Pardis

Shahriar Pourazin

Computer Engineering Department,
AmirKabir University of Technology, Tehran, IRAN
Tel: +(9821) 6419411, Fax: +(9821) 6413969
pourazin@ce.aku.ac.ir

1 Introduction

Pardis, was one of the entries in RoboCup-99, simulation league. It had a optimistic timing in communication with the server. And lost most of the cycles in the real league, because of relying on the enough network bandwidth. So unfortunately it had chance to be only in the first round robin. It used an experimental model, consisting of finite set of categories for each player. Each softbot in Pardis team, was a player acting as designed in a specific category. The coach had the ability to map each player in the opponent team with one of the same categories. It dynamically changed the characteristics (category) of the facing teammate to be effective against the analyzed opponent player. Although in the real league, there was no chance to see the use of the coach and it was never activated. The players read their behavioral configuration once at the start of the game and kept playing that way.

2 Team Development

Pardis was the result of 9 man-month development effort, mostly done by three people, the team leader and two undergraduate students.

Team Leader: Shahriar Pourazin
Team Members:

Ali Ajdari Rad
- team member, coding and representation of categories
- IRAN
- undergraduate student
- attended the competition

Houman Atashbar
- team member, coded the low level parts, communication etc.
- IRAN
- undergraduate student
- attended the competition

Web page http://www.pnu.ac.ir/~ pourazin/rc99

M. Veloso, E. Pagello, and H. Kitano (Eds.): RoboCup-99, LNAI 1856, pp. 614–617, 2000.
Springer-Verlag Berlin Heidelberg 2000

3 World Model

The spatial model of the field consisted of separated squares making some regions. Players had patterns which described their behavior in each region. According to (say) their distance to the opponent's goal, they selected their action among the choices to pass, dribble, kick, etc. The more the player approached to the opponent's goal, the more eager it would become to kick to goal. The desire to kick, dribble, pass and other actions, had been stored as fuzzy values in an array called the *desire array*. Having no ball, near our own goalie, the player had to notice the ball and kick it away, and at the middle regions, it keeps trying not to let opponents receive any pass and if gets the ball passes it to teammates.

The real calculations on the array of fuzzy values for actions, depended on the exact region the player is in, the result of the game so far, the Boolean flag indicating that the player has the ball, and the position of other players. The player will do the action with the highest value.

All the codes were written in C++ from the scratch without any use of external prewritten libraries such as libsclient.

4 Communication

We had designed a method for message passing between players, by doing some special actions in front of the teammate, e.g., if the player does four 45 degree turns each one in the opposite direction of the previous, means that, received ball will be sent back soon. This mechanism was designed to reduce the `SAY` messages, but we had no chance to see its effect in the qualification of teams when no `SAY` is possible. They were supposed to receive from the coach some information (making the inter-player messages unnecessary), and also commands (to change their desire array).

5 Skills

Pardis has the goalie, as the player which receives the number one. It is different from the other players, such that, has no desire to take the ball toward opponent's goal, etc. It stays near the goal, uses fast moves when has the ball and the opponents are near the goal. So the goalie has no special difference in structure. All it has, is a different desire array.

6 Special Team Features

The players were designed to be as single threaded processes (i.e. no parallel processing in the players). The coach had to have a huge parallelism. It should have a plenty of models instead of each opponent, looking which model plays the same as the real opponent. This lets the coach determine the characteristics of the opponent player.

7 Conclusion

Our approach to opponent modeling had a severe assumption that the roles of the players in the opponent team are static or at least not rapidly changing. The unstructured program was not suitable for maintenance. That's why we had no chance to come up with the slow network in Stockholm. So the right thing to do is, throwing away the code and writing a new structured one.

After analyzing the timing of the whole system, we learned that working with the server could be very complicated. Even in the case that the server remains unchanged, the effect(s) of the networking problems could not be easily predicted. So how could we start our work? Is it necessary to think of all anomalies in all of our processes?

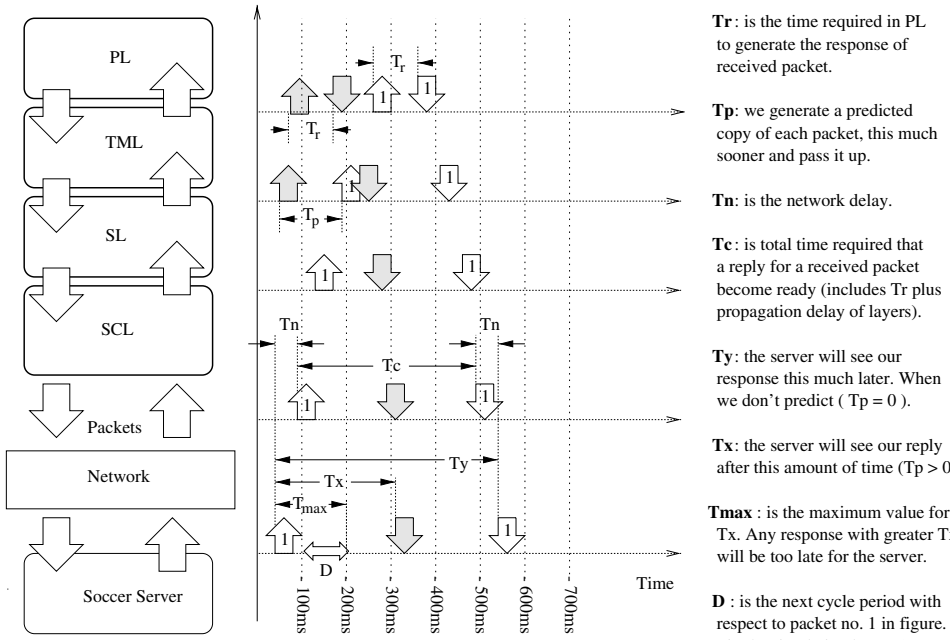

Fig. 1. The timing of an exchanged packet.

The answer could be found in the isolation of the anomalies. To do so we have to design a layered architecture for our next generation player. The structure has four layers: (1) Sense/Control Layer, (2) Synchronization Layer, (3) Task Management Layer and (4) Planning layer from bottom to top. The lowest layer (SCL) is to be modified to let us use upper layers in other leagues (F2000,

F180 or Legged). SL is to isolate the timing anomalies from the upper layers. TML is to translate abstract plans (of PL) into strings of smaller activities. This approach lets us, try to work on the (say) intelligent part of our player without being mixed up with unreliable data. Anyone who wants the corrupted data in the PL, could modify SL later. Figure 1 shows the four layers, receiving packets from server and sending back the responses. It is also shown that the response should be in the D interval. The grayed arrows stand for predicted packets. T_p should be large enough to shift the gray arrows to the left, making the player on time. Now we are working on the knowledge representation in the PL and the dynamic estimation of T_p.