Management of Streaming Multimedia Content using Mobile Agent Technology on Pure P2P-based Distributed e-Learning System

Masayuki Higashino*, Tadafumi Hayakawa, Kenichi Takahashi, Takao Kawamura, and Kazunori Sugahara

Department of Information and Electronics,
Graduate School of Engineering,
Tottori University,
Tottori 680-8552, Japan
E-mail: s032047@ike.tottori-u.ac.jp
E-mail: s042047@ike.tottori-u.ac.jp
E-mail: takahashi@ike.tottori-u.ac.jp
E-mail: kawamura@ike.tottori-u.ac.jp
E-mail: sugahara@ike.tottori-u.ac.jp
*Corresponding author

Abstract: Nowadays, many e-Learning systems are widely deployed in educational schools. Typical e-Learning systems are implemented as the client-server model, but it requires an expensive central server for scalability. Thus, in order to reduce the load on the central server, we have been developing a pure P2P-based distributed e-Learning system. All learning contents are realized as mobile agents in this system, which works by interactions of mobile agents. However, it is not enough that data are simply distributed learning contents into nodes. Since the size of multimedia content is different, it makes imbalances of resource usage among nodes. Additionally, users cannot play smoothly multimedia contents. Therefore, we propose a method that divides a multimedia content into small fragments, keeps links of locations among these fragments, and caches these fragments in nodes.

Keywords: e-learning system; mobile agent; multi-agent; distributed system; P2P; multimedia; streaming.

Reference to this paper should be made as follows: Higashino, M., Hayakawa, T., Takahashi, K., Kawamura T., and Sugahara K. (xxxx) 'Management of Streaming Multimedia Content using Mobile Agent Technology on Pure P2P-based Distributed e-Learning System', *International Journal of Grid and Utility Computing*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Masayuki Higashino was born in 1983. He received his B.Eng. and M.Eng. degrees from Tottori University, Japan, in 2008 and 2010, respectively. He is currently a Ph.D. student in Tottori University. His research interests include mobile agent systems and distributed systems. He is a student member of IEEE, ACM, and IPSJ.

Tadafumi Hayakawa was born in 1986. He received his B.Eng. and M.Eng. degrees from Tottori University, Japan, in 2009 and 2012. His research interests include mobile-agent-based distributed e-Learning systems.

Kenichi Takahashi Received the B.Eng., M.Eng. and D.Eng. degree in Computer Science and Electrical Engineering, Kyushu University in 1999, 2001 and 2004, respectively. From 2004 to 2010, he worked at Institute of Systems & Information Technologies/KYUSHU as a researcher. Now he is an associate professor of Department of Information and Electronics at Tottori University. His research interests include security, ubiquitous computing and multi-agent system. He is a member of IEICE, IPSJ and IEEJ.

Takao Kawamura was born in 1965. He obtained his B.Eng., M.Eng. and Ph.D. degrees in Computer Engineering from Kobe University, Japan in 1988, 1990 and 2002, respectively. Since 1994 he had been in Tottori University as a research associate and has been in the same University as a professor in the Faculty of Engineering since 2009. His current research interests include mobile-agent systems and distributed systems. Prof. Kawamura is a member of IPSJ and IEICE.

Kazunori Sugahara received the B.Eng. degree from Yamanashi University in Japan in 1979 and M.Eng. degree from Tokyo Institute of Technology, Japan, in 1981. In 1989, he received the D.Eng. degree from Kobe University, Japan. From 1981 to 1994, he was staff of the Department of Electronic Engineering, Kobe City College of Technology. In 1994, he joined Tottori University as an associate professor of the Department of Electrical and Electronic Engineering and he is a professor of the Department of Information and Knowledge Engineering. His current interest lies in the fields of mobile-agent system applications, computer architectures and hardware realizations of 1D and multidimensional signal processing algorithms. Prof. Sugahara is a member of IEEE, IEICE and IPSJ.

1 Introduction

E-Learning systems, especially asynchronous Web-Based Training systems (WBT) (Driscoll, 2002; Uskov, 2010) are very popular. A WBT does not require interactions with instructors and allows a learner to study on his/her own time and schedule. The mainstream e-Learning system is based on a client-server model. Although the client-server model has an advantage of easy construction and maintenance, however, the client-server model requires expensive initial investments to execute management and to offer the contents by the server machine. Therefore, in order to reduce the load on the central server, we have proposed a pure P2P-based distributed e-Learning system. All learning contents are realized as mobile agents in this system, which works by interactions of mobile agents.

In the past, the Internet users usually communicate with each other by using texts or images mainly; but nowadays, they interact with each other by using multimedia. Thus, multimedia distribution services (e.g., iTunes Store (Apple Inc., 2013) and Netflix (Netflix, Inc., 2013).) that provide music and videos via the Internet are become popular. It is same even in the field of education. E-Learning systems using multimedia contents are also popular because multimedia contents increase efficiency of learning. By using multimedia contents, e-Learning systems enable to provide more understandable learning contents compared with texts and/or images.

However, it is not enough to simply distribute learning contents to nodes in a pure P2P-based distributed e-Learning system, because the size of a multimedia content is different. It makes imbalances of resource usage among nodes. Additionally, users cannot play smoothly multimedia contents because the load of providing multimedia contents concentrates on a node which manages popular multimedia contents. Therefore, in this paper, we describe basic features of our pure P2P based e-Learning system and propose a method of multimedia contents management. The method divides a multimedia content into small fragments to balance usage of nodes' storage, keeps links of locations among these fragments to reduce costs of look-up for learning contents, and caches these fragments in nodes.

The rest of this paper is organized as follows. Section 2 describes our pure P2P based distributed e-Learning system. Section 3 presents our method that manages multimedia contents. Section 4 describes the experimental results of the method. Section 5 discusses related work. Finally, Section 6 draws the conclusions.

2 Pure P2P-based Distributed e-Learning System

2.1 Design Concepts

We have been proposing a pure peer-to-peer-based distributed e-Learning system that is designed from gathered inexpensive computers. Our e-Learning system can distribute not only data but also processes into computers by mobile agent technologies.

Data in our e-Learning system consist of learning contents and personal information of users, such as account information, history of studies, answers, score, etc. Processes consist of providing of contents, displaying of contents, scoring of answers, login/logout, etc. In order to distribute data into nodes, our e-Learning system uses a DHT (Distributed Hash Table). However, a node that has popular learning contents gets concentrated requests of learning contents from many nodes. The node may not have enough performances to provide learning contents to many nodes since the system consists of inexpensive computers. In order to solve this problem, in our e-Learning system, a learning content is managed by a mobile agent that has ability to migrate to the requested nodes and to process own tasks. Thus, these processes are distributed into users' node. Because these mobile agents work only in a sandbox of user's node, they do not cause problems about a security on user's node.

2.2 Mobile Agents

Our e-Learning system is mainly constructed from mobile agents in order to realize scalability and understandability of the system. In our e-Learning system, there are following mobile agents (Figure 1).

- EA (Exercise Agent)
- CA (Category Agent)

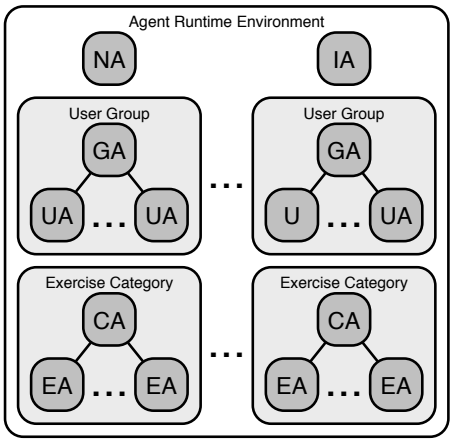


Figure 1 Composition of mobile agents on an ARE.

- UA (User Agent)
- GA (Group Agent)
- NA (Network Agent)
- IA (Interface Agent)

There are various methods for scoring users' answers of the learning contents. However, a general e-Learning system provides only several templates of the learning contents and methods of scoring. On the other hand, our e-Learning system has no limits of methods of scoring learning contents and can provide methods to make high-flexible learning contents because a learning content is managed by a mobile agent.

A mobile agent can migrate to users' node and execute optional processes at the users' node. Therefore, a mobile agent as a learning content can score users' answers appropriate to the learning contents. In our e-Learning system, such a mobile agent is named an **EA (Exercise Agent)**. An EA has a learning content. When an EA receives a message of a request of learning, the EA create a clonal EA of self; and a cloned EA migrates to learner's computer, provides a learning content, displays a learning content, scores its answers. After that learning is finished, the clonal EA migrates to node which the original EA exists and informs the original EA that the learning finished.

Generally, a user chooses sequentially learning contents and solves them. For example, a user chooses sequentially learning contents such as 1th, 2nd, 3rd, ..., and n-th. Therefore, in order to reduce a cost of look-up, managing learning contents by a group is more efficient than managing them individually. In our e-Learning system, EAs of a group are managed by **Category Agent (CA)**. A CA has a list of identifiers of EAs, and each identifier is a hash value of learning contents. Additionally, a CA has a category name which is used as a key of a DHT to look up a location of its CA. A location of a CA can be looked up with its category name from DHT, and a location of an EA can be looked up from a list of identifiers that the CA has. When churn (joining or leaving) of a DHT occurs, a CA and managed EAs by the CA migrates together to a location specified by DHT. Thus, if a user looks up learning content as an EA, the user can obtain effectively the EA by using a category name of a CA and an identifier of an EA.

In our e-Learning system, personal information of a user is stored into a mobile agent called **User Agent (UA)**. An UA has personal information and controls accesses of the information from users. Personal information contains a user name, a password, history of studies, answers, scores, etc. Because the size of history of studies, answers, and scores, becomes large according to the time, UAs are distributed into nodes in order to balance the usage of storages of nodes. When an UA receives a message of a request of login, the UA create a clonal UA of self; and the clonal EA migrates to learner's computer and authorizes the user to join the system by checking a user name and a password. After authorizing the user, the UA stays at the user's computer and has responsibility of updating of personal information (e.g., history of studies, answers, scores) until the logout. Thus, processing of login/logout and recording of scores is executed locally at the user's node and the load of the processes is distributed to user's node.

Generally, a teacher manages scores of students by group such as academic subjects, classes, or courses. Therefore, in order to reduce a cost of look-up of a DHT, managing UAs by a group is more efficient than managing individually them. In our e-Learning system, UAs of a group are managed by **Group Agent (GA)**. Here, relationships between GA and UAs are based on same idea of relationships between CA and EAs. A GA has a group name that is used as a key of a DHT to look up a location of its GA. A location of a GA can be looked up with its group name from DHT, and a location of an UA can be looked up with a user name that the UA has. When churn (joining or leaving) of a DHT occurs, a GA and managed UAs by the GA migrates together to a location specified by DHT. Thus, if a user looks up an UA, the user can obtain effectively the UA by using a group name of a GA and a user name of an UA.

As previously mentioned, the location of a CA and a GA is managed on an overlay network of a DHT as a pair of a name and a location. In our e-Learning system, this overlay network is provided by agents called **Network Agent (NA)**. One NA exists on one node and has responsibility of providing of an overlay network. A NA checks living of near nodes by that whether a child agent of the NA can migrate to the node. When churn (joining or leaving) of a DHT occurs, NAs send a message of a request of migration to EAs and GAs in order to re-distribute mobile agents and to balance usages of resources. Additionally, a NA provides API of a key-value interface of a DHT and delivers its messages.

Since humans cannot understand messages of mobile agents, mediators between humans and mobile agents are required. In our e-Learning system, there is **Interface Agent (IA)** in order to mediate between a user and mobile agents. An IA contains a GUI component based

on a web application, and a user can access a GUI of the IA through a web browser. When a user operates a GUI (e.g., a button, a text box, or etc.), the IA translates from these operations into messages that mobile agents can understand, interacts among other mobile agents, and returns the results to the user.

3 Management of Streaming Multimedia Data

3.1 Multimedia Content Distribution

If a learning content is a multimedia such as a video, a size of a multimedia will be reached MiB or GiB. Multimedia contents are really different in data size to text contents. Thus, if simply distributing a learning content as a unit of a file or a group, the inequality of resource usage (e.g. CPU time, memory usage, disk usage, network bandwidth, etc.) between nodes occurs. Additionally, a transfer of a text content is finished at short time. However, in the case of a video content, it takes a long time. We have to devise the management of multimedia contents.

Therefore, if a learning content contains multimedia data, we divide the multimedia data into small fragments. Here, we define a multimedia agent (MA) to manage these small fragments. An EA needs not manage multimedia data anymore; but manages only a description of a multimedia and references of locations to these MAs. Thus, the load of EAs becomes low, because only a description and references are required to be transfered. Additionally, the transfer of multimedia data is distributed on each MAs.

3.2 Distributed Multimedia Content Streaming

Each fragment is mapped on DHT based on its keys. Since a multimedia content is composed of some fragments, we have to know the location of these fragments on DHT. Therefore, each fragment has the key of a next fragment. Then, each fragment can find next fragment according to its key. However, it needs many messages to find a node which manages next fragment, because a DHT requires a cost of a look-up up to $O(n)$.

Here, n is the total number of nodes. This may impede smooth playing of the multimedia data. In order to solve this issue, before the multimedia content is required from learners, each fragment finds a node which manages next fragment and records its location. Thus, every fragment is linked in the order of time series of the multimedia data. Consequently, we can reduce messages to find the location of fragments except first fragment's search.

When a node joins or leaves (churn), fragments are transferred to other nodes. If a fragment is transferred, the link between fragments becomes useless anymore. To keep the link, when a node joins or leaves, each fragment records not only the location of a next fragment but also a previous fragment. When a fragment is transferred, the fragment notifies its new location both to a next and previous fragment. Thus, the link between fragments is kept continuously even when a node joins or leaves.

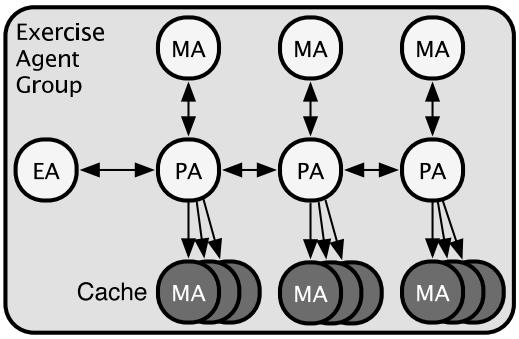


Figure 2 Distributed multimedia content caching by mobile agents.

3.3 Distributed Multimedia Content Caching

When a user learns using a multimedia content, fragments of the multimedia content temporarily gather in the learner's node. Therefore, this node can be used as a cache node. When a user learns using a multimedia content, the user's node stores the caches of the fragments into own node; and a node that has an original fragment, memorizes references of nodes storing cache. Thus, the node with original fragments has references of caches of fragments. Therefore, this node can balance a self-load like a DNS round robin (Figure 2).

In Figure 2, a PA is a pointer agent which manages MAs and caches of MAs. A MA is a multimedia agent which holds a fragment of multimedia data. These agents are managed by a EA. Each agent is mapped on an DHT network.

4 Experiments

4.1 Experimental Environment

Experiments have done with our e-Learning system that runs on computers (Intel Core i5 processor 3.2 GHz and 4 GiB RAM with JRE 1.6 and Debian GNU/Linux 5.0.5) connected through Ethernet of 100 BASE-T. Our e-Learning system uses a CAN (Ratnasamy et al., 2001) as the P2P network and is implemented in Java-based mobile agent framework called Maglog (Motomura et al., 2006). An ARE runs on a computer. A format of a video files for experiments is a FLV (Flash Video) format. The physical network layout is shown in Figure 3. In the experiments of Section 4.2 to 4.5, we have used 12 nodes. In the experiment of Section 4.6, we have increased the number of nodes from 1 to 9.

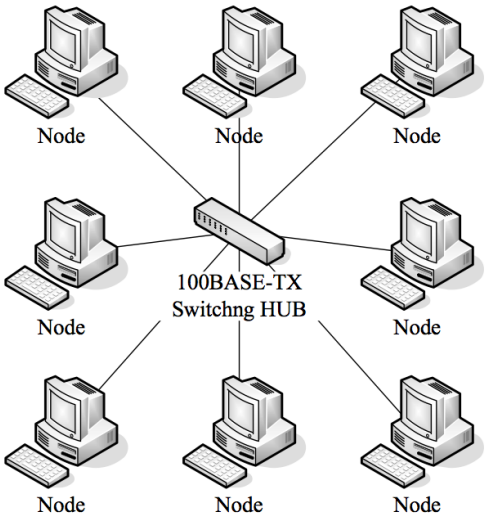


Figure 3 The physical network layout of the experimental environment.

4.2 Smooth Play of Multimedia Data

In order to show the effectiveness of our proposed method, we have evaluated whether multimedia data are played smoothly. Measuring a difference time length between a time length of multimedia data and an actual playing time length, we can evaluate whether multimedia data are played smoothly. This difference time length is a total pausing time length when the multimedia data are played; and close to zero means that the multimedia content is played smoothly. Thus, we have measured a total pausing time length in the case of *with link* and *without link*.

In this experiment, we have defined an about 16 MiB sized video file which has 21 seconds as multimedia data, and the video file has divided into fragments of 200, 225, 250, 275, or 300 KiB by time series.

Figure 4 shows the result of this experiment. In *without link*, a total pausing time length increases when the division size decreases. For example, when division size is 200 KiB, the total time of pausing is 25 seconds. This is because each MA has to lookup next MA on DHT. In contrary, in *with link*, a total pause time length does not depend on division size. Thus, the multimedia data can be played smoothly. This result shows that as the division size smaller, in other words, as more the multimedia data are divided and distributed in a DHT network, our proposed method is more effective.

4.3 Total Number of Arrival Agents in Each Elapsed Time

In order to examine more details of effectiveness in our proposed method, we have measured a total number of arrival agents in each elapsed time.

This experiment has done with the same configurations of the e-Learning system, computers, and video file in Section 4.2; but a division size has set only to 200 KiB.

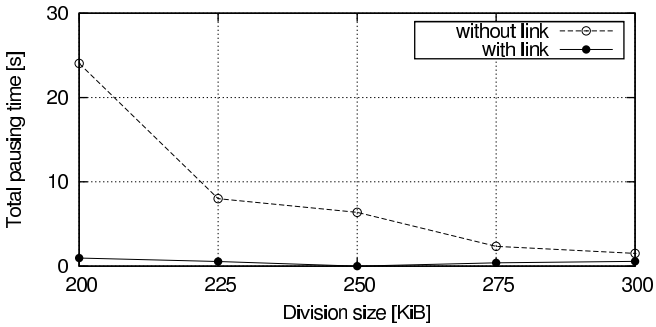


Figure 4 Total pausing time according to division size.

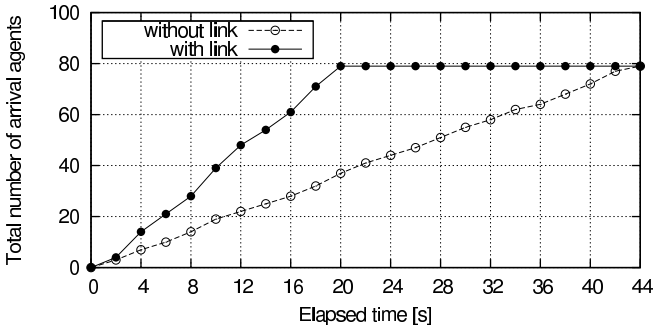


Figure 5 Number of arrival agents according to elapsed time.

Figure 5 shows this experimental result. In *without link*, many agents are migrating after exceeding the 21 seconds; therefore, the file data cannot be played smoothly. In contrary, in *with link*, many agents have arrived until 21 seconds, however, the load on the node will be concentrated. The influences about this situation are examined after this Section.

4.4 Limitation of Simultaneous Agent Migrations

If division size is large, the multimedia file may not be smoothly played because some agents migrate to a requesting node simultaneously. The simultaneous agent migrations will cause extreme resource consumption. Thus, when multimedia files play, the node has to limit the number of simultaneous agent-migrations.

We have investigated in four cases. In case 1, an agent sends a requesting message to next agent immediately. In case 2, an agent sends a requesting message to next agent after it waits for 5 seconds. In case 3, an agent sends a requesting message to next agent after it waits for 10 seconds. In case 4, an agent sends a requesting message to next agent after the migration to a requesting node finishes.

In this experiment, we have investigated duration of agent migrations to a requesting node. Each agent manages a fragment of 10 MiB that is a part of a video file of 192 MiB, and the video file has 372 seconds. All other experiment configurations are the same as Section 4.2.

Figure 6 shows the results of these experiments. We can see from Figure 6(a) that the case 1 spends much time for agent migrations. For example, a 1st agent starts

migration at time 0 and finishes at time 2, the 15th agent starts migration at time 14 and finishes at time 24. However, in other cases appeared in Figure 6(b) and Figure 6(c), almost agents can migrate to a requesting node in shorter time. Thus, the agent migration time decreases with increasing delay time. However, the duration of agent migrations has to less than the playing time of the multimedia file. For example, in the case 1 appears in Figure 6(d), agent migration time is shorter than other cases, but the video file played not smoothly.

4.5 *Influence of Video Smoothness on Memory Size*

We have investigated the total pausing time in the case 1 (no delay) and the case 3 (10 seconds of delay) in each maximum memory size.

In this experiment, we have adopted the maximum heap size of JVM (Oracle and/or its affiliates., 2013) to the maximum memory size of a node, and the heap size has set a value which from 156 to 236 with step 20 MiB. All other experiment configurations are the same as Section 4.2.

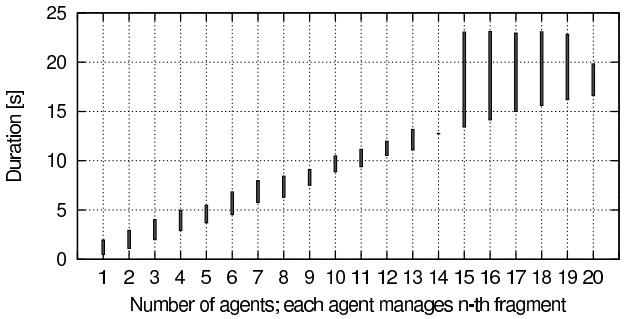
Figure 7 shows the result of this experiment. In the case 1 (no delay), since many agents have to migrate simultaneously, the heap usage becomes full soon. Therefore, many agents are swapped out, and it obstructs the smooth play of multimedia data. In contrary, in the case 4 (10 seconds of delay), this does not happen. Thus, it is necessary to control the migration of agents when a many agents try to migrate simultaneously. In the case 1, the maximum heap size smaller provides total pausing time longer. However, in the case 3, by delay to get fragments, total pausing time is reduced if maximum heap size is small.

4.6 *Effectiveness of Distributed Multimedia Data Cache*

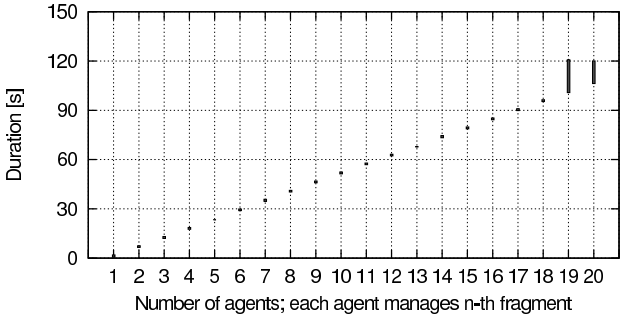
We have investigated the effectiveness of distributed multimedia data cache.

This experiment has done with the same configurations of the e-Learning system, computers in Section 4.2; but the video file size is 5.6 MiB and its file is divided into three fragments.

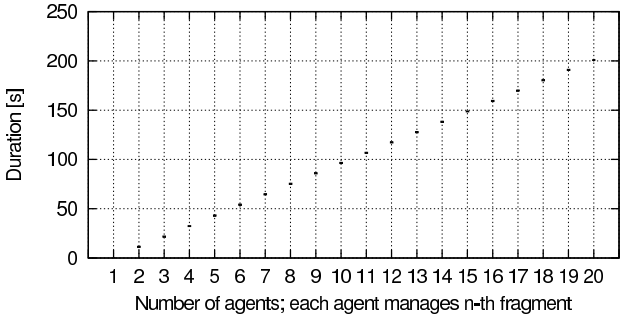
Figure 8 shows the result of this experiment. When the number of simultaneous requests to get a fragment from distributed nodes is 1 or 2, the response time of *with cache* is longer than *without cache*. Because, in spite of the requested node has no large load, the requested node routes these requests to other nodes which have cache of fragment of video file. However, when the number is greater than or equal to 3, the response improving. Thus, greater the number of requests, the cache is more effective.



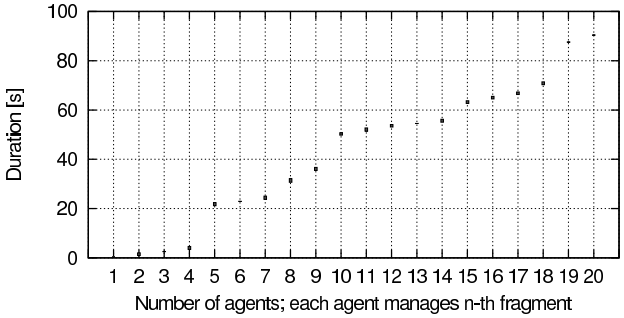
(a) Case 1: Serial requests each has **no** delay; and parallel agent migrations occur.



(b) Case 2: Serial requests each has **5** seconds of delay; and parallel agent migrations occur.



(c) Case 3: Serial requests each has **10** seconds of delay; and parallel agent migrations occur.



(d) Case 4: Serial requests and agent migrations with **no** delay.

Figure 6 Duration from the finish time of a migration of an agent to the start time of playing of a multimedia content.

5 Related Work

Several researches have proposed P2P-based e-Learning systems. EDUTELLA (Nejdl et al., 2002) is a one of the earliest P2P-based e-Learning system. This is based on JXTA (Verstrynge, 2010) that is a P2P application framework. There are JXTA-based e-Learning systems

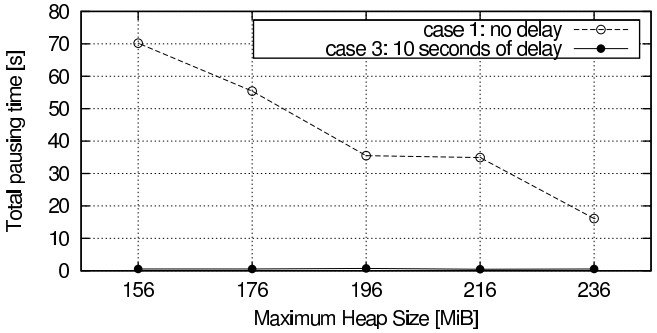


Figure 7 Effect of maximum heap size on total pausing time.

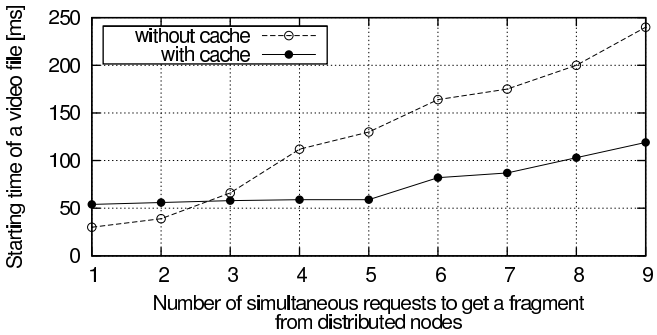


Figure 8 Comparison of response time between *with cache* and *without cache*.

such as PLANT (Li et al., 2007) and HYDRA (Zualkernan, 2005). In addition, several researches have proposed a new overlay network for a P2P-based e-Learning system, such as PROSA (Carchiolo et al., 2008) and PeerLearning (Wang et al., 2010), to support efficient method to look up learning contents. However, these systems are different from our proposed system in that multimedia contents are divided into fragments and managed by DHT.

(Tagashira et al., 2006) has proposed an index caching mechanism for CAN. On the other hand, our proposed system caches not only indexes but also data of the learning contents. (Huang et al., 2006) is proposed an agent-based collaborative virtual environment architecture using grid technologies, however, this is not based on a pure P2P model. On the other hand, our proposed system is based on a pure P2P model.

In Nearcast (Tu et al., 2008), a locality-aware P2P live streaming method has proposed. However, in a live streaming, a source node will be a single point of failure. If the source node is forced outage, then the streaming will stop. On the other hand, in our proposed system, a multimedia content is preliminarily divided and is preliminarily distributed into many nodes before all nodes are notified of the multimedia content existence. Therefore, the multimedia content streaming will not be forced the outage by economic or social reasons. This characteristic is similar to pure P2P-based file sharing systems, such as Winny (Kaneko, 2005), Gnutella (Kirk, 2013), etc., but these systems do not expect to the multimedia streaming and cannot stream multimedia.

6 Conclusion

In this paper, a method to play multimedia data smoothly on a pure P2P-based distributed e-Learning system is proposed. In our system, multimedia data are 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 are huge, multimedia data cannot be played smoothly because many agents have to migrate to a requesting node simultaneously. Therefore, we devised the timing of sending a requesting message to next agent. This method enables the smooth play of multimedia data on P2P-based distributed e-Learning system.

References

Apple Inc. (2013). iTunes Store. available at `http://www.apple.com/itunes/`.

Carchiolo, V., Longheu, A., Mangioni, G., and Nicosia, V. (2008). Adaptive e-learning: An architecture based on PROSA p2p network. In *Proceedings of the 21st International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: New Frontiers in Applied Artificial Intelligence*, pages 777–786.

Driscoll, M. (2002). *Web-Based Training: Creating e-Learning Experiences*. John Wiley & Sons, New York, USA, second edition.

Huang, C., Xu, F., Xu, X., and Zheng, X. (2006). Towards an agent-based robust collaborative virtual environment for e-learning in the service grid. In *Proceedings of the 9th Pacific Rim International Conference on Agent Computing and Multi-Agent Systems*, pages 702–707.

Kaneko, I. (2005). *The Technology of Winny*. ASCII MEDIA WORKS Inc.

Kirk, P. (2013). Gnutella - a protocol for a revolution. available at `http://rfc-gnutella.sourceforge.net`.

Li, M., Zhu, H., and Zhu, Y. (2007). Plant: a distributed architecture for personalized e-learning. In *Proceedings of the 6th International Conference on Advances in Web Based Learning*, pages 20–30.

Motomura, S., Kawamura, T., and Sugahara, K. (2006). Logic-based mobile agent framework with a concept of "field". *IPSJ Journal*, 47(4):1230–1238.

Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., and Risch, T. (2002). EDUTELLA: a p2p networking infrastructure based on RDF. In *Proceedings of the 11th International Conference on World Wide Web*, pages 604–615.

Netflix, Inc. (2013). Netflix. available at `http://www.netflix.com/`.

Oracle and/or its affiliates. (2013). java - the Java application launcher. available at `http://docs.oracle.com/javase/6/docs/technotes/tools/solaris/java.html`.

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

Tagashira, S., Shirakawa, S., and Fujita, S. (2006). Proxy-based index caching for content-addressable networks. *IEICE Transactions on Information and Systems*, E89-D(2):555–562.

Tu, X., Jin, H., Liao, X., and Cao, J. (2008). Nearcast: A locality-aware p2p live streaming approach for distance education. *ACM Transactions on Internet Technology (TOIT)*, 8(2):7:1–7:23.

Uskov, V., editor (2010). *Web-based Education*. ACTA Press.

Verstrynge, J. (2010). *Practical JXTA II*. Lulu Enterprises Inc.

Wang, G., Yuan, Y., Sun, Y., Xin, J., and Zhang, Y. (2010). Peerlearning: A content-based e-learning material sharing system based on p2p network. *World Wide Web*, 13(3):275–305.

Zualkernan, I. A. (2005). Hydra: A light-weight, scorm-based p2p e-learning architecture. In *Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies*, pages 484–486.