Celsius Bloodhound
Automatizing searching and fetching records from library servers.

Prof. Ing. Marisa R. De Giusti[1]
PrEBi UNLP - Comisión de Investigaciones Científicas de la Provincia de Buenos Aires
marisa.degiusti@ing.unlp.edu.ar

Gonzalo L. Villarreal[2]

Abstract - **The Library Linkage Project (PrEBi) has as main objective to share the bibliographic acervus resident in ISTEC Institutions, in order to satisfy the demand of Researchers, Teachers and Students. To manage the bibliography requirements (journal articles, book chapters, congress proceedings, theses and patents) made by users, it's been developed the Celsius Software. Nowadays the main job of Celsius administrators is to lookup requests in ISTEC and non-ISTEC Institutions' catalogs by hand which takes many hours to check all until the resource is found. In order to perform this task more efficiently, it has been created in PrEBi the Celsius Bloodhound Project. This project, currently in analysis and design stage, intends to automatize this tedious job by searching and fetching records from library servers. Bloodhound's design is flexible enough to allow connections from almost any application and to connect to many different remote catalogues. Bloodhound will be able to interact transparently with many servers, taking advance of existing technologies and protocols, improving quality service in the search of resources in PrEBi and other Institutions as well.**

Index Terms – ISTEC Library Linkage Initiative, document search and retrieval, Celsius Software.

INTRODUCTION

The Library Linkage Project (Proyecto de Enlace de Bibliotecas - PrEBi) [1] of La Plata National University (UNLP) [2] is part of the LL initiative of the Ibero American Science and Technology Education Consortium (ISTEC) [3]. The main objective of LL is to share the bibliographic acervus resident in ISTEC Institutions, in order to satisfy the demand of Researchers, Teachers and Students. In this model, PrEBi serves both local and international requests.

PrEBi can request bibliographic material from the online catalogues of over 50 Libraries from America and Spain: Book chapters, Congress Proceedings, Journal Articles, Theses and Patents not available in UNLP libraries.

There are also other 300 libraries from Europe and America that belong to Institutions not member of ISTEC, that make with PrEBi the same kind of exchange.

For the management of the users' bibliographic requests – local and foreign –, the Celsius software (actually in its version 1.6 stable) and the Celsius Network (still on testing stage) were created. The Celsius software, completely developed and supported by PrEBi UNLP, offers the Administrators and Referencists a simple and clear interface to perform the integral management of user requests, making the tracking of them from the moment they are started until the time when they are handed in to the users (in digital format or printed paper), offering complete listings as well as a variety of functions to interact with the requests, the users and the library catalogues, and providing a wide range of statistics online, available to all visitors of the site, which allows to corroborate the quality of the services and the global functioning of the project. Celsius also allows the users of PrEBi to access a personal site, where they can do a follow up of their current requests, download available documents, accessing personal statistics and historical requests listings and update their personal information.

Nowadays Celsius is used in over 30 institutions in America and Spain, and has its own website where its possible to download patches, translated files, new versions, administration manuals and accessing the forums. In Celsius website (http://celsius.prebi.unlp.edu.ar) visitors may find the Celsius Directory, in which there is a list of the installed Celsius Instances available.

Even though Celsius offers a complete list of Library Catalogues and permits registering searches on the catalogues, Celsius operators currently have to access each catalogue and search through each one for each user request. This tasks demand an incredible amount of time, possibly taking several hours or even days until the request is found – if so happens- which makes the operators to be unable not attend other requests and therefore possibly generating

[1]Engineer Profesor Marisa Raquel De Giusti. Researcher Comisión de Investigaciones Científicas de la Provincia de Buenos Aires - CIC. Director of Proyecto de Enlace de Bibliotecas (PrEBi[4]) an Servcio de Difusión de la Creación Intelectual (SeDiCI)[5]

[2]Gonzalo Lujan Villarreal. Computing Analyst. PrEBi SeDiCI member. gonetil@sedici.unlp.edu.ar

bottlenecks because of the great amount of requests in search. To work out this situation, the Celsius Bloodhound project was created, which pretends to realize the search tasks in a more efficient way, automating partially or completely the search and finally optimizing the service.

CELSIUS BLOODHOUND

Celsius Bloodhound is an application that will have as primordial objective the optimization of the search of bibliographic requests, basically, Celsius Bloodhound will search in background the users requests that are in *pending* state, and will inform of the results to the Celsius operators, together with the information corresponding to each request. But the final objective of Celsius Bloodhound goes beyond the optimization of the Celsius searches; any application will be able to request to Bloodhound to search one or several documents into the catalogues, receiving the obtained results.

In the same way that Celsius Bloodhound will allow different applications to connect with it and request searches, it will also be able to connect to many different catalogues - where each one of these catalogues can be using a different protocol and technology to expose its databases online and have different search mechanisms inside these databases -, and obtain results from them in order to deliver those results to whom has requested them, everything in a transparent way to the end user or client application. Although Celsius Bloodhound will recover much information from the remote source, it will emphasize in which volumes the library has.

The variety of objectives proposed in the Celsius Bloodhound Project (BH) presents a long list of challenges of different complexity, that will have to be accomplished to allow a good performance and an appropriated use of the existing resources (processor, memory, database, network). This list includes (but is not limited to):

- Displaying a simple and clear interface to interact with any application that requires searches.
- Connecting to different repositories, with different technologies and search mechanisms.
- Maximizing the concurrence, so the incoming of new requests does put off the active searches.
- Allowing parallel searches in the different repositories, balancing the load between pending searches and active searches.
- For each search request, keeping queue lists registering results obtained for the moment, and registry of pending repositories.
- Offering security mechanisms, to allow that only the authenticated and authorized users (applications) can make search requests.
- Offering mechanisms to limit the search requests per application in order to avoid abusive use.
- Optimizing the bandwidth, by restricting it in high traffic hours and increasing it in low traffic hours.
- Keeping a historical registry of made searches and the obtained results.
- Presenting a broad variety of statistics, including data per application, requested title, catalogue, times, delays, down servers, etc.

These objectives raise the need of making a modular and scalable design, where each module has a simple and well defined function that allows a first basic implementation for then incorporating new modules as those are developed. The design details will be treated in the next section.

THE DESIGN

Celsius Bloodhound is constituted by 3 general modules that perform the main tasks. Each one will be constituted by smaller modules, that will perform a specific task in order to reduce the coupling between modules. This kind of design offers advantages, like the possibility of implementing each module in different computers in order to perform parallel tasks, or the high flexibility which will prevent an error or expansion of a specific function from affecting the rest of the application. The three main modules are:

- Input/Output
- Kernel or Core
- Connection to catalogues

INPUT/OUTPUT MODULE

Objectives.

- Receiving requests, accepting or rejecting them.
- Communicating with the Core module, sending the user requests in a format, or using a protocol understood by this module.
- Receiving the results sent by the Core module, and returning them to whoever made the request.

Description.

This is the module that receives the information requested to be searched by the end user or clients (we will call them just Clients from now on). Because this information might be sent from an application, a website or even a remote server, a standard way of communication easy to implement must be considered (from the BH side and the Client side as well).

Once the request of search has been received, this module must – as first measure – validate the Client, which implies:

1. an existing and enabled user
2. that has not reached the maximum number of searches allowed
3. the amount of requests pending and currently in process in acceptable (the server is no overloaded)

The first point (1) implies that the input module must contain some mechanism to validate and authenticate users, and a mechanism to enable and disable existing users. The second point (2) points out that there will be, as previously mentioned, statistics of access and use of the search engine by the users. In the same way, every user will have a maximum amount of allowed searches in a determined period of time, and once reached no more searches will be

allowed for this Client. This will allow a control to avoid abusive use, and it will make possible to create personalized agreements with future users. The third point (3) indicates that the input module will keep a queue list of pending requests, being able to know the condition of the requests in process of the Core module, to identify high loads of the system and to avoid overloads.

Points 1 and 2 indicate the need of a sub-module of user management, in charge of making the common management tasks and validating and enabling/disabling the use of CB, but also of keeping statistics, counters and allowed limits.

Point 3 indicates that the application will have a sub-module dedicated to managing queue lists. This queues will allow to register entries, departs and priorities; to know the state of a request, and finally to generate statistics. It is important here to emphasize that, because of the nature of this application, the queue management will be present in several modules which indicates the need of a module that is generic enough to be used from any other module.

Finally, the I/O module will work as an interface between the users and the main search engine, taking the search requests and sending them to the Core module, everything in a transparent way to the users. This actions will require parsing of the requests, analyzing the data (authentication, verification, optimization, among others) and finally sending it to the Core.

MAIN OR CORE MODULE

Objectives.

- Accessing the information in the catalogues, generating and administrating the objects that will make the remote connections.
- To balance the load between entering, attention pending and attended requests.
- To parse the information received from the catalogues and deliver it to the I/O module in a previously agreed format (standard).

The Core module works as a link between the I/O module (that represents the final users of the application for the core) and the Connections module, receiving data from the first and accessing the second in an organized and concurrent way. For that matter, once the data has been received –already parsed and verified – from the I/O module, Core makes up (or creates an instance of) the objects that will make the remote connections with the catalogues; delivers to them the corresponding information, and waits for the search results. Since many of these objects can exist for every search request received from I/O, some mechanisms of instantiation and parallel/concurrent execution must be considered, if allowed by the hardware, in order to avoid interferences between the different requests and to minimize the search time for each request. Then, if a search takes more time or if a remote server falls down, the rest of the searches won't be affected and will continue without problem.

As it receives the results, the Core module stores them in an own data structure and disposes the connection object no longer in use. Once it has obtained results from every object (timeout, empty, records set, etc.), it organizes them and sends them to the Output module. Once this is done, the search is "ended", which will release the allocated resources and will attempt to produce new searches.

This module also communicates with the Catalogue module, which is in charge of keeping the information related to the catalogues, information used to make the connections (used technology, IP address or URL, port, etc.).

CONNECTION MODULE

Objectives.

- Connecting with remote servers and receiving data from them.
- Delivering search results to the Core module
- Handling typical errors such as timeout, down servers, incorrect formats, among others.

This module is in charge of making the remote connections, sending the data and receiving the results. It is composed by many modules, each one of which allows to make connections with a type of server. Therefore, there will be a sub-module to make connections Z39.50[4], another for connections SRU[5], SRW [6], and so on for every possible technology.

These sub-modules can be thought as specific drivers for particular technologies. Each one of this drivers has three main functions:

- Generating data packages to send: With the information received from the Core, formatting the data according to the definition in the technology in question. This could be generating an URL address, a XML, a command and its parameters.
- Connecting with the catalogue: it will start the remote connection and send already formatted data. Again, the connection could imply references to URLs, sending XML via SOAP, starting a Z39.50 connection and sending data one by one, etc. Once the connection is ended it will have to close the connections and eventually, die.
- Parsing the results: once the catalogue has answered the registry, it will have to parse them to send them to the Core module.

The design of the Connection module allows the rest of the application to remain independent of the types of technology used, of possible network problems, down servers, etc. In fact, a problem in the local network could pass unnoticed for the rest of the application, since the only module directly affected will be the Connection module. In addition to this, this type of design will allow to incorporate new connection forms based in new technologies, without having to modify the rest of the application.

CONCLUSION

The search of documents is actually an extremely complex task, slow and repetitive; the design of an application that allows to speed up this task will result in a great benefit for a big number of services (such as the Library Linkage Project) or final users that constantly require the access to scientific publications.

Although the application won't always find the searched documents, or will find too generic information – as for example magazine titles but not its existences inside the catalogue– it will allow to extremely reduce the universe of library catalogues where the searches must be performed; and therefore, professors, students and researchers will be able to invest more time with there teaching, studying and investigating tasks Finally, an application of this kind will allow to strengthen ties between libraries, universities and institutions dedicated to research, and to promote the sharing of materials and collaborative work between them.

REFERENCES

[1] Proyecto de Enlace de Bibliotecas - PrEBi. Web Site: http://www.prebi.unlp.edu.ar

[2] Universidad Nacional de La Plata – UNLP. Web Site: http://www.unlp.edu.ar

[3] Ibero American Science and Technology Education Consortium – ISTEC. Web Site: http://www.istec.org

[4] Z39.50 Protocol http://www.bcl.jcyl.es/zeta/

[5] Search Retrieve via URL – SRU. Web site: http://www.loc.gov/standards/sru/

[6] Search / Retrieve Web Service - SRW . Web Site: http://www.loc.gov/standards/sru/srw/index.html