Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots

Frederico Santos[1], Gustavo Currente[2], Luís Almeida[2], Nuno Lau[2] and Luís Seabra Lopes[2]

[1] DEE, Instituto Superior de Engenharia de Coimbra, Rua Pedro Nunes, 3030-199 Coimbra, Portugal
fred@isec.pt
[2] IEETA - DETI, Universidade de Aveiro, Campus Universitário de Santiago, 3810-193 Aveiro, Portugal
{gustavo, lda, nunolau, lsl}@ua.pt

Abstract. Interest on using mobile autonomous agents has been growing, recently, due to their capacity to cooperate for diverse purposes, from rescue to demining and security. However, such cooperation requires the exchange of state data that is time sensitive and thus, applications should be aware of data temporal coherency. This paper describes the communication and coordination architecture of the agents that constitute the CAMBADA (Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture) robotic soccer team developed at the University of Aveiro, Portugal. This architecture is built around a partially replicated real-time database refreshed in the background, transparently to the higher software layers. The paper presents the communication mechanisms that were devised to support the real-time database management and focuses on the self-configuration of the protocol, according to the current number of active team members.

1 Introduction

Coordinating several autonomous mobile robotic agents in order to achieve a common goal is an active topic of research [3]. This problem can be found in many robotic applications, either for military or civil purposes, such as search and rescue in catastrophic situations, demining or maneuvers in contaminated areas.

The technical problem of building an infrastructure to support the perception integration for a team of robots and subsequent coordinated action is common to the above applications. One recent initiative to promote research in this field is RoboCup [5] where several autonomous robots have to play football together as a team, to beat the opponent. We believe that researching ways to solve the perception integration problem in RoboCup is also very relevant to real-world applications.

Currently, the requirements posed on such teams of autonomous robotic agents have evolved in two directions. On one hand, robots must move faster and with accurate trajectories to close the gap with the dynamics of the processes they interact with, e.g., a ball can move very fast. On the other hand, robots must interact more in order to develop coordinated actions more efficiently, e.g., only the robot closer to the ball should try to get it while other robots should move to appropriate positions. The former requirement demands for tight closed-loop motion control while the latter demands for an appropriate communication system that allows building a global information base to support cooperation.

In this paper we describe the communication and coordination architecture of the robotic agents that constitute the CAMBADA middle-size robotic soccer team of the University of Aveiro, Portugal, which is well suited to support the requirements expressed above. The software architecture is based on a real-time database in which the state values of other agents are updated transparently to the higher software layers, using an adequate communication protocol. This paper focuses on such protocol that dynamically adapts to the conditions of the communication channel and to the current number of active agents in the team. Particularly, some results are shown that illustrate the latter self-configuration capability.

2 Computing/Communications Architecture

The computing architecture of the robotic agents is layered with two levels as illustrated in Fig. 1. The higher level is built around a main processing unit that handles both the external communication with other agents as well as the local vision system. A distributed low-level sensing/actuating system handles the robot attitude (holonomic motion control), odometry, kicking and power monitoring. The latter one is out of the scope of this paper.

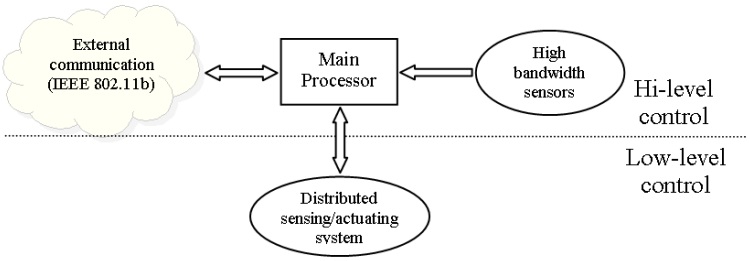

Fig. 1. CAMBADA robotic architecture

The main processing unit is currently implemented on a laptop that delivers sufficient computing power while offering standard interfaces to connect the other

systems, namely USB, FireWire and WiFi. The wireless interface is either built-in or added as a PCMCIA card. The laptop runs the Linux operating system with the timeliness support necessary for time-stamping, periodic transmissions and task temporal synchronization provided by a specially developed user-level real-time scheduler, the Pman – Process Manager [1]. This approach provides a sufficient timeliness support for soft real-time applications, such as multiple robot coordination, and allows profiting from the better development support provided by general purpose operating systems [2].

The agents that constitute the team communicate with each other by means of an IEEE 802.11b wireless network as depicted in Fig. 2. The communication is managed, i.e., using a base station, and it is constrained to using a single channel, shared by, at least, both teams in each game. In order to improve the timeliness of the communications, our team uses a further transmission control protocol that minimizes collisions of transmissions within the team. Each robot regularly broadcasts its own data while the remaining ones receive such data and update their local structures. Beyond the robotic agents, there is also a coaching and monitoring station connected to the team that allows following the evolution of the robots status on-line and issuing high level team coordination commands.

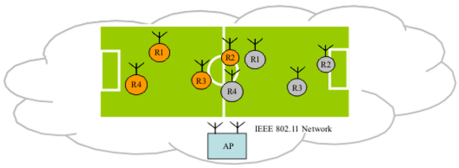

Fig. 2. Global communications architecture

3 RTBD - The Real-Time Database

Similarly to other teams [4, 6], our team software architecture emphasizes cooperative sensing as a key capability to support the behavioral and decision-making processes in the robotic players. A common technique to achieve cooperative sensing is by means of a *blackboard* [7, 8], which is a database where each agent publishes the information that is generated internally and that maybe requested, by others. However, typical implementations of this technique seldom account for the temporal validity (coherence) of the contained information with adequate accuracy. This is a problem when robots move fast because their state information degrades faster, too. Without adequate refreshing, the data in a blackboard may easily lose temporal validity thus becoming too old to be useful. Another problem of typical implementations is that they are based on the client-server model and thus, when a robot needs a datum, it has to communicate with the server holding the blackboard, introducing an undesirable delay. To avoid this

delay, we use two features: firstly, the dissemination of the local state data is carried out using multicast, according to the producer-consumer cooperation model, secondly, we replicate the blackboard according to the *distributed shared memory* model [9]. In this model, each node has local access to all the process state variables that it requires. Those variables that are remote have a local image that is updated automatically in the background by the communication system (Fig. 3).

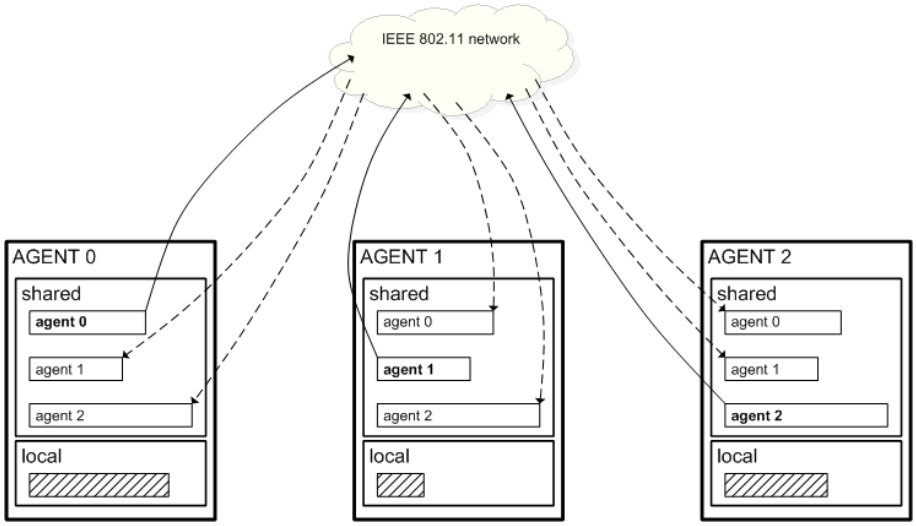

Fig. 3. Each agent broadcasts periodically its subset state data that might be required by other agents

We call this replicated blackboard the Real-Time DataBase (RTDB), similarly to the concept presented in [10], which holds the state data of each agent together with local images of the relevant state data of the other team members. A specialized communication system triggers the required transactions at an adequate rate to guarantee the freshness of the data.

4 Communication Among Agents

As referred in section 2, agents communicate using an IEEE 802.11 network, sharing a single channel with the opposing team and using managed communication (through the access point). This raises several difficulties because the access to the channel cannot be controlled [11] and the available bandwidth is roughly divided by the two teams.

Therefore, the only alternative left for each team is to adapt to the current channel conditions and reduce access collisions among team members. This is

achieved using a dynamic adaptive TDMA transmission control, with a predefined round period called *team update period (T_{tup})* that sets the responsiveness of the global communication. Within such round, there is one single slot allocated to each running team member so that all slots in the round are separated as much as possible (Fig. 4). This allows calculating the target inter-slot period T_{xwin} as T_{tup}/N, where N is the number of running agents. The transmissions generated by each running agent are scheduled within the communication process, according to the production periods specified in the RTDB records. Currently a rate-monotonic scheduler is used. When the respective TDMA slot comes, all currently scheduled transmissions are piggybacked on one single 802.11 frame and sent to the channel. The required synchronization is based on the reception of the frames sent by the other agents during T_{tup}. With the reception instants of those frames and the target inter-slot period T_{xwin} it is possible to generate the next transmission instant. If these delays affect all TDMA frames in a round, then the whole round is delayed from then on, thus its adaptive nature. Fig. 5 shows a TDMA round indicating the slots allocated to each agent and the adaptation of the round duration. The adaptive TDMA protocol was first proposed by the authors in [12].

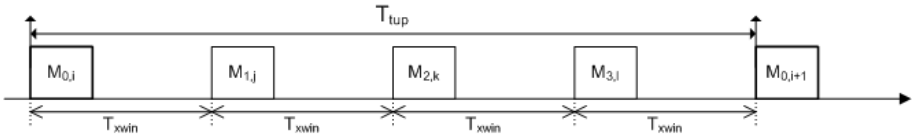

Fig. 4. TDMA round

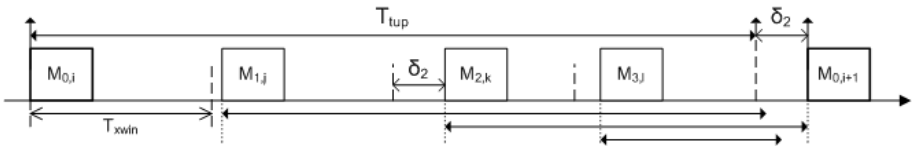

Fig. 5. Adaptive TDMA round

When a robot transmits at time t_{now} it sets its own transmission instant $t_{next} = t_{now} + T_{tup}$, i.e. one round after. However, it continues monitoring the arrival of the frames from the other robots. When the frame from robot k arrives, the delay δ_k of the effective reception instant with respect to the expected instant is calculated. If this delay is within a validity window $[0, \Delta]$, with Δ being a global configuration parameter, the next transmission instant is delayed according to the longest such delay among the frames received in one round (Fig. 5), i.e.,

$$t_{next} = t_{now} + T_{tup} + max_k(\delta_k)$$

On the other hand, if the reception instant is outside that validity window, or the frame is not received, then δ_k is set to 0 and does not contribute to update t_{next}.

The practical effect of the protocol is that the transmission instant of a frame in each round may be delayed up to Δ with respect to the predefined round period T_{tup}. Therefore, the effective round period will vary between T_{tup} and $T_{tup} + \Delta$. When a robot does not receive any frame in a round within the respective validity windows, it updates t_{next} using a robot specific configuration parameter β_k in the following way

$$t_{next} = t_{now} + T_{tup} + \beta_k \quad with \quad 0 \leq \beta_k \leq \Delta$$

This is used to prevent a possible situation in which the robots could all remain transmitting but unsynchronized, i.e. outside the validity windows of each other, and with the same period T_{tup}. By imposing different periods in this situation we force the robots to resynchronize within a limited number of rounds because the transmissions will eventually fall within the validity windows of each other.

One of the limitations of the adaptive TDMA protocol as proposed in [12] is that the number of team members was fixed, even if the agents were not active, causing the use of T_{xwin} values smaller than needed. Notice that a smaller T_{xwin} increases the probability of collisions in the team. Therefore, a self-configuration capability was added to the protocol, to cope with variable number of team members. This is the specific mechanism proposed in this paper, which supports the dynamic insertion / removal of agents in the protocol. Currently, the T_{tup} period is still constant but it is divided by the number of running agents at each instant, maximizing the inter-slot separation between agents T_{xwin} at each moment.

However, the number of active team members is a global variable that must be consistent so that the TDMA round is divided in the same number of slots in all agents. To support the synchronous adaptation of the current number of active team members a membership vector was added to the frame transmitted by each agent in each round, containing its perception of the team status.

When a new agent arrives it starts to transmit its periodic information in an unsynchronized mode. In this mode all the agents, including the new one, continue updating its membership vector with the received frames and continue refreshing the RTDB shared areas, too. The T_{xwin} value, however, is not yet adapted and thus the new agent has no slot in the round. When all the team members reach the same membership vector, the number of active team members is updated, so as the inter-slot period T_{xwin}. The protocol enters then in the scan mode in which the agents, using their slightly different values of T_{tup}, rotate their relative phases in the round until they find their slots. From then on, all team members are again synchronized. The removal of an absent agent uses a similar process. After a predefined number of rounds without receiving frames from a

given agent, each remaining member removes it from the membership vector. The change in the vector leads to a new agreement process similar to described above.

Fig. 6 shows an example of the self-reconfiguration process with the dynamic insertion and removal of agents. It shows the instants at which the packets from the several agents in a team are received in a monitoring station, relative to the start of the round in an arbitrary agent (agent 2 in this case). The line on top shows the reception instants of that agent, which give us an indication of the effective TDMA round duration. Before point A, agent 2 is alone, using a TDMA round with 1 single slot, and at that point agent 4 joins the team and starts transmitting. Agent 2 detects these transmissions and divides the TDMA round in 2 slots, one for each agent. Naturally, the transmissions of agent 4 are outside the respective validity window, thus agent 4 uses a sliding relative phase until it reaches the right slot, at which point it stays synchronized with the team (near flat portions of the graph). At point B, agent 4 left the team, i.e., stopped transmitting. Agent 2 detected this situation and reconfigured the TDMA round to 1 single slot again. The remaining situations are all similar, with agent 4 re-joining at point C and agent 5 at point D, who leave the team at points E and F, respectively. From D to E the TDMA round is configured to 3 slots and after the withdrawal of agent 4, it is reconfigured to 2 slots again. Notice that the mechanisms are fully distributed and all agents execute exactly the same code.

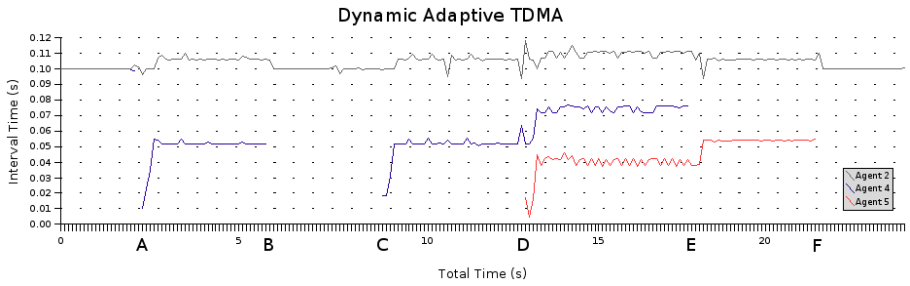

Fig. 6. Self-configuration of the slot time according to the number of running agents

5 Coordinating Multiple Soccer Agents

The purpose of the communication protocol described above is to support the management of the RTDB, which is the central element for sharing information and thus for coordination of the team of agents. In this section we present a brief reference to some of the coordinated behaviors that are currently implemented on top of the RTDB, thus highlighting the effectiveness of the communication.

The team can be in several different play modes, from kick off to free kick, throw in, corner kick, etc., decided by the referee, which are broadcast among the team by the remote station through the RTDB.

The RTDB also supports the integration of the individual agent perceptions to improve their knowledge about the current positions and velocities of the others robots and of the ball. It is very important for our coordination model to keep an accurate estimation of the absolute position of the ball by each robot. The role assignment algorithm is based on the absolute position of the robot, its team mates and ball. Each robot determines its self localization and ball position through its local vision system and shares it with the others through the RTDB.

Communication is also used to convey the coordination status of each agent allowing robots to detect uncoordinated behaviors, for example, several robots with the same exclusive role, and to correct this situation reinforcing the reliability of coordination algorithms.

6 Conclusion

Cooperating robots is a field currently generating large interest in the research community. RoboCup is one example of an initiative developed to foster research in that area.

This paper described the computing and communication architecture of the CAMBADA middle-size robotic soccer team being developed at the University of Aveiro. Such architecture is based on a partially replicated real-time database, i.e., the RTDB, which includes local state variables together with images of remote ones. These images are updated transparently to the application software by means of an adequate real-time management system. Moreover, the RTDB is accessible to the application using a set of non-blocking primitives, thus yielding a fast data access.

Earlier work from the authors led to the development of a wireless communication protocol that reduces the probability of collisions among the team members. The protocol called adaptive TDMA, adapts to the current channel conditions, particularly accommodating periodic interference patterns. In this paper the authors extended that protocol with on-line self-configuration capabilities that allow reconfiguring the slots structure of the TDMA round to the actual number of active team members, further reducing the collision probability. This paper ends with a brief reference to global team coordination based on the RTDB concept, using the described communication protocol.

Future work includes the further dynamic reconfiguration of the TDMA round interval, i.e., the team update period, according to the communication channel load and current number of agents in the team.

References

1. Pedreiras, P., Almeida, L.: Task management for soft real-time applications based on general purpose operating systems. In: Proceedings of the 9th Brazilian Workshop on Real-Time Systems, Belém, Brazil (May 2007)

2. Gopalan, K.: Real-time support in general purpose operating systems. Technical report (2001)
3. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press (1999)
4. Dietl, M., Gutmann, J.S., Nebel, B.: Cooperative sensing in dynamic environment. In: Proceedings of the IROS2001 International Conference on Intelligent Robots and Systems, Maui, Hawaii (October 2001)
5. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: Robocup: The robot world cup initiative. In: Proceedings of the IJCAI-95 Workshop on Entertainment and AI/Alife, Montreal (August 1995)
6. Weigel, T., Gutmann, J.S., Nebel, B., Muller, K., Dietl, M.: Cs freiburg: Sophisticated skills and effective cooperation. In: Proceedings of the EEC01 European Control Conference, Porto, Portugal (September 2001)
7. Erman, L.D., Hayes-Roth, F., Lesser, V.R., Reddy, R.: The hersay-ii speech understanding system: Integrating knowledge to resolve uncertainty. ACM Computing Surveys **12**(2) (1980) 213–253
8. Carver, N., Lesser, V.: The evolution of blackboard control architectures. Technical Report UM-CS-1992-071 (1992)
9. Milutinovic, V., Stenstrom, P.: Special issue on distributed shared memory systems. In: Proceedings of the IEEE. Volume 87. (March 1999) 399–404
10. Kopetz, H.: Real-Time Systems Design Principles for Distributed Embedded Applications. Kluwer Academic Pub (1997)
11. Decotignie, J.D., Dallemagne, P., El-Hoiydi, A.: Architecture for the interconnections of wireless and wireline fieldbus. In: Proceedings of the FeT01 IFAC Conference on Fieldbus Technologies, Nancy, France (November 2001)
12. Santos, F., Almeida, L., Pedreiras, P., Lopes, L.S., Facchinetti, T.: An adaptive tdma protocol for soft real-time wireless communication among mobile autonomous agents. In: Proceedings of the WACERTS04 Workshop on Architectures for Cooperative Embedded Real-Time Systems (in conjunction with RTSS2004 - 3rd International Symposium on Robotics and Automation), Lisbon, Portugal (December 2004)