatt, 2000). A localização do usuário pode ser feita utilizando uma infra-estrutura de rede sem fio local ou *Wireless Local Area Network* (WLAN), através da medição da intensidade do sinal recebido, ou *Received Signal Strength* (RSS) ((Kaemarungsi & Krishnamurthy, 2004; Swangmuang & Krishnamurthy, 2008; Noh et al., 2008; Honkavirta et al., 2009)). Assim, qualquer infra-estrutura de rede local ou WLAN pode ser utilizada para sistemas de posicionamento, uma vez que técnicas de mapeamento de RSS tem se mostrado melhor que a utilização do sistema de posicionamento global ou GPS em ambientes fechados (Swangmuang & Krishnamurthy, 2008).

Atualmente existem diversas técnicas para determinar em qual cômodo um determinado usuário está localizado. Dentre essas, citam-se duas vertentes, cuja primeira faz uso de modelos matemáticos usando princípios de propagação de sinal Savvides et al. (2001); Nunes (2006) e a segunda linha faz uso de *Inteligência Artificial* (IA) para determinar a posição do dispositivo (Ferris et al., 2007; Bento et al., 2005).

O problema de localização se mostra mais complexo quando se verifica a existência de diversos dispositivos eletrônicos, transmitindo simultaneamente sinais na mesma *radiofreqüência* (RF), que podem ser usados para determinar posição de um dispositivo eletrônico, porém em padrões diferentes, como por exemplo *Bluetooth* (IEEE 802.15.1) (Altini et al., 2010), *Zigbee* (IEEE 802.15.4) (Noh et al., 2008) e *Wi-Fi* (IEEE 802.11) (Song et al., 2004). Diante do exposto, esse trabalho aborda o problema de localização em ambientes fechados utilizando técnicas de RNA e *Sistema Multiagente* (SMA) aplicados aos padrões *Bluetooth* (IEEE 802.15.1), *Zigbee* (IEEE 802.15.4) e *Wi-Fi* (IEEE 802.11).

## 1.1 Caracterização do Problema

A localização de dispositivos eletrônicos oferece mais informações ao contexto, visando adequar a oferta de serviços de acordo como as necessidades dos usuários. Segundo Bilurkar et al. (2002); Rodriguez et al. (2004); Tsai et al. (2008), uma das técnicas mais utilizadas para localização de dispositivos móveis é a RNA. Os algoritmos de RNA que mais se destacam na literatura são: *Backpropagation* Tsai et al. (2008), *Levenberg-Marquardt* Wei et al. (2008) e *Kohonen* Parodi et al. (2008), sendo estes aplicados aos protocolos, *Bluetooth*, *Zigbee* e *Wi-Fi*. Os algoritmos *Backpropagation*, *Levenberg-Marquardt* serão descritos no Capítulo 3; enquanto o algoritmo de *Kohonen* não será abordado neste trabalho devido ao fato dele não se adequar a granularidade do espaço físico utilizado conforme será exposto no Capítulo 5.

Battiti et al. (2002) foi um dos primeiros artigos no tema, o qual propõe a utilização de uma RNA com dois neurônios na camada de saída, correspondendo desta forma às coordenadas $X$ e $Y$. Já em Saha et al. (2003), foi proposto uma RNA com dezenove neurônios na camada de saída, onde cada um desses neurônios corresponde a uma célula do ambiente. Em Song et al. (2004) encontramos resultados envolvendo um ambiente real de redes sem fio, onde os resultados foram obtidos utilizando dados empíricos coletados no campus de *Dartmouth College* com uma precisão de 65% a 72% na predição utilizando a técnica de *Markov* de ordem $k$, dentre outras técnicas.

Os resultados de Battiti et al. (2002); Saha et al. (2003); Song et al. (2004) demonstram a viabilidade da utilização de detecção de padrões para resolver o problema de localização de dispositivos móveis. Resultados publicados em Bahl & Padmanabhan (2000) demonstram ainda uma considerável eficiência que pode ser obtida através do uso de modelos matemáticos com aspectos de propagação de sinais.

Como visto, o processo de localização de dispositivos móveis em ambientes fechados apresenta uma considerável complexidade ligada principalmente à escolha de várias técnicas e algoritmos juntamente com a escolha de qual protocolo de rede deve ser utilizado. Outro fator crítico para este problema relaciona-se ao grau de acurácia necessário na aplicação, uma vez que altos níveis de acerto podem demandar um refinamento no algoritmo, troca do protocolo de rede ou escolha de outras técnicas e modelos.

A utilização de modelos matemáticos com princípios de propagação de sinais ou *Received Signal Strength Indicator* (RSSI) é extremamente trabalhosa e pode onerar o processo de decisão em ambientes dinâmicos. Dessa maneira, técnicas de IA estão sendo constantemente estudadas para a solução do problema de localização de usuários com dispositivos móveis.

## 1.2 Justificativa

Atualmente cresce a necessidade de aplicações que possuam ciência de pelo menos duas informações de contexto relacionadas aos usuários: o contexto social do usuário (*social-awareness*) e o contexto de localização (*location-awareness*). O primeiro contém informações sobre quem são os colaboradores, amigos, quais são as suas preferências, interesses e atividades. Já o último contém informações sobre onde está o usuário, a partir da aplicação de uma ou da combinação de várias técnicas de localização existentes

(Want et al., 1992; Bahl & Padmanabhan, 2000; Hightower & Borriello., 2001). Alguns fatores que vem alavancando as pesquisas nessa área são:

- Utilização da infra-estrutura de rede sem-fio;
- Utilização de SMA e RNA.

Uma vez de posse das informações de localização dos usuários, as aplicações são capazes de perceber em tempo real a proximidade do usuário. Dessa maneira, as aplicações podem pró-ativamente alertar o usuário através de seu dispositivo móvel, ou ainda, recomendar o novo serviço, caso ainda não o tenha; criando serviços que se beneficiam da localização do usuário para tomar alguma decisão específica, LBS.

Com o propósito de dar suporte ao desenvolvimento de aplicações para LBS, têm surgido diversas propostas, que tentam capturar as informações de posicionamento dos dispositivos móveis dos usuários (Hightower & Borriello., 2001; Kaemarungsi, 2005; Ahmed et al., 2011). No entanto, as soluções existentes suportam parcialmente as redes sem-fio mais comuns, em sua maioria não há possibilidade de extensão e não há garantia de qualidade na informação apresentada.

Conforme o exposto, não há solução atual facilmente extensível que auxilie na tarefa de detecção de usuários móveis em ambientes fechados. Neste sentido, o uso de SMA mostra-se de fundamental importância. Tarefas como obtenção dos RSSI, execução de diferentes algoritmos de RNA e avaliação de conhecimento encontrado são adequadas para integração com SMA, considerando os aspectos de distribuição, automatização de tarefas ou execução de algoritmos em paralelo. Assim, este trabalho tem como objetivo propor uma solução extensível que auxilie na localização de usuários móveis em ambientes fechados com uso de qualidade de serviço ou QoS.

## 1.3 Objetivos

Este projeto tem por objetivo geral a avaliação do uso de algoritmos de IA mais especificamente de RNA, com dados RSSI vindos de três padrões de rede sem fio (*Bluetooth, Zigbee e Wi-Fi*), a fim de desenvolver um prototipo de ferramenta de localização de dispositivos móveis em ambientes fechados com uso da abordagem de SMA e aspectos de QoS.

Como objetivos acessórios podemos citar:

- Avaliar o uso de diferentes algoritmos de RNA para levantar indicadores de desempenho, discrepância, erro, acurácia entre outras características;
- Integrar os aspectos característicos dos dispositivos móveis, dos protocolos de rede sem-fio ao ambiente físico através de SMA e
- Desenvolver um protótipo que seja extensível para o desenvolvimento de *middlewares*, *frameworks* integráveis de forma flexível.

Dada a importância da aplicação de técnicas de IA para localização de dispositivos móveis e considerando o fato de não termos encontrado trabalhos na literatura que utilizem RNA com abordagem de SMA, os quais apresentem uma análise comparativa dos

resultados com padrões *Bluetooth*, *Zigbee*, *Wi-Fi*, o presente trabalho vai na direção de preencher esta lacuna.

A partir dos resultados obtidos, foram realizadas comparações entre os três padrões com o intuito de indicar desempenho, discrepância, erro, acurácia entre outras características. Todos os resultados servirão de subsídios para trabalhos que necessitem da localização de dispositivos móveis em ambientes fechados.

Um outro aspecto importante da proposta é a integração de SMA com RNA e QoS, permitindo que a tarefa de localização de usuários seja parte de uma arquitetura maior, sendo integrada ao ambiente, o que permite uma maior interação entre os usuários e o mundo real.

Por definição, cada agente de um SMA possui uma respectiva autonomia, com capacidade de comunicação, cooperação, raciocínio, detendo mecanismos de planejamento para executar suas ações, sendo então adaptável às mudanças ocorridas no ambiente. Vale ressaltar que outras características de oferta de serviço orientado a contexto podem ser exploradas, pela, junção de SMA com RNA para localização de usuários em ambientes fechados, como por exemplo *Location Agent* explorado neste trabalho, através de uma solução extensível e integrável a aplicações sensíveis ao contexto.

## 1.4 Contribuições

Uma vez que este trabalho de pesquisa cobre múltiplas abordagens as principais contribuições do *LocationAgent* com suporte eficazmente a LBS foram:

- Solução comparativa entre protolos de redes *Bluetooth*, *Zigbee* e *Wi-Fi* com uso da abordagem SMA, RNA e aspectos de QoS;
- A aquisição de dados usando diferentes sensores de diferentes tecnologias IEEE 802.11.1 (*Bluetooth*), IEEE 802.15.4 (*Zigbee*) e IEEE 802.11 (*Wi-Fi*), possibilitando a integração de novas tecnologias de rede que possam surgir;
- Obtenção dinâmica da localização para algoritmos do tipo *Backpropagation*, *Backpropagation Momentum* e *Levenberg-Marquart*;
- Possibilidade de testes nos mais diversos ambientes, pois todos os sensores usados são dispositivos móveis autônomos, porém o mapa de *fingerprint* deve ser adicionado juntamente com os novos agentes que implementam os algoritmos de RNA;
- O protótipo implementado pode ser usado por diversos serviços que dependam da localização através da Interface de comunicação;
- O modelo arquitetural desenvolvido em camadas é suficientemente genérico e aberto, possibilitando a adição de novos protocolos de rede e técnicas de localização.

## 1.5 Estrutura do Documento

A partir da revião da literatura relativa às áreas envolvidas neste trabalho, os Capítulos 2 e 3 apresentam o estudo das áreas de padrões de redes sem-fio, SMA e RNA. O estudo da arte na área de localização de usuários em ambientes fechados foi e apresentada no

Capítulo 4 para que se pudesse conhecer melhor o contexto de aplicação desta pesquisa, onde são apresentados alguns trabalhos correlatos e a análise dos principais estudos na área.

A partir desses estudos, no Capítulo 5 é apresentada a proposta de solução para o problema de localização de usuários em ambientes fechados. Inicialmente foi proposta uma solução baseada na integração de SMA e RNA, a qual foi testada. No Capitulo 5 é apresentado o protótipo de uma arquitetura de integração de SMA, RNA e padrões de redes sem-fio, denominada *Location Agent* para automatização do processo de aquisição dos sinais utilizando RNA. O Capitulo 6 descreve os experimentos realizados utilizando o protótipo do *Location Agent* desenvolvido. Assim, Conclusões e Trabalhos Futuros são discutidos no Capítulo 7.

# Capítulo 2

# Protocolos de Rede Sem Fio

Uma alternativa aos sistemas baseados em sinais de ultra-som, de infravermelho, de antenas de celular e de satélites para resolver o problema de localização em ambientes fechados é a utilização de sinais de redes *wireless*. A exploração desses sinais de rede minimiza o custo do investimento em infra-estrutura, uma vez que faz uso da infra-estrutura atual. O sinal de rádio-freqüência captado por um aparelho receptor possui basicamente três atributos que representam informações relevantes para localização:

- o ângulo em que o sinal chega;
- o instante em que o sinal chega (informação que pode ser usada para estimar o tempo de percurso entre a antena emissora e receptora) e
- a potência com que esse sinal é recebido (em sistemas de localização essa informação é chamada de RSSI).

Quanto mais exata a informação, melhor a qualidade das estimativas de localização. Portanto existem diversos padrões de comunicação criados pelo *Institute of Electrical and Electronics Engineers* (IEEE) que utilizam tecnologia sem fio. A Figura 2.1 ilustra esses padrões, classificando-os de acordo com o alcance (*Wireless Personal Area Networks* (WPAN), WLAN, *Wireless Metropolitan Area Networks* (WMAN), *Wireless Wide area Network* (WWAN)) e taxa de transmissão de dados (0.01, 0.1, 1, 10, 100 e 1000 Mbps).

Neste capítulo apresentaremos os padrões *Bluetooth* (IEEE 802.15.1), *Zigbee* (IEEE 802.15.4) e *Wi-Fi* (IEEE 802.11) utilizados neste trabalho.

## 2.1 *Bluetooth*

Em 1994, a *Ericsson Mobile Communications*, iniciou pesquisas para criar uma interface de comunicação entre telefones celulares e acessórios com restrições de baixo consumo de energia e custo reduzido. O objetivo de estudo era encontrar meios de se eliminar o excessivo número de cabos entre os dispositivos. Juntou-se a *Ericsson* as empresas *IBM, Intel, Nokia e Toshiba*; que formaram, em maio de 1998, o *Bluetooth Special Interest Group*. Em julho de 1999 este grupo publicou a especificação 1.0 do *Bluetooth*. O objetivo foi definir uma especificação aberta para o *Bluetooth* e critérios de qualidade para seus produtos. Atualmente, esse grupo é composto por mais de 2000 empresas Group.

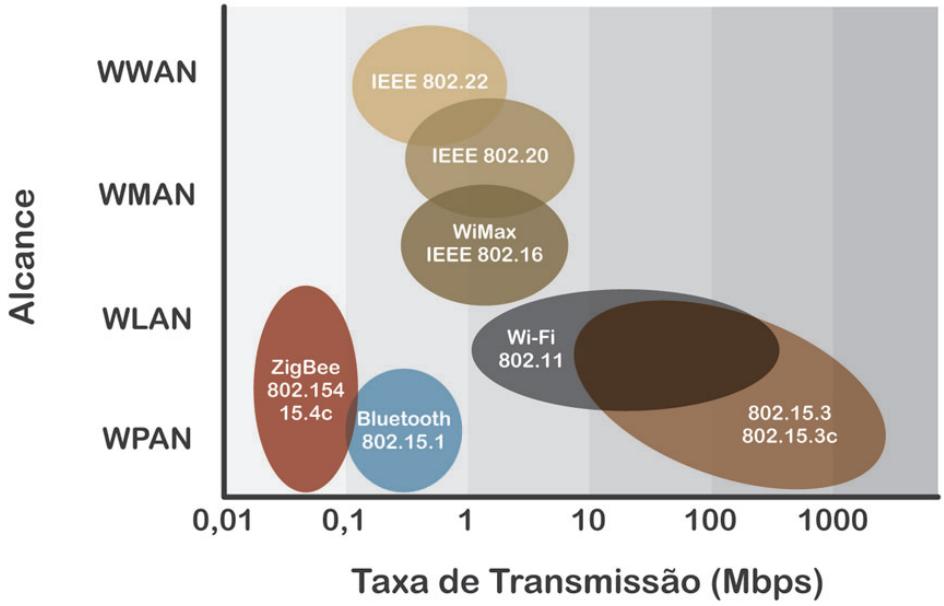



Figura 2.1: Divisão dos Padrões de Redes Sem-Fio (adaptado de (Stallings, 2004)).

A origem do nome deve-se a um rei Viking Dinamarquês que viveu no Século X, Harald Blåtand (em inglês Blåtand significa dente azul). Blåtand foi proveniente de um dos hábitos alimentares de seu pai, Gorm, que tinha simpatia por uma fruta chamada *Blueberries*, cuja forte coloração azul-escura pigmentava os dentes. Harald unificou e controlou Dinamarca e Noruega. Deste fato surgiu a relação de unificar e controlar dispositivos através do *Bluetooth*.

*Bluetooth* é um dos padrões para comunicação sem-fio, por RF de curto alcance. Segundo Miller (2001), foi originalmente concebido como uma tecnologia de substituição de cabos. O *Bluetooth* é modelo de conexão transparente, de baixo custo e consumo de energia. Por essas características se tornou ideal para os dispositivos portáteis.

A especificação define os requisitos de *hardware* e *software* para permitir comunicação ponto-a-ponto e ponto-a-multiponto em RF e WPAN. Múltiplos dispositivos de comunicação e computação podem ser conectados sem a necessidade de cabos proprietários para a transmissão simultânea de voz e dados entre os dispositivos.

Uma das vantagens do *Bluetooth* são as redes pessoais *ad hoc* sem-fio, conhecidas como *Personal Area Networks* (PAN). As glsPAN são redes altamente dinâmicas e transparentes, onde o usuário pode se associar ou desvincular-se rapidamente. O *Bluetooth* opera na banda *Industrial, Scientific and Medical* (ISM) em 2.4GHz. Esta banda sofre pouca interferência por ser pouco utilizada. Porém, fontes como fornos de microondas e tecnologias de redes locais (IEEE802.11 e *HomeRF*) podem gerar consideráveis interferências. Para minimizar as interferências há a técnicas de espalhamento espectral, onde criar boas instalações podem reduzir esse problema.

Outra característica do *Bluetooth* é que pela especificação são definidos dois sistemas de saltos de freqüência, um que trabalha sobre 79 canais, e outro sobre 23. Esta divisão é devido à distribuição irregular da faixa de freqüência disponível para a banda 2.4 GHz ISM no mundo. Países como Japão e Espanha enfrentam este problema, pois nesses países

somente o sistema de 23 canais é operacional.

Como o *Bluetooth* foi concebido para operar em ambientes com elevadas fontes de interferência. O esquema de rápido reconhecimento de pacotes e saltos de freqüência torna a conexão mais robusta. O módulo de rádio evita a interferência de outros sinais alterando a freqüência após transmitir ou receber um pacote. Comparando com outras tecnologias que operam na mesma freqüência *Bluetooth* é o menos suscetível a interferências, pois executa saltos com maior regularidade e utiliza pacotes menores (Miller, 2001).

A especificação *bluetooth* define três classes possíveis de dispositivos, a Tabela 2.1 lista as classes e as configurações.

Tabela 2.1: Tabela de Classe e Configurações (adaptada de (Miller, 2001)).

| CLASSE | POTÊNCIA MÁXIMA DE TRANSMISSÃO (mW) | POTÊNCIA MÁXIMA DE TRANSMISSÃO (dBm) | ALCANCE (m) |
|---|---|---|---|
| 1 | 100 | 20 | 100 |
| 2 | 2.5 | 4 | 10 |
| 3 | 1 | 0 | 0,1 |

### 2.1.1 Pilha de Protocolos

Seguindo o modelo de referência *Open System Interconnection* (OSI), a especificação *Bluetooth* utiliza a segmentação em camadas com o objetivo de alcançar interoperabilidade entre aplicações de dispositivos remotos. O *Bluetooth* possui protocolos específicos da tecnologia como: *Link Management Protocol* (LMP) e *Logical Link Control and Adaptation Layer* (L2CAP), e protocolos comuns a outras plataformas como *Object Exchange Protocol* (OBEX), *User Datagram Protocol* (UDP) e *Wireless Application Protocol* (WAP). A Figura 2.2 apresenta os protocolos.

Pelo fato do *Bluetooth* utilizar protocolos já existentes nas camadas superiores fica mais fácil a adaptação de aplicações para trabalhar sobre a tecnologia *Bluetooth*, inclusive auxiliando na interoperabilidade. A pilha de protocolos, ilustrada na Figura 2.2, consiste de quatro camadas organizadas como descrito na Tabela 2.2 (Group).

### 2.1.2 Canal Físico

O canal físico é representado por uma seqüência de saltos pseudo-aleatórios através de dois sistemas de saltos, um de 79 freqüências e outro de 23. O canal é divido em segmentos de tempo de 625 $\mu$ s, onde cada segmento corresponde a um salto de freqüência numerado ciclicamente de 0 a 227–1 de acordo com o relógio do mestre.

A técnica de Saltos de Freqüência faz uso da freqüência de recepção dividida pela transmissão da portadora que muda periodicamente. Durante a transmissão de uma informação, o dispositivo transmite em uma freqüência por um curto intervalo de tempo (um segmento), e então salta para outra freqüência. Pela utilização deste método há a redução da probabilidade de interferência. Os saltos de freqüência são definidos por um código, que é derivado da configuração dos dispositivos envolvidos.

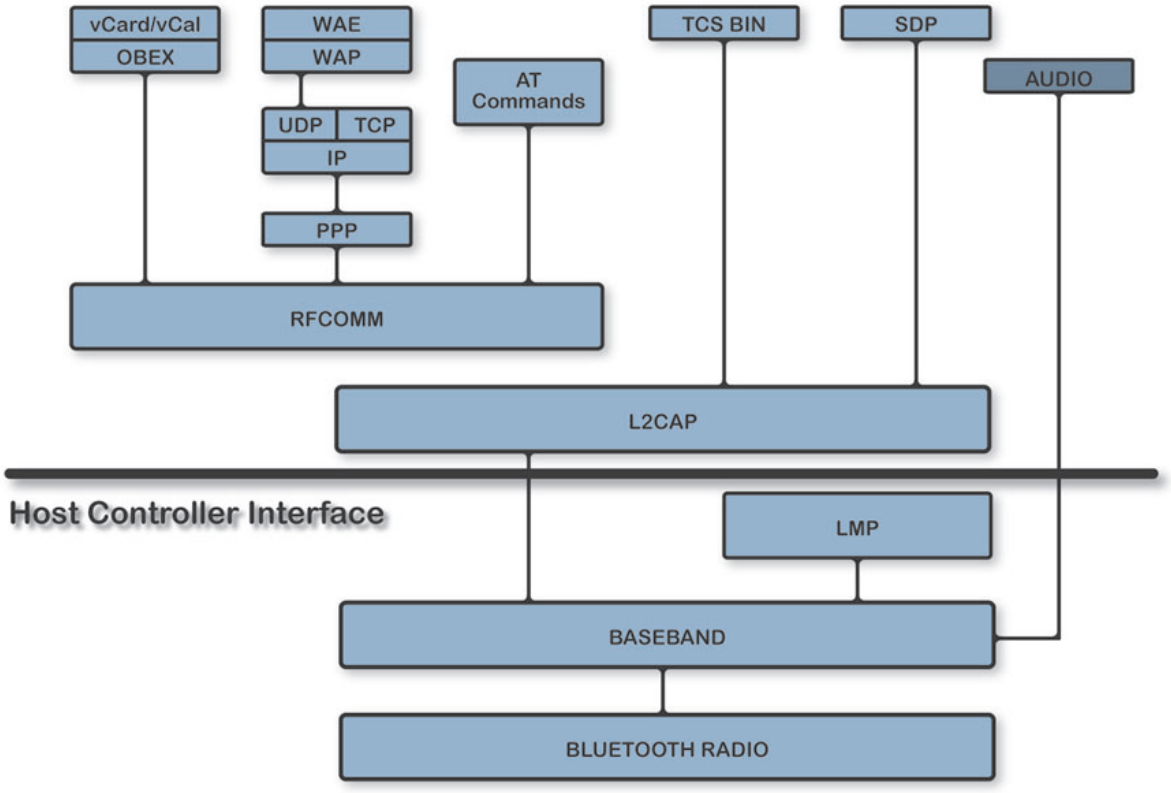



Figura 2.2: Segmentação do *Bluetooth* (adaptado de (Miller, 2001)).

Tabela 2.2: Tabela de Camadas e Configuração (adaptado de (Group)).

| CAMADA DO PROTOCOLO | PILHA DE PROTOCOLOS |
|---|---|
| Bluetooth Core Protocols | Rádio<br>Baseband<br>Link Management Protocol<br>Logical Link Control and Adaptation Protocol (L2CAP)<br>Service Discovery Protocol (SDP) |
| Cable Replacement Protocols | Radio Frequency Communication (RFCOMM) |
| Telephony Control Protocols | Telephony Control Specification Binary (TCS BIN)<br>AT-Commands |
| Adopted Protocols | Point-to-point Protocol (PPP)<br>UDP/TCP/IP<br>Object Exchange Protocol<br>*Wireless* Application Protocol<br>VCard<br>Infrared Mobile Communications (IrMC)<br>*Wireless* Application Environment (WAE) |

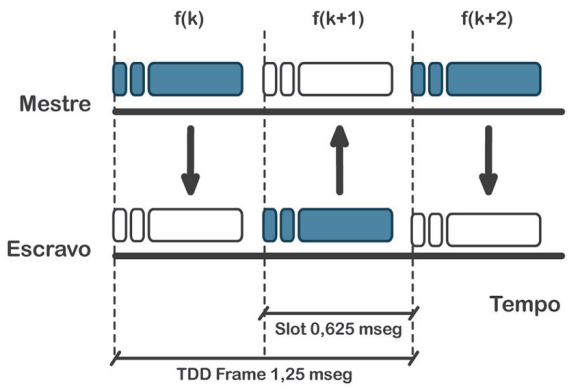



Figura 2.3: Divisão do Tempo (adaptada de (Miller, 2001)).

No *Bluetooth*, cada *piconet* (conjunto de dispositivos) possui somente um mestre, cujo endereço determina a seqüência de saltos. A seqüência de saltos é determinada pelo relógio do mestre, pois deve haver sincronização entre os dispositivos. O mestre controla o tráfego no canal através de um esquema de *polling*. Por definição, o mestre é a unidade que inicia a conexão. Os escravos possuem uma estimativa do relógio do mestre e adicionam um *offset* para se sincronizar ao mestre.

O esquema *Time Division Duplexing* (TDD), que pode ser visualizado na Figura 2.3, é utilizado para a transmissão alternada entre mestre e escravo. O TDD utiliza um único canal para enviar e receber as informações, alternando a direção da transmissão ao recebê-las ou transmití-las. A diferenciação entre os sinais é feita de acordo com o domínio do tempo.

A comunicação é *full-duplex*, ou seja, a princípio, a comunicação segmentada invalida a afirmação anterior, mas o rápido chaveamento entre recepção e transmissão inibe a

percepção do atraso. Isto o torna adequado para o tráfego de voz e dados. Além disso, o montante da largura de banda alocada em cada direção é flexível.

Para manter a sincronização, não há saltos durante a transmissão dos pacotes. O mestre sempre transmite nos segmentos pares, e o escravo nos ímpares. Os pacotes podem ocupar 1, 3 ou 5 segmentos. Na transmissão de pacotes que ocupam mais que um segmento, a freqüência é mantida constante.

### 2.1.3 Endereçamento e Pacotes *Baseband*

Os dispositivos *Bluetooth* possuem um endereço único de 48 *bits*, o *Bluetooth Device Address* (BD-ADDR). Este endereço é derivado do padrão IEEE 802. O BD-ADDR é dividido em três partes:

- *Lower Address Part* (LAP), 24 *bits* utilizados para geração dos códigos de *Inquiry*;
- *Upper Address Part* (UAP), 8 *bits* utilizados para geração do código de verificação da integridade do cabeçalho, *Header-Error-Check* (HEC);
- *Non-significant Part* (NAP) , 16 bits irrelevantes ao endereçamento, sendo útil somente para o discernimento do fabricante do equipamento.

Desta forma, o endereço BD-ADDR tem como parte significante o LAP e UAP. Toda informação transmitida pelas camadas superiores é segmentada em pacotes *Baseband*. Cada pacote é composto por 3 campos: *access code* (68/72 *bits*), cabeçalho (54 *bits*), e *payload* (0-2745 *bits*).

O *access code* é utilizado para sincronização, compensação do *offset*, *paging* e *inquiry*. O *access code* é derivado do endereço do dispositivo (BD-ADDR), sendo que há três tipos diferentes de *access code*:

- *Channel Access Code* (CAC), código que identifica a *piconet*. Este código é derivado do BD-ADDR do mestre e está presente em todos os pacotes trocados no canal da *piconet*;
- *Device Access Code* (DAC), código utilizado para procedimentos especiais de sinalização durante o processo de estabelecimento da conexão;
- *Inquiry Access Code* (IAC), código utilizado para a operação de *inquiry*. O IAC conta com duas variações:
  - *General Inquiry Access Code* (GIAC), utilizado para descobrir os dispositivos ao alcance;
  - *Dedicated Inquiry Access Code* (DIAC), utilizado para descobrir grupos de dispositivos com características específicas. O GIAC é comum a todos os dispositivos, enquanto que o DIAC é um primeiro nível de filtro de serviços, pois é comum somente à classe (tipo) do dispositivo.

O cabeçalho contém informações para o controle da conexão. Os campos contidos no cabeçalho são utilizados para o reconhecimento, controle de fluxo, endereço do escravo, verificação de erros e ordem dos pacotes.

O *payload* contém as informações a serem transportadas: voz ou dados. No *payload* dos pacotes de dados há um cabeçalho adicional de 2 *bytes* de tamanho que indica o canal lógico, controle de fluxo sobre os canais lógicos, e possui um indicador do tamanho do *payload*.

## 2.2 *Zigbee*

A *Zigbee Alliance* é uma associação que atualmente é composta por mais de 150 empresas em sua maioria fabricantes de semicondutores Alliance. A *Zigbee Alliance* trabalha para desenvolver um padrão capaz de possibilitar flexibilidade, mobilidade, com um controle seguro, de baixo custo e potência em redes sem fio para o monitoramento remoto e soluções no campo da automação. Dessa maneira, o *Zigbee* foi proposto no ano 2003 e foi construído e homologado pelo padrão IEEE 802.15.4 o qual define a camada física e a camada de controle de acesso ao meio MAC.

O protocolo *Zigbee* pertence ao grupo das WPAN, sendo criado para a comunicação sem fio de dispositivos em curtas distâncias e baixa taxa de transmissão, obtendo como resultado uma racionalização no consumo de energia.

A Tabela 2.3 mostra as diferenças entre as tecnologias como o *Zigbee*, *Bluetooth*, *Wi-Fi* e *GPRES/GSM*. Contudo, há certas semelhanças em suas características ou aplicações, apresentado na tabela.

Tabela 2.3: Tabela Comparativa (adaptado de (Koubaa et al., 2007)).

| | ZIGBEE 802.15.4 | BLUETOOTH 802.15.1 | WI-FI 802.11B | GPRS/GSM 1XRTT/CDMA |
|---|---|---|---|---|
| Foco da Aplicação | Monitoramento e Controle | Substituição de cabos | Web, Vídeo, E-mail | WAN, Voz/Dados |
| Recursos do Sistema | 4-32KB | 250KB + | 1MB + | 16MB + |
| Durabilidade da Bateria (dias) | 100-1000 + | 1-7 | 1-5 | 1-7 |
| Nós por Rede | 255/65k + | 7 | 30 | 1.000 |
| Taxa de Transmissão (Kbps) | 20-250 | 720 | 11.000 + | 64-128 |
| Alcance (metros) | 1-75 + | 1-10 + | 1-100 | 1.000 + |
| Atributos | Confiável, Baixa Potência, Baixo Custo | Custo, Conveniência | Velocidade, Flexibilidade | Alcance, Qualidade |

### 2.2.1 IEEE 802.15.4

O padrão IEEE 802.15.4 define as duas primeiras camadas do protocolo *Zigbee*. A PHY e a camada de enlace, também denominada de Controle de Acesso ao Meio MAC. O padrão especifica uma rede de comunicação sem fio caracterizada por baixa taxa de transmissão de dados, baixo consumo de energia, curto alcance, baixa complexidade e custo reduzido.

### 2.2.2 Camada de enlace de dados

A camada de enlace de dados (*Data Link Layer* (DLL)) é dividida em duas subcamadas, a subcamada MAC e a subcamada de Controle de Ligação Lógica (*Logical Link*

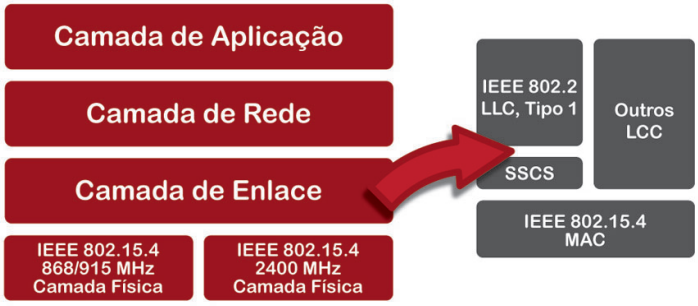



Figura 2.4: Camadas do Padrão *Zigbee* (adaptado de (Faludi, 2010)).

*Control* (LLC)). O LLC é comum a todos os padrões IEEE 802, embora a subcamada MAC dependa do hardware e apresente variação na implementação de sua camada física. A Figura 2.4 mostra o padrão IEEE 802.15.4 baseado no modelo de referência OSI.

A subcamada LLC tem a função de ser a interface entre a camada lógica e as camadas de nível superior, tendo acesso a vários meios definidos pelo IEEE 802.15.4. As trocas de informações de diferentes LLC são feitas através de primitivas.

A subcamada MAC faz a interface entre as camadas físicas e de aplicação, utilizando o algoritmo *Carrier Sense Multiple Access with Collision Avoidance* (CSMA-CA) para se comunicar com a camada física, verificando se o canal de rádio esta liberado antes de começar a transmitir.

Subcamada MAC ainda conta com outras funções que são:

- Acesso ao canal dos dispositivos;
- Determinação do tipo de dispositivo da rede;
- Gerenciamento de energia com baixa complexidade;
- Transporte viável de dados com envio e reconhecimento dos quadros;
- Manutenção da rede associando e desassociando dispositivos.

Ainda, a subcamada MAC é subdividida por duas seções ou subcamadas que se comunicam entre si como mostrado na Figura 2.5: *MAC Commom Part Layer* (MCPL) e a seção MAC *Sub Layer Managment Entity* (MLME). Ambas acessam respectivamente a camada física inferior pelo *Physical Data Service Acess Point* (PD-SAP) e pelo *Physical Layer Managment Entity* (PLME). O MLME gerencia todos os comandos, respostas, indicações e confirmações usadas para gerenciar uma PAN. Já o *MAC Common Part Sublayer* (MCPS) trata os comandos relacionados aos dados como requisição, indicação e confirmação de dados (Koubaa et al., 2007).

### 2.2.3 Arquitetura do protocolo *Zigbee*

O protocolo *Zigbee* possui cinco camadas baseadas no modelo OSI, onde cada uma executa serviços específicos. As camadas funcionam como se fossem uma entidade de dados e gerência para servir à camada acima. A permuta de serviços entre as camadas é

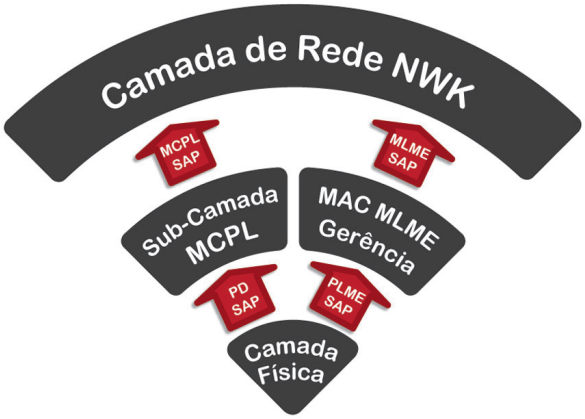



Figura 2.5: Comunicação das Camadas Física e Rede (adaptado de (Koubaa et al., 2007)).

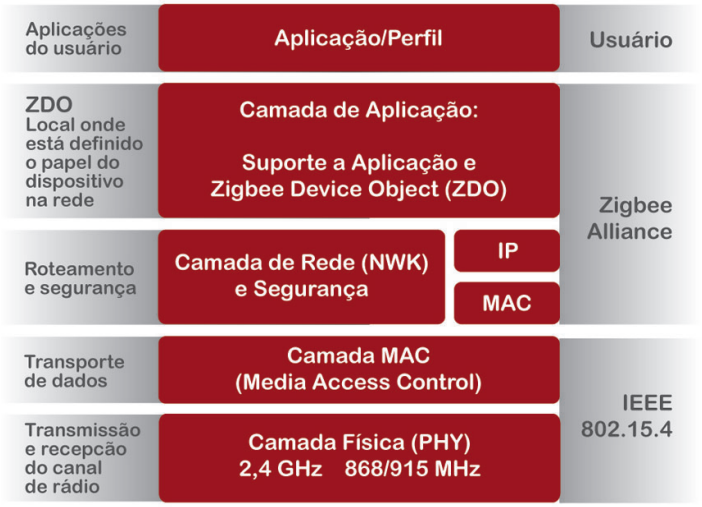



Figura 2.6: Camadas do Padrão *Zigbee* (adaptado de (Faludi, 2010)).

realizada por pontos de acesso ao serviço, chamados *Service Access Point* (SAP), e cada SAP suporta um número de primitivas de serviço para ativar a função solicitada.

A Figura 2.6 ilustra as camadas do protocolo, onde as duas camadas mais inferiores são definidas pelo padrão IEEE 802.15.4. A terceira é a camada física PHY e a quarta é a camada de controle de acesso ao meio MAC.

A *Zigbee Alliance* é responsável pelo aprimoramento das camadas citadas, além de desenvolver a camadas de rede *Network Layer* (NWK) e o *framework* para a camada de aplicação. Outro trabalho da *Zigbee Alliance* é aprimoramento da camada de aplicação que suporta as subcamadas suporte à aplicação *Application Support Sublayer* (APS), objetos de dispositivo *Zigbee Device Object* (ZDO) e os objetos de aplicação definidos pelo fornecedor (*Application Object*).

A camada de rede NWK faz o roteamento das mensagens, estabelecendo os novos endereços e membros da rede. Essa camada possibilita o crescimento da rede sem a

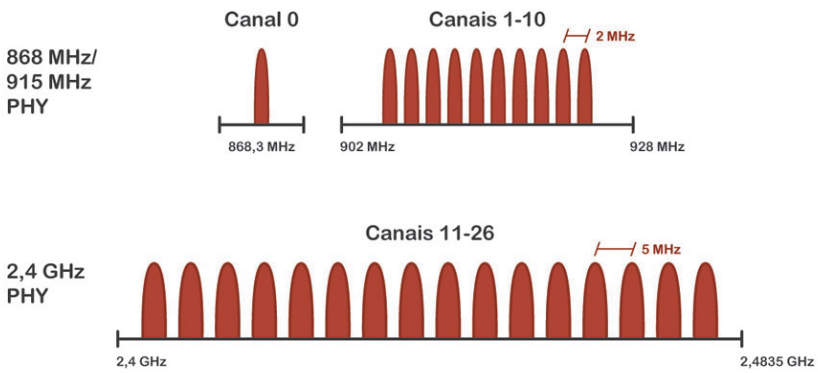



Figura 2.7: Freqüência e Canais (adaptada de (Faludi, 2010)).

necessidade de utilizar equipamentos de transmissão de potência elevada.

A camada APS faz o roteamento dos diferentes pontos de aplicação que funcionam no nó, mantendo as tabelas de ligação (*binding*). Estas tabelas mantêm as conexões compatíveis entre diferentes pontos finais.

Na camada de aplicação há o ZDO responsável pela gerência do dispositivo total e da camada de segurança. Outra função, é encontrar dispositivos na rede e determinar os tipos de serviços que estes dispositivos provêm à rede.

### 2.2.4 Camada Física

É a camada responsável por permitir a transmissão e recepção dos pacotes através de ondas RF, definindo como os dispositivos da rede se comunicam através do canal de comunicação sem fio.

A camada PHY acomoda as interfaces de baixo custo e utiliza técnica de transmissão de seqüência direta *Direct Sequence Spread Spectrum* (DSSS), na qual os dados são transmitidos em várias freqüências (canais) ao mesmo tempo. Essa camada utiliza um canal na banda de 868 MHz, dez canais na banda de 915 MHz, e dezesseis canais na banda de 2.4 GHz, como mostrado na Figura 2.7. Dependendo da freqüência utilizada varia a velocidade de transmissão como mostrado na Tabela 2.4.

Tabela 2.4: Tabela de Relação Freqüência Velocidade de Transmissão do Padrão *Zigbee* (adaptada de (Faludi, 2010)).

| Frequência | Kb/s |
|---|---|
| 868 MHz | 20 |
| 915 MHz | 40 |
| 2,4 GHz | 250 |

### 2.2.5 Dispositivos *Zigbee* e Topologias de Rede

Nas redes *Zigbee* podem ser identificados dois dispositivos principais:(i) dispositivo de função completa *Full Function Device* (FFD) e (ii) o dispositivo de função reduzida *Reduce Function Device* (RFD). Esses dispositivos tem as seguintes funções:

- FFD - pode ser utilizado como função de Coordenador *Zigbee* (ZC), ou como Roteador *Zigbee* (ZR). O coordenador contém todas as informações da rede para poder gerenciá-la, também serve de ponte com outras redes para distribuir mensagens e armazena informações dos nós da rede. O roteador pode fazer intercâmbio de dados através de outros dispositivos, tendo as características de um nó normal na rede, porém com habilidades extras para servir de permuta entre nós;
- RFD - comumente utiliza-se como dispositivo final (*Zigbee End Device* (ZED)). A sua função principal é coletar informações de variáveis por meio de sensores e permanecer em estado de espera (*sleep*) por longos períodos para diminuir seu consumo de energia.

## 2.3 *Wi-Fi*

*Wi-Fi* (*Wireless Fidelity*) é um termo que identifica redes e dispositivos que implementam a especificação IEEE 802.11 para redes sem fio (NETWORKS, a). Uma rede *Wi-Fi* estruturada é composta por dispositivos que se comunicam por sinais de rádio-freqüência dentre os quais um ou mais são PAs. Um PA é um dispositivo que, por um lado, conecta-se à rede cabeada e, por outro, comunica-se com os outros dispositivos *Wi-Fi* através de sinais de RF, servindo como ponte para que tais dispositivos acessem a rede.

Muitos edifícios como *shopping centers*, escolas, escritórios, fábricas e outros já possuem vários PAs em funcionamento, fornecendo acesso às suas respectivas redes. Essa infra-estrutura pode ser utilizada por um sistema de localização que analise os sinais das redes para inferir coordenadas de posição dos dispositivos no ambiente. À primeira vista, apesar de sofrer dos mesmos problemas de imprevisibilidade da forma de propagação do sinal, que os sistemas que usam antenas de celular sofrem, o fato das células de cobertura das redes *Wi-Fi* serem menores, por si só, já implicam maior precisão e exatidão.

### 2.3.1 Especificação ANSI/IEEE 802.11

A especificação ANSI/IEEE 802.11 é um documento que descreve o comportamento e as interfaces da camada física PHY e da subcamada de controle de acesso ao meio MAC para conectividade sem fio em dispositivos fixos ou móveis em uma área local NETWORKS (a). Além da MAC e PHY existe uma entidade que é responsável pelo seu gerenciamento *Layer Management Entity* (LME).

A Figura 2.8 mostra uma pilha com as primeiras camadas do modelo de interconexão de um dispositivo que implementa a especificação 802.11. Ao lado da pilha há uma indicação da correspondência com o modelo OSI e destacando a parte que é descrita na especificação. O quadrado na lateral da pilha representa a LME que faz parte também da especificação.

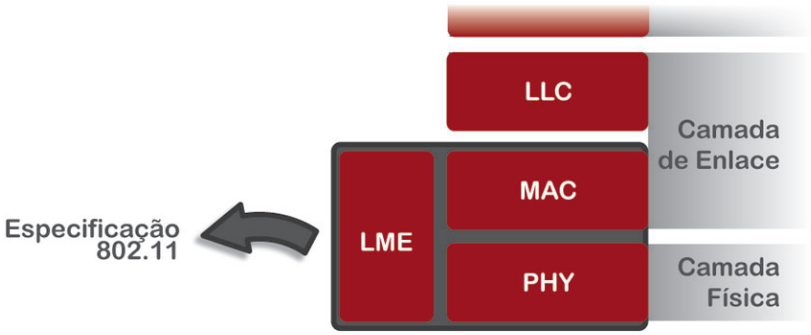



Figura 2.8: Relação Camadas e Sub-Camadas (adaptado de (Stallings, 2004)).

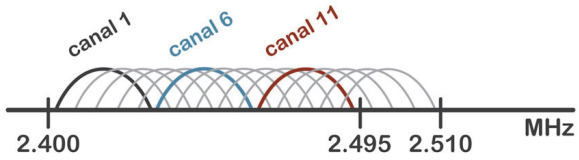



Figura 2.9: Relação de Canais (adaptado de (Stallings, 2004)).

O entendimento do funcionamento dessa estrutura é importante, pois é ela que oferece a informação das RSSI usada no serviço de localização. A potência do sinal recebido, ou o RSSI, é uma informação que os *drivers* da rede *Wi-Fi* sempre fornecem, já que é uma informação útil para a escolha de associação entre possíveis redes disponíveis.

### 2.3.2 Canais de transmissão de dados e interferências

Os sinais de RF emitidos pelos dispositivos *Wi-Fi* concentram-se em uma faixa de freqüência determinada, de 2.400MHz a 2.500MHz. Esta faixa é dividida em 14 canais de banda larga (cerca de 30MHz) onde o sinal transmitido é modulado utilizando uma técnica de espalhamento espectral (Pickholtz et al., 1982). Essa técnica é responsável por diminuir as interferências, que possam ser causadas por emissores externos de RF dentro do mesmo espectro (principalmente fornos micro-ondas (Ahmed et al., 2011)), ou causadas pelos múltiplos caminhos de um mesmo sinal (Kavehrad & McLane, 1987). Pela regulamentação norte-americana dos 14 canais especificados no documento, somente os primeiros 11 podem ser utilizados. Contudo, no padrão europeu são usados 13 canais e no japonês, apenas um, o último.

A especificação 802.11 determina que, nessa faixa de freqüências, os canais devam estar distribuídos de forma adjacentes separados por apenas 5MHz. Tendo como largura da banda de 30MHz, isso implica que os canais se sobrepõem uns aos outros. Dessa maneira, apesar de haver disponíveis de 11 a 13 canais, o número máximo de canais sem sobreposição é três. Tal fato é exemplificado pela Figura 2.9.

A configuração ideal para um ambiente completamente coberto por redes *Wi-Fi* é mostrada na Figura 2.10. Nota-se que dois *grids* adjacentes nunca utilizam o mesmo canal de

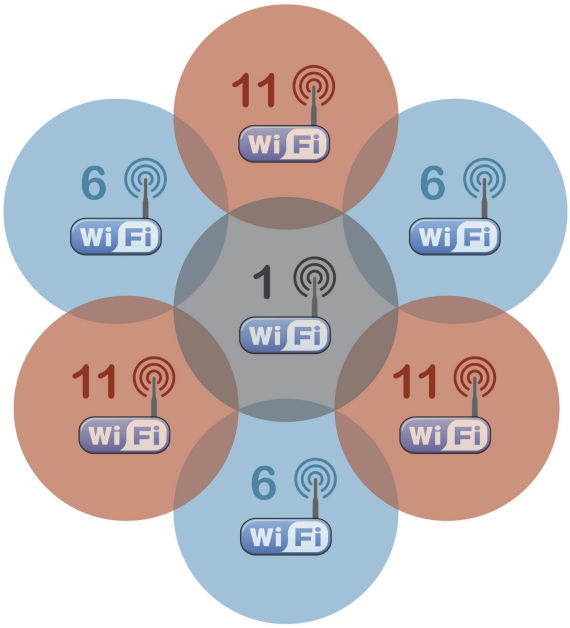



Figura 2.10: Composição de *Grids* e PAs (adaptado de (NETWORKS, b)).

transmissão e nem mesmo canais que se sobrepõem. Segundo especificação (NETWORKS, b), neste caso, um sinal recebido vindo de um *gird* ao qual certo dispositivo está associado, deve ter uma taxa de erro de quadro de no máximo 8%. Mesmo que este sinal esteja na presença de um segundo sinal proveniente de um *grid* adjacente e que tenha intensidade 35dB maior que o primeiro na posição onde se encontra o dispositivo. Desse modo, *grids* vizinhos coexistem sem prejuízo de desempenho, garantindo um funcionamento bastante confiável da rede.

Contudo, na prática essa configuração é rara. Primeiro porque a instalação de PAs que fornecem acesso a uma rede em uma determinada área é feita sem um controle central e organizado. Segundo a quantidade de PAs presentes podem ser excessivos, sendo comum que mais do que três *grids* se sobreponham.

### 2.3.3 A camada PHY

A camada PHY, conforme apresentada na Figura 2.8 é a responsável por receber os quadros da MAC e transformá-los para que possam ser transmitidos na forma de sinais de radio. Ao mesmo tempo em que escuta o meio, detecta ausência e presença de sinais, e quando presentes extraem dos sinais a informação para reconstrução dos quadros que devem ser passados a MAC.

A camada PHY ainda é divida em duas subcamadas: a *Physical Layer Convergence Protocol* (PLCP); e a *Physical Medium Dependent* (PMD). A primeira é responsável por permitir que a MAC opere independente da forma em que se dará o tratamento físico dos quadros. A segunda é a responsável pelo tratamento, providenciando os meios e os métodos de transmissão e recepção de dados através de sinais de RF.

### 2.3.4 A subcamada MAC

A subcamada MAC, conforme apresentada na Figura 2.8, tem como função básica receber quadros que vêm da rede através da camada PHY, repassando-os para a LLC, caso este for destinado ao dispositivo; e controlar o momento de enviar quadros de dados para a rede quadros que a LLC tenha lhe passado com esse propósito, seguindo política de acesso ao meio que procura evitar que dois dispositivos enviem dados ao mesmo tempo. A interface entre a LLC e a MAC é bastante simples, sendo formadas basicamente de primitivas de passagem de quadros em ambas as direções. Contudo, a MAC envia à rede mais do que quadros de dados. Há duas outras classes de quadros que a MAC utiliza: quadros de gerenciamento e quadros de controle.

### 2.3.5 Rastreamento de redes *Wi-Fi*

O processo de rastreamento de redes tem particular importância para a localização *Wi-Fi*, pois ele é a forma mais simples e direta de se obter as RSSIs necessárias para o processo de localização.

Os usuários de dispositivos *Wi-Fi* geralmente têm acesso à LME através de um programa de gerenciamento. Através desse programa os usuários pedem um rastreamento das redes disponíveis para associação. Esse pedido é repassado a MAC através de uma determinada interface, cuja descrição detalhada encontra-se na especificação. Após um intervalo de tempo, a MAC retorna à LME identificações das redes disponíveis, cada qual com seu respectivo indicador de potência de sinal e outros dados. Então, os usuários escolhem uma rede para se associar, fazendo esse pedido à LME, através do programa de gerenciamento, que por sua vez repassa o novo pedido a MAC. Só a partir do momento em que a MAC estabelece a associação é que os dados passados da LLC para a MAC são encaminhados para a rede. Fazer rastreamento de redes disponíveis e se associar a alguma delas entre outras coisas são processos que a MAC executa através da troca de quadros de gerenciamento.

A camada MAC de um dispositivo funciona como PA em uma rede *Wi-Fi*, pois de tempos em tempos envia para a rede um quadro de gerenciamento chamado *Beacon*, que contém a identidade da rede assim como algumas de suas características. Todo *Beacon* que chega é passado à MAC, e vem com endereço de *broadcast*, sendo assim sempre lido, dessa forma temos dois tipos de rastreamento:

- Passivo - consiste em ficar aguardando por um tempo a chegada de *Beacons* que forneçam informação sobre as redes presentes, ou retornar as informações armazenadas de *Beacons* recebidos recentemente. Esse tipo de rastreamento pode ser demasiadamente incerto, ou muito demorado;
- Ativo - consiste em enviar para rede um quadro de *broadcast* chamado *Probe Request*. A MAC de um PA, ao receber um *Probe Request*, retorna imediatamente à rede um *Probe Response* cujo conteúdo é simplesmente o mesmo de um *Beacon* para o endereço MAC do quadro original. Assim, o processo se torna mais ágil e preciso.

Com o rastreamaneto podemos obter o RSSI, que é a informação base para o trabalho proposto. Essa informação é obtida através de uma interface MAC-PHY que se comunica, seja para enviar ou receber dados. Há duas primitivas simples para passar dados da

subcamada MAC para a PHY, transmissão e recepção, que servem para o envio. Já para receber dados são necessárias mais primitivas da MAC para a PHY:

- Pedido de início de transmissão e
- Aviso de final de transmissão.

Da subcamada PHY para a MAC:

- Indicação da ocupação do meio, que é enviada toda vez que o estado, ocupado ou desocupado, muda;
- Indicação de início de recebimento de dados e
- Aviso de final de recebimento.

Para um serviço de localização *Wi-Fi*, faz-se necessário conhecer o funcionamento da recepção de um quadro MAC envolvendo as duas camadas. Isso porque é nesse processo que a PHY passa a MAC a informação da RSSI. Quando se percebe uma alteração na energia do meio, ultrapassando um determinado nível, a PHY passa a MAC uma indicação de que o meio está ocupado. Em seguida, a PHY examina o cabeçalho referente à sua camada no quadro. Caso a conclusão da análise seja que o cabeçalho é válido e o quadro não está corrompido, a PHY envia a MAC à indicação de início de recebimento. Essa indicação é passada juntamente com um vetor de informações dentre as quais está a RSSI percebida pela PHY naquele quadro específico. O processo de recebimento então tem seqüência com a passagem dos dados que chegam da PHY para a MAC, até que, por fim, a primitiva de final de recebimento é enviada.

O RSSI é observado na PHY a cada quadro que chega, e essa informação é sempre transmitida a MAC. Contudo, não há na especificação uma determinação de como a MAC deve usá-la. Dessa forma, pode-se esperar que a informação sobre a RSSI associada a uma rede seja passada à LME, depois de uma requisição de rastreamento, onde o RSSI do último quadro *Beacon* ou *Probe Response* tenha sido recebido dessa mesma rede.

### 2.3.6 Extensões da especificação 802.11

As extensões freqüentemente implementadas são 802.11a, 802.11b, 802.11g e 802.11n (NETWORKS, a). Onde todas as três especifições contam com uma ou mais formas de aumentar a taxa de transmissão de dados utilizando banda larga com espalhamento espectral. A 802.11a utiliza uma faixa de freqüência diferente, acima de 5GHz, com 8 canais entre 5.150MHz e 5.350MHz, e 4 canais entre 5.725MHz e 5.825MHz de acordo com o padrão norte americano, implicando em uma menor interferência. Dentro desta faixa, os canais de transmissão foram especificados de forma a apresentar menos sobreposição. Contudo, há a desvantagem de ser mais facilmente absorvido por obstruções no caminho, o que restringe seu uso a espaços mais abertos ou a células menores.

Já 802.11b além de não apresentar essas desvantagens, aos poucos tomou conta do mercado por utilizar uma tecnologia parecida com a *Code Division Multiple Access* (CDMA) utilizada em celulares o que garantia uma infra-estrutura para produção de *chipsets* que podia ser aproveitada (Pickholtz et al., 1982). No entanto, a taxa de transmissão que se atingia com 802.11b era cerca de 5 vezes menor que aquela que se atingia com 802.11a.

Como evolução foi então desenvolvida a extensão 802.11g, que usava a mesma faixa de RF da 802.11b, 2.4GHz, e ao mesmo tempo implementava a tecnologia de transmissão da 802.11a, com modulação *Orthogonal Frequency-Division Multiplexing* (OFDM). Os dispositivos que implementam 802.11g garantem compatibilidade com 802.11b, o que tem uma conseqüência: se houver na rede algum dispositivo usando 802.11b, todos que usam 802.11g passam a funcionar com a modulação mais lenta.

As taxas de transmissão suportadas por cada uma das extensões são diferentes. A especificação 802.11 original previa duas taxas de transmissão possíveis: de $1 Mbits/s$ e $2 Mbits/s$. Com o esquema de modulação proposto, a extensão 802.11a especifica taxas de transmissão que variam de $6 Mbits/s$ a $54 Mbits/s$. Já a 802.11b, alcança taxas de $5.5 Mbits/s$ e $11 Mbits/s$. A extensão 802.11g possibilita as mesmas taxas de transmissão especificadas em 802.11a e 802.11b, além de algumas outras intermediárias. A versão mais recente é a 802.11n que possibilita taxas de transferências entre 65 Mbps a 300 Mbps, possui método de transmissão *Multiple-Input and Multiple-Output Orthogonal Frequency-Division Multiplexing* (MIMO-OFDM) operando na faixa de freqüência 2,4 GHz e/ou 5 GHz.

Para que um dispositivo perceba o sinal de um determinado PA, mesmo que sua placa seja compatível com todas as extensões, o PA deve transmitir segundo a mesma extensão para a qual o dispositivo está configurado. É possível alterar dinamicamente a configuração do modo de transmissão e recepção da placa, possibilitando o rastreamento de redes em cada uma das extensões, mas isso torna o processo mais dispendioso e praticamente inviável para o emprego em um serviço de localização. Portanto, ao desenvolver um serviço de localização em redes *Wi-Fi* é importante garantir que todos os PAs do ambiente trabalhem com a mesma extensão.

## 2.4 Considerações Finais

Com o crescente interesse no desenvolvimento das aplicações baseadas em contexto, a localização é um dos atributos mais utilizados para compreender o contexto corrente do usuário móvel. Dessa forma, o objetivo principal do uso de diferentes protocolos de rede sem-fio é a utilização da infra-estrutura existente. Além disso, estes recursos provêm uma abordagem natural, onde é utilizado a potência de sinal para localização de dispositivos móveis em ambientes fechados. O Capítulo 3 apresenta as abordagens neste trabalho SMA e RNA.

# Capítulo 3

# Abordagens Utilizadas

Neste capítulo será apresentanda uma breve descrição de SMA e RNA utilizados para localização de usuários em ambientes fechados.

## 3.1 Sistema Multiagente

Com a complexidade cada vez maior dos sistemas de computação, novos modelos de desenvolvimento foram criados . Um dos modelos que podemos citar é o baseado em agentes, que faz uso de IA. Segundo Malone (1988) a visão moderna de IA está relacionada aos agentes racionais, classificados em:

- Resolução Distribuída de Problemas ou *Distributed Problem Solving* - decompõe o problema em módulos através de uma abordagem descendente (*top-down*). Este modelo é usado especificamente para um problema em particular, onde grande parte do raciocínio sobre a solução é inserida pelo próprio projetista;
- Sistema Multiagente (SMA) ou *Multi-Agent Systems* - caracteriza-se pela existência de agentes que interajam de forma autônoma, os quais trabalham juntos para resolver um determinado problema ou objetivo e
- Inteligência Artificial Paralela ou *Parallel Artificial Inteligence* - notabiliza-se por desempenho do que por avanços conceituais, preocupando-se principalmente em desenvolver linguagens e algoritmos de computação paralela.

SMA faz uso da tecnologia de agentes, que vem ao longo dos últimos anos ganhando uma importância cada vez maior em muitos aspectos da computação, principalmente na área de IA. Os conceitos de autonomia, de aplicações capazes de executar tarefas de forma inteligente e independente são amplamente abordados pelo modelo de agentes. Os agentes trabalham juntos para um objetivo conjunto.

### 3.1.1 Definições Importantes

SMA são sistemas constituídos de múltiplos agentes que interagem ou trabalham em conjunto de forma a realizar um determinado conjunto de tarefas ou objetivos. Estes podem ser comuns a todos os agentes ou não.

Os agentes dentro de um SMA podem ser heterogêneos ou homogêneos, colaborativos ou competitivos, entre outros. Sendo que a definição dos tipos de agentes depende da finalidade da aplicação e do ambiente em que os agentes estiverem inseridos.

Os tipos de agentes podem ser reativos e/ou cognitivos. Agentes reativos simples não possuem inteligência ou representação de seu ambiente e interagem utilizando um comportamento de percepção/ação. A inteligência surge conforme a necessidade de interações entre os agentes e o ambiente.

O SMA constituído por agentes cognitivos são geralmente compostos por uma quantidade bem menor de agentes se comparado aos SMA reativos. Os agentes cognitivos são inteligentes e contêm uma representação parcial de seu ambiente e dos outros agentes. Assim, comunicam-se entre si, podendo negociar uma informação ou um serviço e planejar uma ação futura. Esse planejamento de ações é possível, porque em geral os agentes cognitivos são dotados de conhecimento, competência, intenção e crença, o que lhes permite coordenar suas ações visando à resolução de um problema ou o alcance de um objetivo.

Atualmente a pesquisa em SMA está interessada principalmente na coordenação das ações e comportamentos dos agentes, como eles coordenam seu conhecimento, planos, objetivos e crenças com o objetivo de tomar ações ou resolver problemas. Segundo Jennings et al. (1998), algumas razões para o crescimento do interesse em pesquisas com SMA são:

- A capacidade de fornecer robustez e eficiência;
- A capacidade de permitir interoperabilidade entre sistemas legados;
- A capacidade de resolver problemas cujo dado, especialidade ou controle é distribuído.

### 3.1.2 Agentes

Existem varias abordagens de diferentes autores para definição de um conceito único ou padrão para agente. Outros fatores estão relacionados às suas mais diversas aplicações, tornando uma definição precisa cada vez mais complicada.

Pela definição de Luft (1998), um agente é uma entidade que age por ou no lugar de outra segundo autoridade por ela outorgada. Geralmente realizando tarefas determinadas, sendo especialistas no que fazem, possuem capacidades que outras pessoas não têm, tem acesso a informações relevantes para sua tarefa e as realizam a um custo bem menor do que se pessoas comuns tentassem realizar. Para agentes artificiais as características acima citadas são também aplicáveis apesar de não haver uma definição universalmente aceita. Contudo, um consenso é que os agentes são entidades que se adaptam a um meio, reagem a ele e provocam mudanças neste meio.

Segundo a FIPA, agentes podem ser definidos como: "uma entidade que reside em um ambiente onde interpreta dados através de sensores que refletem eventos no ambiente e executam ações que produzem efeitos no ambiente. Um agente pode ser software ou hardware puro".

Para Wooldridge & Jennings (1994), existem duas definições gerais, conforme uma noção fraca e noção forte de agentes. Na noção fraca de agentes, eles são sistemas computacionais, sendo hardware ou software, com certas propriedades, tais como autonomia, habilidade social, reatividade e pró-atividade. Na noção forte de agentes, mais adotada

pelos pesquisadores ligados a área de IA, possui, além das propriedades acima citadas, noções relacionadas ao comportamento humano, tais como o conhecimento, a crença, a intenção e a obrigação.

Comparações entre objetos e agentes são inevitáveis, porém, apesar de existir semelhanças, as diferenças são extremamente evidentes. De acordo com Weiss (1999), o agentes tem uma noção mais forte de autonomia em relação aos objetos, são capazes de um comportamento flexível e pela característica de um SMA ser inerentemente *multi-thread*.

Em suma, agentes seriam entidades de *software* autônomas que atuam em determinado ambientes de forma a interagir com este e com outros agentes. Estes podem produzir ações e percepções sem requerer intervenções humanas constantes. Em abordagem mais ligada a IA, um agente ideal teria que ser capaz de funcionar continuamente e adquirir experiências e conhecimentos acerca do ambiente que está interagindo. Ou seja, ser capaz de "aprender" e tomar decisões a partir de situações diferentes.

Algumas propriedades são essenciais para uma melhor caracterização e compreensão de agente. Uma delas é a autonomia que o agente inteligente deve ter, para tomar decisões e ações importantes para a conclusão de uma tarefa ou objetivo sem a necessidade da interferência do ser humano ou qualquer outra entidade. Ou seja, ser capaz de agir independentemente com seu ambiente através de seus próprios "sensores" ou com as suas próprias percepções com o objetivo de realizar alguma tarefa, seja ela externa ou gerada por ele. Outra característica é a de operarem sem a intervenção humana e tem algum tipo de controle sobre suas ações e seu estado interno. Essa autonomia está ligada à pró-atividade, que é a capacidade do agente de tomar decisões por iniciativa própria. Em conjunto a essa característica está a reatividade, que é a capacidade de reagir às alterações no ambiente, ou seja, percebe o meio e responde de modo oportuno.

Os agentes devem ter também robustez, fazendo com que sejam capazes de tomar decisões baseando-se em informações incompletas ou escassas, lidar com erros e ter uma capacidade de adaptação ou aprendizagem através da experiência. Para alcançar seus objetivos, os agentes devem ter a habilidade de comunicar-se. Esta comunicaçao pode ser via acesso a; repositório de dados, seja via outro agente, ou, o próprio ambiente. O fundamental é uma constante troca de informações.

Outra especificidade do agente é o raciocínio, que talvez seja o aspecto mais importante que distingue um agente inteligente dos outros agentes. Dizer que um agente tem raciocínio implica que ele tem a capacidade de analisar e inferir baseando-se no seu conhecimento e nas suas experiências. Esse raciocínio pode ser:

- Baseado em regras - onde eles usam um conjunto de condições prévias para avaliar as condições no ambiente externo; e
- Baseado em conhecimento - onde eles têm à disposição grandes conjuntos de dados sobre cenários anteriores e ações resultantes, dos quais eles deduzem seus movimentos futuros.

Para Weiss (1999), características como flexibilidade de ações, reatividade, pró-atividade e sociabilidade, são suficientes para classificar um agente como inteligente. Assim, um agente deve ter a capacidade de cooperar: agentes inteligentes podem, e devem trabalhar juntos para benefício mútuo na execução de uma tarefa complexa e um comportamento adaptativo, no qual possa examinar o meio externo e adaptar suas ações para aumentar

a probabilidade de ser bem sucedido em suas metas. Entretanto, um agente não precisa ter todas as características listadas de forma concomitante. Existem agentes que possuem algumas e outros que possuem todas. Atualmente existe pouca concordância sobre a importância dessas propriedades e se é necessária sua obrigatoriedade para a caracterização de um agente. O consenso é que essas características tornam em muito um agente diferente de simples programas e objetos.

### 3.1.3 Categoria de Agentes

Segundo Ferber (1999), os agentes podem ser classificados de acordo com vários aspectos:

- Agentes móveis - agentes que tem a mobilidade como principal característica. Ou seja, tem capacidade de mover-se, seja por uma rede interna local (*Intranet*) ou até mesmo pela *Web*, transportando-se por plataformas levando dados e códigos. Fazem uso da heterogeneidade das redes. Esse tipo de agente auxilia em tomadas de decisões baseadas em grandes quantidades de informação.
- Agentes estacionários - opostos aos móveis, são fixos em um mesmo ambiente e ou plataforma.
- Agentes competitivos - agentes que competem entre si para a realização de seus objetivos ou tarefas, nesse caso não há colaboração entre os agentes.
- Agentes coordenados ou colaborativos - tem a finalidade de alcançar um objetivo maior, realizam tarefas especificas, porém coordenando-as entre si de forma que suas atividades se complementem.
- Agentes reativos - agentes que reagem a estímulos sem ter memória do que já foi realizado no passado e nem previsão da ação a ser tomada no futuro. Não tem representação do seu ambiente ou de outros agentes e são incapazes de prever e antecipar ações.
- Agentes cognitivos - podem raciocinar sobre as ações tomadas no passado e planejar ações a serem tomadas no futuro. Assim, um agente cognitivo é capaz de resolver problemas por ele mesmo. Esse tipo de agente tem objetivos e planos explícitos, os quais permitem atingir seu objetivo final. Sua representação interna e seus mecanismos de inferência o permitem atuar independentemente dos outros agentes e lhe dão uma grande flexibilidade na forma de expressão de seu comportamento. Além disso, devido a sua capacidade de raciocínio baseado nas representações do mundo, são capazes de memorizar situações, analisá-las e prever possíveis reações para suas ações.

### 3.1.4 Comunicação entre Agentes

Em um SMA, faz-se necessário que a comunicação seja disciplinada para que os objetivos sejam alcançados efetiva e eficientemente, necessitando assim uma linguagem que possa ser entendida pelos agentes presentes no ambiente. O principal objetivo da comunicação é à partilha do conhecimento com os outros agentes e a coordenação de atividades

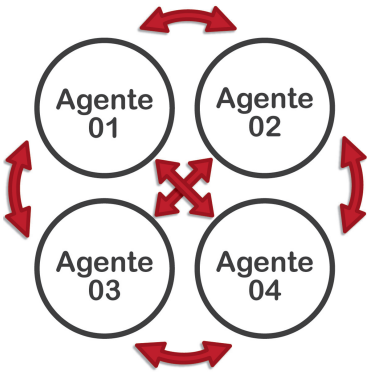



Figura 3.1: Comunicação Direta entre Agentes (adaptada de (FIPA)).

entre eles. Assim, deve permitir que agentes troquem informações entre si e coordenem suas próprias atividades resultando sempre em um sistema coerente.

Há diversas maneiras de trocar informações entre agentes. Para Chauhan & Baker. (1998), agentes podem trocar mensagens diretamente, comunicação direta, ou podem trocar informações através de um agente "facilitador", especial em sistema "federado" (comunicação assistida), ou podem também utilizar a difusão de mensagens (*broadcast*) e até utilizar o modelo de comunicação através de *blackboard* ou quadro-negro.

Na comunicação direta, ou comunicação via troca de mensagens direta, cada agente comunica-se diretamente com qualquer outro agente sem qualquer intermediário. Conforme apresentado na Figura 3.1, há um estabelecimento de uma ligação direta (ponto-a-ponto) entres os agentes através de um conjunto de protocolos que garante a chegada de mensagens com segurança. Nesse tipo de comunicação faz-se necessário que cada agente envolvido tenha conhecimento da existência dos outros agentes e de como endereçar as mensagens.

A vantagem principal para utilização deste tipo de comunicação entre agentes é o fato de não existir um agente coordenador da comunicação. A utilização de agentes coordenadores pode levar a um "gargalo" ou até ao bloqueio do sistema caso haja, por exemplo, um grande número de troca de mensagens. Porém, há desvantagens como; o custo da comunicação que se torna grande, principalmente quando há um grande número de agentes no sistema, e a própria implementação que se torna muito complexa em comparação às outras formas de comunicação.

Segundo Chauhan & Baker. (1998), a comunicação por sistemas federados ou comunicação assistida, existe quando os agentes utilizam algum sistema ou agente especial para coordenar suas atividades. Tal como apresentado na Figura 3.2, uma estrutura hierárquica de agentes é definida e a troca de mensagens dá-se através de agentes especiais designados, "facilitadores" ou mediadores. Essa alternativa à comunicação direta diminui muito o custo e a complexidade necessária aos agentes individuais na realização da comunicação. Comumente é utilizado quando o número de agentes dentro do sistema é muito grande.

A comunicação por difusão de mensagens ou broadcast geralmente é utilizada em situações onde a mensagem deve ser enviada para todos os agentes do ambiente, ou quando o agente remetente não conhece o agente destinatário ou seu endereço. Desta forma, todos os agentes recebem a mensagem enviada.

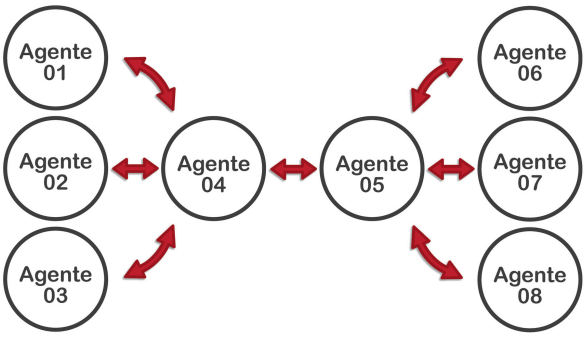



Figura 3.2: Comunicação por sistemas federados ou comunicação assistida (adaptada de (FIPA)).

Chauhan & Baker. (1998) cita que a comunicação por quadro-negro ou *blackboard* é bastante usada na IA como modelo de conhecimento compartilhado. Ou seja, nada mais é que um quadro negro (repositório) onde os agentes escrevem mensagens a outros agentes e obtêm informações sobre o ambiente.

#### 3.1.4.1 KQML

KQML é uma linguagem e um protocolo de comunicação de alto nível para troca de mensagens entre agentes independente de conteúdo e da ontologia aplicável. KQML foi desenvolvida pelo *Knowledge Sharing Effort* (KSE), para servir ao mesmo tempo como um formato de mensagem e um protocolo de gerenciamento de mensagens. Não existe muita preocupação com o conteúdo da mensagem, mas sim com a especificação da informação necessária à compreensão do conteúdo.

Segundo Gudwin (2009), o significado de performativas reservadas e padrões no KQML é pouco claro gerando dificuldades sobre KQML:

- Ambigüidade e termos vagos;
- Performativas com nomes; e
- Falta de performativas.

Pelos problemas citados alguns autores afirmam que KQML será substituído por FIPA-ACL. A Figura 3.3, ilustra uma estrutura básica de uma mensagem no formato KQML.

#### 3.1.4.2 FIPA-ACL

A FIPA-ACL é uma linguagem baseada em ações de fala. Com sintaxe bastante semelhante ao KQML. Contudo, o conjunto de performativas (atos comunicativos) é diferente. Sua especificação consiste de um conjunto de tipos de mensagens e descrições dos efeitos da mensagem sobre os agentes que a enviam e sobre o que a recebem.

Essa linguagem possui uma semântica definida precisamente com uma linguagem de descrição de semântica. Segundo Meneses (2001), KQML é muito criticada por usar o

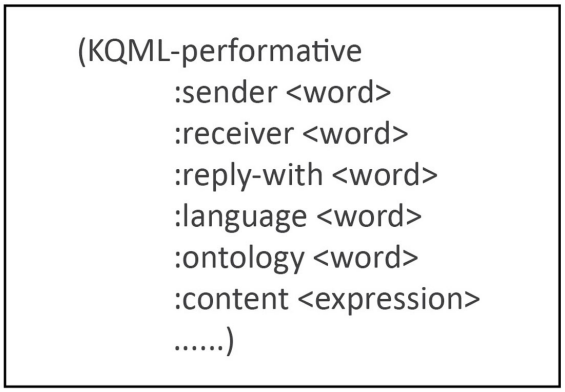



Figura 3.3: Formato básico de mensagem KQML (adaptada de (Weiss, 1999)).

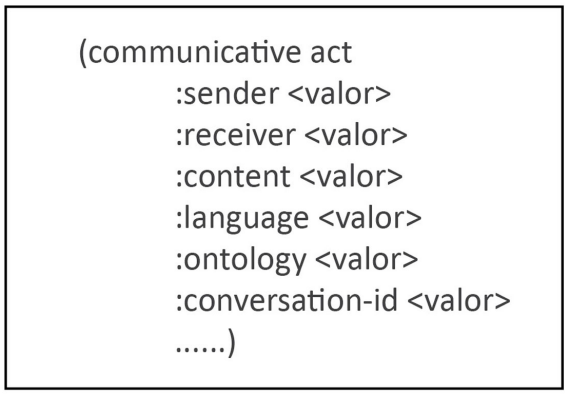



Figura 3.4: Formato básico de mensagem (adaptada de (FIPA)).

termo performativo para se referir às primitivas de comunicação. Já na FIPA-ACL, essas primitivas são chamadas de ações ou atos comunicativos (*communicative acts*). A Figura 3.4 exemplifica uma estrutura de uma mensagem em , com as partes que compõem uma mensagem básica.

### 3.1.5 Coordenação entre Agentes

Segundo Weiss (1999), a coordenação é uma característica fundamental para um sistema de agentes que executam alguma atividade em um ambiente compartilhado. A coordenação está muito relacionada com o compartilhamento de conhecimento entre os agentes, sendo, seu principal objetivo, tornar as ações individuais de cada agente coordenadas para se atingir o objetivo final do SMA. Ainda há uma preocupação com a coerência de modo a se discutir como o SMA se comporta de forma global enquanto está resolvendo o problema. O principal motivo para uma preocupação maior com a coordenação entre agentes é o fato de que um só agente, dentro de um SMA, não terá informação ou ca-

pacidade suficiente para resolver muitos dos problemas. Isso porque muitos dos objetivos não podem ser atingidos por agentes agindo isoladamente.

Dessa forma, a coordenação seria a capacidade dos agentes trabalharem em conjunto e combinar seus objetivos de forma a concluírem o objetivo final do sistema. Muitas vezes para uma cooperação ser bem sucedida, cada agente deve manter um "modelo" dos outros agentes e também desenvolver um modelo de interações futuras ou possíveis. Ela pode ser dividida em cooperação e negociação.

Segundo Weiss (1999), a negociação é a coordenação entre agentes antagônicos ou egoístas, *self-interested*. Assim, a negociação está relacionada à coordenação de agentes competitivos. Na coordenação competitiva são usados protocolos de negociação para determinar as regras de negociação e são definidos os conjuntos de atributos sobre os quais se pretende chegar a um acordo.

Segundo Weiss (1999), a cooperação seria a coordenação entre agentes não antagônicos. Por isso, uma coordenação com agentes que não possuem objetivos antagônicos ou conflitantes é chamada de cooperação. Desta forma, os agentes se auxiliam mutuamente para resolver problemas, os quais provoque custos individuais. O papel da coordenação cooperativa é manter um conjunto de agentes para realizar suas tarefas como um "trabalho de equipe", visando sempre o objetivo final do sistema.

### 3.1.6 Ambiente Multiagente

Segundo Weiss (1999), os ambientes de SMA possuem as seguintes caractéristicas:

- Provém uma infra-estrutura especificando protocolos de comunicação e interação entre os agentes;
- São geralmente abertos e não possuem apenas um projetista, sendo descentralizados;
- Contêm agentes que são autônomos e distribuídos, atuando de forma competitiva ou cooperativa.

Outras propriedades e valores do ambiente são apresentados na Tabela 3.1. O ambiente de desenvolvimento SMA podem ser divididos em plataformas de agentes móveis e plataformas de agentes não móveis.

As plataformas para agentes móveis fornecem todo um *framework* e ferramentas que ajudam na criação de agentes móveis juntamente com uma estrutura que facilita a sua implementação. As plataformas para agentes não móveis são exatamente as que fornecem ferramentas que ajudam na criação de agentes inteligentes.

Muitas das plataformas móveis ou não móveis são específicas, ou seja, os agentes criados por elas são específicos a uma determinada área. Contudo, não significa dizer que uma plataforma classificada como plataforma para agentes não-móveis não possa criar agentes móveis. Ao contrário, muitos dos agentes criados por esses ambientes possuem até certa mobilidade, contudo muito mais simples e com funcionalidades mais primitivas comparados com os agentes criados por plataformas especialmente dedicadas para agentes móveis.

Tabela 3.1: Tabela de Características de um Ambiente Multiagente (adaptada de (Weiss, 1999)).

| PROPRIEDADE | VALORES |
|---|---|
| Autonomia de Projeto | Plataforma/Protocolo de Interação/ Linguagem/Arquitetura Interna |
| Infra-estrutura de Comunicação | Memória Compartilhada (*Blackboard*) ou Baseada em mensagens, Orientada à conexão ou Não-conectada à conexão (*e-mail*), Ponto-a-ponto, *Multicast* ou *Broadcast*, Síncrono ou Assíncrono |
| Diretório de Serviço (*Directory Service*) | *White pages*, *Yellow Pages* |
| Protocolo de Mensagens | KQML, FIPA-ACL, HTTP ou HTML, OLE, CORBA |
| Serviço de Mediação | Baseado em Ontologia ou Transações |
| Serviço de Segurança | *Timestamps*/Autenticação |
| Operações de Suporte | Armazenamento/Redundância/Restauração/*Accounting* |

### 3.1.7 Padrão FIPA

A FIPA, é uma fundação internacional com o objetivo de definir padrões para a interoperabilidade de agentes heterogêneos e interativos em sistemas baseados em agentes. Fundada em 1996 em Genebra, a FIPA tem incumbência de facilitar a interligação de agentes e SMA entre múltiplos fornecedores de ambientes. Oficialmente sua missão é a de promover tecnologia e especificações de interoperabilidade que facilitem a comunicação entre sistemas de agentes inteligentes no contexto comercial e industrial moderno, em outras palavras, interoperabilidade entre agentes autônomos.

A FIPA se apóia basicamente em duas concepções principais. A primeira é que o tempo para alcançar um consenso e para completar a padronização não deve ser longo para que não impeça o progresso nessa área de pesquisa. A segunda é que apenas os comportamentos externos dos componentes é que podem ser especificados, deixando detalhes de implementação e arquiteturas internas para os desenvolvedores de agentes.

Algumas das linguagens de desenvolvimento de agentes que estão relacionadas com a FIPA são: *Agent Development Kit*, FIPA-OS, *Lightweight Extensible Agent Plataform3* (LEAP), ZEUS e o próprio JADE *framework*.

## 3.2 JADE

*Java Agent DEvelopment framework* (JADE), é um ambiente para desenvolvimento de aplicações baseada em agentes conforme as especificações da FIPA para interoperabilidade entre SMA, o qual é totalmente implementado em Java. Desenvolvido e suportado pela Universidade de Parma na Itália sendo *open source* através da licença *Lesser General Public License* (LGPL).

Segundo Bellifemine et al. (2007), o principal objetivo do JADE é simplificar e facilitar o desenvolvimento de SMA garantindo um padrão de interoperabilidade através de um abrangente conjunto de agentes de serviços de sistema. O JADE possui definido um protocolo de comunicação entre os agentes, de acordo com as especificações da FIPA, além de outros serviço tais como: de nomes (*naming service*) e páginas amarelas (*yellow-page*

*service*), transporte de mensagens, serviços de codificação e decodificação de mensagens e uma biblioteca de protocolos de interação (padrão FIPA).

A comunicação entre agentes é feita via troca de mensagens. Além disso, o JADE lida com todos os aspectos que não fazem parte do agente em si e que são independentes das aplicações, tais como transporte de mensagens, codificação e interpretação de mensagens e ciclo de vida dos agentes. Podendo ser considerado como um "*middleware*" de agentes que implementa um *framework* de desenvolvimento e uma plataforma de agentes. Ou seja, uma plataforma de agentes em complacência com a FIPA e um pacote de bibliotecas para o desenvolvimento de agentes em Java.

De acordo com Bellifemine et al. (2007), o JADE foi escrito em Java devido a características particulares da linguagem como a programação Orientada a Objeto em ambientes distribuídos heterogêneos. Foram desenvolvidos tanto pacotes Java com funcionalidades prontas pra uso, quanto interfaces abstratas para se adaptar de acordo com a funcionalidade da aplicação de agentes.

### 3.2.1 Características

O JADE oferece para a programação de SMA diversas características, algumas delas são:

- Plataforma distribuída de agentes - pode ser dividida em vários *hosts* ou máquinas, desde que eles possam ser conectados via *Remote Method Invocation* (RMI). Apenas uma aplicação Java e uma *Java Virtual Machine* (JVM), é executada em cada *host*. Os agentes são implementados como *threads* Java e inseridos dentro de repositórios de agentes chamados de *containeres* (*Agent Containers*) que provêm todo o suporte para a execução do agente;
- *Graphical User Interface* (GUI) – Interface visual que gerencia vários agentes e *containeres* de agentes inclusive remotamente;
- Ferramentas de *Debugging* - ferramentas que ajudam o desenvolvimento e depuração de aplicações multiagentes baseados em JADE;
- Suporte a execução de múltiplas, paralelas e concorrentes atividades de agentes - através dos modelos de comportamento (*behaviours*);
- Ambiente de agentes complacente a FIPA – No qual incluem o sistema gerenciador de agentes (*Agent Management System* (AMS)), o diretório facilitador (*Directory Facilitator* (DF)) e o canal de comunicação dos agentes (*Agent Communication Channel* (ACC)). Todos esses três componentes são automaticamente carregados quando o ambiente é iniciado;
- Transporte de mensagens no formato FIPA-ACL dentro da mesma plataforma de agentes;
- Biblioteca de protocolos FIPA - Para interação entre agentes JADE dispõe uma biblioteca de protocolos prontos para ser usado;
- Automação de registros - Registro e cancelamento automático de agentes com o AMS fazendo com que o desenvolvedor se abstraia disso. Serviços de nomes (*Naming Service*) em conformidade aos padrões FIPA. Na inicialização dos agentes,

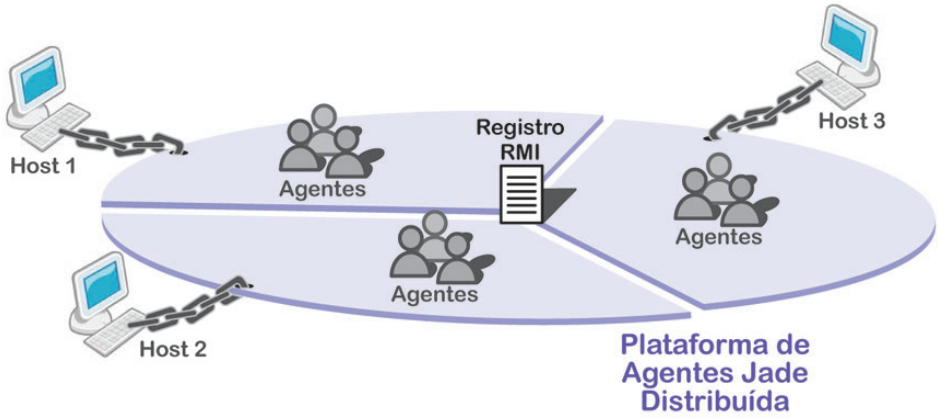



Figura 3.5: Plataforma de Agentes JADE (adaptado de Bellifemine et al. (2007)).

estes obtêm seus *Globally Unique Identifier* (GUID) da plataforma, os quais são identificadores únicos em todo o ambiente;

- Integração - Mecanismo que permite aplicações externas, os quais carregarem agentes autônomos JADE.

### 3.2.2 Arquitetura Interna

A arquitetura da plataforma JADE é baseada na coexistência de várias JVM, podendo ser distribuída por diversas máquinas independentes de sistema operacional. A Figura 3.5 apresenta uma visão distribuída da plataforma de agentes JADE dividida em 3 *hosts*. Em cada host temos uma JVM com o seu *Java Run-time Enviroment* (JRE) enfatizando o conceito de independência de plataforma. Cada JVM tem um container de agentes que fornece um ambiente completo para execução destes agentes, além de permitir que vários agentes possam ser executados concorrentemente no mesmo processador (*host*). Ou seja, durante a execução deve existir uma JVM por processador, sendo possíveis vários agentes por JVM. A comunicação entre as JVM é feita através de invocação remota de métodos RMI de Java.

O *main-container*, localizado no *host* 1 da Figura 3.5, é o *container* onde se encontra o AMS, o DF e o registro RMI. Esse registro RMI nada mais é que um servidor de nomes que o Java usa para registrar e recuperar referências a objetos através do nome. Sendo assim o meio que JADE usa em Java para manter as referências aos outros containeres de agentes que se conectam a plataforma através do registro RMI.

Os outros containeres de agentes conectam-se ao *main-container* fazendo com que o desenvolvedor fique abstraído da separação física dos *hosts*, ou das diferenças de plataformas, que os *host* possam ter. Essa abstração é ilustrada na Figura 3.5 na parte central na qual a plataforma JADE integra todos os três *hosts* atuando como um elo e provendo um completo ambiente de execução para qualquer conjunto de agentes JADE.

### 3.2.3 Agentes JADE

A FIPA não especifica as estruturas internas dos agentes. Tal fato foi uma escolha explícita em conformidade com a opinião de que a interoperabilidade pode ser garantida apenas especificando os comportamentos externos dos agentes. Dessa maneira, cada agente é autônomo possuindo um processo de identificaçao independente com comunicaçao com outros agentes, seja ela por colaboração ou por competição, para executar totalmente seus objetivos (Bellifemine et al., 2007). Assim, o *framework* JADE é absolutamente neutro no que diz respeito à definição de um agente, não limita ou estabelece que tipo de agente pode ser construído pela plataforma.

Um agente em JADE funciona como uma *thread* que pode executar múltiplas tarefas ou comportamentos e conversações simultâneas. O agente JADE é implementado como uma classe Java chamada *Agent* que está dentro do pacote *JADE.core*. Essa classe *Agent* atua como uma super classe para a criação de agentes de *software*, provendo métodos para executar tarefas básicas de agentes, tais como:

- Passagens de mensagens usando objetos ACLMessage, sejam direta ou *multicast*;
- Suporte completo ao ciclo de vida dos agentes, incluindo iniciar ou carregar, suspender e "matar" (*killing*) um agente;
- Escalonamento e execução de múltiplas atividades concorrentes;
- Interação simplificada com sistemas de agentes FIPA para a automação de tarefas comuns de agentes (e.g. registro no DF, entre outros).

Devido a sua própria arquitetura, os agentes podem migrar e clonar-se entre os containeres, provendo suporte ao desenvolvimento de agentes móveis. O JADE também disponibiliza uma biblioteca (*jade.domain.mobility*), que contém a definição de ontologias para a comunicaçao de agentes móveis, com vocabulário, uma lista de símbolos usados e todas as classes Java que implementam essas ontologias.

Para o desenvolvedor, um agente JADE é simplesmente uma instância da classe *Agent*, no qual os programadores deverão escrever seus próprios agentes como subclasses de *Agent*, adicionando comportamentos específicos de acordo com a necessidade e objetivo da aplicação. Esse agente contém um conjunto básico de métodos, os quais utiliza as capacidades herdadas que a classe *Agent* dispõe, tais como mecanismos básicos de interação com a plataforma de agentes (registro, configuração, gerenciamento remoto, entre outros).

A Figura 3.6 descreve a arquitetura interna de um agente genérico em JADE. Na parte superior têm-se os comportamentos ativos do agente que representam as ações/intenções de cada agente. O modelo computacional de um agente em JADE é multitarefa, onde tarefas (ou comportamentos) são executadas concorrentemente. Cada funcionalidade ou serviço provido por um agente deve ser implementado como um ou mais comportamentos. Esses comportamentos podem ser vários, uma vez que JADE permite uma variedade de comportamentos em um mesmo agente. Na parte inferior esquerda da Figura 3.6 há uma fila privativa de mensagens ACL. Todo agente JADE tem essa fila onde decide quando ler as mensagens recebidas e quais mensagens ler.

No centro, encontra-se o escalonador de comportamentos e o gerenciador do ciclo de vida. O primeiro é responsável por organizar a ordem de execução dos comportamentos, podendo seguir alguma ordem de precedência ou não. O gerenciador do ciclo de vida e

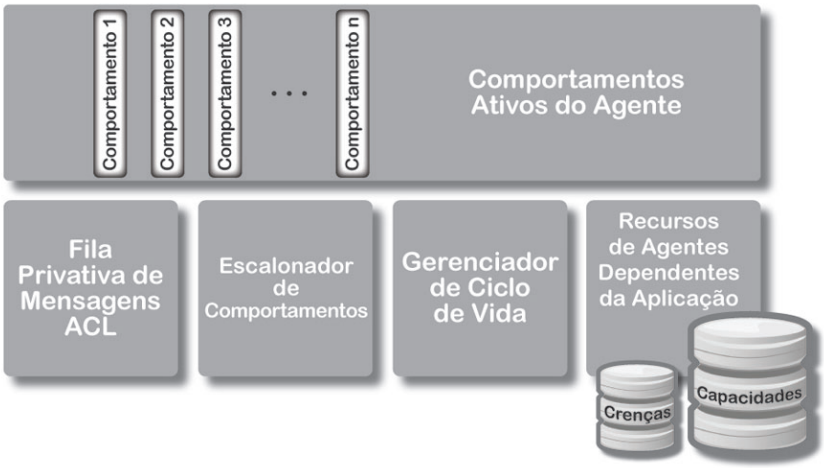



Figura 3.6: Arquitetura Básica JADE (adaptado de (Bellifemine et al., 2007)).

o controlador do estado atual do agente. Como uma característica importante do modelo de agentes JADE é autônomo, cada agente possui a autonomia implementada pela possibilidade de controlar completamente sua *thread* de execução. O gerenciador do ciclo de vida é o meio que os agentes utilizam para determinar seu estado atual (e.g. ativo, suspenso, entre outros.).

No lado direito da Figura 3.6, existem os recursos dependentes da aplicação que os agentes acessam. Nesse local são armazenados as crenças e capacidades que o agente adquiriu em sua execução.

### 3.2.4 Ciclo de Vida

Os agentes JADE podem estar em um dos vários estados de acordo com ciclo de vida das especificações FIPA (*Agent Platform Life Cycle*). A Figura 3.7, representa o ciclo de vida de um agente definido pela FIPA e seus estados possíveis. Esses estados são representados em JADE como constantes estáticas da classe *Agent*, a saber:

- *AP_INIATED* - o objeto da classe *Agent* foi instanciado, mas ainda não se registrou no AMS, não tem nome ou endereço e não pode comunicar com outros agentes;
- *AP_ACTIVE* - o objeto da classe *Agent* está registrado no AMS, tem nome formal e endereço e tem acesso a todas as funcionalidades do JADE;
- *AP_SUSPENDED* - o objeto da classe *Agent* está no momento interrompido. Sua *thread* interna está suspensa e nenhum comportamento está sendo executado;
- *AP_WAITING* - o objeto da classe *Agent* está bloqueado, esperando por alguma coisa. Sua *thread* interna está "dormindo" (*sleeping*) sob um monitor Java e irá acordar quando alguma condição ocorrer (geralmente quando uma mensagem chega);
- *AP_DELETED* - o agente está definitivamente "deletado" ou encerrado. Sua *thread* interna terminou, sua execução e o agente não está mais registrado no AMS;

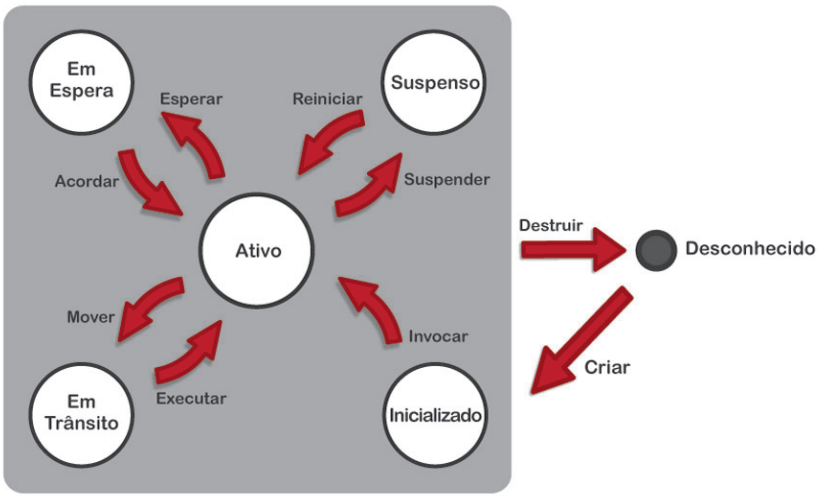



Figura 3.7: Ciclo de Vida no JADE (adaptado de (FIPA)).

- *AP_ TRANSIT* - um agente móvel entra nesse estado enquanto tiver migrando para uma nova localização. O sistema continua a armazenar mensagens em um *buffer* que serão enviadas para essa nova localização;
- *AP_ COPY* - esse estado é usado internamente pelo JADE para agentes que foram clonados;
- *AP_ GONE* - esse estado é usado internamente pelo JADE quando um agente móvel migrou para outra localização e se encontra em um estado estável.

A troca de estados é feita por meio de métodos como podemos visualizar na Figura 3.7 (e.g. reiniciar, esperar, suspender, entre outros). Existem métodos públicos disponibilizados pela classe *Agent* de JADE que fazem as transições entre esses estados acima citados.

Os agentes podem executar seus comportamentos/*behaviours* apenas quando estiver no estado ativo. Em qualquer dos comportamentos de um agente quando é chamado o método *doWait ()*, o agente como um todo e todas suas atividades serão bloqueadas, não só o comportamento que chamou o método. Ao invés disso, o método *block()* da classe *Behaviour* evita essa bloqueio total, uma vez que permite a suspensão de apenas um comportamento, e além de se desbloquear caso uma mensagem tenha sido recebida.

### 3.2.5 Comportamentos

Os comportamentos ou as ações cumpridas por um agente dentro de um SMA são fundamentais para que o objetivo final da aplicação seja alcançado. O JADE contém comportamentos específicos para a maioria das tarefas comuns na programação de agentes, tais como envio e recebimento de mensagens e até construção de tarefas mais complexas.

A Figura 3.8 representa os comportamentos que o JADE possui, sendo uma estrutura de comportamentos escalonada, interno à super classe *Agent*, que gerencia automaticamente a execução dos comportamentos (*behaviours*). Para programação concorrente um

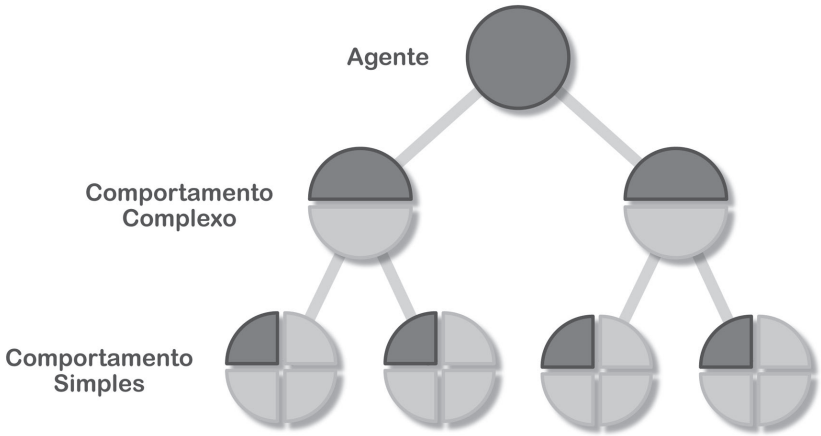



Figura 3.8: Comportamentos em JADE (adaptada de (Bellifemine et al., 2007)).

agente é um objeto ativo com uma *thread* de controle interna. JADE usa o modelo de uma *thread* por agente ao invés de uma *thread* por tarefa ou conversação, com o objetivo de manter um número pequeno de *threads* necessárias para executar a plataforma de agentes. Adotando a abordagem de uma nova *thread* para cada comportamento, o desempenho do sistema poderia ser comprometido.

Seguindo (Bellifemine et al., 2007), o escalonamento dos comportamentos são feitos por um escalonador, implementado na super classe *Agent* e abstraído do programador, que realiza a política de *round-robin* não preemptivo, em todos os comportamentos disponíveis na fila. No escalonamento *round-robin* cada comportamento adicionado ao agente é colocado em uma fila. Quando esse comportamento passa para o estado de execução, ou seja, o processo passa para a *Central Processing Unit* (CPU) , existe um tempo limite para a sua utilização.

Quando esse tempo denominado *time-slice* ou *quantum* expira, sem que antes o processador seja liberado pelo processo, este volta ao estado de pronto, dando a vez para outro processo. Assim, permite a execução da classe de um comportamento completo. Ou seja, caso a tarefa ou comportamento cesse sem o controle ter sido completado, a tarefa será re-escalonada no próximo ciclo. Essa tarefa poderá não ser re-escalonada caso esteja bloqueada (um *behaviour* pode se bloquear no instante em que espera mensagens para evitar desperdício de tempo de CPU e evitar uma espera indevida).

Um comportamento é executado durante um determinado tempo (*quantum*). Quando o tempo se esgota, o comportamento pode voltar para o fim da fila, caso não tenha acabado de executar sua tarefa. O fato de ser não preemptivo significa dizer que não existe prioridade entre os comportamentos, ou seja, cada comportamento adicionado deve ir para o fim da fila e todos têm a mesma fatia de tempo.

### 3.2.6 Troca de Mensagens

As trocas de mensagens realizadas no JADE são feita por métodos próprios e com o uso de instâncias da classe *ACLMessage*. Esta classe possui um conjunto de atributos que estão em conformidade com as especificações da FIPA, implementando a linguagem

FIPA-ACL. Dessa maneira, um agente que envie uma mensagem instancia um objeto da classe *ACLMessage*, preenchendo as informações necessárias e chamando o método *send()* da classe *Agent*.

Para receber mensagens, o método *receive()* ou *blockingReceive()* da classe *Agent* deve ser chamado. Outra meio de enviar ou receber mensagens no JADE é através do uso das classes de comportamentos *SenderBehaviour* e *ReceiveBehaviour*. Fato que torna possível que as trocas de mensagens possam ser escalonadas como atividades independentes no comportamento dos agentes.

A classe *ACLMessage* implementa uma mensagem na linguagem FIPA-ACL complacente às especificações da FIPA. Todos seus atributos podem ser acessados via métodos *set* e *get*. A *ACLMessage* ainda define um conjunto de constantes (*ACCEPT_PROPOSAL, AGREE, CANCEL, CFP, CONFIRM*, entre outros), os quais são usados para se referir às performativas da FIPA. Essas constantes são referidas no construtor da classe com o objetivo de definir o tipo de mensagem.

Na referência FIPA, todas trocas de mensagens entre agentes devem se basear em strings ou textos. Esse padrão é adotado por JADE, no qual os métodos *getContent* e *setContent* trabalham com *strings*. Porém JADE permite que não só *strings* sejam transmitidas por mensagens, mas também outros objetos Java. No método *setContentObject* da classe *ACLMessage* e permitindo definir como conteúdo de uma mensagem um objeto Java do tipo *java.io.Serializable*. Já o método *getContentObject* retorna o conteúdo definido pelo método *setContentObject*.

### 3.2.7 Interoperabilidade

Para possibilitar a comunicação entre diferentes ambientes, JADE utiliza-se do protocolo *Internet Inter-ORB Protocoll* (IIOP). Já na comunicação de agentes na mesma plataforma JADE utiliza outros meios (RMI ou via eventos). Assim, JADE possui um comportamento variado em relação à forma com que realiza sua comunicação, aonde o mecanismo é selecionado de acordo com a situação do ambiente. Tal fato tem o único objetivo de atingir o menor custo possível de passagem de mensagens. No caso, quando um agente JADE envia uma mensagem, as seguintes situações são possíveis:

- O agente receptor reside no mesmo container de agentes, então o objeto Java representante da mensagem ACL é passado para o receptor via eventos, sem invocações remotas;
- O agente receptor reside na mesma plataforma JADE, mas em containeres diferentes, a mensagem ACL é enviada usando Java RMI;
- O agente receptor reside em uma diferente plataforma de agentes, o protocolo IIOP é usado de acordo com o padrão FIPA.

O meio mais eficiente é escolhido de acordo com a localização do agente receptor da mensagem. Na parte inferior da Figura 3.9 existem as formas possíveis de comunicação. O cachê local, localizado no ACC, armazena as referências dos objetos dos outros containeres. Essas referências são adicionadas ao cachê sempre que uma mensagem é enviada. O modelo de plataforma padrão especificado pela FIPA é apresentado na Figura 3.10, conforme na Seção 3.2.1


Figura 3.9: Interoperabilidade em JADE (adaptado de Bellifemine et al. (2007)).

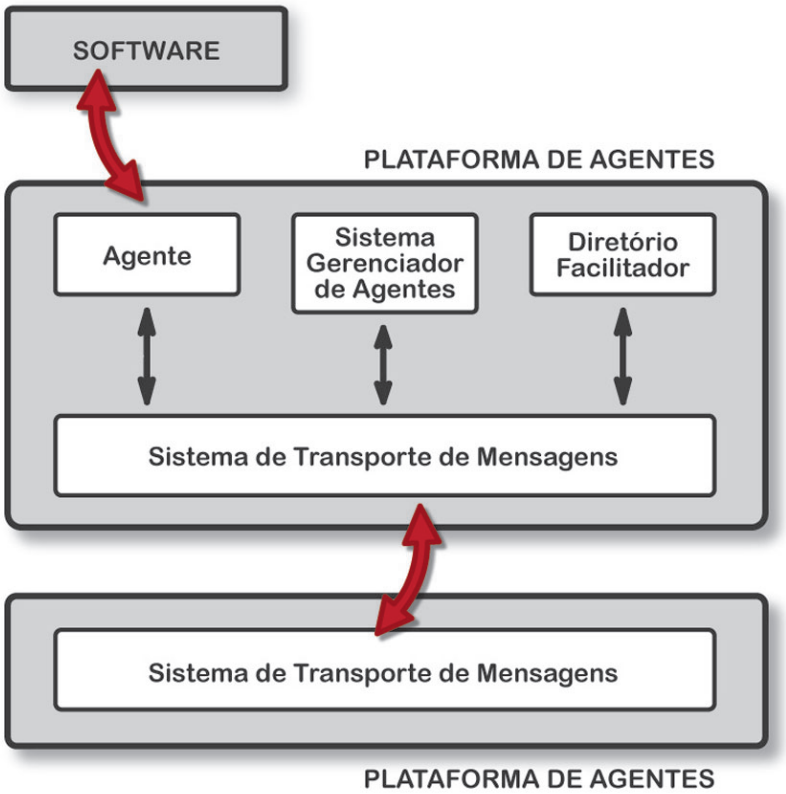



Figura 3.10: Modelo padrão de plataforma definido pela FIPA (adaptado de (FIPA)).

### 3.2.8 Considerações Finais

Conforme apresentado, SMA contém atributos adequados para o desenvolvimento de aplicações que dependam da localização de dispositivos móveis. Os protótipo apresentado no Capítulo 5 faz uso das características como percepção, coleta, representação e interpretação. Pelo exposto, o desenvolvimento do framework proposto neste trabalho é baseado em SMA.

Com o crescimento do desenvolvimento das aplicações baseadas em contexto, a localização é uma das bases para aplicações sensíveis ao contexto. Dessa forma, uma possível maneira de estimar a localização é usar RNA. As características desta técnica de IA são apresentadas na Seção 3.3.

## 3.3 Rede Neural Artificial

Os primeiros estudos em IA ocorreram no encontro de *Darthmouth College* em 1956. O primeiro artigo de Shannon & McCarty (1956) apresentou uma abordagem de um paradigma da arquitetura computacional. Em meio a esse novo estudo surgiram duas vertentes: a IA simbólica (Dorffner et al., 1993; Peirce, 1995) e a conexionista (Sun, 1999; Haykin, 2001). Basicamente as duas se diferem por não considerar os mecanismos responsáveis pelo comportamento simbólico, e pela construção de máquinas que imitem as estruturas cerebrais para representar a inteligência conexionista.

As RNAs tomam como base o modelo de funcionamento dos neurônios biológicos, contudo atualmente as RNAs estão muito distantes das redes neurais naturais, mas tendo uma semelhança mínima (Haykin, 2001).

Segundo Haykin (2001), uma RNA é um sistema que pode ser composto por vários neurônios. Estes neurônios podem ser ligados a outros neurônios através de conexões sinápticas, formando camadas de neurônios. Assim tem-se que: (i) os neurônios de entrada, que recebem informações do mundo exterior; (ii) neurônios em camadas intermediárias; e (iii) neurônios de saída, que enviam respostas para o mundo externo.

Em Haykin (2001) um neurônio recebe uma entrada $p$, que pode ser uma matriz que é multiplicada por um fator $w$ (peso). O produto $wp$ se soma a uma entrada polarizada $b$, (escalar), e o resultado é passado como argumento para a função de transferência do neurônio produzindo a saída $a$. Durante o processo de treinamento de uma RNA, os valores de $w$ e de $b$ são reajustados de modo a produzir saídas mais convenientes. A Figura 3.11, apresenta um modelo de RNA.

A idéia da caixa preta visualizada na Figura 3.11, onde o neurônio consiste num conjunto de entradas ($S1 \ldots SN$) multiplicadas por pesos sinápticos ($w1 \ldots wN$. Um somador, que vai somar o produto de cada uma das entradas do neurônio com o peso sináptico respectivo ($\Sigma$), e uma função de ativação (*Non-linear activation function*) que limita a amplitude da saída do neurônio ($Y$). Segundo Battiti et al. (2002) esta função é muito útil para escolhas booleanas, ou seja, cujas respostas sejam sim ou não (Brunato & Battiti, 2005).

A RNA MLP apresentada na Figura 3.12 é muito utilizada em sistemas *fingerprinting*, é a ligação de múltiplos neurônios, quer em associação, série ou em associação paralela, cujo sinal caminha seqüencialmente desde a camada de entrada até a de saída, indo da saída de um neurônio a entrada de outro. As camadas que se situam entre a entrada e

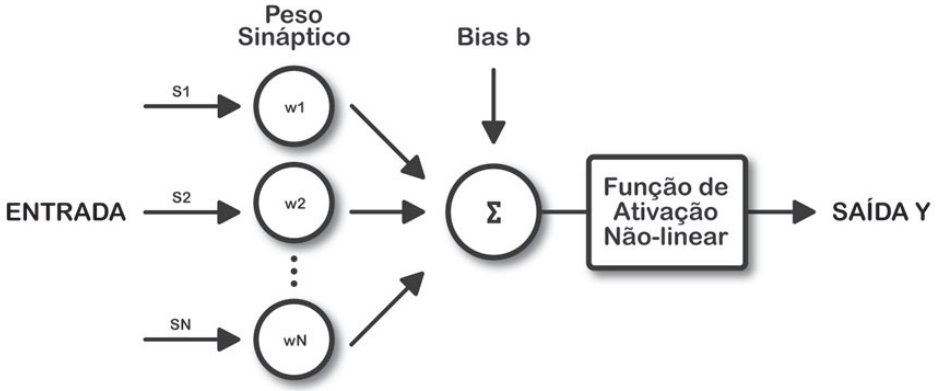



Figura 3.11: Visão da caixa preta constituída por um conjunto de neurônios (adaptada de (Beale & Jackson., 1990)).

a saída denominam-se camadas escondidas. (Saha et al., 2003) e (Battiti et al., 2002) apresentam exemplos de sistemas de localização que usam esta técnica.

### 3.3.1 *Backpropagation*

O modelo de *Backpropagation*, também conhecido como retropropagação, consiste em um modelo de aprendizagem supervisionado de múltiplas camadas, onde o erro obtido na saída é propagado para as camadas mais internas (camadas escondidas) para correção da saída. Esse algoritmo apresenta duas fases apresentadas pelas Figura 3.13 e Figura 3.14.

Segundo Cardon & Muller. o treinamento desse tipo de RNA é bastante extenso e dependendo dos padrões estudados, pode haver a necessidade de uma grande fase de pré-processamento. O treinamento é dividido em três fases: propagação, cálculo do erro e propagação reversa.

Durante a fase de treinamento deve ser apresentado um conjunto formado pelo par: entrada para a rede e valor desejado para resposta a entrada. A saída será comparada ao valor desejado e será computado o erro global da rede, que influenciará na correção dos pesos no passo de retropropagação. Apesar de não haver garantias que a rede forneça uma solução ótima para o problema, este processo é muito utilizado por apresentar uma boa solução para o treinamento de MLP.

De acordo com Haykin (2001) por ser um algoritmo de treinamento supervisionado, para cada vetor de entrada $E = E1, E2, ..., Ei$ onde é informada uma saída esperada $S = S1, S2, ..., Sk$. A saída $S$ é formada pela terceira fase do algoritmo, nessa fase cada entrada $E$ recebe um sinal de entrada. O sinal de entrada é multiplicado por um peso $vij$, esse resultado passa pela função de ativação nas camadas intermediárias. O resultado da função da ativação passa novamente pela função de ativação da camada de saída, formando um vetor de saída $S$. Esse vetor é comparado com o vetor padrão, dessa comparação são ajustados os pesos.

O erro é calculado a partir do resultado esperado, sendo este usado como ajuste dos pesos entre as camadas. Para correção entre as camadas de saída e intermediárias, o fator de ajuste dos pesos pode ser calculado de acordo com a Equação 3.1, onde $y_{ink}$ é à entrada

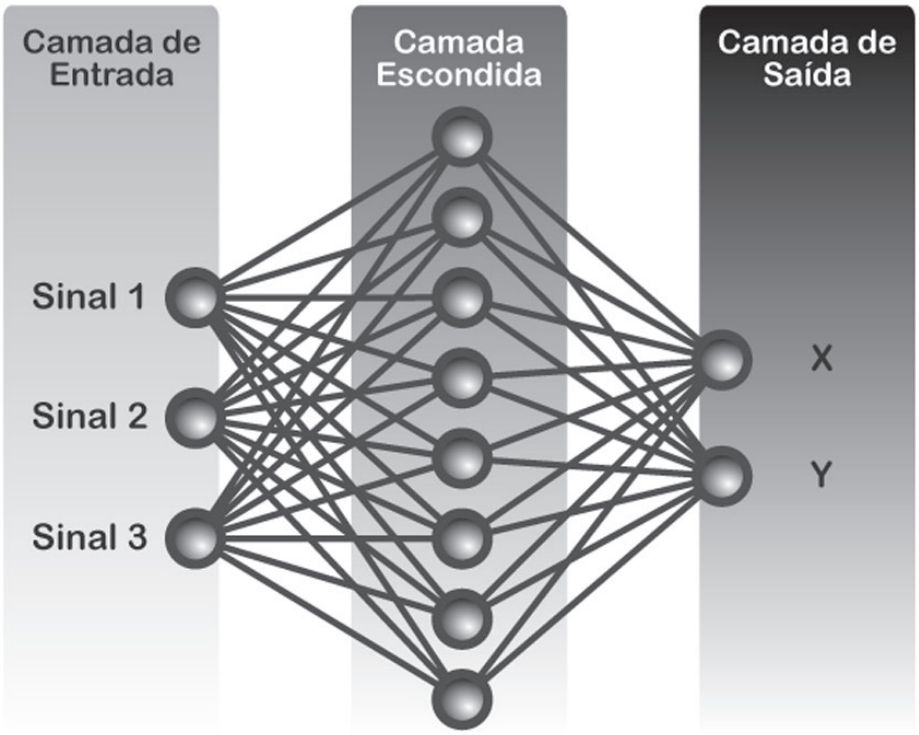



Figura 3.12: Visão da MLP (adaptado de (Freeman & Skapura., 2004)).

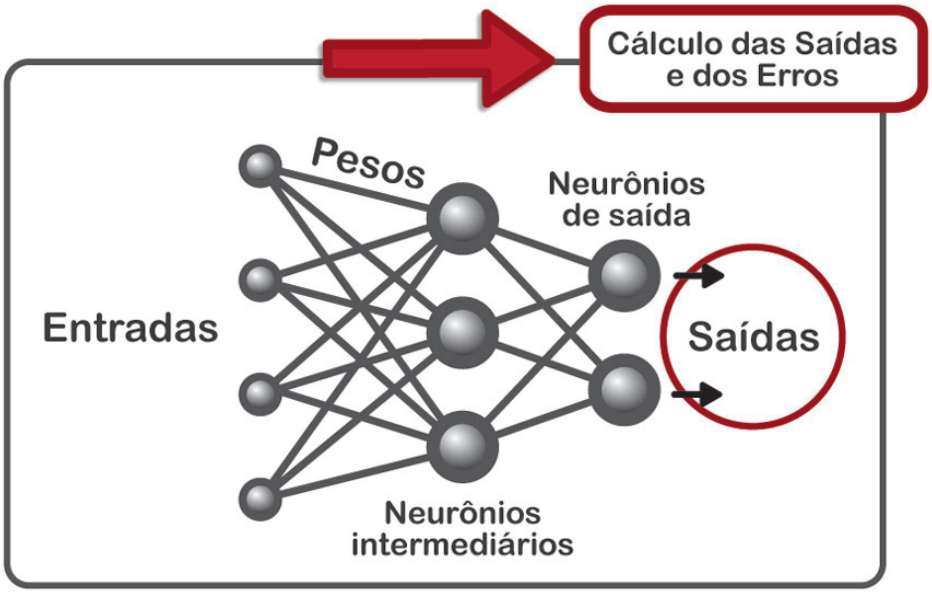



Figura 3.13: $1^o$ Fase - Propagação: Depois de apresentado o padrão de entrada, a resposta de uma unidade é propagada como entrada para as unidades na camada seguinte, até a camada de saída, onde é obtida a resposta da rede e o erro é calculado. (adaptado de (Haykin, 2001)).

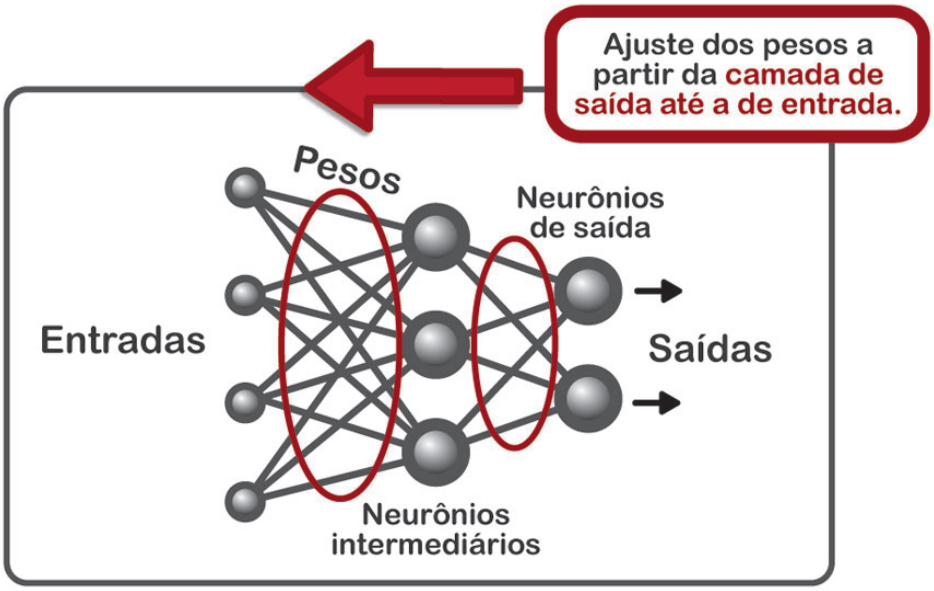



Figura 3.14: $2^o$ Fase - Retropropagação ("*backpropagation*"): Desde a camada de saída até a camada de entrada, são feitas alterações nos pesos sinápticos. (adaptado de (Haykin, 2001)).

da unidade e $f$ a função de ativação, $S$ é a pontência interna do neurônio e $k$ é a camada .

$$\delta k = (Sk - yk) \cdot f(y_{ink}) \tag{3.1}$$

Já entre a camada de entrada e camada escondida usamos a Equação 3.2, seguindo a mesma notação da Equação 3.1, onde $m$ é o numero de camadas de saída e o símbolo $z_{in}$ significa a entrada da camada escondida.

$$\delta j = (m\Sigma k = 1\delta \cdot w) \cdot f(z_{in}) \tag{3.2}$$

Depois de calculado todos os fatores de correção é que os pesos são atualizados segundo a Equação 3.3 e a Equação 3.4 onde $\alpha$ é o fator de correção ou parâmetro que regula a velocidade da convergência (chamado de taxa de aprendizagem ou *learning rate*) e $j$ é a unidade escondida. Na Equação 3.3 $i$ é o peso associado aos neurônios da camada de entrada e $j$ da camada escondida. Na Equação 3.4 $i$ é peso associado aos neurônios da camada escondida e $k$ da camada de saída. $E$ é uma função objetivo, onde $E$ nas Equações 3.3 e 3.4) deve ser minimizado.

$$vij = vij + \alpha \cdot \frac{\delta E}{\delta vij} \tag{3.3}$$

$$wik = wjk + \alpha \cdot \frac{\delta E}{\delta wkj} \tag{3.4}$$

### 3.3.2 *Backpropagation Momentum*

O *Backpropagation Momentum* se baseia no *Backpropagation* com a adição do método *momentum*, o qual tem como objetivo acelerar a velocidade de convergência da RNA (Wasserman, 1986; Rumelhart et al., 1986; Beale & Jackson., 1990; Freeman & Skapura., 2004).

O *Backpropagation Momentum* utiliza a heurística do gradiente decrescente para ajustar os pesos $w$, seguindo ajuste dos erros em direção a um ponto mínimo (Wasserman, 1986). Assim, adiciona ao cálculo do valor da mudança do peso uma fração proporcional à alteração anterior. Dessa maneira, tendendo a aumentar a estabilidade do processo de aprendizagem. A Equação 3.5 especifica o ajuste das conexões entre as camadas pela aplicação do termo *momentum*, onde $\alpha$ representa o termo momentum, $0 < \alpha < 1$.

$$\Delta w^{(l),\lambda}\_ij\ _{(t+1)} = \eta\delta^{(l),\lambda}\_ix^{(l-1)}\_j + \alpha(w^{(l),\lambda}\_ij\_(t) - w^{(l),\lambda}\_ij\_(t-1)) \tag{3.5}$$

Dessa forma, assim que um mínimo é encontrado, seja global ou local, o aprendizado cessa (Freeman & Skapura., 2004). Se a rede alcançar um mínimo local, do seu ponto de vista limitado, todas as direções em sua volta representam valores maiores que o alcançado e, conseqüentemente, a convergência para o mínimo global não é atingida. Nesse caso, a magnitude do erro da rede pode ser muito alta e, portanto, inaceitável.

Caso a rede neural encerre o aprendizado antes que uma solução satisfatória seja obtida, o redimensionamento do número de unidades ocultas ou da taxa de aprendizagem e do termo *momentum* podem ser suficientes para resolver o problema. De acordo com (Freeman & Skapura., 2004), outra possibilidade para se tentar encontrar o mínimo global é realizar o treinamento a partir de um conjunto de pesos inicial diferente daquele utilizado anteriormente.

### 3.3.3 *Levenberg-Marquardt*

Segundo Hagan & Menhaj (1994) o modelo *Levenberg-Marquardt* se caracteriza por ser um dos métodos mais rápidos para treinamento de RNA acíclicas de tamanho moderado. Este modelo é capaz de minimização de funções com um pequeno número de parâmetros.

Em Marquardt (1963); Hagan & Menhaj (1994) uma das características mais importantes desse modelo é o tempo de convergência da rede, que tende a ser menor que o *Backpropagation*. Isso porque o *Backpropagation* padrão utiliza a redução do erro como método de aproximação do mínimo da função de erro, enquanto *Levenberg-Marquardt* faz uso de uma aproximação pelo método de Newton, sendo esta obtida da modificação do método de Gauss-Newton introduzindo o parâmetro $\mu$ gerando a Equação 3.6.

De acordo com Marquardt (1963); Hagan & Menhaj (1994), o parâmetro $\mu$ funciona como um fator de estabilização do treinamento, ajustando a aproximação de forma a utilizar a rápida convergência do método de Newton e evitando passos muito grandes que possam levar a um erro de convergência.

$$\Delta x = [J^t(x)J(x) + \mu I]^{-1}J^t(x)e(x) \tag{3.6}$$

Onde $I$ é a matriz identidade, $e(x)$ é o erro e $J$ é a matriz Jacobiana. O parâmetro $\mu$ é multiplicado por um fator ($\beta$) toda vez que um passo resultar num aumento na função erro,

que se deseja minimizar. Quando um passo resultar no decremento da função genérica $V(x)$, $\mu$ é dividido pelo fator $\beta$. Isto significa que se houver uma convergência para o mínimo da função, $\mu$ é pequeno e o algoritmo se aproxima do método de Newton (passo $\frac{1}{\mu}$). No caso de não haver convergência, o método se aproxima da redução do erro.

Esse método apresenta convergência em menos iterações, mas requer mais cálculos por iteração devido ao cálculo de matrizes inversas. Apesar do grande esforço computacional, ele segue sendo o algoritmo de treinamento mais rápido para redes neurais, quando se trabalha com um número moderado de parâmetros na rede. Se esse número é elevado, a utilização desse algoritmo é pouco prática.

### 3.3.4 Considerações Finais

Conforme exposto, a idéia central para o uso de RNA é que computações complexas podem ser obtidas por processamento simples. Sendo a localização um dos atributos mais utilizados para aplicações sensíveis ao contexto, um dos meios para estimar o posicionamento em ambientes fechados é usar RNA. Uma das vantagens de utilizar RNA para localização *indoor* é a robustez contra ruídos e interferências que são um dos principais fatores que afetam a precisão da localização. A proposta apresentada na Seção 5 considera o uso de RNA para localização *indoor* de usuários móveis. No próximo Capítulo 4 serão apresentos alguns trabalhos correlatos ao projeto proposto.

# Capítulo 4

# Revisão de Sistemas de Localização

A localização de dispositivos móveis é de grande importância para aplicações sensíveis ao contexto, considerando que a localização dos dispositivos freqüentemente corresponde à localização do usuário portador do dispositivo móvel.

Dessa forma, neste Capítulo serão apresentados alguns métodos, bem como alguns dos principais algoritmos utilizados para localização de dispositivos móveis em ambientes fechados.

## 4.1 Métodos de Localização

Existem diversas formas de inferir a posição e a localização de dispositivos móveis. Algumas já são conhecidas há muito tempo como GPS e iinfravermelho que são tecnologias usadas na localização.

Pode-se dividir esses métodos de localização em dois grupos principais: para ambientes abertos e para ambientes fechados. Em ambientes abertos destaca-se o uso do GPS, sendo uma das técnicas mais precisas em termos de tecnologias para inferir a posição do dispositivo. No entanto, além de ter alto custo de manutenção, por causa do satélite e de todo o equipamento especializado, apresenta problemas quando usada para localizar dispositivos que estejam dentro de ambientes fechados, edifícios ou casas. A Figura 4.1 exemplifica esse processo (Bishop et al., 2008), onde celular recebe sinais dos satélites GPS e calcula a localização, o aparelho transmite a localização através de uma chamada de dados ou voz. A empresa recebe a chamada de dados ou voz juntamente com a localização.

Outra tecnologia largamente utilizada é aquela usada para localização de telefones celulares. Utiliza técnicas como TDoA e AoA para inferir a posição do dispositivo (Bishop et al., 2008). Essa técnica apresenta desempenho satisfatório quando usada em ambientes abertos, mas sofre de diversos problemas em ambientes fechados, como reflexão do sinal pelas paredes e andares dos prédios, por exemplo. A Figura 4.2 exemplifica esse processo, onde o celular faz uma chamada e três antes captam essa chamada e medem a diferença de tempo no sinal, assim traduzindo essa diferença em localização. A empresa recebe a chamada com a localização.

Para ambientes fechados, uma das tecnologias mais usadas no começo dos estudos sobre localização de dispositivos móveis em ambientes fechados foi o de sinal infravermelho. O *Active Badges* (Want et al., 1992) foi um projeto pioneiro nessa área e era constituído de um dispositivo que emitia sinais infravermelhos, enviando os sinais a cada 10 segundos.


Figura 4.1: Localização por GPS (adaptado de (Bishop et al., 2008)).


Figura 4.2: Localização por TDoA ou AoA (adaptado de (Bishop et al., 2008)).

Sensores posicionados nas salas do edifício captavam esses sinais e o utilizavam para inferir a posição do dispositivo. Muitas variações do *Active Badges* foram criadas, no entanto, sinais infravermelhos não têm alcance muito elevado e isso prejudica a escalabilidade da solução. Em locais com muitos ambientes praticamente um sensor infravermelho deveria ser instalado para cada uma, elevando o custo de manutenção e instalação. Além disso, sinais infravermelhos sofrem atenuação quando expostos à luz do sol, prejudicando a captação pelos sensores.

Pelos altos custos do uso do infravermelho cientistas passaram a usar o sinais RSSI fornecidos por dispositivos *wireless* que já se encontravam no ambiente. Dessa forma, a técnica de análise por *fingerprinting*, é uma metodologia usada para ultrapassar as adversidades encontradas em ambientes fechados. Essa é uma metodologia que analisa um determinado cenário recorrendo às características eletromagnéticas disponíveis no ambiente. Esta metodologia, sendo genérica para qualquer sinal RF, é implementada tendo como base um dicionário de domínios denominado de Mapa de *Fingerprinting* (MF). O principal desafio decorre na elaboração coerente e eficaz deste mapa, tendo em conta uma das principais características deste tipo de estrutura de dados: a chave não pode ser repetida, ou seja, sendo a chave o conjunto de valores de RSS captados num dado ponto do mapa, a combinação destes valores deve ser única, sob pena de ambigüidade de resultados.

Apesar de se poder obter *fingerprints* a partir de qualquer propriedade do sinal a medir, é mais comum usar o valor de RSSI, pois este é normalmente cedido pelo controlador do dispositivo móvel e caracteriza mais uma determinada localização do que outro valor também normalmente acessível: relação Sinal-Ruído (SNR) (Bahl & Padmanabhan, 2000). O valor de RSS não apresenta, no entanto, uma variação linear com a distância nem se pode considerar constante com o tempo para um dado ponto, sendo considerada uma variável aleatória e o seu tratamento pode ser efetuado recorrendo à estatística descritiva. A aplicação deste método requer a elaboração de um Mapa para encontrar um valor aproximado da localização de um determinado dispositivo móvel numa rede sem fios, surgindo assim à necessidade de dividir o trabalho em duas fases:

- Fase de Calibração ou Fase *Offline* - procede-se à elaboração do MF, ou seja, construir uma plataforma de mapeamento entre os valores dos sinais e as coordenadas ou significantes;
- Fase *Online* - Medem-se os valores de RSS captados pelo dispositivo móvel, ou seja, os sinais em tempo real, sendo estes processados pelo respectivo algoritmo de localização, que vai mapeá-los para valores de coordenadas, resultando do processo um valor aproximado da localização desse mesmo dispositivo.

Os algoritmos de cálculo da localização, ou LEA são algoritmos que exploram a dependência entre as coordenadas definidas para localização e os *fingerprints* guardados na base de dados, de modo a calcular a posição das amostras de RSS obtidas na fase online (Kaemarungsi, 2005). Estes algoritmos não são mais do que detectores de padrões, pois separam as amostras que vão chegando da fase *online*, consoante algum padrão em diferentes classes, classificando-as como pertencentes a uma determinada zona e por vezes extrapolando as suas possíveis coordenadas. Estes algoritmos podem ser classificados em dois tipos:

- Determinísticos - utilização das distâncias no domínio dos sinais para acharem o ponto que melhor é caracterizado pelo *fingerprint* obtido na fase *online*. O processamento é feito, sobretudo nesta fase, e.g., *Nearest Nieghboor* e RNA;
- Probabilísticos - considera informações características do cenário, funções de probabilidade e teorema de *Bayes*. O processamento é feito, sobretudo na fase de calibração, e.g., *Support Vector Machines* (SVM) e métodos que usam o teorema de *Bayes*.

## 4.2 Algoritmos usados pelo método *Fingerprinting*

O processo de criação de mapas *fingerprinting* é um método de localização em ambientes fechados, o qual consiste de duas fases: (i) *offline* coleta um conjunto de valores RSSI de PAs, assim criando a base do MF e (ii) analisa um determinado número de valores RSSI coletados em tempo real e compará-los com os pertencentes ao MF recorrendo a algoritmos LEA.

Nesse Seção serão abordados os principais LEA, que são utilizados pelo método de *fingerprinting*.

### 4.2.1 *Nearest Neighbor*

O algoritmo *Nearest Neighbor* caracteriza-se pelo cálculo das distâncias entre os valores medidos na fase *online* e todos os valores presentes no mapa de *fingerprinting*, obtido durante a fase de calibração (Lin & Lin, 2005). A distância é calculada usando a função $Dist(S, f_i)$, uma representação abstrata das funções de distância, (e.g., a Euclidiana).

Tendo como $M$ um mapa de *fingerprinting* com $l$ pontos distintos de coordenadas, $P = (p1, p2, ..., pl)$, e pelos correspondentes sinais, $F = (f1, f2, ..., fl)$, sendo $pi$ e $fi$ os pontos de coordenadas e sinais respectivamente, em que em todos os $pi$ corresponde um e um só $fi$. Então para um sistema de $m$ dimensões espaciais e $n$ referências temos as Equações 4.1 e 4.2:

$$p_i = (pi_1, pi_2, ..., pi_m)_T \tag{4.1}$$

$$f_i = (fi_1, fi_2, ..., fi_m)_T \tag{4.2}$$

Na etapa de calibração, o algoritmo ingênuo retira um conjunto considerável de valores dos sinais em cada ponto e considera cada $fi$ como o valor médio de todos os valores de RSS retirados no ponto $pi$. Sendo ps o ponto cuja localização é desconhecida e $S = (s_1, s_2, ....s_n)_T$ o vetor de valores de RSS obtido na fase *online* desse ponto, o LEA a calcula o minimizaste $fi$ da função $Dist(S, f_i)$ adotada, conforme a Equação 4.3.

$$Dist(S, f_i) < Dist(S, f_\kappa) \forall \kappa = i \tag{4.3}$$

Estando de posse de um MF que contém apenas uma amostra limitada de pontos do espaço é necessário fazer extrapolação dos resultados. Assim, em vez de se usar somente os pontos cuja distância dos sinais é mínima, pode-se utilizar uma relação entre os $k$ pontos em que este valor é menor. Este método denomina-se de $k - Nearest - Neighbor$.

Esses algoritmos são do tipo determinístico, pois apenas necessitam de algumas operações estatísticas dos *fingerprints* tal como a média e o desvio padrão.

### 4.2.2 Redes Neurais

O uso de redes neurais nos sistemas de localização por *fingerprinting* pressupõe que o comportamento do RSS é extremamente complexo para ser analisado matematicamente (Kaemarungsi, 2005). Assim, uma estratégia seria utilizar uma estrutura de processamento vista exteriormente como uma caixa preta constituída por um conjunto de neurônios, ao invés de tentar modelar o sistema recorrendo a uma função complexa e não linear, conforme apresentado na Seção 3.3.

Dessa forma, pode-se utlizar RNA para diminuir a complexidade do problema. Onde cada RNA pode ser treinada com várias amostras de *fingerprints* da base de dados calculando-se os pesos sinápticos associados a cada uma das entradas. O processo de treino consiste em manipular os *fingerprints* captados na fase de calibração para que se obtenha a saída desejada (as coordenadas dessa localização). Dentre as várias técnicas de treino pode-se citar a *One Step-Secant* (OSS) presente no trabalho de (Battiti et al., 2002) e a de *backpropagation error* usada em (Saha et al., 2003).

A principal limitação deste método é o tempo necessário para treino da rede neural. Nesse período caso faça-se iterações além do necessário, pode levar ao *over-training* que degrada a desempenho do sistema.

### 4.2.3 Método Bayesiano

O Método Bayesiano utiliza como base o teorema de Bayes para calcular a localização. Para tanto, pressupõe-se um conhecimento prévio da distribuição de probabilidade da localização do dispositivo móvel (Youssef & Agrawala, 2003). Informações sobre o comportamento padrão (Roos et al., 2002) e um estudo dos modelos de propagação (Brunato & Battiti, 2005), podem ser utilizadas também. O estudo dos modelos de propagação pode ser analítico ou empírico. No modelo analítico faz-se necessário adotar o modelo de propagação mais adequado, fazendo com que não sejam necessárias medidas na fase de calibração, sendo os *fingerprints* calculados com recurso a modelos matemáticos. No modelo empírico são necessárias muitas observações para se definir um modelo confiável (Brunato & Battiti, 2005).

Usando $P(F|L)$ a probabilidade do conjunto de *fingerprints* $F$ para as coordenadas $L$, denominada de função de distribuição de probabilidade. Em (Roos et al., 2002) são apresentados dois métodos para a obtenção/estimação desta função, a saber:

- *Kernel Method* - Dadas $n$ amostras do valor de RSS pertencentes a uma determinada referência detectada num ponto específico, este método gera uma função de probabilidade, como a função Gaussiana, que é denominada de *kernel*. Considerando a função Gaussiana como *kernel* onde $\rho$ é a média, que devido ao fato desta função ser uma *kernel function*, este valor tem de ser igual a uma das amostras e $\sigma$ é o valor de ajuste da função *kernel*, como apresentado na Equação 4.4.

$$P(f_i|L) = \frac{1}{n}\sum_{j=1}^{n}\frac{1}{\sqrt{2\pi\sigma}}\exp(-\frac{(s-p)^2}{2\sigma^2}) \tag{4.4}$$

- *Histogram Method* - Este método está ligado as conversões de discreto para contínuo, estimando funções de probabilidade contínuas usando funções de probabilidade discretas (Roos et al., 2002). Para a sua implementação é definido um conjunto de *bins*, ou seja, um conjunto de intervalos não sobrepostos que cubram todos os valores possíveis da variável, desde o seu mínimo até ao seu máximo, sendo anotada a freqüência das amostras contidas no respectivo intervalo.

De posse da função de distribuição de probabilidade, assumindo que cada ponto tenha uma probabilidade *a priori*, $P(L)$, a qual no início da calibração do sistema pode ser considerada a mesma. Assim, considerando $F$ o vetor de *fingerprints* obtido na fase *online* e $M$ o $MF$.

A localização que melhor corresponde à atual *fingerprint* vai ser aquela cuja probabilidade $P(L|F)$ for mais elevada. Considerando dois *fingerprints* $F_A$ e $F_B$ candidatos à localização $L$, $F_A$ é escolhido em relação à $F_B$ como apresentado na Equação 4.5.

$$P(F_A|L) > P(F_B|L) \tag{4.5}$$

Este método considera a informação do cenário e pode obter melhor desempenho do que nos determinísticos apesar da sua fase de calibração ser bastante complexa.

### 4.2.4 *Support Vector Machine Methods*

Em Brunato & Battiti (2005) é apresentado um algoritmo de localização usando SVM. Este algoritmo usa o princípio de *stactical learning theory* apresentado em Vapnik (1998), que combina várias áreas como estatística, aprendizagem de máquina e redes neurais. Diferente do método probabilístico utilizando teorema de Bayes, este método não necessita de informações detalhadas como o conhecimento de modelos de propagação para o mapeamento entre os valores de sinais para as coordenadas. Uma das vantagens deste método é a sua característica de generalização da classificação, o que vai minimizar o erro durante a fase de testes, não ocorrendo o problema de *overfitting*.

O conceito central desse algoritmo é *Structural Risk Minimization* (SRM), ou seja, tenta minimizar uma função denominada de função de risco. Neste caso a função de risco é uma estimativa do valor da função *loss*, função que, em análise de padrões, é definida como o grau de proximidade entre o padrão real e o aproximado. A função geral de risco é limitada pela função empírica de risco e o intervalo de confiança de Vapnik-Chervonenkis (Kaemarungsi, 2005).

Segundo Kaemarungsi (2005), o problema do RSS *fingerprinting* pode ser classificado como sendo um caso não-linear. O seu funcionamento é feito em duas partes:

1. Os vetores de *fingerprinting* são mapeados para uma dimensão superior denominada *feature space* usando uma função chamada de *kernel of the* SVM de modo a ser efetuada a transformada do vetor. Em Brunato & Battiti (2005) são apresentadas várias funções deste tipo como, por exemplo: polinomiais, *Radial Basis Functions* (RBF), *Sigmoid kernel* e *Analysis of Variance Kernel* (ANOVA).

2. O algoritmo SVM cria uma superfície de decisão (*optimal separating hyperplane*) na região denominada de *feature space* de modo a efetuar a classificação. Tipicamente este hiperplano não é único e geralmente é ótimo quando se apresenta à máxima

distância possível do ponto de treino mais próximo, sendo assim os vetores de suporte considerados para a elaboração do hiperplano de decisão os *training vectors*.

## 4.3 *Location-based Services*

LBS como o próprio nome indica, são serviços baseados em localização. Esta é uma área emergente, que integra localização ou posicionamento de dispositivos móveis com outras informações. Há cada vez mais pesquisas e aplicações, as quais se baseiam em localização, como por exemplo: guia em campus universitário, guia em museu, aplicativo que informa os restaurantes mais perto, entre outros.

Os serviços baseados em localização estão crescendo e evoluindo conjuntamente com os dispositivos móveis. Existem cada vez mais equipamentos móveis com mais capacidade, que suportam aplicativos já com alguma complexidade. O modo de obter localização, pode ser feito de várias formas. Este processo é relativo ao problema em questão e também à precisão necessária.

Dessa forma, nesta Seção serão apresentados e analisados três trabalhos sobre o problema da localização de dispositivos móveis em ambientes fechados.

### 4.3.1 Projeto RADAR

Um dos primeiros projetos para de localizar um dispositivo que use uma tecnologia semelhante à *Wi-Fi* foi o projeto RADAR (Bahl & Padmanabhan, 2000). Este projeto usa a WaveLan, que é uma tecnologia de rede sem fios proprietária, a qual antecedeu o *Wi-Fi*. No projeto foram usados computadores pessoais para servirem de estações fixas que tinham como função medir a potência do sinal enviado pelo computador móvel. Pelo fato do sinal ser medido nas estações fixas gerava uma limitação, que caso fosse necessário monitorar um grande número de dispositivos móveis, seria necessário passar à medição da potência de sinal nos dispositivos móveis (Bahl & Padmanabhan, 2000). Isso porque um servidor, na época, não teria desempenho computacional suficiente para processar as localizações de um grande número de dispositivos móveis.

O RADAR funcionava em duas fases: uma *offline*, na qual era realizado o treino do sistema, e outra *online*. A fase *online* poderia funcionar em tempo real ou então poderia funcionar em duas subfases, sendo feito primeiro toda a aquisição de dados do dispositivo móvel e posteriormente feita à análise ao percurso seguido pelo dispositivo móvel durante o intervalo de tempo monitorizado.

Na fase *offline*, o usuário tinha de indicar em um mapa qual a sua coordenada espacial $(x,y)$, como também a sua orientação $(d)$. Por fim era registrado o *timestamp* $(t)$ e os dados inseridos pelo usuário eram guardados em uma tupla no formato $(t, x, y, d)$.

Para calcular a posição do dispositivo móvel era necessário comparar as potências dos sinais recebidos por três computadores fixos com as amostras recolhidas na fase *offline* de treino. Ao encontrar a amostra que mais se assemelha ao sinal atual, encontrava qual era a posição em coordenadas respectivas para aquele conjunto de potências de sinais. Foi utilizada a técnica de *Multiple Nearest Neighbour* (MNN), que consiste no cálculo da diferença dos sinais *Wi-Fi* medidos na fase *online* em relação a todos os sinais registrados na fase *offline*. A menor diferença entre os valores já registrados e os que o dispositivo

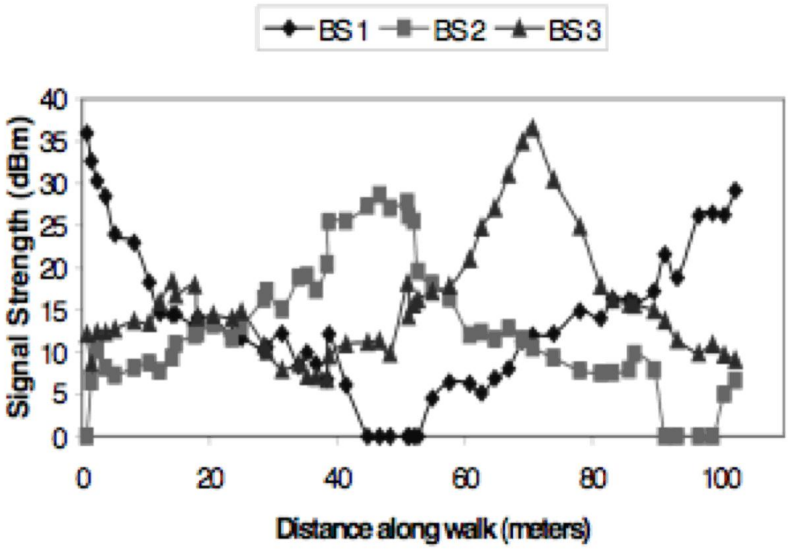



Figura 4.3: Relação da distância e potência dos sinais *Wi-Fi* recebidos pelos PA (Bahl & Padmanabhan, 2000)

móvel recebe no presente momento, indica qual a posição cujo algoritmo estimou, assim é a posição que mais se aproxima das potências de sinal recebida.

Como os sinais RSSI têm um comportamento pouco previsível, sendo difícil de afirmar que a distância é o principal fator de atenuação da potência. A Figura 4.3, apresenta três computadores fixos (BS1, BS2 e BS3) registraram valores pouco consistentes na relação da distância entre o dispositivo móvel e os computadores, com a atenuação da potência do sinal *Wi-Fi*. Esta fraca correlação entre distância e atenuação foi em grande parte causada pelo local. Pelos dados apresentados foi concluído que a disposição das paredes e outros obstáculos influenciaram em muito os resultados (Bahl & Padmanabhan, 2000).

### 4.3.2 Projeto Horus

Outro sistema de localização foi desenvolvido pela Universidade de Maryland em 2005 chamado Horus (Youssef et al., 2003). Tinha como objetivo atingir dois importantes marcos: alta precisão de localização e baixos requisitos computacionais. Devido a técnicas de *location- clustering*, o algoritmo usado no Horus torna-se muito leve, permitindo a localização de um grande número de dispositivos móveis (Youssef et al., 2003). A alta precisão atingida deve-se em grande parte à identificação que o Horus faz às diferentes causas da variação dos sinais RSSI e como ele as consegue resolver.

O Horus é instalado nos dispositivos cliente, local onde mede a potência de sinal recebido vindo de todos os PA alcançáveis pelo cliente. O sistema funciona em duas fases, devendo ser treinado e calibrado na primeira fase para a construção de um mapa *fingerprinting*. O mapa serve de base para o processo de localização dos dispositivos móveis.

Sabendo da variação de potência do sinal ao longo do tempo, (Youssef et al., 2003) mostra que a auto-correlação das amostras recolhidas de um ponto de acesso é de 90%

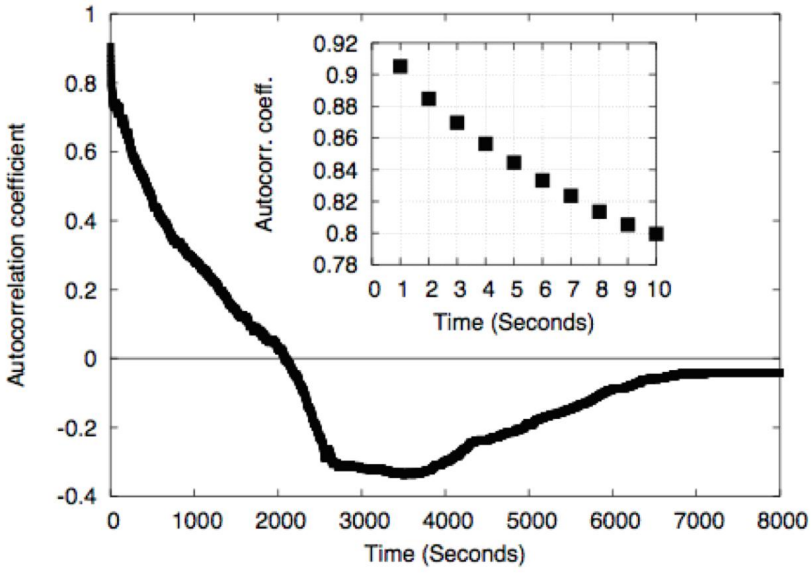



Figura 4.4: Auto-correlação entre potência de sinais vindos de um PA (Youssef & Agrawala, 2003)

para um curto espaço de tempo (Figura 4.4), tendo proposto um modelo auto-regressivo (Equação 4.6) para explicar a auto-correlação dos valores.

$$st = \alpha st - 1 + (1 - \alpha)vt; 0 \leq \alpha \geq 1 \tag{4.6}$$

Na Equação 4.6 $vt$ é um processo de ruído e $st$ é uma série que representa as amostras de um ponto de acesso. Com base neste modelo, os investigadores desenvolveram uma regra (Equação 4.7) para explicar a variância dos sinais. Na fase de treino o valor de $\alpha$ é estimado e guardado como o parâmetro de distribuição para que durante a fase de localização, a distribuição Gaussiana seja adaptada com o seu valor.

$$\frac{1+\alpha}{1-\alpha}\sigma^2 \tag{4.7}$$

A localização de um dispositivo é determinada estimando quais os locais $p(i)$ com a maior probabilidade do dispositivo se encontrar, sendo depois achado qual o centro de massa desse conjunto de locais $N$ (Equação 4.8), sendo representados por coordenadas espaciais. O centro de massa $X$ calculado passará a ser o novo ponto representativo da localização do dispositivo.

$$x = \frac{\sum_{i=1}^{N} p(i)xi}{\sum_{i=1}^{N} p(i)} \tag{4.8}$$

Sabendo que a potência dos sinais recebidos pelo dispositivo não são constantes para uma dada posição, o Horus usa um compensador de pequena escala para corrigir estas variações. O compensador detecta as variações de pequena escala calculando a distância entre duas localizações consecutivas estimadas.

Para reduzir os requisitos computacionais do Horus, foi utilizada a técnica *Incremental Triangulation Clustering*, a qual consiste em dividir em várias partes o ambiente em que o dispositivo a localizar se encontra (Youssef et al., 2003). O ambiente foi dividido em vários conjuntos de PAs. Dessa maneira, são analisados apenas os sinais vindos dos PAs estritamente necessários à localização, não sendo necessário analisar todos os sinais recebidos pelo dispositivo móvel.

No processo de localização são comparadas as, probabilidades das estimativas das localizações e caso, a primeira estimativa tenha a maior probabilidade do que a segunda estimativa, a localização pára e é devolvida à primeira. Este processo é recursivo, podendo verificar várias estimativas até se chegar ao resultado pretendido. Com estas técnicas probabilísticas o Horus consegue localizar um dispositivo móvel em redes *Wi-Fi*, com um erro médio de 0,6 metros em 60% dos casos.

### 4.3.3 Sistema RTLS

Outro sistema relevante é o *Ekahau Real-Time Location System* (RTLS), que foi desenvolvido na Finlândia pela empresa Ekahau (Ekahau). O Ekahau é atualmente o sistema de localização mais preciso, estando o erro médio da localização de dispositivos móveis em *Wi-Fi*, abaixo de 1 metro. Esta precisão juntamente com o fato da localização ser processada em um dispositivo central, permite que seja possível localizar dispositivos móveis com pouco poder computacional. Dessa forma possibilita que possam ser localizados com grande confiabilidade os recursos de qualquer tipo de instituição, sejam estes recursos humanos ou materiais.

Este sistema de localização usa mapas *fingerprinting* previamente calculados, para poder determinar as posições dos dispositivos móveis. Não há necessidade de uma elevada concentração de PAs para seu funcionamento. Contudo, a fase de calibração deve ser muito precisa, sendo executada com o maior rigor de modo a que o sistema possa localizar com o mínimo de erros.

O Ekahau RTLS utiliza o *Ekahau Positioning Engine* 4.0, servidor baseado em Java que disponibiliza diversos serviços de localização a aplicações clientes (Ekahau). Dos sistemas de localização, o Ekahau é dos poucos a ser comercializado, tendo alcançado bons resultados em ambientes hospitalares, na localização de *stocks* e na indústria automotiva.

### 4.3.4 Projeto MoCA

O MoCA da Pontifícia Universidade Católica do Rio de Janeiro é um *middleware* baseado em serviços, flexível e extensível. O MoCA é composto por um conjunto de serviços e Application Programming Interface (API)'s, que auxiliam ao suporte no desenvolvimento de aplicativos distribuídos sensíveis ao contexto. Os aplicativos desenvolvidos são direcionados a utilizarem uma infra-estrutura de rede local, com suporte a conexões sem fio *Wi-Fi* (IEEE 802.11b/g). O MoCA ainda fornece meios para recolhimento, armazenamento e tratamento de dados dos diferentes contextos aos quais os dispositivos estão submetidos.

Segundo (Rubinsztejn et al., 2004), o MoCA possui em sua arquitetura principal os seguintes componentes:

- *Context Information Service* (CIS) - receber, armazenar e processar informações de contexto enviadas pelos monitores em execução nos dispositivos móveis, que podem ser acessadas de forma síncrona ou assíncrona.
- LIS - responsável por inferir a localização aproximada de um dispositivo móvel, através da comparação do padrão de RF, com padrão de sinais mensurados em pontos de referência pré-definidos, sendo possível definir regiões simbólicas, feitas por uma associação de nomes a regiões físicas definidas (ex: prédios, salas).
- *Symbolic Region Manager* (SRM1) - estabelece relações entre as regiões definidas pelo LIS, caracterizando uma hierarquia.
- *Configuration Service* (CS) - responsável pelo armazenamento e gerenciamento das informações de configuração de cada dispositivo utilizadas pelo CIS, armazenando-as em tabelas *hash*, e indexadas pelo endereço MAC do dispositivo.
- *Discovery Service* (DS) - armazena informações como nome, propriedade e endereço de qualquer aplicação e seus componentes ou serviço registrado e disponibiliza seu endereço para outros clientes e serviços.
- *Monitor* - é executado em cada um dos dispositivos móveis, sendo responsável por armazenar dados relativos ao estado e ambiente de execução do dispositivo e enviá-los para o CIS. Dentre os dados coletados estão incluídos dados relativos a qualidade da conexão, energia, uso de CPU, uso de memória, entre outros.

A Figura 4.5 apresenta a arquitetura do LIS. O *stub* CIS requisita periodicamente o RSSI dos PA, coleta-os e prepara-os para inferir a localização. O *Inference Engine* utiliza o algoritmo MNN , assim como utilizado no projeto RADAR (Bahl & Padmanabhan, 2000).

Os pontos mensurado pelo MoCA foram escolhidos aleatoriamente em uma área de 600 m^2, utilizando oito PAs *Wi-Fi*. A Tabela 4.1 apresenta o erro em metros obtido com 50%, 70% e 90% dos pontos utilizados para teste. A Tabela 4.1 mostra uma variação de 1.28m entre 90% e 50%.

Tabela 4.1: Tabela de Resultados do projeto MoCA (adaptada de (Rubinsztejn et al., 2004)).

| Porcentagem (%) | Erro |
|---|---|
| 30 | 1,56 |
| 70 | 1,74 |
| 90 | 2,84 |

## 4.4 Análise Comparativa dos Principais Trabalhos

Um dos trabalhos mais relevantes com relação ao estudo de algoritmos e sistemas de localização foi realizado por Kaemarungsi (2005) . Nesse trabalho foi realizado o estudo

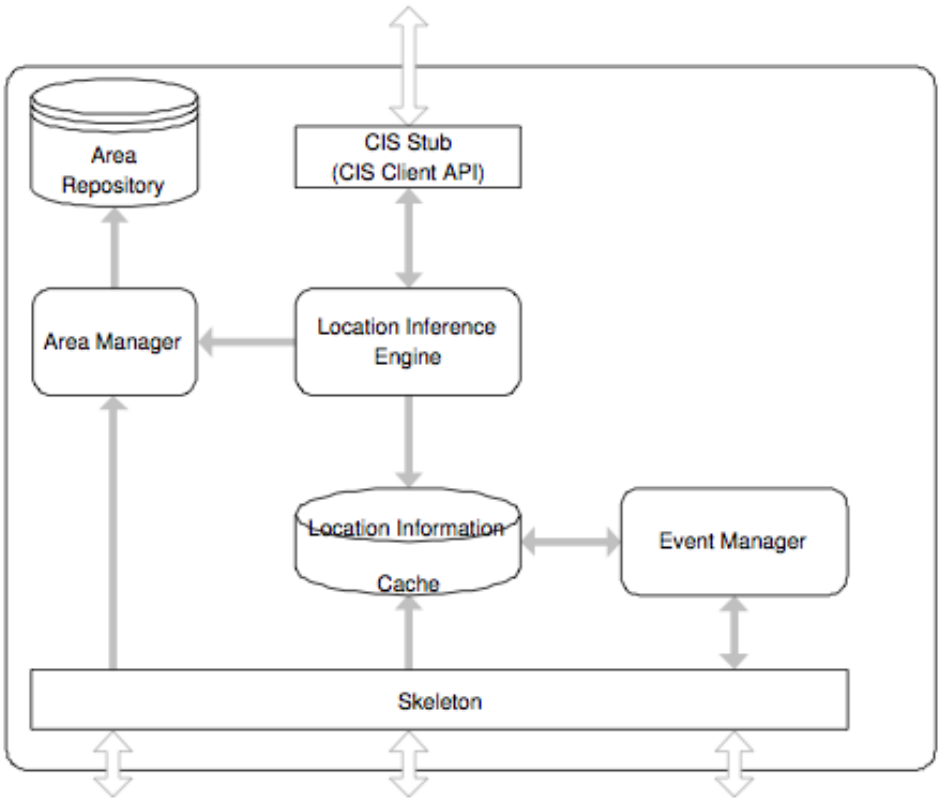



Figura 4.5: Arquitetura do LIS, (Rubinsztejn et al., 2004).

referente à tecnologia IEEE 802.11 uma vez que é a mais estudada para localização de dispositivos móveis. Contudo, não foram realizados experimentos com outros padrões de rede, Kaemarungsi (2005) optou por pesquisar implementações e comparações de diferentes LEA, garantindo assim uma análise de desempenho uniforme.

Em Lin & Lin (2005) é feito um teste de comparação entre três algoritmos de proximidade, o *k-Nearest Neighbor*, um recorrendo a RNA e um algoritmo probabilístico *Bayesiano* sendo analisada precisão, exatidão, complexidade dos algoritmos, escalabilidade e robustez. As conclusões foram que o *k-Nearest Neighbor*, nas mesmas condições do que os outros métodos, como RNA e *Bayessiano*, conseguem atingir a melhor precisão e exatidão, segundo (Lin & Lin, 2005). Entretanto, considerando o parâmetro da complexidade definido em Lin & Lin (2005) como o tempo computacional da fase *online* é o pior. Dessa maneira, o menos complexo é o que faz uso de RNA, porém é o menos preciso e exato. Caso for reduzido o valor do parâmetro $k$, é possível a elaboração de um algoritmo *k-Nearest Neighbor* competitivo no nível de complexidade compatível com os demais e com uma exatidão e precisão aceitável.

Em relação à robustez, todos os métodos atingiram um alto grau de robustez, no primeiro caso, se aparecer um valor de um PA desconhecido. Em relação ao segundo caso em que uma dos PAs deixa de funcionar tanto o probabilístico como o *k-Nearest Neighbor* reage sem grande dificuldade, mas precauções devem ser adotadas nas RNA.

A respeito da escalabilidade mais uma vez não há problemas com o probabilístico, tão pouco com o *k-Nearest Neighbor*, sendo necessária uma setorização para as RNA (Lin & Lin, 2005).

Em seu trabalho Battiti et al. (2002), observou que apesar do método SVMs ser dos mais sofisticados no reconhecimento de padrões, o seu desempenho é bastante próxima dos estudados em (Brunato & Battiti, 2005). Por exemplo, comparando-o com método com o *k-Nearest Neighbor*, temos um erro de 3.96m para este método e 3.93m para o $k - NearestNeighbor$. Contudo, o uso do SVM é mais confiável do que qualquer outro no problema de localizaçao em ambientes fechados (Brunato & Battiti, 2005).

A Tabela 4.2 é uma adaptação de Kaemarungsi (2005), onde apresenta em que condições foram realizados os diferentes testes dos diferentes autores analisados. Pelo fato de não haver uma padronização em relação ao procedimento de análise, cada um dos autores apresentados na Tabela 4.2 é apresentada a maneira de caracterização do cenário e análise de resultados. Conforme exposto, o processo comparativo se apresenta de forma complexa.

Tabela 4.2: Tabela de Comparativa de LBS (adaptada de (Kaemarungsi, 2005)).

| Seção | Trabalho | Tipo de LEA | Exatidão (m) | Precisão (%) | Espaçamento | Posições | PA's | Ambientes |
|---|---|---|---|---|---|---|---|---|
| 4.3.1 | Radar | *Nearest Neighbor* | 2,13 | 30 | Não Uniforme | 70 | 3 | Corredor |
| 4.3.2 | Horus | *Bayesiano* | 2,13 | 90 | 1,5 m | 110 | 4 | Corredor |
| 4.3.3 | RTLS | *Ekahau Positioning Engine* | 1 | N/A | -75 dBm | N/A | N/A | Fechado |
| 4.3.4 | MoCA | *Nearest Neighbor* | 1,56<br>1,74<br>2,84 | 50<br>70<br>90 | N/A | N/A | 8 | 1 piso |

Empiricamente parece óbvio que um sistema que possua maior quantidade de PA terá o melhor desempenho, uma vez que o seu número de sinais é superior alcançando uma melhor distinção entre pontos neste domínio. Uma segunda hipótese é o fato de

que caso o sistema funcione apenas num corredor, sendo uma área mais reduzida, teria menor quantidade de pontos a analisar, conseqüentemente a seu desempenho aumentaria em relação aos realizados em ambientes complexos (e.g com mais de um piso). Porém, o que verifica-se é que ambas as hipóteses falham confrontando com os resultados da Tabela 4.2. A primeira hipótese não se comprova porque Bahl & Padmanabhan (2000) que conta com 3 PAs e apresenta exatidão de $2,13$m sendo melhor que Rubinsztejn et al. (2004) que conta com 8 PAs e apresenta exatidão de $2,84$m. Já a segunda hipótese também não é valido por (Bahl & Padmanabhan, 2000) e (Youssef & Agrawala, 2003) apresentam a mesma precisão de 90% . Dessa forma, esta análise mostra as desvantagens de não haver uma padronização de análise de desempenho nesta área.

# Capítulo 5

# Solução Proposta

Para criação de um sistema de localização baseado em SMA, RNA e QoS, foi definida e apresentada em (Fonseca et al., 2011) uma arquitetura capaz de utilizar a técnica de *fingerprinting* e usá-lo para encontrar as coordenadas necessárias para localização de usuários móveis em ambientes fechados.

Este arquitetura de software foi desenvolvida para ser flexível, considerando a rápida evolução das tecnologias de redes sem fio. Neste trabalho a abordagem usada foi à determinística conforme apresentado na Seçao 4.1. Segundo Kaemarungsi (2005), os resultados da abordagem deterministicas são equiparáveis com a probabilística e podem ser automatizadas.

A implementação da arquitetura de software proposta engloba as fases de aquisiçao do RSSI, análise das coordenadas e utilizaçao de RNA. Há também que se implementar uma interface para que possa ser utilizado por qualquer aplicação que necessite do serviço de localização de usuários em ambientes fechados.

Na Seção 5.1 será feita uma descrição da arquitetura do software de localização usado, enquanto que na Seção 5.2 se descreverá de forma mais detalhada a arquitetura de cada agente.

## 5.1 *Software* de Localização

O protótipo de localização consiste num conjunto de processos que implementam a metodologia de análise de um mapa de sinais RSSI e coordenadas absolutas, ou seja, a técnica *fingerprinting*, sendo dividido pelas seguintes camadas, como se pode verificar na Figura 5.1:

- Interface com LBS - Possibilita a comunicação entre o protótipo de localização e uma aplicação LBS;
- Aquisição de Dados - Permite obter os sinais RSSI de um determinado usuário móvel;
- Controle de Conflitos - Prepara os dados obtidos na camada anterior para serem usados na camada RNA;
- RNA - Efetua o cálculo da localização;


Figura 5.1: Diagrama em Camadas do Protótipo

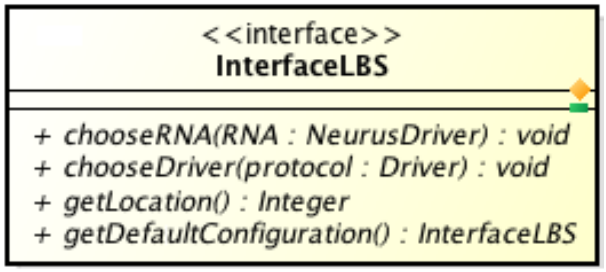



Figura 5.2: Interface Location Base-Service

- QoS - Analisa os dados obtidos da camada RNA, sendo que os resultados da camada RNA são submetidos a dois níveis de QoS, pré-definidos: (i) o primeiro do erro máximo aceitável (1.5m de acordo com (Castro & Favela., 2005)); (ii) o segundo avalia a potência do sinal para avaliar a proximidade a um PA (o erro aceitável é 1.0m nesse nível).

### 5.1.1 Interface com LBS

Esta camada define uma interface que possibilita a comunicação entre o protótipo e outra aplicação que faça uso do serviço de localização. A Figura 5.2 apresentada esta interface sendo denominada InterfaceLBS, a qual possui quatro métodos:

- chooseRNA - escolhe qual o rede neural deverá ser utilizada;
- chooseDriver - seleciona o protocolo de rede;
- getLocation - calcula a localização;
- getDefaultConfiguration - permite que o sistema faça uso de todos os agentes de redes neurais e protocolos de redes implementados.

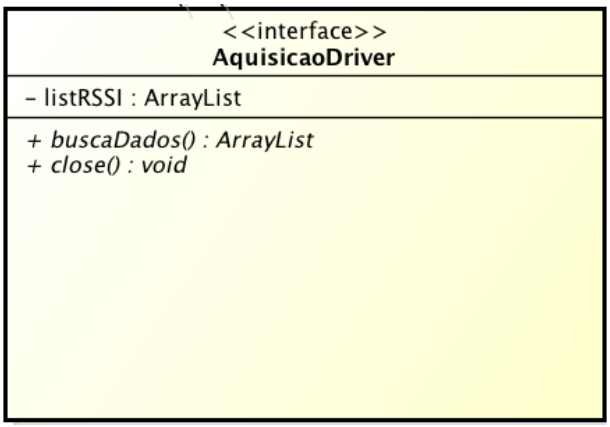



Figura 5.3: Interface AquisicaoDriver

### 5.1.2 Aquisição de Dados

Esta camada é responsável pela coleta dos sinais RSSI dos usuários móveis, detectados num determinado ponto do ambiente. Para prover maior flexibilidade a essa camada foi necessário usar uma interface denominada de *AquisicaoDriver* (Figura 5.3). Esta interface é usada para comunicação com a camada de Dados através dos métodos *buscaDados*() e *close*(). Essas interfaces provêem ao protótipo a utilização de múltiplos protocolos de rede para obtenção do RSSI do usuário.

A implementaçao do método *buscaDados*() utiliza como resultado um *array* de RSSI, como se pode constatar no código do método abaixo, o qual esta escrito para o padrão *Bluetooth*, mas é similar aos padrões *Zigbee* e *Wi-Fi*. Um conjunto de RSSI é um objeto que representa a coleta dos sinais RSSI de quatro PA presentes no ambiente, mais a intensidade de sinal do dispositivo móvel do usuário. O método *close*() serve para encerrar a chamada ao controlador do dispositivo móvel previamente aberta.

```
public ArrayList<Integer> buscaDados() { Bluecover blue = new
  Bluecover(); int numAP = blue.findNumberDevices(); String[ ]
  macDevice = blue.findAll(); for(int x = 0; x < numAP; x++) { String
    mac = macDevice[x]; int RSSI = blue.getRSSI(mac);
    listArray.add(RSSI); map.put(mac,RSSI); } return listArray; }
```

### 5.1.3 Controle de Conflitos

A camada de Controle de Conflitos é responsável por preparar os dados necessários para que o RNA efetue o cálculo da localização. É nesta camada que é implementada a fase de calibração assim como a preparação de um dado proveniente da camada anterior para o RNA, na fase *online*. Dessa forma, a parte de gerência de dados é subdividida em três subcamadas, duas da fase de calibração e uma da fase *online*, que são respectivamente:

- Registro de Dados -responsável pelo registro dos pares de coordenadas($x$ e $y$)/RSSI. Esses pares são obtidos em cada um dos pontos do ambiente, gerando um histórico que é fundamental para a elaboração eficaz do mapa *fingerprinting*;


Figura 5.4: Subcamadas da Camadas de Controle de Conflito


Figura 5.5: Interface LEADriver

- Filtro de Dados - responsável por ler os dados obtidos pelo registro de dados, organizando-os de modo a obter o mapa coordenadas/RSSI do ambiente, atendendo aos requisitos das RNA usado na camada superior. Contém filtros que só consideram as medidas válidas entre os presentes no histórico guardado pela camada anterior. Por exemplo, um filtro válido poderá ser um determinado intervalo de tempo em que as medidas foram coletadas;
- Transformação dos Dados - prepara os valores de entrada para serem usados na camada RNA. Os valores provenientes da camada de Aquisição de Dados são adaptados, efetuando a filtragem das coordenadas úteis, assim como a revalidação e checagem entre as coordenadas detectadas na camada de Aquisição de Dados e os presentes.

A Figura 5.4 apresenta o modelo esquematizado das camadas de controle de conflito. Essa camada ainda é responsável pela comunicação com QoS e validação das respostas de localização obtidas da camada LEA. Sendo assim a camada de Controle de Conflito é uma camada estratégica de suma importância no projeto.

### 5.1.4 LEA

É nessa camada onde a amostra obtida na fase *online*, adaptada pela subcamada de Transformação de Dados, vai ser confrontada com o mapa de sinais, obtido na subcamada de Organização de Dados. A partir do resultado serão obtidas as coordenadas absolutas do local onde se encontra o dispositivo móvel (usuário).

A fim de garantir a fácil integração de novos agentes LEA, definiu-se uma classe abstrata denominada de LEADriver, como registra a Figura 5.5.


Figura 5.6: Interface QoS

Os algoritmos utilizados nesta camada foram determinísticos, com RNA do tipo *Backpropagation* (Seção 3.3.1), sua variação *Backpropagation Momentum* (Seção 3.3.2), e *Leverbenquat-Marquart*(Seção 3.3.3). Assim, possibilitando a comparação de várias implementações da RNA, conforme apresentada na Seção 3.3.

### 5.1.5 QoS

Essa camada é responsável por determinar se as coordenadas retornadas pela camada LEA devem ser aceitas. Essa camada pode contar com diversos níveis de qualidade de serviços sendo integrados ao *core* principal. Para criação de novos níveis de qualidade os agentes devem estender a classe InterfaceQoS, Figura 5.6 e publicar o novo nível de QoS.

Na camada de QoS foram criados dois agentes com as seguintes características:

- MonitorError - esse agente monitora os erros de localização. Nesse agente foi definido como erro aceitável um erro de 1.5m (erro máximo obtido por (Castro & Favela., 2005)). Caso a localização retornada pelos algoritmos de LEA fossem maior que o erro aceitável, esse agente entra em ação e invalida o resultado e pede uma nova medição dos dados RSSI para re-calculo da localização;
- NearAps - esse agente trabalha com a potência do sinal medido, ou seja, segundo (Kaemarungsi, 2005) sinais entre $-40$dB e $-60$dB, com uma variação de 7dB, estão próximos aos PA. Por esse fator adotou-se um erro aceitável de 1.0m. Em casos onde o resultado da localização extrapolasse o erro aceitável o agente pedirá nova aquisição de dados RSSI para novo cálculo de posição.

### 5.1.6 Flexibilidade e Adaptabilidade da Arquitetura

É notório que haja evolução da tecnologia, aparecendo novos tipos de redes sem fios, novos algoritmos de LEA ou necessidade de utilizar algum nível de QoS não utilizado neste trabalho.

No aspecto de protocolos de redes todos os receptores destas redes têm que receber o sinal de radiofreqüência, podendo ser quantificado o valor de RSSI. Para atender as necessidades do surgimento de novos protocolos de rede e integrá-la neste protótipo é necessário que implemente a camada de Aquisição de Dados, conforme apresentado na da Figura 5.4, ou seja, que o seu controlador implemente a interface *AquisicaoDriver* garantindo assim que este modelo seja facilmente estendido.

O protótipo implementado prevê o funcionamento simultâneo com todos os tipos de rede sem fio, pois as subcamadas de Organização de Dados como a de Transformação dos Dados já contemplam a origem do sinal recebido, ou seja, a que rede pertence. Dessa forma, pode ocorrer uma complementaridade entre tecnologias de rede, até mesmo novas tecnologias que possam surgir.

Na questão da adição de novos agentes para tratar algoritmos de LEA criados recentemente e como outros níveis de QoS é ainda mais simples. Somente há a necessidade de estender as classes *LEADriver* e $InterfaceQoS$, respectivamente, e adicionar os novos agentes. As camadas de LEA e QoS podem de perceber os novos agentes e coordenar suas ações com os agentes já existentes.

## 5.2 Serviço de Localização

Para a elaboração do serviço de localização usou-se a linguagem de programação Java juntamente com a *framework* JADE (Seção 3.2). O uso dessas ferramentas se deu porque JADE faz uso da linguagem Java, implementa o padrão FIPA, sendo o JADE um dos *frameworks* mais utilizados no mundo para desenvolvimento de SMA.

Na parte de LEA foi utilizado o programa Matlab versão 6.5 R13, como ferramenta para a definiçao de RNAs. O Matlab tem a possibilidade de integração com Java através de *sockets* ou interface C. Na utilização do Matlab para criação das RNA e elaboração de um MF foi detectado a necessidade de três informações fundamentais:

- a informação referente às coordenadas do ponto do mapa a ser construído;
- a identificação de todas as referências detectadas nesse mesmo ponto e
- o valor de RSSI respectivo para cada referência.

Com relação à QoS na abordagem de SMA foi utilizado toda a estrutura JADE e Java presente, para o desenvolvimento de níveis de QoS de forma simples, rápida e sem impactos na arquitetura. A Figura 5.7 ilustra o ciclo de vida dos agentes bem como a arquitetura utilizada. Note que existem três tipos de agentes inseridos no *LocationAgent*, *RadarAgent*,$ConflictAgent$ e *LEAAgent*. Passaremos a descrever cada agente.

### 5.2.1 *Radar Agent*

Na arquitetura proposta (Figura 5.7) há o agente de controle sobre os agentes que implementam a interface Drive, *RadarAgent*. O *RadarAgent* é um agente do tipo reativo, conforme descrito na Seção 3.1.1. A Tabela 5.1 apresenta suas percepções e ações do *RadarAgent*.

Tabela 5.1: Tabela de Percepçao e Açao do *RadarAgent*

| Percepções | Ações |
|---|---|
| Capturar os sinais de RSSI | Coordenar as buscas por sinais RSSI |
| Novos usuários | Verificar a presença de novos usuários |
| Novos *Radar Agents* | Registrar os agentes que implementam a interface *Drive* |

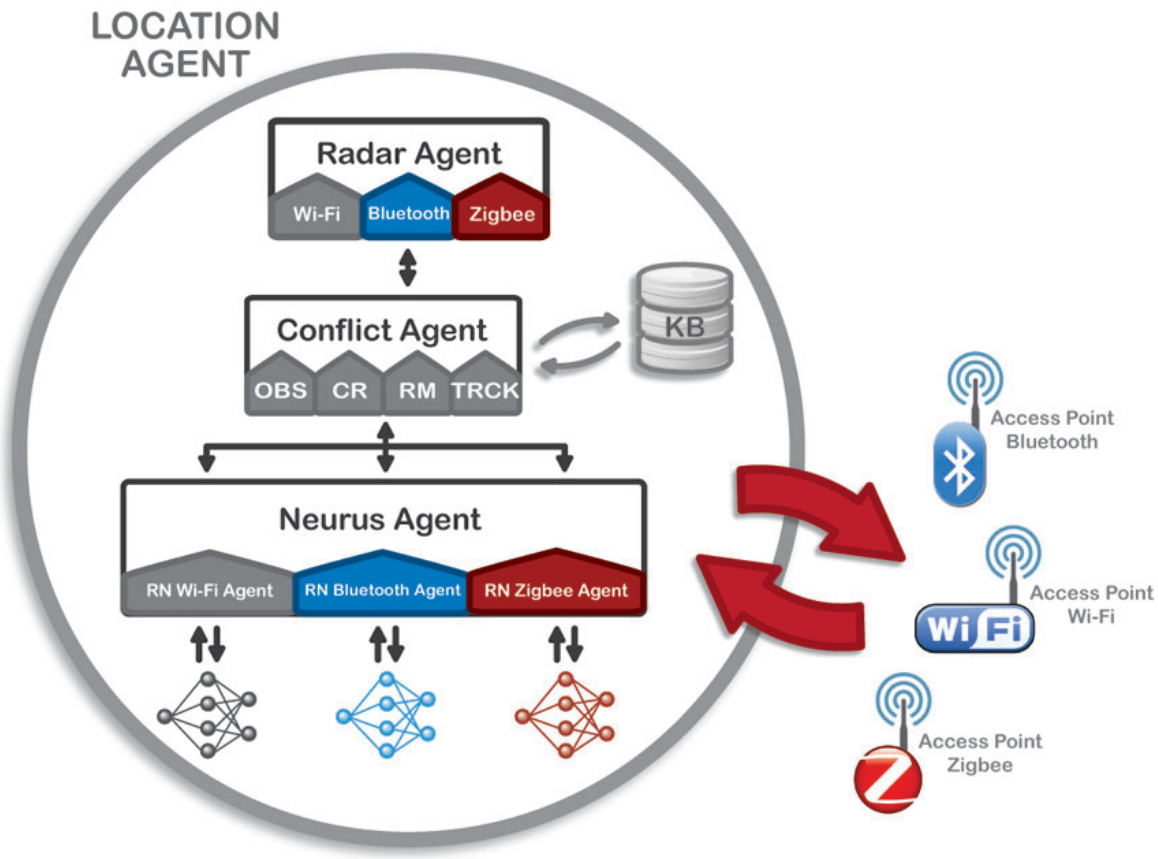



Figura 5.7: Arquitetura do Serviço de Localização

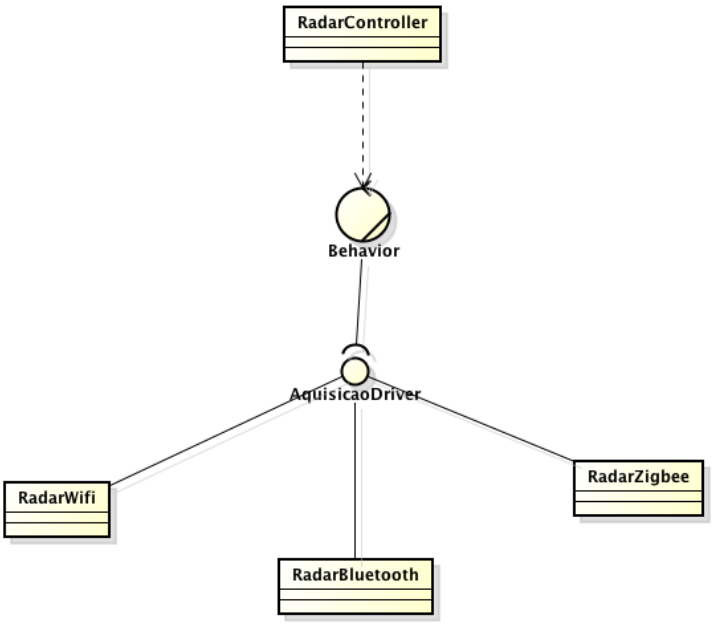



Figura 5.8: Diagrama de Classe do Agente RadarAgent

A Figura 5.8 ilustra o *RadarAgent* com suas respectivas classes que implementam os componentes apresentado na Tabela 5.1. Os agentes *Bluetooth*, *Zigbee* e *WiFi* compõe o *RadarAgente*, herdando suas principais características, ou seja, são agentes do tipo reativo. Porém somente contam com a primeira percepção e ação do *RadarAgent*

Para a aquisição de sinais RSSI usou-se um esquema orientado ao cliente, ou seja, PAs fixos requisitando o RSSI do usuário. Essa técnica é composta por um conjunto de referências e um dispositivo móvel a ser localizado. Neste trabalho, foram usados os seguintes drivers: IEEE 802.15.1 *RadarBluetooth*, IEEE 802.15.4 *RadarZigbee*, IEEE 802.11 *RadarWifi* e o *AllDriver*. Os drives IEEE se tornaram agentes que implementam a interface Driver. Houve também a criação do agente *AllDriver* que implementa a interface *Drive*. Contudo, o agente *AllDriver* é um agente genérico que pode ser utilizado para buscar RSSI de protocolos não implementados ainda ou utilizar os resultados dos agentes existentes.

#### 5.2.1.1 Agente *Bluetooth*

O agente *Bluetooth*, ou *RadarBluetooth* (Figura 5.7) foi desenvolvido em Java para suportar as normas *Java Specification Request* (JSR)-82, assim possibilitando a comunicação com os dispositivos que possuam a tecnologia *Bluetooth* (Thompson et al., 2008; Oracle). Para tanto, foi utilizada a API existente chamada de *Bluecove* (BlueCove), que implementa uma parte substancial das normas definidas pela JSR-82. Com este recurso torna-se possível a comunicação entre a plataforma *Java 2 Platform, Standard Edition* (J2SE) os dispositivos *Bluetooth*.

A configuração da comunicação é do tipo cliente-servidor em que o dispositivo móvel é o cliente e o computador é o servidor. Essa comunicação entre o cliente e o servidor foi

estabelecida considerando a norma JSR-82, onde o dispositivo móvel é um celular ou um *smartphone* que suporte esta norma.

O servidor utiliza a $APIBluecove$. Este servidor *Bluetooth* aguarda que um cliente entre em sua área de alcance, para que sejam coletadas as informações de endereço MAC ($getId()$) e RSSI ($getRSSI()$).

#### 5.2.1.2 Agente *Zigbee*

Para o agente Zigbee, ou $RadarZigbee$ (Figura 5.7) funcionar de maneira adequada, foram posicionado os módulos de forma estática no ambiente. Nesse trabalho foram usados os emissores de IEEE 802.15.4 concebidos para o efeito que consistem num *Programable Integrated Circuit* (PIC), o 18LF2620, um módulo XBee 802.15.4 OEM RF Module (Digi, 2006-2008).

Nesse protocolo o dispositivo móvel deve ser entendido como um $sniffer$ programado de modo a receber os pacotes enviados pelos emissores, retirando a informação sobre o valor de RSSI assim como uma identificação do emissor.

O $sniffer$ pode ser um módulo $XBeePro$ para testes ligados por *Universal Serial Bus* (USB) ao computador, apresentado na Figura 5.9. Esse $sniffer$ funciona somente no sistema operacional Windows.

#### 5.2.1.3 Agente *Wi-Fi*

O agente *Wi-Fi* ou $RadarWifi$ como apresentado na Figura 5.7, busca o RSSI do protocolo de rede IEEE 802.11. Nesse trabalho foi utilizado a placa de rede a $AirPortExtreme$ (802.11a/b/g/n). Para suporte a implementação da interface Driver foram desenvolvidas duas aplicações, sendo uma aplicação bat para ser executada em ambiente Windows e um shell para Linux:

- Na plataforma Windows foi usado o comando *netsh* (Microsoft, a). Para obter as informações de RSSI usadas pelos agentes de Localização. O comando executado foi $netshshownetworksmode = bssid$ (Microsoft, b), onde o parâmetro *shownetworks* serve para listar todas as redes IEEE802.11 no alcance da estação com o serviço de localização. Ativado o parâmetro *mode* serve para selecionar as informações. Há dois modos de visualizar as informações:
  - *Service Set Identifier* (SSID) - mostra as informações básicas da rede, como por exemplo, o SSID, o tipo de encriptação, o tipo de Rede, ocultando as informações referentes aos PAs que a difundem;
  - *Basic Service Set Identifier* (BSSID) - mostra, além das informaçoes descritas anteriormente, as informaçoes dos PAs detectados que estão utilizando a rede como o MAC do PA, o valor de RSSI entre outros.

  Por ser mais completo preferiu o uso do modo *Mutiple Service Set Identifier* (MSSID). A Figura 5.10 apresenta a saída do comando *netsh*. Para obter outras informações talvez necessárias para outros serviços que dependam do serviço de localização, elaborou-se um filtro de modo a obter o valor correspondente ao endereço MAC (identificação desta fonte de potência), definido como BSSID; e o valor de RSSI definido, sendo dado em percentagem.


Figura 5.9: Componentes Zigbee

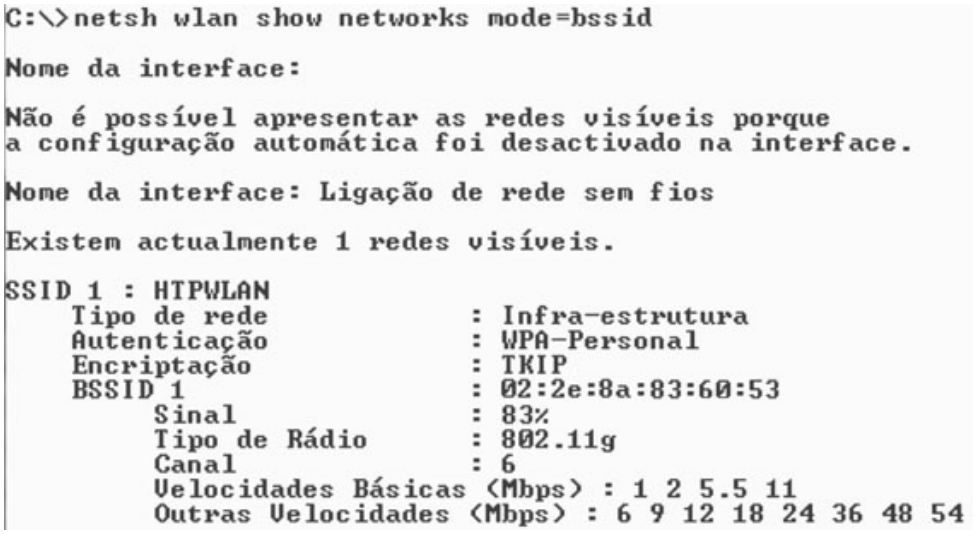



Figura 5.10: Retorno do comando netsh wlan no Windows

- Para a aquisição de dados no ambiente Linux foi usado o comando de Linux/Unix *iwlist* para obter informações sobre as redes disponíveis e dispositivos móveis. Este comando pode ser utilizado com diversos parâmetros, para obter as informações desejadas. Neste trabalho, executou-se com os seguintes parâmetros:

  – scan - serve para listar as várias informações relativas a todos os PAs que estão ao alcance e

  – eth1 - serve para selecionar a interface de rede pretendida.

  A saída deste comando pode ser observada na Figura 5.11. Para obter os valores necessários foi criado um filtro, que filtrou as linhas correspondentes ao endereço MAC (identificação desta fonte de potência) definida como *Address* e o correspondente a *Signal Level* (valor de RSSI).

#### 5.2.1.4 Agente AllDrivers

Como descrito na Seção 5.1.2, pode-se implementar uma class cuja interface seja um Driver, no entanto esta não capta dados apenas de uma tecnologia, mas de um conjunto delas (no caso deste projeto *Bluetooth*, *Zigbee* e *Wi-Fi*). Atendendo à versatilidade da interface Driver, pode-se construir uma classe que a implemente, e nela instanciar outras classes, que implementem esta mesma interface, pois o método *buscaDados*() retorna uma lista, que pode ser constituída pela concatenação de todas as listas devolvidas. A Figura 5.12 apresenta o funciomaneto do *AgenteAllDrivers*.

### 5.2.2 Conflict Agent

Este agente é o principal agente do serviço de localização de dispositivos em ambientes fechados. O *ConflictAgent* é um agente do tipo cognitivo, uma vez que o conceitos da

```
esn@Mack:~$ iwlist eth1 scan
eth1      Scan completed :
          Cell 01 - Address: 02:2E:8A:83:60:53
                    ESSID:"WLANESN"
                    Protocol:IEEE 802.11bg
                    Mode:Master
                    Channel:6
                    Frequency:2.437 GHz (Channel 6)
                    Encryption key:on
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 9 Mb/s; 11 Mb/s
                              6 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s; 36 Mb/s
                              48 Mb/s; 54 Mb/s
                    Quality=94/100  Signal level=-48 dBm  Noise level=-48 dBm
                    IE: WPA Version 1
                        Group Cipher : WEP-40
                        Pairwise Ciphers (1) : WEP-40
                        Authentication Suites (1) : PSK
                    Extra: Last beacon: 35ms ago
```

Figura 5.11: Retorno do comando iwlist eth1 scan no Mac OS

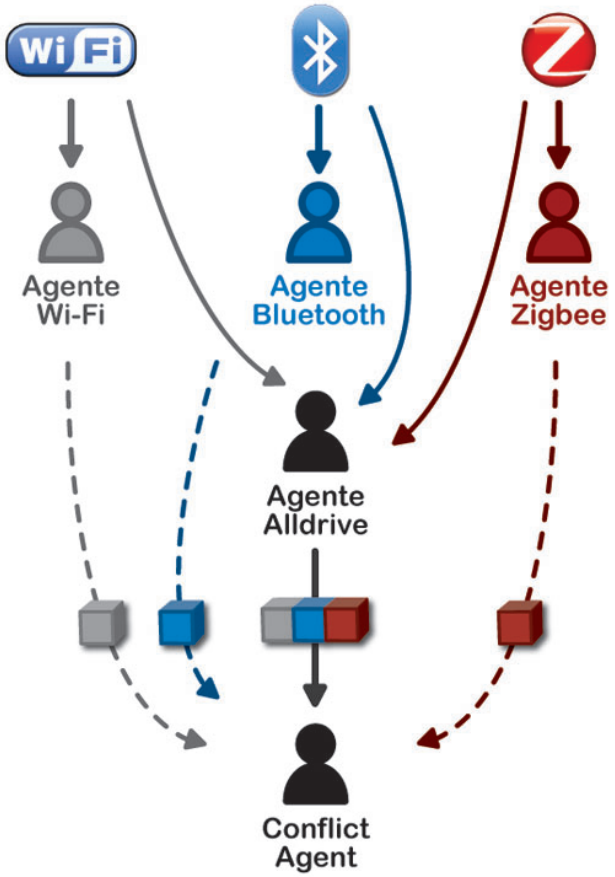



Figura 5.12: Funcionamento do Agente Alldrive

Tabela 5.2: Tabela de Percepção e Ação do *ConflictAgent*.

| Percepções | Ações |
|---|---|
| Dados coletados pelo *Radar Agent* | Chamada controle da aquisição de dados |
| Filtro dos dados coletados pelo *Radar Agent* | Controle da aquisição de dados |
| Obter a localização | Chamada dos agentes de RNA |
| Validação da posição absoluta a ser informada | Chamada dos módulos de QoS |

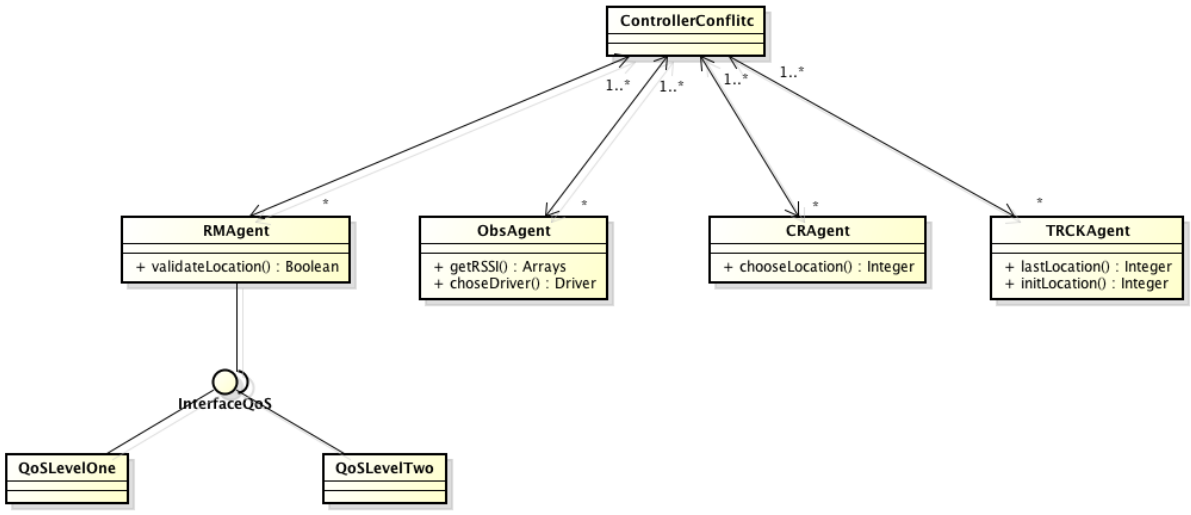



Figura 5.13: Diagrama do *ClonflictAgent*

Seção 3.1.1, a Tabela 5.2 apresenta suas percepções e ações . Já a Figura 5.13 ilustra o diagrama de classes do agente *ConflictAgent* (Figura 5.7).

O *ConflictAgent* foi implementado de modo a tornar possível a comparação entre os diversos algoritmos de RNA projetados para os diferentes protocolos, para que fosse possível uma análise estatística posterior, possibilitando estabeler níveis de QoS dos sinais captados.

O *ConflictAgent* controla todo o curso das atividades desenvolvidas no processo de descoberta de usuário móvel até a descoberta da localização do mesmo. Dessa forma, é um agente orientado a objetivos, com acesso as tarefas básicas de descoberta de conhecimento a serem executadas, disponíveis na base de conhecimento ou *Knowledge Base* (KB). Este agente é capaz também de negociar, a partir dos resultados das tarefas básicas, novas tarefas para atingir seus objetivos.

A Figura 5.13 apresenta o diagrama básico das classes do *ConflictAgent*, nota-se que esse agente possui duas metas não funcionais:

- encontrar usuários móveis e obter o RSSI, sendo essa meta decomposta em duas submetas (verificar o ambiente e obter o RSSI do dispositivo móvel), essa tarefa é realizada através de negociações com a camada de Aquisição de dados, (Figura 5.14).
- obter a localização do usuário, essa meta é decomposta em três submetas apresentados na Figura 5.14, onde os agentes são representados por círculos, as metas por

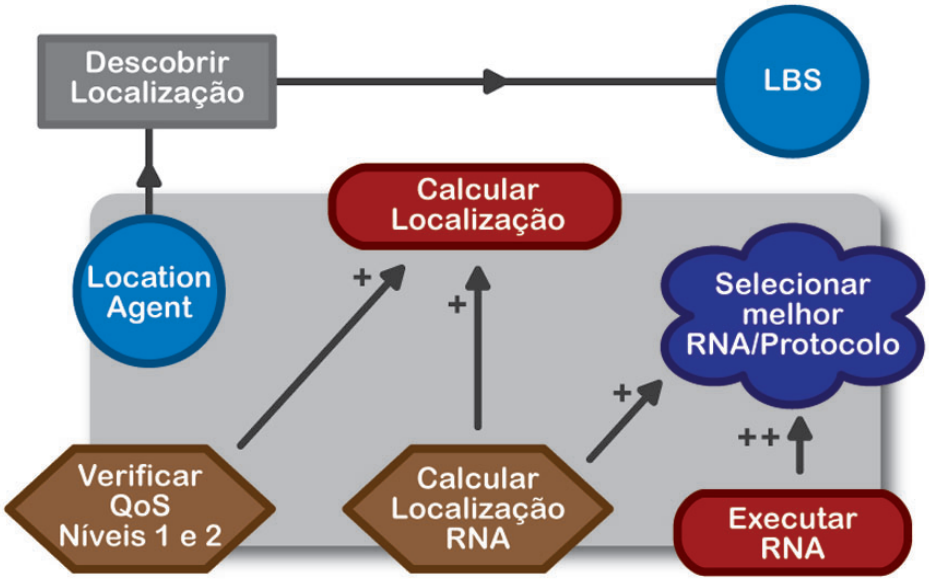



Figura 5.14: Diagrama das submetas do *ConflictAgent*

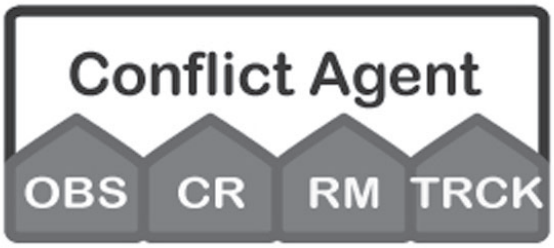



Figura 5.15: Módulos do Conflict Agent

retângulos ovais, as metas fracas por nuvens, as tarefas por hexágonos e os recursos por retângulos, conforme representação de (Giorgini et al., 2005).

- verificar QoS;
- executar RNA e
- calcular localização RNA.

O *ConflictAgent* ainda é composto por quatro módulos, apresentado na Figura 5.15. As características desses módulos são:

- Módulo de Observação ou *Observation* (OBS) - é responsável pela requisição do RSSI, negociando diretamente com a camada de Aquisição de Dados, sendo que este modulo pode selecionar qual protocolo de rede irá obter os dados RSSI.
- Módulo de Resolução de Conflito ou *Conflict Resolution* (CR) - responsável por decidir qual posição ira usar, através do acesso ao modulo Controle de Conhecimento.

O CR pode fazer novas requisições de RSSI ao modulo de Observação caso haja conflito na resposta da posição para mais de um protocolo de rede utilizado.

- Módulo Controle de Conhecimento ou *Knowledge Managed-based* (RM) - responsável pelas regras de inferência (*If-Then*), ou seja, este modulo faz o controle de qualidade da resposta obtida do *LEAAgent*. Nesse trabalho foram definidas duas regras básicas:
  - aceitar as coordenadas com erros menores que 1,5m (erro originário do trabalho de (Castro & Favela., 2005)) e
  - caso o sinal obtido esteja próximo de um PA o erro não poderá ser maior que 1.0m, definição baseada no trabalho de (Ekahau).
- Módulo Tracking ou *Tracking* (TRCK) - responsável pelo monitoramento do usuário. O TRCK colabora com o modulo de RM, informando a movimentação do usuário e evitando deslocamentos abruptos nas salas. Este módulo é apenas uma uma interface, o qual deverá ser implementado futuramente. O TRCK deverá considerar características do usuário, como por exemplo a idade. Este módulo proverá o RM com informações de contexto do usuário portador do dispositivo móvel.

### 5.2.3 O Agente LEA ou *LEA Agent*

*LEAAgent* conta com três agentes de LEA, os quais utilizam os seguintes algoritmos, conforme apresendado na Seção 3.3: *Backpropagation*, *Backpropagation Momentun* e *Levenberg-Marquardt*. O *LEAAgent* é uma agente do tipo cognitivo conforme Seçao 3.1.1, pois utiliza raciocínio baseado no cálculo da RNA a Tabela 5.3 apresenta suas percepçoes e açoes.

Tabela 5.3: Tabela de Percepçao e Açao do *LEAAgent*

| Percepções | Ações |
|---|---|
| Dados adquiridos pelo *Conflict Agent* | Adquirir o conjunto de pares RSSI |
| Comunicação externa | Estabelecer uma comunicação com o *MatLab* |
| Delegar tarefa a agente RNA | Selecionar RNA |
| Suprir necessidades dos agentes RNA | Enviar conjunto de pares RSSI |
| Dados submetidos ao *Conflict Agent* | Responder com par (X,Y) |

Os três agentes LEA fazem uso do *MatLab* para implementar os algoritmos. A Figura 5.16 apresenta o diagrama de classes com os principais atributos e relacionamentos dessa camada.

A comunicação feita entre os agentes LEA e o aplicativo *MatLab* foi desenvolvida em C através de *socket*, conforme apresentado no código abaixo.

```
/* Funcao para conectar no host */ int abrirConexao(char *
servidorConexao, int portaConexao) { #ifdef WIN32 WSADATA wsaData;
  WORD wVersionRequested; int err;

        wVersionRequested = MAKEWORD(2, 0); err =
```

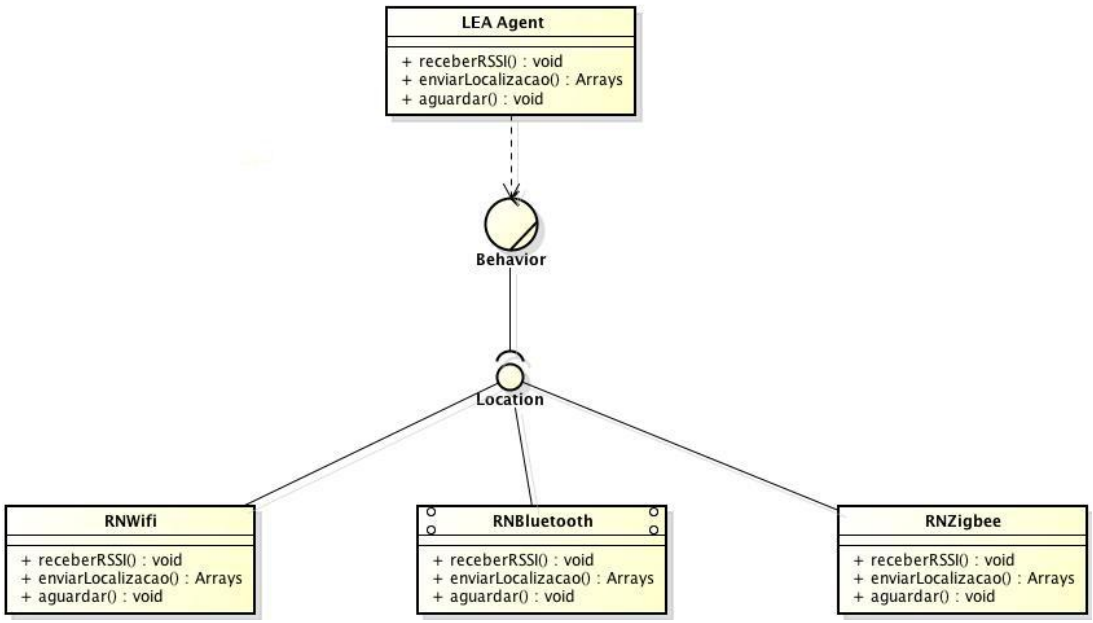



Figura 5.16: Diagrama de Classe do Agente LEA Agent

```
    WSAStartup(wVersionRequested, &wsaData); if (err != 0) {
      fprintf(stderr, "Nao achou WinSock DLL.n"); exit(-1); }
    #endif

  int meuSocket; struct sockaddr_in sockAddr; struct hostent *hEnt;

  /* Vamos obter os dados do host */ hEnt =
  gethostbyname(servidorConexao); if (hEnt == NULL) { printf("Erro
    ao obter os dados do host.n"); exit(-1); }

  /* Agora vamos obter o socket */ meuSocket = socket(AF_INET,
  SOCK_STREAM, getprotobyname("tcp")->p_proto); if (meuSocket == -1)
  { printf("Erro ao obter socket tcp.n"); exit(-1); }

  /* Vamos definir o TCP_NODELAY (usado para comunicacoes de ida e
  volta para melhorar a performance) */ int flagTcpNoDelay = 1; int
  resTcpNoDelay = setsockopt(meuSocket, IPPROTO_TCP, TCP_NODELAY,
  (char *) &flagTcpNoDelay, sizeof(int)); if (resTcpNoDelay < 0) {
    printf("Erro ao setar tcp_nodelay.n"); exit(-1); }

  /* Vamos definir tambem o timeout do socket (Voce pode precisar
  mudar em redes lentas) */ struct timeval tv; int timeouts = 0;
  tv.tv_sec = 3; tv.tv_usec = 0; if (setsockopt(meuSocket,
SOL_SOCKET, SO_RCVTIMEO, (char *) &tv, sizeof tv)) { printf("Erro ao
definir timeout."); exit(-1); }
```

```
   /* Entao conectamos */ memcpy(&sockAddr.sin_addr, hEnt->h_addr,
hEnt->h_length); sockAddr.sin_family = AF_INET; sockAddr.sin_port =
htons(portaConexao); if (connect(meuSocket, (struct sockaddr *)
&sockAddr, sizeof(sockAddr)) < 0) { printf("Erro ao conectar no
servidor.n"); exit(-1); }

   /* E voltamos o socket conectado */ return meuSocket; }
```

O *AgenteLEA* foi parametrizado para que fosse possível a escolha do protocolo de rede e algoritmo RNA. Através desse conjunto de parâmetro serão retornadas as coordenadas absolutas da localização do dispositivo móvel. Esses parâmetros passados para o *AgenteLEA* ficam a critério da Camada Estratégica, uma vez que esta é responsável por critérios de qualidade do serviço e de comunicação com a Camada de Aquisição de Dados. Dessa forma, é possível comparar os níveis de erro das diferentes combinações de Algoritmos de RNA/Protocolo de Rede.

# Capítulo 6

# Experimentos Realizados

Neste capítulo serão apresentados os experimentos realizados desde a criação das RNA, até a integração com o protótipo proposto. O objetivo é relatar a evolução da pesquisa através dos diversos experimentos realizados. Dessa forma, os resultados iniciais apresentados pelas redes neurais foram comparados com o protótipo que utiliza abordagem de SMA e aspectos de QoS. O protótipo desenvolvido apresentou resultados superiores nos aspectos de localização dos usuários móveis e da qualidade no serviço proposto.

Uma vez implementadas as ferramentas necessárias para criação das redes neurais com abordagem de prototipagem do sistema de suporte a LBSs, foram realizados os testes necessários para validar sua viabilidade. Os testes foram efetuados em dois níveis avaliados em precisão e exatidão:

- Nível do Algoritmo de Rede Neural - consiste em analisar o desempenho dos diferentes LEA, alterando seus parâmetros de entrada ou informaçoes exteriores tentando sempre maximizá os resultados;
- Nível do Protótipo - tem como meta verificar a utilização das RNAs criadas aplicando as regras de qualidade adotadas (QoS).

## 6.1 Criação de Mapas de *Fingerprinting*

Para a metodologia de análise do mapa de *fingerprinting* foi crucial implementar um algoritmo de localização eficaz. Para a elaboração deste mapa é necessário definir um cenário, selecionar os pontos que estarão presentes no mapa, assim como os RSSI. Neste trabalho, foi utilizado o prédio do SG11 ligado ao Departamento de Engenharia Elétrica (ENE) da Universidade de Brasília (UnB). A Figura 6.1 representa esquematicamente esse prédio com as três salas utilizadas nos experimentos.

O conjunto de pontos que servirão de suporte para o mapa, considerando como eixo referencial um sistema de eixos ortogonais ($X$,$Y$), de origem na entrada da sala de reunião (GPDS) (0,0) e cujo sentido do eixo das abscissas é Norte-Sul e o das ordenadas Leste-Oeste. O espaçamento entre os diferentes pontos foi de 0.4m resultando em 158 pontos de medição.

O procedimento usado foi o anteriormente descrito para a metodologia de *fingerprinting*. A quantidade de amostras retiradas por ponto foi de 20. Para garantir a qualidade

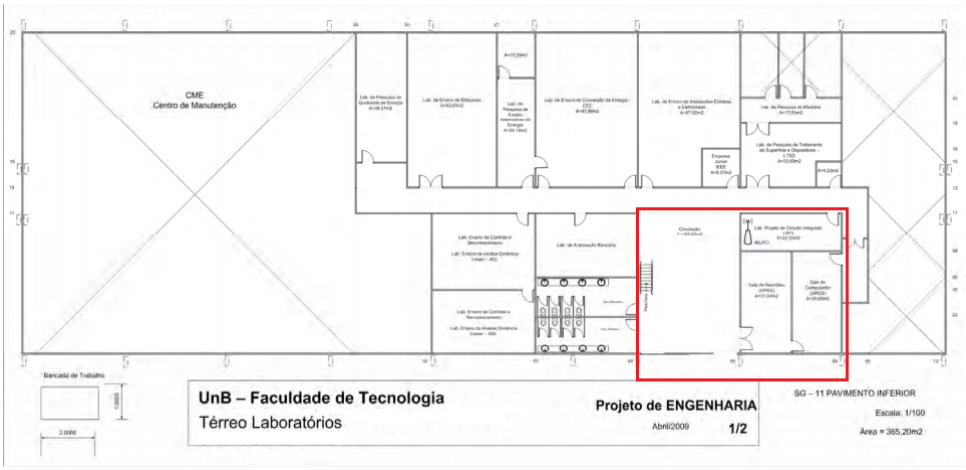



Figura 6.1: Planta do SG11 Engenharia Elétrica da Universidade de Brasília

de rede e eliminar possíveis fontes de erro externas, os PAs usados neste trabalho foram utilizados unicamente para o propósito de coleta dos dados RSSI.

Para coleta dos RSSI do protocolo 802.15.1 (*Bluetooth* foram utilizados *donkle* da marca *Bluetooth Dongle integrated module wireless USB BT Transceiver* da emprese *Hewlett-Packard Company*.Para captura dos sinais para o protocolo 802.15.4 (*Zigbee*) foram utilizados os módulos *XBee* da empresa *MaxStream*, sem a necessidade de criação de uma rede para tanto., assim como o protocolo 802.15.1 não houve a necessidade da criação de uma rede especifica para essa coleta.

Para aquisição dos dados para o protocolo de rede 802.11 (*Wi-Fi*) foi criada a rede com identificador "Teste802.11", sendo esta uma rede aberta. Os PAs utilizados foram os modems roteador *Ethernet* ADSL externo da *D-Link*. A Figura 6.1 ilustra a disposição dos PAs do *Bluetooth*, *Zigbee* e *Wi-Fi*.

## 6.2 Método de avaliaçao dos Mapas de *Fingerprinting*

Uma vez analisados e discutidos como foi criado o MF usados neste trabalho, analisaremos agora os diferentes algoritmos de RNA definido e implementados tendo considerado a sua exatidão e precisão. Como referido na Seção 3.3. Os algoritmos implementados são do tipo probabilístico, sendo todas redes Acíclicas. Os três algoritmos têm as seguintes características:

- *Backpropagation*: Também chamado de retropropagação, pois consiste em um modelo de aprendizado supervisionado de múltiplas camadas, cujo erro obtido na saída é transferido às camadas escondidas;
- *Backpropagation Momentum*: introduz termo $\mu$ na equação da *Backpropagation*, para acelerar convergência. Essa é uma maneira de aumentar a taxa de aprendizado sem levar à oscilação a modificar a regra delta generalizado para incluir o termo *momentum*, uma constante que determina o efeito das mudanças passadas dos pesos na direção atual do movimento no espaço de pesos;

- *Levenberg-Marquardt*: Este modelo caracteriza-se como um dos métodos mais rápidos para o treinamento de Redes Neurais Acíclicas de tamanho moderado, além disso, é eficaz na minimização de funções com um pequeno número de parâmetros.

Neste trabalho foi definido empiricamente que todos os testes teriam o número de épocas de 10.000. Para garantir o sucesso do aprendizado foi usada uma velocidade de aprendizado baixa de 0.1. As redes neurais criadas contam com três camadas, uma de entrada, uma escondida e uma de saída. A camada de entrada irá receber o conjunto de cinco RSSI e a camada de saída retornará as coordenadas $X$ e $Y$. Para teste variou-se o número de neurônios da camada escondida, onde ela começará com cinco e será incrementando de cinco em cinco até completar vinte. Depois o incremento será de dez em dez até completar sessenta e finalmente o incremento será de vinte em vinte até completar cem neurônios na camada escondida. Dessa forma, teremos como resultado dez redes neurais.

As condições de teste dos algoritmos foram realizados com os mesmo números de pontos e de forma idêntica na criação do MF. Os parâmetros a analisar foram o valor da exatidão, precisão, desempenho e regressão do algoritmo. Considerando um ponto de coordenadas ($X$,$Y$), para obter o valor da exatidão calcula-se para cada ponto o valor da distância Euclidiana entre o ponto obtido e o ponto esperado, $\rho_i$, como se pode ver na Equação 6.1, em que o ponto obtido é definido pelas coordenadas espaciais ($X_{obtido}$, $Y_{obtido}$) e o esperado pelas ($X_{esperado}$, $Y_{esperado}$), sendo $i$ o registro a considerar. Os valores espaciais são dados em metros.

$$\rho_i = \sqrt{(X_{obtido} - X_{esperado})^2 + (Y_{obtido} - Y_{esperado})^2} \tag{6.1}$$

Tendo $\rho_i$, o valor do erro médio para o ponto $p$, definida por $\rho_p$, em metros, vai ser dada por, onde $n$ são as vinte amostras por ponto conforme apresentado na Equação 6.2.

$$\rho_p = \frac{1}{n} \sum_{i \in p}^{n} \rho_i \tag{6.2}$$

Calculando o valor de $\rho_p$ para todos os pontos da analise, ($p$), sendo $m$ é o total de pontos. O valor da exatidão será o da Equação 6.3.

$$\rho = \frac{1}{m} \sum_{p \in p}^{n} \rho_p \tag{6.3}$$

Para a obtenção da precisão ($\delta$) efetua-se o cálculo do desvio padrão das amostras $\rho_i$ agrupadas pelo ponto esperado, ($\delta_p$), em metros, sendo seu valor dado pela Equação 6.4.

$$\delta = \frac{1}{m} \sum_{p \in p}^{n} (1 - \frac{\delta_p}{\rho_p}) * 100 \tag{6.4}$$

Já os resultados desempenho do algoritmo e regressão do algoritmo são dados por gráficos apresentados no *Matlab*. O resultado do desempenho da rede neural foi tomado como fator de escolha, ou seja, a rede neural que mais se aproximasse do objetivo (0) será escolhida.

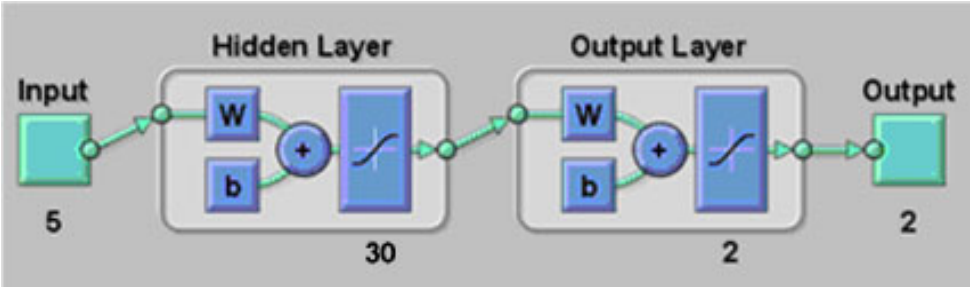



Figura 6.2: Arquitetura de Rede Neural *Backpropagation Wi-Fi*

## 6.3 Experimento 1 - Algoritmo *Backpropagation*

Seguindo os passos descritos na Seções 6.2, foram testadas dez redes neurais que implementaram o algoritmo *Backpropagation*. Em (de M. Neves et al., 2010) foram apresentados resultados dos experimentos dessa rede neural com os protocolos *Wi-Fi* e *Zigbee*. Já em (Fonseca et al., 2011) foram apresentados os resultados do uso da *Backpropagation* com uso dos três protocolos de rede e os nivéis de QoS.

A rede que apresentou melhor resultado foi a que utilizou os sinais oriundos do protocolo *Wi-Fi*, conforme apresentado no gráfico de desempenho da Figura 6.3c (0,0055). A RNA contou com trinta neurônios na camada escondida, conforme apresentado na Figura 6.2.

Em segundo lugar de melhor performace tivemos a rede que recebeu dados do protocolo *Zigbee* com quinze neurônios na camada escondia (0,0064). A Figura 6.3b apresenta o gráfico de desempenho.

Em terceiro lugar, a rede que foi treinada com os dados do protocolo *Bluetooth* que teve vinte neurônios na camada escondida. A Figura 6.3a apresenta o gráfico de desempenho (0,0068).

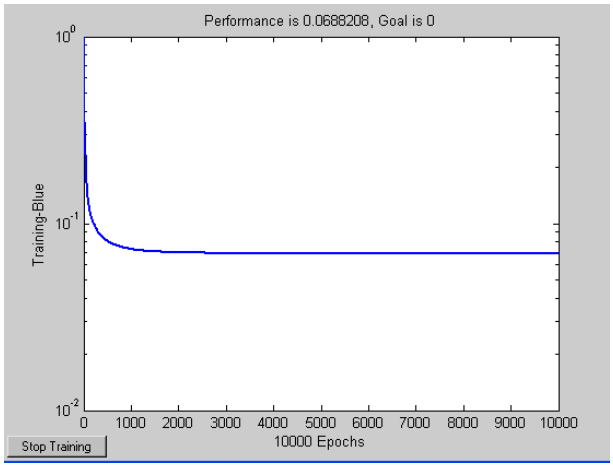
A Tabela 6.1 mostra os resultados dos testes de desempenho apresentados para todas as redes criadas por cada protocolo de rede sem-fio.

Tabela 6.1: Tabela de Resultados de Desempenho *Backpropagation*

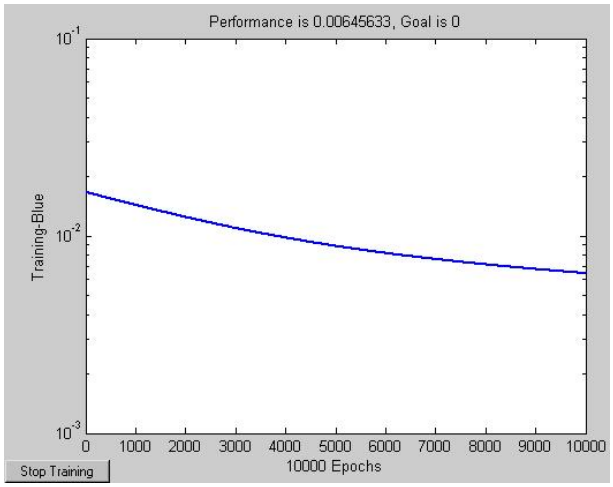
| Desempenho | Neurônios na Camada Escondida | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 5 | 10 | 15 | 20 | 30 | 40 | 50 | 60 | 80 | 100 |
| Wi-Fi | 0,94 | 0,89 | 0,65 | 0,07 | ***0,0055*** | 0,97 | 0,75 | 0,08 | 0,56 | 0,32 |
| Zigbee | 0,7 | 0,008 | ***0,0064*** | 0,58 | 0,87 | 0,098 | 0,078 | 0,69 | 0,95 | 0,081 |
| Bluetooth | 0,9 | 0,087 | 0,09 | ***0,085*** | 0,25 | 0,14 | 0,087 | 0,092 | 0,32 | 0,099 |

No quesito regressão as três melhores redes obtiveram resultados bastante próximos, chegarando muito próximo do objetivo de 1.0. As Figuras 6.4a, 6.4b e 6.4c mostram os gráficos de regressão das três melhores redes neurais para *Bluetooth* (0,985), *Zigbee* (0,9844) e *Wi-Fi* (0,9859) respectivamente.

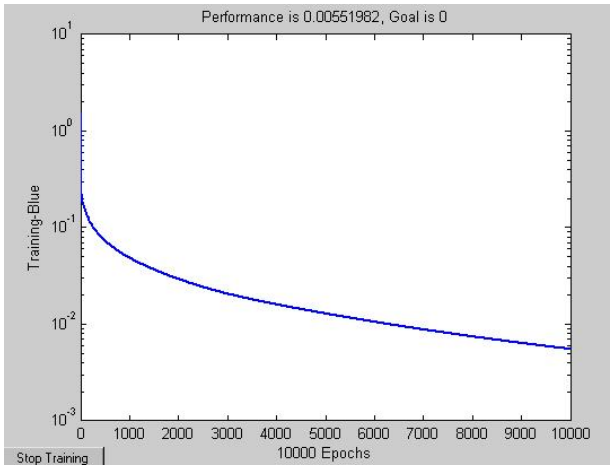
Em relação às saídas obtidas com a utilização das três melhores redes neurais, notou-se que os resultados da rede neural treinada com os dados RSSI do *Wi-Fi* foram superiores aos demais. A Tabela 6.2 mostra o comparativo dos três protocolos de rede sem-fio com relação à precisão e a distância.




(a)

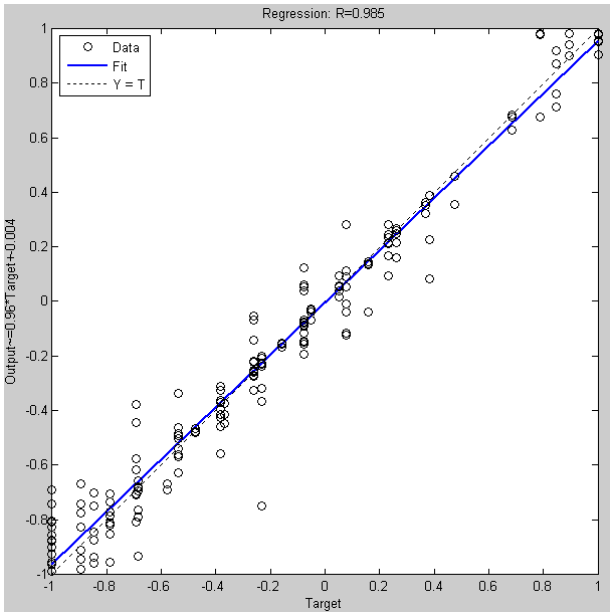



(b)

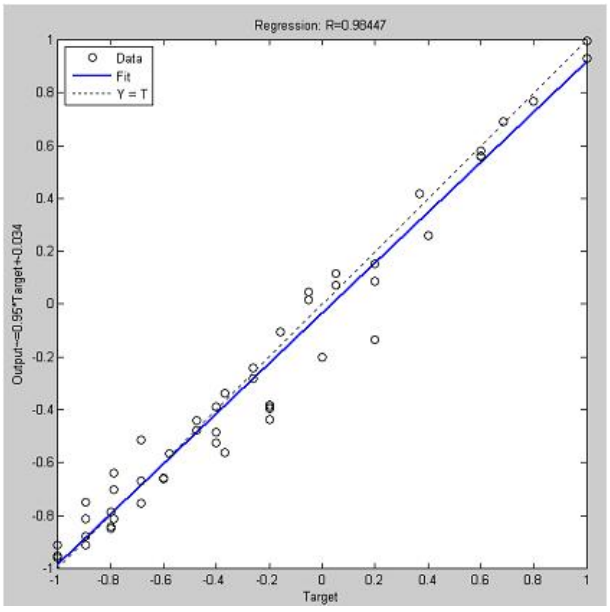



(c)

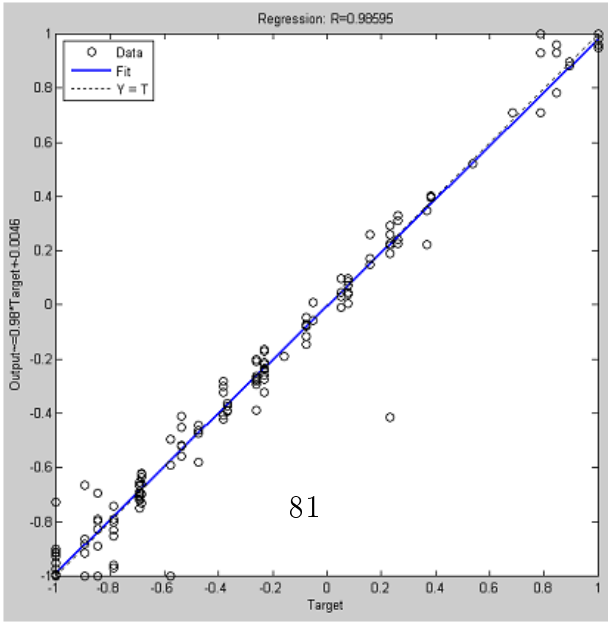
Figura 6.3: Performance das melhores *Backpropagation* (a)*Bluetooth*, (b) *Zigbee* e (c)*Wi-Fi*


Regression: R=0.985
Data
Fit
Y = T
Output~=0.96*Target+0.004
Target

(a)


Regression: R=0.98447
Data
Fit
Y = T
Output~=0.95*Target+0.034
Target

(b)


Regression: R=0.98595
Data
Fit
Y = T
Output~=0.98*Target+0.0046
Target



(c)

Tabela 6.2: Tabela comparativa de precisão e distâncias das melhores RNA *Backpropagation*

| | Precisão (%) | Distância (m) |
|---|---|---|
| Wi-Fi | 73 | 0,42 |
| Zigbee | 67 | 2,03 |
| Bluetooth | 63 | 3,35 |

A titulo de exemplificação a Tabela 6.3 mostra para cada protocolo de rede analisado as melhores e piores coordenadas, considerando o alvo e o erro.

Tabela 6.3: Tabela comparativa de coordenadas das melhores RNA *Backpropagation*

| | Melhor Coordenada | Alvo | Erro (m) | Pior Coordenada | Alvo | Erro (m) |
|---|---|---|---|---|---|---|
| Wi-Fi | (0,45;0,40) | (0,4;0,4) | 0,05 | (1,09;1,33) | (0,8;2,8) | 1,5 |
| Zigbee | (0,09;0) | (0;0,4) | 0,41 | (3,16;3,2) | (3,2;7,6) | 4,4 |
| Bluetooth | (0,4;0,39) | (0,4;0,4) | 0,05 | (2,4;6,4) | (2,8;0,8) | 6,83 |

A Figura 6.5 apresenta o gráfico comparativo das melhores redes neurais de cada protocolo de rede, sobre a perspectiva do erro euclidiano para uma amostra de teste de vinte e sete pontos. Neste gráfico fica claro a superioridade em relação à qualidade do resultado da rede treinada a partir dos dados providos pelo protocolo *Wi-Fi*.

## 6.4 Experimento 2 - Algoritmo *Backpropagation Momentum*

Os testes do algoritmo *Backpropagation Momentum* seguindo a descrição da Seção 6.2, foram realizados com dez redes neurais que implementaram o algoritmo *Backpropagation Momentum*. Contudo esse algoritmo apresenta uma variação que é o momento (termo $\mu$), essa variável foi definida para cada rede em 0.1, 0.3 e 0.5. Os valores escolhidos para $\mu$ foram empíricos. porém não foi elevado o valor para que não perdesse a capacidade de generalização, ou seja, *overtraining*.

Tal como o algoritmo *Backpropagation* a rede que apresentou melhor resultado foi que utilizou os sinais do protocolo *Wi-Fi*, contendo vinte neurônios na camada escondida. Em segundo lugar de desempenho a rede que recebeu dados do protocolo *Zigbee* com quarenta neurônios na camada escondia. A rede neural que apresentou o pior resultado foi treinada com os dados do protocolo *Bluetooth* com sessenta neurônios na camada escondida. As Figuras 6.6a, 6.6b e 6.6c apresentam os melhores desempenho das redes neurais criadas para os protocolos *Bluetooth* $(0,0085)$, *Zigbee* $(0,0061)$ e *Wi-Fi* $(0,0059)$ respectivamente.

A Tabela 6.4 mostra os resultados dos testes de desempenho apresentados para todas as redes criadas por cada protocolo de rede sem-fio.

Os resultados de regressão das três melhores redes obtiveram resultados bastante próximos. Aonde todos chegaram muito próximo do objetivo de 1.0. As Figuras 6.7a, 6.7b e

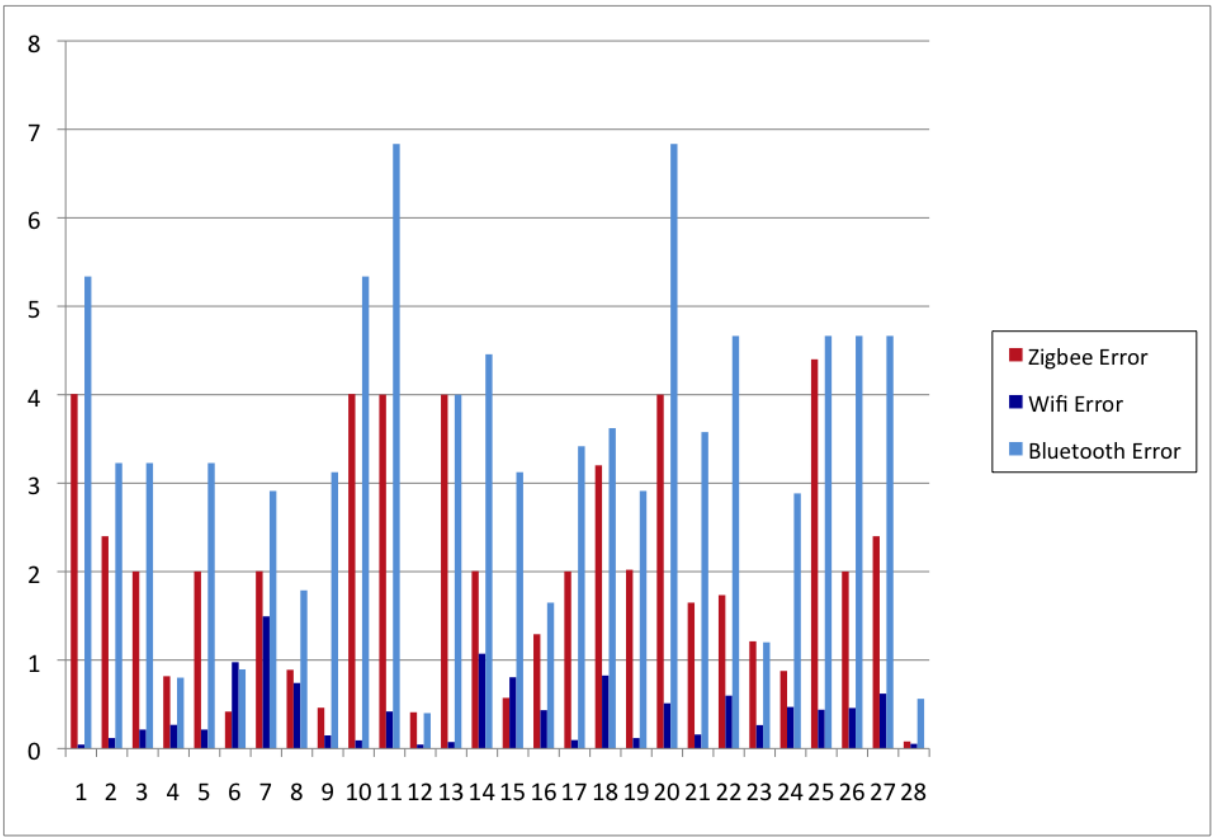
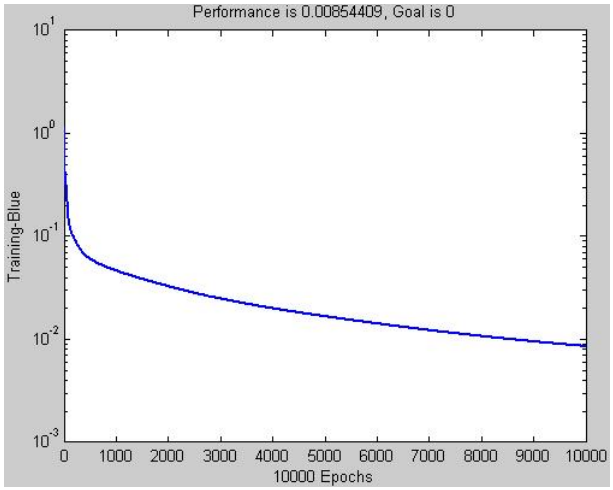
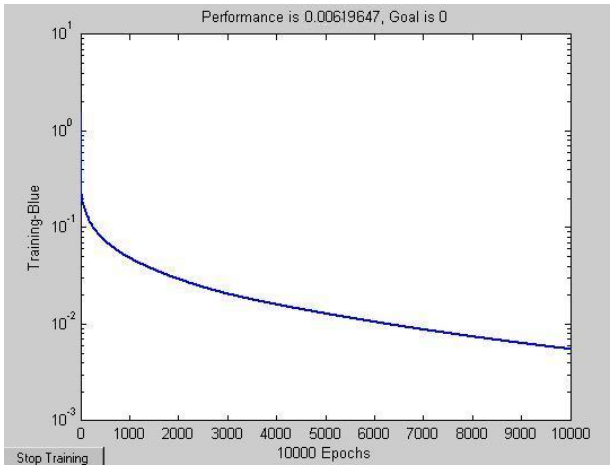


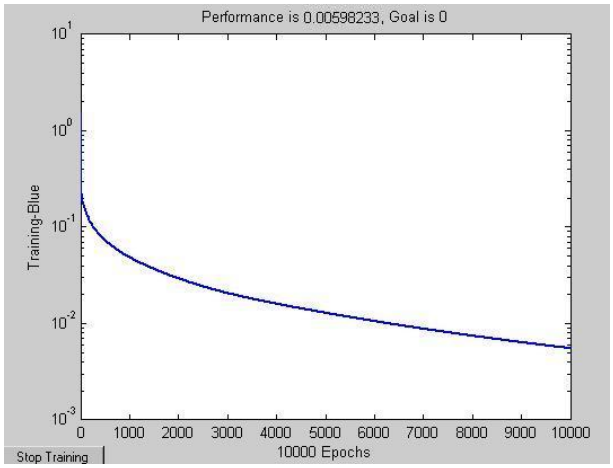

Figura 6.5: Gráfico comparativo das melhores redes neurais




(a)




(b)




(c)

Figura 6.6: Performance das melhores *Backpropagation Momentum* (a)*Bluetooth*, (b) *Zigbee* e (c)*Wi-Fi*

Tabela 6.4: Tabela de desempenho das RNA *Backpropagation Momentum*

| Desempenho | Neurônios na Camada Escondida | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 5 | 10 | 15 | 20 | 30 | 40 | 50 | 60 | 80 | 100 |
| Wi-Fi | 0,42 | 0,79 | 0,065 | ***0,059*** | 0,0061 | 0,097 | 0,75 | 0,08 | 0,063 | 0,32 |
| Zigbee | 0,07 | 0,008 | 0,0064 | 0,81 | 0,87 | ***0,061*** | 0,077 | 0,09 | 0,095 | 0,071 |
| Bluetooth | 0,087 | 0,087 | 0,09 | 0,25 | 0,53 | 0,089 | 0,73 | ***0,085*** | 0,32 | 0,099 |

6.7c mostram os gráficos de regressão das três melhores redes neurais para os protocolos *Bluetooth* (0.98595), *Zigbee* (0.95958) e *Wi-Fi* (0.98447) respectivamente.

Analisando as saídas obtidas com a utilização das três melhores redes neurais, verificou-se que os resultados da rede neural treinada com os dados RSSI do *Wi-Fi* foram superiores aos demais. A Tabela 6.5 mostra o comparativo dos três protocolos de rede sem-fio com relação à precisão e distância.

Tabela 6.5: Tabela comparativa de precisão e distâncias das melhores RNA *Backpropagation Momentum*

| | Precisão (%) | Distância (m) |
|---|---|---|
| Wi-Fi | 72 | 0,64 |
| Zigbee | 71 | 1,83 |
| Bluetooth | 65 | 2,95 |

As Figuras 6.8, 6.9 e 6.10 exemplificam a questão da distância real e as distâncias obtidas pelas redes neurais. A Figura 6.11 elucida a Tabela 6.5 é mostra graficamente que a rede neural treina da com sinais *Wi-Fi* foi superior as demais.

Com relação à exatidão novamente o protocolo *Wi-Fi* foi melhor. A Figura 6.11 demonstra o gráfico comparativo das melhores redes neurais de cada protocolo de rede, em uma amostra de teste de vinte sete pontos. Neste gráfico fica claro a superioridade em relação à qualidade do resultado da rede treinada a partir dos dados providos pelo protocolo *Wi-Fi*.

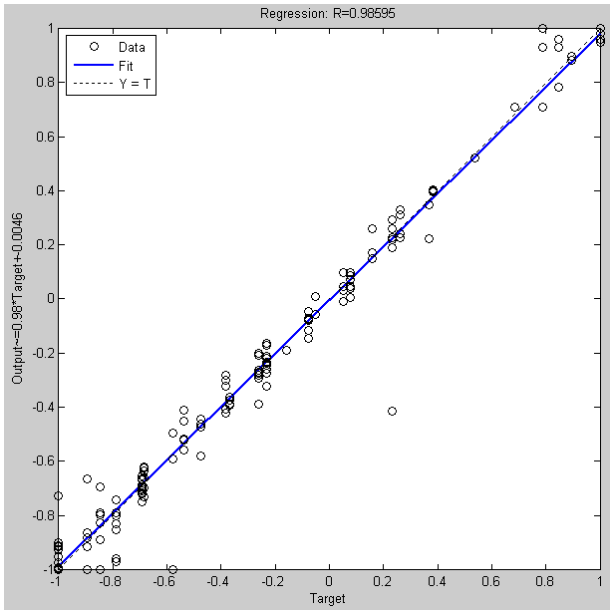
## 6.5 Experimento 3 - Algoritmo *Levenberg-Marquart*

Assim como os algoritmos anteriores o *Levenberg-Marquart*, seguiram os passos descritos na Seções 6.2. Sendo testadas dez redes neurais que implementaram este algoritmo.

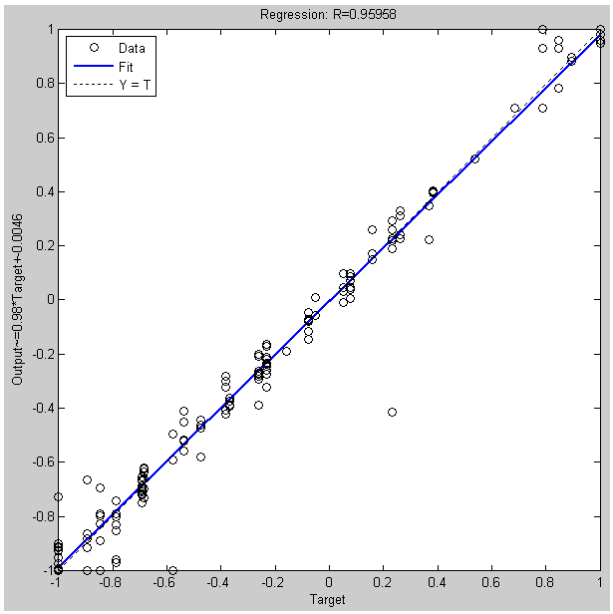
Nesse algoritmo houve somente uma inversão do segundo e terceiro melhor resultado, ou seja, o *Bluetooth* apresentou melhor desempenho que o *Zigbee*. Novamente quem apresentou melhor resultado foi o protocolo *Wi-Fi*.

A melhor rede neural com uso de dados *Wi-Fi* teve vinte neurônios na camada escondida, a melhor com dados *Bluetooth* teve trinta neurônios na camada escondida e a que utilizou os sinais do *Zigbee* teve cinqüenta neurônios na camada escondida.

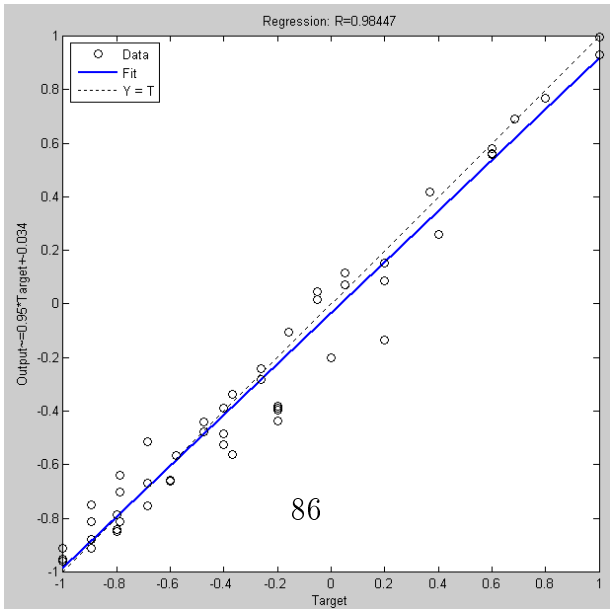
As Figuras 6.12a, 6.12b e 6.12c apresentam os melhores desempenho das redes neurais criadas para os protocolos *Bluetooth* $(0,0064)$, *Zigbee* $(0,0095)$ e *Wi-Fi* $(0,0062)$ respectivamente.


Regression: R=0.98595
Data
Fit
Y = T
Output~=0.98*Target+-0.0046
Target

(a)


Regression: R=0.95958
Data
Fit
Y = T
Output~=0.96*Target+-0.0046
Target

(b)


Regression: R=0.98447
Data
Fit
Y = T
Output~=0.95*Target+-0.034
Target

(c)

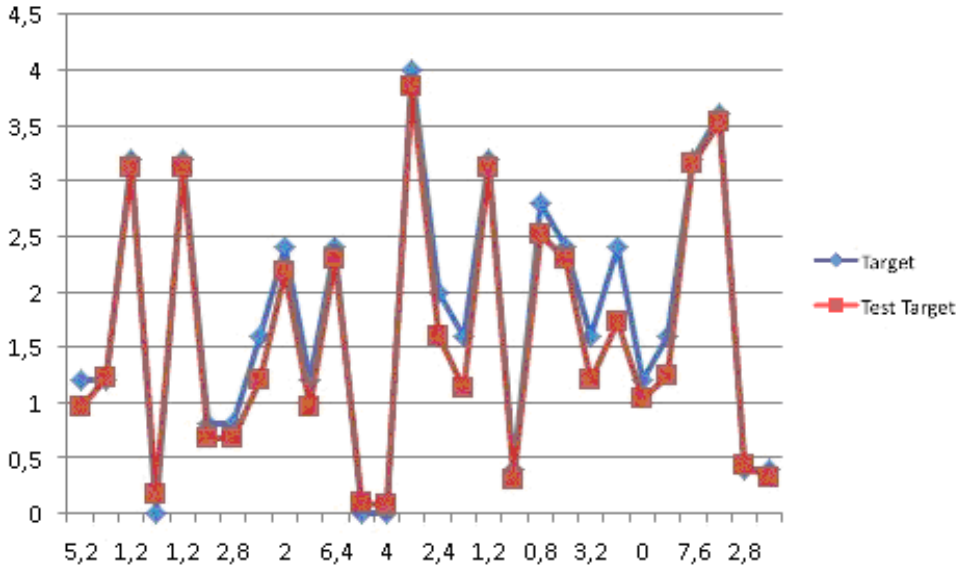



Figura 6.8: Diferença entre coordenada Esperada x Obtida RNA *Backpropagation Momentum Wi-Fi*

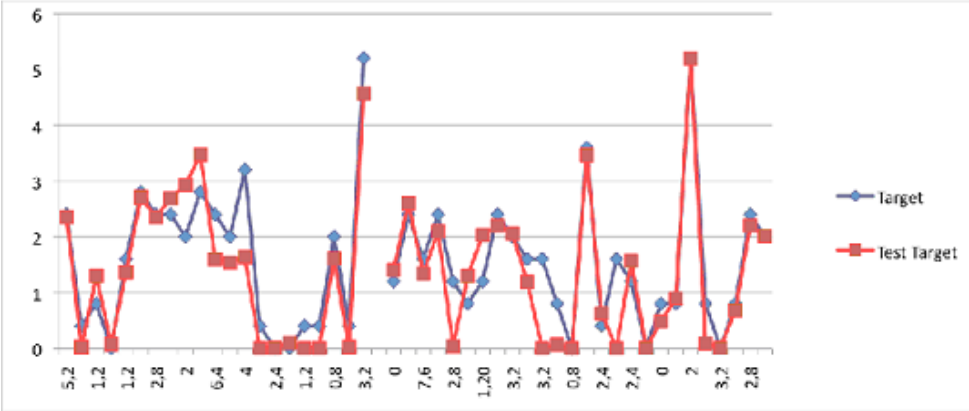



Figura 6.9: Diferença entre coordenada Esperada x Obtida RNA *Backpropagation Momentum Bluetooth*

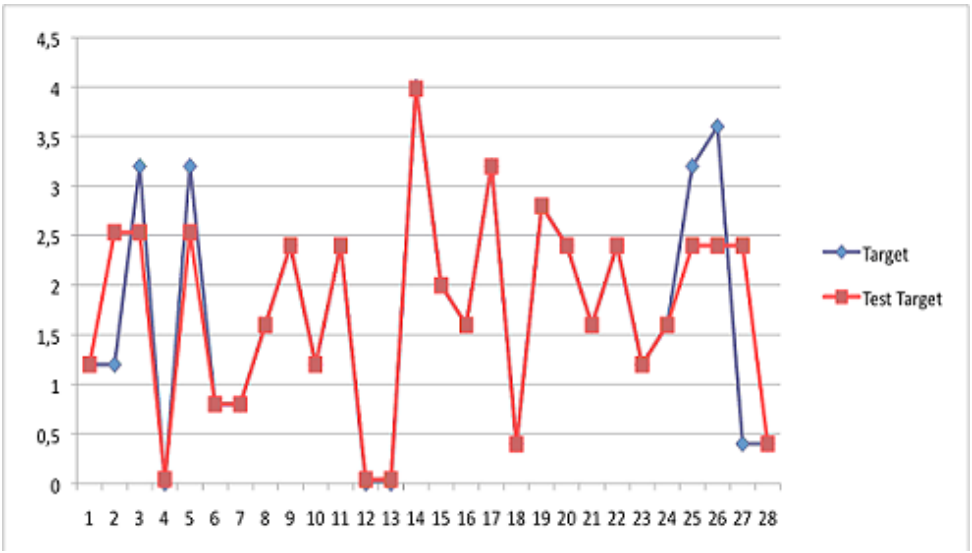



Figura 6.10: Diferença entre coordenada Esperada x Obtida RNA *Backpropagation Momentum Zigbee*

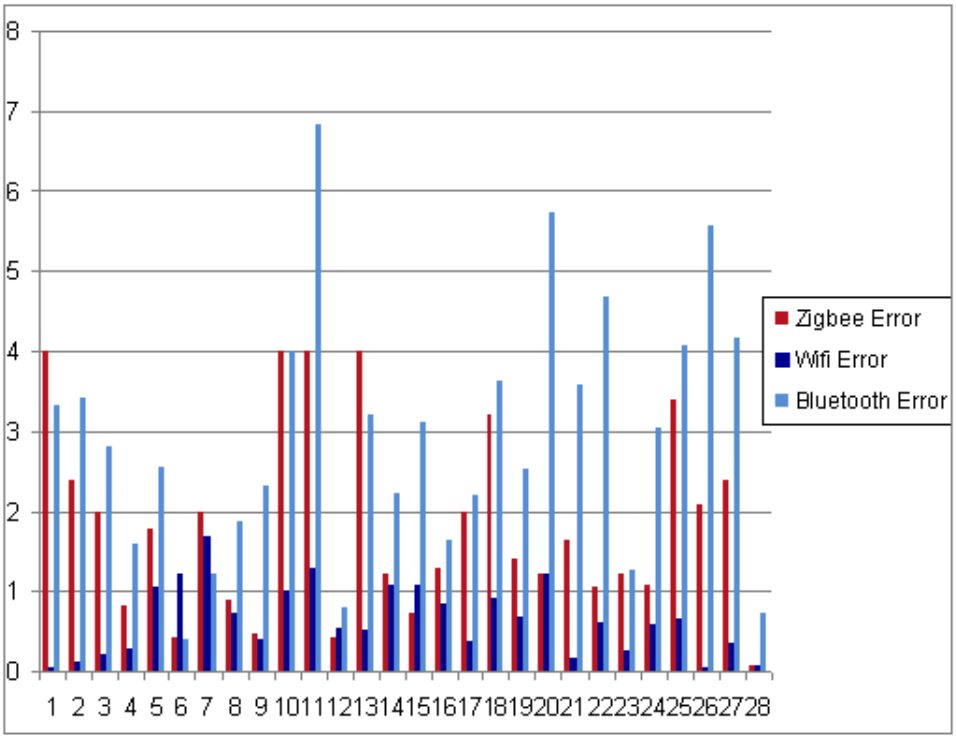
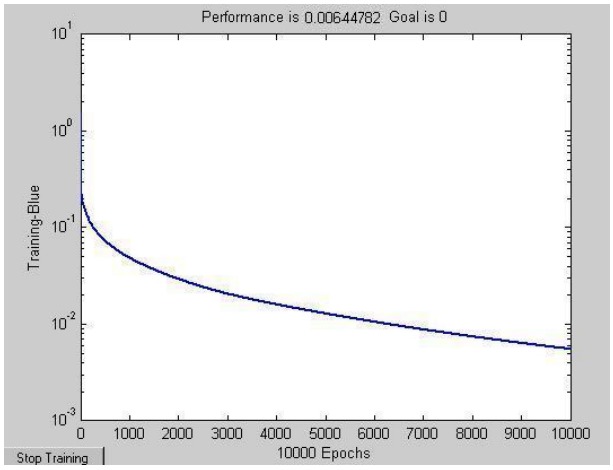
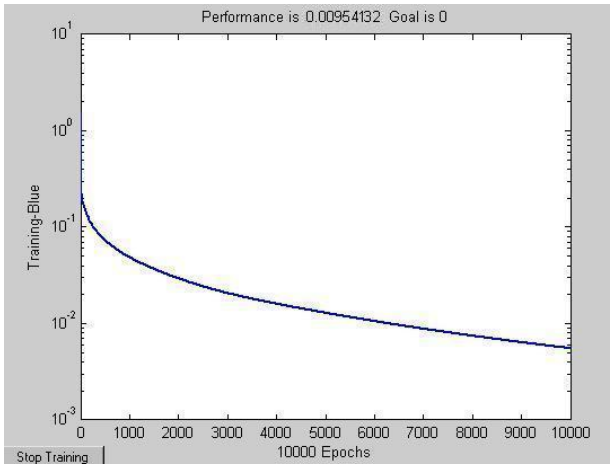


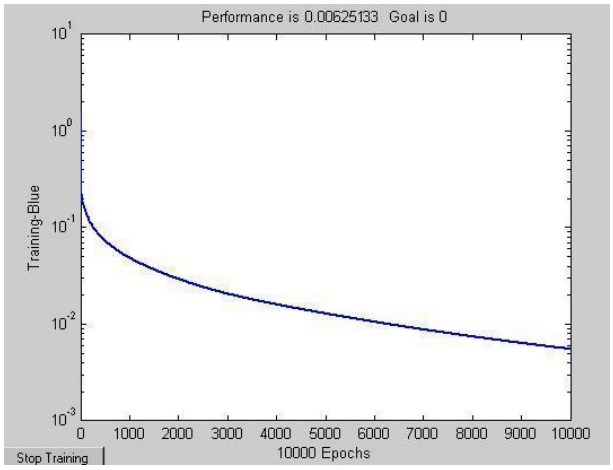

Figura 6.11: Gráfico comparativo das melhores redes neurais




(a)




(b)




(c)

Figura 6.12: Desempenho das melhores RNA *Levenberg-Marquart* (a)*Bluetooth*, (b) *Zigbee* e (c)*Wi-Fi*

Assim como os outros dois algoritmos analisados, este também obteve bons resultados com relação à regressão, todos chegaram muito próximo do objetivo de 1.0. As Figuras 6.13a, 6.13b e 6.13c mostram os gráficos de regressão das três melhores redes neurais para os protocolos *Bluetooth* $(0,995)$, *Zigbee* $(0,886)$ e *Wi-Fi* $(0,998)$ respectivamente.

Em relação às saídas obtidas com a utilização das três melhores redes neurais, notou-se que os resultados da rede neural treinada com os dados RSSI do *Wi-Fi* foram superiores aos demais. A Tabela 6.6 mostra o comparativo dos três protocolos de rede sem-fio com relação a precisão e a distância.

Tabela 6.6: Tabela comparativa de precisão e distâncias das melhores RNA *Levenberg-Marquart*

| | Precisão (%) | Distância (m) |
|---|---|---|
| Wi-Fi | 70 | 0,89 |
| Zigbee | 67 | 1,94 |
| Bluetooth | 69 | 2,15 |

A Tabela 6.7 mostra para cada protocolo de rede analisado as melhores e piores coordenadas, com o seu alvo e erro.

Tabela 6.7: Tabela comparativa de coordenadas das melhores RNA *Levenberg-Marquart*

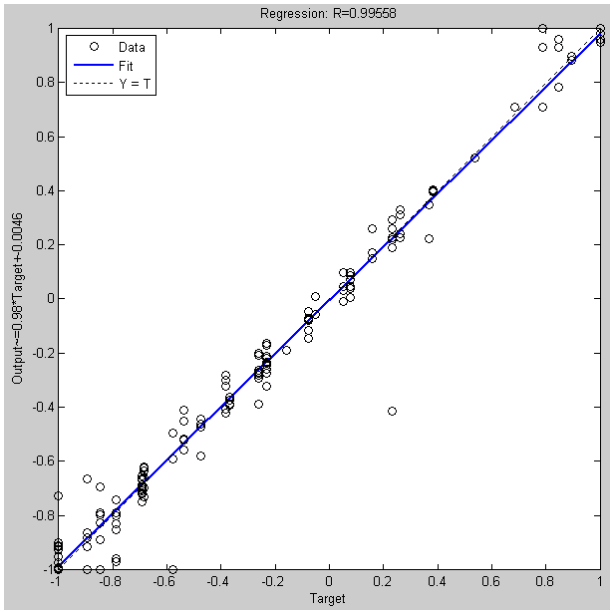
| | Melhor Coordenada | Alvo | Erro (m) | Pior Coordenada | Alvo | Erro (m) |
|---|---|---|---|---|---|---|
| Wi-Fi | (1,16;5,2) | (1,2;5,2) | 0,04 | (3,16;3,2) | (3,2;7,6) | 4,4 |
| Zigbee | (0,42;2,18) | (0,4;2,8) | 0,62 | (0,3;0,4) | (0,4;3,6) | 3,2 |
| Bluetooth | (0,04;0,8) | (0;0,8) | 0,89 | (0,54; 4,56) | (2,0;2,4) | 3,12 |

A Figura 6.14 apresenta o gráfico comparativo das melhores redes neurais de cada protocolo de rede, sobre a perspectiva da exatidão em uma amostra de teste de vinte sete pontos. Neste gráfico fica claro a superioridade em relação à qualidade do resultado da rede treinada a partir dos dados providos pelo protocolo *Wi-Fi*.

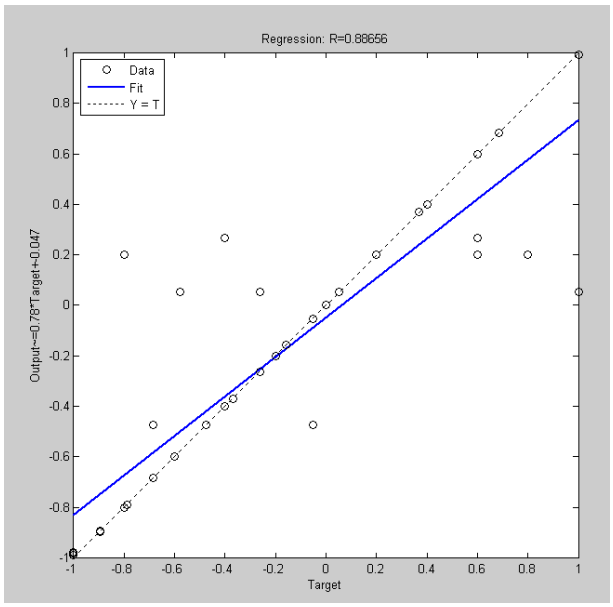
## 6.6 Experimento 4 - Regras de Inferência

Nesse experimento foi utilizado o protótipo criado usando SMA e as melhores RNA de cada protocolo de rede testado. Para dados *Bluetooth* usou-se o algoritmo *Levenberg-Marquart*, para os dados *Zigbee* utilizou-se o algoritmo *Backpropagation Momentum* e para os dados *Wi-Fi* foi utilizada a rede neural que utiliza o algoritmo *Backpropagation*. Os resultados do protocolo *Wi-FI* com uso do algoritmo *Backpropagation* e QoS já foi publicado e apresentado em (Fonseca et al., 2011).

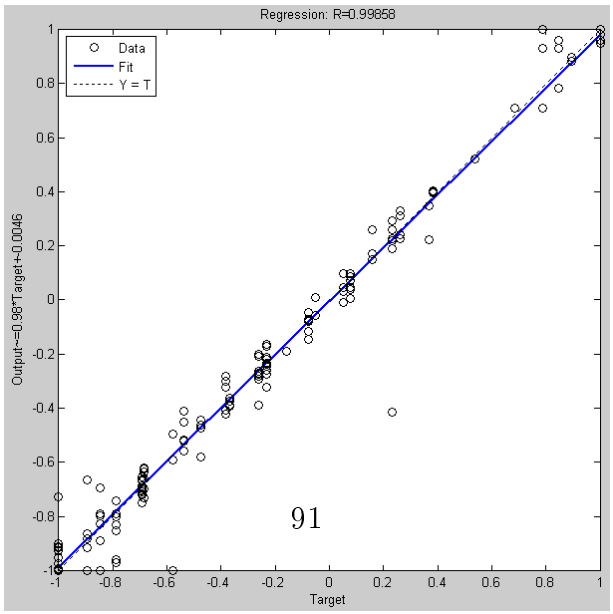
Nos testes foram usados 28 pontos. Quando usamos o primeiro nível de QoS, o *AgenteMonitorErro*, obtemos uma exatidão de 76% para os dados *Bluetooth*, 77% para os dados *ZigBee* e 83% para os dados *Wi-Fi*. Combinando com o segundo nível de QoS, *AgenteNearAps*, elevou-se a acurácia para 78% para o *Bluetooth*, 88% para o *ZigBee* e 89% no caso do *Wi-Fi*. A Tabela 6.8 apresenta os resultados para os três protocolos, obtidos pelas RNA usando os dois níveis de QoS.


Regression: R=0.99558
Data
Fit
Y = T
Output~=0.98*Target+-0.0046
Target

(a)


Regression: R=0.88656
Data
Fit
Y = T
Output~=0.78*Target+-0.047
Target

(b)


Regression: R=0.99858
Data
Fit
Y = T
Output~=0.98*Target+-0.0046
Target

(c)

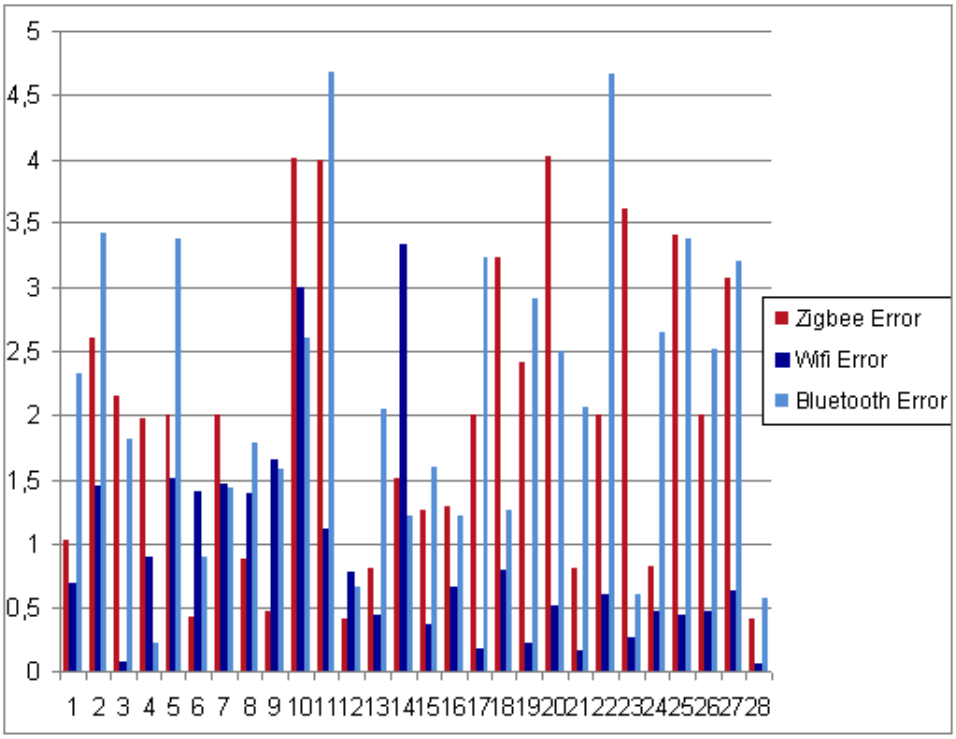



Figura 6.14: Gráfico comparativo das melhores redes neurais

Tabela 6.8: Tabela Resultados das regras de inferência

| Localização | RSSI | ANN | Erro ANN | QoS Nível 1 | Erro QoS Nível 1 | QoS Nível 2 | Erro QoS Nível 2 |
|---|---|---|---|---|---|---|---|
| Wi-Fi (0,8) | 61-52-53-51-60 | (0,0;1,2) | 0,4 | - | - | - | - |
| Wi-Fi (1,4) | 72-51-61-40-50 | (2,1;5,3) | 1,7 | (1,3;4,1) | 0,31 | - | - |
| Bluetooth (3,2;1,2) | 48-72-49-73-63 | (2,53;2,0) | 3,2 | (2,5;1,8) | 0,92 | - | - |
| Bluetooth (1,6;2,4) | 67-44-68-47-49 | (3,0;3,6) | 1,8 | (3,1;3,2) | 1,9 | (2,0;2,6) | 0,44 |
| Zigbee (1,2;5,2) | 54-77-76-69-76 | (0,01;1,2) | 4 | (1,4;4,8) | 0,44 | - | - |
| Zigbee (2,0;2,4) | 79-66-65-73-55 | (1,59;2,0) | 0,57 | - | - | - | - |

## 6.7 Comparativo

Analisando todos os resultados fica evidente que o melhor protocolo de rede a ser utilizado é o *Wi-Fi*, como visto na Tabela 6.9. Esta resultado já era esperado devido a suas características de potência de sinal. Contudo ao utilizar o *Wi-Fi* consume-se muito da bateria do dispositivo móvel,diferentemente dos outros protocolos.

A Tabela 6.9 compara exclusivamente os resultados obtidos pelas RNA, *Backpropagation*, *Backpropagation Momentum* e *Levenberg-Marquart* com os protocolos de rede *Bluetooth*, *Zigbee* e *Wi-Fi*. Essa Tabela apresenta o desempenho, regressão, precisão e distância.

Em relação aos algoritmos de RNA testados, obteve-se resultados aceitáveis e até superiores ao da literatura. Comparando os algoritmos RNA houve uma melhora de $35,8\%$ para *Bluetooth* , $9,8\%$ no *Zigbee* e $52,8\%$ *Wi-Fi*. Percentualmente o *Bluetooth* ficou atrás do *Wi-Fi*, porém foi o que mais progrediu alcançando uma melhoria de 1.2m

Tabela 6.9: Tabela Comparativa de RNA

| | Desempenho | Regressão | Precisão (%) | Distância (m) |
|---|---|---|---|---|
| Backpropagation Bluetooth | 0,0068 | 0,985 | 63 | 0,42 |
| Backpropagation Zigbee | 0,0064 | 0,9844 | 67 | 2,03 |
| Backpropagation Wi-Fi | 0,0055 | 0,9859 | 73 | 3,35 |
| Backpropagation Momentum Bluetooth | 0,085 | 0,98595 | 65 | 2,95 |
| Backpropagation Momentum Zigbee | 0,061 | 0,95958 | 71 | 1,83 |
| Backpropagation Momentum Wi-Fi | 0,0059 | 0,98447 | 72 | 0,64 |
| Levenberg-Marquart Bluetooth | 0,0064 | 0,995 | 70 | 2,15 |
| Levenberg-Marquart Zigbee | 0,0095 | 0,886 | 67 | 1,94 |
| Levenberg-Marquart Wi-Fi | 0,0062 | 0,998 | 69 | 0,89 |

em acurácia. A Tabela 6.10 mostra a diferença encontrada entre os melhores e piores algoritmos utilizados nos experimentos.

Tabela 6.10: Tabela de resultados dos algoritmos RNA

| | Melhor Resultado | Pior Resultado | Melhora (%) |
|---|---|---|---|
| Bluetooth | 2,15 m (Levenberg-Marquart) | 3,35 m (Backpropagation) | 35,8 |
| Zigbee | 1,83 m (Backpropagation Momentum) | 2,03 m (Backpropagation) | 9,8 |
| Wi-Fi | 0,42 m (Backpropagation) | 0,89 m (Levenberg-Marquart) | 52,8 |

Analisando o protótipo juntamente com os melhores RNA para cada protocolo de rede, nota-se a melhoria nos resultados de todos os protocolos. Pela definição de níveis de qualidade todos os protocolos passaram a trabalhar com o mesmo nível de erro aceitável, isso proveu aos protocolos *Bluetooth* e *ZigBee* um incremento de 9% e 17% respectivamente em suas acurácias. Em aspectos de economia de energia e localização o protocolo *ZigBee* fica a frente do *Wi-Fi*, uma vez que a diferença entre os dois é de 1%.

A Tabela 6.11 compara os trabalhos apresentado na Seção 4.3 com os resultados obtidos neste protótipo. Nos trabalhos estudados há a limitação referente a utilização de diferentes tipos de protocolos de rede sem-fio. Nota-se também que mesmo o *LocationAgent* não apresentando uma acurácia de 90% com nos quatro projetos apresentado, o *LocationAgent* tem uma precisão igual o superior, podendo ser 6.76 vez mais preciso do que os demais.

Tabela 6.11: Tabela Comparativa de Trabalhos.

| Seção | Trabalho | Protocolo | Precisão (%) | Exatidão (m) | Algoritmo |
|---|---|---|---|---|---|
| 4.3.1 | Radar | WaveLan | 30 | 2,13 | *Nearest Neighbor* |
| 4.3.2 | Horus | Wi-Fi | 90 | 2,13 | *Bayesiano* |
| 4.3.3 | RTLS | Wi-Fi | N/A | 1 | *Ekahau Positioning Engine* |
| 4.3.4 | MoCA | Wi-Fi | 50<br>70<br>90 | 1,56<br>1,74<br>2,84 | *Nearest Neighbor* |
| 5.1 | Location Agent | Bluetooth<br>Zigbee<br>Wi-Fi | 78<br>88<br>89 | 2,15<br>1,83<br>0,42 | *Levenberg-Marquart*<br>*Backpropagation Momentum*<br>*Backpropagation* |

# Capítulo 7

# Conclusões

Os serviços de localização devem não somente prover a informação de posição do usuário, mas devem estar aptos a transmitir essa informação para outros serviços que dependam da localização. Dessa forma, e dada à onipresença do serviço de localização, as questões derivadas desse problema foram abordadas pela criação do protótipo de localização de usuários móveis em ambientes fechados.

Considerando o objetivo previamente definido no projeto e os recursos à disposição existentes, notou-se um elevado número de possibilidades de elaboração de um sistema base para um algoritmo de localização. Sendo assim as diversas características identificadas por esse tipo de sistema, optou-se pela adoção de SMA, RNA e QoS. Nessa proposta, foi apresentado o *LocationAgent*, uma arquitetura integrando agentes de aquisição de dados, controle de conflitos, algoritmos de RNA, e suas implementações utilizando a plataforma JADE.

Para o desenvolvimento do *LocationAgent* foram realizados estudos nas áreas de SMA, RNA e QoS. Foram estudados algoritmos de RNA, os quais podem ser utilizados independentes do protótipo criado. Outro estudo realizado foi sob o framework JADE, a fim de possibilitar o desenvolvimento do protótipo utilizando a abordagem de SMA.

O *LocationAgent* foi concebido sobre uma arquitetura de cinco camadas (interface com LBS,aquisição de dados, controle de conflitos, RNA e QoS) (Figura 5.1 com três super grupos de agentes (*RadarAgent*, *ConflictAgent* e LEA*Agent*), conforme apresentado no Capítulo 5. O *LocationAgent* foi definido para utilizar diferentes protocolos de rede em conjunto com diferentes tipos de algoritmos de RNA, utilizando uma abordagem cooperativa, sob uma perspectiva integrada, onde o principal objetivo é melhorar o retorno da localização do usuário móvel.

Além disso, o *LocationAgent* provê a automação da aquisição de dados RSSI, utilizando a abordagem de SMA para as primeiras fases do processo de localização. Enquanto o *ConflictAgent* possui autonomia para validar os dados obtidos automaticamente, assim reajustando os parâmetros de configuração e adaptando aos recursos disponíveis. Dessa forma, tornando o protótipo flexível para operar com os três protocolos e três redes neurais, podendo ser estendido a novas tecnologias.

Nesse trabalho, aplicamos o *AgentLocation* como uma proposta de solução para o problema de localização de dispositivos móveis em ambientes fechados. Para validar a proposta vários experimentos foram realizados e os resultados apresentados no Capítulo 6. Os resultados mostram o potencial da solução proposta, como na Tabela 6.10 que

apresenta o desempenho do protótipo com as RNA e protocolos de redes. As heurísticas aplicadas aos níveis de qualidade mostraram eficaz melhora no processo de $35,8\%$ para o *Bluetooth*, $9,8\%$ para o *ZigBee* e $52,8\%$ para o *Wi-Fi*.

O problema de localização de dispositivos móveis em ambientes fechados é muito importante, não somente como serviço de localização de usuários, mas também em outros serviços contextualizados que dependem da localização. Dessa forma, o *LocationAgent* atente os seus propósitos e contribuições, conforme a Seçao 1.4, provendo a informação de localização dos usuários com um erro máximo de 1.5m (para os três protocolos), como também provê uma interface de comunicação para que outros serviços façam uso da informação de localização.

## 7.1 Trabalhos Futuros

Este trabalho apresentou uma ferramenta genérica para a localização de usuários móveis em ambientes fechados, através de três protocolos do uso redes sem fio, com abordagem SMA, RNA e QoS. Além das contribuições já apresentadas, há a necessidade de análises de desempenho, integração de novos algoritmos de LEA e técnicas de localização, ficando como trabalhos futuros:

- Otimizar o processo de coleta de dados em termos de tempo e/ou automaticidade, para criação do mapa de *fingerprints*;
- Criação de métodos que conservem a validade do MF, uma vez que existe o problema de alteração de posicionamento de PAs;
- Integrar novas tecnologias de localização de usuários móveis, como o MNN e SVM, para ter mais mecanismos de localização de dispositivos móveis em ambientes fechados;
- Criação de agentes que trabalhem com o conceito de etiquetas (ex: Sala da Professora Célia) em vez de coordenadas absolutas ($X$,$Y$), cobrindo desta forma maiores ambientes fechados;
- Estender a arquitetura de proposta para um sistema distribuído, recorrendo a RMI e a *Web Services*, podendo oferecer serviços de forma dinâmica e flexível a diferentes agentes; e
- Desenvolver um *middleware* ou um *framework* baseada no modelo proposto, que possibilite a utilização de diferentes protótipos.

## 7.2 Publicações

Durante a realização desta pesquisa foram publicados resultados em artigos científicos. A seguir são listados os trabalhos publicados, aceitos para publicação:

- (de M. Neves et al., 2010) - Ana Regia de Mendonça Neves, Humphrey Fonseca, Letícia Zoby, and Célia Ralha. Localização de usuários em ambiente interno utilizando abordagem de sistema multiagente. iSys - Revista Brasileira de Sistemas de Informação, 3, 2010. ISSN Eletrônico: 1984-2902.

- (Fonseca et al., 2011) - Humphrey C. Fonseca, Ana Régia de M. Neves, and Célia Ghedini Ralha. A user location case study using diferent wireless protocols. In Proceedings of the 9th ACM international symposium on Mobility management and wireless access, MobiWac '11, pages 143-146, New York, NY, USA, 2011. ACM.

# Referências


G. D. Abowd and E. D. Mynatt. Charting past, present, and future research in ubiquitous computing. In *Trans. Comput.- Hum. Interact.*, volume 7, pages 76–88. ACM, March 2000. 1

G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a better understanding of context and context-awareness. In *1st international symposium on Handheld and Ubiquitous Computing.*, page 304–307, London, UK, 6 1999. Springer-Verlag. 1

I. Ahmed, S. Orfali, T. Khattab, and A. Mohamed. Characterization of the indoor-outdoor radio propagation channel at 2.4 ghz. In *GCC Conference and Exhibition (GCC), 2011 IEEE*, pages 605 –608, feb. 2011. 3, 17

ZigBee Alliance. Zigbee specification. http://www.zigbee.org/ Último acesso em Novembro de 2011. 12

M. Altini, D. Brunelli, E. Farella, and L. Benini. Bluetooth indoor localization with multiple neural networks. In *Proceedings of the 5th IEEE international conference on Wireless pervasive computing*, ISWPC'10, pages 295–300, Piscataway, NJ, USA, 2010. IEEE Press. ISBN 978-1-4244-6855-3. URL http://portal.acm.org/citation.cfm?id=1856330.1856382. 1

P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775 –784 vol.2, 2000. xii, 2, 3, 47, 51, 52, 55, 58

R. Battiti, T. L. Nhat, and A. Villani. Location-aware computing: a neural network model for determining location in wireless lans. Dit-02-0083, Universita degli Studi di Trento, Trento, Italy, 2002. 2, 39, 40, 49, 57

R. Beale and T. Jackson. Neural computing : an introduction. Bristol, England, 1990. xi, 40, 43

F. Bellifemine, G. Caire, and T. Trucco. Jade programmer's guide jade programmer's guide usage restricted according to license agreement., 2007. xi, 30, 31, 32, 33, 34, 36, 38

C. Bento, J. Peixoto, and M. Veloso. A case-based approach for indoor location. In *The Sixth International Conference on Case-Based Reasoning, Lecture Notes in Artificial Intelligence series*, Chicago, Illinois (USA), August 2005. 1

P. Bilurkar, N. Rao, G. Krishna, and R. Jain. Application of neural network techniques for location predication in mobile networking. In *Neural Information Processing,ICONIP. Proceedings of the 9th International Conference*, volume 5, pages 2157–2161, November 2002. 2

A. N. Bishop, B. Fidan, K. Dogancay, B. D. O. Anderson, and P. N. Pathirana. Exploiting geometry for improved hybrid aoa/tdoa-based localization. *Signal Processing*, 88(7): 1775–1791, 2008. xi, xii, 45, 46

Team BlueCove. Bluecove. http://www.bluecove.org/ Último acesso em Novembro de 2011. 66

M. Brunato and R. Battiti. Statistical learning theory for location fingerprinting in wireless lans. *Comput. Netw.*, 47:825–845, April 2005. ISSN 1389-1286. 39, 49, 50, 57

A. Cardon and D. N. Muller. Introdução as redes neurais artificiais. http://www.ulbra.tche.br/ danielnm/ia/rna/rna.html. 40

L. A. Castro and J. Favela. Continuous tracking of user location in wlans using recurrent neural networks. In *Proceedings of the Sixth Mexican International Conference on Computer Science*, ENC '05, pages 174–181, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2454-0. 60, 63, 73

D. Chauhan and A. Baker. Developing coherent multiagent systems using jafmas. *Multi-Agent Systems, International Conference on*, 0:407, 1998. 26

A. R. de M. Neves, H. Fonseca, L. Zoby, and C. Ralha. Localização de usuários em ambiente interno utilizando abordagem de sistema multiagente. *iSys - Revista Brasileira de Sistemas de Informação*, 3, 2010. ISSN Eletrônico: 1984-2902. 79, 95

A. Dey, D. Salber, and G. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, 2001. 1

A. K. Dey. Understanding and using context. In *Personal Ubiquitous Computing*, volume 5, pages 4–7, January 2001. 1

International Inc Digi. Xbee multipoint rf modules. www.digi.com/pdf/ds_xbeemultipointmodules.pdf Último acesso em Novembro de 2011, 2006-2008. 67

G. Dorffner, E. Prem, and H. W. Trost. Symbols, and symbol grounding. Univie, 1993. 39

Inc. Ekahau. Positioning engine 4.1. http://www.ekahau.com/file.php?id=99366 Último acesso em Novembro de 2011. 54, 73

R. Faludi. *Building Wireless Sensor Networks: With ZigBee, XBee, Arduino, and Processing*. O'Reilly Series. O'Reilly Media, 2010. ISBN 9780596807733. xi, xiv, 13, 14, 15

Jacques Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Professional, February 1999. ISBN 0201360489. 25

B. Ferris, D. Fox, and N. Lawrence. Wifi-slam using gaussian process latent variable models. In *20th International Joint Conference on Artificial Intelligence*, volume 7, pages 2480–2485, Hyderabad, India, 2007. IJCAI. 1

FIPA. Agent management specification. component, foundation for intelligent physical agents. http://www.fipa.org/specs/fipa00023/ Último acesso em Novembro de 2011. xi, 26, 27, 28, 35, 38

H. C. Fonseca, A. R. de M. Neves, and C. G. Ralha. A user location case study using different wireless protocols. In *Proceedings of the 9th ACM international symposium on Mobility management and wireless access*, MobiWac '11, pages 143–146, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0901-1. doi: http://doi.acm.org/10.1145/2069131.2069156. URL http://doi.acm.org/10.1145/2069131.2069156. 59, 79, 90, 96

J. A. Freeman and D. M. Skapura. Neural networks algorithms, applications and programmming techniques. Addelfon-Wefley, 2004. xi, 41, 43

P. Giorgini, J. Mylopoulos, and R. Sebastiani. Goal-oriented requirements analysis and reasoning in the tropos methodology. *Eng. Appl. Artif. Intell.*, 18:159–171, March 2005. ISSN 0952-1976. 72

Bluetooth Special Interest Group. Bluetooth network encapsulation protocol (bnep) specification. www.bluetooth.org Último acesso em Novembro de 2011. xiv, 6, 8, 10

R. R. Gudwin. Introdução à teoria de agentes. http://www.dca.fee.unicamp.br/ gudwin/-courses/IA009/, 2009. 27

M. T. Hagan and M. B. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 8(6):989–993, 1994. 43

S. Haykin. *Redes Neurais: Princípios e Prática*, volume 2. Bookman, 2001. xi, 39, 40, 41, 42

J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34:57–66, 2001. ISSN 0018-9162. 3

V. Honkavirta, T. Peraäla, S. Ali-Loöytty, and R. Piché. A comparative survey of wlan location fingerprinting methods. In *6th Workshop on Positioning, Navigation and Communication 2009 (WPNC'09)*, page 243–251, March 2009. 1

N. R. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1:7–38, 1998. ISSN 1387-2532. 10.1023/A:1010090405266. 23

K. Kaemarungsi. *Design of indoor positioning systems based on location fingerprinting technique*. PhD thesis, Pittsburgh, PA, USA, 2005. AAI3188962. xiv, 3, 47, 49, 50, 55, 57, 59, 63

K. Kaemarungsi and P. Krishnamurthy. Modeling of indoor positioning systems based on location fingerprinting. In *Trans. Comput.- Hum. Interact.*, volume 2. INFOCOM 2004, 2004. 1

M. Kavehrad and P. McLane. Spread spectrum for indoor digital radio. *Communications Magazine, IEEE*, 25(6):32 – 40, jun 1987. ISSN 0163-6804. 17

A. Koubaa, A. Cunha, and M. Alves. A time division beacon scheduling mechanism for ieee 802.15.4/zigbee cluster-tree wireless sensor networks. In *Proceedings of the 19th Euromicro Conference on Real-Time Systems*, pages 125–135, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2914-3. xi, xiv, 12, 13, 14

T.-N. Lin and P.-C. Lin. Performance comparison of indoor positioning techniques based on location fingerprinting in wireless networks. In *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, volume 2, pages 1569–1574. IEEE, 2005. ISBN 0-7803-9305-8. URL http://dx.doi.org/10.1109/WIRLES.2005.1549647. 48, 57

C. P. Luft. *Minidicionário Luft*. Editora Ática. São Paulo, 1998. 23

T. W. Malone. Modeling coordination in organizations and markets. In A. H. Bond and L. Gasser, editors, *Readings in Distributed Artificial Intelligence*, pages 151–158. Kaufmann, San Mateo, CA, 1988. 22

D. Marquardt. An algorithm for least squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963. 43

E. X. Meneses. Jornada de atualização em inteligência artificial – integração de agentes de informação, 2001. 27

Corporation Microsoft. Como usar a ferramenta netsh.exe e as opções de linha de comando. http://support.microsoft.com/kb/242468/pt-br Último acesso em Novembro de 2011, a. 67

Corporation Microsoft. Microsoft technet, netsh commands for wireless local area network (wlan). http://technet.microsoft.com/enus/library/cc755301.aspx Último acesso em Novembro de 2011, b. 67

M. Miller. *Discovering Bluetooth*. SYBEX Inc., Alameda, CA, USA, 2001. ISBN 0782129722. xi, xiv, 7, 8, 9, 10

IEEE 802.11™ WIRELESS LOCAL AREA NETWORKS. Ieee 802 standard. http://www.ieee802.org/11/ Último acesso em Novembro de 2011, a. 16, 20

IEEE 802.11TM WIRELESS LOCAL AREA NETWORKS. Ieee 802.11™: Wireless local area networks (lans). http://standards.ieee.org/about/get/802/802.11.html Último acesso em Novembro de 2011, b. xi, 18

F. Niessink and H. V. Vliet. Towards mature it services. *Software Process: Improvement and Practice*, 4(2):55–71, 1998. 1

A. S. I. Noh, W. J. Lee, and J. Y. Ye. Comparison of the mechanisms of the zigbee's indoor localization algorithm. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, volume 0, page 13–18. ACIS International Conference, 2008. 1

B. Nunes. *Um Sistema de localização para redes Wi-Fi baseado em níveis de sinal e modelo referenciado de propagação*. PhD thesis, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil, 2006. 1

Corporation Oracle. Java me at a glance. http://java.sun.com/javame/index.jsp Último acesso em Novembro de 2011. 66

B.B. Parodi, A. Szabo, H. Lenz, J. Bamberger, and J. Horn. Sll: Relations to kohonen soms. In *Position, Location and Navigation Symposium*, pages 641–645. IEEE/ION, May 2008. 2

C. S. Peirce. *Semiótica*, volume 2. Perspectiva, 1995. 39

R. L. Pickholtz, D. L. Schilling, and L. B. Milstein. Theory of spread-spectrum communications – a tutorial. *IEEE Transactions on Communications*, 30(5):855–884, May 1982. 17, 20

M. D. Rodriguez, J. Favela, E. A. Martinez, and M. A. Munoz. Location-aware access to hospital information and services. *Information Technology in Biomedicine IEEE*, 8(4): 448–455, Dec. 2004. 2

T. Roos, P. Myllym aki, H. Tirri, P. Misikangas, and J. Siev anen. A Probabilistic Approach to WLAN User Location Estimation. *International Journal of Wireless Information Networks*, 9(3):155–164, July 2002. 49, 50

H. K. Rubinsztejn, M. Endler, V. Sacramento, K. M. Gonçalves, and F. N. Nascimento. Support for context-aware collaboration. In *MATA*, pages 37–47, 2004. xii, xiv, 54, 55, 56, 58

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing : exploration in the microstructure of cognition. Cambridge : MIT Press*, (4):448–455, Dec. 1986. 43

S. Saha, K. Chaudhuri, D. Sanghi, and P. Bhagwat. Location determination of a modile device using ieee 802.11b access point signal. *Wireless communication and networking*, 3:1987–1992, Mar. 2003. 2, 40, 49

A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 166–179, New York, NY, USA, 2001. ACM. ISBN 1-58113-422-3. 1

C. E. Shannon and J. McCarty. Automata studies. 1956. 39

L. Song, D. Kotz, R. Jain, and X. He. Evaluating location predictors with extensive wi-fi mobility data. In *Twenty-third Annual Joint Conference of the IEEE Computer and Communications (INFOCOM)*, volume 2, pages 1414–1424. IEEE, Mar. 2004. 1, 2

W. Stallings. *Wireless Communications & Networks (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004. ISBN 0131918354. xi, 7, 17

R. Sun. Artificial intelligence: Connectionist and symbolic approaches, 1999. 39

N. Swangmuang and P. Krishnamurthy. Location fingerprint analyses toward efficient indoor positioning. In *Pervasive Computing and Communications*, page 100–109. PerCom Sixth Annual IEEE International Conference on, 2008, 2008. 1

T. J. Thompson, C. B. Kumar, and P. J. Kline. *Bluetooth Application Programming with the Java APIs Essentials Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008. ISBN 0123743427, 9780123743428. 66

C. Y. Tsai, S. Y. Chou, S. W. Lin, and W. H. Wang. Location determination of mobile device for indoor wlan application using neural network. In *Intelligent Environments, 2008 IET 4th International Conference*, volume 7, pages 1–8, Hyderabad, India, 21-22 July 2008. IJCAI. 2

V. N. Vapnik. *Statistical learning theory*. Wiley, 1 edition, September 1998. ISBN 0471030031. 50

R. Want, A. Hopper, V. Falc ao, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10:91–102, January 1992. ISSN 1046-8188. 3, 45

P. D. Wasserman. Neural computing : Theory and practice. New York : Van Nostrand Reinhold, 1986. 43

P. Wei, L. Zhizhan, and Z. Yi. Multi-person location and tracking method based on bp neural network. In *Cybernetics and Intelligent Systems - IEEE Conference*, pages 28–31, Sept. 2008. 2

G. Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, 1999. ISBN 978-0-262-23203-6. xi, xiv, 24, 28, 29, 30

M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 1994. Submitted to Revised. 23

M. Youssef and A. Agrawala. On the optimality of wlan location determination systems. In *In Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 205–218, 2003. xii, 49, 53, 58

M. A. Youssef, A. Agrawala, and A. U.Shankar. Wlan location determination via clustering and probability distributions. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, PERCOM '03, pages 143–, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1893-1. URL http://dl.acm.org/citation.cfm?id=826025.826335. 52, 54