The Base Station Application of the CAMBADA Robotic Soccer Team

Nuno Figueiredo, António Neves, Nuno Lau, José Azevedo, Artur Pereira and Gustavo Corrente

***Abstract* – The base station is the software application responsible to provide automatic processing of soccer game refereeing events and to allow high level monitoring and control of the robots internal states. This paper presents the base station developed for CAMBADA, the robotic soccer team of the University of Aveiro. It describes the main requirements and specifications of the base station and presents the architecture of the application, giving special attention to the description of the main modules and to the connection between them. It also describes the multi-window system, among other issues, namely the classes implemented and the mechanism of passing information among the several modules.**

***Resumo* – A estação base é a aplicação de software responsável pelo processamento automático dos eventos que ocorrem num jogo de futebol robótico e pela monitorização e controlo do estado interno dos robôs. O presente artigo descreve a estação base desenvolvida para a equipa CAMBADA, o projecto de futebol robótico da Universidade de Aveiro. Neste artigo são descritos os principais requisitos e especificações da aplicação, bem como a sua arquitectura, dando especial atenção aos principais módulos e à forma como eles se interligam e comunicam entre si. O artigo descreve também o sistema de janelas múltiplas bem como algumas outras questões, nomeadamente, as classes implementadas e o mecanismo de passagem de informação para diversos módulos do sistema.**

I. Introduction

Robotic soccer is nowadays a popular research domain in the area of multi-robot systems. RoboCup[1] is an international joint project to promote research in artificial intelligence, robotics and related fields. RoboCup chose soccer as the main problem aiming at innovations to be applied for socially relevant problems. It includes several competition leagues, each one with a specific emphasis, some only at software level, others at both hardware and software, with single or multiple agents, cooperative and competitive.

In the context of RoboCup, the Middle Size League (MSL) is one of the most challenging. In this league, each team is composed of up to 6 robots with a maximum size of $50cm \times 50cm$ width, $80cm$ height and a maximum weight of 40Kg, playing in a field of $18m \times 12m$. The rules of the game are similar to the official FIFA rules, with minor changes required to adapt them for the playing robots [1].

The rules of this league establish several constraints to simplify perception and world modeling. In particular, the ball is orange, the field is green, the field lines are white and the players are black. The duration of a game is 30 minutes, not including a half-time interval of 5 minutes. The game is refereed by a human and his orders are communicated to the teams using an application called "referee box" operated by an assistant referee. The referee box sends the referee orders to the team through a wired LAN TCP link connected to the external computer of each team. It is the team responsibility to communicate these orders to the robots through the field wireless network.

No human interference is allowed during the games except for removing malfunctioning robots and reentering robots in the game. Each robot is autonomous and has its own sensorial means. They can communicate among each other and with an external computer through a wireless network. This external computer, that has no sensor of any kind, runs the base station application. The base station only "knows" what is reported by the playing robots and the orders received from the referee box. The agents should be able to evaluate the state of the world and take decisions suitable to fulfill the cooperative team objective.

CAMBADA[2], *Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture*, is the Middle Size League Robotic Soccer team from the University of Aveiro. The CAMBADA research project started in 2003, coordinated by the Transverse Activity on Intelligent Robotics (ATRI)[3] group of the Institute of Electronic and Telematic Engineering of Aveiro (IEETA)[4]. Since then, it has involved people working on several areas for building the mechanical structure of the robot, its hardware architecture and controllers [2] and the software development in areas such as image analysis and processing [3]-[7], sensor and information fusion [8], reasoning and control [9].

Since its creation, the team has participated in several competitions, both national and international. Each year, new challenges are presented, and new objectives are defined, always with a better team performance

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

[2] http://www.ieeta.pt/atri/cambada

[3] http://www.ieeta.pt/atri

[4] http://www.ieeta.pt

in sight. After achieving the first place in the national competition Robótica 2007 and Robótica 2008, the 5th place in the world championship RoboCup 2007, this year the team achieved the first place in the world championship RoboCup 2008.

This paper presents the base station developed for the robotic soccer team CAMBADA. Being this application of extreme importance for the team, the requirements and specifications of the project had to be carefully analyzed. These issues are presented in Section II. Section III presents the software architecture of the base station. Section IV describes the implementation details, in particular the classes developed and the information update mechanism. Finally, Section V draws some conclusions.

II. Requirements and Specifications

The base station is the software application responsible to provide automatic processing of soccer game refereeing events (coming from the referee box) and to allow high level monitoring and control of the robots internal states. This application must be able to show all relevant information of the robots, namely position in the field, velocity, battery charge, among other information, and send basic commands/information to the robots, in particular the run and stop commands, play mode, etc. Besides that, the base station has a fundamental role in a game, while receiving the commands from the referee box, translating them to internal game states and broadcasting the results to the robots. During a game, no human interference is allowed except for removing malfunctioning robots and re-entering robots in the game.

The role of the base station during the game led to the fulfillment of some requirements, being the most important the following:

- Reliability / Stability: during the game, the base station is not accessible for human interaction of any kind and thus, it has to be a very robust application.
- Usability: the information displayed in the base station should be easy to interpret, allowing, for instance, for a fast detection of a problem in a robot. Moreover, it should be possible to choose different levels of details in the displayed information.
- Usability in the team development stage: the base station has to be easy to use, allowing an intuitive management of the robots.
- Adaptability: a robotic soccer team is in a permanent development stage, which may lead to significant changes within a short period of time.

These requirements led to the following specifications:

- Modular construction: a robot, for instance, should be an instantiable entity in the application in order, for instance, to allow the inclusion of more robots in the team. This modular construction leads to a progressive development, allowing the test of each module separately, thus increasing the reliability and stability of the whole application.
- Multi-windows solution: the application should be a multi-window environment, allowing the user to choose between different levels of information. At least, three different levels of information should be provided: a work level that presents the controls of the robots and basic status information; a visual level that presents visual information of the position of the robots and, finally a detailed level that shows all the information related to the robots.
- Robust communication skills: the correct operation of the team during the game is fully dependent on the communication between the robots, the base station and the referee box. Hence, the base station should provide a robust communication layer.
- Automatic processing of the game states: the base station should process the commands received from the referee box allowing the robots to change their internal game states accordingly. One specific action should be the changing of the field side at half time.
- Adaptable windows geometry: the multi-windows system should adapt to monitors with different resolutions. According to the new (upcoming) MSL rules, the teams base stations must use an external monitor provided by the organizing committee.

In order to use the base station during the team development phase, the following specifications should also be met:

- Local referee box: the base station should provide an interface widget to emulate a real referee box in order to simulate events of a real game.
- Manual positioning of the robots in the field: it should be possible to move the robots, through a mouse driven operation, to a specific position on the field.
- Manual role assignment: acting as a cooperative team, each robot has a specific role which is, during a real game, dynamically assigned. In the development phase, it should be possible to statically assign a role to a specific robot.

III. System Architecture

The central software component in the architecture of the CAMBADA robots is the Real-Time Database (RTDB) [10]. The RTDB is a distributed data structure by means of which all the team agents share their world models. It is updated and replicated in all robots, base station and coach (the coach is a software application, that running in the same computer of the base station, that can, in some specific situations, assign the robots roles). The RTDB contains all information related to the robots and game, namely positions in the field, roles, behaviors, game sates, etc.

All the interaction between base station and the CAMBADA robots is performed through the RTDB. Due to this fact, one of the central modules in the

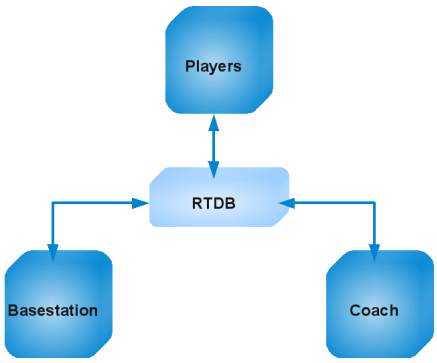

Fig. 1 - All the commands and information exchanged between all the team agents is accomplished through the RTDB.

base station architecture is the one responsible for handling the communication with RTDB, named as `UpdateWidget` (Fig. 6). It creates an abstraction layer to the RTDB interaction mechanism.

Another module in the system is the `RobotWidget` (Fig. 2). This module is responsible to send commands to the robots and shows robots information, such as position in the field, battery charge, etc.

Fig. 2 - The `RobotWidget` module showing the information and the commands available for one robot.

The `FieldWidget` module is responsible to create a visual interface to the user (Fig. 3). This module draws the field and the robots and can provide a simple way to control the position of the robots using a mouse driven interaction.

The `RobotInfoWidget` module shows all the information related to the robots stored in RTDB (Fig. 4).

Another important module in the base station system is the `RefBoxWidget`. This module provides a local referee box for test purposes and also manages the information received from a real referee box during a game. It provides an easy interface to configure the connection to the external referee box and allows the temporary suspension of the handling of the referee box messages maintaining the connection.

The base station implements a three windows based solution, allowing the user to choose the better set of information/actions that he wants to see/perform.

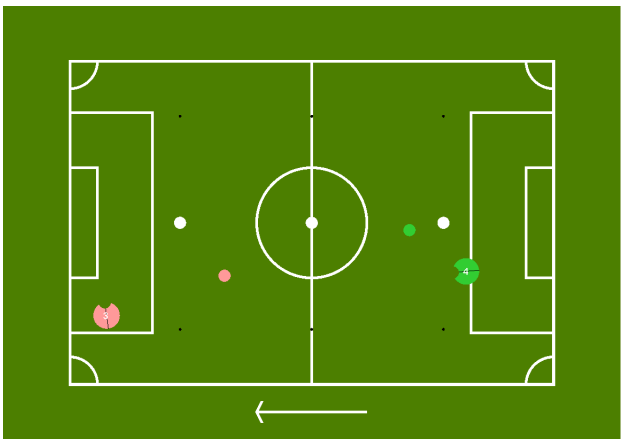

Fig. 3 - The `FieldWidget` showing the position of two robots and the position of the ball as estimated by each of them.

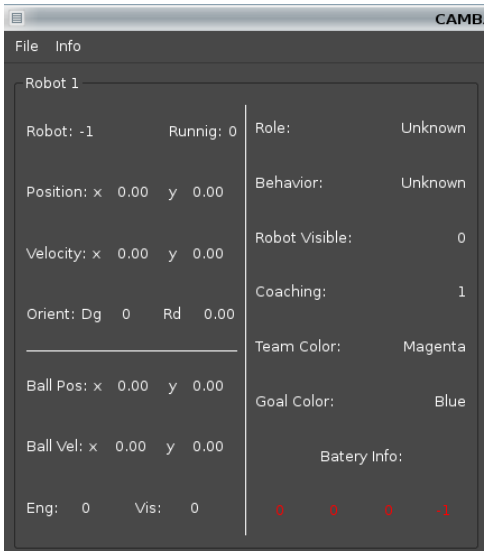

Fig. 4 - The information of one robot in the base station information window. This window shows the information of all robots of the team.

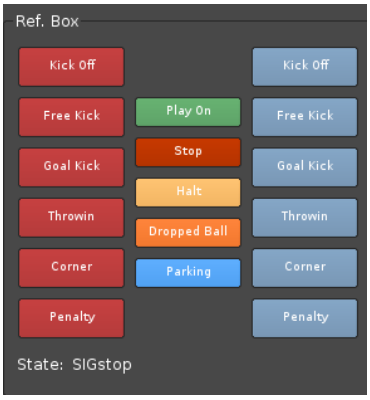

Fig. 5 - The `RefBoxWidget` module providing a local referee box for test purposes.

When the application starts, it shows one window, the `MainWindow` (Fig. 7). The user can, at any time, open the other windows: the `FieldWindow` (Fig. 8) and the `InfoWindow` (Fig. 4). These two windows can be

opened and closed independently. This solution allows the user to have more than one screen with different windows on each one. If the `MainWindow` is closed, all the other windows will also be closed too.

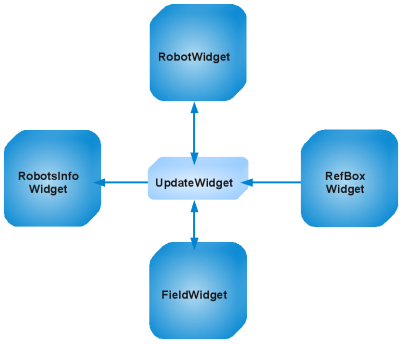

Fig. 6 - The most important base station modules.

IV. IMPLEMENTATION

The base station project was developed in C++ in a Linux environment using the Qt4 libraries [11]. The Qt4 libraries were developed in C++ and provide graphical and communication functions. This section describes the most important classes developed for the base station application and the most relevant issues in the development process.

A. Base station Classes

In the base station project, the most important classes that have been implemented are the following:

`UpdateWidget`: responsible for the connection between the base station and the RTDB. In this class, it is declared a local image of the RTDB that is passed, through a reference based mechanism, to the other classes in the project. This class is responsible for the connection to the RTDB and for the update of the information present in the local data structure.

`FieldWidget`: implemented using the integrated class "GLWidget", offered by the Qt4 library, that merges the Qt4 communication mechanisms and the graphical engine OpenGL. This class is responsible for drawing the field and the robots. This class implements a mechanism that allows the user to select a robot and pass a new point in the field to where the robot should move, using a mouse based mechanism.

`RobotWidget`: this class implements all the visual elements, like buttons, combo boxes, etc. related to the robot information and control. It is also responsible to process the information/commands related to the robots. This class is instantiated as many times as the total number of robots existing in the team.

`RobotInfoWidget`: this class is responsible for the visual elements that show all the information stored in the RTDB. Like `RobotWidget`, this class is instantiated as many times as the total number of robots existing in the team.

`RefBoxWidget`: this class is responsible for the creation, handling and destruction of the connection between the base station and the external referee box during a game. This class is also responsible for processing the game information and to perform the change of the team side at half time. This class also implements a local referee box.

`MainWindowWidget`: this is the application main class. It constructs the other classes, handles the mechanism of communication between UpdateWidget and the other classes and implements all the visual elements concerning team commands (coordinating all the robots at same time). This class manages the other windows.

`FieldWindowWidget`: this class implements the visual elements of the `FieldWindow`.

`InfoWindowWidget`: this class implements the visual elements of the `InfoWindow`.

B. UpdateWidget Mechanism

This mechanism allows sharing the same memory space with all modules in the application. It was implemented using the concept of parent and child, where a parent shares, with its children, the pointer to the memory space.

The `UpdateWidget` includes the method `DB_Robot_Info *get_info_pointer(void)` that returns the pointer to the structure where a local image of RTDB is stored. In all modules that interact with this information, there is a method named `void get_info_pointer( DB_Robot_Info* )` which has to be called to give access to the structure information.

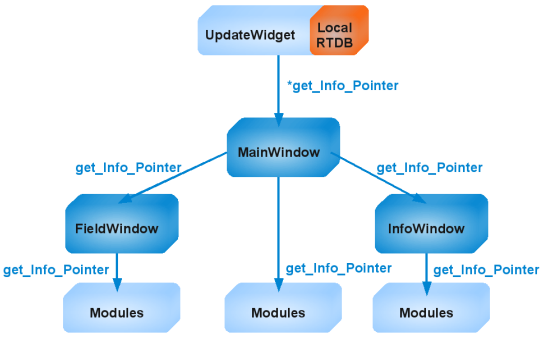

Fig. 9 - The `UpdateWidget` references mechanism.

Figure 9 shows the process of passing the references in all main widgets of the application.

The process begins in the `MainWindow` object. After calling the constructor of the `UpdateWidget`, the method `get_info_pointer` from `UpdateWidget` is called. This function returns a pointer to the memory space. After that, the `MainWindow` class passes this information to all its internal modules and other classes. All the classes pass the pointer to their children.

The implementation of this mechanism rises two main questions:

Fig. 7 - The `MainWindow` of the base station application.

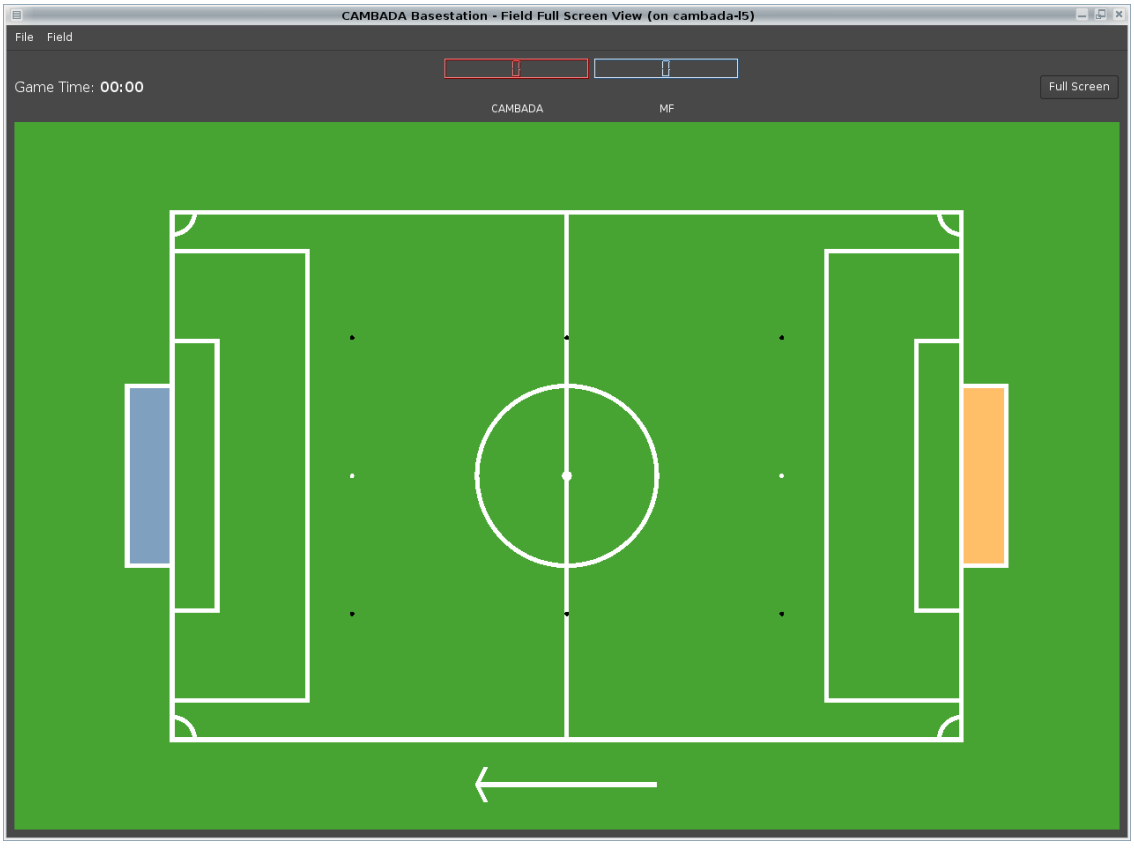

Fig. 8 - The `FieldWindow` of the base station application.

Why is this mechanism not included in the constructor of each class?
To include some classes in the design procedure it is mandatory that they have default constructors with predefined input parameters.

Why doesn't this mechanism has problems of mutual exclusion?
All interaction with the memory space is performed inside slot methods (Qt4 mechanism that responds to a specific signal) which guarantees the mutual exclusion (thread-safe) [12].

C. *RefBoxWidget incoming messages handling mechanism*

One important issue in processing the messages coming from the referee box is to guarantee the consistency of the information in all robots. It is very important that all robots share the same game states (play mode). This is guaranteed by the broadcast mechanism of the RTBD [13], [14].

The internal game states implemented in the CAMBADA are: `Start`, `Stop`, `DropBall`, `OurKickOff`, `TheirKickOff`, `OurPenalty`, `TheirPenalty`, `OurFreeKick`, `TheirFreeKick`, `OurGoalKick`, `TheirGoalKick`, `OurThrowIn`, `TheirThrowIn`, `OurCornerKick` and `TheirCornerKick`.

The base station has the notion of the team color and compares all orders to decide which is the next internal game state. An order like `Magenta Free Kick` could be an `OurFreeKick` if the team has the Magenta color. However, if the team color is cyan, the internal game state will be `TheirFreeKick`.

The referee orders could be classified into two classes: `Game State` orders and `Game Status` orders. The `Game State` orders are concerned with the state of the game. There are orders like `Start, Stop, MagentaGoalKick`, etc. The `Game State` orders have some special requirements to be processed. The order of reception is important and, after each order, the robots have to be informed of each game state. This is more relevant in case of more than one command is received in the same referee box message. However, a status message doesn't have these requirements. If more than one command is sent in the same message, the order of reception will be irrelevant. The algorithm implemented in `RefBoxWidget` reflects that.

V. CONCLUSIONS

The application described in this paper was used in the Portuguese Robotics Open "Robótica 2008" where CAMBADA team was, for the second time, national champion. Moreover, it was also used in world championship "RoboCup 2008" where CAMBADA team has won, for the first time, the world champion title. During these events, the application showed a high level of stability and reliability that was identified as a special requirement in the beginning of this project. Besides that, all other requirements were fulfilled, concluding that the base station was an important agent in the success of CAMBADA, contributing to the excellent results obtained by the team in the last year.

ACKNOWLEDGMENT

This work was supported in part by the FCT project PTDC/EIA/70695/2006.

REFERENCES

[1] MSL Technical Committee 1997-2008, "Middle Size Robot League Rules and Regulations for 2008", 2007.

[2] J. L. Azevedo, B. Cunha, and L. Almeida, "Hierarchical distributed architectures for autonomous mobile robots: a case study", in *Proc. of the 12th IEEE Conference on Emerging Technologies and Factory Automation, ETFA2007*, Greece, 2007, pp. 973–980.

[3] A. J. R. Neves, G. Corrente, and A. J. Pinho, "An omnidirectional vision system for soccer robots", in *Proc. of the EPIA 2007*. 2007, vol. 4874 of *Lecture Notes in Artificial Inteligence*, pp. 499–507, Springer.

[4] A. J. R. Neves, D. A. Martins, and A. J. Pinho, "A hybrid vision system for soccer robots using radial search lines", in *Proc. of the 8th Conference on Autonomous Robot Systems and Competitions, Portuguese Robotics Open - ROBOTICA'2008*, Aveiro, Portugal, april 2008, pp. 51–55.

[5] D. A. Martins, A. J. R. Neves, and A. J. Pinho, "Real-time generic ball recognition in RoboCup domain", in *Proc. of the 11th edition of the Ibero-American Conference on Artificial Intelligence, IBERAMIA 2008, IROBOT Workshop*, Lisbon, Portugal, october 2008, pp. 37–48.

[6] P. M. R. Caleiro, A. J. R. Neves, and A. J. Pinho, "Color-spaces and color segmentation for real-time object recognition in robotic applications", *Revista do DETUA*, vol. 4, no. 8, pp. 940–945, June 2007.

[7] B. Cunha, J. L. Azevedo, N. Lau, and L. Almeida, "Obtaining the inverse distance map from a non-svp hyperbolic catadioptric robotic vision system", in *Proc. of the RoboCup 2007*, Atlanta, USA, 2007.

[8] J. Silva, N. Lau, J. Rodrigues, and J. A. Azevedo, "Ball sensor fusion and ball interception behaviours for a robotic soccer team", in *Proc. of the 11th edition of the Ibero-American Conference on Artificial Intelligence, IBERAMIA 2008, IROBOT Workshop*, Lisbon, Portugal, october 2008, pp. 25–36.

[9] L. S. Lopes N. Lau and G. A. Corrente, "CAMBADA: Information sharing and team coordination", in *Proc. of the 8th Conference on Autonomous Robot Systems and Competitions, Portuguese Robotics Open - ROBOTICA'2008*, Aveiro, Portugal, april 2008, pp. 27–32.

[10] L. Almeida, F. Santos, T. Facchinetti, P. Pedreira, V. Silva, and L. S. Lopes, "Coordinating distributed autonomous agents with a real-time database: The CAMBADA project", in *Proc. of the 19th International Symposium on Computer and Information Sciences, ISCIS 2004*. 2004, vol. 3280 of *Lecture Notes in Computer Science*, pp. 878–886, Springer.

[11] Trolltech Co, "Qt cross-platform application framework". **URL:** *http://www.trolltech.com*

[12] Trolltech Co, "Signals and slots across threads". **URL:** *http://doc.trolltech.com/4.0/threads.html#signals-and-slots-across-threads*

[13] F. Santos, L. Almeida, P. Pedreiras, L.S. Lopes, and T. Facchinetti, "An Adaptive TDMA Protocol for Soft Real-Time Wireless Communication among Mobile Autonomous Agents", in *Proc. of the Int. Workshop on Architecture for Cooperative Embedded Real-Time Systems, WACERTS 2004*, 2004, pp. 657–665.

[14] F. Santos, G. Corrente, L. Almeida, N. Lau, and L.S. Lopes, "Selfconfiguration of an Adaptive TDMA wireless communication protocol for teams of mobile robots", in *Proc. of the 13th Portuguese Conference on Artificial Intelligence, EPIA 2007*, 2007.