Multi Agent-based Approach for Asynchronous Web-based Training System

Takao KAWAMURA, Shinichi MOTOMURA, Ryosuke NAKATANI, and Kazunori SUGAHARA
Tottori University
4–101, Koyama-Minami, Tottori 680–8552, JAPAN
+81 857 31 5217
{kawamura,motomura,rnakatan,sugahara}@ike.tottori-u.ac.jp

Abstract—*In this paper, we present a novel framework for asynchronous Web-based training. The proposed system has two distinguishing features. Firstly, it is based on P2P architecture for scalability and robustness. Secondly, all contents in the system are not only data but also agents so that they can mark user's answers, can tell the correct answers, and can show some extra information without human instruction. We also present a prototype implementation of the proposed system on Maglog. Maglog is a Prolog-based framework for building mobile multi-agent systems we have developed. The user interface program of the proposed system is built on Squeak. Performance simulations demonstrate the effectiveness of the proposed system.*

1. INTRODUCTION

The term e-Learning covers a wide set of applications and processes, such as Web-based training (hereafter we abbreviate as WBT), computer-based training, virtual classrooms, and digital collaboration. We are concerned with asynchronous WBT that allows the learner to complete the WBT on his own time and schedule, without live interaction with the instructor.

Although a large number of studies have been made on asynchronous WBT[1, 2], all of them are based on the client/server model. The client/server systems generally lack scalability and robustness. In the recent years, P2P research has grown exponentially. Although the current P2P systems are famous for its file sharing ability, and the consequent legal problems, P2P systems are gradually proving to be a very promising area of research. Because they have potential for offering a decentralized, self-sustained, scalable, fault tolerant and symmetric network of computers providing an effective balancing of storage and bandwidth resources.

In this paper, we present a novel framework for asynchronous WBT. The proposed system has two distinguishing features. Firstly, it is based on P2P architecture and every user's computer plays the role of a client and a server. Namely, while a user uses the proposed e-Learning system, his/her computer (hereafter we refer to such a computer as a node) is a part of the system. It receives some number of contents from another node when it joins the system and has responsibility to send appropriate contents to requesting nodes. Secondly, each content in the system is not only data but also an agent so that it can mark user's answers, tell the correct answers, and show some extra information without human instruction.

This paper is organized in 8 sections. We describe the supposed situation for the proposed system in Section 2. In Section 3, we describe our design goals. In Section 4, we describe the design of the proposed system and a prototype implementation of the system. In Section 5, we describe the user interface program built on Squeak. In Section 6, we present performance simulations. In Section 7, we briefly review related work. Finally, in Section 8, we describe some concluding remarks.

2. SUPPOSED SITUATION

As mentioned in the previous section, we focus on asynchronous WBT, that is to say, a user can connect to the proposed e-Learning system anytime and anywhere he/she wants. Once connection is established, the user can obtain exercises one after another through specifying categories of the required exercises. User's answers for each exercise are marked as correct or incorrect right away. Extra information may be provided for each answer, which can be viewed when the correct answer is shown.

While a user uses the proposed e-Learning system, his/her computer is a part of the system. Namely, it receives some number of categories and exercises in them from another node when it joins the system and has responsibility to send appropriate exercises to requesting nodes.

The important point to note is that the categories a node has are independent of the categories in which the node's user are interested as shown in Figure 1. Figure 1 illustrates that user A's request is forwarded at first to the neighbor node, next forwarded to the node which has the requested category.

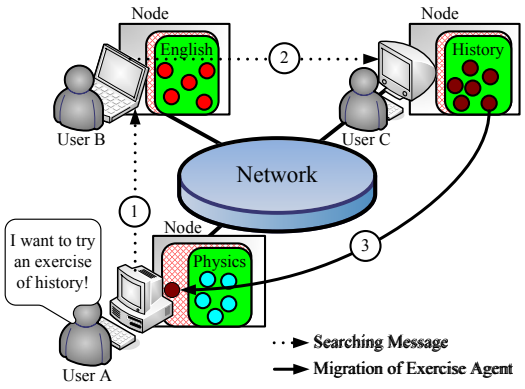

Figure 1 - Proposed e-Learning system.

3. BASIC CONCEPTS

As mentioned above, the proposed system has two distinguishing features. Firstly, it is based on P2P architecture. Secondly, each exercise is not only data but also an agent so that it can mark user's answers, tell the correct answers, and show some extra information about the exercise. In this section, we describe these features in detail.

P2P Aspect

All exercises in the proposed system are classified into categories such as "Mathematics / Expression / Equation", "English / Grammar", and "History / Rome", etc.

When the proposed system begins, one initial node has all categories in the system. When another node joins the system, it is received some number of categories from the initial node. The categories are distributed among all nodes in the system according as nodes join the system or leave the system.

We would like to emphasize that in existing P2P-based file sharing systems such as Napster[3], Gnutella[4], and Freenet[5] each shared file is owned by a particular node. Accordingly, files are originally distributed among all nodes. On the other hand, the categories in the proposed system are originally concentrated. Consequently, when a new node joins the system, not only location information of a category but the category itself must be handed to the new node. Considering that, the P2P network of the proposed system can be constructed as a CAN[6].

A CAN has a virtual coordinate space that is used to store $(key, value)$ pairs. To store a pair (K_1, V_1), key K_1 is deterministically mapped onto a point P in the coordinate space using a uniform hash function. The corresponding $(key, value)$ pair is then stored at the node that owns the zone within which the point P lies. In the proposed system, we let each category be a key and let a set of exercises belonging to the category be the corresponding value.

Mobile Agent Aspect

Generally, in addition to service to show an exercise, a WBT server provides services to mark the user's answers, tell the correct answers, and show some extra information about the exercise. Therefore, for the proposed system which can be considered a distributed WBT system, it is not enough that only exercises are distributed among all nodes. Functions to provide the above services also must be distributed among all nodes. We adopt mobile agent technology to achieve this goal. Namely, an exercise is not only data but also an agent so that it can mark user's answers, tell the correct answers, and show some extra information about the exercise.

In addition, mobile agent technology is applied to realize the migration of categories, that is, each category is also an agent in the proposed system.

4. DESIGN AND IMPLEMENTATION

We have implemented a prototype of the proposed system on Maglog that is a Prolog-based framework for building mobile multi-agent systems we have developed[7].

As shown in Figure 2, a node consists of the following agents and a user interface program. The components of a node are divided into two type, those that move to other node referred as mobile components in Figure 2, and those that keep their station referred as stational components in Figure 2.

Node Agent There is one node agent on each node. It manages the zone information of a CAN and forwards messages to the category agents in the node.

Category Agent Each category agent stands for a unit of a particular subject. It manages exercise agents in itself and sends them to the requesting node.

Exercise Agent Each exercise agent has a question and functions to mark user's answers, tell the correct answers, and show some extra information about the exercise. These data are formatted in HTML.

Interface Agent There is one interface agent on each node. It is an interface between the user interface program and other agents.

Agents communicate with other agents through 'field's provided by Maglog framework. A field is kind of a preemptive queue. Roughly speaking, the above-mentioned four kinds of agents execute a message dispatch loop. Each message to an agent is queued into the field owned by the agent. The user interface program also communicates with the interface agent through a field via XML-RPC[8]. Table 1 shows a partial summary of message types.

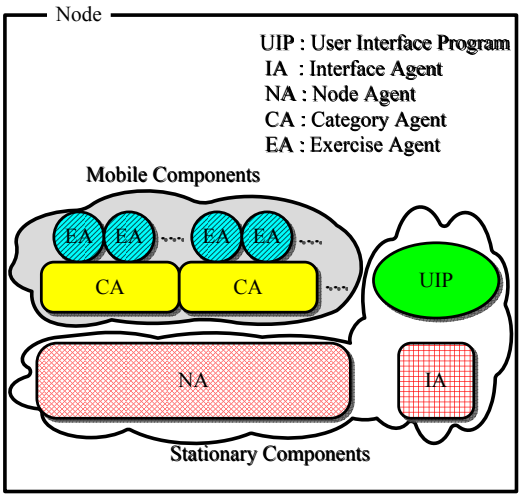

Figure 2 - Architecture of a node.

Table 1 - Partial summary of message types.

Dispatcher	Type	Description
Node	join	To join the system.
Node	leave	To leave the system.
Node	update	To update the neighbor's zone information.
Node	request	To get an exercise agent.
Category	add	To add an exercise agent.
Category	go	To go to another node.
Category	send	To send an exercise agent.
Category	receive	To receive an exercise agent.
Exercise	go	To go to the requesting node.
Exercise	show	To show a question.
Exercise	mark	To mark a user's answer.
Exercise	answer	To show the correct answer.
Exercise	info	To show the extra information.
Interface	retrieve	To get an exercise agent.
Interface	release	To let an exercise agent go home.
Interface	arrived	To be notified the arrival of an exercise agent.

5. User Interface Program

Features

As mentioned above, the user interface program of the proposed system has been developed through extending Scamper which is a simple web browser runs in Squeak[9].

Figures 3, 4, 5, 6, 7, 8, 9, and 10 are screen-shots of the user interface program. The main window of it consists of three panes as shown in 3. Firstly, the button pane includes three buttons to get exercises. Secondly, the category pane shows categories of exercises. Thirdly, the exercise pane shows an exercise. Categories are classified into a hierarchical tree structure, as shown in Figure 4. By clicking the left button of a mouse on the category, a user can select it. After selection of the category, a user can obtain an exercise belonging to the category by clicking the left button of a mouse on one of buttons in the button pane. After a while an appropriate exercise agent comes from some node and the user can try the question as shown in Figure 5. The user can require to mark his/her answer anytime by clicking the submit button. Figure 6 shows an example result of marking. Figure 7 shows the correct answers and extra information about the exercise that are shown by clicking the answer button. If a user can not understand the correct answers he/she can request for help to other on-line users as shown in Figure 8. The interface agent connected to the user interface program creates an agent and make it visit all nodes to search for an appropriate user who has answered the problem correctly. The selected user is asked whether he or she is willing to help as shown in Figure 9, and if the user answers with 'Yes', a chat connect is established as shown in Figure 10.

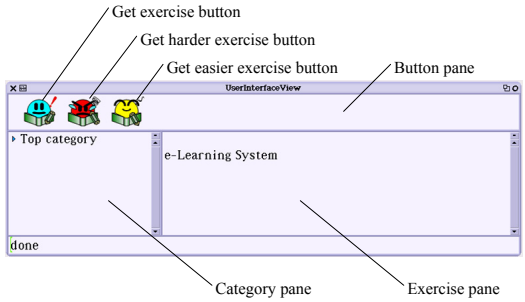

Figure 3 - The window of the user interface program.

Through the interactiveness of Squeak, a user "do it" a sentence in the answer window which includes the correct answers and extra information about the exercise that he/she is trying shown as Figure 7.

In addition, a user can operate the user interface program through not only GUI but also sending messages to the object that realize the user interface program. Figure 11 shows an example of sending messages to the user interface object through a workplace.

Implementation

The user interface program consists of the following classes.

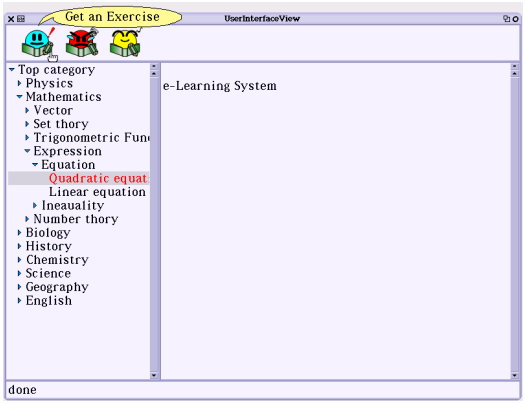

Figure 4 - An exercise can be obtained by clicking the left button of a mouse on a buttons in the button pane.