PaSo-Team 2000

Carlo Ferrari, Francesco Garelli, Enrico Pagello

Dept. of Electronics and Informatics, The University of Padua, Italy

1 Introduction

Following the experience done in previous competitions, it has been developed the 2000 version of PaSo-Team (The University of PAdua Simulated Robot SOccer Team), a reviewed release of Paso-Team99. During the RoboCup '99 competition in Stockholm some teams suffered syncronization problems with the soccer server: these problems greatly influenced their performances and prevented them from playing successfully. While developing PaSo-Team 2000 the main efforts were dedicated to better understanding timing and syncronization techniques for real-time multi-agent systems. Following the interesting experience done by Kostiadis in developing the Essex Wizzard team [1] we redesigned the syncronization procedures using the multi-threading paradigm. Solving syncronization in a multi-threading environment gives important theoretical hints to approch the coordination for those multi-agent systems made by thousand of very simple concurrent interacting modules. During the Stockholm competition PaSo-Team99 suffered another major problem regarding the actions a player must take when the game is stopped (i.e. when the ball is outside or when a team is offside). For example if a player has to throw-in the ball, he must go outside the field, turn toward the field and eventually kick the ball, performing different actions even if the state of the game doesn't change. As in a reactive architecture the current behaviour can change only when the game state changes, PaSo-Team99 introduced virtual states to ensure a change of behaviour. Instead in PaSo-Team 2000 we simplify the design of these actions introducing multi-step behaviours.

2 Team Development

Team Leader: Enrico Pagello *# (Associate Professor of Computer Science)
Team Members:

- Carlo Ferrari * (assistant professor of Computer Science)
- Francesco Garelli * (graduate student)
- Stefano Carpin * (graduate student)
- Andrea Sivieri * (undergraduate student)

Web page http://www.dei.unipd.it/~robocup
* Dept. of Electronics and Informatics, The University of Padua, via Gradenigo 6a, 35131 Padua, Italy
# **Ladseb-Cnr,** C.so Stati Uniti 4, 35100 Padua, Italy.
E mail: {epv, carlo, garelli, shamano, tigre, gremlin, avatar, keter}@dei.unipd.it

P. Stone, T. Balch, and G. Kraetzschmar (Eds.): RoboCup 2000, LNAI 2019, pp. 429-432, 2001.

3 Software architectures

PaSo-Team 2000 was developed using the Linux Operating System with the GNU C++ compiler. The used library was the standard GNU libc with Posix PThread Extensions; we didn't use other libraries like libsclient.

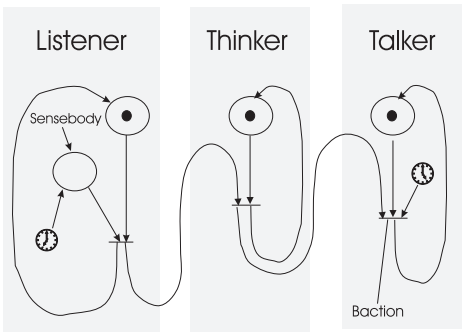

Fig. 1. Threads coordination in PaSo-Team 2000

PaSo-Team 2000 is based on three different concurrent threads:

- Listener: it gets the information from the soccer server and it manages the client timing and syncronization.
- Thinker: it updates the player's internal memory and it executes the suitable behaviour for the current simulation step.
- Talker: it manages the queue of the messages to be sent to the server .

The coordination between the threads follows the rules represented in Figure 1. The listener can be activacted either by the sensebody signal received from the server or by a timeout signal. The timeout signal assures a correct activation even if the sensebody signal was lost for an excessive load of the server (or the net); of course the timeout is dinamically computed considering the average delay between the last received sensebodies. Althought it is not represented in the figure, the listener module either can delay its activation to wait for a seeinfo message or it can be activated by a message which is not a sensebody. When the listener is activated it empties the socket queue and it re-arranges the possible messages received from the server. Finally it sends a activation signal to the thinker. When the thinker received the activation signal from the listener it updates the player's memory, i.e. the internal representation of the environment, and it sends messages for the server to the talker thread. The talker thread forwards the messages to the soccer server assuring a minimum time gap between them.

4 The world model

Like PaSo-Team99, PaSo-Team 2000 uses Synthetic Visual Maps (SVMs) for motion control [5] . SVM are a concise representation of the free space around the player. The SVM maps each movement direction of the player with a boolean value that says whether that direction is free or prohibited. The SVM can be seen as a polar representation of the free space in a proper disk centered in the player. This representation can be easily updated at each sensing cycle, in order to consider new game elements that become important, either because they are moving towards the player or because the player itself is moving towards them.

5 Communication

During the 1999 competition it was clear that our team had reduced sensors information. The soccer server sends to the robocup clients only a partial knowledge about the environment around: in particular the server supplies information about the objects inside a 90° view only. The robocup simulation league is an environment where clients can't arbitrate their behaviours in a deterministic fashion: they can't choice the optimal behaviour in every situation because their representation of the current state of the server is not complete. Anyway the players can improve their knowlegde of the environment and therefore their arbitration function sharing their partial information. Infact the soccer server provides the command *say* by which a singular player can broadcast a message to their mates every simulation step. In Paso-Team 2000 we introduced a communication layer between the agents just to insure a better world model construction; we did not use the *say* command to coordinate the clients although. We confirmed the idea of not realizing any coordination via explicit communication at behaviour level: the communication layer involves only the representation of the environment and it doesn't affect the selection of current bevaviours. The communication layer uses a token-based protocol because the soccer server doesn't allow many players to send messages concurrently. At every simulation step only one player, the owner of the token, can broadcast messages to their mates; we called this player *observer*. The observer should be the player who owns the best information about the environment for the current game situation. Hence the observer should have good information about the objects around the ball (where the game is) but he should not be involved in the current action because he has not to be the ball owner. Moreover during the match the observer changes because the relative position of the players change; at every step the choice of the next observer should be done by the player with the best knowledge of the server state, i.e. the current observer. According to these requirements we developed a protocol to distribute the knowledge between the clients: at every step the observer send to the mates his representation of the environment and the next observer id-number. When every client receives the message he integrates his representation with the observer's one and if required he becomes the new observer. Unluckly this protocol is not robust enough in the Robocup environment where

the delivery of the packets is not assured. Infact because of network problems the observer's token can be lost. To overcoming this problem we introduced in the communication layer a monitor module. Every player checks periodically if an active observer is present in the team; if the check fails he starts a procedure to elect a new observer.

6 Skills

In PaSo-Team 2000 we introduced multi-step behaviours. Multi-step behaviours are complex actions which can be completed in many simulation steps: they proved to be very effective in stopped game situations (throw-in, catch...). A multi-step behaviour can be aborted when a critical event (like either the lost of the ball or an offside) happens by throwing a C++ exceptions.

7 Conclusions

In the PaSo-Team'99 project we experimentally investigate how much the reaction schema for intelligent agents team must be integrated with some kind of high level reasoning. In PaSo-Team 2000 the major emphasis was on syncronization issues that were solved introducing multi-threading. The overall software architecture was redesigned and some kind of explicit communication was introduced to enhance the accuracy of the world representation procedures.

Acknowledgements

This research work could not be done without the enthusiastic participation of the students of Electronics and Computer Engineering Undergraduate Division of Padua University Engineering School. Financial support has been provided by both CNR, under the Special Research Project on "Real-Time Computing for Real-World" and MURST, under the 60% and 40% Grants.

References

[1] Kostiadis K. and Hu H., "A Multi-threaded Approach to Simulated Soccer Agents for the RoboCup Competition" In Veloso M., Pagello E. and Kitano H., editors, RoboCup-99: Robot Soccer World Cup III. Springer Verlag. Berlin, 2000.

[2] H-D. Burkhard, M. Hannebauer, J.Wendler, "Belief-Desire-Intention Deliberation in Artificial Soccer", AI Magazine, Fall 1998.

[3] E. Pagello, A. D'Angelo, F. Montesello, F. Garelli, C. Ferrari, "Cooperative behaviors in multi-robot systems through implicit communication". Robotics and Autonomous Systems, 29 (1999), 65-77

[4] P.Stone, "Layered Learning in Multi-Agent Systems", Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, December 1998.

[5] C. Ferrari, F. Garelli, E. Pagello, "PaSo-Team99" In Veloso M., Pagello E. and Kitano H., editors, RoboCup-99: Robot Soccer World Cup III. Springer Verlag. Berlin, 2000.