TextToSpeech: a Heavy-weight Edge Service

Maria Barra, Raffaella Grieco, Delfina Malandrino, Alberto Negro, Vittorio Scarano
Dipartimento di Informatica ed Applicazioni "R.M. Capocelli"
Università di Salerno
84081, Baronissi (Salerno), Italy
{marbar,delmal,alberto,vitsca}@dia.unisa.it

1. INTRODUCTION

Among the characteristics that made the Web a success, an important role is played by its simplicity and ubiquity: a common interface allows to access resources that are equally *close* to the user (i.e. "just a click away").

During the technical evolution of the Web, keeping intact its simplicity has been one of the overall objectives: any added capability is to be realized on top of the three existing and (rather) simple standards, such as HyperText Markup Language (HTML), HyperText Transfer Protocol (HTTP) and Uniform Resource Locator (URL). Moreover, every new capability relied on the underlying client-server architecture, allowing (only) a remarkable variety of places where computation took place ranging from the classical *fat server* paradigm (in typical multi-layer architectures) to the *fat-client* paradigm where the most part of the computation is performed on the client-side (Java applets, external plug-ins, etc.).

Recently, a computational paradigm shift has affected the Web by focussing on an (until then) obscure and neglected component of the overall architecture: the proxy.

Actually, the HTTP specifications [5], already, suggested to use proxies to increase the efficiency of the Web by appropriate and intelligent caching of resources. As a result, until the end of the last century, proxies and caching were almost synonyms in the Web architecture. The goal was reducing the cost of connecting to the Internet, by allowing intermediate caching of the resources accessed via the bandwidth-consuming World Wide Web, and reducing the user-perceived latency.

As the nature of the services delivered by the Web grew more and more rich and heterogeneous, it became necessary to design and implement more and more elaborate caching strategies and infrastructures that later evolved into the "Content Delivery Networks" (CDNs) that, first, introduced the concept of "*Edge*" of the network. In fact, CDNs are based on a conceptualization of the World Wide Web in three concentric layers: the inner core is the Data/Service Centers that originate the content; the inner core is connected to a middle Transport layer that is, finally, connected to the Access Network layer that takes care of the "last mile" delivery of the data flow. The idea of CDNs is to develop caching services close to the "Edge" of this diagram, that is, as close as possible to the end-users that are, indeed, the ultimate judges of the quality of the service. Of course, this was not an easy task since the evolution and richness of the services provided by the Web require complex systems to make sure that the user is accessing updated content while efficiently caching the (parts of the) HTML page on the edge.

While caching proxies (and their evolution, the Content Delivery Networks) are important elements of the Web architecture and contribute to the scalability and efficiency of the services provided to the end-user, it is now becoming clear that the Edge of the network is an important component that can be used as a platform to deploy additional heavy-weight services that cannot be provided neither client- nor server-side. The recent hype on "Edge Computing" emphasizes the role of "intermediaries" [3] on the information flow as an ideal place to add value to the traditional client-server interaction.

Many are the application fields of Edge-computing Services that our vision encompasses: from the *geographical personalization* of the navigation of pages, with insertion/emphasis of content that can be related to user geographical location, to *translation services*; from support for *group navigation and awareness* for social navigation to advanced services for *bandwidth optimization* such as adaptive compression and format transcoding. A particularly useful application field for EcSs is the accessibility of Web sites. These applications are usually heavy-weight and therefore, assuring their scalability is particularly critical. A typical example is an ubiquitous service that provides text-to-speech translation of the pages navigated by the user, regardless of the location and configuration of the local machine (i.e. without installing particular software).

In this paper we describe a prototype implementation of this service using a framework for Scalable Edge-computing Services (SEcS) [1] based on IBM Web Based Intermediaries (WBI) [2, 3], a programmable HTTP proxy. SEcS architecture is illustrated by Fig. 1 where we show the relationships among the three basic components: Dispatcher, Remote Proxy and Distributed Registry.

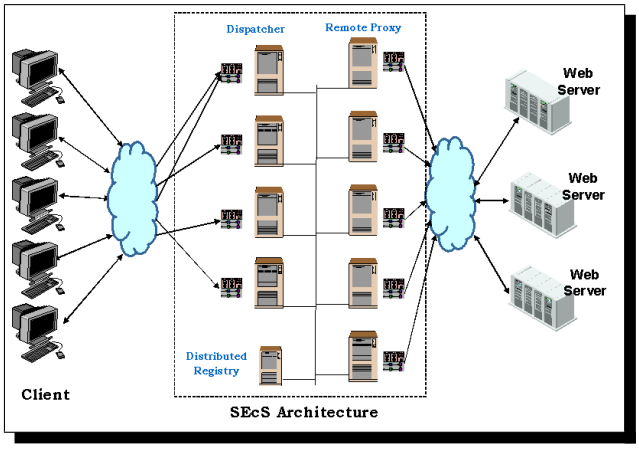

Figure 1: SEcS Architecture.

The architecture consists of several Dispatchers (i.e. the "local"

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM xxx.

proxies that deal directly with the clients' requests) and several Remote Proxies (i.e. proxies whose components (MEGs) can be invoked and used by any Dispatcher).

In a nutshell, the Dispatchers act as "proxy" (in the HTTP context) and can either provide the service by themselves or invoke remote MEGs that are placed on the Remote Proxies. Each Dispatcher/Remote Proxy is executed within a separate WBI session as a plugin. The Distributed Registry allows to keep the internal status in terms of addresses (i.e. hosts) of available Services. Load information (to balance the workload) are sent with the responses to the remote invocations to each Dispatcher. It should be emphasized that a Dispatcher and a Remote Proxy can also be run jointly on the same (Java Virtual) machine, in a common WBI session as two separate plugins.

In SEcS distributed architecture, it is necessary to spread evenly the requests among the Dispatchers. Solutions to this problem had to rely exclusively on standard mechanisms (i.e. no modification of clients was allowed). Our solution is using a Javascript Client Autoconfiguration file that implements an hash function to evaluate the target Dispatcher to be used.

2. TextToSpeech

In our society, people communicate essentially through two modes: visual and acoustics. Usually, by listening to music, melodies, messages, etc., is possible to perceive a small part of the "input" that regulates our daily life, while the other consistent part is graphically represented in various way. The obstacles to people with visual handicap can be, in some cases, minimized by using adequate traslation of graphical stimuli in auditory ones. A significant help is given, in this context, by the use and diffusion of software that is able to reproduce by audio the information provided in graphical (written or image) form.

The TextToSpeech Service has been designed to make Web Content accessible to people with visual disabilities. The nature of this service makes it an ideal testbed for our definition of Edge-computing Services since it is a heavy-load service that should be ubiquitous, i.e. regardless of the software installed on the local client. In fact, several solutions are provided to overcome this handicap while using a computer but all of them require the local installation of a particular software to navigate (such as IBM Home Page Reader [6] or CAST eReader [4]).

To implement this Service we have followed the W3C Guidelines [9] that discuss accessibility issues and provide accessible design solutions. This Guidelines address typical scenario that may create problems for users with disabilities. The main goal of W3C's Guidelines is to promote accessibility, in addition they can help the user to find information on the Web more quickly.

Moreover, the TextToSpeech Service can help the comprehension of documents that are written in foreign languages, since a reader that is partially familiar with the spoken language and not with its written form can be supported in getting, at least, the meaning of the information contained in the document. Finally, the service can be used as a support toward kids that want to learn a language since provides a written/spoken presentation of material of their interest during a "natural" experience as navigating the Web.

To build the prototype of this service in SEcS we used a freely available tex-to-speech synthesizer called FreeTTS [8]. FreeTTS 1.1.0 is a vocal synthesizer written entirely in Java and it is based on a small motor developed at the Carnegie Mellon University. FreeTTS includes a motor for the vocal synthesis that supports a certain number of voices (male and female) at different frequencies. It is supported by JSAPI 1.0 (Java Speech Bees) which provide the method to check and to use FreeTTS.

In Fig. 2 the TextToSpeech Service is shown. The service consists of a single remote MEG, `TTWGenerator`. When invoked, TTWGenerator contacts the origin server to get the HTML document (Doc), it parses the Doc document to eliminate useless characters, the resulted document (Text) is parsed to extract the text (from HTML to text) using the "lynx" command, then the TextToSpeech is passed to the FreeTTS synthesizer.

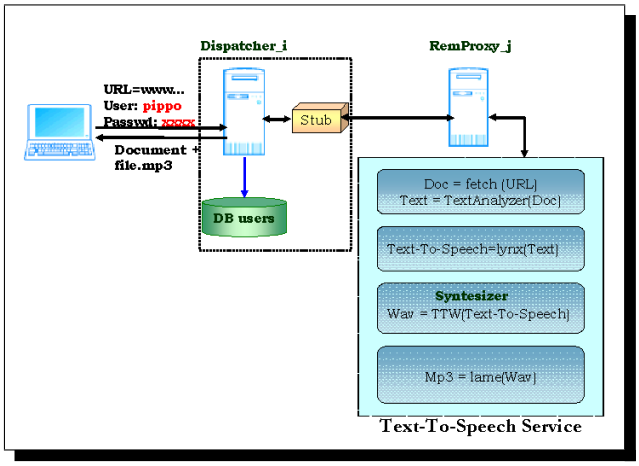

Figure 2: The structure of the TextToSpeech service in SEcS.

Once the synthesizer outputs the WAVE file (Wav), by using the "lame" application [7], the WAVE file is translated into a (more compact) MP3 file (Mp3) that is incorporated into the HTML page through the `EMBED` tag and, finally, the requested document is returned to the client.

When the output reaches the browser, the result will be a normal page HTML with contemporaneous reading of the content.

3. REFERENCES

[1] M. Barra, R. Grieco, D. Malandrino, A. Negro, and V. Scarano. Secs: Scalable edge-computing services. (submitted for publication).

[2] R. Barrett and P. P. Maglio. Intermediaries: New places for producing and manipulating web content. In *Proc of* 7^{th} *International WWW Conference*, 1998.

[3] R. Barrett and P. P. Maglio. Intermediaries: An approach to manipulating information streams. *IBM Systems Journal*, 38(4):629–641, 1999.

[4] CAST. eReader. http://www.cast.org/tools/teachingtoolsreader.html.

[5] R. Fielding, J. Gettys, J. Mogul, H. F. Nielsen, and T. Berners-Lee. HTTP version 1.1, January 1997. RFC 2616.

[6] IBM. Home Page Reader. http://www.austin.ibm.com/sns/hpr.html.

[7] The Lame Project. http://www.mp3dev.org/mp3.

[8] Speech Integration Group of Sun Microsystems Laboratories, FreeTTS. http://freetts.sourceforge.net/docs/.

[9] Web Content Accessibility Guidelines 1.0, W3C Recommendation, May 1999. http://www.w3.org/TR/WCAG10/.