<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>HyperCuP Lightweight Implementation: A Universal Solution for Making Application Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Grzonkowski, Slawomir; Kruk, Sebastian Ryszard</td>
</tr>
<tr>
<td><strong>Publication Date</strong></td>
<td>2006</td>
</tr>
<tr>
<td><strong>Item record</strong></td>
<td><a href="http://hdl.handle.net/10379/636">http://hdl.handle.net/10379/636</a></td>
</tr>
</tbody>
</table>

Some rights reserved. For more information, please see the item record link above.
ABSTRACT

Contemporary applications need an efficient solution for communication to implement robust information retrieval mechanisms and fault tolerant networks. Apart from implementing an robust, scalable communication protocol the solution should be accessible with easy to use API that would not require too much of an effort to use it.

In this article we present HyperCuP Lightweight Implementation (HLI) which delivers an alternative P2P architecture based on web services. This implementation has already been deployed with diverse systems like JeromeDL, a semantic digital library and FOAFRealm, a distributed identity management system based on social networking.

We describe an architecture of the HyperCuP Lightweight Implementation. We show how to deploy it with one’s own application and how to take advantage of the established hypercube topology.

Keywords
Distributed Computing, HyperCuP, P2P networks

1. INTRODUCTION

The contemporary applications must be able to process many queries per second, especially digital libraries that are affiliated with large universities and host huge databases to thousands of students. Because of the fact that universities keep both daily and extra-mural studies, digital libraries are overloaded during end-of-term examinations period. Furthermore, many of these libraries offer fancy features like collaborative groups, searching in network of federated libraries or Single sing-on registration. Unfortunately, as long as digital libraries do not utilize semantics, users will repeat similar queries many times, because first search results usually do not respond to desired information being sought.

Operations like looking for resources and authentication in distributed environment cause undesirable network traffic. Our work identifies and combines several techniques from the Semantic Web to P2P networks, which results in improving efficiency of communication in e.g. searching for resources and managing users profiles.

The remainder of this short article is organized as follows: section 2 describes problems and requirements of distributed systems. Section 3 provides a short description of the HyperCuP Lightweight Application. Finally in section 4 we describe the overview of the demo we would like to present during ESWC 2006.

2. P2P INFRASTRUCTURE FOR SCALABLE DISTRIBUTED COMMUNICATION

Eventhough most of contemporary applications implement distributed (or sometimes even ubiquitous) computing paradigm there is lack of support for developing this paradigm in a lightweight fashion. Although the requirements are usually similar, we can found as many various solutions as projects we encounter. Unfortunately, hardly any of existing solution have satisfied our requirements. First of all, the application like a digital library needs an efficient broadcast algorithm. Moreover, during the search process all nodes must be equally balanced in order to prevent from Denial Of Service (DoS) attack. Secondly, new digital library servers should not affect the overall network efficiency. Therefore the solution has to be scalable. Finally, we required an open-source lightweight framework that could be easily adapted to existing applications delivering new axis of distributed computing with least effort possible.

After investigating the problem we have encountered the idea of HyperCuP (Hyper Cube in P2P) network. The HyperCuP [4] protocol was invented by Schlosser, Sintek, Decker and Nejdl as a P2P protocol based on a topology also known as Galey graph structure.

The protocol provides a fast and an efficient broadcast algorithm which sends the minimum number of messages across the network. Moreover, HyperCuP lets nodes to join and leave the network at any time. The HyperCuP infrastructure tends to be balanced most of the time. This can help in prevent the application utilizing HyperCuP for communication from Distributed Denial of Service attacks. In the balanced stated, a total number of messages sent to the network in each broadcast is always equal to \( \log(n) \), where \( n \) is the number of nodes in the network.

The reference implementation of HyperCuP has been developed in the Edutella [1] project. Although the source code of Edutella is available as an opensource project, we could not extract the actual core of the HyperCuP proto-
col to use it in our projects. In addition, Edutella contains many modules which are firmly depended each other. Those facts induced us to design and implement our own application. Based on the requirements presented earlier we have decided to take the lightweight approach.

3. HYPERCUP LIGHTWEIGHT IMPLEMENTATION

The aim of HyperCuP Lightweight Implementation (HLI) implementation is to make the opensource system that provides an easy to use, lightweight framework for extending almost any kind of applications with distributed computing features. HyperCuP provides programmer friendly API that do not require too much effort in order to start using it in existing projects. This section provides a short overview of the architecture and describes the practical aspects of using HLI.

The Broadcast Module allows user to decide how a peer (an instance of HLI-enabled application) behaves upon the arrival of the request from the network. According to the lightweight approach only a couple of requirements have to be met to enable HyperCuP in the application. The most important one is to implement the broadcast paths. There are no constraints on either an implementation of the broadcast processing that is independent on the actual application that is HLI-enabled. The demonstration will consist of:

3. Connecting nodes to the hypercube topology.
4. Executing the query and show the results.
5. Presenting of existing solutions based on HyperCuP:

4.1 Acknowledgments

This work was supported by Science Foundation Ireland Grant No. SFI/02/CE1/I131 and by the Knowledge Web project (FP6 - 507482) and partially by KBN, Poland under grant No. 4T11C00525. The authors would like to acknowledge Stefan Decker, Paweł Bugalski, the DERI Semantic Web Cluster and the Corrib.org working group for fruitful discussions.

5. REFERENCES