

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

Frederico Santos
DEE - ISEC
Inst. Politécnico de Coimbra, Portugal
fred@isec.pt

Luís Almeida, Luís Seabra Lopes
IEETA - DETI
Universidade de Aveiro, Portugal
{lda, 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 while achieving timeliness with RF communication is intrinsically difficult due to the openness of the medium. This paper describes a communication layer that improves the timeliness of periodic data exchanges among the team reducing the chances of lost packets caused by collisions between team members. In particular, the paper extends a previous proposal for an adaptive TDMA protocol with new self-configuration capabilities according to the current number of active team members. This feature further reduces the likelihood of collisions within the team. Several experimental results with an actual system implementation show the effectiveness of the proposed solution.

1 Introduction

Coordinating several autonomous mobile robotic agents in order to achieve a common goal has been an active topic of research for more than a decade [8]. 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 initiative that was created to promote research in this field is RoboCup [5] where several autonomous robots have to play soccer together as a team, to win a match against another team of autonomous robots.

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 knowledge base to support cooperation.

In this paper we describe the wireless communication layer of the robotic agents that constitute the CAM-BADA middle-size robotic soccer team of the University of Aveiro, Portugal. The software architecture is based on a real-time database [1] in which the state values of other agents are updated transparently to the application, using the proposed communication layer. This paper focuses on the protocol that dynamically adapts to the conditions of the communication channel and to the current number of active agents in the team. The main contribution of the paper is the self-configuration capability with respect to the number of active team members. The protocol follows a modified TDMA approach, called *adaptive TDMA* [7], which synchronizes on message receptions and does not require clock synchronization. Moreover, the protocol is designed to cope with uncontrolled load in the channel generated by nodes external to the team. The mechanism described in this paper reconfigures the TDMA round automatically and in a fully distributed way according to the current number of active agents with the purpose of minimizing the possibility of collisions among team members.

The paper is structured as follows, Section 2 briefly presents the computing and communication architecture of the team. Section 3 presents the core of the paper, mainly in Subsection 3.2 that describes the self-reconfiguration mechanisms of the protocol. Section 4 presents an experimental validation of the devised mechanisms. Section 5 addresses other features and possible lines for future work that would enhance the framework. Finally, Section 6 discusses some related work and Section 7 concludes the paper.

2 Computing/Communication Architecture

The computing architecture of the robotic agents is illustrated in Figure 1. It is a hierarchical two-tier archi-

ture. The higher level (tier 2) is built around a main processing unit that handles the external communication with other agents, the local vision system and the robot behavior. A distributed low-level sensing/actuating system (tier 1) handles robot attitude, odometry, kicking and power monitoring. Tier 1 is out of the scope of this paper.

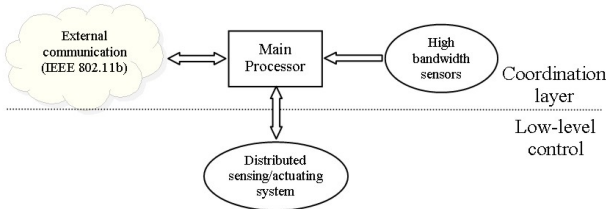


Figure 1. CAMBADA robotic architecture

The main processing unit is currently implemented on a laptop with a built-in wireless interface. 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 [6]. This approach provides 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 [3].

The team agents communicate with each other by means of an IEEE 802.11b wireless network as depicted in Figure 2. The communication is managed, i.e., using an Access Point (AP), 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.

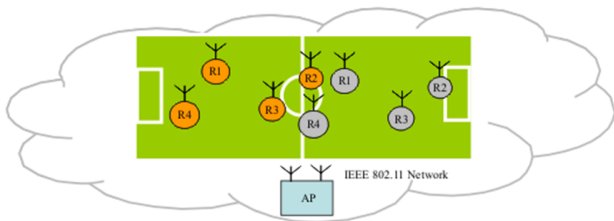


Figure 2. Communications environment

3 Communication Among Agents

As mentioned in Section 2, agents communicate through the AP using an IEEE 802.11 network, sharing

a single channel with the opposing team. This raises several difficulties because the presence of uncontrolled traffic in the channel is unavoidable. Conversely, there are also some benefits in terms of consistency in the team, which is enforced by the AP. An agent is considered reachable by the team when it has an active link with the AP and unreachable otherwise. This is a realistic application in many application scenarios. For example, for teams of surveillance robots within large indoor spaces, such as malls, it is normally feasible to provide an AP that guarantees the radio coverage of all robots. In demining applications, or even search and rescue, it is possible to place the AP on top of one of the robots deployed near the center of the operations area that will provide coverage for the other ones. In all these cases, it is also impossible to control the access to the channel and thus the technical solutions adopted must cope with such circumstance.

3.1 Adaptive TDMA

Since the load in the network cannot be totally controlled by the team, the only alternative left is to adapt to the current channel conditions and reduce access collisions among team members. This is achieved using an adaptive TDMA transmission control as proposed in [7]. The TDMA round period is set off-line and called *team update period* (T_{tup}), setting the responsiveness and the temporal resolution of the global communication. It is, thus, an application requirement. T_{tup} is divided equally by the number of team members generating the TDMA slot structure. With equal slots, if the agents transmit at the beginning of their slots, their transmissions are separated as much as possible (Figure 3). The target inter-slot period can be computed as

$$T_{xwin} = \frac{T_{tup}}{N} \quad (1)$$

where N is the number of team agents. Normally each robot will only use a fraction of its slot and the unused part constitutes leeway to accommodate the uncontrolled load. When the respective TDMA slot comes, all currently scheduled transmissions are piggybacked on one single 802.11 frame and sent to the channel. The presence of uncontrolled load may lead to a delay (δ) of the packet reception. Each agent uses this delay to compute its next transmission instant, thus adapting the effective TDMA round period (Figure 4).

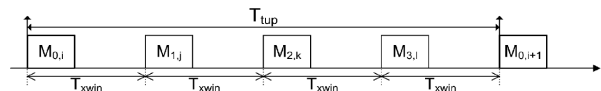


Figure 3. TDMA round

When a robot j transmits at time $t_{j,now}$ it sets its own transmission instant $t_{j,next} = t_{j,now} + T_{tup}$, i.e. one round after. However, it continues monitoring the arrival of the frames from the other robots. When the frame from robot i arrives, the delay δ_i of the effective reception instant with

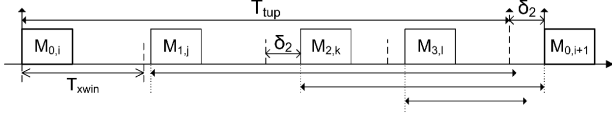


Figure 4. Adaptive TDMA round

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 (Figure 4), i.e.,

$$t_{j,next} = t_{j,now} + T_{tup} + \max(\delta_i)_{i=0..N-1, i \neq j \wedge \delta_i \leq \Delta}$$

On the other hand, if the reception instant is outside that validity window, then δ_i is set to 0 and does not contribute to update $t_{j,next}$.

The practical effect of the adaptation in 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 within $[T_{tup}, T_{tup} + \Delta]$.

When a robot does not receive any frame in a round within the respective validity windows, it updates $t_{j,next}$ using a robot specific configuration parameter β_j in the following way:

$$t_{j,next} = t_{j,now} + T_{tup} + \Delta + \beta_j$$

with $0 < \beta_j < \Delta$ and $\beta_j \neq \beta_i$ for $j \neq i$

This robot specific parameter is used to force different effective transmission periods, generating a sliding relative phase thus preventing 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.

The effectiveness of the adaptive TDMA approach has been experimentally validated and presented in [7].

3.2 Dynamic reconfiguration of the TDMA round

One of the limitations of the adaptive TDMA protocol as proposed in [7] is that the number of team members was fixed, even when the agents were not active, causing the use of T_{xwin} values smaller than needed, i.e., the slots were unnecessarily short since some of them were not used. Note that a smaller T_{xwin} reduces the leeway to accommodate delays caused by the uncontrolled traffic and thus increases the probability of loss of synchronization and of collisions within the team. Therefore, a self-configuration capability is 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 in a fully distributed way. Currently, the T_{tup} period is

still constant but it is divided by the number of running agents at each instant, designated K , with $K \leq N$, maximizing the inter-slot separation between agents T_{xwin} at each moment.

$$T_{xwin} = \frac{T_{tup}}{K} \quad (2)$$

The validity window used in the TDMA adaptation also becomes a function of K , i.e., $\Delta_K = T_{xwin} \times \epsilon$, $0 < \epsilon < 1$ where ϵ is a configuration parameter.

However, the number of active team members K 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 (see Figure 5). The number of fields in the membership vector is the maximum number of team mates N , defined offline.

One important aspect concerns the slots and nodes identification. In the previous adaptive-TDMA mechanism, each node had a unique ID ($j = 0..N - 1$) that was also used as slot ID. However, with the reconfiguration mechanism, the slots are dynamic and thus such direct ID mapping cannot be used. Thus, within this protocol the nodes are identified by a dynamic ID that corresponds to the ID of the slot they are assigned to ($k = 0..K - 1$). A simple rule is used to map the static unique node ID to a dynamic slot ID. The currently lowest static ID among the running robots is assigned slot 0, the following static ID is assigned to slot 1 and so on until the highest static ID among the running robots that is assigned to slot $K - 1$. This assignment is carried out every time there is a change in the slots structure, i.e., every time a robot joins or leaves the group.

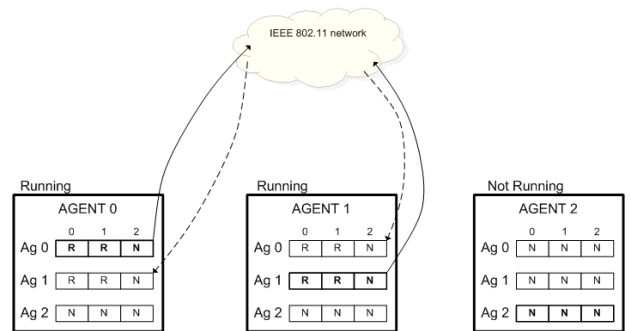


Figure 5. Dissemination of the membership vectors

The dynamic insertion / removal of agents is carried out in a distributed way based on the dissemination of the membership vectors. To manage this process, each agent runs the state machine presented in Figure 6 for each of the potential team members. For each agent the implemented state machine has 4 states:

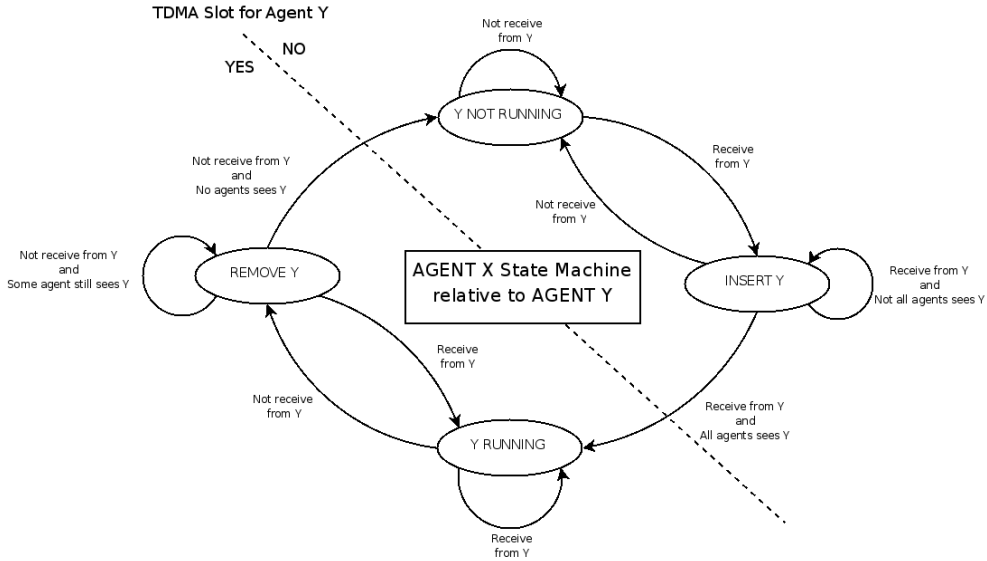


Figure 6. State-machine for capturing the state of another agent

- **Not running** – The agent is not powered up or is unreachable, i.e., not associated with the wireless AP;
- **Insert** – The agent started transmitting but has not yet been detected by all the current team mates. In this state the agent has no slot yet in the TDMA round and thus it is transmitting out of phase;
- **Running** – The agent has been detected by all team members and its own slot in the TDMA round has been created. All agents resynchronize and continue transmitting in their new slots;
- **Remove** – The agent is not transmitting or its message was not received, e.g., due to an error.

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. At this moment the new agent transmits its state as *Insert*. The T_{xwin} value, however, is not yet adapted and thus the new agent has no slot in the round and thus no dynamic ID. When all current team members agree on the existence of the new agent, i.e., they all detected its transmissions, the number of active team members K is updated, so as the inter-slot period T_{xwin} , the dynamic IDs are reassigned and the state of the new agent is changed to *Running*. The protocol enters then in the *scan mode* in which all agents, except the one using slot 0, use a slightly longer update period (see T_{next} below) to rotate their relative phase in the round until they find their slots. Once they find them, and from then on, all team members are again synchronized. Agent 0 (dynamic) is used as the reference TDMA round with which the others synchronize. This is important to avoid cliques, i.e., unsynchronized subgroups of nodes that are synchronized internally to each subgroup.

$$T_{next} = T_{now} + T_{tup} + \Delta_k + \beta_k \quad (3)$$

$$\text{with } \beta_k = \begin{cases} 0, & k = 0 \\ \beta, & k \geq 1 \end{cases}$$

In this equation, β is a configuration parameter that sets the speed of phase rotation within the *scan mode*. The time to reach synchronization depends on β and on the instant within the TDMA round in which the node starts scanning for its assigned slot. The number of rounds to synchronize should vary uniformly between 0, when the node, by chance, starts scanning exactly in the slot that was assigned to it, and a certain upper bound, R , that depends directly on β as follows:

$$R \leq \left\lceil \frac{T_{tup}}{\beta} \right\rceil \quad (4)$$

This is illustrated in Figure 7. Note that this expression is rather pessimistic since it assumes that the adaptive-TDMA mechanisms is constantly rotating at the maximum period of $T_{tup} + \Delta_K$, which would only occur under very strong uncontrolled load. However, when the uncontrolled load is low, the actual TDMA round will be close to T_{tup} and thus the maximum R will be closer to $\frac{T_{tup} - \Delta_K}{\beta}$.

The state machine of the joining agent, used at startup by all robots, is rather simple and it is depicted in Figure 8. Basically, a joining agent initially enters the *Start* state and waits for a given number of rounds without transmitting, currently 10 and using a preconfigured T_{tup} . This period of time allows the agent to determine whether there is a team already operating in the area and learn some of its operational parameters such as the actual T_{tup} , N and K . After such period it enters the *Insert* state in which it starts transmitting its data periodically but not synchronized with the team (note there is no slot yet created in the TDMA round for this node). Eventually the joining agent will receive at least one transmission from another team

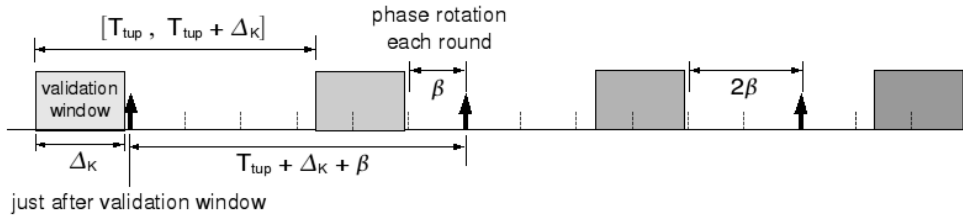


Figure 7. Phase rotation to achieve synchronization

member with a membership vector indicating that it has been acknowledged as a *Running* agent, meaning that the TDMA round has been reconfigured with a new slot. At this point, the agent enters in the *Running* state and stays there.

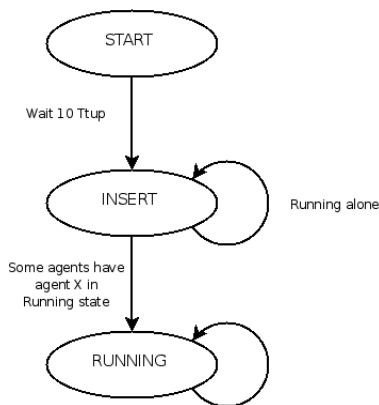


Figure 8. State machine of a joining agent

The removal of an absent agent uses a similar process. When in the previous rounds, currently in the last 5, no reception from an agent is detected, the state of that agent is changed to *Remove*. When all other running team members have also marked the agent as *Remove* then the agent is considered as *Not Running*, the number of active members K is decremented, the inter-slot period T_{xwin} is increased and the dynamic IDs reassigned. Again, the protocol enters in the scan mode until each agent finds its respective new slot.

Figure 9 shows an example of the reconfiguration process within the Adaptive TDMA protocol caused by the inclusion of a new agent in the team. Note that the arrows denote the instants of transmission and the blocks after transmissions are the respective membership vectors (represented with a larger scale to improve readability).

- **A, B** – Two team members (agents 0 and 1) are synchronized and transmitting their membership vectors with: agent 0 – *Running* (R), agent 1 – *Running* and agent 2 – *Not running* (N). The inter-slot interval is $T_{xwin} = \frac{T_{tup}}{2}$;
- **C** – A new member (agent 2) starts transmitting in *Insert* (I) state, without slot;
- **D, E** – Agent 0 and 1, in their last T_{tup} sense the new agent and update their membership vector, changing

agent 2's state to *Insert*;

- **F** – Agent 2 changes in its membership vector the states of agents 0 and 1 to *Running*;
- **G** – At this moment agent 2 is considered to be *Running* and the inter-slot interval is updated to $T_{xwin} = \frac{T_{tup}}{3}$;
- **H** – Agent 0 takes the same action as in **G**, but since it is the reference agent, the other agents will use this instant time as reference to synchronize;
- **I** – Agent 2 switches its own state to *Running*, since at least one team member considered it as *Running*, but it is out of sync and it will use the extra delay β ;
- **J** – Agent 1 denotes that it is out of sync and as in **I** it will use the extra delay β ;
- **K** – Agent 2 reaches its TDMA slot;
- **L** – In the last round agent 0 has not received any information from agent 1 and changes its state to *Remove* (r). The lack of transmission from agent 1 in a complete round is caused by the addition of β and Δ_k over the base update period T_{tup} ;
- **M** – Since instant **L** agent 0 receives again data from agent 1 and adjusts its state back to *Running*. Note that in *Remove* state the T_{xwin} parameter is not updated and from the point of view of the protocol, no changes occur;
- **N** – Agent 1 reaches its TDMA slot and from now all the agents are in their respective TDMA slots and thus synchronized.

4 Experimental validation

In order to verify the effectiveness of the proposed self-reconfiguration adaptive-TDMA protocol, several experiments were carried out with an actual implementation. The configuration parameters used (shown next) are reasonable values but sensitivity analysis of each of these parameters will be done in future work.

- $N = 10$
- $T_{tup} = 100ms$

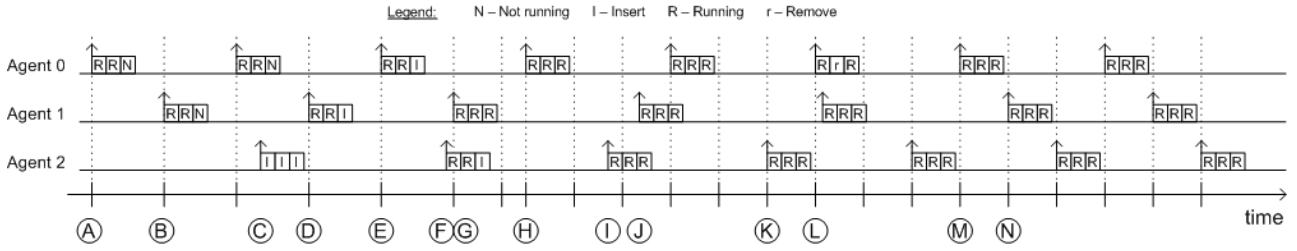


Figure 9. Timeline of the joining process for a new team member

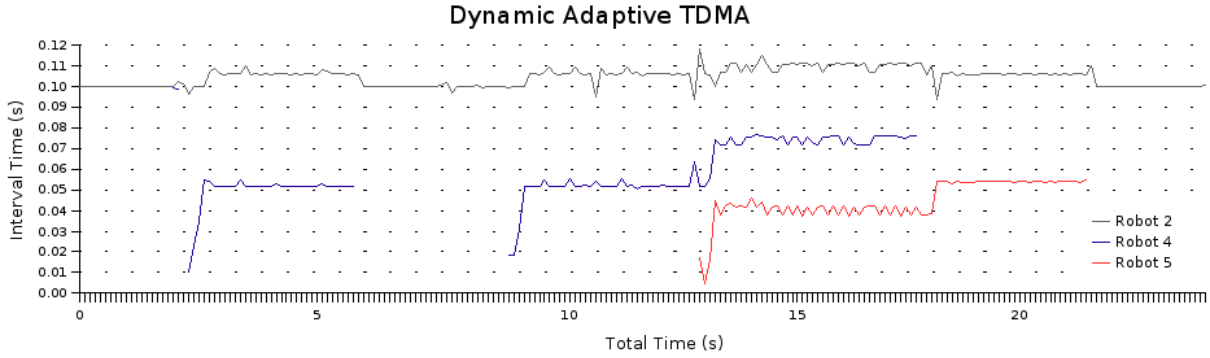


Figure 10. Self-configuration of the slot time according to the number of running agents

- $\epsilon = \frac{2}{3}$, thus $\Delta_K = \frac{T_{tup}}{K} \times \frac{2}{3}$
- $\beta = 8\% T_{tup} = 8ms$

4.1 Normal operation

This experiment aims at showing the normal operation of the proposed protocol with the dynamic insertion and removal of agents. Figure 10 shows a log of the actual self-reconfiguration process showing the time offset of the reception of each agent packets with respect to the start of the round in the reference agent 2 (dynamic ID 0). It clearly shows the adaptation mechanism, with agent 4 joining at second 2, leaving at second 6, rejoining at second 9 and leaving again at second 18. Similarly, agent 5 joins the team at second 13 and leaves at second 22. The almost vertical traces that appear when a new agent arrives show the scan mode, with the sliding relative phase.

4.2 Time to synchronize

In this experiment we verify the upper bound R for the time that a robot takes to synchronize. In this case, we used only two robots ($K = 2$), one permanently running and the other constantly forcing resynchronizations. This was achieved with a script killing the communication process and restarting it again, after a random amount of time, to guarantee an arbitrary start phase. In this case, Equation 4 results in 13 rounds. This, however, is pessimistic as referred when explaining Equation 4, since the two robots were alone in a relatively isolate area with practically no uncontrolled traffic. Therefore, in this case we can use the more accurate bound $R \leq \frac{T_{tup} - \Delta_K}{\beta}$ with $\Delta_K = 33.3ms$ resulting in 9 rounds. Figure 11 shows the histogram of the measured number of rounds needed

to synchronize after 1000 trials. As expected, the distribution is approximately uniform and the significant maximum is 8 rounds, thus below the expected upper bound R . There are a few occurrences of larger intervals to synchronize that are caused by seldom packet losses and other occasional uncontrolled delays.

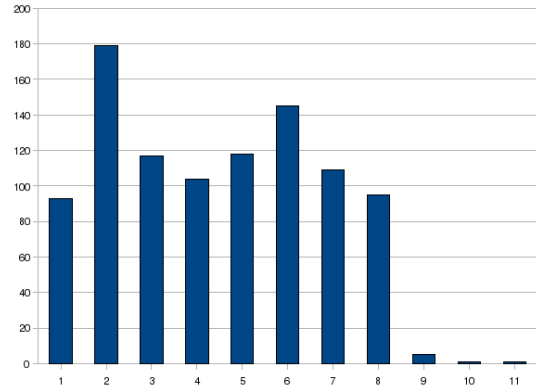


Figure 11. Histogram of the time to synchronize (in # of TDMA rounds)

4.3 Effectiveness of the self-reconfiguration

To assess the benefits of the proposed scheme we compared it with the simple adaptive-TDMA as proposed in [7]. Note that the benefits of the adaptive-TDMA approach with respect to unsynchronized communication were already shown in that work. Moreover, we nearly saturate the network with uncontrolled load, generated by two external stations exchanging ping packets between

them with a rate of 1 every 10ms and with 1450 data bytes. This caused a total load, measured with an external device, slightly above 60%.

In the adaptive TDMA case we consider now 2 nodes using two consecutive slots in a TDMA round with 10 slots. Figure 12 shows a log of the difference between each transmission from one agent to the preceding transmission from the other (inter-transmission delay). Since the slots are consecutive, we expect this difference to be about 10ms in one agent and 90ms in the other. Both intervals can be longer as enforced by the round period adaptation mechanism. We see that the frequent large delays caused by the uncontrolled load often bring the team out of synchronization. During the run that lasted 2min, the team stayed synchronized during only 50% of the time, and 29 packets were lost (1,2% of total packets sent).

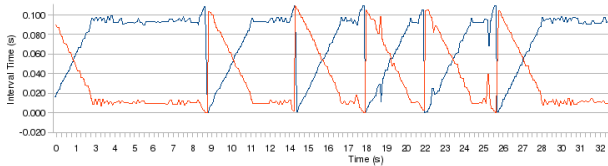


Figure 12. Log of the time differences between transmissions with 2 nodes under adaptive-TDMA

The same experiment was repeated using the self-reconfiguration procedure where the protocol divided the whole round in just two slots assigned to the two agents. These slots were 5 times larger than those in the adaptive-TDMA case thus creating a much larger leeway to accommodate the delays caused by the uncontrolled traffic. Figure 13 shows the corresponding log for this situation in which we can see that both agents transmit with an average interval of 50ms and the team seldom lose synchronization. Actually, the figure shows one of the rare cases of such loss of synchronization and subsequent resync (large peak at second 46). The team stayed synchronized 97% of the time and lost 7 packets, corresponding to 0.3% of all sent packets.

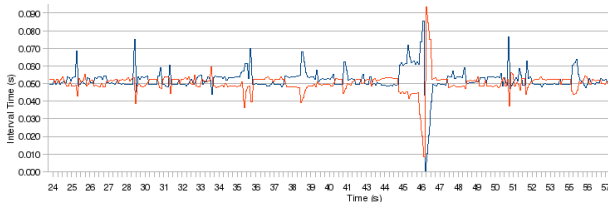


Figure 13. Log of the time differences between transmissions with 2 nodes with the self-reconfiguration process

5 Other features/issues

5.1 Asymmetric bandwidth distribution

One interesting feature that is already supported in the current protocol implementation is the possibility to assign more than one slot to each agent. This can easily be achieved by allowing an agent to use multiple static IDs. The protocol, transparently, will consider each of these IDs as a different virtual agent and assign the respective real agent the corresponding number of slots. This mechanism has been tested and shown to work as expected.

5.2 Dynamic TDMA round period

The TDMA round period establishes the responsiveness and temporal resolution of the protocol. Thus, during periods of intense team dynamism it could be interesting to use a shorter value for T_{tup} while a larger value could be adequate for periods of quietness, profiting from less transmissions. The on-line adaptation of T_{tup} also requires a consensus procedure, similarly to the membership vector referred in Section 3. However, if there is a need for a sudden change from a large to a short T_{tup} , the nodes with such need can start transmitting asynchronously until the consensus is achieved. This mechanism is currently being implemented.

5.3 Ad-hoc mode

The use of an AP is an helpful and practical option to simplify consensus among a set of distributed agents and thus simplify their management and synchronization. It is also easy to deploy since the AP can be carried by one of the agents in the team. However, it might be beneficial to avoid the AP, mainly in terms of fault-tolerance and channel and energy efficiency. Without the AP our protocol becomes susceptible to cliques as soon as nodes start to move apart creating connections with more than 1 hop. We are currently working to prevent this by using distributed consensus techniques suited for ad-hoc networks.

6 Related work

In wireless communications (data or voice) it is common to use slot assignment. In [10, 4, 9], the authors propose using dynamic TDMA frameworks where slots in a fixed round structure are assigned dynamically to nodes. Moreover, these approaches are normally centralized with a master collecting slot requests and assigning them dynamically.

In [2] the authors also do synchronization based on the reception time instants of the transmission of other nodes in a team. However, the protocol was not developed to cope with uncontrolled load in the channel. In fact all nodes must comply with RI-EDF for the protocol to work.

In IEEE 802.11e and ZigBee in synchronous mode, hardware-based mechanisms are used to enforce synchronization and support different levels of Quality of Service.

However, specific network adapters need to be used, increasing their cost and reducing their availability.

Finally, IEEE 802.11-DCF and ZigBee unsynchronized mode are based on the well-known CSMA-CA technique which is well adapted to asynchronous communication, leading to an efficient use of the channel bandwidth. However, under high loads, their performance degrades because of collisions and their efficiency is reduced.

None of these protocols is simultaneously fully distributed, synchronous, adaptive / reconfigurable and built on top of widely available COTS network interfaces, which our protocol is. Thus we believe it constitutes a novel and efficient solution to the problem of supporting the coordination of teams of mobile robots.

7 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 wireless communication layer of the CAMBADA middle-size robotic soccer team being developed at the University of Aveiro.

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. Several experiments are shown that illustrate the effectiveness of the protocol. It is worth noting that the protocol has actually been under use for over one year in RoboCup competitions.

Future work includes the further dynamic reconfiguration of the TDMA round interval, i.e., the team update period, according to the instantaneous timeliness requirements imposed by dynamic environments. The changes needed to operate under ad-hoc mode are also being addressed, which will allow operating without an AP and increasing the efficiency in the use of the channel bandwidth.

8 Acknowledgements

The authors would like to acknowledge the helpful discussions with Prof. Daniel Mossé in an earlier version of this paper. This work was partially supported by the European Commission through grant ArtistDesign ICT-NoE-214373 and Portuguese Government through grant FCT - SFRH/BD/29839/2006.

References

- [1] L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, and L. S. Lopes. Coordinating distributed autonomous agents with a real-time database: The cambada project. In C. Aykanat, T. Dayar, and I. Korpeoglu, editors, *ISCIS*, volume 3280 of *Lecture Notes in Computer Science*, pages 876–886. Springer, 2004.
- [2] T. L. Crenshaw, A. Tirumala, S. Hoke, and M. Caccamo. A robust implicit access protocol for real-time wireless collaboration. In *Proceedings of the ECRTS'05 - Euromicro Conference on Real-Time Systems*, pages 177–186, Palma de Mallorca, Spain, July 2005.
- [3] K. Gopalan. Real-time support in general purpose operating systems. Technical report, 2001.
- [4] A. Kanzaki, T. Uemukai, T. Hara, and S. Nishio. Dynamic tdma slot assignment in ad hoc networks. In *Proceedings of the AIDA'96 - Advanced Information and Networking Applications*, pages 330–335, March 2003.
- [5] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of the IJCAI-95 Workshop on Entertainment and AI/Alife*, Montreal, August 1995.
- [6] P. Pedreiras and L. Almeida. 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.
- [7] 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 *Proceedings of the WACERTS04 Workshop on Architectures for Cooperative Embedded Real-Time Systems (in conjunction with RTSS2004 - 25th International Symposium on Robotics and Automation)*, Lisbon, Portugal, December 2004.
- [8] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.
- [9] N. D. Wilson, R. Ganesh, K. Joseph, and D. Raychaudhuri. Packet cdma versus dynamic tdma for multiple access in an integrated voice/data pcn. *IEEE Journal on Selected Areas in Communications*, 11(6):870–884, August 1993.
- [10] C. D. Young. Usap: A unifying dynamic distributed multi-channel tdma slot assignment protocol. In *Proceedings of the MILCOM'96 - Military Communications Conference*, volume 1, pages 235–239, October 1996.