02-DJ

ISCIS

Proceedings of

The 17th International Symposium on Computer and Information Sciences

Edited by

Ilyas Cicekli • Nihan Kesim Cicekli • Erol Gelenbe

ISCIS XVII
October 28-30, 2002
Orlando, Florida USA

Organized by
Computer Science Program
School of Electrical Engineering and Computer Science
University of Central Florida

CRC

CRC PRESS

Boca Raton London New York Washington, D.C.

Dispatching Java agents to user for data extraction from third party web sites

Dmitriy Beryoza, Naphtali Rishe, Andrei Selivonenko, Alejandro Roque, Ian De Felipe
High-performance Database Research Center
School of Computer Science
Florida International University
University Park, Miami, FL 33199, USA
+1 (305) 348-1706
{beryozad, rishen, selivona, aroque03, idefel01}@cs.fiu.edu

Abstract

Data retrieval on the World Wide Web has been a focus of intensive research in the past few years. The majority of existing approaches concentrate on mechanisms for data discovery and schema induction. Few researchers, however, address issues of performance of data extraction systems. Thus, centralized, server-based data extraction solutions often suffer from congestion and low speed of execution. In this work we present a mechanism for enhancing performance of data retrieval through distributing its functionality to the client computer using Mobile Data Retrieval Agents (MDRA).

1 Introduction

The amounts of data accessible on the World Wide Web have exploded in the recent years. Unfortunately this increase was not followed by significant improvement in mechanisms for accessing and manipulating this data. It is still accessed by browsing Web pages, entering information in query forms and *reading* the results that Web sites present. No convenient mechanism exist that would give user more power over the data on the Web, by, for example, allowing her to define custom queries to Web sites or to extract the returned data from HTML pages and use it in external applications.

Querying the Web and data extraction on the Web has recently become a popular research topic. A variety of methods for schema discovery and data extraction from HTML documents have been proposed. We have designed Data Extractor system for Web data retrieval [3] that uses wrappers written in Java for posing queries to sites and extracting resulting data sets. Data Extractor allows us to treat virtually any Web site as a data source. It is implemented both as a standalone server solution and a set of functionality that can be embedded in applications and provide them with live data from the Internet. This system is also used as a Web data provider for MSemODB heterogeneous database system ([7], [8]).

Data Extractor has several inherent inefficiencies:

- *Performance in multi-client conditions.* Data Extractor was designed to be primarily server-based system, with the data extraction functionality executed in a central location. As the number of users (especially remote users) grows, the system may become overloaded with requests.
- *Network performance issues.* Data Extractor server works as an intermediary between data consumer and the Web site that is a source of data. This means that after it is extracted from the Web site, data always has to go through server first instead of coming directly to consumer. In addition to being a longer delivery route this can present problems if Data Extractor server is located in a low-bandwidth or highly-congested network segment.
- *Legal issues.* In rare cases Data Extractor server maintainers may be prevented by law from extracting data from certain sites on behalf of the client, as such extraction may constitute a copyright violation. In these same cases giving user ability to extract data directly, without the services of a middleman, may be legal.

Installing local server for exclusive use of a small number of clients may be one solution for these problems, but costs and complexity associated with such an operation could be high. In this work we are defining an alternative - performing data extraction on the client side through *Mobile Data Retrieval Agents (MDRA).*

2 Architecture

The idea of MDRA is in distributing the data extraction functionality to the client computer, close to consumer of extracted data. MDRA approach is different from shipping the complete data extraction functionality

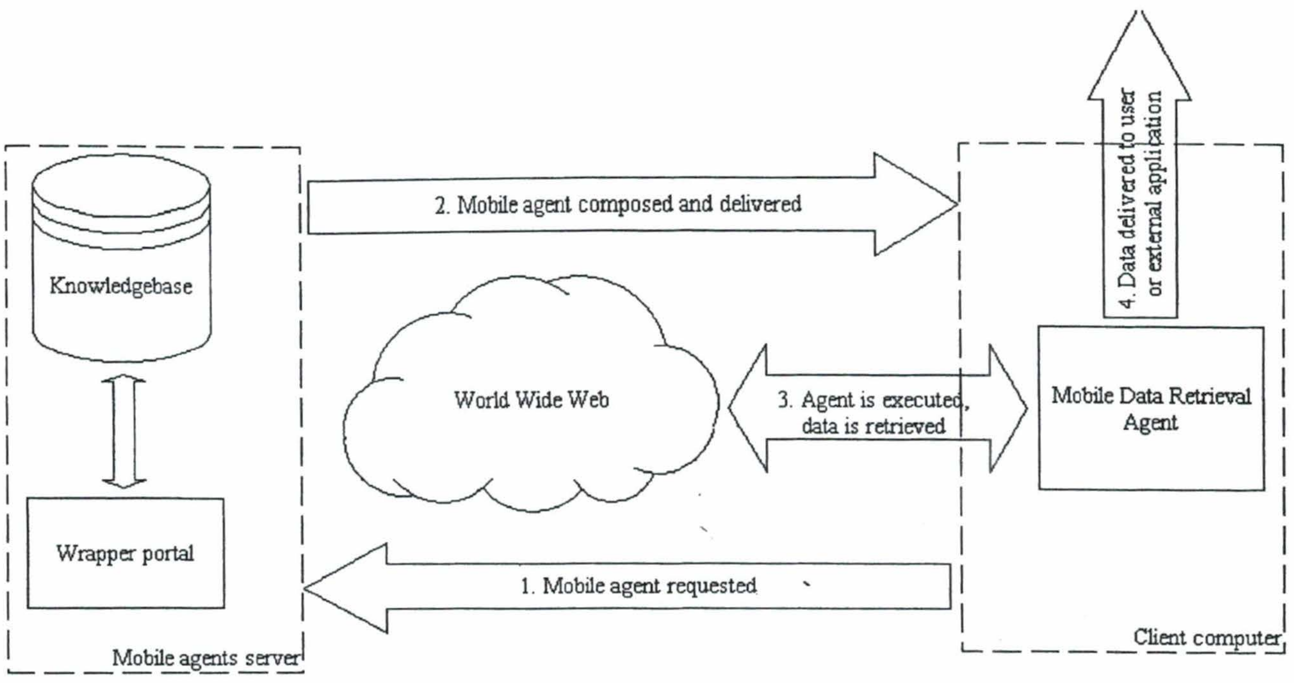
Knowledgebase
Wrapper portal
Mobile agents server
2. Mobile agent composed and delivered
World Wide Web
3. Agent is executed, data is retrieved
1. Mobile agent requested
4. Data delivered to user or external application
Mobile Data Retrieval Agent
Client computer

Figure 1 MDRA composition, delivery and execution

to the client side, because agent composition and maintenance mechanisms remain on the server and are managed centrally.

The server, called *mobile agents server,* hosts *wrapper portal* and a *knowledgebase* (see Figure 1). *Wrapper portal* is a Web-based catalog that allows users to select and execute wrappers. Users who subscribe to the MDRA service connect to wrapper portal and request wrapper or application to be executed on the client computer. In response to that request a package containing functionality necessary to perform data extraction for a particular Web source is constructed and shipped to the client computer. It will then be executed there and extract data for the user. Aside from listing and packaging wrappers, portal authenticates users, allows them to change and save their preferences, and save and retrieve previously created *queries* (references to wrappers together with wrapper parameters).

Knowledgebase used in MDRA server system contains information about available wrappers, their parameters and status. It may also contain information required by the wrapper portal. For example, it could store user account information, such as access privileges and preferences. Names and execution parameters of wrappers that users have run so far can also be stored in this database. Using this information, wrappers can be executed with the same settings on a regular basis. Finally, lightweight applications that use wrapper output or act as intermediaries between wrapper and applications on client computers can be stored in the knowledgebase, together with necessary composition and parameter information.

Architecture of agents generated and packaged by mobile agents server is based on the architecture of Data Extractor system. The internal structure of a Mobile Data Retrival Agent is shown in Figure 2. It consists of the following components:

- *Mobile wrapper controller.* Wrapper controller is responsible for controlling the behavior of wrappers, passing parameters to them and directing the flow of data from them. In this sense it is very similar to wrapper controller used in Data Extractor system, but, perhaps, optimized for shipment to client computer and execution there.
- *Wrappers.* The same wrappers are used in the Data Extractor and MDRA implementations. They will be created and stored on the server and managed centrally for all users of MDRA service. This significantly simplifies service maintenance, ensures correct operation and makes timely updates available to all users of the system.
- *Data Extraction Library.* Data Extraction Library contains functionality that is essential for performing data extraction and networking operations and has to be shipped with every MDRA. Our implementation of it is very compact and will be transmitted to the client computer quickly even on slow links.

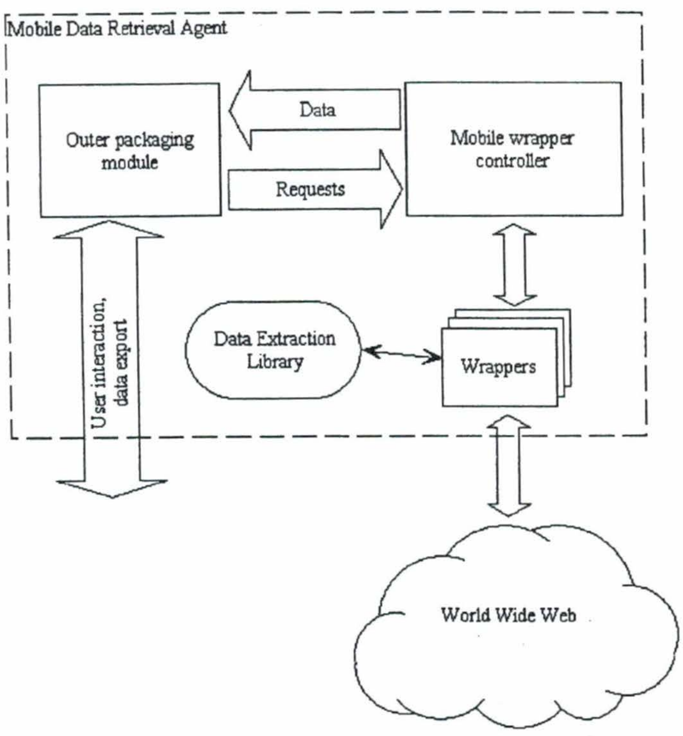
Mobile Data Retrieval Agent
Outer packaging module
Data
Requests
Mobile wrapper controller
User interaction, data export
Data Extraction Library
Wrappers
World Wide Web

Figure 2 Architecture of Mobile Data Retrieval Agent

- *Outer packaging.* Outer packaging component is a module that unites all other modules in the MDRA. It can be implemented as a Java applet, an application, a browser plugin or take some other form. The job of this component is to communicate user commands to the wrapper controller and receive results generated by the wrapper.

  Packaging component can be implemented to take on one of the following functions:
  - ♦ *Browser.* Browser works as a flexible display tool. It displays data that it receives in tabular form, akin to a mini-spreadsheet. Columns can be adjusted, collapsed and sorted. Data can be edited, searched, copied, printed and exported. This mode is useful for browsing and modifying data generated by the wrapper.
  - ♦ *Exporter.* This type of packaging is useful for non-interactive mode of operation. It can be configured to automatically generate on user's computer a data file that will contain data output by the wrapper. Data files can be in a variety of formats – plain ASCII, Comma Separated Values, Microsoft Excel, XML and others.
  - ♦ *Wrapper-based application.* Lightweight applications can be developed to perform simple operations on data generated by wrappers. Such applications can work interactively with user, executing wrappers based on user input and performing complex operations on the data received from wrappers.
  - ♦ *Connector.* This type of packaging is useful in cases when data received from the Web has to be exported into applications running on user's computer. We can write connectors that populate tables in DBMSs or import live data into analytical or financial packages.

3 Composition and execution of agents

3.1 Query formulation

User interaction with the system (Figure 1) begins with connecting to the wrapper portal. Wrapper portal lists available wrappers and packaging configurations that user can run on her computer. This information along with wrappers and applications is stored in the system knowledgebase, which is continuously update by server maintainers.

When the necessary configuration and packaging is selected, user optionally can specify execution parameters and save this configuration for future reference. In some cases additional information may be required from user. This information may include usernames, passwords or credit card information for wrappers that access pay-per-use sites. After all necessary information has been collected from the user, she may ask the system to build, deliver and execute the agent.

3.2 Agent construction and delivery

Once the wrapper portal receives the request for an agent it begins packaging it. Several components, including outer packaging module, wrapper parameter information, wrapper controller, wrapper, and Data Extraction Library can be combined in a single package for delivery to client computer. Optionally, components that change frequently (such as wrappers and their parameters) can be packaged separately from the ones that do not change often. With separate packaging the part that does not change may be cached on the client computer. Depending on particular implementation, the package can be compressed and/or digitally signed. When packaging components, special attention must be paid to keeping agent as compact and platform-independent as possible. Once the package is ready for delivery it is sent to the user computer.

3.3 Agent execution

When the agent is delivered to the client computer it is executed based on parameters supplied to it. Parameters can be specified at the portal or through dialogue with the user. Outer packaging component handles the dialogue with the user and controls wrapper execution through commands to wrapper controller. Wrappers interact with the Web sites, extract data and pass it to the outer packaging component module through the controller. Overall agent execution, including stopping and restarting, is controlled through its user interface.

3.4 Data delivery

When the data is retrieved from the Web it can be returned in several forms: it can be fed into other applications, displayed to the user or exported to the file system.

4 Implementation

The prototype of the MDRA system is currently being implemented in Java programming language. Java was chosen for a variety of reasons. Because the mobile agents are based on existing implementation of Data Extractor and Data Extraction Library, which are implemented in Java, compatibility was important, as was reuse of the existing modules. Portability was also an important consideration because MDRA code is shipped to the client side and executed there, and thus has to be supported with no or little modifications on a variety of platforms.

There are, of course, concerns about Java performance, as it is slower than its compiled counterparts, such as C++. Some of the performance problems did indeed manifest themselves at the early stages of implementation of Data Extractor system and Data Extraction Library functionality. These problems primarily appeared when the load on the Data Extractor system increased dramatically because of multiple connected clients. Most of these problems were identified and resolved following techniques described in [6]. In MDRA framework agent execution will be dedicated to a single client and as a result we do not expect any noticeable performance degradation.

4.1 Framework

For MDRA technology to be easy to use it has to be user-friendly. This starts with installation procedures. For the majority of computer users downloading software from the Web site and installing it on their computer is unattractive. There are many reasons for that: the process of software downloading and installation is often inconvenient and confusing; user may be afraid of viruses or have insufficient permissions to install software on her computer.

One of the easiest ways to ship MDRA functionality to the user's computer and execute it there is through a Java applet. Because it starts automatically and integrates with browsers well, even inexperienced users will be able to use it. When user requests the agent the browser will receive a Java applet packaged with all the necessary system components and libraries. The applet will then execute in browser context, querying the Web and supplying data to the user.

Other options for packaging MDRA technology may include ActiveX controls, Netscape plugins, and, in absence of alternatives, downloadable applications. These options, however, are associated with a set of problems, such as limited portability, size, access restrictions and others.

4.2 Performance

In preliminary tests MDRA agents have shown better performance compared to that of Data Extractor implementation, which was expected as the data extraction is done at client site without unnecessary trips to server and server is no longer the performance and bandwidth bottleneck. On average, the time it took to package an agent and extract data from Web pages was negligible in comparison to the time it took to download pages from Web sites. Download times varied significantly depending on the type of link the client computer had, the saturation of the network segment in which it was located, and the responsiveness of the Web site being queried for data.

4.3 Security

One of the strengths of Java applets—security—becomes a challenge in the context of mobile agents. Browsers prohibit Java applets from accessing system on user computer in order to prevent attacks from maliciously written applets.

MDRA, however, need to access system resources and perform other actions that applets are normally prohibited from doing. These actions include accessing network resources (to extract data) and file system and system resources (to save data on the system or feed it into other applications on the system).

A partial solution to this problem is to create a proxy application on the server where the applet came from (applets are allowed to communicate with the home server). Through such proxy (whose role can be played by a standard HTTP proxy server) the applet would be able to download pages from the third party Web sites. This approach, however, does not solve the problem of data export—applet will still be prohibited from exporting the data that it extracts to anywhere on the user's computer. Also, the proxy application will become a bottleneck that will affect system performance in high-volume data extraction applications.

Another solution—*applet signing*—appears to be more feasible. Applet signing allows developer to "sign" the applet with a digital certificate purchased from a certification authority. Signed applets are allowed certain degree of freedom inside the browser. Users can grant them permission to access network resources or the file system. To take advantage of this functionality applet that contains MDRA will be signed before delivery to the client computer, and the code that requests permissions from browser will be inserted in applet initialization routines.

5 Related work

The idea of employing agents for data extraction has been discussed by numerous researchers ([4], [2], [1], [5]). A system similar to MDRA is described in [4]. The mobile agents' framework discussed there uses an approach different from MDRA - it introduces a cooperating set of agents that browse information servers and execute short queries to sites. In MDRA agents do not cooperate but the emphasis is placed on accuracy of extracted information and performance while serving needs of a single user, which we feel is only achievable through use of custom-crafted agents. MDRA also solves a wider variety of problems, allowing us to build Web data-aware applications and extract high volumes of data in a variety of formats.

Mechanisms for performing data extraction at the client site can also be found in business applications, such as downloadable personal shopping bots from GoTo Shopping (http://shop.goto.com/) and R U Sure (http://www.rusure.com). User downloads and installs an application associated with one of these services on her computer and then can use it to perform price comparison for a variety of products by querying Web sites in real time. Up-to-date instructions on how to get prices from a variety of online stores are periodically comparison downloaded by the shopping application from the central location. Technologies used in these services are proprietary and cannot be evaluated to discuss their merits and disadvantages. MDRA is an open platform that is not geared only towards comparison but will support any application that requires data extraction from the Web.

6 Conclusion

We have presented a Mobile Data Retrieval Agents framework, which is an application of Data Extractor Web data retrieval technology. MDRA "lease" data extraction services to users. Using it users modify, download and execute queries to Web sites, and receive extracted datasets. The extracted data can then be imported into other applications and further analyzed and processed. Mobile agents significantly increase performance of the Web data extraction mechanism by moving it off the centralized server and to client site. This distributed approach allows us to free-up server resources and increase overall system performance.

As part of our future research effort we intend to experiment with different implementations of MDRA, both standalone and embedded inside client-side applications. We will also work on improving the performance of client-side agent execution.

7 Acknowledgements

This research was supported in part by NASA (under grants NAG5-9478, NAGW-4080, NAG5-5095, NAS5-97222, and NAG5-6830), NSF (CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409), ONR (N00014-99-1-0952), the Department of Interior, and the FSGC.

8 References

[1] Bauer, M. and Dengler, D. TrIAs: trainable information assistants for cooperative problem solving. *Proceedings of the 3rd Annual Conference on Autonomous Agents*, 1999, Pages 260 - 267

[2] Bauer, M., Dengler, D. and Paul, G. Instructible information agents for Web mining. *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, 2000, Pages 21 - 28

[3] Beryoza, D., Rishe, N., Graham, S., De Felipe, I. Data Extractor Wrapper System. *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics*, 2002, Vol. VII, Pages 425-431

[4] Dharap, C. and Freeman, M. Information agents for automated browsing. *Proceedings of the 5th International Conference on Information and Knowledge Management*, 1996, Pages 296 - 305

[5] Frank, M., Szekely, P. Collapsible user interfaces for information retrieval agents. *Proceedings of the 1999 International Conference on Intelligent User Interfaces*, 1999, Pages 15 - 22

[6] Heydon, A., and Najork, M. Performance Limitations of the Java Core Libraries. In *Proceedings of the 1999 ACM Java Grande Conference*, pages 35-41, June, 1999.

[7] Rishe, N., Athauda, R., Yuan, J. and Chen, S.C. Knowledge Management for Database Interoperability. *International Conference on Information Reuse and Integration*, Honolulu, Hawaii, November 1 - 3, 2000.

[8] Rishe, N., Yuan, J., Athauda, R., Lu, X., Ma, X., Vaschillo, A., Shaposhnikov, A., Vasilevsky, D. and Chen, S.C. SemanticAccess: Semantic Interface for Querying Databases. *The International Conference on Very Large Data Bases (VLDB 2000)*, September 10-14, 2000.

1490
ISBN 0-8493-1490-9
90000
9 780849 314902