Indexing and Searching Learning Objects in a Peer-to-Peer Network

Aleksander Bułkowski and Edward Nawarecki
AGH University of Science and Technology
Dept. of Computer Science
Cracow, Poland
Email: nawar@agh.edu.pl

Andrzej Duda
Grenoble Institute of Technology
Grenoble Informatics Laboratory
Grenoble, France
Email: duda@imag.fr

***Abstract*—In this paper, we explore the idea of using Peer-to-Peer (P2P) networks as advanced educational tools. A P2P network can disseminate complex learning objects that act as anchors for creating collaborative learning communities. The goal is to create collaborative spaces of learners with similar interests to exchange knowledge, opinions, and experience so that they can learn, understand, and help/teach each other. The use of the P2P technology for disseminating learning objects requires extending current P2P systems with a support for precise indexing and searching. In this way, potential learners can easily find and choose relevant objects. The present paper describes the design of an indexing and searching scheme for disseminating complex mutable SCORM learning objects over BitTorrent.**

***Index Terms*—learning objects; peer-to-peer systems; indexing; DHT**

I. Introduction

In this paper, we explore an idea for future educational tools based on Peer-to-Peer (P2P) networks. Although such networks offer a wide-spread means for disseminating media content, currently they are mostly limited to music and movies. However, a P2P network may also provide easy access to educational resources instead of, or in addition to, standard publishing of course material on Web servers, which often requires considerable management and maintenance effort. In a similar way to video or music files, we can easily disseminate complex *learning objects* in a standard format such as SCORM [1] among interested learners. As P2P networks operate in an autonomous and spontaneous way with minimal management overhead, users can use P2P applications with little or no training.

We have begun to investigate how educational activities can benefit from P2P networks and how we need to enhance current P2P applications for disseminating learning objects. This goal requires addressing the problem of precise indexing and searching so that potential learners can easily find and choose relevant objects. Another idea to explore is to use learning objects as anchors for creating *collaborative learning communities*. Current P2P applications only disseminate content, but users are not aware of who downloads a given file. Knowing that somebody else is interested in a movie or a music file may lead to interesting user interactions such as following the choice of somebody else or establishing collaborative opinions and advices. If we can attach some information about users interested in specific content to files disseminated over P2P networks, we would enable creation of collaborative spaces of users with similar interests. Once such a social network starts, its members can exchange knowledge, opinions, and experience as well as they can extend their activities to learning processes by providing support, explanations, discussions, and even mutual teaching. One interesting feature of such a set up is that collaborative learning communities may enhance the existing content and add more resources to a given course material. Thus, we need to take into account that learning objects become mutable unlike files disseminated in current P2P networks.

The present paper describes the design of an indexing and searching scheme for disseminating complex mutable SCORM learning objects over BitTorrent [2]. Our approach takes advantage of attributes extracted from SCORM objects and of indexing concepts and terms added by the user from a domain ontology. We use OpenDHT [3], an open distributed storage service implemented over Planet Lab [4] to store all needed indexes and associations. We have implemented the proposed schemes in GLEN (Global Lecture Exchange Network), a prototype P2P client based on Azureus (Vuze) [5].

II. Design goals

Our goal is to disseminate learning objects in a P2P network in a way that enables creation of collaborative communities of learners. We recall below the design goals of our system:

- Support wide and easy distribution of learning objects. Extend existing P2P applications for sharing, searching, and downloading learning objects in a fully distributed way without a central server.
- Change the traditional hierarchical teaching model into a flat one in which anybody can teach anybody. Anybody can download any learning object, extend its content, or add a new learning object.
- Create collaborative communities of learners having common interest and objectives based on the access to the same learning objects.
- Use learning objects as anchors for easy interactive communication between learners (e.g. through Voice over IP (VoIP) applications such as Skype or Gizmo). Down-

978-1-4244-6571-2/10/$26.00 ©2010 IEEE

loading a learning object gives information about other learners that share the same interests.

Based on this set of requirements we have developed several new schemes for structuring, indexing, and searching learning objects. We describe them below.

III. Structure of complex learning objects

We consider learning objects in the SCORM 1.2 format that allows linking various resources and adding descriptions in the LOM format [6]. Popular engines like Blackboard or Moodle can display such learning objects and several editors exist such as eXe [7].

A. BitTorrent and DHT

We have decided to use BitTorrent as the underlying P2P network and enhance a P2P client based on Azureus (Vuze) [5] to support advanced functions of indexing, searching, and retrieving learning objects. BitTorrent disseminates objects through a *torrent* file that contains the information on an object needed for downloading (name, piece length, number of pieces, tracker, etc.). A *tracker* is a HTTP server that keeps information on *peers*—nodes downloading the same object. There are two categories of peers: *seeders* that have already downloaded the whole object and can provide its pieces to *leechers*, the peers that have not yet obtained all the pieces.

To download an object, a peer first downloads a torrent file, computes the SHA1 hash function on its information part, and contacts the tracker to obtain seeders (their IP addresses and ports) that can send the pieces of the object. Then, it queries them to obtain a bitmap of pieces stored at each seeder and requests their download. A leecher also provides already downloaded pieces to other peers.

In the *trackerless* operation of BitTorrent, the information usually maintained by a tracker (peers, their addresses and ports) is stored in the DHT (Distributed Hash Table) organized on peers in a fully distributed way. Even if Azureus only uses the DHT for storing the information usually maintained by a tracker, we explain below the operational principles of the DHT, because we rely on such a distributed storage service for indexing and searching.

Azureus uses a modified Kademlia implementation for its DHT [8] to store $(key, value)$ pairs on different nodes. The DHT interface offers the following operations:

- $Ping$ to verify routes to other peers,
- $Lookup(key)$ to find nodes that are near to the key in the keyspace (nodes are identified by the SHA1 hash of the node IP/port combination),
- $Store(key, value)$ to store the $value$ on the nodes close to the key in the keyspace,
- $Get(key)$ to retrieve the $value$ from the nodes close to the key in the keyspace.

To explain the operation of the DHT, we take an example of storing files in the DHT (other P2P networks such as eDonkey or eMule for example use a DHT to store and retrieve files). A file is identified by a key derived from its filename—a key is a 20 byte SHA1 hash function on a filename: $key = h(filename)$. The DHT maps keys to nodes in the P2P network: the keyspace is the set of 20 byte strings and each node has its place in the keyspace. To store a file, a node first performs the $Lookup(key)$ operation to find nodes that are near to the key and then stores the file as the value of the $Store(key, value)$ operation at 20 nodes close to the key in the keyspace. Retrieving the file proceeds in the inverse way: a node looks up nodes close to a key and gets the file from one of the nodes.

When Azureus operates in the *trackerless* mode, it uses its DHT to only store the *torrent* information on *peers*, then downloading of pieces proceeds in parallel from a large number of peers. The key of the *torrent* information is a 20 byte SHA1 hash function on the descriptor (the information part of a torrent): $key = h(descriptor)$ and the stored value corresponds to the information on the seeders usually maintained by the tracker and necessary for downloading.

B. Learning objects and their structure

We want to design the structure of our learning objects so that they may evolve while they disseminate in a collaborative community: we want to be able to enhance the existing content and add more resources, which results in multiple versions of a learning object. However, existing P2P objects like an MP3 or a video file are immutable, i.e. they cannot be modified once published on a P2P network. Thus, we need to enhance the structure of P2P objects so they can be modified within a collaborative community.

We obtain this goal by associating multiple identifiers with a learning object. The first identifier is called `LOID` (Learning Object Identifier) and multiple versions are identified by a version specific `VSLOID`. Both identifiers follow the same scheme as identifiers in BitTorrent—they are unique 20 byte values generated by computing SHA1 function on the object descriptor. Finally, each version is associated with a *torrent* that allows to download a given learning object.

We need to maintain the associations between the identifiers and an object by storing them in a persistent database. In the spirit of P2P networks, we wanted to use a DHT to store all needed associations as well as the indexing information described below. One choice of a DHT would be to use the Azureus DHT for this purpose, however running an operational service would require upgrading many nodes with our modified version of Azureus, which was difficult to achieve. Instead, we have used an open distributed storage service called OpenDHT, a similar system that presents the same basic DHT interface of the operations $Store(key, value)$ and $Get(key)$. It provides a continuous DHT service over a large number of Planet Lab nodes distributed all over the world.

Our learning objects are thus represented as the following associations of identifiers and stored in OpenDHT as $(key, value)$ pairs (the left part of the association serves as the key and the right part as the value):

```
LOID -> VSLOID
VSLOID -> LOID
```

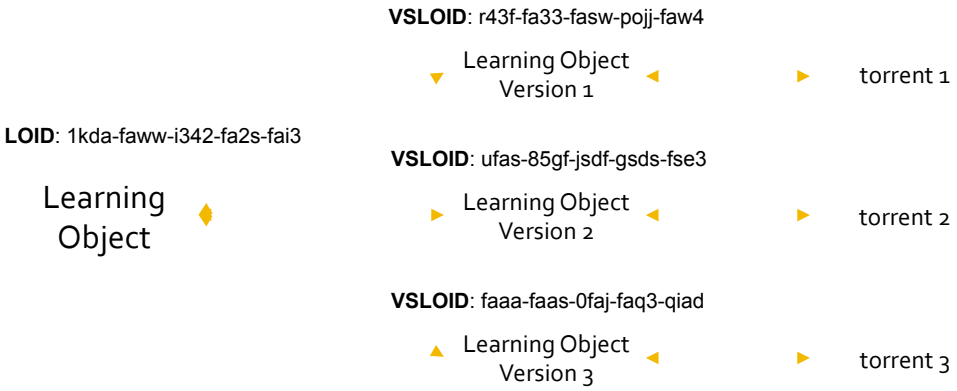

Figure 1. Learning object identifiers.

`VSLOID -> torrent`

When we know an identifier of a given learning object, we can retrieve its version identifiers and the torrent information required for downloading. Figure 1 illustrates the notions of `LOID`, `VSLOID`, and their relationships.

IV. Metadata and Domain Ontology

We propose to enable precise searching through the use of *metadata* extracted from a learning object such as traditional attributes Author, Title, Description, and others defined for instance in the Dublin Core [9]. Metadata also includes *concepts* from a domain ontology defined for a given class of learning objects. A domain ontology is a network of domain model concepts (topics, knowledge elements) that defines the elements and the semantic relationships between them [10].

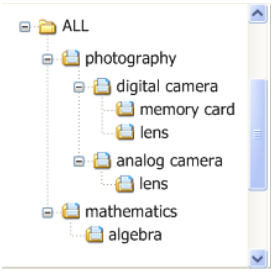

Figure 2. Example of a domain ontology.

Figure 2 presents an example of a predefined ontology with simple inheritance relationships created in the RDF format [11] and built-in in our modified Azureus client. The ontology forms a tree of concepts and terms from the most general one to more specific concepts and terms. For instance, we have the concept of `digital camera` that belongs to the concept of `photography` or `algebra` belonging to `mathematics`. The advantage of using such an ontology is that we can add more general concepts to indexing terms and use them as querying criteria to find related objects. We can obtain an opposite effect for terms that have several meanings—we can distinguish between them by specifying a more general parent concept (such as `lens` for `digital camera`) to restrict a query to more specific and precise results.

Indexing consists of creating an inverted index that associates metadata information with a given `VSLOID` representing a learning object. We use attributes from the SCORM description (Dublin Core) or the LOM part of the object, for instance attributes like `Title: Digital Photography Tips`, `Author: John Smith`. The example attributes are stored in the inverted index as the following relationships:

```
Title_Digital -> VSLOID
Title_Photography -> VSLOID
Title_Tips -> VSLOID
Author_John -> VSLOID
Author_Smith -> VSLOID
```

Similarly to maintaining the structure of a learning object, we store the inverted index in OpenDHT as $(key, value)$ pairs. In the example above, the first association is for instance stored as $key =$ `Title_Digital` and $value =$ `VSLOID`.

When indexing an object, the user can also add indexing terms from the domain ontology. We encode ontology terms by concatenating `ONT` prefix to a term or a concept. Thus, the association has the following form in the DHT:

```
ONT_<TERM> -> VSLOID
```

For instance, when indexing a course on digital photography, the author chooses the concept of `lens` to describe the learning object, we also add more general concepts from the domain ontology—the parent relation between `digital camera` and `photography`. Thus, the following indexes are

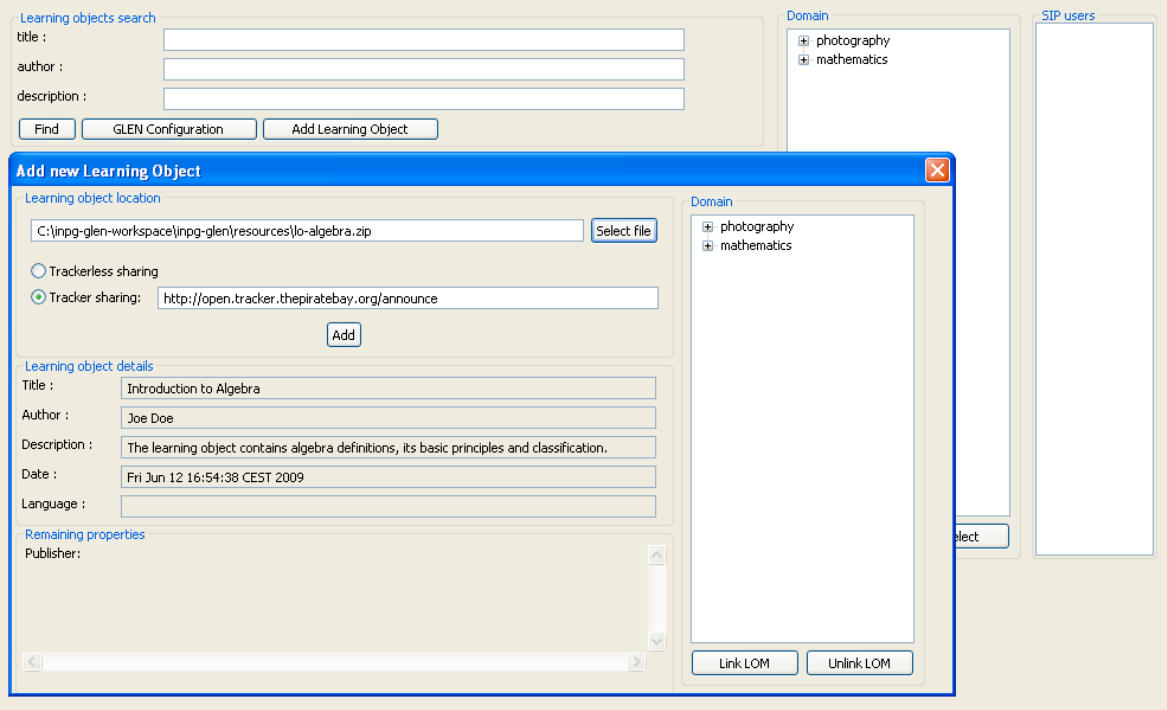

Figure 3. Indexing interface of GLEN.

stored in the DHT:

```
ONT_lens -> VSLOID
ONT_digital-camera -> VSLOID
ONT_photography -> VSLOID
```

Using the ontology for indexing enables for instance finding the course relevant to digital photography even if the user specifies `lens` as the searching term.

Finally, we add support for creating communities by associating the information of VoIP addresses with a given learning object. In this way, those who download a given object can contact other learners in an easy way via VoIP communication. We store the object identifier with the `COM` prefix and the SIP address of the user as the following association in the DHT:

```
COM_'VSLOID' -> SIP-ID
```

When the user finds and downloads an object, we need to present its metadata information to the user. For this goal, we store direct indexes in the DHT—they have the following form in our example:

```
VSLOID -> Title_Digital
VSLOID -> Title_Photography
VSLOID -> Title_Tips
VSLOID -> Author_John
VSLOID -> Author_Smith
VSLOID -> ONT_lens
VSLOID -> ONT_digital-camera
VSLOID -> ONT_photography
```

V. Implementation

We have developed GLEN (Global Lecture Exchange Network), a prototype of a P2P client based on Azureus (Vuze) [5]. We have extended the Azureus interface to offer the function of adding a learning object and searching the network. The indexing and searching schemes presented above use the DHT functionality of OpenDHT. In the current version, we use a tracker for downloading and we work on the development of the trackless version.

To present the possibilities of the prototype, we illustrate its main features. Figure 3 shows the window for adding a new learning object. The user publishes the object in the SCORM format and the most important semantic attributes are extracted from the object. Then, the user can add terms and concepts from the domain ontology (on the right of the figure) with the two main built-in concepts for testing purposes: `photography` and `mathematics`. GLEN stores the indexing information in OpenDHT as described previously and offers a searching interface to the user.

The user can specify search attributes such as title, author, or description keywords, and choose a concept from the domain ontology. The parent terms of the concept are automatically added and the system queries OpenDHT to find relevant `VSLOID`s.

Figure 4 presents the searching interface in which the user specifies `Author: joe`. GLEN retrieves `VSLOID`s linked to the attribute and lists the learning objects relevant to the query (in our example there is only one relevant object—`Introduction to Algebra`). When the user clicks on the title, more details are shown and user can decide to

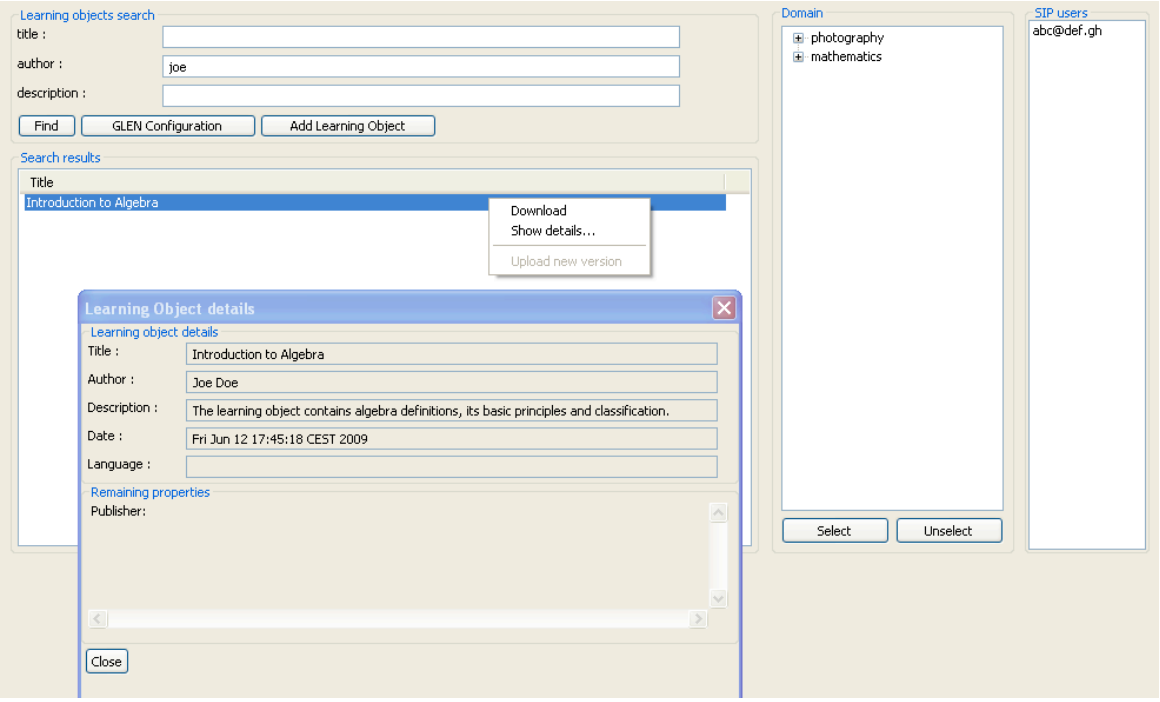

Figure 4. Searching interface of GLEN.

download the object. Its SIP address is stored in OpenDHT and she becomes the member of the community related to the object.

VI. Related work

Some work has already pointed out the importance of applying P2P technologies to education. Edutella has proposed a peer-to-peer architecture for exchanging RDF-based metadata [12]. It builds upon Semantic Web techniques and the JXTA middleware. Its purpose is to make the reuse of globally distributed learning resources easier. Berman and Annexstein has considered P2P technologies as crucial in future educational systems [13]. In particular, they propose to integrate them in a new *personal knowledge management* paradigm, which is useful for students and educators in many activities encountered in everyday teaching, researching, and learning.

Much work considered indexing schemes for P2P networks and we mention here only a few proposals. Felber et al. describe techniques for indexing data in P2P networks based on hierarchically organized multiple indexes and distributed across the nodes of the network [14]. They present several interesting properties, but require a profound modification in the DHT operation of existing systems. Our design attempted to reuse the existing DHT structure for indexing and searching learning objects. Semantic Overlay Networks (SON) represent another approach to searching in P2P networks: nodes connect to other nodes with similar content so that queries can be propagated to appropriate semantically related nodes [15].

Although DHT networks that we used in our indexing scheme offer efficient access to searched items, they can not effectively support partial-match or approximate (proximity) queries. Cohen et al. proposed a design based on unstructured architectures such as Gnutella and FastTrack with the support for partial match queries and relative resilience to peer failures while obtaining orders of magnitude improvement in the efficiency of locating rare items [16].

VII. Conclusion

In this paper, we have presented an approach to enhance current advanced educational tools. Our idea is to take advantage of widely-spread and adopted P2P networks for disseminating learning objects. We consider them as an ideal means for creating collaborative communities of learners with similar interests.

We have described a scheme for indexing and searching complex mutable SCORM learning objects over the P2P network of BitTorrent. We use OpenDHT, an open distributed storage service implemented over Planet Lab to store all needed associations: links between a learning object, its versions, and a torrent, as well as inverted indexes and the information on SIP addresses of community members. Our indexing scheme extends traditional attribute-based approaches with precise indexing terms from a domain ontology. We have enhanced Azureus, an open source BitTorrent client with our indexing support and added a searching interface. Our first experiences with the prototype show that the indexing functionalities contribute to improved searching. We continue to test the support for creating collaborative communities and providing easy VoIP communication between its members.

REFERENCES

[1] ADL, "Sharable Content Object Reference Model (SCORM)," in *http://www.adlnet.org*, 2004.

[2] BitTorrent. [Online]. Available: http://www.bittorrent.com

[3] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: A Public DHT Service and Its Users," in *SIGCOMM*, 2005.

[4] PlanetLab. [Online]. Available: http://www.planet-lab.org

[5] Azureus, "An Open Source BitTorrent Client." [Online]. Available: http://azureus.sourceforge.net

[6] R. T. Mason and T. J. Ellis, "Extending SCORM LOM," *Issues in Informing Science and Information Technology*, vol. Volume 6, 2009.

[7] eXe, "The eLearning XHTML editor." [Online]. Available: http://exelearning.org

[8] P. Maymounkov and D. Mazieres, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *1st International Workshop on Peer-to-peer Systems*, 2002.

[9] Dublin Core Metadata Initiative, "Dublin core metadata element set (DCMES) Version 1.1. Recommendation," in *Dublin Core Metadata Initiative*, 1999.

[10] E. Nawarecki, G. Dobrowolski, S. Ciszewski, and M. Kisiel-Dorohinicki, "Ontology of Cooperating Agents by Means of Knowledge Components," in *LNCS, Vol. 2691, Multi-Agent Systems and Applications III,*, 2003.

[11] W3C, "Resource Description Framework (RDF)," 2004.

[12] W. Nejdl et al., "Edutella: a P2P Networking Infrastructure based on RDF," in *Proc. of 11th World Wide Web Conference*, 2002.

[13] K. Berman and F. Annexstein, "An Educational Tool for the 21st Century: Peer-to-peer Computing," in *Ohio Learning Network Conference, Windows on the Future Conference*, 2003.

[14] P. Felber, E. Biersack, L. Garces-Erce, K. Ross, and G. Urvoy-Keller, "Data Indexing and Querying in P2P DHT Networks," in *ICDCS, Tokyo*, 2004.

[15] H. Garcia-Molina and A. Crespo, "Semantic Overlay Networks for P2P Systems," Stanford InfoLab, Technical Report 2003-75, 2003.

[16] E. Cohen, A. Fiat, and H. Kaplan, "Associative Search in Peer To Peer Networks: Harnessing Latent Semantics," *Computer Networks*, vol. 51, no. 8, 2007.