Human-centric Computing and Information Sciences
a SpringerOpen Journal

RESEARCH **Open Access**

Dynamic content synchronization between learning management systems over limited bandwidth network

Royyana M Ijtihadie[1,2*], Bekti C Hidayanto[2], Achmad Affandi[2], Yoshifumi Chisaki[1] and Tsuyoshi Usagawa[1]

*Correspondence:
royyana@hicc.cs.kumamoto-u.ac.jp
[1]Kumamoto University, Kumamoto, Japan
[2]Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Abstract
Well-designed instructional material is equally important for successful e-Learning implementation. Teachers and instructors play a major role in terms of designing and building learning content. In one respect, it requires costs in terms of effort, time and experience. In other respects, a good learning content is likely a result of recurring revisions as a result of teaching experience as well as evaluating student activities. In the case of higher educational institutions (HEI) in developing countries (such as Indonesia), resource sharing in many aspects is highly recommended effort against high cost and redundant works, e-Learning is no exception. Sharing and re-using e-Learning content on particular subject between Learning Management Systems (LMS) can be one of the methods. In addition, collaborative teaching may cause a content develops gradually while conducting content sharing. Thus, the capability of synchronizing the content between LMS is necessary. On the other hand, typical e-Learning implementation might not be appropriate due to the concerns of network infrastructure in developing countries. In some areas, the network has less bandwidth and even frequent disconnections. This paper introduces a novel method of sharing e-Learning content between distributed Learning Management Systems by using dynamic content synchronization. This method also suites the need of course sharing which supports collaborative teaching activity. Moreover, this approach is designed to address the needs of content sharing in areas with network infrastructure limitation in terms of bandwidth and availability.

Keywords: Dynamic content synchronization, Distributed learning management systems, Course sharing, Unidirectional content synchronization, Limited bandwidth, Developing countries

Introduction

As Information Technology (IT) spreads very rapidly, its usage becomes less expensive and more affordable whereas power and storage capacity improves greatly. The use of IT to support every aspect of life has became commonplace, as well as to support learning activities in educational institutions such as in school or university with e-Learning.

Well-designed instructional material is equally important for successful e-Learning implementation. Teachers and instructors play a major role in terms of designing and building learning content. In one respect, it requires costs in terms of effort, time and experience. In other respects, a good learning content is likely a result of recurring revisions as a result of teaching experience as well as evaluating student activities. In the case of higher educational institutes (HEI) of developing countries such as Indonesia, in which the resource gap is still prominent, such concerns need to be addressed. Several younger HEIs may not be able to afford to hire experienced professors to teach certain subjects as well as to create learning content on Learning Management System (LMS). Accordingly, sharing the learning content among individual university's/HEI's LMS over the network can be an appropriate solution against performing redundant effort.

Furthermore, a collaborative teaching may take place and may lead to active evaluation upon particular content on LMS at the origin/provider side. As a result, the content may dynamically change at times within a short period of time. On another LMS, while the course is in progress and conducting student activities, the existing shared content might need to be updated with the recent changes from the origin. Such update process should refrain from affecting current student activities. Despite simple practical methods are available, for example, using existing Moodle's backup-and-restore tool or database's dump-and-restore utility, both of which do not consider existing course related student activities information while performing the restoring process. As a result, all information related with student activities will be erased which could cause a harmful situation.

Some parts of the world, such as developing countries, are still suffering from high prices and unstable Internet bandwidth as well as connectivity problems which makes the effort of e-Learning useless due to slow access to the LMS [1]. Aside from the resource gap, certain developing countries such as Indonesia are suffering from limitation of network infrastructure in terms of bandwidth and stability in which we referred to as limited bandwidth network.

In addition , there is still low ICT penetration in broader areas and insignificant deployment of advanced network technology for data communication, particularly in rural and isolated areas. Some areas may still be enjoying older technology such as Plain Old Telephony System (POTS) in which the connection speed is relatively slow. Moreover, wide variation of physical infrastructure quality may cause frequent disconnection or disruption which is not good for data communication as well as for sharing learning content among LMSs.

Accordingly, sharing the content by means of providing accessible single LMS to other HEI's students might inappropriate unless the network is stable and has enough bandwidth to pass simultaneous requests. In addition, student's registration information needs to be stored in the LMS side, which typical HEIs might be reluctant to do.

This paper discusses a novel approach of sharing learning content among distributed Learning Management Systems over limited bandwidth network. In addition to content sharing, this approach performs dynamic synchronization between the shared content to provide support for collaborative teaching activity on the provider side. The rest of this article is organized as follows. Section "Unidirectional content synchronization" describes content synchronization concept. Section "Implementation" outlines the overall implementation and structure of our system. Section "Experiments and results" presents the example of the working system and some experimental results. Section "Contributions" presents our contributions. Section "Related works" gives a short survey of related works, and section "Conclusions and future works" gives conclusions and future work.

Unidirectional content synchronization

This section describes the concept behind unidirectional content synchronization which comprises of background and illustration upon how the course sharing activities taking place. The following definitions are used for the rest of this paper. Let Master LMS refers to Learning Management Systems (LMS) which shares its content (in which its scope is limited to a particular course within LMS) while Slave LMS refers to LMS which employs the content obtained from Master LMS. The term unidirectional is used to depict the direction of content sharing which is carried out in a single direction (Master LMS to Slave LMS).

The original content is expected to be designed by certain HEIs that has a group of experienced resources on a particular subject which can then be reflected to e-Learning content. Assuming to address the resource gap for developing a well-designed learning content in the context of e-Learning, sharing the content unidirectionally is preferred and more appropriate while the user side of the content will only be conducting student activities and the teaching. The feedback that might come from the user side is supposed to be directed to the course author at origin side. Bi-directional activity might be inadequate to be conducted in particular network situation in which down-link bandwidth is much bigger than its up-load bandwidth.

The synchronization process is basically a process to distribute changes of particular content on Master LMS to Slave LMS by means of network infrastructure with the demand from Slave LMS. To accommodate continuous evaluation as well as collaborative teaching on a particular content, Master LMS allows it to be modified gradually along the way. For example, a particular content might be conducting only 2 introduction units at the beginning of the semester and several units later on. In order to follow the change of the relevant content, the outdated content at the Slave LMS needs to perform synchronization process with the one at Master LMS while keeping the student activity record unaffected.

For the case of unidirectional synchronization, after the content has been dis-tributed and shared among LMSs, only the origin side of the content (Master LMS) is allowed to make modification while the content at the Slave LMS is not, thus to avoid content inconsistency and confusion. Also the content of Master LMS is copyrighted as the original.

Figure 1 shows the illustration of content synchronization among Learning Management Systems at different schools. In this system, any particular school's LMS can act either as a Master or Slave. School A shares its contents with School B while at the same time employing contents from School C.

Figure 2 shows the illustration of synchronized content allocation among LMSs. Course C1 from School C is being shared to School A while Course A1 and A3 are shared to School B. In LMS's point of view, the synchronized content would be accounted as regular content. Thereby, while particular LMS can have a flexible role, it also allows the synchronized content to exist together with the already existing content.

Implementation

Platform and system architecture

In essence, this system comprises two entities which communicates in order to maintain the consistency of the content at both sides. The system employs a hybrid online/offline

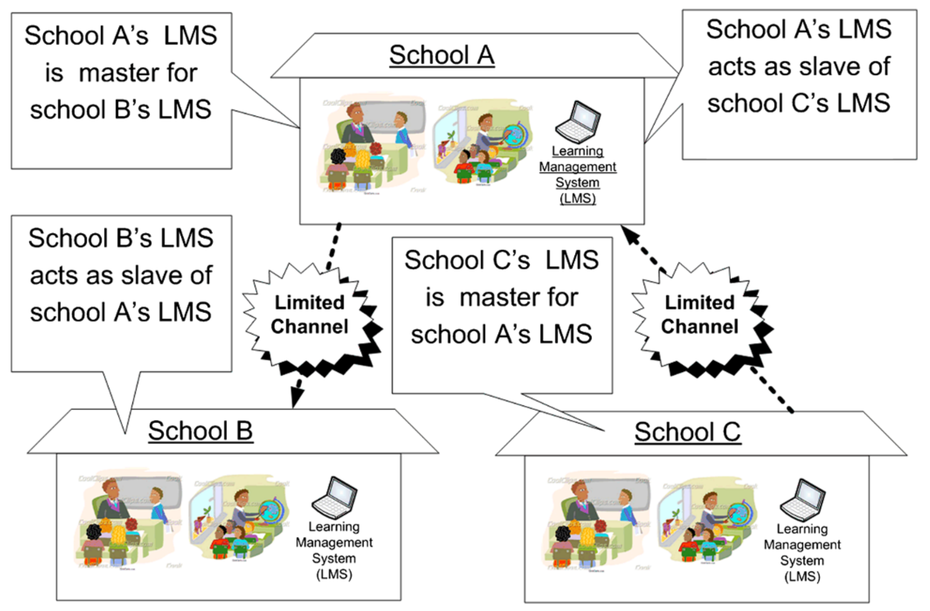

School A's LMS is master for school B's LMS
School A
School A's LMS acts as slave of school C's LMS
Learning Management System (LMS)
School B's LMS acts as slave of school A's LMS
Limited Channel
School C's LMS is master for school A's LMS
Limited Channel
School B
School C
Learning Management System (LMS)
Learning Management System (LMS)

Figure 1 Concept of content synchronization among the schools.

system [2,3] which puts the remote content in local network while keeping both content similar with respect to the available bandwidth [1]. Content synchronization and differential updates delivery are performed in a separate subsystem from current LMS. At Master LMS, a subsystem called Master subsystem is responsible for determining the differential updates to be delivered to Slave LMS. This Master subsystem has a course packer to transform involved Moodle internal tables into synchronization tables. The general architecture and its relationship between components are shown in Figure 3.

The Master synchronizer and Slave synchronizer are responsible for ensuring both synchronization tables on both sides are consistent. While the content at Master LMS altered, the content at Slave LMS will become outdated making the synchronization tables between them inconsistent. During the synchronization process, the Slave synchronizer will discover the inconsistent part and will make a request to the Master

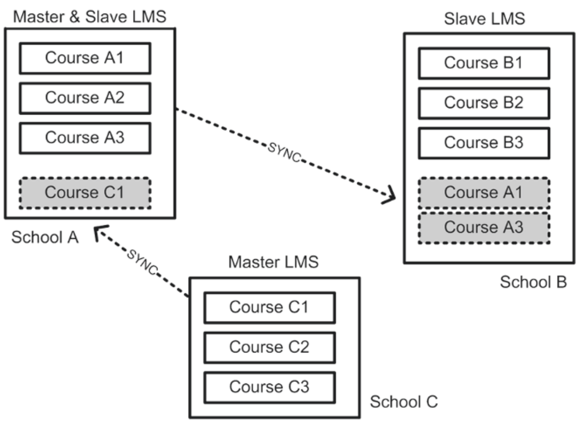


Figure 2 Course allocation among three LMSs.

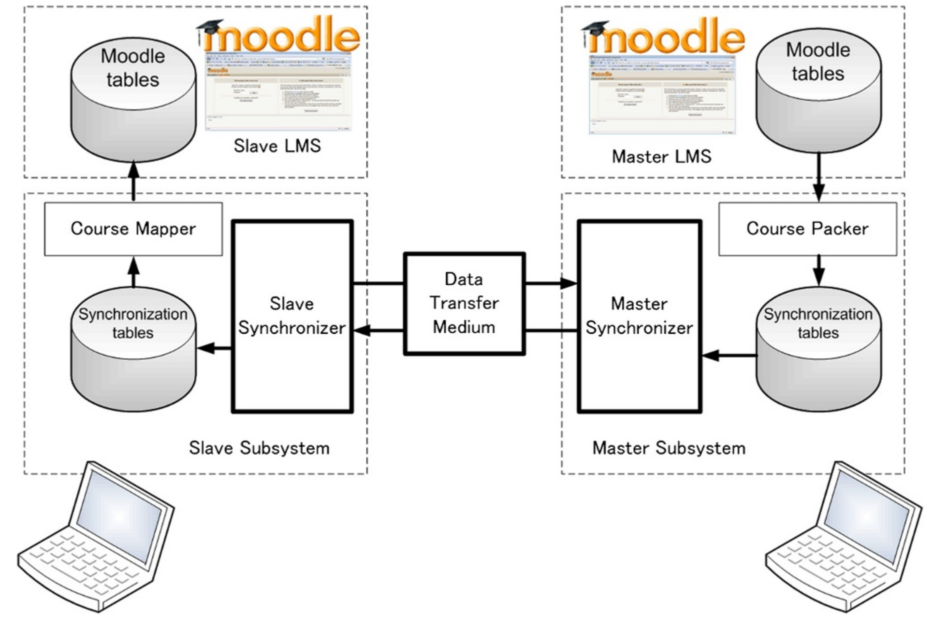


Figure 3 System architecture and components.

synchronizer to obtain the missing part. After such parts are obtained, the synchronization table at Slave LMS is said to be consistent and is ready to be converted back to Moodle table's format.

This system is using Moodle [4], a Learning Management System which is considered to be one of the most popular web-based LMSs and designed based on social construction pedagogy [5], aiming to help teachers to create effective online learning communities [6].

In order to be operational, both sides (Master LMS and Slave LMS) need to run the same version of Moodle (1.9). In addition, this system is running as an independent module running along with existing Moodle. LMSs that are assigned as a Master LMS and Slave LMS need to run applications called Master console and Slave console, respectively. Both applications need PHP and MySQL database to be running which are also the requirement of Moodle.

The Slave synchronizer requests the updates by communicating with the Master synchronizer by means of well-known Hyper Text Transfer Protocol (HTTP). In this case, pull based communication is performed whereas Slave LMS is the initiator of every synchronization process. Since HTTP is based on TCP/IP, it is possible to use this mode in conjunction with with common wide-area-network technology using IP network such as either satellite network, PSTN dial-up networking, or Mobile GPRS.

A backup system is not provided within this system. Regular backup operation on database is supposed to be provided by the administrator separately. The advantage is that this system stores all necessary information such as synchronization tables and multimedia objects within the same database as particular Moodle database, thereby making it easier for the administrator to focus upon database backup instead of additional file system backup. In the case that Master LMS crashed, the Slave will consider as a new course has been in place. In order to avoid such condition, backup procedure should be taking care of synchronization tables beside of the current Moodle database.

Master and slave communication

As described, content modification at the Master LMS is expected to occur as a result of development process of course material and is assumed to be carried out occasionally within a short period of time. With respect to limited bandwidth channel, efficient use of the communication link needs to be taken into account. Instead of distributing the entire content, sending only the necessary part is preferred. Common differential delivery technique and usage of hashes will be used throughout the synchronization process.

Differential delivery is employed to distribute the modified part by finding the differential between the previous version of content with the new version. In addition, differential delivery has been commonly used and known in computer systems for dealing with transfer over limited resource [7] including in software version control system such as git [8] and concurrent version system [9].

Hash table has been widely used in computer science as well, mainly for quick search of things in each of different value, and is able to identify certain location [10,11]. Another popular use of hash is message digest or fingerprint such as MD5 [12] to ensure integrity in a large amount of data (such as files or records) by either identifying or verifying using a small amount of information. Hash is also used in several major applications for identifying the changed part within large datasets such as remote file synchronization tool [13,14] and squid proxy server [15].

Figure 4 shows the communication flow between Master synchronizer and Slave synchronizer during synchronization process. Internal processes are drawn in dotted pattern boxes, which are *version matching*, *determining updates* and *applying updates*. The communication between Master synchronizer and Slave synchronizer will be carried out entirely using HTTP. The Slave side initiates by sending (*CID*, *version*, *hashes*) tuple to Master synchronizer designated URL (Uniform Resource Locator) where *CID* is mapped course ID, *version* is the version of current synchronization table, and *hashes* contains all the hashes from each record of synchronization table.

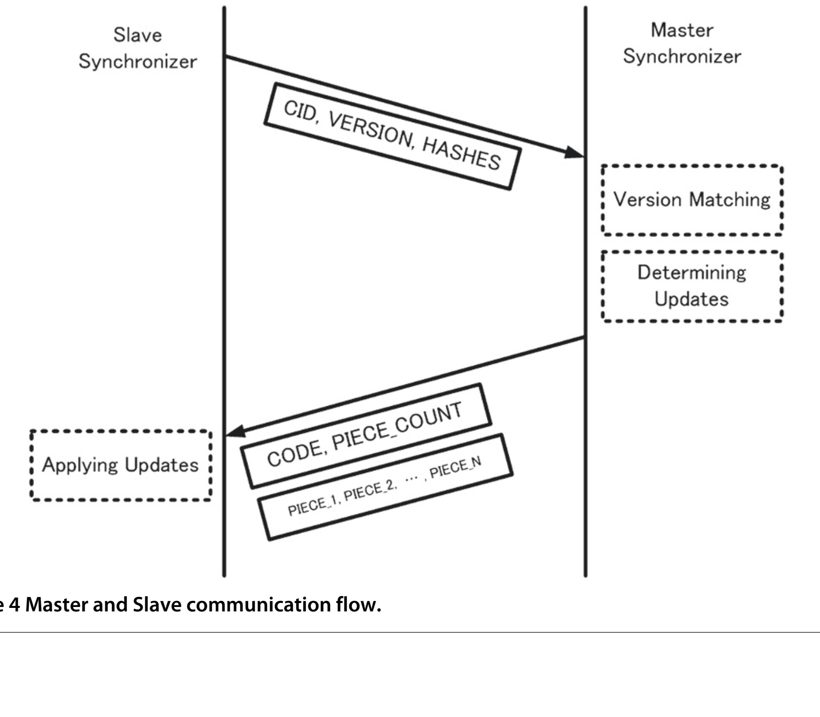


Figure 4 Master and Slave communication flow.

The Master synchronizer performs *version matching* and then *determining updates* while serving the request. Resultant updates will then be replied back to requesting Slave in the form of (*code*, *piece_count*) and followed by the series of updated information in the form of *piece_1* until *piece_n*. The *code* has two possible values to reflect either "no changes" are made or "detected" changes have been made. The number of the updates to be transferred is reflected in *piece_count*. *Applying updates* process will be performed immediately if the successful receiving process is done.

If disrupted information exchange (such as failed connection or disconnected transfer) occurs between Master synchronizer and Slave synchronizer, the system will drop the synchronization giving notification to the Slave console thus indicating inappropriate time to synchronize the particular course. The operator of the Slave console might try to do it later on.

Synchronization process

Differential operation is employed to obtain differential updates between the previous content and the current content. During synchronization process, content will be handled by both Master subsystem and Slave subsystem. Particular content is transformed into synchronization table which (for the rest of this paper, will be referred to as *synctable*). *Synctable* comprises number of records depending on the content size with every records containing (*ID*, *content*, *hash*) tuple information. *ID* is record identifier, *content* contains transformed records from corresponding Moodle tables, and finally *hash* contains MD5 hashes of *content*.

Suppose the Slave subsystem has a copy of outdated *synctable*(*ts*) and Master subsystem has the current version of *synctable*(*tm*) in which the abstract source code is shown below. In order to detect inconsistency between them, a set of hashes and a set of IDs in each *synctable* record will be used to compare *ts* and *tm* instead of comparing the whole records they have.

```
r_chgs = [ ]
r_tsi = tsi
r_tsh = tsh
l_diff = diff(tmi,r_tsi)
foreach id in l_diff
    r_chgs = r_chgs + ( id,append,tm[id] )
    r_tsh[id] = tmh[id]
end for

r_diff = diff(r_tsi,tmi)

foreach id in r_diff
    r_chgs = r_chgs + (id,delete,)
    delete r_tsh[id]
end for

if (r_tsh not equal tmh) then
    foreach id in tmh
          if (tmh equal r_tsh) then
                 continue
          end if
          r_chgs = r_chgs + (id,update,tm[id])
    end for
end if
```

At Master subsystem, let *tmi* be a set of IDs belonging to *tm* and *tmh* be a set of hashes which correspond to *tmi*. Similarly, at Slave subsystem, let *tsi* be a set of IDs belonging to *ts* and *tsh* be a set of hashes which corresponds to *tsi*.

In addition, *vm* and *vs* will define the version information of *tm* and *ts* respectively. Any modifications taking place at Master subsystem should change the content of *vm, tm, tmh,* and *tmi*.

Version matching

Version matching is carried out using MD5 hash algorithm which is meant to detect whether modifications at the other end are already in place. By exchanging the version information, Slave synchronizer only needs to send several bytes of data to represent the whole of the records. In the beginning of synchronization process, Slave synchronizer will send *vs* to Master synchronizer in order to check the version of latest *vm*. If these are different, Slave synchronizer will learn that modifications have taken place and then proceed to the next step. Otherwise, the process will stop since both contents are similar. Hence, further operation is not necessary.

Determining updates

Modifications at Master subsystem have several possibilities: *addition*, *deletion*, or *modification* of records within *tm*. To obtain such a list of modifications, determining differential content between *tm* and *ts* is carried out during synchronization. The Slave synchronizer is required to send both *tsi* and *tsh* to the Master synchronizer as a representation of *ts*. Before continuing, please note that the changes will be stored in a list named *r_chgs*.

At Slave subsystem, *r_tsi* and *r_tsh* are used for representing remote *tsi* and remote *tsh* respectively. The first step is to determine which *r_tsi* exist at *tm* but not at *ts*. This can be obtained by using relative complement of *r_tsi* in *tmi* denoted *diff(tmi,r_tsi*) which later will be stored in *l_diff*, appended into *r_chgs*list and marked as *'append'*.

The subsequent step is actually the reverse of the former step, determining *r_diff* containing *ts* which does not exists in *tm* (relative complement of *tmi* in *r_tsi*). In order to make *tm* and *ts* consistent, such *r_diff* values are supposed to be removed. Accordingly, those values will be added in *r_chgs* list and marked as *'delete'*.

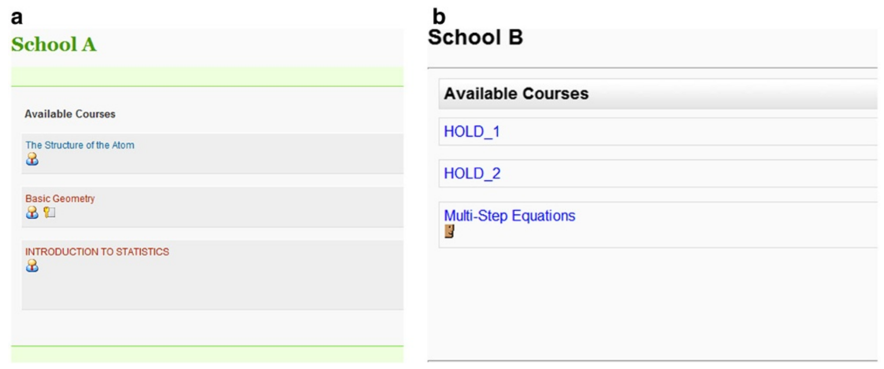
The last step is to check consistency between *tm* and *ts* by matching each of the *tmh* and *r_tsh* values. Should differences be found, the corresponding records and IDs will be appended to *r_chgs* and marked *'update'*.

Applying the received updates

Finally, the final content of *r_chgs* will be replied back to the Slave synchronizer as a response of sending the *tsh* and *tsi* beforehand. When the process is completed, the updates will be applied to *synctable* and later convert back to Moodle tables. In further synchronization process, when *tm* has not been altered, *tm* and *ts* will have similar content identified by *vm* and *vs*.

Experiments and results

This section describes the conducted experiments and their results. First of all, the running example will be described, followed by experiment of content sharing between universities in Japan and Indonesia using the Internet as well as an experimental satellite.


a
School A

Available Courses

The Structure of the Atom

Basic Geometry

INTRODUCTION TO STATISTICS

b
School B

Available Courses

HOLD_1

HOLD_2

Multi-Step Equations

Figure 5 Snapshot of school A and school B courses list (initial). (**a**) Course list at school A (Master LMS). (**b**) Course list at school B (Slave LMS).

Content synchronization running example

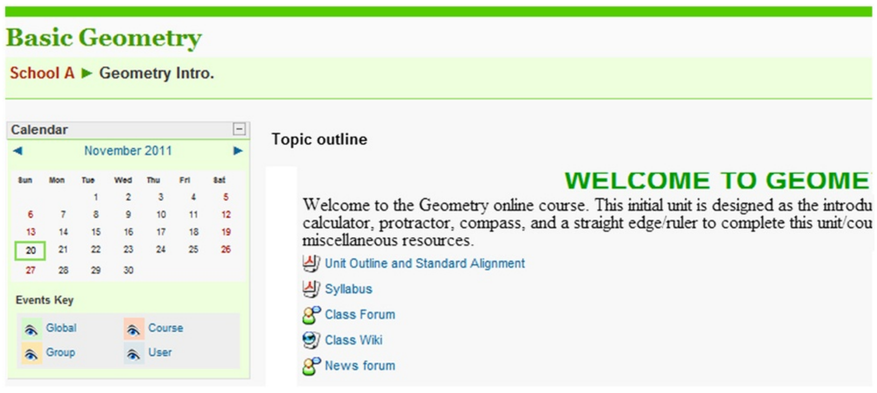
This subsection presents a running example to show the functionality of the system. Supposing School A shares some of their content (in the form of courses) to School B hence School A has a role as the Master LMS while school B is the Slave LMS. School B has several existing courses yet also desires to utilize several of School A's courses. All the course content in this example are obtained from Moodleshare [16]. For showing examples of this system, a series of snapshot images are shown in Use Figures 5, 6, 7, 8 and 9.

Figure 5a shows School A shares courses, namely "Basic Geometry" and "Introduction to statistics", while School B already has a course named "Multi-step equations" (Figure 5b). Figure 6 shows part of course content "Basic Geometry" in school A.

To be able to receive the synchronized contents, School B is required to provide empty course for the synchronization process holder, namely "HOLD_1" and "HOLD_2". "HOLD_1" and "HOLD_2" will be a holder for "Basic Geometry" and "Introduction to statistics", respectively.

Master side

As described in Figure 3, the Master side needs to perform function in course packer to transform current learning content into synchronization tables. Accordingly, in order to


Basic Geometry

School A ▶ Geometry Intro.

Calendar

November 2011

Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Events Key

Global Course
Group User

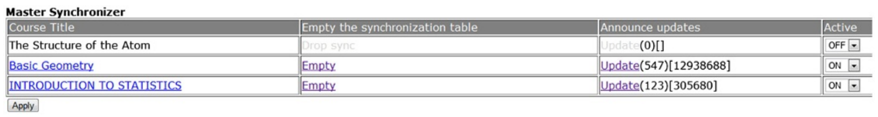
Topic outline

WELCOME TO GEOME

Welcome to the Geometry online course. This initial unit is designed as the introdu
calculator, protractor, compass, and a straight edge/ruler to complete this unit/cou
miscellaneous resources.

Unit Outline and Standard Alignment
Syllabus
Class Forum
Class Wiki
News forum

Figure 6 Snapshot of some part of content of school A's "Basic Geometry" (Master LMS).



Master Synchronizer

Course Title	Empty the synchronization table	Announce updates	Active
The Structure of the Atom	Drop sync	Update(0)[]	OFF
Basic Geometry	Empty	Update(547)[12938688]	ON
INTRODUCTION TO STATISTICS	Empty	Update(123)[305680]	ON

Apply

Figure 7 Snapshot of master synchronization control panel at school A (Master LMS).

inform other LMSs regarding the latest content available, this function needs to be re-executed every time content modification occurs. Course packer function is carried out by running master synchronizer control panel as shown in Figure 7.

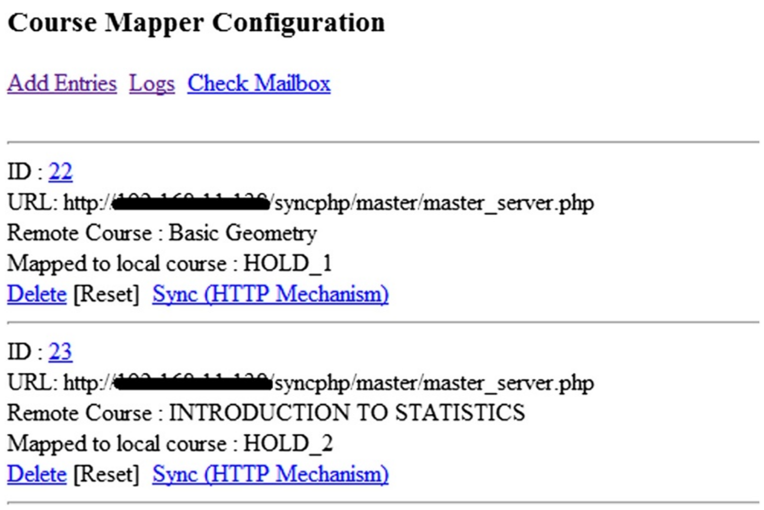
The figure shows all available course list. A particular course can be shared by activating the option on the right side of the screen. In this experiment, the course named “Basic Geometry” and “Introduction to statistics” are available to other LMSs while the course “Structure of the atom” is not. If the changes occur to any particular course, a relevant teacher/assistant needs to publish those changes by clicking the ‘Update’ link.

Slave side

The Slave side will perform course mapper function (shown in Figure 3) by the end of the synchronization process. The process is initiated from the Slave side through the Slave console application named course mapper.

School B is required to form a map in order to define the pair of *destination course holder* and *origin course*. The map requires information about : 1) the web service URL address where the course will be obtained from, 2) the course holder identifier where the course will be written into. Such map entry is created by course mapper application which is shown in Figure 8. This figure shows the map configuration for definition pair between the “HOLD_1” holder and the “Introduction to Geometry” course from School A.

By using the course mapper application at School B, either a teacher or an assistant can start the synchronization process by clicking the “Sync” link on any particular map definition as shown in Figure 8.



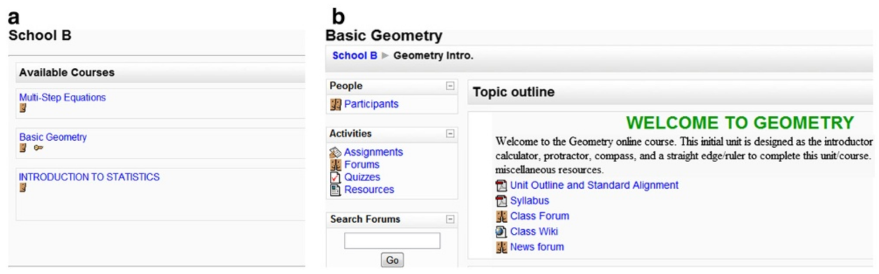
Course Mapper Configuration

Add Entries Logs Check Mailbox

ID : 22
URL: http:/[redacted]/syncphp/master/master_server.php
Remote Course : Basic Geometry
Mapped to local course : HOLD_1
Delete [Reset] Sync (HTTP Mechanism)

ID : 23
URL: http:/[redacted]/syncphp/master/master_server.php
Remote Course : INTRODUCTION TO STATISTICS
Mapped to local course : HOLD_2
Delete [Reset] Sync (HTTP Mechanism)

Figure 8 Snapshot of course mapper at school B (Slave LMS).



Welcome to the Geometry online course. This initial unit is designed as the introductor
calculator, protractor, compass, and a straight edge/ruler to complete this unit/course.
miscellaneous resources.

Figure 9 Snapshot of school B (Slave LMS) after synchronization. (**a**) Snapshot course list at school B (Slave LMS) after synchronization. (**b**) Snapshot of ‘Basic Geometry’ at school B (Slave LMS) after synchronization.

With synchronization process having taken place, school B will receive the updates, and the courses list will be modified as shown in Figure 9a. Figure 9b shows the part of course “Basic geometry” at school B which is formerly named “HOLD_1”.

Experiments using experimental satellite and Internet

Experiment environment

In 2007, the Japanese organization JAXA (Japan Aerospace Exploration Agency) and NICT (National Institute of Information and Communications Technology) launched an experimental satellite called WINDS (Wideband Internetworking engineering test and Demonstration Satellite). WINDS primary features are ultra-high-speed Internet communication capability and wide coverage. Its primary purpose is related to education, disaster management, and communication technology. It is also designed to have a capability of 155 Mbps down-link speed with a relatively smaller size ground station which is less difficult to install. As of 2008, the satellite had capability to cover one third of the globe [17].

Since its launch, JAXA-NICT has given opportunity for overseas and domestic institutions to carry out experiments using WINDS related to education, disaster management, and communication technologies. Having such opportunity, Kumamoto University (KU), Japan and Institut Teknologi Sepuluh Nopember (ITS) Surabaya, Indonesia created a collaborative experiment despite very limited available time and uncertain quality of the connection condition. In fact, the expected high downstream bandwidth was never realized. Approximately, fluctuating 1-5 Mbps of TCP connection is available instead.

In spite of the fact that the experiment was conducted between two parties, several institutions other than JAXA and NICT were involved for establishing connection in between. Figure 10 shows end-to-end connection configuration between KU and ITS. Connection between KU and ITS was established over many sub-networks. On the Japan side, KU established a connection to National Institute of Informatics (NII) and NICT (National Institute of Information and Communications Technology) via SINET (Academic network in Japan) and JGN2plus (Advance Testbed Network for R&D), then connected to WINDS satellite. The WINDS’s transponder provides connection to the ground station at ITB (Institut Teknologi Bandung, Indonesia). On the Indonesia side, connection to ITS is established through ITB by means of INHERENT (Indonesia Higher Education Network) network. Aside from using satellite, authors also conducted experiments between

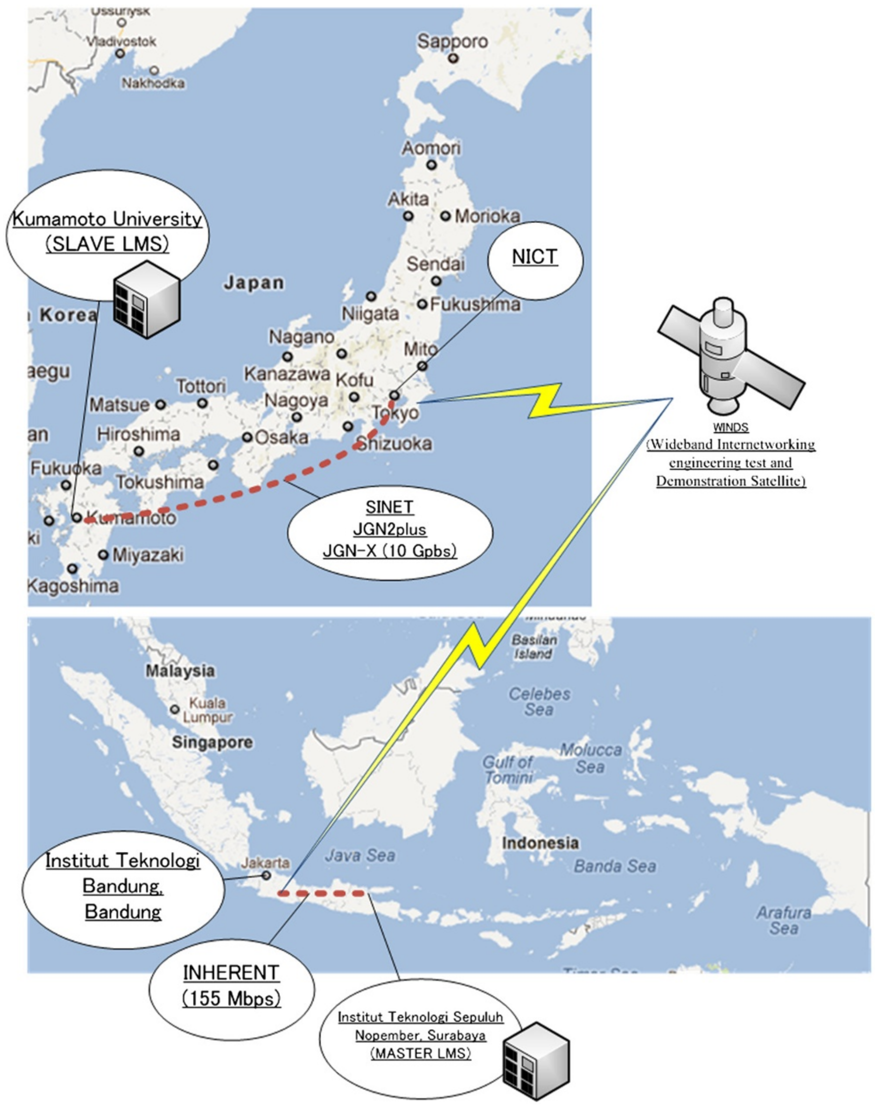

Kumamoto University (SLAVE LMS)
NICT
WINDS (Wideband Internetworking engineering test and Demonstration Satellite)
SINET JGN2plus JGN-X (10 Gpbs)
Institut Teknologi Bandung, Bandung
INHERENT (155 Mbps)
Institut Teknologi Sepuluh Nopember, Surabaya (MASTER LMS)

Figure 10 General experiment configuration between ITS(Indonesia) and KU(Japan).

both sides using the Internet during work hours. Authors measured that the performance of TCP connection between the involved servers is approximately 1.1 Mbps.

The experiment was conducted by carrying out a typical content situation on Master LMS's particular course (referred to as content thereafter). The subject of experiment is a course titled "Basic of geometry" which has size approximately 16 MB. In the rest of this section, MASTER and SLAVE is used to refer to Master subsystem and Slave subsystem respectively. On the other side, the slave side's operator clicks the 'Sync' link (referred to as 'Sync' hereafter) on each situation while at the same time recording bandwidth utilization. Several content situations are listed in Table 1. More detailed description about the activities/actions are as follows :

- 1) *New content.* Assuming content is just available at Master LMS (new) and Slave LMS has nothing of it. A large volume of data transfer is expected since SLAVE will request content from MASTER.

Table 1 Typical course situations for experiment

No.	Activity of MASTER's content
1	New content
2	No changes
3	Course section order is modified
4	Adds labels for particular course
5	Content's file is changed

- 2) *No changes*. No content changes at Master LMS. Slave LMS already has the latest content. A small volume of data transfer is expected since MASTER learns that SLAVE already has the latest content. Thereby, no download transfer is taking place.
- 3) *Course section order is modified*. Operator makes content changes at Master LMS, modifying course section order. A small volume of data transfer is expected since only few changes are made. MASTER replies the SLAVE with the updated part only.
- 4) *Adds labels for particular course*. Operator makes content changes at Master LMS, adding some additional labels and links to the content which is quite usual in e-Learning activities to give more resources for student to study. A small volume of data transfer is expected since only few changes are made. MASTER replies SLAVE with the updated part only.
- 5) *Content's file is changed*. Operator makes changes on certain files relevant to the content. A medium volume of data transfer is expected depending upon the file size. MASTER replies back to the SLAVE with the relevant file.

In the case of experiment using satellite, a conditional setup is carried out prior to the experiment as follows:

- Master LMS, placed in ITS, having done setup for a particular course to be available and sharable using Master synchronizer console.
- Slave LMS, placed in KU, having done setup for a particular course mapping (course-origin : course-local) using Course mapper console.
- For each typical content situation, Master side's operator performs the particular action. On the other side, operator clicks the 'Sync' link.
- Conducted 5 times, during work hours in December 2011 (11:00 JST - 17:00 JST).

Prior to conducting the experiment using Internet, a quite similar conditional setup is carried out as follows:

- Master LMS, placed in KU, having done setup for a particular course to be available and sharable using Master synchronizer console.
- Slave LMS, placed in ITS, having done setup for a particular course mapping (course-origin : course-local) using Course mapper console.
- For each typical content situation, Master side's operator performs the particular action. On the other side, SLAVE's operator clicks the 'Sync' link.
- Conducted 5 times, during work hours in June 2012 (11:00 JST - 17:00 JST).

Bandwidth utilization measurement

Bandwidth utilization measurement is conducted for each particular activity to see whether the concept works well. In normal course sharing activity, bandwidth utilization

Table 2 Measurement of bandwidth utilization

Activity	WINDS DL (kbps)	WINDS UL (kbps)	Internet DL (kbps)	Internet UL(kbps)
1	3220	0.04	43.98	3.20
2	1	9.32	0.13	94.66
3	9.8	6.10	14.66	53.80
4	12.5	7.12	20.30	50.50
5	66.5	0.38	51.22	464

on each activity would be appear approximately similar with activity number 1 in which it will be inefficient. Table 2 shows that the activity number 1, takes utmost amount of bandwidth utilization (3220 kbps (WINDS) and 43.98 kbps (Internet)).

Activity 2 takes the least utilization of among the other activities since both have the latest version of the content already. In this activity, SLAVE will send only its version and metadata information. From the version information, MASTER will learn that SLAVE already has the latest and MASTER will reply with no reply message. Such interaction appears to make the upload utilization (9.32 kpbs (WINDS), 94.66 kbps (Internet)) bigger than the download utilization (1 kbps (WINDS), 0.13 kbps (Internet)).

Activity 3 takes a quite bigger utilization than activity 2. MASTER replies with several updates regarding the changes. Therefore, it does not utilize bandwidth as performed in activity 1. The download takes 9.8 kbps (WINDS) and 55.80 kbps (Internet). Activity number 4 does quite similar with activity 3.The download takes 7.12 kbps (WINDS) and 50.50 kbps (Internet).

Despite quite similar with previous activity (3,4), activity 5 takes quite bigger utilization since file transfer are performed. In this system, file will be stored in synchronization table rather than its origin form in file system. If a particular file was removed from content and being replaced by another, the transfer will reflect the size of the file plus overhead of its synchronization table's metadata.

From the measurement, it appears that this system works as expected to utilize the bandwidth efficiently. By using differential delivery, the utilization reflects the degree of changes happened at origin.

Evaluation

This approach and the implementation has been working well in terms of functionality as shown in section "Content synchronization running example". Slave LMS has managed to synchronize a learning content from Master LMS. Experiment had been conducted both using Internet and limited time satellite network (WINDS).

Previously, authors made experiment using Moodle's course sharing plugin (Sharing cart [18]) and it takes approximate bandwidth of 51.6 kbps for each of the actions.

The measurement shows that by using this system, bandwidth utilization during synchronization process depends on the changes has been made on the Master side in which authors referred to as utilizing the bandwidth efficiently.

In addition, this system can be connected with other similar system in master-slave fashion allowing particular Slave LMS to become a Master LMS if necessary. This extended capability makes it possible to carry out further course sharing among LMSs.

Contributions

The main contribution of this paper is to provide a novel method for sharing e-Learning content between distributed LMS by using dynamic content synchronization. Furthermore, the contribution has an importance of addressing the need of resource sharing in educational sector within developing countries, in a form of course sharing between LMSs which supports collaborative teaching activity. In addition, the method presented is designed to address the concern of content sharing among LMSs in areas with network infrastructure limitation in terms of bandwidth and availability.

Related works

This section describes some existing sites and tools which are related to our work. Moodleshare [16] is a website dedicated to sharing courses to Moodle course format. This website appears to be a hub/repository for a Moodle user community to share their Moodle course. In each particular course, each contributor permits the public to re-use their content by providing a single archive file which is downloadable and ready to use with Moodle.

Community hub [19] is a recent advancement of later version of Moodle. Community hub allows people to share their work of learning content on a central repository server. Prior to this, sharing cart [18] has been existed to perform similar function. Both allows people sharing their work among distributed LMS.

Despite the fact that these methods provide re-usable content for sharing either on-line or off-line, they only provide the content to be shared in a static way. That means any particular content can be downloaded and then restored in particular user LMS, and using it within a semester without considering changes at the origin. In addition, such a way of sharing does not support collaborative teaching activity that may take place in the source side . In such teaching activity, a particular content may change at times within a short period of time. On the other LMS, while the course is in progress and conducting student activities, the existing shared content might need to be updated with recent changes from the origin side. Such update process should refrain from affecting current student activities which has not been addressed in the works mentioned earlier. Our approach tries to overcome such concerns by performing synchronization of courses with regards to maintain the current teaching activity as well as utilizing the bandwidth efficiently.

Conclusions and future works

An approach for unidirectional dynamic content synchronization between LMSs is discussed. This approach was evaluated by experimental connections between distributed LMSs to realize a distributed e-Learning environment. The experiments show that this system works well, and is capable of doing content synchronization between Learning Management Systems efficiently.

Bandwidth concerns in some parts of the world may still remain in the immediate future. Therefore, a further approach for delivering content updates in dynamic content synchronization in an efficient manner still needs to be considered. In this regards, e-mail system capability to deal with separated network (sender and recipient do not need to be in direct path route) making it potential to deliver the differential updates over unstable network channel infrastructure.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

AA, BCH, TU, and YC conducts the preliminary work including grand concept and the idea of course synchronization among Learning Management Systems. Meanwhile, RMI conducts the implementation including system design, experiments, acquisition of evaluation data as well as manuscript drafting and revising. All authors read and approved the final manuscript.

Acknowledgements

A part of this research is supported by Strategic Information and Communications R&D Promotion Programme –Regional topic (SCOPE) #082310003. The authors would also like to thank Institut Teknologi Bandung (ITB) member (Dr. Utoro and Dr. Basuki Suhardiman) for their great support in establishing a network link for the WINDS project experiment.

Received: 12 December 2011 Accepted: 17 October 2012
Published: 5 November 2012

References

1. Hassler B, Jackson AM (2010) Bridging the bandwidth gap: Open educational resources and the digital divide. IEEE Trans Learn Technol. 3:110–115
2. Usagawa T, Affandi A, Hidayanto BC, Rumbayan M, Ishimura T, Chisaki Y (2009) Dynamic synchronization of learning contents among distributed moodle systems. In: Proceeding of the 25th Annual Conference of JSET. JSET, pp 1011–1012
3. Affandi A, Firmansyah A, Hidayanto BC, Ishimura T, Chisaki Y, Usagawa T (2009) Performance of unidirectional LMS synchronization in various network capacity. In: Proceeding of The Asian Conference on Education, IAFOR, pp 1241–1246
4. Moodle.org (2011) Open-source community-based tools for learning. http://moodle.org/
5. Chavan A, Pavri S (2004) Open-source learning management with moodle. Linux Journal 2004: 2
6. Qing L, Lau RWH, Shih TK, Li FWB (2008) Technology supports for distributed and collaborative learning over the internet. ACM Trans Internet Technol. 8:5:1–5:24
7. Hunt JW, McIlroy MD (1976) An algorithm for differential file comparison. Computing Science Technical Report. Bell Laboratories
8. Git (2011) Distributed version control system. http://git-scm.com/
9. Free Software Foundation (2011) CVS - Open Source Version Control. http://www.nongnu.org/cvs/
10. Goodrich MT, Tamassia R (2004) Data Structures and Algorithms in Java, Wiley
11. Mehlhorn K, Sanders P (2008) Algorithms and Data Structures The Basic Toolbox, Springer
12. Rivest R (1992) Rfc 1321, the md5 message-digest algorithm
13. Tridgell A, Mackerras P (1996) The rsync algorithm. Technical Report TR-CS-96-05, Australian National University, Department of Computer Science (see also: http://rsync.samba.org)
14. Suel T, Noel P, Trendafilov D (2004) Improved file synchronization techniques for maintaining large replicated collections over slow networks. In: Data Engineering, 2004. Proceedings. 20th International Conference on, IEEE, pp 153–164
15. Squid (2011) Squid cache website. http://www.squid-cache.org
16. Moodleshare (2011) A place for Moodlers to share their Moodles. http://courses.moodleshare.org
17. WINDS (2011) Wideband internetworking engineering test and demonstration satellite. http://www.jaxa.jp/projects/sat/winds/index_e.html
18. Moodle sharing chart plugin (2011) Moodle Sharing chart plugin. http://docs.moodle.org/23/en/Sharing_Cart
19. MOOCH (2011) Moodle.org Open Community Hub. http://docs.moodle.org/23/en/Community_hubs

doi:10.1186/2192-1962-2-17
Cite this article as: Ijtihadie *et al.*: **Dynamic content synchronization between learning management systems over limited bandwidth network.** *Human-centric Computing and Information Sciences* 2012 **2**:17.