Proteus: An Architecture for Adapting Web Page on Small-Screen Devices

M.F. Caetano[1], A.L.F. Fialho[1], J.L. Bordim[1], C.D. Castanho[1], R.P. Jacobi[1], and K. Nakano[2]

[1] Department of Computer Science, University of Brasilia, Brasilia, 70910–900, Brazil
{caetano, alfoltran, bordim, carlacastanho, rjacobbi}@cic.unb.br
[2] Department of Information Engineering, School of Engineering, Hiroshima University 1-4-1 Kagamiyama, Higashi-Hirhoshima, 739-8527, JAPAN
nakano@hiroshima-u.ac.jp

Abstract. Reading the contents of Web page with a small-screen device, such as a PDA or cell-phone, is still far from being a pleasant experience. Owing to the device limitations, current mobile browsers cannot handle all HTML tags, such as tables, for instance. Thus, most mobile browsers provide a linearized version of the source HTML page, leading to a large amount of scrolling, not to mention the difficulty in finding the desired content. The main contribution of this work is to propose an architecture for adapting web page on small-screen devices. Among the features that our architecture offers, we can cite on-the-fly Web page adaptation and customization according to the user and device characteristics; text summarization; page blocks identification and content mapping to easy the task of locating user interests.

1 Introduction

Today, browsing the Web and reading emails while on the move has become common place. This has been possible due to cheaper and faster wireless network interfaces and the availability of mobile devices with augmented storage, memory, and battery capacity. Even though current cell-phones and PDAs have considerable processing power, their limited screen size and resolution makes it quite hard to visualize a Web page content. In general, Web pages are designed to be visualized on larger screens and, when one attempts to fit it on a small-screen device, most of its content is not visible. To better visualize a Web page content, the following approaches can be taken : create pages specially designed for the device or adapt them whenever needed. In the latter case, the content of a Web page could be adjusted, using a proxy-server for instance, to meet the device needs. The Wireless Application Protocol (WAP) [5] and i-Mode [7] work on the idea of creating Web content tailored for mobile devices. This, however, has a major drawback as most of the Web content is not available in such formats.

Recently, a number of works have explored solutions to improve the ways mobile devices with limited resources display Web content [3, 4, 11, 2]. Among them, we can observe two different approaches: those who preserve the source

page layout [4, 2, 6], and those who do not [12, 10, 11]. OperaMini [10], is an example of the latter which linearizes the entire content of a Web page. Although it is simple to implement and does not demand much processing at the client side, it leads to a large amount of vertical scrolling. Buyukkokten et al. [11], proposed to reduce the source Web page textual content through the use of text summarization techniques. On the same line, Schilit et al. [12], analyze the original page and have its content mapped to a previously established layout interface. As the above approaches modify the source page layout, users may have difficulty finding the information they are looking for. It is worth mentioning that most Web portals arrange the information to in a way that provide easy access to the topics with higher relevance. Also, when the user is familiarized with the page layout, s/he probably knows where to look for the information s/he wants.

Clearly, when one attempts to fit an HTML page into a mobile device display, most of its content is hidden from the user as the page is usually larger than the mobile display. A common way to provide means for users to actually visualize the entire Web page on a mobile device is to create a preview image of the original page. In this context, Chen at. al. [4] proposed to split the source page structure into blocks and associate each block to the corresponding area in the generated thumbnail. In the same direction, the work in [2] focused on the identification and delimitation of a page's regions to provide a two-level hierarchy.

In this work, we propose an architecture, named Proteus for adapting web page on small-screen devices. Proteus takes into consideration user preferences, such as image rate compression, thumbnail generation, text summarization preferences, mobile display size and topics of interest. The user preferences are stored in a profile. When a client requests an HTTP page to the Proteus server, the page is adapted based on the user's profile, and then delivered to the client. As we will describe in latter sections, the Proteus's adapted pages assist the user to identify relevant information even in a source page preview.

The rest of this paper is organized as follows. In **Section 2** we present an overview of the proposed architecture and **Section 3** presents the implementation details. Preliminary results are shown in **Section 4** and conclusions are drawn in **Section 5**.

2 Proteus Architecture

Proteus architecture is based on a client/server model. In our case, the client's function is basically to display the information provided by the server and implement ways to navigate through it. The benefits of using a proxy server include: (*a*) Better use of the client's battery resources; (*b*) Help reducing the communication cost and download time; (*c*) The adapted pages are well formed; (*d*) Allows to cache adapted pages for latter retrieval. In addition, with the use of a proxy server, it is possible to provide the user the possibility to inform parameters that will be analyzed and used during the treatment of the requested pages. For instance, the user can customize how the Web content will be delivered to his/her device, allowing the user adjust parameters to better suit his/her needs.

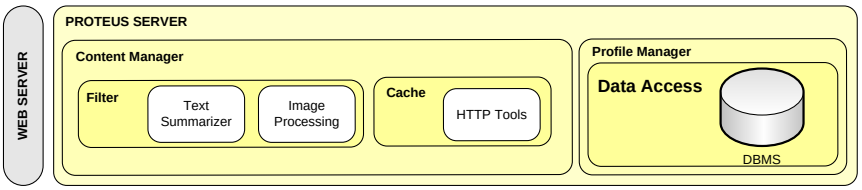

Fig. 1. The Conversion Server Architecture of Proteus Project.

The works proposed in [12, 2, 4] also make use of a proxy server. The model presented by Schilit et al. [12] provides a type of conversion through a proxy known as *m-link service*. However, the provided service neither foresee the reuse of converted pages nor considers the users' preferences. The works in [2, 4] only mention the possibility of using a proxy server but do not show its implementation details. The Proteus project's architecture is shown in Figure 1. In what follows, we will present in more details the component and services implemented by this architecture.

2.1 Web Page Analysis

On receiving a client request, the server tries to fetch the source HTML. When the page is successfully retrieved, the *content manager* parses the HTML to validate its code. In this phase, unnecessary information, like code remarks, is removed to avoid sending the user irrelevant data. After the validation, the page is represented in memory using the DOM (Document Object Model) structure [1]. After this preprocessing, co-related page blocks are identified, in a process similar to the one proposed in [2]. Next, each page block is analyzed and processed according to the user's profile.

2.2 Profile Manager

The *profile manager* is the module responsible for registering and managing the user profile. The profile information is used by the *content manager* during the page conversion process. Currently, the user profile can be accessed through the Web, where the user fills in an electronic form. In the form, the user customizes and selects the services s/he wishes to include in the conversion system. Among the conversion possibilities, we can cite:

- **Figures** - The user can choose to visualize only the textual contents. In this case, all the figures are withdrawn from the HTML source page;
- **Compression Rate** - Informs the compression rate that will be applied to the pictures in a Web page. This option will be disabled in case the user chooses not to receive figures;
- **Text Summarization Rate** - All text blocks of the source page will be summarized according to the stipulated rate;

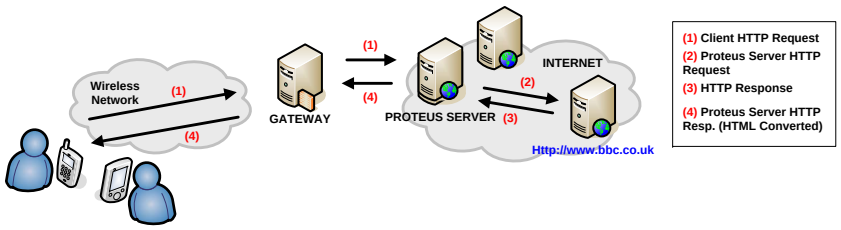

Fig. 2. Data flow involved from the moment the client makes a request until the answer is provided by the server.

- **Content Highlight** - Highlights page regions that matches the user interests or preferences;
- **Subjects of Interest** - The user can inform the subjects of interest, which will be used by the content manager to identify the relevant areas on the accessed pages;
- **Thumbnail** - The converted page can be sent to the client as a thumbnail or conventional HTML text;
- **Diplay Size** - The size of the visualization area on the mobile device. This informartion is used to adapt requested pages to fit the user's device.

Based on the data provided by the user, the system will create an unique key that identifies which type of conversion should be performed by the content manager.

2.3 Content Manager

The *content manager* is responsible for the conversion, storage and retrieval of the converted pages. It is composed by a cache system and a filter system. After the identification and division of the page into blocks, the content manager uses the *filter system* to modify the blocks according to the user's profile. Concerning the content manager attributions, we can cite, among others: (a) Text summarization; (b) Identification of the user's degree of interest in each block; (c) Image resizing; (d) Identification and registration of new subjects of interest.

On a system that provides conversion services, it is important to implement strategies for conversion reuse, aiming to improve its performance. In our architecture, already converted pages are cached, which makes it possible to quickly retrieve them. More details concerning the content manager implementation and the techniques adopted will be presented in Section 3.

2.4 Data Flow

Figure 2 shows the data flow from the moment the user makes a request until an answer is provided by the server. Initially, the client submits an HTTP request to

the Proteus Server for a specific URL. In case the client has a registered profile, his/her profile ID is sent, as cookie, along with the HTTP request. Otherwise, a standard profile is used. Suppose that the user issuing the HTTP request has a profile which specifies that a thumbnail should be generated from the requested page. At the time that the request is received, the server checks with the content manager whether there is a converted page according to the URL and the size of the user's mobile device display. If there is a hit, the thumbnail is retrieved, and the areas of user's interest (according to his profile) are highlighted and then sent to the client (the way the areas of interest are located in the stored thumbnail will be detailed in the next section). Otherwise, the server tries to fetch the source page. In case of success, the server goes on to analyze and convert the page, generate the thumbnail, and highlight areas of interest that matches the user profile. When the source page cannot be fetched, the server sends an appropriate message the client.

3 Implementation Details

3.1 Summarization

Recall that the Proteus goal is to adapt Web contents on-the-fly to better fit into a mobile device display. Hence, we considered summarization techniques with the following characteristics: language independent, fast, and able to generate summaries that express, with a reasonable degree, the essence of the original text. With that in mind, in this work we have focused our attention on *extraction* techniques. Among the existing extraction techniques, we have selected the *Keyword* [8] and *Term-Frequency Inverse Sentence Frequency – TF-ISF* [9].

In our tests, Keyword performed better with short texts while TF-ISF generates better results on larger inputs. In order to summarize the content of a Web page, we first have to parse the HTML file and extract its textual contents. It is important to note that HTML Tags may provide significant information about its contents as well. So, the text found in certain tags are retained during the parsing phase. Also, it is important to mention that words having the same radical should be counted as the same word. However, for doing so, it is necessary to find its root, thus eliminating prefixes, suffixes, and considering gender, number, tense, and case. For each text to be summarized, we first identify the number of words in the text. Based on this value, either Keyword or TF-ISF is used. With the help of the Keyword and TF-ISF, it is possible to allow users to define the desired degree of summarization. Also, as we are able to extract keywords from the accessed Web pages, one can use such information to enhance the accuracy of the system, as it will be shown latter. We will not go deeper into text summarization and extraction techniques here, for further details, we refer the reader to the references provide above and therein.

3.2 Web Page Content Mapping

Our architecture allows users to receive the requested HTML pages either in HTML format or thumbnail image. If an HTML format is to be received, the

source page is adapted according to the information stored in the user's profile. When a reduced image is to be received, a thumbnail of the source page is created. In either case, while parsing the requested HTML page, a *blueprint* of the page is extracted. The blueprint will hold information concerning the source page structure, such as:

- Location of page blocks.
- Block content description.

The blueprint is cached along with the source HTML page. When a thumbnail is to be returned, the server uses the blueprint to identify the blocks that match to the user profile. Such blocks are highlighted to inform the user that its content matches the user preferences. According to the degree at which the contents match the profile, different colors, or shades, can be used to express the degree or relationship between the block and user preferences. When the user selects the block, its associated information is brought to the user. That is, on clicking on a thumbnail region, the HTML content of the region is displayed. This approach gives a two-level visualization: a reduced image on a first level, and the associated HTML content on a second one. Also, the user may define a rate for text summarization. In this case, the second-level will show a summarized text according to the rate defined in the user profile. The summarized text provides a link to see the entire content. Note that this latter approach introduces a three-level visualization.

3.3 Profile

In our architecture, each device has a profile which is stored on the server. The profile can be created and accessed via a mobile device itself or any another terminal. In any case, the user receives a key which shall be used when accessing the Proteus Server. The key will identify the device and users preferences, which will affect the ways the Web pages and its contents will be delivered to the mobile terminal. In case the user has no profile in the server, a standard one is used.

In the profile the user can define if s/he wants to receive the Web page in either thumbnail or HTML format. If s/he selects the thumbnail format, the navigation on the page relies on a two level visualization where the first level is the reduced image and the second level is the associated HMTL contents. The profile still permits the user to define the summarization rate of the text regions on the second level, introducing in fact a third layer of visualization. The profile also allows a user to specify the categories s/he is interested in, such as business, travel, etc. The Proteus Server will then create an association among the specified categories and the relevant keywords that match the selected categories. For example, consider a user with a previously stored profile connecting to the Proteus Server to request an HTML file. In this case, the requested page is treated by the server to identify regions, summarize text, generate the thumbnail and create the mappings (assuming these options match the user profile). Since the text summarization works by ranking keywords, if a match among the ranked

keywords and the categories is found, the associated block receives a higher rank. We call this matching process as *static content matching*. In contrast to this, we also provide a *dynamic content matching*, which is described bellow.

In the dynamic content matching, the keywords that have obtained higher ranks during the summarization step, are stored in a database called *Dynamic Content Database – DCB*. When a user makes an HTTP request through the server, the system checks the higher ranked keywords on that page against the keywords stored in the DCB. If there is a match, the blocks containing the keywords are ranked higher, as in the static content matching. To prevent old keywords to have an impact on current pages, each keyword in the database is associated with a time stamp T_s, which is updated according to the user requests for Web content. When T_s expires, the associated keyword is removed from the DCB. Another point worth mentioning is that the DCB should not grow above a certain threshold as this has an impact on the server workload. For this reason, we have devised the following policy to update the DCB.

Let K_i denote the set of keywords stored in the DCB which are associated with the user profile P_i, where $0 < i \leq n$, and n denotes the number of users. Also, $|K_i| \leq \delta$, where δ is the maximum number of keywords allowed per user in the DCB. Now, suppose that a new set of keywords, call it S_i, have been obtained for user U_i. Then, the keywords satisfying $K_i \bigcap S_i$ have their time stamp T_s renewed in the DCB. The keywords in S_i may be incorporated in DCB up to the threshold δ.

3.4 Cache Module

As the mobile clients make HTTP requests to the server, the server caches information so that latter requests can be served in a faster way whenever there is a cache hit. As the mobile devices have different screen-size and resolution, we have organized them into categories, according to their characteristics. When a user requests to receive a preview of the original page, the generated thumbnail is stored in the appropriate category. Thus, when a user requests a page that has a cached thumbnail, the server checks whether the thumbnail matches the device category. If affirmative, the cached thumbnail is returned. Otherwise, a new thumbnail is created and returned. The Proteus cache module uses the LRU (Least Recently Used) replacement policy, which means old and least used cached information is removed when necessary.

4 Preliminary Results

4.1 Mapped Regions

Our first example illustrates how Proteus identifies the contents in a requested page that matches the user preferences. In this example we focus on static contents only. The user profile for this example is shown in Table 1. The profile shows that the user wishes to create a thumbnail of the original page, in a way

Description	Value
Keep Figures	Yes
Compression Rate	58%
Summarize Content	0%
Subject / Keywords	Games, Business
Highlight Content	Yes
Create Thumbnail	Yes
Size of the Display	470x770 pixels

Table 1. Example of profile specified by the user.

that is applied to the figure a compression rate of 58%. Also, we can verify that the user has interest on subjects related to Games and Business. Using the keywords contained on the profile, the conversion server is capable of identifying, on the HTML page, which regions are of interest to the user. The identified regions are highlighted so that the user knows that they have information that matches his interests. By clicking over the region, the associated HTML is shown. The HTML content of other regions are reached in the same way.

Figure 3(a) shows the result obtained when the CNN site is submitted to Proteus Server under the profile presented in Table 1. The Proteus Server has identified four areas related to games (delineated in dashed-lines rectangles) and the two areas related to business (delineated in solid-lines rectangles). In the menu bar of the site, the option Business is circled. This options was ignored because it is located in the menu region, which is identified during the page analysis. On the bottom-left of the page, the figure Business2.0 was selected by the algorithm because its ALT tag contains a valid text which was identified. From the thumbnail, the user is able to see the whole page. However, s/he may not be able to identify, on a first level of visualization, the information that s/he is looking for. Highlighting areas of interest is a way of restraining the search area and avoid unnecessary zoom.

4.2 Summarization Results

This sub-section presents a combination of the region mapping and summarization. The profile selected for this example is shown in the Table 2. Here, the user requested an article published at the CNN website under the title: "MAC fans clamor for iPhone". The thumbnail result is shown in Figure 3(b), where the area delineated in dashed-line rectangle was identified by the content manager as an area of user interest. The highlighted text is composed of 789 words. By choosing this area, the content summarization will be applied. The result that is sent to the user is represented by the frame at the top of the page thumbnail. It contains an abstract with 21,68% of the original text. Should the user require the entire (original) text, s/he can access it via a link *more*, which is appended by the server. This approach is similar to the use of RSS feeds, in which a brief text is shown to allow the user to have a glance of the link contents.

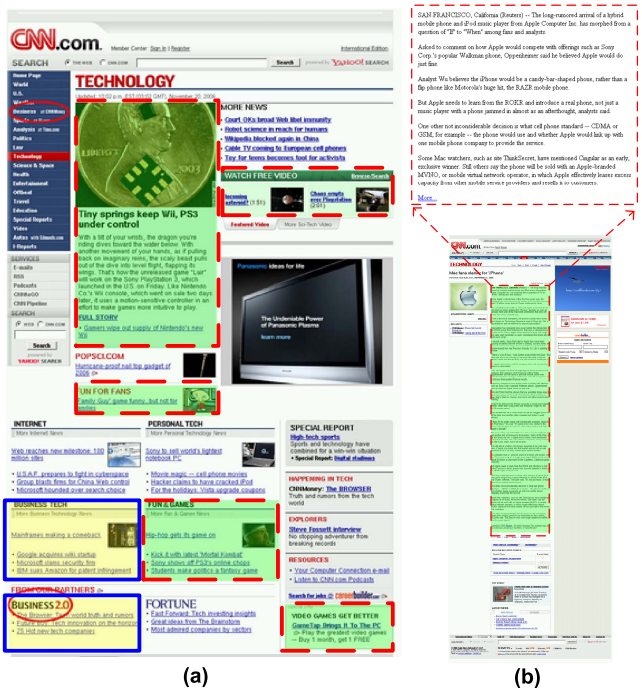

Fig. 3. (a):CNN website created from the profile presented in Table 1. b): Summarized text aaccessedfrom the Web page tthumbnail

5 Conclusions and Future Works

Up to this point, we have made a number of experiments, however, we have not made our architecture available to the general public. Before doing that, we plan to test our architecture with a selected number of users, so that their feedback may help us to fine tune and improve the system. Also, we are currently developping a mobile web browser that will have features to improve the ways the thumbnails are presented to the user. We are also considering the use of Web mining techniques and the extraction of semantic information of HTML pages as a mean to provide the possibility to find related contents on a page and suggest them as areas of interest to the user.

References

1. V. Apparao, S. Byrne, M. Champion, S. Isaacs, I. Jacobs, A. L. Hors, G. Nicol, J. Robie, R. Sutor, C. Wilson, and L. Wood. Document object model (dom) level 1 specification (second edition). Technical report, W3C, September 2000.
2. D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Vips: a vision-based page segmentation algorithm. 2003.

Description	Value
Keep Figures	Yes
Compression Rate	87,70%
Summarize Content	80%
Subject / Keywords	Technology
Highlight Content	Yes
Create Thumbnail	Yes
Size of the Display	124 x 320 pixels

Table 2. Example of the profile specified by the user. The converted page is shown in Figure 3.

3. Y. Chen, W.-Y. Ma, and H.-J. Zhang. Detecting web page structure for adaptive viewing on small form factor devices. pages 225–233, 2003.
4. Y. Chen, X. Xie, W.-Y. Ma, and H.-J. Zhang. Adapting web pages for small-screen devices. *Internet Computing, IEEE*, 9:50–56, February 2005.
5. R. Gellens. Wireless device configuration (otasp/otapa) via acap, 1999.
6. Zhigang Hua and Hanqing Lu. Web Browsing on Small-Screen Devices: A Multi-client Collaborative Approach. *IEEE Pervasive Computing*, Vol 5. Nro2, pp. 78–84, 2006
7. Mobell. What is i-mode? Disponvel em: <`http://www.mobalrental.com/imode.asp`>. Accessed in 11/09/2006.
8. P. H. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2:159–165, 1958.
9. J. Larocca Neto, A. D. Santos, C. A. A. Kaestner, and A. A. Freitas. Document clustering and text summarization. In *Proceedings of the 4th International Conference on Practical Applications of Knowledge Discovery and Data Mining*, pages 41–55, London, 2000.
10. Opera. Opera Mini. Disponvel em: <`http://mini.opera.com`>. Accessed in 11/09/2006.
11. O. Buyukkokten, O. Kaljuvee, H. Garcia-Molina, A. Paepcke, and T. Winograd. Efficient web browsing on handheld devices using page and form summarization. *ACM Trans. Inf. Syst.*, 20(1):82–115, 2002.
12. B. N. Schilit, J. Trevor, D. M. Hilbert, and T. K. Koh. m-links: An infrastructure for very small internet devices. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 122–131, New York, NY, USA, 2001. ACM Press.