

# Tackling Web dynamics by programmable proxies

Delfina Malandrino \*, Vittorio Scarano

*ISISLab, Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno, Via S. Allende, 84081 Baronissi (SA), Italy*

Available online 20 December 2005

---

## Abstract

Web services are becoming increasingly complex as users become more experienced in their requests to access an ever growing collection of information on the Web. In this paper we review the state of the art in programmable HTTP proxies.

We discuss and present the evolution of HTTP proxies and present some of the environments that were proposed to support the design and implementation of proxy-based services. Then, we present a new computational model, called *collateral-push*, that is well suited for several proxy applications whose goal is to cope with the dynamics of the Web. Finally, after presenting existing examples of the collateral-push services in the literature, we present some new examples that were quickly prototyped using two programmable proxy environments.

© 2005 Elsevier B.V. All rights reserved.

*Keywords:* Web intermediaries; Dynamic Web content; Push/pull paradigms; HTTP proxies

---

## 1. Introduction

The World Wide Web, as a ubiquitous environment for finding information, exchanging ideas and transacting business, is now globally recognized as the universe of information accessible from any networked computer or device. Due to its simplicity, scalability, ubiquity and visibility it is considered the ideal testbed to perform distributed computation and to deploy complex distributed applications.

On the other hand, its explosive success, the growing amount of information available on Internet, the highly dynamic nature of these documents and, finally, the advent of pervasive computing, have changed the way the Internet services are

offered to end users. In fact, the classical client/server model of the Web is considered not fully suitable any more for the provision of dynamic and personalized applications. Contents from multiple servers, geographically distributed in the network, for example, can be retrieved, manipulated, assembled in single Web pages, and finally delivered to end users, that, on the other hand, can access them by using different terminal devices, with different software technologies and network communications features. In addition another important challenge comes from the HTTP protocol, a stateless protocol, originally designed for document retrieving and not for interactive applications.

Furthermore, to ensure an efficient provision of such complex services other challenges must be addressed: (a) scalability, that is, network services that incrementally support growth in both number of users and services, (b) accessibility, that is,

---

\* Corresponding author.

*E-mail addresses:* [delmal@dia.unisa.it](mailto:delmal@dia.unisa.it) (D. Malandrino), [vitsca@dia.unisa.it](mailto:vitsca@dia.unisa.it) (V. Scarano).

applications available on demand according to the client devices' requirements and their system resources (CPU, memory, storage, etc.), (c) high availability and robustness, that is, services which exhibit a  $24 \times 7$  availability even through interruptions and failures, and finally (d) cost-effectiveness, that is, services developed by leveraging existing software systems, to save the cost of re-implementing part of the software.

In this paper, we present our thesis that programmable proxies (i.e., software infrastructures whose functionalities can be easily developed and changed) can be a consistent aid in coping with the dynamic nature of the Web, and tackling the challenges that come from the ever growing need for advanced services from a mature audience of users as well as the huge corpus of information that is daily accessed by anyone using a Web browser. We support our thesis by presenting a proxy-based computational model, named *collateral-push*, that merges the traditional push and pull paradigms. In this model, the proxy builds the response to the client by intertwining the response by the HTTP server with the additional information flow that has been pushed toward it or with information that it has requested elsewhere. The result is, then, provided to the client as a response to its (pull) HTTP request.

Several instances of collateral-push proxy applications can be found in the literature for Web Dynamics and we report on them in Section 3.2. As an example, consider the service News Alert that we developed (explained in detail in Section 3.3.3): the goal is to provide a push-based information stream of news into the HTML pages that are traditionally requested with the pull model to an HTTP server. Therefore, each page requested by the user is modified so that the topmost part contains links to recent news that are relevant to the user's interests (as personally configured by the user). News are, therefore, collaterally pushed toward the client with the same update frequency as the Web access rate of the user. No modification is needed on the client side; the pushing is unobtrusive (the user is only informed when actively navigating the Web) and the additional information flow is piggy-backed on top of HTTP responses, thereby saving bandwidth toward the client.

We present in Section 2 of the paper how the role of an HTTP proxy has evolved from the very beginning of the World Wide Web until recently. The description of some programmable proxies shows how the research is aiming to offer increasingly

sophisticated programming environments to offer a suitable setting to facilitate the development of advanced intermediary services.

In Section 3, we first provide some considerations on the nature of the computations provided by HTTP proxies that help us to present the *collateral-push* model, whose goal is to merge effectively the advantages of server push and client pull models in proxy-based applications. In order to further support our thesis and the introduction of this new paradigm for proxy-based applications, we also present some evidence of several collateral-push solutions existing in literature in different fields that deal with Web Dynamics and, finally, briefly describe our implementation of four new collateral-push applications whose target is to help user navigation by providing support and/or advanced services in the face of the dynamics of the Web.

Finally, in Section 4, final remarks and comments conclude the paper.

## 2. The role of programmable HTTP intermediaries

In this section we present HTTP programmable proxies, i.e., the tools we advocate as an ideal framework for tackling a relevant part of the problems that are generated by the dynamic nature of the Web. We begin, first, with a discussion on the evolution of HTTP proxies in the last decade and, then, proceed with a review of some HTTP programmable proxies that are available currently. Each tool is briefly introduced and some of its characteristics are described.

### 2.1. HTTP proxies and edge services

An HTTP proxy is a server that sits in-between an HTTP client application (browser) and an HTTP server, thereby being able to intercept all the requests and responses that are exchanged. Proxies were present since the very beginning of the World Wide Web architecture specifications (HTTP 1.0) and their main intended purposes were to improve performance, by offering caching mechanisms, and to filter/log requests so that they play the role of a single point of access to the Web for corporate intranets.

Proxy roles have evolved during the past 10 years of World Wide Web. At the beginning, the emphasis was on their performance in order to make an optimized usage of bandwidth and, at the same time, offer an improvement on request latency at the

client-side. Then, the evolution of firewalls extended proxies with more network-oriented services and offered an all-encompassing product both to large corporate intranets as well as to SOHO (Small-Office Home-Office) environments.

Of course, the evolution of proxies is, now, leading toward more advanced services that are provided to users in order to cope with the Web's ever-evolving structure and contents as well as with the maturity of the Web users that call for *cooperative, ubiquitous* and *personalized* access to a variety of dynamic information.

Cooperatively navigating on the Web enables social activities so that users can take advantage of other people's activity, to get recommendations based on others' opinions. In this way, Web navigation is used as a kind of communication channel promoting knowledge and information exchange among user communities.

The ubiquity requirement is mainly due to the emerging of wireless technologies that can offer access to the Web *anytime and anywhere* by using terminal devices (cell phones, PDAs, digital cameras, digital pagers, portable computers, etc.) with different system capabilities (in terms of computational power, graphical display size, battery power), software capabilities and bandwidth connectivity.

As the Internet continues to evolve with an increasing diversity and heterogeneity, there is a growing demand for solutions that enable universal access to Web content. Such universal access requires mechanisms for personalizing and adapting Web contents to different client devices' capabilities, communication systems, network conditions and, finally, user preferences/abilities.

So far, the dominant paradigm of communication on the WWW, because of its simple request/response model, cannot efficiently address such requirements, and, consequentially, it has to be necessarily augmented with new components that attempt to enhance the scalability, the performance and the ubiquity of the Web. HTTP proxies, acting on the HTTP data flow exchanged between client and server, allow content adaptation as well as other complex functionalities beyond the traditional caching and content-replication services.

These value-added services are called *edge services* and include personalization and customization, aggregation from multiple sources, geographical personalization of the navigation of pages (with insertion/emphasis of content that can be related to user geographical location), translation services,

group navigation and awareness for social navigation, advanced services for bandwidth optimization such as adaptive compression and format transcoding, mobility and ubiquitous access to the Internet content.

The main advantages of proxy servers are that they can be transparently deployed without involving hardware and/or software changes on client and server systems, they can reduce complexity on servers (that will only take care to provide the requested resources) as well as moderate system requirements on clients (independently from devices' capabilities), add new services without stopping the servers, by ensuring their fault-tolerance, offer increased flexibility (where new components can be deployed) and scalability, and, finally, improve the quality of access to the resources available on the Web.

## 2.2. Programmable proxy infrastructures

A *programmable proxy* is a software infrastructure whose provided functionalities can be quickly and easily changed as well as entirely developed from scratch. By leveraging on an execution environment and by exploiting a specific programming model, a programmable proxy offers quick prototyping and easy deployment of new system core functionalities and application services into the intermediary infrastructure.

In order to accomplish the programmability, several important requirements of the framework include: *configurability*, to customize services according to users' preferences and devices' capabilities, *life-cycle support* of the applications, by allowing the automatic deployment of new functionalities (without dealing directly with the host software infrastructure), and the *maintainability* of the software, that is, the availability of the developed software (source code) in order to ensure a quick adaptation of the software to the demands for new advanced services.

We next describe some existing programmable proxy environments that are commonly used on the Web. An extensive survey and comparison of programmable proxies can be found in [30] where a taxonomy is also provided. In this section, we will focus mainly on the two characteristics of programmability and configurability (i.e., the requirement of configuring the set of services that can be invoked during Web transactions, a simple configuration for enabling and disabling services and a more

detailed configuration realized through services' parameters). Of course, we also describe, for each environment, the set of services that are provided with the proxy itself, whose goal is to address the basic functionalities of an HTTP proxy (such as filtering, transcoding, personalization, caching, etc.).

### 2.2.1. RabbIT

RabbIT [<http://rabbit-proxy.sourceforge.net>] is a Web HTTP proxy that accelerates the delivery of Web contents to end users by compressing text pages and images, by removing unnecessary parts of HTML pages (background images, advertisements, banners, etc.) and, finally, by caching filtered documents before forwarding them to the clients.

The RabbIT proxy is written in Java, developed and tested under Solaris and Linux, and has a modular architecture that allows an easy definition and implementation of new functionalities. It is highly programmable since new filters can be developed by using a set of APIs provided with the distribution. It is also highly configurable: new modules, in fact, can be configured at runtime (no parameter configuration is provided) by simply accessing a specific URL, but the new services cannot be dynamically reloaded since the proxy must be restarted. It provides some examples of configuration files as a starting point to develop new configurations.

Each service is implemented as a Java class, that derives from some superclasses from RabbIT. Methods have to be extended that deal with HTTP requests and responses. Some utility APIs are also provided (e.g., HTML parsing).

Several filters are provided with the RabbIT distribution such as, for example, an authentication service for access control, a service that allows RabbIT to work as a reverse proxy (i.e., proxy that is placed close to the origin server [24]), or simple filtering services (for removing page background, ad banners, blink and /blink tags, etc.).

In spite of its ease of programming, deploying services is not intuitive and rather elaborate, since it requires careful editing of the configuration file that is time-consuming and error-prone.

### 2.2.2. Webcleaner

WebCleaner [<http://webcleaner.sourceforge.net>] is a filtering HTTP proxy that provides functionalities for removing advertisements, banners, Macromedia Flash and Java-script code, for reducing image (by size) and compressing HTML pages (with gzip), etc. It removes, adds and modifies HTTP

headers, supports mechanisms for user authorization, provides modules for security and virus detection, etc.

The WebCleaner proxy, developed in C and Python and available both under Linux and Windows platforms, is programmable and highly configurable: new services can be added to the system through a graphical user interface by simply specifying appropriate configuration parameters. All information about the deployed service, implemented as a Python script, are stored in the proxy configuration XML file. The end user is not forced to modify this file, since both proxy settings and services' configuration information can be provided through the GUI.

Several filters have been provided with the distribution, and in particular, services for HTML filtering, for image reducing, for Web pages blocking, etc.

### 2.2.3. Privoxy Web proxy

Privoxy [<http://www.privoxy.org>] is a Web proxy with advanced filtering capabilities for protecting privacy, modifying Web page content, controlling access, and removing advertisements, banners, pop-ups, etc.

Privoxy, based on Junkbuster [<http://internet.junkbuster.com/ijb.html>], is an Open Source project licensed under GPL. It is implemented in C and available for different platforms (Unix/Linux, Windows, Mac OSX, etc.).

Privoxy is programmable and highly configurable, and services can be customized according to individual user needs. Specifically, Privoxy can be configured with various configuration files, accessed through specific URLs, intercepted by the proxy. The environment provides a set of API and a detailed documentation on how to develop and deploy new services.

The functionalities provided by Privoxy services are specified through "Actions" that define what type of action Privoxy will take for a specific URL, and thus how to handle images, scripts, banners and other elements embedded in the Web pages of the handled Web transaction. These "Actions" can be defined in configuration files and in particular, a default file is provided (i.e., *default.action*) to set the initial values for all actions, while the default profiles and their associated actions are predefined in the *standard.action* configuration file.

Different filters have been provided with the distribution and, in particular, services for ad-filtering

by link and by size, ad-blocking by URL, GIF de-animation, etc.

#### 2.2.4. Web based intermediaries

Web Based Intermediaries (WBI) [4–6] is a dynamic and programmable framework, developed in Java at IBM Almaden Research Center [<http://www.almaden.ibm.com/cs/wbi>], whose main goal is to personalize the Web by realizing an architecture that simplifies the development of intermediary applications. WBI defines a programming model that can be used to implement all forms of intermediaries, from simple server functions to complex distributed applications. WBI is freely usable but its source code is not publicly available.

A WBI transaction is defined as a complete HTTP request–response and flows through a combination of four types of basic stages, called MEGs: *RequestEditor* and *Editor* MEGs, that modify HTTP requests and responses respectively, the *Generator* MEG produces the response to an HTTP request and the *Monitor* MEG that monitors HTTP requests and responses without any modification. A group of such MEGs is called a WBI plugin.

WBI provides a rule-based system for dynamically determining the data path through the various MEGs allowing complex boolean expressions. With WBI, rules can be dynamically altered to add or delete MEGs, or to change their triggering conditions. Finally, the WBI GUI provides a convenient way to manage and administer WBI, and to help users debug the plugins loaded into the proxy.

#### 2.2.5. Scalable Intermediary Software Infrastructure

The main goal of the Scalable Intermediary Software Infrastructure (SISI) project [19] is to realize a framework for easy and efficient deployment of advanced services implemented by an intermediary on the WWW. The SISI framework is developed on top of existing open-source, mainstream applications, such as Apache Web server and `mod_perl`. Apache is a high quality choice, since it represents one of the most successful open-source projects (if not the most successful) that delivers a stable, efficient and manageable software product to the community of Web users. The Apache module `mod_perl`, on the other hand, brings together the Apache and Perl software technologies by providing APIs to implement in Perl Apache modules that can be easily integrated into the server and that perform efficiently [9].

The SISI framework is programmable and highly configurable, and its modularity allows an easy definition of new functionalities implemented as building blocks in Perl. These building blocks, packaged into *Plugins*, produce transformations on the information stream as it flows through them. Moreover, they can be combined in order to provide complex functionalities. Technically, SISI services are implemented as Apache handlers by using `mod_perl`.

SISI consists of several modules and each of them acts in a specific phase of the Apache HTTP Request lifecycle. In particular, two of them, the *ProxyPerl* and the *FilterPlugin* modules encapsulate the intermediary functionalities by intercepting the HTTP client request and dispatching them to the requested service according to users' preferences; the *Authorization* module has the main goals of restricting access to the system and distinguishing between users in order to allow personalized services; and finally, the *Deploy module* allows the programmers to automatically create services without having detailed knowledge about Apache and `mod_perl`.

Several services have been provided and, in particular, services for cleaning HTML pages from unnecessary information, for removing Java script code, banners, advertisements, for image manipulation (scaling, monochrome images, etc.), for de-animating GIF images, for allowing Web accessibility, etc. SISI has been implemented under the Linux and Windows platforms.

Finally, SISI allows each user to define one or more personal profiles according to the capabilities of the increasingly heterogeneous client devices and users' preferences, knowledge or needs.

Other examples of proxy-based infrastructures include AT&T Mobile Network iMobile [31], a proxy-based platform designed to provide personalized mobile services. It provides a modular architecture that allows an easy definition of new functionalities implemented as building blocks in Java, on top of a programmable proxy, called iProxy. The BARWAN project [23] at UC/Berkeley has the goal to provide intermediary systems to support ubiquitous access to Internet services from mobile and thin clients. An important system component of this project is the programmable proxy architecture, TACC, that acts as intermediary between servers and mobile clients. The *extensible Retrieval Annotation and Caching Engine* (eRACE) [15] is a modular, programmable and distributed intermediary infrastructure whose main goal is to

provide personalized services for a wide range of client devices (Desktop PC, mobile and thin clients), such as personalization, customization, filtering, aggregation and transformation. Finally, MARCH [2] is a distributed content adaptation architecture that provides functionalities for adapting Web content according to the capabilities of client devices that access Internet services.

### 3. Collateral-push by proxies

In this section, we propose a computational model for proxies that merges the traditional push/pull paradigms in what we call collateral-push computation that retains most of the advantages of the server push model while keeping the ease of deployment that is typical of client pull model.

Then, we present several examples of existing applications in the area of Web dynamics [25] that adhere to the collateral-push paradigm. This evidence further supports our claim that programmable proxies represent an ideal setting for tackling a relevant part of Web dynamics. Finally, we show how crucial programmability is in proxies by exhibiting four new intermediary services that we have quickly prototyped to address new questions in this field.

#### 3.1. A paradigm shift: from push/pull to collateral-push

Two important features of Web intermediaries concern the communication mode, synchronous or asynchronous, and the access model, push or pull, between Web components.

Web intermediaries, for example transcoding proxies, perform their activities in a synchronous manner, and they receive a request and reply with the corresponding response while the user is still connected to the system. Moreover, other intermediaries can provide functionalities whose results can only be used later.

The pull model is based on the request/response paradigm where the client issues a request and the origin server answers either synchronously or asynchronously. In this paradigm the transaction is always initiated by the client. A classical example, obviously, is the World Wide Web architecture.

On the contrary, the push model is based on a publish/subscribe paradigm, where the origin server takes care of disseminating information efficiently without sending unnecessary information. For

example, publish/subscribe systems, thanks to their asynchronicity and persistent connections, represent an optimal solution for developing mobile applications that require dynamic reconfiguration after disconnections [11]. In such systems, proxy components can be employed to store notifications, published by servers, during disconnections, in order to deliver them to subscribers after their reconnection [35].

The interaction on the Web is, traditionally, pull-based since the client takes care of making the query and, then, initiates the transaction; if changes occur to Web page contents, it is user's responsibility, and not system's, to detect such modifications by constantly monitoring the content source and act accordingly. In other words, it is the client that has to be aware that changes may have occurred in the information it is processing.

The push-based model reverses this communication pattern by dynamically disseminating information to clients. The end user will not repeatedly ask for new data, she/he needs only to subscribe and, then, receive updated information as soon as it becomes available. Of course, traditional Web clients are not suited for receiving information flow that is pushed and, therefore, need to use enriched browsers that are equipped, for example, with Java applets, plugins, etc. that allow asynchronous notifications.

Push systems offer many advantages over the pull-based ones. First, they are able to always ensure freshness of data since users are notified as soon as the updated information becomes available. Secondly, users can provide their personal interests and preferences allowing the push systems to store, analyze and perform tasks according to such information. Finally, push-based systems are able to reduce network traffic, avoiding redundant client requests as in pull-based approaches. Clients contact each server once, and then there is no additional (other than the subscription) traffic from client to server.

Of course, in an already existing infrastructure such as the Web and its variety of browsers, a solution like adding capabilities to clients is almost infeasible<sup>1</sup> and, moreover, imprudent because of

<sup>1</sup> The deployment of new functionalities on browsers is expensive because of the diversity of browsers, OS versions, etc., and, furthermore, frequently requires tedious installations of additional software (virtual machines, libraries, etc.) that are often hesitantly undertaken by the average Internet user.

the security problems that, nowadays, afflict the Web and the Internet. Therefore, the task of constantly monitoring changes in the information of interest to the clients presents an architectural challenge since modifications to clients are, in practice, disallowed. As a consequence, proxies represent the ideal place where these tasks can be placed in a non-intrusive manner [8]: some experimental studies of mixing and interleaving of the two strategies (push/pull) did employ proxies [13] to avoid costly modifications to clients.

Here we present a model that merges the two paradigms push and pull in order to provide the capabilities of the push paradigm on an infrastructure that was designed for the simpler pull paradigm (see Fig. 1). In the *collateral-push* model, the HTTP proxy is able to push collaterally (or *c-push*) relevant information by means of adding to or modifying the HTML data that is transmitted to the client under each request/response (i.e., pull) cycle. In a sense, the information flow (to be pushed back to a client) is piggybacked onto the responses to its requests (performed according to the traditional pull model).

In our model, the modifications of the HTML to c-push data vary from simple addition of plain-text

information, to enriching the links (by modifying the behavior, or adding tooltips), to adding active content (script files, applet, etc.) or embedding multimedia files (such as audio) into the page.

An intermediary is individually configured by each user in order to specify the nature of the intermediary services that are requested and the value of the parameters for these services. The configuration of proxy services is performed by either intercepting special URLs that respond with forms that allow configuration management (see, e.g., Rabbit, SISI and Privoxy) or by local GUI applications (see, e.g., Muffin and WBI).

The proxy is able to push information by embedding HTML (or any active content) into the responses that are provided to the user. In this way, the pull paradigm drives the client navigation while the push paradigm is driving the proxy that can be asynchronously updated by servers of any kind (i.e., not only HTTP) as long as the data is pushed to the client by insertion into the HTML page.

Several advantages can be identified with our collateral-push paradigm. First of all, it allows preserving the features of the traditional push model:

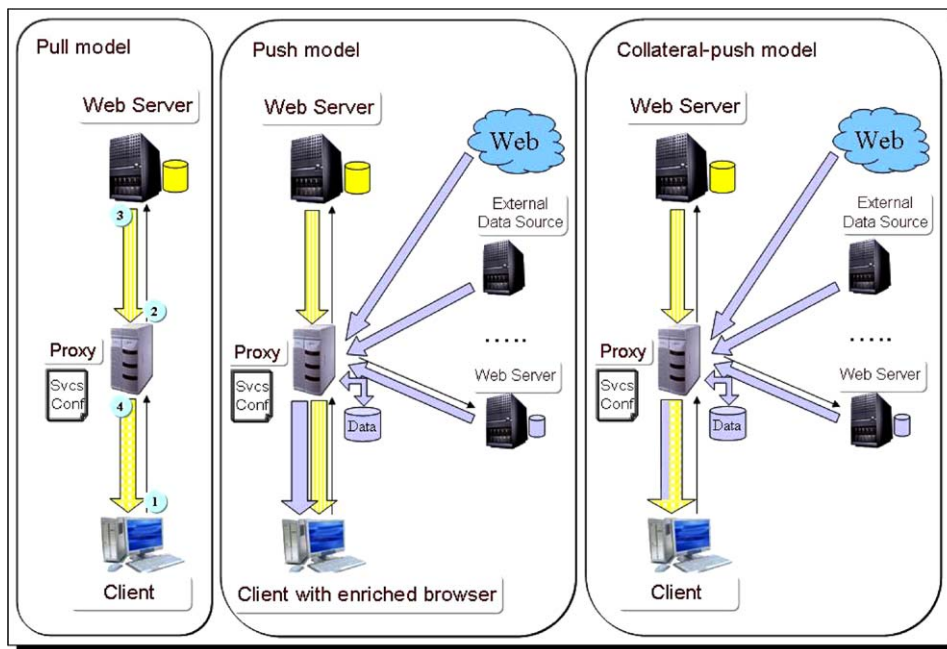


Fig. 1. (Left) In the (traditional) pull model, a client sends an HTTP request to the server, via the proxy, and receives the response, possibly modified/filtered by the proxy. (Center) In the push model, the proxy can asynchronously push toward an enriched browser (e.g., with Java applets, browser plugins, etc.) an information flow that is augmented with additional data from other sources. (Right) The collateral-push model derives from and integrates the pull and the push models for proxies, by adding to or modifying the HTML data that is transmitted to the client under each request/response (i.e., pull) cycle.

information is pushed toward the client at the same rate the client is navigating the Web, i.e., the (client view of) update frequency is “automatically” bound to the client access frequency (of any page anywhere on the Web). We believe that non-intrusiveness is a pivotal characteristic of this model since the moderation in information pushing does not afflict the user when he/she is not navigating: it is a common experience for users to turn off indiscriminate alerts that are too invasive. Our model does offer (moderate) timeliness of information as well as a tactful presence during a user’s navigation.

Secondly, collateral-push does not require any modification to the browser and needs only a negligible change in browser configuration, i.e., setting up the proxy to be used. Moreover, our model preserves the bandwidth of the client (with respect to the “pure” push model) since updates are pushed toward the proxy where the client pulls its data from, and not toward the client that has, presumably, a more limited bandwidth.

By adding relevance to the role of the proxy, we are also envisioning a scenario where services can be added to the traditional navigation and contents, that are horizontally usable in any context and are not domain specific. As an example, adaptivity and cooperation can be realized by leveraging on the HTTP proxy, thereby alleviating the burden on the servers and, furthermore, offering these services in other contexts or for different contents. Finally, services can be easily aggregated on a single proxy but also cascaded through a chain of collaborating HTTP proxies, according to the standard HTTP specifications. In the next subsection, we provide evidence of several existing and new services for Web Dynamics that fit within our model.

Of course, the model is not without faults. First of all, when stringent timeliness is a crucial requirement of the application, collateral-push cannot ensure timely delivery of important, updated information toward the client: in this case, the only alternative is a pure push model with its consequent (but inevitable) costs. Also, the navigational patterns of end-users is, probably, modified if information is c-pushed on its traditional requests (for example, with more frequent reloads of the same page).

In general, however, all proxy-based applications (not only collateral-push but also the traditional pull/push ones) suffer from some drawbacks. In fact, simply adding HTML to the page is, sometimes, not enough for complex services where more interaction is needed (as well as a more elaborate

user interface). Injecting Javascript (or any other content) is complicated and error-prone, because of its interaction with existing active content<sup>2</sup> on the original page. Moreover, any strategy that is to alter the content in any complex way is doomed to be performing always worse as the time passes and the average complexity of HTML pages increases<sup>3</sup> or new standards/styles are adopted in the community of content authors. Finally, of course, our approach is applicable only on standard HTML content and all the “flashy” introductions are ruled out by any HTTP-proxy intervention.

In conclusion, it should be emphasized that not every computation performed by an HTTP proxy fits into this model. For example, filtering and adapting the content (i.e., transcoding, user adaptation, etc.) does not imply pushing information into the HTML page, rather adapting and modifying existing information so that it is better tailored to the end user expectations.

### 3.2. Collateral-push proxy applications in literature

We show, here, different research areas in Web Dynamics where proxy-based solutions naturally fit into the new paradigm of collateral-push. Of course, not all the proxy-based solutions to problems in this research area (as described in [25]) fit within the collateral-push model such as for example, solutions for crawling the Web [32] or for coping with content adaptation [16–18,39].

#### 3.2.1. Web data mining

As defined in [25] Web data mining can be classified into three categories: Web content, structure and usage mining.

Web content mining concerns the discovery of useful information from Web documents. Web structure mining refers to the discovery of information about the link structure of the Web, for example, discovering hubs and authorities, and, finally, Web usage mining can be defined as the automatic discovery of user access through the analysis of

<sup>2</sup> Understanding what is the behavior of an HTML page with active content is computationally undecidable since it can be easily reduced to the halting problem of a Turing machine.

<sup>3</sup> Our unfortunate experience with Group Adaptive Systems (GAS) [3] suggests that any modification on the HTML must be simple (i.e., adding something to links) or placed on the top of the page, if one wants to retain good chances that the service will be working for a significant amount of time over a significant portion of the Web.

Web data logs in order to characterize, for example, Web behaviors of other users.

Efficient solutions for information retrieval, categorization/filtering and personalization can be deployed on the top of intermediary systems [3,20,22].

The Group Adaptive System [3] is a proxy-based infrastructure that offers a collaborative and adaptive Web navigation environment to users through a collaborative interface added to Web pages. Its main objective is to offer Web users adaptive group memberships and group knowledge adaptation. The defined model takes into account the structure of the underlying portion of the Web (using the CLEVER algorithm) and the user interactions on the navigated spaces (gathered by the collaborative environment provided by the system) to offer users in a group a “consistent” information space around their interest topics and activities.

The GAS system enables people to perform asynchronous collaborative navigation: it assembles together resources discovered by a group’s users and suggestions by the CLEVER algorithm providing and c-pushing them as end users group recommendations.

### 3.2.2. Web navigation

The major usability problem that users can experience by surfing the Web is the disorientation due to the complex and always growing structure of the Web (often referred to as the *lost in hyperspace* phenomenon): users cannot usefully navigate Web sites and, thus, finding the desired information is often impossible. Moreover, it is often related to the *cognitive overhead* [10] that people experience when arriving at a particular point in a document they forget what they want to do there, what they are looking for, whether to follow a link or not, etc.

Given this general context of confusion, a lot of users continue their navigation by choosing the first link which appears to suit their requirements, even though it often is not the right choice.

Solutions that attempt to reduce this cognitive overhead should provide a compact overview of the navigated hyperspace or the augmentation of Web documents by enhancing link capabilities.

A Web site Navigation Engine [26] builds trails or sequences of linked pages which can be relevant for a given user query. This proxy-based system uses an adaptive algorithm to compute these trails (it mimics the user navigation session and stores the trails) by displaying them in a tree-like structure

that is c-pushed toward the clients. By expanding the tree, and by placing the cursor on a trail, a pop-up shows information about the links (title, summary of the page’s content, etc.) that are collected from the Web. Given this compact overview of the hyperspace, users can navigate without losing their orientation.

Scone [37] is a modular Java framework whose main goal is to improve the navigation and orientation on the Web, by generating new views of Web documents, offering workgroup tools to support collaborative navigation, enriching Web pages with new navigational elements, etc. Scone has a modular architecture and offers several components, which can be used, enhanced and programmed using a plugin concept.

### 3.2.3. Web annotation

The main goal of Web annotation systems is to adapt page and link presentations to the goals, knowledge, and other characteristics of an individual user. By annotations we mean comments, notes, explanations, or other types of external remarks that can be c-pushed to any Web document or a selected part of the document without changing the original content of the document.

Several systems provide the above functionalities. The Context-aware Annotation Proxy-based System (CAPS) [34] is a proxy-based annotation system that is written in Java (with WBI) and performs several tasks such as logging of user requests, constructing a repository of access statistics, ranking of pages according to these access statistics, etc. When the user browses the Web, CAPS modifies the pages to highlight links’ popularity. Each link recognized by CAPS is annotated with a small image next to it: *known* (yellow), *popular* (yellow–red) and *hit* (red) and, finally, c-pushed towards end users.

HyperScout [36] c-pushes additional information (Title, Author, Language, Last Visit, Bookmark, link action, link status, etc.) about links on HTML pages, while CritLink mediator [38] adds extra toolbars, annotations into the content, and (at the end of the document) metadata and backward links. Finally, Yawas [12] is a Java application that permits the creation, visualization and retrieval of private annotations.

Similarly, shared bookmarks can be c-pushed to Web documents to help organize them under different topics, to easily find them later, to help find related material and to collaboratively filter book-

marked material, as realized in PowerBookmarks [27]. PowerBookmarks, in fact, supports automated bookmark classification (based on the document's content) and also provides many useful personalized services, such as automated discovery of dead links and inactive link removal, and the subscription to new or modified documents (by providing, for example, keyword or document similarity).

#### 3.2.4. Notification systems

The highly dynamic and heterogeneous nature of the Web has involved an increased demand for mechanisms that are able to efficiently detect changes in Web documents and notify them in a timely manner. Users could be interested in knowing changes in specific Web pages, in news which they are subscribed to, students could be interested in changes in their courses' pages, etc.

The idea of notification systems is to avoid a continuous polling of the Web to see if information are changed or not, allowing data dissemination without sending unnecessary or redundant information. This push paradigm, that relieves the user of retrieving information as it changes, takes care of informing the users when something of interest to them happens.

Given the client pull model of the Web, notification systems need to be enhanced with intelligent intermediate components, to provide additional and more powerful functionalities.

An example of an intermediary notification system is WebRACE [28], a prototype of an HTTP Retrieval, Annotation and Caching Engine developed in Java. It is an agent-proxy that collects, processes, annotates and finally caches content from information sources on the WWW. Processing includes, for example, title-extraction, summarization, etc. The output of this processing is sent to a dissemination module whose main goal is to c-push this information towards the clients (by selecting the appropriate data format to deliver, the connection modality, etc.).

In Section 3.3.3 we describe a service that we have implemented for receiving News Alert; it represents another example of a notification system that uses the collateral-push model.

#### 3.2.5. Information filtering

Information filtering aims at helping people to find desired information, from a large amount of available data, by avoiding unnecessary access to irrelevant information.

In particular, to address this issue, two different approaches seem to be particularly useful: *collaborative filtering*, where users' profiles are analyzed to detect relationships to other users, and *content filtering*, that analyzes the content of annotations provided by users, relaying, then, only on the preferences expressed by them. Typical applications of information filtering include recommendations, news filtering, Web navigation, etc.

A collateral-push example of collaborative filtering is the SELECT project [1], a proxy-based system whose main goal is to help users to easily and quickly find interesting information by reducing the information overload. SELECT exploits two filtering techniques: the first makes recommendations according to the users' decisions taken in the past, and the second takes advantage from other users' experiences by making recommendations according to their behaviors.

The SELECT project provides two important tools: a rating tool that allows storing and evaluating the rating of users about Web documents, and a filtering tool, needed to automatically parse and filter Web documents before their delivery. In particular, an implicit rating is realized by taking into account information such as keywords, language of the document, the number of links embedded, etc. An explicit rating, on the other hand, is realized by providing users' profiles in which users specify their interests.

Proxy-based frameworks can be also programmed to operate as recommender systems, assisting users in their navigation and making users aware of other similar users (activities, interests and knowledge). The WebMemex system [29] was developed on top of a high level architecture, an augmented Web proxy-based system, that provides linking, user authentication, storage, retrieval and access capabilities. In particular, HTML documents are saved with information about the user making the request (i.e., URL with access time, IP address of the client machine and user ID). Finally, recommendation are c-pushed in a small pop-up window.

#### 3.2.6. Collaborative navigation

Due to its popularity and its ease of use, the Web is an attractive platform to support distributed cooperative work [7].

Collaborative navigation on the Web can improve the task of finding information by making users aware of other users' activities. Users can

learn about each other as well as learn about the content of Web resources.

The MEMOIR [14] framework supports users working on a large amount of distributed information by assisting them in finding both relevant documents and other similar users, that is, users with similar experience or interests. The main goal of this proxy-based framework is to collect and store information about the trail of the document that the user visits (history of browsing), and to provide the *similarity* by c-pushing it on top of the displayed Web page.

The main goal of the Group Annotate Transducer (GrAnT) [33] is to create and share comments, suggestions or ideas about the content of documents accessible via the World Wide Web. In particular the system supports the creation, presentation, and control of user-created meta-information, which is displayed within the documents but stored separately from them.

Most of the functionalities of the system, such as annotations (i.e., the author, the subject, the creation's date, the comment itself), are stored by an Annotation server and added to the Web content by a proxy component that, finally, c-pushes them toward end users.

### 3.3. New intermediary services

In this section, we describe some services that we have quickly implemented with two different proxy infrastructures, i.e., WBI [5] and SISI [19] that are briefly described in Section 2.2. In fact, the goal of the section is to further support our view that an advanced proxy environment must fully support the application programmer so that he/she can quickly prototype new collateral-push services that can address some of the issues in the area of Web Dynamics. In a way, what we claim is that quick prototyping is a key requirement given the impetus of the issues that come to light as the Web evolves (in sometimes unexpected directions).

With the first service shown here, the File Extension Manager service, we aim at showing how to augment Web pages with additional information that could be useful for end users while they are navigating on the Web. Secondly, we show the TrafficLights service that provides a clue (on each link in the page) about how fast a link is going to be, by adding red/yellow/green lights.

Then, the News Alert service represents a classical example of the collateral-push paradigm that

we described before (see Section 3.1), and its goal is to show to the user important news reports while navigating on the Web. Finally, the goal of the MetaSearch Engine service is to improve the quality of any search engine employed by the user, by tagging the results that are highly ranked in other search engines.

#### 3.3.1. File extension manager service

While browsing the Web, several forgetful or inexperienced users try to access or download files without knowing their format and, consequently, without information about the needed applications to view them.

The goal of the *File Extension Manager* service (FEM Service) is to address this issue by dynamically providing additional information about unknown files formats available on the Web, relieving thus the users from doing this task.

For each resource embedded in a Web page, the FEM service analyzes its file extension and adds a specific-application program icon into the HTML by adding the information obtained by the distributed database at the URL [<http://filext.com>], such as the application program, the MIME type, some associated links, etc. (see Fig. 2). The service is written in Perl under SISI.

#### 3.3.2. TrafficLights service

This service was originally developed by the WBI team at IBM. Here, we describe a simpler service that we implemented in Perl under SISI. The goal of the TrafficLights service is to test the availability of the links of Web pages navigated by end users, in order to detect the broken ones and then allow the users to avoid surfing them. It is an ideal service to users that want to avoid overcrowded links that dynamically change the quality of user experience of the Web.

For each link embedded in a Web page, the TrafficLights service instances an HTTP connection with the origin server that provides the resource and estimates its reply time. According to the estimated responses times, it adds an image at each link (see Fig. 3). A green color image states that the link is up and quickly reachable (time less than 200 ms), a yellow color image states a medium connection (within 200 and 500 ms) and, finally, a red color image states that the link connection is very low or the link is broken or unreachable.

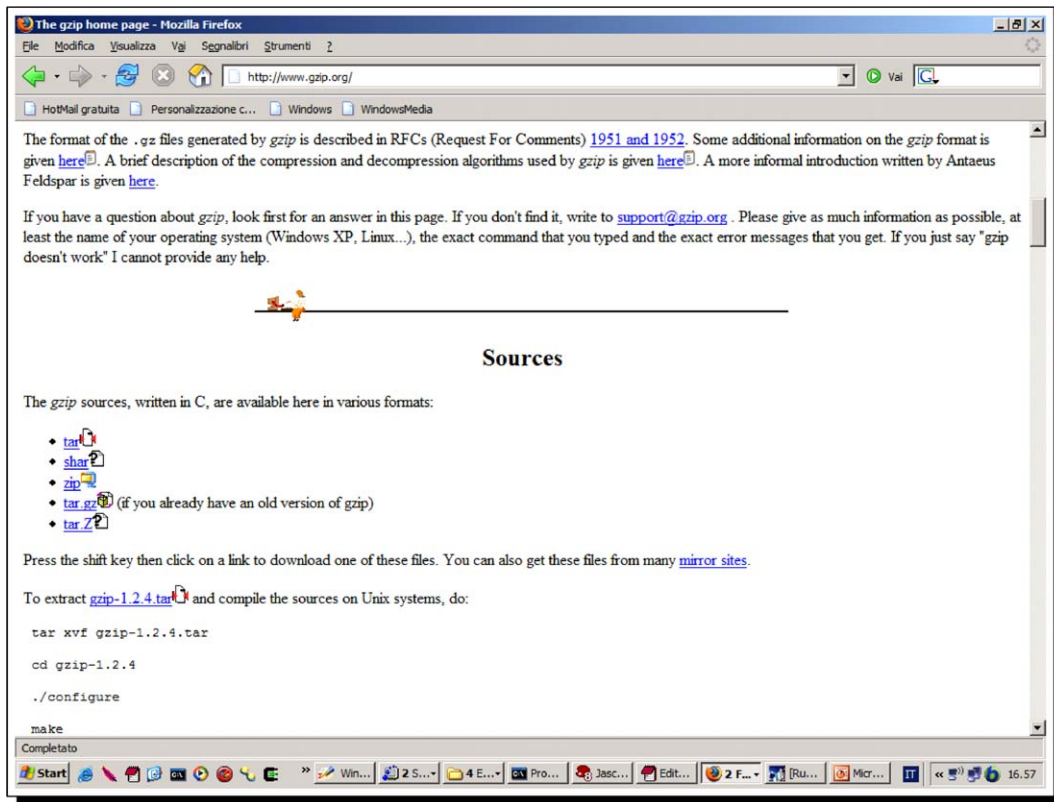


Fig. 2. The File Extension Manager Service (FEM service).

### 3.3.3. News Alert service

The goal of a News Alert service is to deliver new information about specific event in a timely manner while the user is navigating. It is implemented on WBI and, therefore, is written in Java.

In particular, users can be interested in getting articles from news portals and Internet magazines, messages from forums, information about new software releases, etc. Rather than continuously polling Web portals to get new information, news alert systems provide it as soon as it becomes available.

Several systems exist with the aim of collecting information from different resources in the network, in particular we can cite the Google News Alert provided by the Google search engine. The new information, about topics previously specified, are sent via email as soon as it becomes available. GetNews [<http://www.getnewsgroup.com/about-getnews.php>] is a program for collecting the news articles from different Web sites. It can automatically check the enabled channels and save the retrieved information in the news archive. RssReader [<http://www.rssreader.com>] collects news in the background at

user-configurable intervals and warns with a little pop-up in the system tray that a new message has arrived. Any user can click the news headline to see a short description of the news and click or open the original news Web page in an RssReader browser or default browser window.

The goal of our News Alert service is to provide timely information to end users during their navigation on the Web. The user must subscribe to the service by filling a form (offered proxy-side) that requires, among different categories, the topics he/she is most interested in (*politics, science, news in the World, news in Italy*, etc.), and (possibly) some keywords for further refinements.

The News Alert service maintains, for each category, a file containing two items of information, that is, the URL of the news source (i.e., [<http://news.google.it>]) and the patterns that are needed by an HTML parser, instantiated by the service, that retrieves single news items. Then, news items are filtered by keywords (if specified) and, finally, the information will be c-pushed to end users on the Web page that they are surfing. By using user

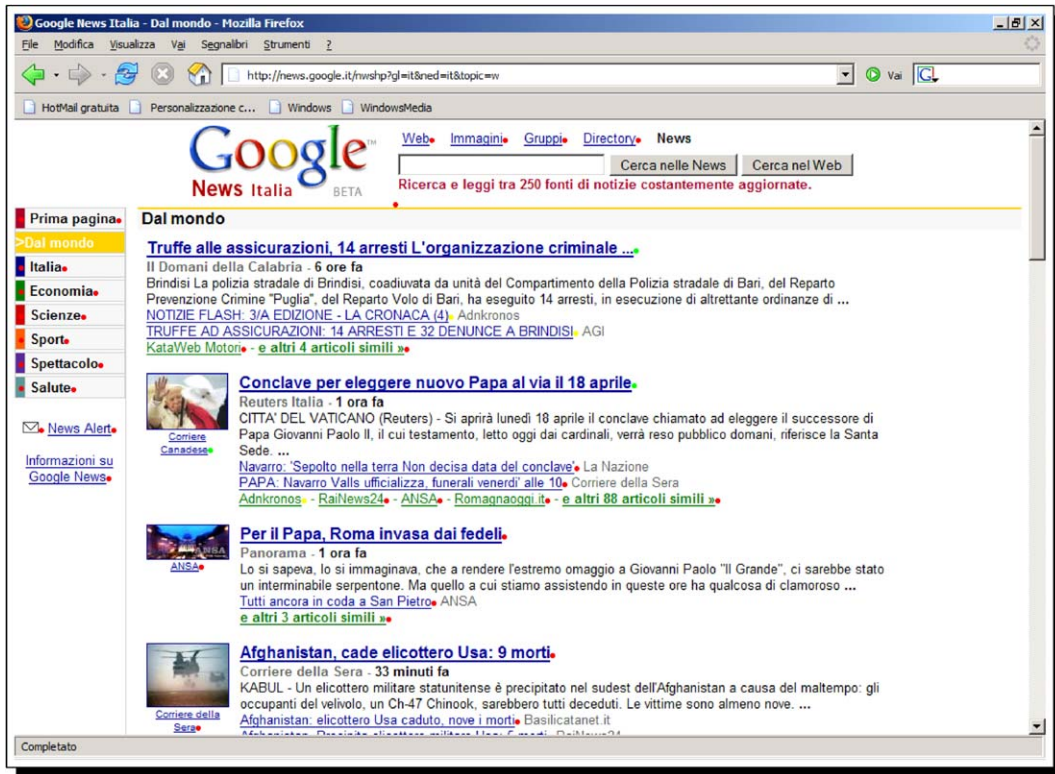


Fig. 3. The TrafficLights Service. Here, the Web page is the news provided by Google (in Italian).

profiles, only the requested categories are shown on the top of the page (see Fig. 4), with an hypertext link (with the title of the news) that shows more details on the news in a separate browser window.

No further interaction is requested from the user: transparently and asynchronously, the system gathers new information (at regular intervals) and pushes the information to end users.

### 3.3.4. Metasearch engine service

Metasearch engines are powerful tools that search multiple engines simultaneously. Unlike crawler-based search engines such as Google, Yahoo or Altavista, they do not maintain indexes of Web pages, they do not categorize them, but they simply use results built by others, aggregate them from different sources and show them in a unique and compact way.

Traditional metasearch engines accept user queries, and forward them to multiple engines in parallel. Among the advantages, first, we remark on *efficiency*, since multiple search engines are simultaneously contacted for a given query relieving the user from visiting each of the engines in turn, and *effectiveness* since a search made on multiple sources

allows more interesting results to be obtained. Finally, metasearch engines represent an excellent solution if users are interested in getting a broad and multi-source overview of specific topics.

Metasearch engines present results in different ways. Some of them, such as dogpile [<http://www.dogpile.com>], simply list the top-10 results from each engine. Others analyze the results and rank them according to internal rules, aggregating results in a single list in the displayed Web page. Examples are IxQuick [<http://eu.ixquick.com>], Vivissimo [<http://vivissimo.com>] and Metacrawler [<http://www.metacrawler.com>].

It is well-known [21] that the top-10 results of different search engines have unexpectedly low overlap and therefore studying and comparing the rank of each link in different engines is important. In fact, the added-value for the user of knowing that a particular page is present in more than one search engine's top-10 results for the specific query must not be ignored and, on the contrary, should be emphasized as a stronger validation of the relevance of the page under different search engines (which means different dataset, different algorithms, different times of Web crawls, etc.).

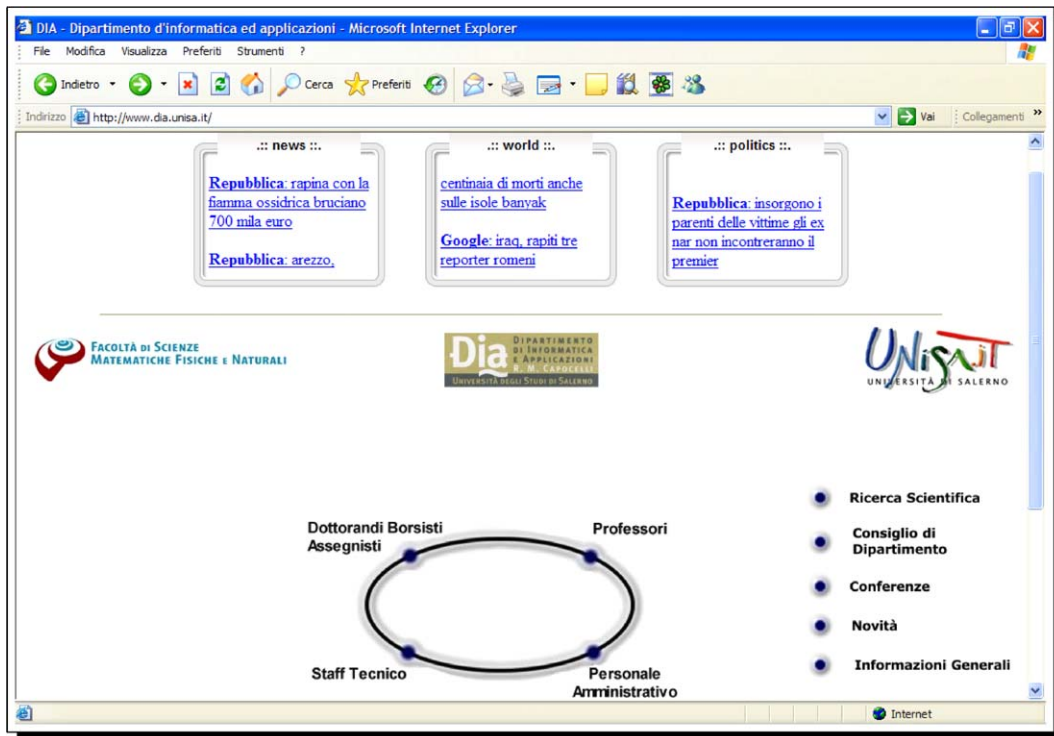


Fig. 4. The News Alert Service: the topmost part is inserted on-the-fly by the proxy and contains news stories of top interest. In this example, the news sources are [<http://news.google.it>] and [<http://www.repubblica.it>], the Web site of “La Repubblica”, one of the major newspapers in Italy.

We developed our MetaSearch Engine service in Perl under SISI: once the results, for a given user query, are retrieved by the user’s favorite search engine (for example, Google), our MetaSearch Engine augments the HTML page with information that helps users to identify the links that are present in other top-10 lists from other engines. As a consequence, the user can choose to query any of the six search engines (chosen among the most popular), i.e., Google, Altavista, Yahoo, Virgilio (an Italian search engine with a portal), AskJeeves and AlltheWeb, and the service will show the results that overlap with the top-10 list of the remaining five engines.

When the user issues an HTTP request to a specific search engine, the MetaSearch Engine, by using the `LWP::Parallel::UserAgent` Perl library (freely downloadable from the CPAN Perl Archive), performs parallel queries on multiple search engines. In order to preserve efficiency, the parsing is also performed in parallel and begins as soon as the beginning of the response becomes available. Results that are in common with other search engines are stored in a hash table. The response consists of the results of the query to the indicated

search engine but each result is augmented by indicating if the result does also appear in other search engines among the top 10. In particular, the service shows in the HTML response the results that are in common with the other search engines by placing a small mnemonic icon (see Fig. 5).

Several commercial/free metasearch engines exist but our service is different in nature. In fact, being provided by a proxy, it does not alter our “favorite” search engine results and simply augments it with additional information on the relevance of some links. As a matter of fact, conventional metasearch engines may be using different weights for results coming from different search engines, and, often, their choice is hidden to users (i.e., IxQuick). Other solutions provide all the results together and it may be difficult to single out the contribution of our favorite search engine (e.g., dogpile, Mamma MetaSearch, MetaCrawler, etc.).

On the contrary, our proxy-based solution leaves the user in control to keep his/her favorite search engine (for example, a nationalized version or a highly specialized one) but enhances its performances by signaling, on top of it, the links that

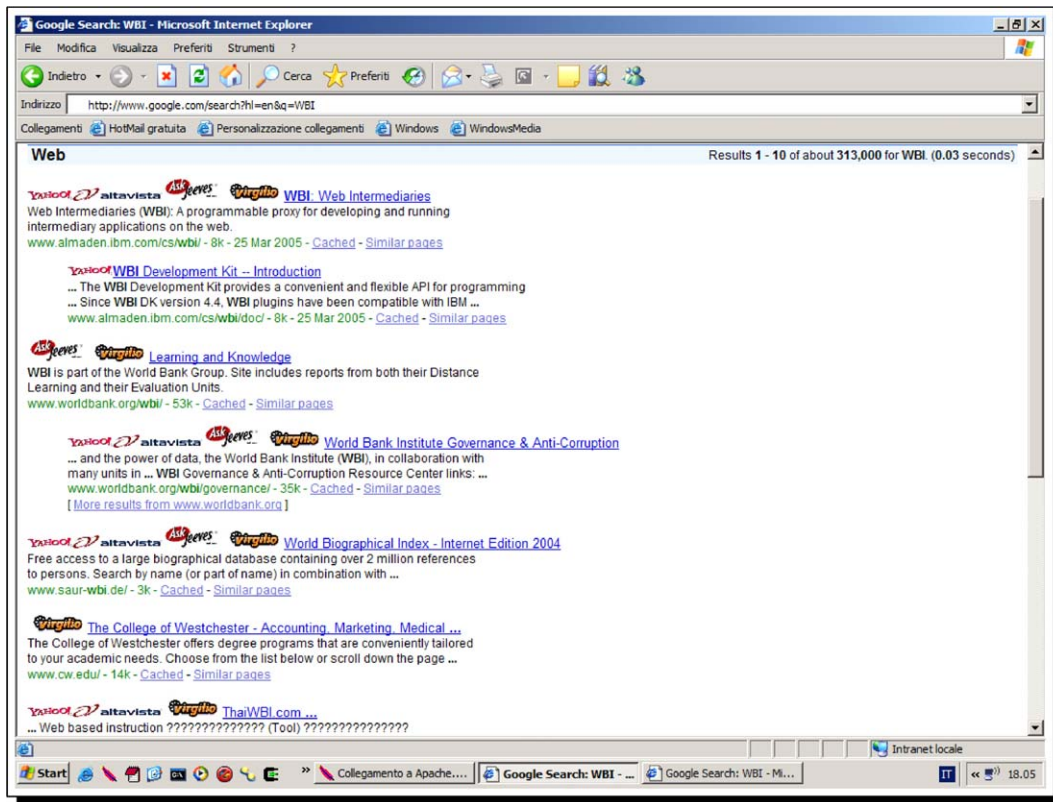


Fig. 5. An example of our MetaSearch Engine Service. The query is to Google for the keyword “WBI” and some results are shown to be present also in top-10 results given by Yahoo, Altavista, etc.

appear to be in other top-10 lists. As a consequence, some results are emphasized to the user but in an unobtrusive way and without any modification of the traditional search patterns of the user.

#### 4. Conclusions

In this paper we have reviewed the state of the art in programmable proxies and have introduced a new computational model, called *collateral-push*, for proxies to deal with the dynamic nature of the Web. The model is derived from the combination of the traditional push and pull paradigms, with the objective to include most of their advantages. We presented some evidence of the effectiveness of a proxy-based approach by examining several services developed in the area of Web Dynamics and by showing four new services that offer support and advanced capabilities to a user navigating the Web via a standard browser.

These intelligent components represent an ideal solution for tackling a relevant part of the problems that come from the continuous changes of docu-

ments available on the Web. In fact, proxies can be deployed at any point along the content path between client and server by providing functionalities that meet the issues of providing advanced services and Web contents on behalf of one or more origin servers and relieve clients of the burden of new responsibilities that go beyond their natural, simple role in the Web architecture.

Of course, this approach to deal with the dynamics of the Web with intermediaries is not absolutely faultless. In fact, the same dynamics that the services are asked to ameliorate constitutes a source for the need of continuous upgrades of the services themselves, since the HTTP-flow the services work on changes as the Web adjusts itself to new standards, new practices, etc. Moreover, adding new components to the Web architecture increases the management load of new hardware/software for any new service to be deployed. Also, the integration with existing (server-based) content/service providers is going to be a challenging task. Just to name one problem, the authentication that is required on the proxy-side needs to be synchronized

in some way with authentication schemes that are enforced on the server-side. Currently, there is no solution from standards to this problem.

Nevertheless, our thesis is that the architecture of the services currently deployed on the Web are (almost) fully exploiting the capabilities of client and servers, while the perspectives offered by proxies are still to be fully explored and will offer an environment for effectively tackling Web Dynamics issues.

## Acknowledgments

We gratefully thank for all the helpful feedback provided by the members and students of ISISLab in our Department. In particular, interesting discussions with Raffaella Grieco and Alberto Negro were particularly fruitful. Some of the services shown are based on the prototypes programmed with Domenico Giordano and Marco Del Percio. We also gratefully thank the anonymous reviewers for several useful and interesting comments on the paper. This research work was (partially) financially supported by the Italian FIRB 2001 project number RBNE01WEJT “WEB-MiNDS” (Wide-scale, Broadband Middleware for Network Distributed Services) <http://web-minds.consorzio-cini.it/>.

## References

- [1] R. Alton-Scheidl, J. Ekhal, O. van Geloven, L. Kovacs, A. Micsik, C. Lueg, R. Messnarz, D. Nichols, J. Palme, T. Tholerus, D. Mason, R. Procter, E. Stupazzini, M. Vassali, R. Wheeler, SELECT: social and collaborative filtering of web documents and news, in: Proceedings of the 5th ERCIM Workshop on User Interfaces for All: User-Tailored Information Environments, November 1999, pp. 23–37.
- [2] S. Ardon, P. Gunningberg, B. LandFeldt, M.P.Y. Ismailov, A. Seneviratne, MARCH: a distributed content adaptation architecture, *International Journal of Communication Systems*, Special Issue: Wireless Access to the Global Internet: Mobile Radio Networks and Satellite Systems 16 (1) (2003) 97–115.
- [3] M. Barra, P. Maglio, A. Negro, V. Scarano, GAS: group adaptive system, in: Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2002), ACM Press, 2002, pp. 47–57.
- [4] R. Barrett, P.P. Maglio, Intermediaries: new places for producing and manipulating web content, *Computer Networks and ISDN Systems* 30 (4) (1998) 509–518.
- [5] R. Barrett, P.P. Maglio, Intermediaries: an approach to manipulating information streams, *IBM Systems Journal* 38 (4) (1999) 629–641.
- [6] R. Barrett, P.P. Maglio, WebPlaces: adding people to the Web, in: Poster Proceedings of the 8th International World Wide Web Conference, ACM Press, Toronto, Canada, 1999.
- [7] G. Cabri, L. Leonardi, F. Zambonelli, A proxy-based framework to support synchronous cooperation on the Web, *Software—Practice and Experience* 29 (14) (1999) 1241–1263.
- [8] S. Chakravarthy, N. Pandrangi, A. Sanka, WebVigil: an approach to just-in-time information propagation in large network-centric environments, in: Proceedings of the 2nd International Workshop on Web Dynamics, May 2002. Available from: <<http://www.dcs.bbk.ac.uk/webDyn2/onlineProceedings.html>>.
- [9] M. Colajanni, R. Grieco, D. Malandrino, F. Mazzoni, V. Scarano, A scalable framework for the support of advanced edge services, in: Proceedings of the 2005 International Conference on High Performance Computing and Communications (HPCC-05), September 2005, pp. 1033–1042.
- [10] J. Conklin, Hypertext: an introduction and survey, *Computer* 20 (9) (1987) 17–41.
- [11] G. Cugola, H.-A. Jacobsen, Using publish/subscribe middleware for mobile systems, *SIGMOBILE Mobile Computing Communication Review* 6 (4) (2002) 25–33.
- [12] L. Denoue, L. Vignollet, An annotation tool for Web browsers and its applications to information retrieval, in: Proceedings of the 6th Conference on Content-Based Multimedia Information Access, RIAO2000, Paris, April 2000.
- [13] P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamrithan, P. Shenoy, Adaptive push-pull: disseminating dynamic web data, in: Proceedings of the 10th International World Wide Web Conference, ACM Press, Hong Kong, 2001, pp. 265–274.
- [14] D. DeRoure, W. Hall, S. Reich, A. Pirkakis, G. Hill, M.A. Stairmand, An open architecture for supporting collaboration on the web, in: WETICE '98: Proceedings of the 7th Workshop on Enabling Technologies, Washington, DC, USA, 1998, IEEE Computer Society, pp. 90–95.
- [15] M. Dikaiakos, D. Zeinalipour-Yiazti, A distributed middleware infrastructure for personalized services, Technical Report TR-2001-4, University of Cyprus, December 2001.
- [16] A. Fox, E.A. Brewer, Reducing www latency and bandwidth requirements by real-time distillation, in: Proceedings of the Fifth International World Wide Web Conference on Computer Networks and ISDN Systems, Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1996, pp. 1445–1456.
- [17] A. Fox, Y. Chawathe, E.A. Brewer, Adapting to network and client variation using active proxies: lessons and perspectives, *IEEE Personal Communications* 5 (4) (1998) 10–19.
- [18] A. Fox, I. Goldberg, S. Gribble, D. Lee, A. Polito, E. Brewer, Experience with top gun wingman, a proxy-based graphical web browser for the user palmpilot, in: Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), September 1998, pp. 407–426.
- [19] R. Grieco, D. Malandrino, F. Mazzoni, V. Scarano, F. Varriale, An intermediary software infrastructure for edge services, in: Proceedings of the 1st International Workshop on Services and Infrastructure for the Ubiquitous and Mobile Internet (SIUMI'05) in Conjunction with the 25th International Conference on Distributed Computing Systems (ICDCS'05), June 2005, pp. 259–265.
- [20] E. Herder, H. Weinreich, Interactive web usage mining with the navigation visualizer, in: CHI '05: CHI '05 Extended

- Abstracts on Human Factors in Computing Systems, ACM Press, New York, NY, USA, pp. 1451–1454.
- [21] J. Ilan, M. Levene, M. Hassan, Dynamics of search engine rankings—a case study, in: Proceedings of the 3rd International Workshop on Web Dynamics, May 2004. Available from: <<http://www.dcs.bbk.ac.uk/webDyn3/>>.
- [22] T. Joachims, D. Freitag, T. Mitchell, Webwatcher: a tour guide for the World Wide Web, in: Proceedings of the 15th International Conference on Artificial Intelligence, 1997, pp. 770–775.
- [23] R.H. Katz, E.A. Brewer, E. Amir, H. Balakrishnan, A. Fox, S. Gribble, T. Hodes, D. Jiang, G.T. Nguyen, V. Padmanabhan, M. Stemm, The Bay Area research wireless access network (BARWAN), in: Proceedings of the 1st IEEE International Computer Conference, IEEE Computer Society, 1996, pp. 15–20.
- [24] B. Krishnamurty, J. Rexford, Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching and Traffic Measurement, Addison-Wesley, 2001.
- [25] M. Levene, A. Poulouvassilis (Eds.), Web Dynamics—Adapting to Change in Content, Size, Topology and Use, Springer, 2004.
- [26] M. Levene, R. Wheeldon, J. Bitmead, A Web site navigation engine, in: Poster Proceedings of the 10th International WWW Conference, 2001.
- [27] W.-S. Li, Q. Vu, D. Agrawal, Y. Hara, H. Takano, Powerbookmarks: a system for personalizable web information organization, sharing, and management, Computer Networks 31 (11–16) (1999) 1375–1389.
- [28] D.Z.-Y.M. Dikaiakos, WebRACE: a distributed WWW retrieval, annotation, and caching engine, in: International Workshop on Performance-oriented Application Development for Distributed Architectures, April 2001.
- [29] A.A. Macedo, K.N. Truong, J.A. Camacho-Guerrero, M. da Grata Pimentel, Automatically sharing web experiences through a hyperdocument recommender system, in: Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia (HYPERTEXT '03), ACM Press, 2003, pp. 48–56.
- [30] D. Malandrino, V. Scarano, A taxonomy of programmable HTTP proxies for advanced edge services, in: Proceedings of the 1st International Conference on Web Information Systems and Technologies (WEBIST), May 26–28, 2005, pp. 231–238.
- [31] C. Rao, Y. Chen, D.-F. Chang, M.-F. Chen, iMobile: a proxy-based platform for mobile services, in: Proceedings of the First ACM Workshop on Wireless Mobile Internet (WMI 2001), ACM Press, 2001, pp. 3–10.
- [32] K.M. Risvik, R. Michelsen, Search engines and Web dynamics, Computer Networks 39 (3) (2002) 289–302.
- [33] M.A. Schickler, M.S. Mazer, C. Brooks, Pan-browser support for annotations and other meta-information on the World Wide Web, Computer Networks and ISDN Systems 28 (7–11) (1996) 1063.
- [34] T. Sharon, H. Lieberman, T. Selker, Searching the Web with a little help from your friends, in: ACM Conference on Computer-Supported Cooperative Work, ACM Press, New Orleans, 2002.
- [35] P. Sutton, R. Arkins, B. Segall, Supporting disconnectedness-transparent information delivery for mobile and invisible computing, in: Proceedings of the 1st International Symposium on Cluster Computing and the Grid (CCGRID '01), Washington, DC, USA, 2001, IEEE Computer Society, p. 277.
- [36] H. Weinreich, W. Lamersdorf, Concepts for improved visualization of Web link attributes, Computer Networks, Selected Papers from the WWW9 Conference 33 (1–6) (2000) 403–416.
- [37] H. Weinreich, H. Obendorf, W. Lamersdorf, The look of the link, concepts for the user interface of extended hyperlinks, in: Proceedings of the 12th ACM Conference on Hypertext and Hypermedia, University of Aarhus, Denmark, The Association for Computing Machinery, ACM New York, 2001, pp. 19–28.
- [38] K.-P. Yee, CritLink: advanced hyperlinks enable public annotation on the Web, in: ACM Conference on Computer-Supported Co-operative Work, November 2002.
- [39] B. Zenel, A general purpose proxy filtering mechanism applied to the mobile environment, Wireless Networks 5 (5) (1999) 391–409.



**Delfina Malandrino** received the Laurea and Ph.D. degrees in Computer Science from the University of Salerno in November 2000 and April 2003, respectively. She is currently a post-doc in “Dipartimento di Informatica ed Applicazioni R.M. Capocelli” at the University of Salerno, Italy.

Her research field is Distributed Systems with a focus on intermediary architecture for Web content delivery and

adaptation.



**Vittorio Scarano**, was born in Naples (Italy). He has received the Laurea degree in “Scienze dell’Informazione” (Computer Science) from the University of Salerno (Italy) in 1990 and the Dottorato di Ricerca (PhD) in Applied Mathematics and Computer Science from the University of Naples (Italy) in 1995. He is currently associate professor at University of Salerno. He has been a visiting professor at the Eotvos Lorand

University in Budapest (Hungary) and, from 1992 to 1994, he worked in the Department of Computer Science of the University of Massachusetts at Amherst (USA) doing research with Prof. Arnold Rosenberg.

His research is currently focussed on Multimedia and Distributed Systems on the World Wide Web, covering several aspects from a theoretical perspective (P2P systems and architectures) to applications (intermediaries, cooperative systems and multimedia).