HyperCuP Lightweight Implementation

A Universal Solution for Making Application Distributed

Sławomir Grzonkowski
WETI, Gdansk University of Technology, Poland
DERI, NUI Galway, Ireland
slawomir.grzonkowski@deri.org

Sebastian Ryszard Kruk
DERI, NUI Galway
IDA Business Park
Lower Dangan, Galway, Ireland
sebastian.kruk@deri.org

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 opensource 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 Caley 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 $\texttt{log}(n)$, 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-

Demos and Posters of the *3rd European Semantic Web Conference (ESWC 2006)*, Budva, Montenegro, 11th - 14th June, 2006

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.

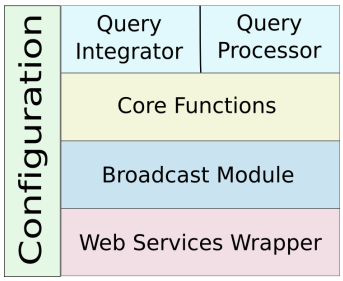

Figure 1: Architecture diagram

3.1 Architecture

The overall architecture of HyperCuP Lightweight Implementation consists of five modules (see Fig. 1). The *Web Services Integration* module is responsible for communication between different instances of HLI. This module is responsible for working in distributed heterogeneous environments. Web services support allows significant interoperability in HLI and delivers the process of turning on SSL support is far easier.

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 `LocalQuery` interface (located in the *Query Processor* module) by the external application. The interface has only one method `performQuery` which is invoked when the broadcast message arrives to the peer. The implementation of this method changes the actual behavior of the peer.

The *Broadcast Query Integrator* module delivers the implementation of the broadcast processing that is independent on the actual application that is HLI-enabled. The final results of the broadcast message consists of the request from the sender and responses from the all peers along the paths integrated in this peer.

Finally the *Core Functions* module delivers HyperCuP protocol essential code for creating networks, joining peers or monitoring the state of the network.

It is worth to mention that the implementation details are transparent from the user–programmer perspective. HyperCuP Lightweight Implementation required implementation of only one one interface in order to make application work. Additionally managing behaviour of HLI can be done with *Configuration* module.

3.2 Practical Use

Deploying Lightweight HyperCuP Implementation with existing application requires several steps. In the beginning, the HyperCuP component has to be initialized by setting some attributes like the address of the web services interface of this HyperCuP component and the implementation of the local query interface (currently defined in Java API). Performing the query results in invoking implementation of the method `performQuery` that should be registered during the initialization step. In result the query is being executed on every node of the P2P network along the broadcast paths. There are no constraints on either an implementation of the `performQuery` method or the way the query message should be handled, except the requirement that objects passed as parameters to this method have to implement the Java `Serializable` interface.

One additional step is required when running application for the first time. Since the topology has to be set up, peers must connect to the HyperCuP network by connecting to any peer in the network. According to the HyperCuP protocol this the connection request is routed to the appropriate peer in order to keep the network in a balanced state.

4. PRESENTATION PLAN

During the demo session we will present how to deliver distributed paradigm with HLI in a couple of steps to an existing application. The demonstration will consist of:

1. Deployment with an example web-application.
2. Preparation and implementation of an example query.
3. Connecting nodes to the hypercube topology.
4. Executing the query and show the results.
5. Presentation of existing solutions based on HyperCuP:
   - *distributed search protocol* in JeromeDL [3] - a semantic digital library.
   - *distributed authentication protocol* in D-FOAF [2] a distributed identity management system.

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

[1] Edutella project: http://edutella.jxta.org/.

[2] S. Grzonkowski, A. Gzella, H. Krawczyk, S. R. Kruk, F. J. M.-R. Moyano, and T. Woroniecki. D-FOAF - Security Aspects in Distributed User Managment System. In *TEHOSS'2005*.

[3] S. R. Kruk, S. Decker, and L. Zieborak. JeromeDL - Adding Semantic Web Technologies to Digital Libraries. In *DEXA Conference*, 2005.

[4] M. Schlosser, M. Sintek, S. Decker, and W. Nejdl. Ontology-Based Search and Broadcast in HyperCuP. In *International Semantic Web Conference, Sardinia*, 2002.