Proceedings of the 18th IASTED International Conference
Parallel and Distributed Computing and Systems
November 13–15, 2006, Dallas, TX, USA
ISBN Hardcopy: 0-88986-638-4 / CD: 0-88986-640-6

BACKUP AND RECOVERY MECHANISM FOR A DISTRIBUTED E-LEARNING SYSTEM

Takao KAWAMURA, Syungo KINOSHITA, Shinichi MOTOMURA, 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 developed a distributed asynchronous Web based training system. In order to improve the scalability and robustness of this system, all contents and a function that scores user's answers are realized on mobile agents. These agents are distributed to computers, and they can obtain using a P2P network that modified Content-Addressable Network. In this system, although entire services do not become impossible even if some computers break down, the problem that contents disappear occurs with an agent's disappearance. In this study, as a solution for this problem, backups of agents are distributed to computers. If a failure of a computer is detected, other computer will continue service using backups of the agents belonged to the computer. The developed algorithm is examined by experiments.

KEY WORDS
Distributed Software Systems and Applications, e-Learning, Mobile Agent,Fault Tolerance

1 Introduction

Nowadays, e-Learning systems are very popular in everywhere, such as college educations, corporate educations, and community educations. 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, 3], 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 file sharing, 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.

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 that 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 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 the proposed system, since exercises and functions are distributed among all nodes, the loads are balanced and not concentrated on one node. In addition, the proposed system can be considered more robust than the client/server systems because even if a node failure occurs, the rest of the system will still continue service. However, when a node failure occurs, the exercises in the node are lost and cannot be studied by anyone afterward.

In this study, as a solution for this problem, backups of agents are distributed to nodes. When a node failure occurs, another node continues service using backups of the agents belonged to the failure node. Our algorithm always tolerates one node failure and tolerates more than one node failure under the condition described below.

This paper is organized in 6 sections. The proposed e-Learning system is described in Section 2. We describe a node joining and leaving procedures without the node backup and recovery mechanism of the P2P network in Sections 3. Subsequently, in Section 4, we explain the node backup and recovery mechanism and describe how a node joining and leaving procedures are modified for the mechanism. In Section 5, we present experimental results to confirm our approach. Finally, some concluding remarks are drawn in Section 6.

2 Proposed e-Learning System

2.1 Overview

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 scored as correct or incorrect right away. Related 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 first to the neighbor node, next forwarded to the node which has the requested category.

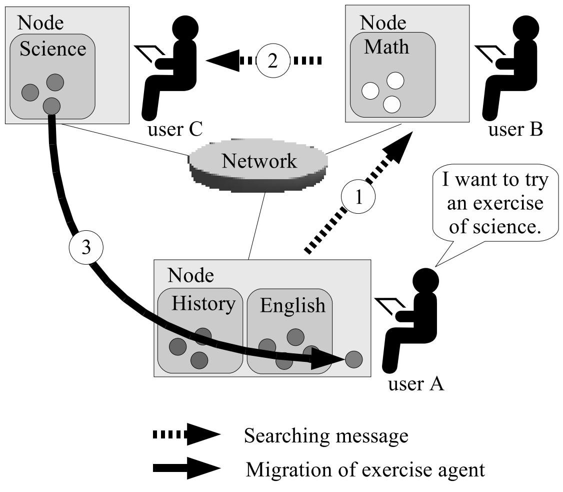

Figure 1. Proposed e-Learning system.

2.2 P2P Network

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

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 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.

Our P2P network is constructed with 2-dimensional coordinate space [0,1] $\times$ [0,1] to store exercise categories, as shown in Figure 2. 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 2, node C takes on the "History/Japan" category from node B and exercises of the category move to node C.

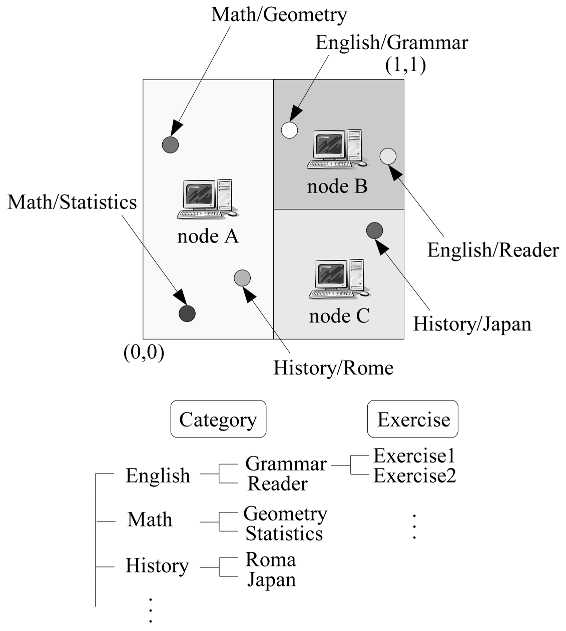

Figure 2. P2P network.

2.3 Mobile Agents

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 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.

3 Joining and Leaving Procedures without a Backup and Recovery Mechanism

3.1 Preliminary

Definition 1 *Two nodes in the proposed P2P network are neighbors if and only if their zones overlap along the X or Y axes. A set of neighbor nodes of a node n is defined as $N(n)$.*

For example, in Figure 2, N(node A) is {node B, node C}, N(node B) is {node A, node C}, and N(node C) = {node A, node B}.

Definition 2 *A procedure for informing the node n's zone information to $N(n)$ is defined as $update(n)$. The zone information of node n consists of the zone coordinates of node n and $N(n)$. The zone coordinates consists of the upper left point and the lower right point of the zone.*

3.2 Joining Procedure

When a node n intends to join the P2P network, the following steps are performed:

1. Node n randomly chooses a point P in the 2-dimensional coordinate space and sends a join request destined for point P. This message is sent into the P2P network via any existing node. Each node then uses the CAN routing mechanism[11] to forward the message, until it reaches node n_p in whose zone P lies. It is assumed that at least one IP address of existing nodes in the P2P network is known by every node.

2. Node n_p then splits its zone in half and assigns one half to node n. Categories and exercises in the half zone are also transferred to node n.

3. Having obtained its zone, node n learns the IP addresses of $N(n)$ from node n_p. Similarly, node n_p updates its $N(n_p)$ to eliminate those nodes that are no longer neighbors.

4. Nodes n and n_p execute $update(n)$ and $update(n_p)$ respectively to inform their neighbors about this reallocation of space.

Figure 3 shows an example of node J joining the P2P network. First, node J chooses a point (x, y), which is notated as P in the above procedure, in the 2-dimensional coordinate space randomly. Next, node J sends a join request to node D of which node J is assumed to know the IP address. This message is forwarded to node F, which is notated as n_p in the above procedure, whose zone the point (x, y) lies as shown in Figure 3 (a). Finally, node F splits its zone in half and assigns one half that the point (x, y) lies to node J as shown in Figure 3 (b).

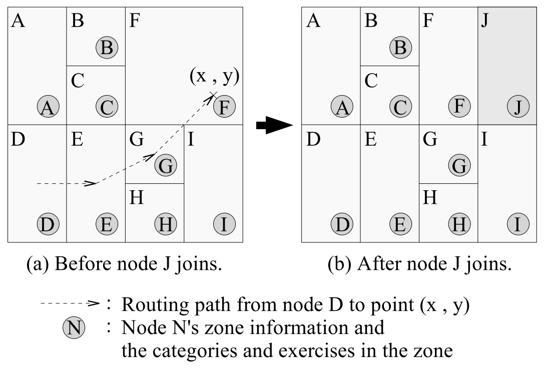
A
B
F
C
(x , y)
D
E
G
I
H
J
(a) Before node J joins.
(b) After node J joins.
Routing path from node D to point (x , y)
Node N's zone information and the categories and exercises in the zone

Figure 3. Change of the P2P network when node J joins.

3.3 Leaving Procedure

When a node n intends to leave the P2P network, the following steps are performed:

1. If there is node m in $N(n)$ where the volume of node n's zone equals the volume of node m's zone then go to step 4 otherwise go to step 2. We refer to such two neighbor nodes whose volumes are equal as a pair of nodes.

2. Node n forwards a search request to a node in $N(n)$ whose volume is smallest. This process repeats until a pair of nodes is found. Let the two nodes of the pair be nodes p and q.

3. Node n and p exchange their zones and execute $update(n)$ and $update(p)$, respectively.

4. The zones of node n and q are merged into a single zone that is assigned to node q. Node q update its $N(q)$ and executes $update(q)$. Categories and exercises in node n's zone are also transferred to node q.

Figure 4 shows an example of node B leaving the P2P network. In this example, because only nodes C and D are mergeable, nodes B and D exchange their zones first as shown in Figure 4 (b), where nodes B, C, and D correspond to nodes n, q, and p in the above procedure, respectively. Next the zones of nodes B and C are merged into a single zone that is assigned to node C as shown in Figure 4 (c).

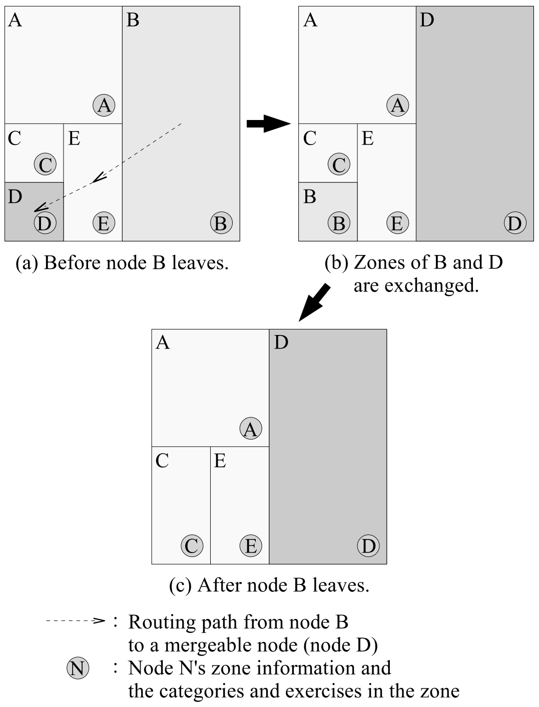
A
B
C
E
D
(a) Before node B leaves.
(b) Zones of B and D are exchanged.
(c) After node B leaves.
Routing path from node B to a mergeable node (node D)
Node N's zone information and the categories and exercises in the zone

Figure 4. Change of the P2P network when node D leaves.

4 Backup and Recovery Mechanism

4.1 Backup of a Node

We intend to maintain backups of all nodes in decentralized manner. To achieve this goal, one of the neighbor nodes of each node takes its backup. If node n takes a backup of node m, node n is called a backup node for node m, while node m is called an original node for node n. An original node knows which node a backup node for it is, and vice versa. A backup of a node consists of its zone information and the categories and exercises in the zone. A neighbor node that has the minimum number of agents is selected for a backup node to balance the load of the system. The number of agents in each node is published always.

The joining procedure described in Section 3.2 should be modified for backup, i.e., a step for making a backup of the joining node should be added after the step 4 of the joining procedure. However, it is not enough. Figure 5 shows an example of node J joins the P2P network with the backup and recovery mechanism. In this example, node J obtains its zone from node F. As the result, node F's zone is changed so that the backup of node F becomes invalid. Node F therefore requests to node C, which is a backup node of node F, for throwing its backup away, and then remakes its backup. A circle in which white letters are printed on a dark background represents a remade backup in Figure 5.

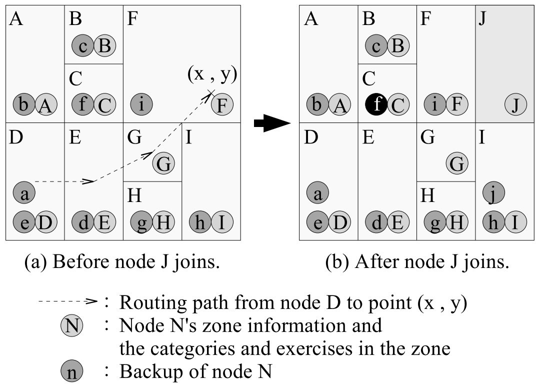
A
B
F
J
C
D
E
G
I
H
(x , y)
(a) Before node J joins.
(b) After node J joins.
Routing path from node D to point (x , y)
Node N's zone information and the categories and exercises in the zone
Backup of node N

Figure 5. Change of the P2P network with a backup and recovery mechanism when node J joins.

Similarly, the leaving procedure described in Section 3.3 should also be modified for backup, i.e., all backups of the affected nodes by the leaving should be thrown away and be remade. Figure 6 shows an example of node B leaves the P2P network with the backup and recovery mechanism. In this example, all backups except of node A are thrown away and remade. A circle in which white letters are printed on a dark background represents a remade backup in Figure 6 as same as in Figure 5. Note that Figure 6 doesn't mean backups are remade at once after the exchanging. It cannot be known when each backup is remade since each node works asynchronously.

Clearly, our algorithm always tolerates one node failure. Let us consider more than one node failure at a time. Naturally, it is impossible that recovery from the failures that occur in both an original node and its backup node at the same time. Our system however can be recovered from other types of combinations of failures.

4.2 Recovery from a Node Failure

Each node n_i execute $update(n_i)$ periodically. The prolonged absence of a message of $update(n_i)$ from a neighbor signals its failure. If a backup node n detects the failure of its original node m, the following steps are performed:

1. Node n generates a temporary node from the backup of m.
2. The temporary node executes the leaving procedure. Consequently, node m's zone is formally handed over

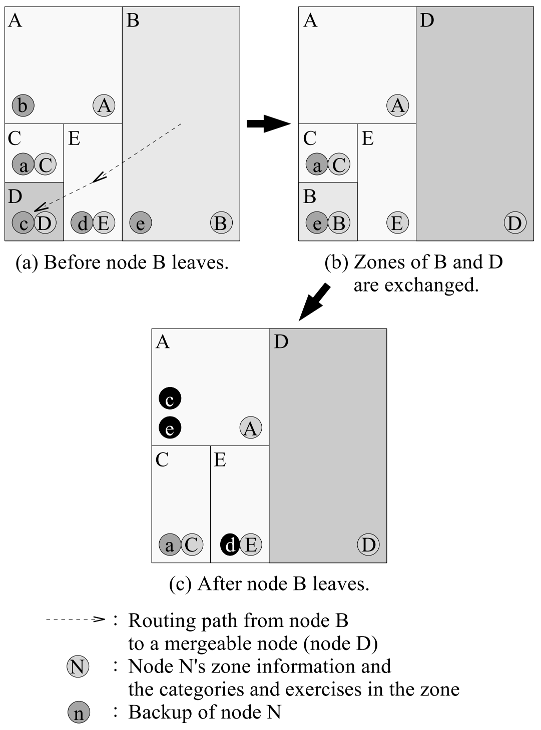
(a) Before node B leaves.
(b) Zones of B and D are exchanged.
(c) After node B leaves.
Routing path from node B to a mergeable node (node D)
Node N's zone information and the categories and exercises in the zone
Backup of node N

Figure 6. Change of the P2P network with a backup and recovery mechanism when node D leaves.

to one of the neighbor nodes.

If an original node detects the failure of its backup node in the contrast, it simply makes a new backup of itself.

Figure 7 shows an example of node B fails. It is similar to the case of leaving shown in Figure 6. The only difference is that a temporary node (node Temp in Figure 7) stands for node B in this case of recovery. The temporary node is generated by node A in this example.

5 Experiments

This section presents the experimental results obtained with the implementation of the proposed backup and recovery mechanism for our distributed e-Learning system. For the experiments, 15 PCs with Intel Pentium4 3.0GHz processor and 1GB of RAM are connected via 1000Base-T network. All the PCs are running on GNU/Linux (kernel version is 2.6.0) operating system.

We calculate the variance of the numbers of agents including backups in each node in the experimental environment under the conditions shown in Table 1. The variance decreases as the number of nodes increases as shown in Figure 8. A lower variance indicates higher load balance of the proposed system.

Next we investigate how the total amount of bytes of transferred agents changes, with or without the back-

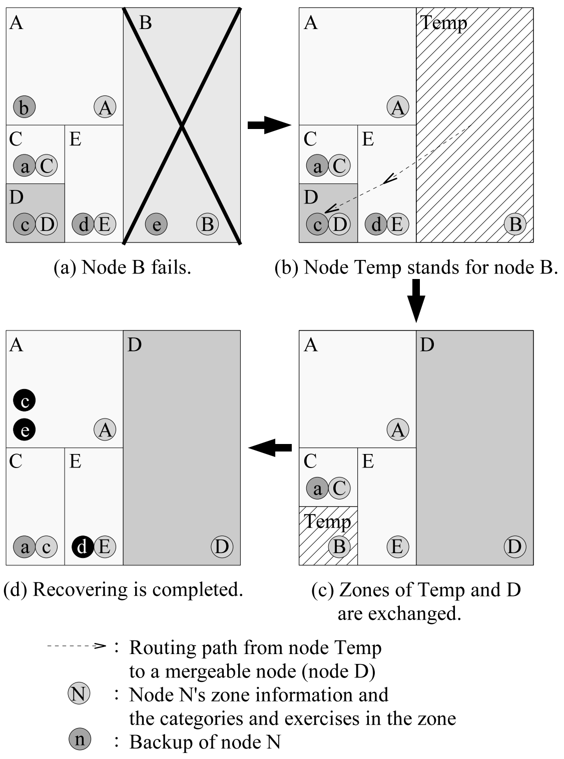
(a) Node B fails.
(b) Node Temp stands for node B.
(c) Zones of Temp and D are exchanged.
(d) Recovering is completed.
Routing path from node Temp to a mergeable node (node D)
Node N's zone information and the categories and exercises in the zone
Backup of node N

Figure 7. Change of the P2P network when node B fails.

Table 1. Experimental Conditions.

Number of Nodes	1–15
Number of Categories	50
Number of Exercises/Category	3

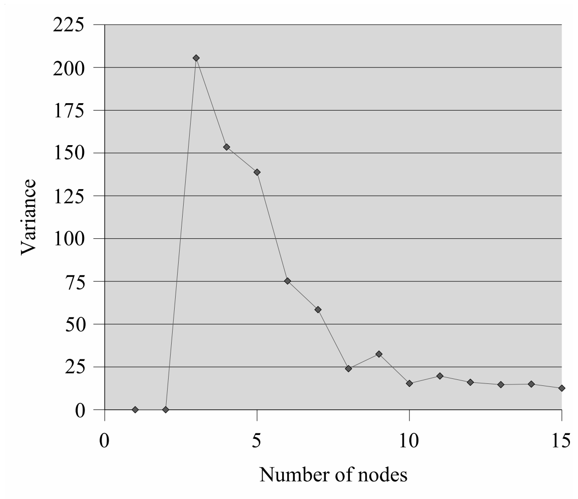

Figure 8. Variance of the number of agents in each node.

ups and recovery mechanism. In Figure 9, the ratio of the total amount of transferred agents with and without the backup and recovery mechanism are shown. Both amounts are measured in bytes between the time when the system is bootstrapped and the time when n nodes have completed to join the system. Here and after, the former amount is referred as $B(n)$ and the latter is $B'(n)$. The results indicate the overhead of the backup and recovery mechanism doesn't increase as the number of nodes increase.

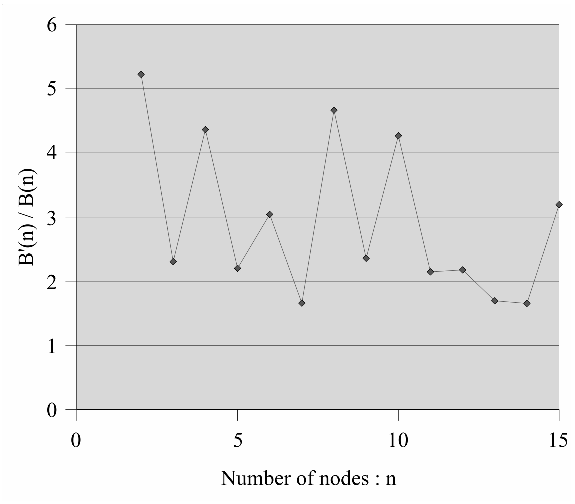

Figure 9. Ratio of $B'(n)$ to $B(n)$.

6 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.

We have developed a backup and recovery mechanism for our distributed e-Learning system. With this mechanism, when a node failure occurs, another node continues service using backups of the agents belonged to the failure node. Our algorithm always tolerates one node failure and tolerates more than one node failure under the condition that there are no failures occurring in both an original node and its backup node at the same time.

Experimental results show that our approach balances the number of agents including backups in each node. In other words, our approach doesn't lose the advantage of the distributed e-Learning system in scalability. Although the overhead of backups and recovery mechanism is independent of the number of nodes, it is significant. This overhead will be tackled in future work.

References

[1] Nishita, T., Kondo, K., Ohno, Y., Takai, Y., Miura, K. T., Dobashi, Y., Ishiuchi, S., Takahashi, T., Kimura, A. and Miyai, A.: Development of a Web Based Training system and Courseware for Advanced Computer Graphics Courses Enhanced by Interactive Java Applets, *Proceedings of International Conference on Geometry and Graphics*, Vol. 2, pp. 123–128 (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 (IJEL)*, 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] Kawamura, T., Motomura, S., Nakatani, R. and Sugahara, K.: Multi Agent-based Approach for Asynchronous Web-based Training System, *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems* (Hexmoor, H., ed.), pp. 537–542 (2005). Waltham, Massachusetts, USA.

[6] Kawamura, T., Motomura, S. and Sugahara, K.: Implementation of a Logic-based Multi Agent Framework on Java Environment, *Proceedings of International Conference on Integration of Knowledge Intensive Multi-Agent Systems* (Hexmoor, H., ed.), pp. 486–491 (2005). Waltham, Massachusetts, 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: napster, http://www.napster.com/.

[9] Gnutella: Welcome to Gnutella, http://welcome.to/gnutella/.

[10] Clarke, I., Sandberg, O., Wiley, B. and Hong, T. W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System, http://freenetproject.org/freenet.pdf (1999).

[11] Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, 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).