RoboCup Soccer Server and CMUnited: Implemented Infrastructure for MAS Research

Itsuki Noda\textsuperscript{1} and Peter Stone\textsuperscript{2}

\textsuperscript{1} Electrotechnical Laboratory, JAPAN
\textsuperscript{2} AT&T Labs – Research, NJ, USA

Abstract. The RoboCup Soccer Server and associated client code is a growing body of software infrastructure that enables a wide variety of multiagent systems research. This paper describes the current Soccer Server and the champion CMUnited soccer-playing agents, both of which are publically available and used by a growing research community. It also describes the ongoing development of Fuss, a new, flexible simulation environment for multiagent research in a variety of multiagent domains.

1 Introduction

The Robot Soccer World Cup, or RoboCup, is an international research initiative that uses the game of soccer as a domain for artificial intelligence and robotics research. Annual international RoboCup events involve technical workshops as well as software and robotic competitions.

The RoboCup Soccer Server [5, 4] and associated client code is a growing body of software infrastructure that enables a wide variety of multiagent systems research. It is used as the substrate for the RoboCup software competitions. Originally released in 1995, Soccer Server has an international user community of over 1000 people.

Soccer Server is a multiagent environment that supports 22 independent agents interacting in a complex, real-time environment. The server embodies many real-world complexities, such as noisy, limited sensing; noisy action and object movement; limited agent stamina; and limited inter-agent communication bandwidth. AI researchers have been using the Soccer Server to pursue research in a wide variety of areas, including real-time multiagent planning, real-time communication methods, collaborative sensing, and multiagent learning [2].

As such, this infrastructure is appropriate for a wide variety of multiagent systems research, with algorithms developed for the Soccer Server likely to apply to other domains. Indeed, such research has been applied to other domains including disaster rescue [3], helicopter fighting [11], and network routing [8], among others.

In addition to the server itself being publicly available in an open-source paradigm, users have contributed several clients that can be used as starting points for newcomers to the domain. One example is the CMUnited-98 simulated soccer team, champion of the RoboCup-98 robotic soccer competition. After winning the competition, much of the CMUnited-98 source code became publicly
available, and several groups used it as a resource to help them create new clients for research and as entries in the RoboCup-99 competition.

Based on the success of Soccer Server and its associated client code, we are now in the process of creating a new flexible utility for simulation systems (FUSS) that will be designed to support simulations of multiple domains. For example, we plan to use the same underlying simulation for an improved simulator of the game of soccer as well as a disaster rescue simulator for use in the RoboCup Rescue initiative [3]. FUSS will also be available as infrastructure for the MAS research community.

The remainder of the paper is organized as follows. Section 2 gives an overview of the RoboCup Soccer Server. Section 3 presents the CMUnited simulated soccer clients for use with Soccer Server. Section 4 motivates and presents the current state of the development of FUSS and Section 5 concludes.

2 The RoboCup Soccer Server

Soccer Server enables a soccer match to be played between two teams of player-programs (possibly implemented in different programming systems). A match using Soccer Server is controlled using a form of client-server communication. Soccer Server provides a virtual soccer field and simulates the movements of players and a ball. A client program can provide the ‘brain’ of a player by connecting to the Soccer Server.

A client controls only one player. It receives visual (‘see’) and verbal (‘hear’) sensor information from the server and sends control commands (‘turn’, ‘dash’, ‘kick’ and ‘say’) to the server. Sensor information tells only partial situation of the field from the player’s viewpoint, so that the player program must make decisions using these partial and incomplete information. Limited verbal communication is also available, by which the player can communicate with each other to decide team strategy.

Soccer Server has been used by researchers to examine multi-agent systems (MAS). The biggest reason why it is used widely is that it simulates soccer, which is known widely in the world. As same as the case of chess, well-known-ness is an important factor for example applications of research.

The second reason is that it uses the middle-level abstraction for representing the client commands and the sensor information. If we used a low-level, physical description, it was felt that such a representation would concentrate users’ attention too much on the actual control of a players’ actions. On the other hand, if we used tactical commands such as pass-ball-to and block-shoot-course, it would produce a game in which the real-world nature of soccer becomes obscured. Thus, our representation, turn, dash, and kick, is a compromise. To make good use of the available commands, clients will need to tackle both the problems of real-world and MAS.