Proceedings of the 19th IASTED International Conference
Parallel and Distributed Computing and Systems
November 19–21, 2007, Cambridge, MA, USA
ISBN Hardcopy: 978-0-88986-703-1 / CD: 978-0-88986-704-8

EXERCISE MANAGEMENT SCHEME FOR A DISTRIBUTED E-LEARNING SYSTEM

Takao KAWAMURA, Kazuo KURAMOCHI, and Kazunori SUGAHARA
Department of Information and Knowledge Engineering
Tottori University
4–101, Koyama-Minami
Tottori, JAPAN
kawamura@ike.tottori-u.ac.jp

ABSTRACT
We have proposed and implemented a distributed asynchronous Web-based training (WBT) system. In order to improve the scalability and robustness of this system, all exercises and functions, such as scores user's answers are realized on mobile agents. These agents are distributed to computers, and they can be constructed with a P2P network that modified Content-Addressable Network (CAN). In this paper, we present the exercise management scheme for the proposed WBT system. In a WBT system based on client/server model, management of exercises is simply achieved by manipulating data, since all contents are concentrated in one server computer. In the proposed system, however, we need to pay attention to distributed agents for management of exercises. In order to achieve operation of exercise management, i.e., adding, deleting and updating exercises on the distributed WBT system, we use steps with considering multi-agent based distributed environment: specifying an agent which provides an exercise and searching its node as pre-operation, and sending an agent to the expected node and notifying other cooperating agents as post-operation.

KEY WORDS
Distributed Agents, P2P, Mobile Agent, e-Learning

1 Introduction

Nowadays, e-Learning systems, especially asynchronous Web-based training systems (hereafter we abbreviate as WBT) are very popular. A WBT 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, 3], all of them are based on the client/server model. The client/server systems generally lack scalability and robustness. In recent years, P2P research has grown exponentially. Although the current P2P systems are well-known for its file sharing ability, and the consequent legal problems, P2P systems are gradually proving themselves 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.

We have proposed and implemented a distributed e-Learning system based on P2P architecture [4, 5] using Maglog that is a Prolog-based framework for building mobile multi-agent systems we have also developed [6, 7]. The proposed e-Learning 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 the requesting nodes. Secondly, each exercise in the system is not only data but also an agent so that it has functions, such as scoring user's answers, telling the correct answers, and showing some related information without human instruction.

In this paper, we present an exercise management scheme which provides three operations to add, to delete and to update exercises for the proposed e-Learning system. In ordinal WBT systems based on the client/server model, all exercises are stored in a server just as data. Therefore, adding, deleting and updating exercises are achieved by manipulating data on the server directly. In the proposed system, on the other hand, exercises are provided by agents which move around any computer of the system. The following pre-operation and post-operation steps are therefore needed when an exercise manager intends to add, delete, or update an exercise:

Pre-Operation Step This step is needed when the intended operation is deleting or updating an exercise. First of all, the agent is specified in which the target exercise is contained. Next, the node is searched on which the agent is running. In addition, when the operation is updating an exercise, the target agent migrates to the exercise manager's node.

Post-Operation Step When the operation is adding or updating an exercise, the target agent migrates to appropriate node. In addition to that, since all functions of the proposed e-Learning system are preformed in co-

operation with multi-agents, it is needed to notify appropriate agents about the operation, i.e., an exercise agent is added, deleted, or updated.

This paper is organized in 5 sections. The proposed e-Learning system is described in Section 2. We describe the design overview of the proposed exercise management scheme in Section 3 and the implementation of the scheme in Section 4, respectively. Finally, some concluding remarks are drawn in Section 5.

2 Proposed e-Learning System

2.1 Overview

All exercises in the proposed system are classified into categories, such as "English/Grammar", "Math/Statistic", and "History/Rome", etc. A user can obtain exercises one after another through specifying categories of the required exercises.

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 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.

2.2 P2P Network

When the proposed system bootstraps, one initial node has all categories in the system. When another node joins the system, it is received certain 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 [8], Gnutella [9], and Freenet [10], 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 [11].

The 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 in 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.

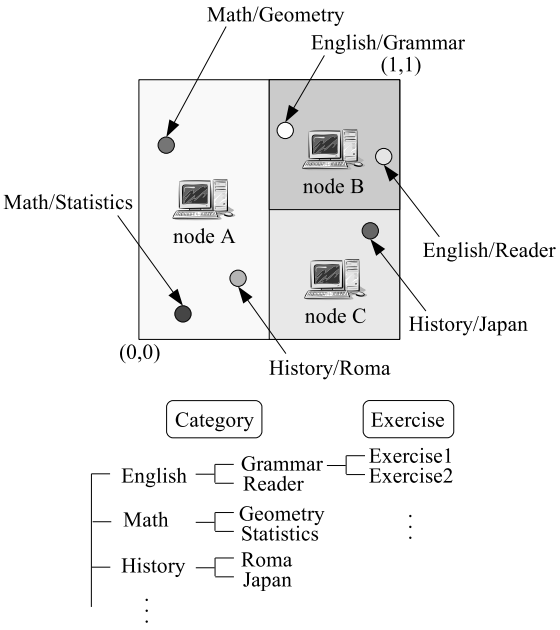

Figure 1. P2P network.

Our P2P network is constructed with 2-dimensional coordinate space [0,1] $\times$ [0,1] to store exercise categories, as shown in Figure 1. The figure shows the situation that node C has just joined the system as the third node. Before node C joins, node A and node B shared the whole coordinate space half and half. At that moment, node A managed "Math/Geometry", "Math/Statistics", and "History/Rome" categories and node B managed "English/Grammar", "English/Reader and "History/Japan" categories, respectively. When node C joins the system, it is mapped on a certain coordinate space according to a random number and takes on corresponding categories from another node. For example, in the case of Figure 1, node C takes on the "History/Japan" category from node B and exercises of the category move to node C.

2.3 Components

Generally, in addition to service to show an exercise, a WBT server provides services to score user's answers, to tell the correct answers, and to show some related information about the exercise. Therefore, for the proposed system that can be considered as 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 has functions, such as scoring user's answers, telling the correct answers, and showing some related 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.

There are following agents and user interface programs on each node.

Node Agent Each node has one node agent. It manages the zone information of a CAN and forwards messages to the category agents in the node.

Exercise Agent Each exercise agent has a question and functions to score user's answers, tell the correct answers, and show some related information about the exercise. An exercise agent records study logs whenever it is tried by a student.

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.

Interface Agent There is one interface agent for each user interface, such as a student interface and an exercise manager interface on each node. It plays a role of interfaces between the interface program and other agents, and between agents and applications.

User Agent Each user has its own user agent. A user agent manages its user information that includes login name, password, IP address of the user's computer, online/offline status. It also has study logs if the corresponding user is a student. Similarly, it also has a list of created exercises if the corresponding user is an exercise manager.

Group Agent A group agent exists for grouping user agents as a category agent exists for grouping exercise agents. Currently, users are grouped by the first letter of their login name, so that there are 26 group agents corresponding to one letter in the English alphabet.

Student Interface One student interface is on each node of which a user logs in as a student. It is a user interface program for studying.

Exercise Manager Interface One exercise manager interface is on each node of which a user logs in as an exercise manager. It is a user interface program for management of exercises.

3 Exercise Management Scheme

3.1 Overview

The proposed exercise management scheme provides three operations, i.e., adding, deleting, and updating exercises, to exercise managers that are users authorized to manage exercises in the proposed e-Learning system. Note: Updating an exercise-A means creating new exercise-B based on exercise-A instead of modifying exercise-A directly.

Since exercises are represented as mobile agents, the pre-operation and post-operation steps described in Section 1 are needed.

An exercise manager can use those functions through an exercise manager interface.

3.2 Pre-Operation: Specify an Exercise Agent and Its Node

This step is needed when the intended operation is deleting or updating an exercise.

Each agent in the proposed system is identified by unique agent-ID. However, it is not convenient for an exercise manager to specify an exercise by agent-ID. For that reason, we would like to manage a list of the agent-ID and the properties of an exercise, and we'd like to use the list for specifying target exercise agent by properties. If the list is stored and maintained as a local data, the list cannot be used in the other nodes. Therefore, a user agent corresponding to an exercise manager maintains a list of all exercises created by the exercise manager.

User agents are registered to group agents by the first letter of exercise managers' user-ID. Group agents are also distributed to nodes and arranged in a virtual coordinate space of the P2P network by the letter. When an exercise manager logs in the e-Learning system using his/her user-ID and password, a group agent authenticates him/her, then a user agent corresponds to the manager migrates to the manager's node. The user agent maintains a list of all exercises created by the exercise manager. Each record of the list consists of the agent-ID, and the properties of an exercise such as created date, category name, exercise title, and short description. An exercise manager can specify and select target exercise agent for deleting or updating from this list through the exercise manager interface.

In order to respond to a request for exercise management, a request message of an exercise manager have to be delivered to a category agent, because a category agent manages exercise agents. However, no one controls the whole proposed e-Learning system, agents on each node do not know which node the category agent are arranged in. Then, we use a virtual coordinate space of our P2P network to search a node to which a category agent belongs. When a node of a category is being searched, the following steps are performed:

1. A node of an exercise manager is assigned to node N.
2. A point of a category in the virtual coordinate space is calculated using a uniform hash function and a name of the category, and then the point of category is assigned to point P.
3. If a zone of node N includes the point P, then exit.
4. A node of neighbors which has closest zone to a point P is assigned to node N. Where, two nodes are neighbors if and only if their zones overlap the same side.
5. Go to step 3.

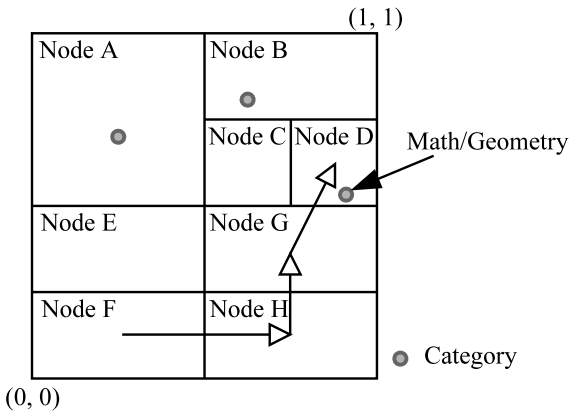

Figure 2. Example of searching a node of "Math/Geometry" category.

In Figure 2, an example of searching a node of "Math / Geometry" by an exercise manager in node F is shown. First, node F is assigned to node N. A point of "Math / Geometry" is not included in the zone of node F, then node H is assigned to node N since node H is one of the neighbors of node F's which has zone closest to the point P. In the same way, node G is assigned to node N, and then node D is assigned to node N. Node D has zone including P, then the node of "Math / Geometry" is specified as node D.

After the above steps, a request of exercise management is transfered to the category agent of the target category in the specified node.

3.3 Pre-Operation: Obtaining an Exercise Agent for Updating

This step is needed when the intended operation is updating an exercise.

A category agent that receives an updating request through the searching procedure described in Section 3.2, it sends a request to the target exercise agent specified by an agent-ID to migrate to the exercise manager's node. The exercise agent that receives the request duplicates itself. Then, the cloned agent migrates to the exercise manager's node and presents the contents of itself to the exercise manager interface.

3.4 Operation: Adding, Updating and Deleting an Exercise Agent

In the process of adding a new exercise, an exercise manager creates materials for an exercise from scratch. In the process of updating an exercise, on the other hand, an exercise manager receives exercise contents from the migrated exercise agent, and then creates new materials based on the received contents.

Once materials for an exercise are prepared, the remaining steps are same in both adding and updating. Firstly, new exercise agent is created using the materials. Secondly, the created exercise agent is registered in the system, as described in the following section.

The process of deleting a specified exercise is also requested through an exercise manager interface.

3.5 Post-Operation: Registering or Unregistering an Exercise Agent

The process of registering an exercise agent is needed when the intended operation is adding or updating an exercise.

This process is also accomplished by using the searching procedure. After searching a node of the target category by the steps described in Section 3.2, the newly created exercise agent migrates to specified node of the category.

In the proposed system, exercise agents are managed by category agents and lists of created exercises are maintained by user agents. Therefore, to inform adding or deleting exercise to the proposed e-Learning system is achieved by the following procedures:

1. The created exercise agent is registered in its category agent when the operation is adding or updating. In contrast, the exercise agent is unregistered from its category agent when the operation is deleting. Registered exercise agents are provided for students immediately, while unregistered exercise agents possibly remain for a while. An unregistered exercise agent disappears completely from the proposed e-Learning system when the last clone of it returns to its category agent.

2. The record corresponding to the created exercise is added to the list maintained by the user agent when the operation is adding or updating. In contrast, the record corresponding to the deleted exercise is marked as 'deleted' on the list of the user agent.

4 Implementation of Exercise Management Scheme

4.1 Overview

Figure 3 illustrates the implementation overview of the proposed e-Learning system that runs on Java platform. Each node runs an agent server written in Java and all agents in a node are threads in the agent server process. Both two of interface applications, i.e., student interface and exercise manager interface are also written in Java.

4.2 Exercise Agent

Functions of an agent are written in Prolog, compiled to Java classes, and then combined into a jar file. The jar file

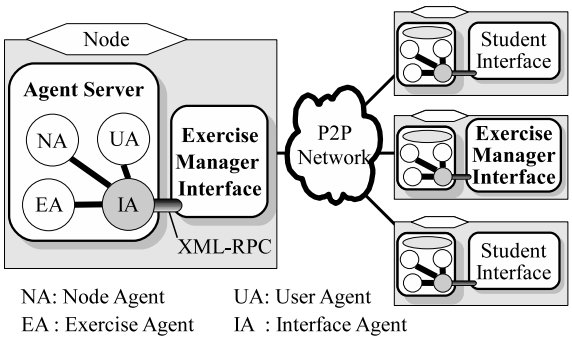

Figure 3. Implementation overview of the proposed e-Learning system.

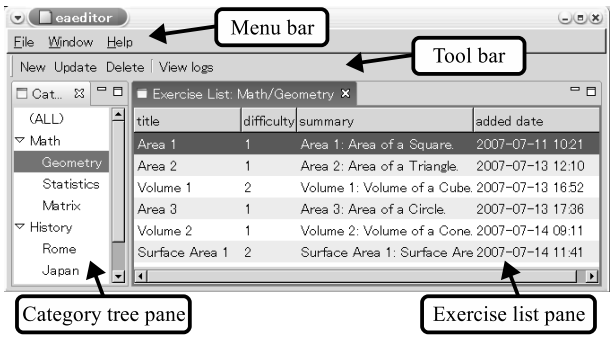

Figure 4. Sample screenshot: Viewing a list of exercises.

also contains the following two types of data:

1. XML document which express questions and comments with display layout, and values of correct answers.
2. Image data associated with the exercise.

An example of XML document in contents of an exercise is shown in Listing 1. In Listing 1, there are <head> element and <body> element as children of <exercise> element. The <head> element has attribute information of the exercise and the <body> element includes main contents of the exercise. The <body> element can include <ansfld> element, which shows an answer form of a question, <img> element, which is for image data, and some other elements as posterities of <body> element.

Listing 1. An example of the XML document in contens of an exercise.

```
<?xml version="1.0" encoding="UTF−8"?>
<exercise>
  <head>
    <category>Math/Geometry</category>
    <title>Area 4</title><difficulty>2</difficulty>
    <summary>Area of a Rectangle: 1</summary>
  </head>
  <body>
    <p><img name="" src="ea−51665.png"/></p>
    <p>What is the area of the above rectangle?</p>
    <p>
      <ansfld class="text">
        <ans>
          <val type="text">4</val>
          <disp>4</disp>
        </ans>
        <cmt>The rectangle has ...</cmt>
      </ansfld>
    </p>
  </body>
</exercise>
```

XML document and image data are transformed into HTML format for viewing with a web browser using XSL Transforms[12]. This transformation may be applied each time when the presentation format is needed. However, we make the presentation format be included in the jar file to reduce waiting time for viewing the content.

4.3 Exercise Manager Interface

An exercise manager interface provides all operations of exercise management scheme for users. To have the exercise manager interface run under multi-platform environment and to make it easy to understand the usage, we have implemented it as an application of Eclipse Rich Client Platform [13], a framework for building rich client application. The Eclipse Rich Client Platform is useful not only for client/server systems but also for distributed systems, such as the proposed e-Learning system.

Figures 4, 5, and 6 show sample screenshots of the exercise manager interface. Figure 4 illustrates the main window of the exercise manager interface. All exercises created by an exercises manager are listed. Functions adding, deleting, and updating exercises or viewing logs of exercises are executed through selecting an item of menu bar. Figure 5 illustrates the editing window through which an exercise manager create a new exercise or update an existing exercise. Figure 6 illustrates the log window through which an exercise manager can view the study logs of an exercise agent.

An exercise manager interface and agents communicate each other via XML-RPC [14]. Requests from an exercise manager interface are sent to an interface agent which delegates requests to other agents.

5 Conclusion

Since existing asynchronous WBT systems are based on the client/server model, they have problems of scalability and robustness. The proposed e-Learning system solves these problems in decentralized manner through both P2P technology and mobile agent technology.

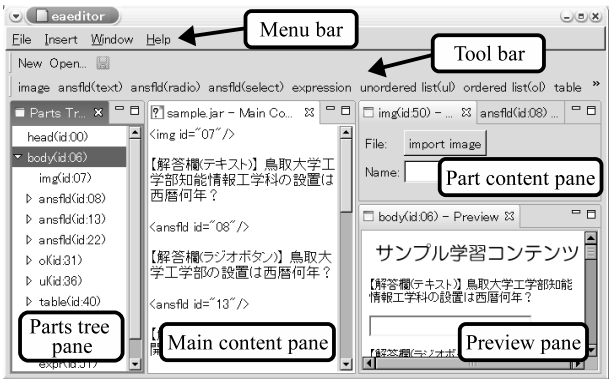

Figure 5. Sample screenshot: Creating a new exercise.

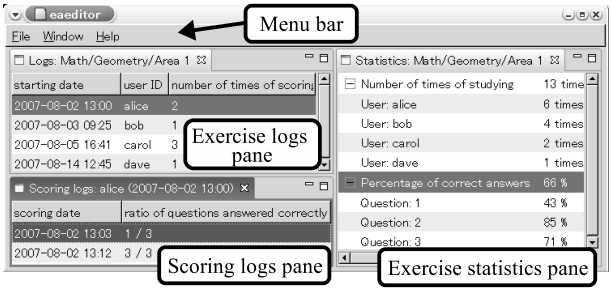

Figure 6. Sample screenshot: Viewing study logs of an exercise agent.

In this paper, we describe exercise management scheme for the proposed e-Learning system. The proposed management scheme provides three operations, i.e., adding, deleting and updating exercises for authorized users. To provide these operations as functions of the proposed distributed e-Learning system, we use following methods: a method to specify a target agent and to let the agent migrate from a searched appropriate node to a user's node, as pre-operation, and a method to send newly created agent to a proper node and to notify adding or deleting an exercise to cooperating agent, as post-operation.

We implement an exercise manager interface as an application of Eclipse Rich Client Platform. An exercise manager interface provides adding, deleting and updating exercises for authorized users. An exercise manager interface communicates with an agent server via XML-RPC.

In the proposed e-Learning system, the data stored in the inside of an agent is initialized when the e-Learning system is restarted. Handing over these runtime data to the rebooted system is left for future work.

References

[1] Nishita, T. and et el: Development of a Web Based Training system and Courseware for Advanced Computer Graphics Courses Enhanced by Interactive Java Applets (2002).

[2] Homma, H. and Aoki, Y.: Creation of WBT Server on Digital Signal Processing, *Proceedings of 4th International Conference on Information Technology Based Higher Education and Training* (2003). Marrakech, Morocco.

[3] Helic, D., Krottmaier, H., Maurer, H. and Scerbakov, N.: Enabling Project-Based Learning in WBT Systems, *International Journal on E-Learning*, Vol. 4, No. 4, pp. 445–461 (2005).

[4] Kawamura, T. and Sugahara, K.: A Mobile Agent-Based P2P e-Learning System, *IPSJ Journal*, Vol. 46, No. 1, pp. 222–225 (2005).

[5] Motomura, S., Nakatani, R., Kawamura, T. and Sugahara, K.: Distributed e-Learning System Using P2P Technology, *Proceedings of the 2nd International Conference on Web Information Systems and Technologies*, pp. 250–255 (2006). Setubal, Portugal.

[6] Motomura, S., Kawamura, T. and Sugahara, K.: Maglog: A Mobile Agent Framework For Distributed Models, *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 414–420 (2005). Phoenix, Arizona, USA.

[7] Motomura, S., Kawamura, T. and Sugahara, K.: Logic-Based Mobile Agent Framework with a Concept of "Field", *IPSJ Journal*, Vol. 47, No. 4, pp. 1230–1238 (2006).

[8] Napster: http://www.napster.com/ (1999).

[9] Gnutella: http://welcome.to/gnutella/ (2000).

[10] Clarke, I., Sandberg, O., Wiley, B. and Hong, T. W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System, http://freenetproject.org/papers/freenet.pdf (2000).

[11] Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Schenker, S.: A scalable content-addressable network, *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, ACM Press, pp. 161–172 (2001).

[12] Clark, J.: XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt (1999).

[13] McAffer, J. and Lemieux, J.-M.: *Eclipse Rich Client Platform: Designing, Coding, and Packaging Java™ Applications*, Addison-Wesley (2005).

[14] Winer, D.: XML-RPC Specification, http://xmlrcp.com/spec (1998).