The Statistics Proxy Server

Ian Frank[1], Kumiko Tanaka-Ishii[2], Katsuto Arai[3], and Hitoshi Matsubara[4]

[1] Complex Games Lab, ETL, Tsukuba, Japan 305, ianf@etl.go.jp
[2] Tokyo University, Japan, kumiko@ipl.t.u-tokyo.ac.jp
[3] Chiba University, Japan, karai@cogsci.l.chiba-u.ac.jp
[4] Future University, Hakodate, Japan, matsubar@fun.ac.jp

Abstract. We present a real-time statistical analysis tool for soccer. This system is designed to promote the advancement of RoboCup by facilitating fundamental research on issues such as learning and team evaluation and assessment. Analysis of a game is carried out by a central server, to which clients can connect to request data. We describe the operation of the system and give examples of its potential applications.

1 Introduction

The fundamental goal of RoboCup is scientific research. In this paper, we present a tool that promotes this goal by providing real-time, in-depth statistics on soccer games. We expect this tool to have many uses, including the evaluation of team modules and the design of automated coaches. We have released all our code as open source to provide maximum accessibility and functionality to all users.

The original motivation for this work was our research on the automatic commentary system MIKE [1, 2]. For commentaries, high-level analysis of a game is clearly important for analysis such as game summaries and for identifying the noteworthy aspects of the game. At the same time, however, we realised that high-level information on a game would also be indispensable to other researchers. We therefore split out all of MIKE's statistical analysis code and used it as the core for a separate analysis program. We chose the server-client model for this program because it provided the maximum benefit for the smallest possible system load. Since it is designed to run in tandem with the RoboCup's Soccer Server, we call our system the Statistics Proxy Server (or sometimes also 'pre-MIKE', in its role as a pre-processor for MIKE).

The proxy server is responsible for collecting and making available the statistics on a game in real-time. Any number of client modules can then attach to the server and request some subset of the data at configurable intervals. We describe both the server and the client protocol below, giving examples of their use (§2). We anticipate that real-time access to such statistics will find many uses within RoboCup. We mentioned above our own work on soccer commentary, and we have also used the statistics to demonstrate the progress of the RoboCup simulation league to date [3, 4]. In §3, we describe how we have further used the proxy server to generate a database of statistics on previous RoboCup games, and how the current proxy server utilises this database to increase the options available to clients. Finally, in §4 we discuss the possibilities for further developments, before giving our conclusions in §5.

2 System Details

The relationship between the statistics proxy server and the core code of MIKE is shown in Figure 1.

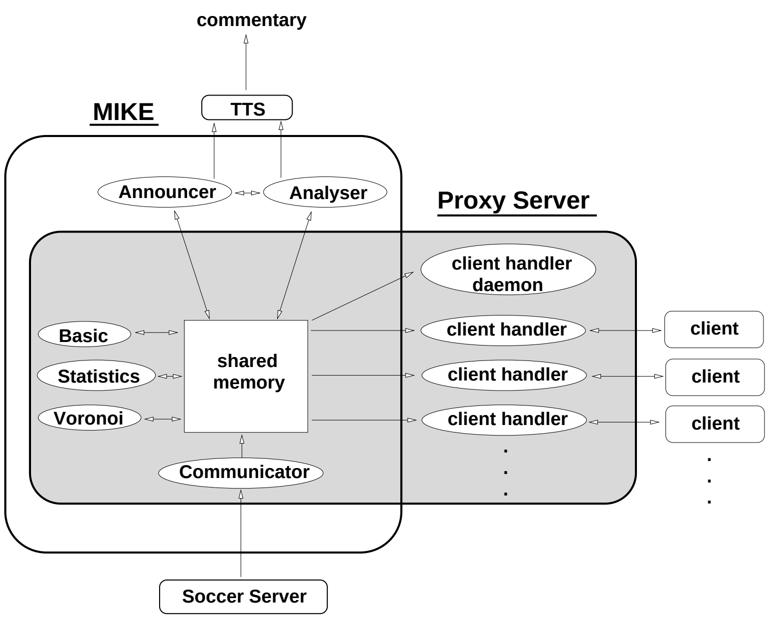

Fig. 1. Producing a proxy server from the core of MIKE

The *client handler daemon* in this figure creates separate *client handler* threads to communicate with each client that connects with the proxy server. Thus, the server is extremely flexible: it can provide different data on different time schedules to many different clients. The most convenient way for a client to make its connection is in *configuration script mode* (there is also a *command-line mode*, which we describe in full detail in the manual bundled with the code). The configuration script specified by the client on the command-line must contain a series of lines each starting with one `command` and followed by a number of `flags`. Each `command` in the configuration script corresponds to a particular statistic. In practice, these commands are divided into three groups.

– **Team Statistics**. This first, and largest, group of commands relates to statistics about entire teams. Examples of team-based commands are

TScore *team*, and
TXAvr *team*,
which request statistics on the score and average X position of the players of the specified *team*, respectively. The variable *team* must be either left or right. However, an easy method for simultaneously requesting statistics on *both* teams is also implemented: simply set *team* to either '*' or '_'.

- **Player Statistics**. The second group of commands relates to statistics on individual players. Examples of player-based commands are
PlShootNum *player*, and
PlDistance *player*,
which request statistics on the number of shots made by *player*, and the distance that *player* has covered, respectively. Here, the variable *player* identifies players via the numbers displayed by the Soccer Server monitor. Again replacing the variable with '*' or '_' defaults to requesting information on all possible players. To make it easier to simultaneously obtain the same statistics for large numbers of players (*e.g.*, all those on one team), the specification of ranges for the *player* variable is supported. For example, PlDistance 1-11 will give the distance covered by all the players on one team.
- **Other Statistics**. The final group of commands cover extra statistics that concern the interactions between two individual players. For example,
Bigram *player1 player2*, and
CorXX *player1 player2*.
Here, *player1* and *player2* are once more the player numbers as displayed by the Soccer Server monitor, and the '*'/'_' or range notation are again available. Bigrams and correlations, together with the complete list of all the other statistics, are described in the proxy server manual.

In total, there are 32 team statistics, 14 player statistics and 5 other statistics. There is also one extra command, all, that requests the entire set of current statistics to be sent. In the configuration script, each command must be accompanied by at least one flag. These flags allow clients to control the way that statistics are sent by the server, as follows:

1. T *num* — a mandatory flag for every command in the configuration file. It specifies the time interval (*num*, in Soccer Server game steps) at which the statistic should be sent to the client.
2. L *num* — specifies a lower bound. The statistic is only sent to the client (at the time determined by the T flag) if it is greater than or equal to the bound *num*.
3. U *num* — specifies an upper bound. The statistic is only sent to the client (at the time determined by the T flag) if it is less than or equal to the bound *num*.
4. C *num* — specifies a minimum change. The statistic is only sent to the client (at the time determined by the T flag) if it has changed by an amount greater than *num* since it was last reported (or if it has never been reported).

These flags allow for the construction of sophisticated reporting regimes. For instance, the following command will produce statistics every 10 time steps on all the left team players except the goalkeeper (player 1) with 5 or more shots on goal: `PLShootNum 2-11 T10 L5`.

In Figure 2 we give a simple example of a complete configuration script. Commented lines start with a hash (#) and blank lines are allowed. When a client successfully connects to the server using such a script, it receives the requested data for the duration of the connection whilst play is still in progress. The format for sending information to the clients is as follows:

$$\langle\texttt{command}\rangle\ \langle\texttt{variables}\rangle\ \langle\texttt{value}\rangle\ \langle\texttt{time}\rangle$$

where `command` is simply an echo of the command that requested the data (just the label identifying the statistic itself), `variables` identifies the subject (a team for team statistics, a player for player statistics, or a pair of players for other statistics), `value` is the result computed by the proxy server, and `time` is the current game time.

```
# Comment: Simple proxy configuration script
# Requests average X and Y location of the left team's players
# every 5 game steps. Also requests, every 10 game steps, the number
# of goals scored by any goalscorer (player with one or more goals).

TXAvr left T5
TYAvr left T5
PlScore * T10 L1
```

Fig. 2. A simple script for a client connecting in configuration script mode

3 The Data Centre

We have already used pre-release versions of the proxy server to analyse previous RoboCup tournaments [3, 4]. The statistics proxy server allowed us to demonstrate how teamwork has improved significantly along the long side of the field but not along the shorter, how teams carry out more marking, and how the link between ball-handling skills and performance has strengthened.

In addition to carrying out such analysis, however, we can also now make data on past games available in real-time through the proxy server. To do this we analysed the logs of all the 243 completed games from the first three years of RoboCup simulator contests (a database of over 500Mbytes). For each game, we recorded the values of each of the proxy server statistics. We also then created two extra sets of data by collecting separately the data for the winners and losers of each game. (Actually, since team designers are allowed to manually change their team settings and code at half-time, we built these extra databases by looking at the winners and losers of each *half-game*, excluding the 97 half-games that were drawn). The statistics proxy server makes the results of this analysis available via the following extra flags:

- `Y` *yr* — requests data on a statistic directly from the data centre for year *yr* (a four digit number) rather than on the current game. The average of the values in the data centre is returned. Overrides all other flags, which are ignored if DC is specified.
- `S` *yr-percent* — specifies a significance level for reporting a statistic. The value of *percent* must be greater than 0 and less than 100. If *percent* is greater than or equal to 50, the statistic is only reported (at the time determined by the `T` flag) if its value in the current game ranks in the top *percent*% of stored values for year *yr* in the data centre. For values less than 50, the statistic is only reported if its value in the current game ranks in the bottom *percent*% of stored values.
- `A` *yr-percent* — specifies a significance level for reporting a statistic. Same as the `S` flag except that the value of the statistic must be significant amongst just *winning* teams.
- `B` *yr-percent* — specifies a significance level for reporting a statistic. Same as the `S` flag except that the value of the statistic must be significant amongst just *losing* teams.

These flags greatly increase the power of the proxy server. For instance, a script containing just the commands '`all S1999-5 T5`' and '`all S1999-95 T5`' will allow a client to receive (every 5 game steps) just the statistics on the current game that are in the top or bottom 5% when compared to the teams in the previous year's RoboCup. Future versions of the proxy server will also include a tool for creating extra data sets from arbitrary collections of log files. This will enable developers to easily compare the performance of new updates of their programs against established benchmark versions.

4 Developments

The possible uses of the proxy server are limited only by the imagination of RoboCup developers. We give a partial list below. Note that machine vision systems for creating Soccer Server style logs from real robot games will also soon allow the statistics proxy server to be used with the real robot leagues.

- Create a program for visualising statistics during a RoboCup game (an HTML version of this is currently underway).
- Add a module that gives information on how various statistics change *during* a game.
- Use the proxy server as the basis for a strong on-line coach client.
- Use the statistics as a testing tool to identify more precisely the strengths and weaknesses of individual teams. At the current stage of RoboCup, winning is taken to identify the teams that employ good techniques. However, the

reverse is not true: teams that lose do *not* necessarily employ bad techniques. The ability to identify strong aspects of teams whose overall performance does not compare to the champions will help prevent good research ideas being overlooked.
– Develop a testing tool that can play two teams against each other until to test a given, concrete hypothesis (*e.g.*, team A is stronger than team B, team A passes more accurately than team B) at a pre-determined confidence level.

Eventually, we hope the proxy server will form the core of an advanced RoboCup expert analysis module incorporating techniques developed by many RoboCup researchers. A good starting point for this would be the work done by the developers of ISAAC [5], who have used C4.5 to build decision trees that describe how teams behave in a game. Since it is hard to build such trees incrementally during a game, ISAAC is typically used as a post-game learning module.

5 Conclusions

We have described the statistics proxy server we have developed for RoboCup. We gave details of the implementation and examples of the use of the system. We discussed how the server has already been used in projects such as MIKE and to assess the progress of RoboCup to date. We also showed that it is possible to incorporate data from past RoboCups and listed some of the promising potential applications of the proxy server system.

It is our hope that the statistics proxy server will be a significant and useful tool in the development of RoboCup. To maximise the chances of this actually happening, we have released our source code into the public domain.

References

1. K. Tanaka-Ishii, I. Noda, I. Frank, H. Nakashima, K. Hasida, and H. Matsubara. MIKE: An automatic commentary system for soccer. In *Proceedings of ICMAS-98*, pages 285–292, 1998. Also available from Electrotechnical Laboratory as Tech Report ETL-97-29.
2. H. Matsubara, I. Frank, K. Tanaka-Ishii, I. Noda, H. Nakashima, and K. Hasida. Automatic soccer commentary and RoboCup. In *Proceedings of the Second International Workshop on RoboCup*, pages 7–22.
3. K. Tanaka-Ishii, I. Frank, I. Noda, and H. Matsubara. A statistical perspective on the RoboCup simulator league: Progress and prospects. In *Proceedings of the Third International Workshop on RoboCup*, 1999.
4. K. Tanaka-Ishii, I. Frank, and K. Arai. Trying to understand RoboCup. *AI Magazine*, 2000. To appear.
5. M. Tambe, G. Kaminka, S. Marsella, I. Muslea, and T. Raines. Two fielded teams and two experts: A RoboCup challenge response from the trenches. In *Proceedings of IJCAI-99*, volume 1, pages 276–281, Stockholm, Sweden, 1999.

This article was processed using the LaTeX macro package with LLNCS style