Use of RSS feeds for Content Adaptation in Mobile Web Browsing

Alexander Blekas
University of Patras
Computer Engineering & Informatics Department
26500 Patras, Greece
Computer Technology Institute
26500 Patras, Greece
+306932321212
mplekas@ceid.upatras.gr

John Garofalakis
University of Patras
Computer Engineering & Informatics Department
26500 Patras, Greece
Computer Technology Institute
26500 Patras, Greece
+302610997866
garofala@ceid.upatras.gr

Vasilios Stefanis
University of Patras
Computer Engineering & Informatics Department
26500 Patras, Greece
Computer Technology Institute
26500 Patras, Greece
+306973281930
stefanis@ceid.upatras.gr

ABSTRACT

While mobile phones are becoming more popular, wireless communication vendors and device manufacturers are seeking new applications for their products. Access to the large corpus of Internet information is a very prominent field, however the technical limitations of mobile devices pose many challenges. Browsing the Internet using a mobile phone is a large scientific and cultural challenge. Web content must be adapted before it can be accessed by a mobile browser. In this work we build on the proxy server solution to present a new technique that uses Really Simple Syndication (RSS) feeds for the adaptation of web content for use in mobile phones. This technique is based in concrete design guidelines and supports different viewing modes. Experimentation shows a significant decrease in the transformed content of about 80% in size facilitating cost-effective web browsing.

Categories and Subject Descriptors

H.4.3 [**Communications Applications**]: Communications Applications - Information browsers; H.5.2 [**Information interfaces and presentation**]: User Interfaces. - Graphical user interfaces

General Terms

Algorithms, Performance, Design, Experimentation, Human Factors.

Keywords

Mobile devices, RSS feeds, content adaptation, mobile browsing, web browsing.

1. INTRODUCTION

During the past decade the world has witnessed a revolution in the field of wireless networks and mobile devices. The term 'mobile device' refers to a device specially designed for synchronous and asynchronous communication while the user is on the move. It includes a wide variety of appliances such as PDAs and mobile/smart phones. Tablet PCs are also included in the category of mobile devices, however they are bigger in size and more costly to acquire. Mobile phones are by far the most popular mobile devices. Among the many facilities they provide is access to the Internet, however browsing using a mobile phone is not as easy as browsing using a common desktop PC. Screen size and resolution, number of supported colors, entering text method, computation power, memory size, rate of data transfer and energy required for proper functionality are the main limitations for using a mobile device to browse through the Internet [7]. On the other hand, users demand access to information anytime, anywhere. Since within a few years, most of the devices accessing the Web will be mobile, there is a need for developing methods and techniques that will allow satisfactory web browsing using mobile devices.

The problem of web browsing through a mobile phone has attracted much attention by the research community and the industry alike. One of the proposed solutions is to use a proxy server which adapts the web content for mobile devices on-the-fly and returns the result to the mobile user. In this paper we build on the proxy server idea to propose a novel solution that makes use of feeds in a web site in order to achieve better content adaptation. RSS, an XML-like notation, is used mainly in large information-centric web sites to summarize web site information. By taking advantage of this information, it is possible to transform the content of the regular web site so that it can be accessed by a mobile phone. The new technique is able to scan any web site that includes RSS, removes 'unwanted' information and presents a set of different packed versions which are both comprehensive to the user and have a small size. The technique is available as an online system that works even for web sites without RSS feeds. Experimental results show a significant reduction of about 80% in the size of the web page processed using this method. Although this technique also removes multimedia information (images, video, animation) its value relies on the very small size of the adapted content, making it accessible even by mobile phones of limited capacity through heavy-loaded or slow wireless networks.

This work is structured as follows. In section 2, a review of content adaptation methods for mobile devices is presented. In section 3, RSS feeds are described while in section 4, the design principles for mobile applications are discussed. Finally, section

W4A at WWW2006, 23rd-26th May 2006, Edinburgh, UK

5 provides a complete presentation of the application that we have developed and in section 6 research suggestions and future work are presented.

2. CONTENT ADAPTATION FOR USE IN MOBILE DEVICES

There are four general approaches for adapting web content for small screen devices: device-specific authoring, multi-device authoring, automatic re-authoring, and client-side navigation [1]. The first two approaches obtain high quality results by authoring device-specific web content. Having two versions of the same web content (one for regular and one for mobile users) is costly and the vast majority of web sites does not use it. This makes most of the Internet actually inaccessible for mobile phone users. On the other hand, automatic re-authoring and client-side navigation techniques do not require the collaboration of page authors and are therefore more widely applicable.

Research prototypes that use automatic re-authoring fall into two main categories: page reformatting and page scaling. An example of techniques based on page reformatting is the Power Browser [3], where images and white space are removed, and the WEST browser [2], which uses flip zooming, a visualization technique that breaks pages into screen-sized tiles and presents them as a stack. Difficulties with recognizing layout and leveraging the desktop browsing experience are common to all these approaches, since they all have an impact on the page layout.

Other techniques based on page reformatting include scaling the page and return the result to mobile device as a thumbnail. An example of this approach is the SmartView system [8]. SmartView creates an image map thumbnail of the page based on its semantic content. The user is able to zoom to the preferred block of the thumbnail and read the content of the web page. The main problem with this approach is the size of the thumbnail; the screen size of mobile devices does not allow the text to be readable in many cases. Another approach tries to solve this problem by combining the previous method with text summarization techniques [6]. The main idea behind this approach is for the first level of the thumbnail to contain only some keywords from every text unit and not the full text of the web pages. When the user zooms to a specific area she/he is able to read the full text contained there. These methods are mainly intended for PDA users and generally for mobile devices with relatively large screens. Small mobile phone screens are not able to present a satisfactory view for the original thumbnail and as such they are not recommended to be used with the techniques mentioned above. The typical screen resolution of a mobile phone is between 128x128 and 176x220 pixels. A web page, with rich content, in a 1024X768 pixels resolution usually requires scrolling even for a 17 inch monitor. It is obvious that the thumbnail for the mobile phone will be so small that the user will not be able to read text, especially if he/she is a first time visitor (and thus unfamiliar with the web site content organization). Finally, using images obviously increases user satisfaction but requires significant computational and power resources from the mobile device.

In client side browsing, the whole content is delivered to the browser and then the browser decides how to present it. For example, the Opera Browser (opera.com) applies the Small Screen Rendering™ technique where the content is presented in a single column, so horizontal scrolling is no more necessary. All client side browsing methods require the full code of a web page to be downloaded to the mobile device. This presupposes the existence of enough main memory to the mobile devices for storing and running the page code. Furthermore, the user is usually billed according the amount of downloaded data, so downloading large content is costly.

3. RSS FEEDS

Our approach is associated with the automatic re-authoring combined with RSS feeds with the purpose to improve web browsing from mobile devices. RSS is a family of XML file formats which summarizes web site information. It is mainly used by news sites and, generally, sites that their content changes very often, for example a weblog (a web-based publication consisting primarily of periodic articles, normally in reverse chronological order). Currently, most web sites use one of the three different RSS versions; version 0.91 [9], 1.0 [10] and 2.0 [11]. The number of web sites that are RSS-enabled is increasing geometrically. According to feedster.com (a search and indexing engine for RSS feeds) more than sixteen million RSS feeds are available in the web including more than 75.000 professionally published sources such as the BBC, CNN and The New York Times [5].

Table 1. RSS feeds tags

RSS tag	Description
<rss>	Start RSS information
<item>	Chunks of summarized information
<ttl>	Time to live (in minutes). Denotes amount of time for which information is considered valid
<title>	The title of the information chunk
<description>	A small description of the information
<link>	A link to the actual information in the web page

Table 1 summarizes the functionality of the main RSS tags. Every RSS file has the root element <rss> where the version of the RSS file is defined. The only child of the <rss> element is the element <channel>. The element <channel> may contain any number of elements <item>, the main element in a RSS file. Every element <item> always contains the tags <title>, <link> and <description>. Beside the above tags, there are other tags supporting more functionalities, depending on the RSS version.

4. DESIGN PRINCIPLES FOR MOBILE APPLICATIONS

The limitations of mobile devices obligate developers to be very careful when designing applications. In order to design the RSS-enabled content adaptation technique, we had to follow certain guidelines [13], [14].

The interface of the application should be based on a consistent and easy to learn navigation model. Mobile users' requirements are different from PC users'. Mobile users do not browse the Web in the classic way but they demand quick access to specific information. Therefore developers of mobile applications should avoid content with a lot of graphics, animations and different colors. In any case they should present the right information at the right time.

Another point that we have to take care is text inputting. For the majority of mobile users, using a keyboard of a mobile phone, even with T9 [12] support, is a slow process. Interaction with the application requires the use of checkboxes, lists and drop down menus. Furthermore, if a page contains forms it should present the forms in one screen so that the user is able to confirm data input without returning to previous screens.

Developers also have to pay attention to navigation hierarchy. Finding services should require only a few clicks to hyperlinks. If the navigation model is too complex (e.g. a deep tree of hyperlinks), the user may get lost and frustrated. Developers also need to avoid creating buttons for functions already implemented by the system. For example, if we create hyperlinks for going to the previous page in a mobile web application, a function already implemented by the browser, valuable space is misused in an already small screen; on the other hand, special buttons are needed so that the user can browse back to the main page or other important pages in a click.

For mobile web applications, grouping the hyperlinks effects the usability of the application. For example, if we have to present twenty relative hyperlinks, it is preferred to present them sorted in the same long page and not to four small pages. In every page the important content and the navigation bar if it exists, has to be located at the top of the page. So, if the user browses to a page which has no important content according to his/her interests, she/he can continue to the next page without vertical scrolling. Finally, pages that welcome the user to the application are unnecessary (and some times frustrating) for mobile users.

A designer of mobile web applications must have in mind that the applications refer to devices with small screen but at the same time these applications should be also user-centric; reaching a consensus there is difficult. Every page has to contain the main content starting from the beginning. The need for vertical scrolling should be kept to a minimum; horizontal scrolling is generally not recommended at all. Every modified web page should contain the tag <title> so that the user may read a short description of the page at a glance. The browsers usually put the content of the <title> tag to a visible place. The title should have a length of fourteen characters, at the most.

The designer has to use the different types of text alignment (right, center, left) in a way that facilitates information grouping and makes the text more readable. Hyperbolic formation of the text like bold or underline letters may bring the opposite results; finally, it is proposed to replace complex and big words because they make sentences longer and the user may have to scroll in order to understand the meaning.

Color usage is also important. Using colors obviously gives a pleasant and friendly interface, but a too colored screen confuses. All the pages of the application must have the same colors so the user can feel that he/she is navigating in the same environment. Furthermore, it is proposed to avoid sentences that refer to colors by their name, such as “Click to the purple link to continue”, because some users may have devices with screens that support few colors.

The use of images in Internet applications is common. Nevertheless, using images in mobile web applications significantly increases download and response time and thus, usage cost. If the browser supports incremental page rendering, a technique where the browser incrementally presents a web page before the whole content has been downloaded, the loading of images in a later time needs more computational power and confuses the users. For the above mentioned reasons we argue that the use of images should be kept to a minimum.

5. AN RSS – BASED CONTENT ADAPTATION BROWSING SYSTEM

5.1 General Architecture

A standard way of processing web pages for viewing on small screen devices is through a proxy server that transforms pages on-the-fly. A proxy server is a program that receives web page requests (here from mobile devices), loads the respective pages, converts them, and serves them to the devices that requested them. In this way the proxy server, which usually runs on powerful servers, unleashes mobile devices from computational needs.

Figure 1 depicts the functional architecture of the system. If the system detects RSS feeds in a web page, it is proposed to the user to use the RSS feeds for the browsing. If the user agrees and chooses one of the available RSS feeds, the system reads every <title>, <link> and <description> element from the feed, adds the application’s menu and returns the XHTML – MP content to the user. If the page has no RSS feeds or the user does not want to use them, the proxy servers adapts the web content for mobile devices, adds the application’s menu and returns the XHTML – MP content to the user as described previously. The technique used to detect and exploit the RSS feeds is presented in the following section.

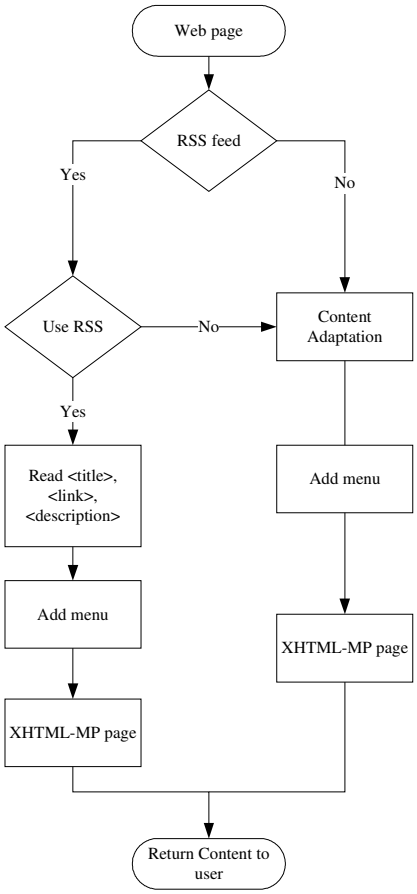

Figure 1. The functionality of the general RSS adaptation mechanism.

5.2 The RSS-feed adaptation technique

There is no need for configuring the device prior to the use of the application. The user has to start the browser of the device and provide the url of the server that hosts the proxy server application. The content that will be returned to the user is the main XHTML – MP page of our application, total size 0.65 KB (figure 2). In the text box of the main page, the user provides the url of the web site he/she desires to browse. This is the only part where the application needs text inputting. After this step, the user uses his device browser's interface. When the user hits the "GO" button, the proxy server creates a connection with the given url, gets the html code, transforms it and returns the new code to the mobile device. Details like the "http://" string in the url does not concern the user, since we want to achieve the minimum keyboard usage. Also, if the web server redirects the original url (code 302 at HTTP response) the application finds and follows the new url without disturbing the user.

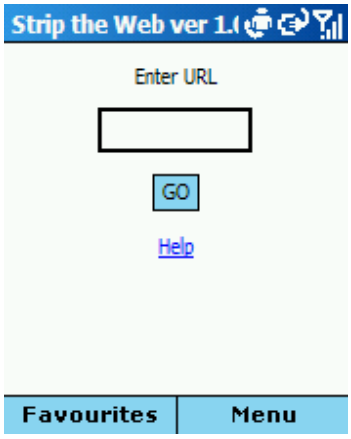

Figure 2. Main page of the application.

The proxy server has now saved the html code of the web site the user had requested. In this point, a parsing algorithm remove tags that are not consistent with the design guidelines of section 4. Those tags are <script>, <noscript>, <style>, <link>, <iframe>, <object> and <embed>. The algorithm also removes the comments from the html code. Comments do not affect the final result because browsers ignore them. But if the final code includes comments, the mobile user pays for useless data. Forms are also removed. This action decreases the usability of the application in many sites since it suspends the use of searching, login, on line transactions and generally web applications that involve interaction of this type with the user. However, the purpose of the system is focused on cost effective browsing and not facilitating more complicated on-line transactions so, in our view, this is acceptable. Furthermore, the majority of mobile users browse the Web for news and entertainment. Therefore, if one user is interested in mobile transactions for example, he will use the special application that the bank may offer and not a proxy server of this type. Furthermore, our application focuses on delivering to mobile users the content of a web page as good as possible and not to implement the functionalities of a random web site from mobile devices.

The use of tables in the web is a common practice because they summarise content elegantly. However, in mobile devices the use of tables is a problem because of the lack of space on the screen. The application solves this problem by removing tables and presenting their content in a new line for every cell. If nested tables in a cell are used, the procedure is applied recursively.

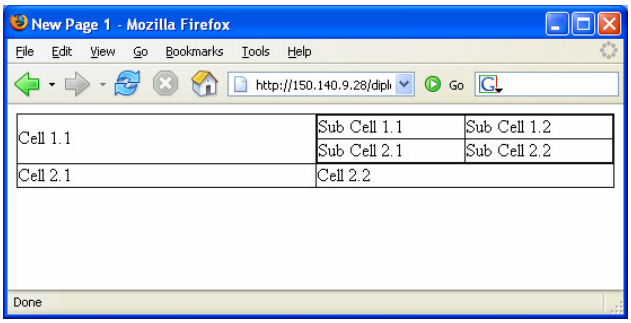

Figure 3. A web page with nested tables.

Figure 3 shows a 2X2 html table as presented by a web browser. A 2X2 table is nested in one cell of another table. If the user requests the above page through the proxy server, the result is presented in figure 4.

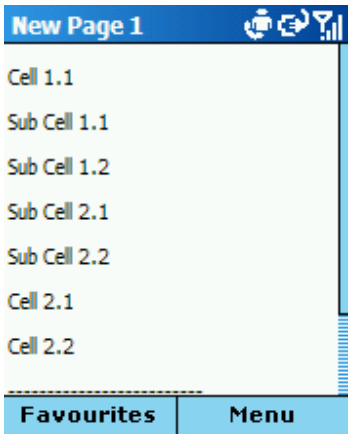

Figure 4. Nested tables as showed from the application.

For further processing of the code, we need some knowledge about its structure (for example if one tag is nested in another). For this purpose, we use the functions of the DOM library of PHP. With DOM functions we create the DOM tree of the web page. The DOM interface of PHP follows the DOM Level 2 standard [4].

The next step is to remove the images from the web page. From the DOM tree that has been created, we locate all <img> tags by using the getElementsbyTagname PHP function. If the image is at the same time a hyperlink to another page, removal of the image must be done in such a way that hyperlink information is not lost. To ensure this, we check the DOM tree. If node <a> has a node <img> as a child, in the original place of the image, a text hyperlink is created. The text of the hyperlink is the string of the alt attribute of the original <img> tag. If the alt attribute is missing or is empty, the new hyperlink contains the text "IMG:" followed by the url of the hyperlink. To read the value of the alt attribute

we use the getAttribute attribute of the class DOMElement. When the new hyperlink has been created, the <img> tag is removed from the code using the removeChild function.

One of the design goals of our approach is also to ensure transparency. When a page has been served to the user, she/he can browse to new pages by following the hyperlinks or by giving a new url to the main page of the application. So, the destination of each hyperlink in the transformed web page must change in a way that requests are redirected through the proxy. For example, if a hyperlink points to http://www.acm.org the new hyperlink must be change to the following format:
http://proxy_address/main.php?url=http://www.acm.org.

Of course, a hyperlink may point to things other that web sites. For example, a hyperlink may point to a file that is hosted at the same server using a related path such as Test page. In this case, the hyperlink points to the file test.html which is located at the directory which is a level higher from the current one. The proxy server that we have developed checks all those cases.

When the parsing algorithm has finished with the content adaptation process, it checks the size of the new page. If the size is bigger than 10KB then it sends, through the proxy, the content in sub-pages with size of about 10KB. If the point where the page has to split is in the middle of a paragraph, the splitting point is moved so that the whole paragraph appears in the same sub-page. The 10KB limit may seem small for new mobile devices however, with this limit we are sure that there will be no memory problems with any device that supports WAP 2.0. One other reason for creating sub-pages is that larger pages may lead to longer response times. Furthermore, a page may contain content which is useless to the user or may be an intermediate page for another destination. In this case, the user may ignore this page having paid only for the 10 KB of the first sub-page. The 10 KB limit can be easily changed or cancelled by the administrator of the proxy server.

At the end of each page or sub-page a menu is added that allows access to the other sub-pages that may exist or returns to the main page of the application to view only the links or only the text of a page. These functionalities are described in detail later in this section. The menu is at the end of each page but is quickly accessible from the user by pressing the 0 key in the keyboard. This function is implemented using the attribute accesskey at the tag <a>.

When the user is navigating in a site she/he usually browses through hyperlinks ignoring most of the content, that is until the proper information is found. To speed up this process the application provides the user with the option to view only the hyperlinks of a page. Similarly, the user has the option to view only the text of a web page. This functionality may be useful if a page contains for example, an article. Therefore, text is the most important part in this case. Of course, the user may switch from one mode to another or return to the original viewing mode.

If the user requests a page that includes RSS feeds then the application does not present the content of the page immediately but recommends the user to browse the site through the information that the RSS feed contains. By default, the RSS feed includes all the latest and interesting information that the site contains. In order to discover if a page contains RSS feeds, the parser checks all the <link> tags before they are removed from the content adaptation process that we described previously. If the attribute type of the tag <link> has the value "application/rss+xml", then the attribute href of the same tag contains the location of the RSS feed's XML file. The parser returns the title of the feed as it is defined by the attribute title of the tag <link>. If a page contains more than one RSS feeds, the parser returns all the feeds in the order that they are detected in the web page code. The application recommends the user to continue browsing through the available RSS feeds. However, the user can choose to browse the site through the process that we have described previously. Table 2 contains the PHP code that locates the RSS feeds to the HTML code.

Table 2. PHP code for RSS feeds detection

```
$rss_check = new DOMDocument();
$rss_check->loadHTML($code);
$tags = $rss_check->getElementsByTagName('link');
$rss_count = 0;
foreach ($tags as $tag)
{
        $type = $tag->getAttribute('type');
        if ($type == 'application/rss+xml')
        {
            $rss_urls[$rss_count] = $tag->getAttribute('href');
            $rss_titles[$rss_count] = $tag->getAttribute('title');
            if(stripos($rss_urls[$rss_count],'http') === FALSE)
              $rss_urls[$rss_count]= $url.$rss_urls[$rss_count];
              $rss_count++;
        }
}
```

If the user selects to browse based on RSS feeds as recommended, an XML parser is used to read the RSS feed. For the implementation of the RSS parser we use the xml_parse_create function from the XML library of PHP. The XML parser works with all three versions of RSS feeds, which are versions 0.91, 1.0 and 2.0. From every RSS feed, the application returns the title, the description and the link for every element <item> of the feed. Just like in every page, users may only view the links or the text of the page. Table 3 presents the code that implements the RSS feed reader.

Table 3. PHP code for the RSS reader

```
$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement",
"endElement");
xml_set_character_data_handler($xml_parser, "characterData");

function startElement($parser, $name, $attrs) {
  global $insideitem, $tag, $title, $description, $link;
  if ($insideitem)
  {
    $tag = $name;
  }
  elseif ($name == "ITEM")
  {
    $insideitem = true;
  }
}
function endElement($parser, $name) {
```

```
  global $code;
  global $insideitem, $tag, $title, $description, $link;
  if ($name == "ITEM")
  {
            $code .= sprintf("<p><b><a href='%s'>%s:</a></b>",
            trim($link),htmlspecialchars(trim($title)));
            $code.=
sprintf("%s</p>",htmlspecialchars(trim($description)));
            $title = "";
            $description = "";
            $link = "";
            $insideitem = false;
  }
}
function characterData($parser, $data) {
  global $insideitem, $tag, $title, $description, $link;
  if ($insideitem)
  {
    switch ($tag)
    {
      case "TITLE":
        $title .= $data;
              break;
      case "DESCRIPTION":
        $description .= $data;
        break;
          case "LINK":
            $link .= $data;
            break;
          }
  }
}
```

We have developed our server using PHP (php.net). The only requirements for the application is a web server with PHP 5 support or newer. All the functions that have been used are included to the default installation of PHP. For our tests we have used the Apache 1.3.33 as a web server and the PHP 5.0.4.

From the client side, the only requirement is a WAP 2.0 compatible browser, a browser that most of the devices today have. The pages of the application are XHTML-MP valid, so we are sure that the content the proxy server sends is presented correct in every category of mobile devices.

5.3 Examples of usage

In the next two paragraphs we present two examples of how our system works with two popular sites. The first site contains RSS feeds and the second does not.

5.3.1 The site cnn.com

In figure 5, we can see the main page of cnn.com as viewed from a web browser. The main page contains RSS feeds, that we have already loaded at the "Bookmarks" menu. The total size of the page is 313 KB. If we try to browse the same site from our application the result is shown in figure 6. At first, the application locates the existence of two RSS feeds, "CNN – Top Stories" and "CNN – Recent Stories". The user may ignore the RSS feeds by choosing "Normal Page". In our example we have selected the first RSS feed which is loaded to a new page. As we can see, the user has the most important content of cnn.com in about 12 KB.

Figure 5. The main page of cnn.com.

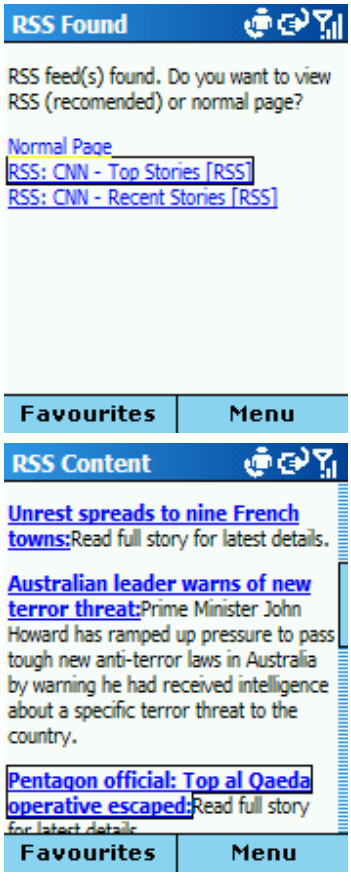

Figure 6. The site cnn.com using RSS feeds.

5.3.2 The site acm.org

As previously, figure 7 shows the main page of acm.org as presented to a web browser.

Figure 7. The main page of acm.org.

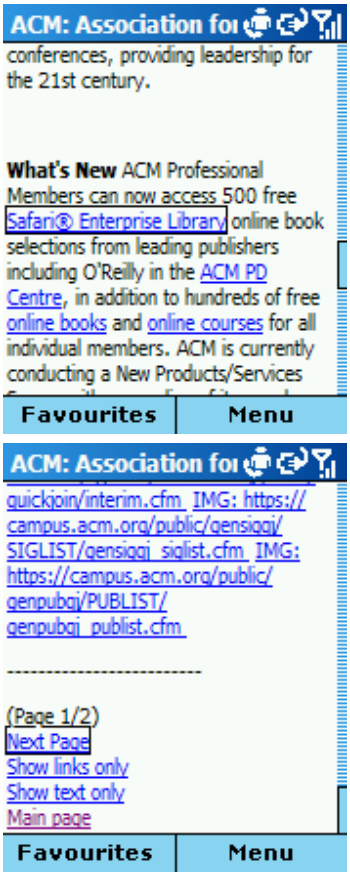

Figure 8. The main page of acm.org from the application.

The acm.org does not contain RSS feeds. Therefore the application follows the content adaptation procedure. The original page is 100 KB. The same page as generated from the application, which is at most 20 KB, is shown in figure 8. The mobile user may access the same content with a minimum benefit of 80% concerning downloaded data. The last observation was confirmed by other experiments we made on different sites.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented a method that allows cost-effective web browsing for users of mobile devices. This method exploits mainly the existence of RSS feeds, metadata that summarise information. We concluded that the existence of RSS feeds improves significantly the content presented to the mobile user decreasing at the same time the size and access cost.

The final result in a random site, through our proxy server, depends entirely on its content. The technique is not suitable for accessing on-line transaction systems since forms are omitted. Finally, the fact that the application removes all the images, may not take full advantage of the capabilities of newest mobile devices. All the above matters will be included in our future work.

7. REFERENCES

[1] Bickmore, T., and Schilit, B., Digestor: Device-Independent Access to the World Wide Web. In Proc. Seventh Intl. WWW Conference 1997, pp. 655–663.

[2] Bjork, S., Bretan, I., Danielsson, R., and Karlgren, J. WEST: A Web Browser for Small Terminals. In Proc. UIST'99, pp. 187-196.

[3] Buyukkokten, O., Gracia-Molina, H., Paepcke, and Winograd, T. Power Browser: Efficient Web Browsing for PDAs. In Proc. CHI 2000, pp. 430-437.

[4] Document Object Model Specifications, http://www.w3.org/DOM/DOMTR.

[5] Feedster.com statistics, http://feedster.blogs.com/corporate/overview/index.html.

[6] Heidi, L. and Baudisch, P. Summary thumbnails: readable overviews for small screen web browsers. In Proc. SIGCHI 2005, pp. 681 – 690.

[7] Kaikkonen, A., and Roto, V., Navigating in a mobile XHTML application. In Proc. SIGCHI 2003, pp. 329 – 336.

[8] Milic-Frayling, N. and Sommerer, R. Smartview: Enhanced document viewer for mobile devices. Technical Report MSR-TR-2002-114, Microsoft Research, Cambridge, UK, November 2002.

[9] RSS version 0.91 Specifications, http://my.netscape.com/publish/formats/rss-spec-0.91.html.

[10] RSS version 1.0 Specifications, http://web.resource.org/rss/1.0/spec.

[11] RSS version 2.0 Specifications, http://blogs.law.harvard.edu/tech/rss.

[12] T9.com http://www.t9.com.

[13] XHTML Guidelines For Creating Web Content, 2005, Nokia Corporation.

[14] XHTML Mobile Profile and CSS Reference, 2003, Openwave Systems