Management of Multimedia Data for Streaming on a Distributed e-Learning System

Tadafumi Hayakawa, Masayuki Higashino, Kenichi Takahashi, Takao Kawamura, and Kazunori Sugahara
Graduate School of Engineering, Tottori University
4-101, Koyama-Minami Tottori, Japan
{s042047, s032047, takahashi, kawamura, sugahara}@ike.tottori-u.ac.jp

***Abstract*—Nowadays, a lot of e-Learning systems are widely deployed in educational schools. Typical e-Learning systems are implemented as client-server model. In the client-server model, the number of clients affects on the load of the server. In order to reduce the load on the server, we developed a P2P-based distributed e-Learning system. The proposed system consists of a lot of mobile agents which manage study contents and some functions such as scoring, showing questions, and correct answers. When a learner requests content, a mobile agent who has its content comes to the learner's computer, and then he/she can start the study. Here, a mobile agent has to manage multimedia data, which may be a huge size of data. Thus, a mobile agent has to migrates to the learner's node with a huge size of data. Then, the learner cannot start the study until the mobile agent finished to migrate. In order to solve this problem, we divide multimedia data into fragments and prepare mobile agents which manages each fragments. Since each mobile agents become small, a learner can start the study soon without waiting for the migration of a mobile agent which has a huge size of multimedia data. We, however, have to search a mobile agent which manages their fragments. Therefore, a mobile agent which manages n-th fragments informs its location to the agent which manages $(n+1)$-th fragments, and vice versa. Since each agent knows the location of a mobile agent which manages next and previous fragment, a learner can play multimedia data smoothly without finding the location of mobile agent which manages holding next fragment. Experiment results show the effectiveness of our method.**

***Keywords*-e-Learning; Multimedia; P2P; Mobile Agent;**

I. Introduction

The evolution of the internet enables us to study some contents in anywhere and anytime, which are called as Web-Based Training system (WBT) and/or e-Learning system. Thus, learners can use their own time and schedule.

WBT is classified into two types, synchronous and asynchronous type. Synchronous type allows learners to access and study simultaneously on any time. Asynchronous type allows each learner to access on own time. In this paper, we aim at asynchronous type e-Learning system.

Typical e-Learning systems are based on client-server model. In the client-server model, since all contents and all functions are centrally managed on a server, the increase of the number of clients cause the load on. When the server crashes by some problems such as overloading, whole system stops, thus learners cannot study anymore.

Therefore, we proposed a distributed e-Learning system based on P2P model [1], [2]. Our system is implemented by Maglog [3] that is a Prolog-based agent framework. In our system, all the contents are managed by mobile agents (hereafter we refer to mobile agent as agent). Each agent has not only learning contents but also functions such as showing contents and answers, explanations, and scoring. These agents are distributed on learners' computer (hereafter we refer to such a computer as node).

In our system, one mobile agent may hold a huge size of data, such as movies and sounds. The size of the agent becomes huge, if the agent manages a huge size of multimedia data. Then, learners cannot start the study soon because they have to wait of the agent migration. In P2P network, popular method of managing multimedia data is dividing into fragments. Users can play multimedia data by searching and downloading these fragments. However, if users cannot download next fragment until the time to play it, multimedia data playing is stopped. Thus, to shorten the pausing time, users download important fragments preferentially by considering importance automatically [4].

To solve this problem, we divide multimedia data into fragments and prepares a lot of agent which manages their fragments. The size of each agent becomes small, learners do not need to wait the migration of an agent which has a huge size of multimedia data, thus, the learners can start the study soon. We, however, have to search a mobile agent which manages their fragments. Therefore, a mobile agent which manages n-th fragments informs its location to the agent which manages $(n+1)$-th fragments, and vice versa. Since each agent knows the location of a mobile agent which manages next and previous fragment, a leaner can play multimedia data smoothly without finding the location of mobile agent which manages holding next fragment.

II. P2P based e-Learning System

We developed P2P based e-Learning system. In our system, when a learner tries to study in our system, the computer of the learners has to be a part of the system as one node and provide its computing resource. When a computer joins to the system as a node, some agents migrate from other nodes to this node. Agents that migrate to this node are determined by Distributed Hash Table (hereafter we refer to Distributed Hash Table as DHT). Well-known DHT is Chord [5] and Content Addressable Network (hereafter we refer to Content Addressable Network as CAN) [6]. Our system equips CAN because of its simplicity. 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 managing the point P.

Our P2P network is structured with 2-dimensional coordinate space [0,1] × [0,1]. Fig.1 shows the situation when node D joined the system as the fourth node. Before node D joins, node A has to manage a half of all domain on DHT, but node B and C manages only a quarter of all domain. In this case, based on domain assigned to each nodes, three agents work on node A; one agent on node B; two agents on node C. When node D joins, it is mapped on a certain coordinate space according to a random number. In fig 1, node D is mapped in the domain node A manages. Therefore, node D takes on a half of the domain from node A, and takes on two agents. Thus, the location of each agent dynamically changes.

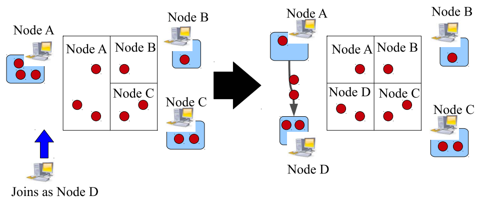

Figure 1. Contents management on DHT.

Our system consists of exercise agents (hereafter we refer to exercise agent as EA) and user agents (hereafter we refer to user agent as UA).

EAs manage learning contents and has some functions such as scoring and the explanation of the learning contents. UAs provide useful user interface to a learner, communicate with other type of agents and helps learners to study. Every these agents is mapped on DHT according to their keys created by hash function. The key of an EA is a hash value created from learning contents. The key of an UA is a hash value of learner's name. When a learner hopes to study some content, his/her UA sends a content request message to an EA which manages the content. The EA migrates to the learner's node.

Sometimes, learning contents need some graph, some figure, and some sound or video. However, if EA manage these multimedia data too, EA's size becomes huge, and learner cannot learn until downloading whole EA. Thus, multimedia data is managed separately as shown Fig. 2. Media agents (hereafter we refer to media agent as MA) manage multimedia data. If the content needs to refer multimedia data, the EA sends a multimedia data request message to a MA. When the MA receives the message, it migrates to the learner's node. Then, the learner can play the multimedia data. Therefore, even if the size of multimedia data becomes huge, the EA can provide learning contents in a constant time but MAs provide multimedia data afterward.

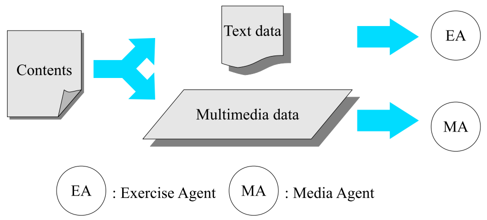

Figure 2. Separation of contents into text data and multimedia data.

III. Multimedia data Management

A learner cannot play multimedia data until the migration of the MA completes. Therefore, the migration of the MA to the learner's node has to finish before the learner needs to play the multimedia data. As a solution for this problem, we divide multimedia data into multiple fragments by time series. Each fragment is managed by each MA as shown in Fig.3.

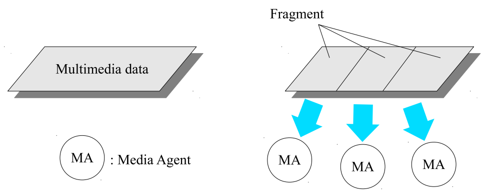

Figure 3. Fragmentation of multimedia data.

When a multimedia data is requested, these MAs migrate to learner's node one by one. When the first MA has migrates, the learner can play multimedia data soon without waiting for the download of whole multimedia data.

Each MA which manages each fragment is mapped on DHT based on its keys. Since a multimedia content is composed of some MAs which manages fragments, we hava to know the location of these MAs on DHT. Therefore, each MA has the key of a MA which manages next fragment. Then, each MA can find next MA according to its key. However, it needs a lot of message to find a node which manages next MA. Because our system uses 2-dimensional CAN, $O(\sqrt{n})$ messages are required to find next MA. Here, n is the total number of nodes. This may impede smooth playing of the multimedia data. To solve this issue, before the multimedia content is required from learners, each MA finds a node which manages next MA and records its location. Thus, every MA is linked in the order of time series of the multimedia data. Consequently, we can reduce messages to find the location of MAs except first MA's search.

When a node joins or leaves, a MA may migrate to other nodes. If a MA migrates, the link between MA becomes useless anymore. To keep the link, when a node joins or leaves, each MA records not only the location of next MA but also previous MA. When a MA migrates, the MA notifies its new location both to next and previous MA as shown in Fig. 4. Thus, the link between MAs is kept continuously even when a node joins or leaves.

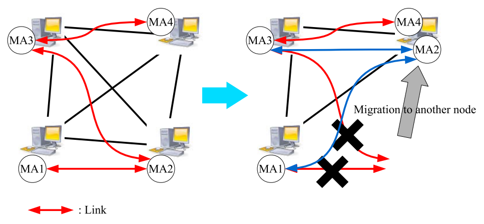

Figure 4. Updating location when MA migrates.

IV. Experiments

A. Smooth Play of Multimedia Data

We investigate whether multimedia data can play smoothly or not. In this experiment, we play multimedia data and compare total time of pausing between when each MA knows the location of next MA (*with link*), and does not know (*without link*). We use a 16.1 mega bytes video file which playing time is 21 seconds. We divide this file into multiple fragments by time series. The size of each fragment is 200, 225, 250, 275, or 300 kb. We use 100 Mbps LAN.

Fig.5 shows the result of total pausing time.

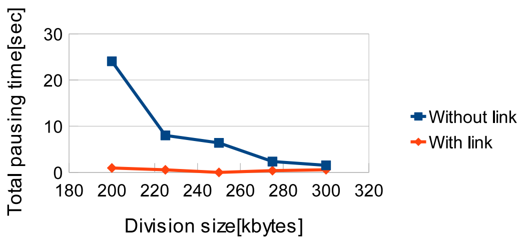

Figure 5. Total pausing time according to division size.

In *without link*, when division size is smaller, in other words, the playing time per one fragment is shorter, the total pausing time increases. For example, when division size is 200 kb, the total time of pausing is 25 seconds. This is because each MA has to find next MA on DHT. In *with link*, total pausing time does not depend on division size. Thus, multimedia data can be played smoothly.

B. The Number of Migration in Each Time Period

We showed how many MAs arrive to a requesting node in each time period. We use the same video file in IV-A, and network configuration.

Fig. 6 shows the result of that division size is 200 kb. In *with link* , a lot of MA arrives simultaneously, thus, the load on the node is concentrated. However, in *without link*, the migrations of MA are averaged in each time period.

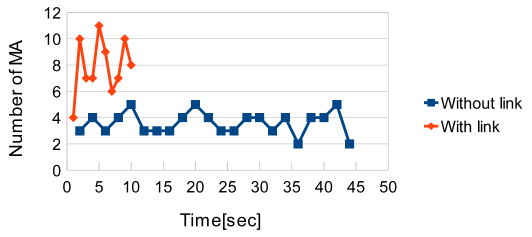

Figure 6. The number of MA migrations to a requesting node.

C. Timing When MA Migrates

If division size is large, multimedia data may not be smoothly played, because some MAs migrates to a requesting node simultaneously. In this experiment, we investigate duration of MA's migration to a requesting node, when each MAs manages 10 mega bytes fragments which is a part of 192.4 mega bytes of video file. Playing time of this video is 372 seconds. We investigate in four cases.

In first case, a MA sends a requesting message to next MA immediately. In second case, a MA sends a requesting message to next MA after the migration to a requesting node finishes. In third case, a MA sends a requesting message to next MA after it waits for 5 seconds from a requesting message from previous MA. In fourth case, a MA sends a requesting message to next MA after it waits for 10 seconds.

Fig. 7, 8, 9, and 10 show the results of these experiments. We can see from Fig. 7 that the first case spends a lot of time for MA's migration, for example, first MA starts the migration at time 0 and finish at time 2, the 15th MA starts the migration at time 14 and finish at time 24.

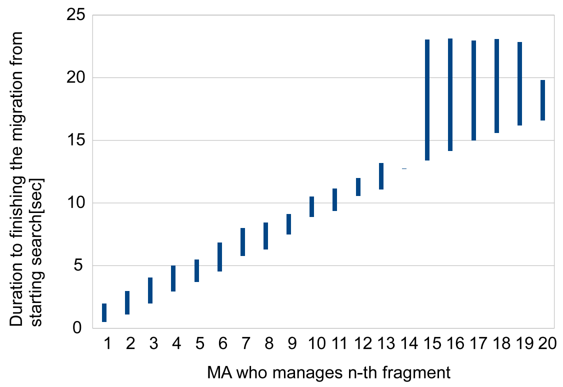

Figure 7. Duration when MA migrates (First case).

However, in other cases, almost MAs can migrate to a requesting node in shorter time.

D. JVM's Heap Size

We investigate the total time of pausing in the first case and the fourth case when Java Virtual Machine's heap

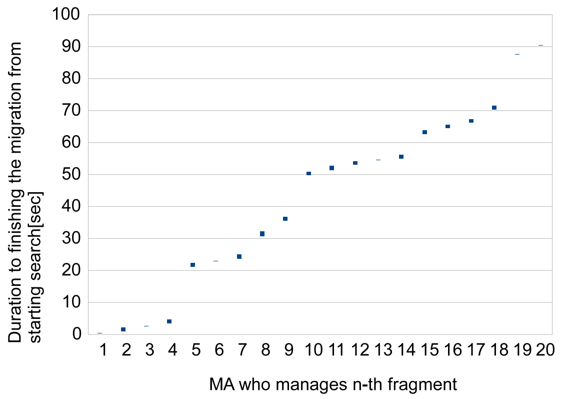

Figure 8. Duration when MA migrates (Second case).

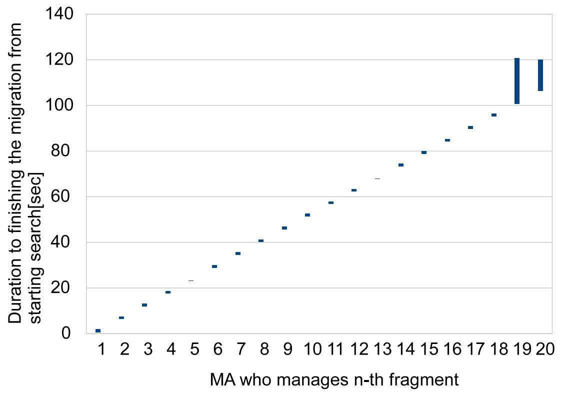

Figure 9. Duration when MA migrates (Third case).

memory size changes. Network configuration is the same as IV-A. Fig. 11 shows the result of this experiment.

In the first case, since a lot of MAs have to migrate simultaneously, heap memory becomes full soon. Therefore, a lot of MAs are swapped out, and it obstructs the smooth play of multimedia data. In the fourth case, this does not happen. Thus, it is necessary to control the migration of MAs when a lot of MAs try to migrate simultaneously.

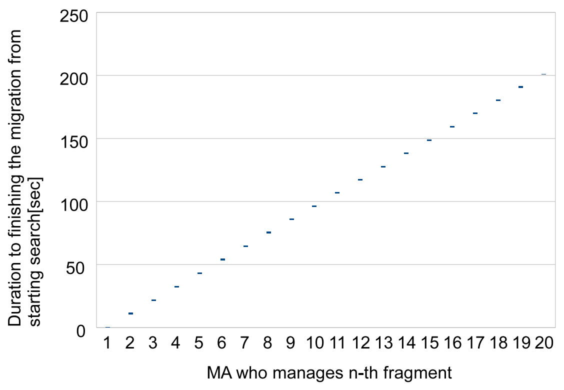

Figure 10. Duration when MA migrates (Fourth case).

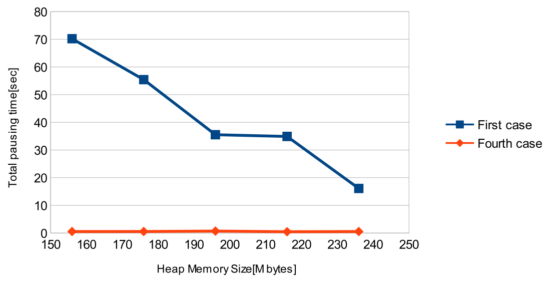

Figure 11. Total pausing time.

V. Conclusion

In this paper, we proposed and implemented a method to play multimedia data smoothly. In our system, multimedia data is divided into multiple fragments by time series, each media agent manages each fragment, and their media agents are linked bidirectional. If division size of multimedia data is huge, multimedia data cannot be played smoothly because a lot of agents have to migrate to a requesting node simultaneously. Therefore, we devised the timing of sending a requesting message to next MA. This method enabled smooth play of multimedia data.

References

[1] T. Kawamura and K. Sugahara, “A Mobile Agent-Based P2P e-Learning System,” *IPSJ Journal*, vol. 46, no. 1, pp. 222–225, 1 2005.

[2] S. Motomura, R. Nakatani, T. Kawamura, and K. Sugahara, “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.

[3] S. Motomura, T. Kawamura, and K. Sugahara, “Logic-Based Mobile Agent Framework with a Concept of “Field”,” *IPSJ Journal*, vol. 47, no. 4, pp. 1230–1238, 4 2006.

[4] S. Sakashita, T. Yoshihisa, T. Hara, and S. Nishio, “A Method to Reduce Interruption Time in P2P Streaming Environments,” *Journal of Information Processing*, vol. 52, pp. 1045–1054, 3 2011.

[5] “Chord,” http://pdos.csail.mit.edu/chord/.

[6] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content-addressable netwrok,” in *Proceedings of the ACM SIGCOMM 2001 Technical Conference*. ACM SIGOMM, 2001, pp. 161–172.