Journal of Web Engineering, Vol. 10, No.1 (2011) 021-047

OFFLINE WEB BROWSING FOR MOBILE DEVICES

YUNG-WEI KAO

Department of Computer Science and Information Engineering, National Chiao Tung University, Taiwan

ywkao@cs.nctu.edu.tw

TUNG-HING CHOW

Department of Computer Science and Information Engineering, National Chiao Tung University, Taiwan

kenn.iit96g@nctu.edu.tw

SHYAN-MING YUAN

College of Computer & Informatics, Providence University, Taiwan

Department of Computer Science and Information Engineering, National Chiao Tung University, Taiwan

smyuan@cis.nctu.edu.tw

Received May 23, 2010
Revised December 10, 2010

Based on the advancements of mobile device, mobile platform, and wireless network technology, browsing Web pages on mobile devices have become more popular. However, with its difference from the Web browsing behaviour on desktop, mobile Web browsing suffers from more environmental challenges. Traditionally, Web pages can only be viewed with stable telecom or wireless network connection. In recent years, Google Gears has been proposed to enable the offline Web browsing on mobile devices. However, the Google Gears mechanism can only be used with the server-side library supported by Web Servers. The authors proposed a Web content middleware with personalized interest list to support offline Web browsing on mobile devices, even though the selected Web servers do not support the server-side Google Gears mechanism. Finally, we compare other offline Web browsing solutions with ours and evaluate the offlineable rate of our framework.

Key words: Mobile device, offline web browsing, intermittent network handling
Communicated by: D. Schwabe, G. Rossi, and J. Vanderdonkt

1 Introduction

Nowadays, the population of mobile device users is growing fast. With the advancements of mobile device, mobile platform, and wireless network technologies [1][2][3], browsing Web pages on mobile

devices has become more popular. More mobile versions of Web applications are developed for mobile devices and there are many researches [4][5][6] that propose solutions to assist mobile users in browsing Web pages more efficiently on mobile devices. Browsing Web pages has become a crucial functionality of mobile devices.

However, in the wireless environment, there are many issues that should be considered when developing a mobile Web application:

1. Capability of target mobile browser: Due to the difference between mobile and desktop browsers, mobile browsers are usually designed to support only a subset of JavaScript functions, CSS stylesheets, or HTML tags. The degree of support of JavaScript/CSS/HTML is different from other mobile browsers. Mobile Web application developers have to select a target mobile browser, and test their products in different execution environments.
2. Unstable Connectivity: The mobility feature enables mobile devices to be used without fix location. It is difficult to guarantee a stable network connection for mobile devices. There will be no network signal received in closed spaces, such as basements, tunnels, or elevators. Also, the connections may be interrupted by surrounding buildings, or high-speed transportation.
3. Cost of packet transmission: Usually, the connection is charged based on the number of transmitted packets. If there is a mobile Web application that requires plenty of packet transmissions, users may not be willing to use it again.

For those limitations listed above, many researchers [7][8][9] have already developed different solutions to solve these problems. In our system, we mainly focus on the second problem. Web browsing depends on internet connection; without internet connection, Web browsing no longer works. Such a problem can be solved by the offline Web page cache solution partially. Although most Web browsers are able to cache a part of Web pages which have been browsed, it cannot keep specific Web pages under users demands. Latest information will be cached and those older information which you need may be replaced. On the other hand, offline Web page saving is usually not convenient on mobile devices. Take Mobile IE for example; Mobile IE doesn't support "Save As" function for Web page backup. It is difficult for users to download and manage Web pages on mobile devices. Traditionally, if user wants to browse a Web page on his/her mobile device, he/she has to download the page to PC, transfer all the related documents and images to the device, and then try to find out where are the documents on the device when offline browsing. It is very inconvenient.

Google Gears [10] is one of solutions for offline Web page storage, which enables Web applications to be executed offline, no matter it is on PCs or mobile devices. Web pages can be pre-fetched and stored locally when the stable network connection exists. If there is no network connection, users can still access the Web pages via local storage. Therefore, the second problem can be solved by this mechanism. Moreover, there will be no packets transmitted for these local pages; as a result, the cost of the third problem listed above can be eliminated.

However, the prerequisites of using Google Gears are that mobile browsers should be installed with a client-side Google Gears plug-in for Web browser, and the visited Web sites should support server-side Google Gears library as well. Currently, the Web sites which support server-side Google Gears library are mainly Google Services, i.e. Gmail, Google Reader, etc. In general, most of Web

sites do not support this function. Until we can make sure all of the Web sites which users are interested in support Google Gears, this problem exists.

Gears-Monkey [11] is a solution for offline Web page storage based on Google Gears. It has an implementation on PCs. Gears Monkey is a kind of script for Greasemonkey [12], a Firefox plug-in, which can execute user defined JavaScript codes on Firefox. Users can develop their own Gears Monkey scripts, which enable Google Gears to work for particular Web sites, even though, those sites do not support server-side Google Gears library. However, there are some disadvantages of Gears Monkey:

1. Gears Monkey is not supported by any mobile Web browser.
2. A high level of scripting technique is required for users to achieve their goals.
3. There is no batch process supported; therefore, users must visit all Web sites which they are interested in to prepare for complete offline data.

Another solution for providing offline Web browsing is the HTML5 standard [13]. Web Browsers following the HTML5 standard should provide local space for offline storage and database. Even the Google Gears is not maintained by Google anymore because this offline Web browsing functionality can be replaced by HTML5. Since the offline capability will be provided as default function of Web browsers supports HTML5, no more plug-in is needed to be installed. However, similar to the server-side library problem of Google Gears, offline Web browsing cannot be provided without the support of server-side Web applications which utilizes the functionalities of HTML5.

Although it seems that HTML5 is a better solution than Google Gears to solve the offline browsing problem, HTML5 is still not supported by many Web browsers, especially by mobile Web browsers, until now. It is difficult to develop HTML5-based system for current needs. Therefore, we decide to choose Google Gears in our current implementation, and hope to replace it with HTML5 in the future if HTML5 is widely supported by mobile Web browsers.

In order to solve the above problems, we propose a Web content middleware with personalized interest lists to support offline Web browsing on mobile devices, with Google Gears as temporarily implementation, even though the selected Web servers do not support the server-side Google Gears mechanism. There are three objectives of the proposed system:

1. Provide mobile offline Web browsing capability, even for those Web sites which do not support server-side Google Gears library
2. The operations of defining interest lists and offline browsing should be convenient for users.
3. Batch processing for multiple specified Web pages

Although there are still some limitations of the proposed system, such as limited storage, limited real-time information, personalized Web pages, and limited protocol supported, these problems also exist in other offline Web content storage solutions. Some of them are physical constrains which are very difficult to be solved; even though, our system is still very useful to users.

The research contains six sections. In Section 2, we introduce some related works, backgrounds, and limitation analysis of our system. In Section 3, we propose the architecture of our system, and discuss the role and functions of each component. We describe a system implementation and detailed

system design in Section 4. Also, we demonstrate our system to present its results in Section 5. System evaluations will be presented in Section 6. Finally, we end up with a conclusion and discuss the future works about our system in Section 7.

2 Related Works, Background, and Limitation Analysis

2.1 Offline Desktop Application

J. Kistler et al. [14] have made a research in this topic. They proposed CODA, a distributed file system that allows transparent disconnected file operations, continued operation during partial network failures and a conflict resolution system via file merging. High performance is achieved via persistent caching in client side. Its merge system is similar to what an offline Web application will need, but it is built only for files and directories, with well known semantics. This wouldn't fit in a data intensive application, where several domain-specific conflicts must be handled.

Apollo is another solution for handling offline desktop application. Apollo can be used to transfer a whole, full Web application to the desktop, being able to operate without the need for a remote server. It also breaks the browsing experience, since the user gains a new application desktop.

As we know, mobile devices are not capable of storing a whole desktop application as Apollo does. The approach of CODA is rather preferable, since there is only the interface of CODA that will be chosen to transplant into the mobile devices. It sounds more possible and feasible.

Edgar Gonzales et al. [15] suggested a new methodology for the development of workflow-enabled Web applications containing certain functionalities that may be usable while offline. A Web application specification shall include a workflow description of all its functionalities; each task may have a personalized user interface, as well as an associated behavior. A development framework will automatically produce both offline and online programs for these tasks descriptions. The programmer will also be able to specify both task and domain model restrictions (e.g., forcing given information to be accessed in the server, only). Finally, the programmer may have to define strategies to allow a user to recover from a conflict in the synchronization stage. These strategies will be associated with the domain model contents and/or with specific workflow tasks. This step may not be fully automated by the framework, since each conflict type is generally application specific. The remaining steps of the development process using the proposed methodology shall not differ from the development of current Web applications. To successfully implement such methodology, three important research topics must be addressed:

A. Offline module identification

B. Offline application creation

C. Changes synchronization

However, there is too much work to bring up the solutions for the two big open issues. Those are "How can both online and offline versions of a functionality be generated from a single source?" and "How will the offline application save its alterations to be synchronized later?" Once these issues have not yet been solved, it is hard to realize any practical works by this new methodology. For current system architecture here, the first point of idea would be considered.

2.2 Offline Web Execution

Offline execution is not a new concept. More than one decade ago, there were multiple solutions [16][17] proposed to enable offline Web browsing. However, their approaches mainly focus on semantic information retrieval, in which the browser pre-fetch Web data upon the searching made by the user. The Web data, therefore, are available on local computer, although which is disconnected from network. J.Pitkow et al. [18] showed that the user experience will be enhanced when the Web data which user needs are pre-fetched automatically (i.e., without user action) based on automatically detected usage pattern of user. This fetching was carried out by a background agent, and helps user keep searching normally.

Ganesh Ananthanarayanan et al. [19] described a readily deployable system designed for Web browsing over slow intermittent networks called OWeB. It subscribes to RSS from Web server and pre-fetches all new content to partially update the old pages. Besides, the interrupted downloads are always resumed and not restarted. HTTP's byte-range request format is utilized for this purpose.

The above approaches provide helpful hints for the implementation of Offline Browsing System. In our system, offline Web execution is implemented; Web pages can be viewed even there is no network connection. In such system, documents which user needs should be pre-fetched before the loss of network connection. Similar to OWeB, a pre-fetch agent is also designed to download offline data in background. However, in order to support more generic Web applications, not only RSS information is considered, but also general raw Web contents (including pictures and JavaScript files) are considered to be browsed or executed offline in our system.

2.3 Intermittent Network Handling

Ganesh Ananthanarayanan et al. [19] considered that intermittent network handling is an important research for their offline Web browsing systems. Typically, problems would be overcome by using techniques like queuing and periodic re-trials into system. Some systems provide mechanism to recover and resume downloads in case of interrupted data transfers. Client-pull based techniques [20] try to intelligently predict the time of change of data at the server and pull the data. Also, intelligence is needed to pull only the "relevant and useful" data.

Notification systems were part of solution but push-based techniques faced an important issue of not being scalable. Both push and pull based techniques faced the important problem determining the importance of the content.

Intermittent network can also be solved by distribution of data from the content server to geographically distributed caches. These caches reduced the access time from the server to the clients resulting in lower latency and better experiences. Caches needed to maintain coherency and also and relevant data. Replication in Web content also needed to conserve bandwidth by "intelligently" pulling only the "useful has problems and issues associated with it [21].

2.4 Web Crawler

Web crawler [22] is a kind of program which browses the World Wide Web in an automated manner. Other terms for Web crawlers are ants, automatic indexers, bots, and worms or Web spider, Web robot. The process on how Web crawler works is called Web crawling or spidering. Many server sites, in

particular search engines, use spidering as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later offline processing. Crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML code. Also, crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses for spam. In general, it starts with a list of URLs to be visited, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to be visited.

Teleport Pro [23] is a Web crawler for getting data from the Internet. Launch up to ten simultaneous retrieval threads, access password-protected sites, filter files by size and type, search for keywords, and much more. Teleport Pro is capable of reading HTML 4.0, CSS 2.0, and DHTML, it searches all of the files on server sites, such as sites with Java Applet. Teleport Pro can download all or part of a Web site to your computer, search a Web site for files of a certain type and size, and download a list of files at specified URLs.

2.5 HTML5

HTML5 [13] is a new Web standard, designed to take the place of existing HTML 4.01, XHTML 1.0 and DOM Level 2 HTML standards. It aims at reducing the need for browser plug-in-based Rich Internet Applications, such as applications based on Adobe Flash [24], Microsoft Silverlight [25], and Sun JavaFX [26]. Web Storage and Web SQL Database are two new APIs provided in HTML5. They are described below:

Web SQL Database: The browsers which support HTML 5 have a local database supported, which is a local SQLite database. With this database, client side applications can store the information what they need via standard SQL communications.

Web Storage: A Name-Value-based data storage is supported for storing HTML or JavaScript data. The stored information is non-volatile; even though the browser or the phone is turned off, the data will still remain in the storage.

2.6 Google Gears

Google Gears is an open source browser extension which supports Web applications to be executed offline. It is applicable on personal desktops, laptops and handheld devices. It can store and serve application resources locally, and run asynchronous JavaScript to improve application responsiveness. Google Gears consists of many components which enable more powerful Web applications on browsers. Three major components are listed as followings:

A. LocalServer caches and serves application resources (HTML, JavaScript, images, etc.) locally

B. Database stores data locally in a fully-searchable relational database

C. WorkerPool makes your Web applications more responsive by performing resource-intensive operations asynchronously

In fact, Google Gears is no longer developed by Google, since its functions can be covered by HTML5, which provides similar functionalities of offline data storage. However, HTML5 is not widely supported yet, especially by the browsers of mobile devices nowadays. Therefore, we decide to

adopt Google Gears as our local storage implementation temporarily, and wish to replace it with the HTML5 standard when most of browsers of mobile device support HTML5 in the future. At that time, the problem of local plug-in installation prerequisite will be solved.

2.7 Gears Monkey

Gears Monkey is a technique for browsing offline, it works based on Google Gears and Greasemonkey. Greasemonkey is a user script manager. It is an extension for the Mozilla Firefox Web browser, which allows users to customize the way a Web page displays using JavaScript. By using Google Gears with Grease Monkey, Google Gears related code can be inserted into any downloaded Web page. Therefore, those users' favorite Web sites can be browsed offline.

2.8 Rich Internet Applications

Rich Internet Applications (RIAs) [27][28] are Web applications that have many of the characteristics of desktop applications, typically delivered either by way of a site-specific browser, via a browser plug-in, or independently via sandboxes or virtual machines. Adobe Flash [24], Sun JavaFX [26], and Microsoft Silverlight [25] are currently the three major platforms of RIA. Other popular RIA techniques also include Adobe AIR [29] and AJAX [30]. RIAs are usually designed to improve the richness of data, business logic, communication, and presentation of Web pages.

Although RIAs can also be designed for offline Web data management, it is not the main purpose of RIA platforms. On the other hand, in the perspective of offline data management, Google Gears focuses on it, and provides a better solution for handling offline issues (e.g. providing well-established offline data pre-fetch mechanism). Therefore, we choose Google Gears to be our current implementation of client-side offline data management platform.

2.9 Mobile Content Adaptation and Personalization

Mobile content adaptation [7][31][32] is an important issue when browsing Web sites. In general, Web pages are designed for PC users with large screens. In this way, as these Web pages are browsed on mobile devices, which usually provide only small screens, the display results are usually very not user friendly. Therefore, mobile content adaptation technologies are proposed to solve this problem. The challenges of mobile content adaptation include different device profiles and different user preferences. Especially for multimedia content, mobile devices usually are not as powerful as PCs, thus only support a part of multimedia formats. In general, there are three categories of mobile content adaptation [33]: client-based application adaption, client-server application adaption, and proxy-based application adaption. In client-based adaptation, client-side application performs content transcoding according to the profile of device which it is deployed on; in server-side adaptation, the server decides which kind of content should be delivered to clients according to their profiles; in proxy-based adaptation, contents are transcoded on-the-fly during the delivery from server to client.

Although mobile content adaptation and mobile offline browsing both deal with contents between Web servers and mobile clients, they can belong to two different conceptual layers; the mobile content adaptation mainly focus on data representation, while offline Web browsing focus on data management. In our previous work [5], we designed a proxy-based mobile adaptation system together

with an authoring tool. Users can specify which part of Web page they are interested in by our authoring tool, and the other parts of the Web page will be filtered out automatically by our content adaptation proxy. In order to make our offline browsing system compatible with this previous work, we decide not to include the content adaptation issues into the offline browsing system. Since our previous work is a proxy-based adaptation, it is very easy to use these two systems at the same time for both content adaptation and offline browsing. During the online stage, the mobile device retrieves the adapted contents via the proxy, and stores them locally. In this way, users can directly access adapted Web contents offline.

2.10 Limitation Analysis

After the review of related works, several limitations of mobile devices could be analyzed, including:

A. **Limitation of storage on mobile device:** there are too many Web contents all over the world; it is impossible to store all of them into a mobile device with limited storage provided nowadays. Storage designed for mobile device is usually no more than 50GB until now, but there are at least billions of pictures on the Internet. Even for a single Web site, the amount of its Web contents could be too huge to be stored locally. Even if there is enough storage for those contents, it also requires a lot of time for transferring data. Therefore, the most efficient way to manage limited storage is to make users able to define their own interest lists. Also, the depth of specified address should be limited.

B. **Limitation of real-time information:** the term "offline" in this paper implies not only telecom or wireless LAN connection, but also any form of connection between mobile device and the Internet doesn't exist. Thus, if any real-time information presents on the Internet, an offline mobile device is impossible to be aware of this change. As a result, many real-time applications such as real-time News, streaming multimedia content, or online chat room are difficult to be implemented under offline scenario.

C. **Personalized Web pages:** Nowadays, many Web pages are provided in the form of personalized pages; different user will have different Web content even though the URL of page is the same. This service usually requires users to login first. However, different authentication mechanisms are provided by different Web sites; it is difficult to automatically login for users during data pre-fetch.

D. **User participation:** In the Web 2.0 era, user participation of Web content is emphasized. However, user participation is usually provided when user is online with his/her account login; therefore, it has the same limitation as described in C.

E. **Limited protocol:** In general, Web pages are not only provided though HTTP, but also HTTPS. Usually, HTTPS is utilized after user login; therefore, it also has the same limitation as described in C. Moreover, even if the user login could be executed during data pre-fetch, a Web application should be implemented in the local server for performing offline encryption/decryption processes with private key, which is usually regarded as a secret for user authentication of Web sites.

Although offline Web browsing has so many limitations, it can also be very attractive to users. To those public information or articles, they are suitable to be browsed offline on mobile devices. For

example, applications such as online novels, online News (maybe one or two days delay is acceptable), weather information, time of movie, address of restaurant, are all very useful to mobile users in the offline environment.

3 System Architecture

3.1 Overview

Traditionally, Web pages will not be available without network connection; it is very inconvenient for users. In this research, we propose a framework, which makes Web browsing possible on mobile devices without any available network to the Internet. The overview of our framework is shown in Figure 3-1. There are three main components in our system: Original Web Sites (OWS), Offline Data Pre-Fetcher, and Mobile Device with Local Server. Web pages can be pre-fetched by Offline Data Pre-Fetcher, downloaded to mobile device when network connection still exists, and browsed by user even if there is no network connection anymore.

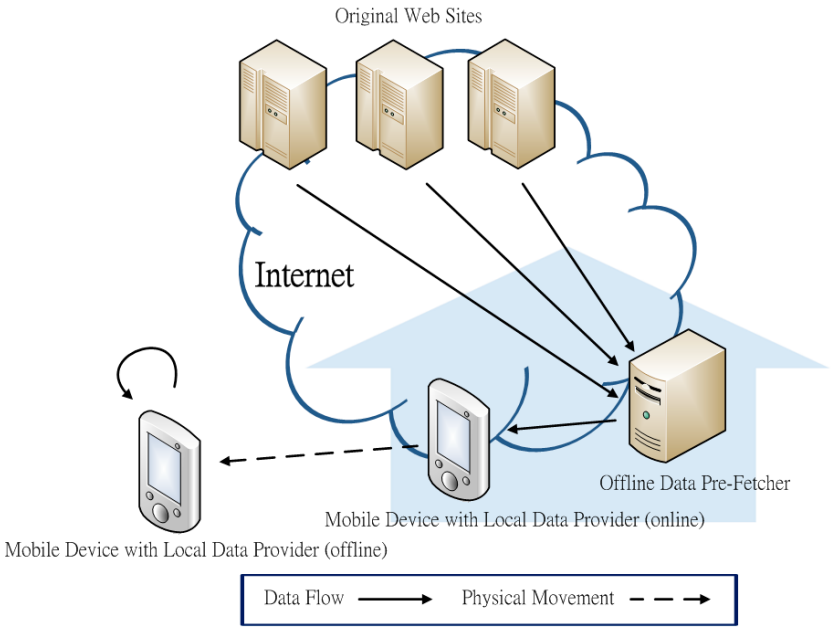

Figure 3-1 Overview of System Framework

Original Web Sites

Original Web Sites are content providers providing Web page services on the Internet. Web documents provided by Original Web Sites can be either browsed from mobile device directly or pre-fetched by Offline Data Pre-Fetcher.

Offline Data Pre-Fetcher

The Offline Data Pre-Fetcher is responsible for offline data organization. It provides a management interface for users to specify which Web pages they are interested in. Based on the users'

interests, the Offline Data Pre-Fetcher pre-fetches Web documents and related data and stores them for cache purpose. It also allows users to define how frequent these Web pages should be updated. After all, those organized Web pages are re-published for mobile device to access.

Mobile Device with Local Data Provider

When there is a network connection available, for example, if a user is going to leave his home, he can download the organized Web pages through Offline Data Pre-Fetcher. After that, even if there is no network connection on his way to office, Web pages are still accessible which are provided by the Local Data Provider on his mobile device.

Instead of preparing offline data directly by mobile devices, there is an Offline Data Pre-Fetcher playing the rule of Web Proxy. There are several advantages of this design; first, users do not have to visit each original Web site respectively each time; it can be performed in a batch way; second, pre-fetched data can be updated more frequently even if there is no connection of mobile device; finally, the Offline Data Pre-fetcher can be accessed and managed not only through mobile device, but also via desktop PC.

3.2 System Architecture

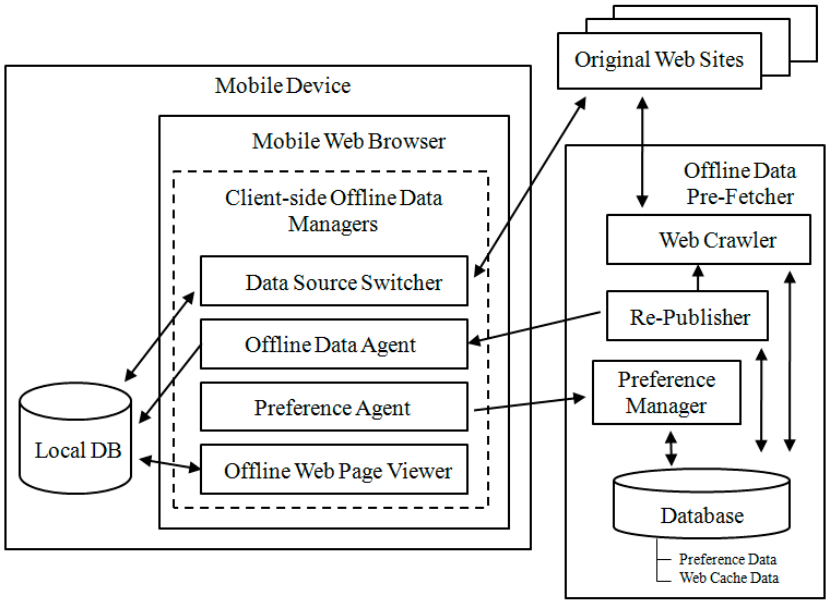

Figure 3-2 System Architecture

The system architecture is shown in Figure 3-2. Based on the user preferences stored in Database, Offline Data Pre-Fetcher downloads the specified Web pages from Original Web Sites by Web Crawler, and stores the cache data into Database. On the other hand, the cache data is be re-published by Re-Publisher for mobile device to access. The most convenient way to browse Web pages on mobile device is via mobile Web browser, which employs client-side offline data managers for offline data management.

The Data Source Switcher is responsible for choosing the data source based on current network status. When a mobile device is connected to the Internet, the data source will be the Internet, which is the same as traditional Web browsing; otherwise, if there is no connection, the local data will be provided by Local DB with the same URL of Web page.

The most important factor of offline Web browsing is offline data management, including specifying data source, pre-fetching offline data, offline storage management, and offline data presentation. Since the size of local storage on mobile device is usually small, it is important to decide which content should be kept. Under this limitation, we should maintain an Interest List for user to maximize the storage utilization. The Interest List information is stored in the Database of Offline Data Pre-Fetcher and managed by the Preference Manager which processes personalized information. Users can manage their Interest Lists through the Preference Agent on their Mobile devices or the Preference Manager directly by the Web browsers of their PCs. After the Offline Data Pre-Fetcher finishes preparing Web cache, Re-Publisher will re-publish this data by maintain a Web server for Offline Data Agent to access, and then store it to the Local DB on mobile device. Finally, offline Web pages can be presented by the Offline Web Page Viewer.

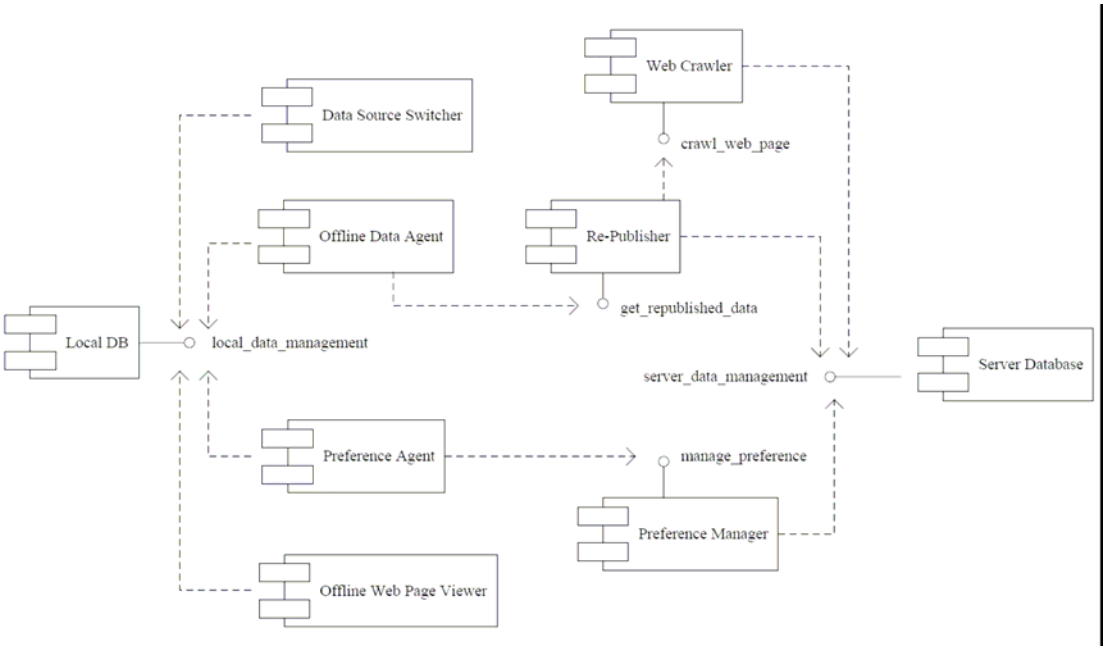

Figure 3-3 Component Diagram of System Architecture

The detailed design of the system architecture is represented as a component diagram shown in Figure 3-3. Local DB and Server Database are managed through the local_data_managent and server_data_management interfaces; The Re-Publisher invokes the Web Crawler to crawl Web pages through the crawl_web_page intergace; The offline data can be acquired through the get_republished _data interface; Also, preference data can be managed thought the manage_preference interface.

In the proposed Offline Browsing framework, there are four states defined: Online State, Subscription State, Pre-Fetch & Re-Publish State, and Offline State. They will be explained respectively as follows.

Online State

Under the online operation state, users have normal network connection, so the way of Web browsing in the system will not be different with the one in general. Due to the mobility, it is difficult to ensure that a normal connection is kept stable. For instance, going down to the basement, walking through the tunnel, or staying inside an elevator, etc. These circumstances lead intermittent connection unstable to obtain network resources. In the majority of Web browsers, when there isn't any network connection available, Web browsers mostly would not be able to go on browsing Web pages. Therefore, a stable online state is not guaranteed.

Subscription State

During the subscription state, users can specify which Web pages they are interested in. After that, Offline Data Pre-Fetcher will access these Web pages, download the webpage data, and store the data based on the specified Interest Lists.

Pre-Fetch & Re-Publish State

On the other hand, Offline Data Pre-Fetcher performs a service as a Web server for Web browsers to retrieve information. Once all the data needed from OWSs has already been downloaded and well prepared, mobile devices can obtain these Web pages by Offline Data Agent.

Offline State

Finally, when mobile devices lose network connections, Web browsers can still browse these offline Web pages through the Offline Web Page Viewer.

4 System Implementation and Detailed System Design

4.1 System Implementation

The system implementation is shown in Figure 4-1, the Web Crawler, Re-Publisher, and Preference Manager of Offline Data Pre-Fetcher, are all implemented as PHP programs executed on Apache HTTP Server. At the client side, Google Gears is used to play the role of Data Source Switcher and Offline Data Agent. Moreover, Google Gears, Preference Agent, and Offline Web Page Viewer are all plug-ins of Mobile IE, which is deployed on the Windows Mobile 5 Operating System.

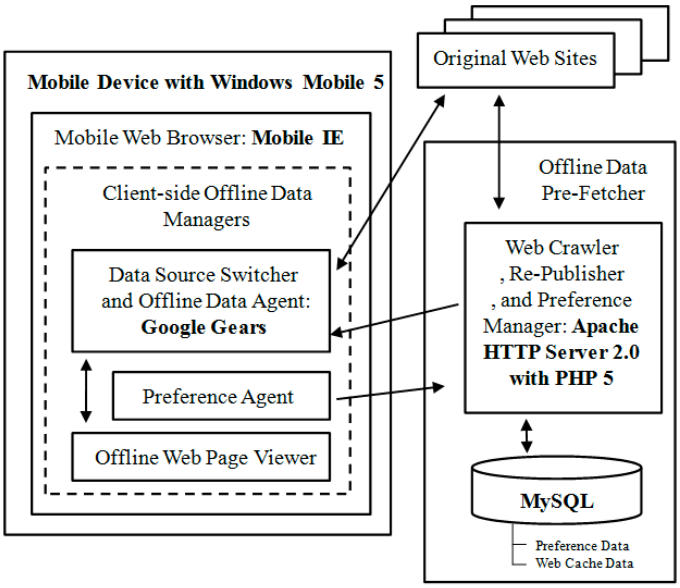

Figure 4-1 System implementation

Google Gears

Currently, the Web browsers and operating systems which support client-side Google Gears plug-in are listed as follows:

- Firefox 1.5+ and Internet Explorer 6.0+ (on Windows XP/Vista),
- Internet Explorer Mobile 4.01+ (on Windows Mobile 5+),
- Firefox, Safari (on Macintosh),
- Firefox (on Linux),
- Chrome Lite (on Android).

Web Crawler

The Web Crawler grabs HTML documents, images, and other files recursively from OWSs based on each URL in the Interest lists specified by users. Moreover, the Web Crawler transforms dynamic pages, such as PHP or ASP pages, into static pages, i.e. HTML pages, for storage. Also, external or static links should be transformed into relative links.

Re-Publisher

There are several problems that exist when caused by using Google Gears if there is no single Re-publisher in the middle of mobile devices and OWSs. First, the prerequisites of using Google Gears include installing a client-side browser plug-in and providing server-side Google Gears mechanism. Until now, most of OWSs do not support this function. One of our challenges is to store the contents of OWSs using Google Gears, no matter OWSs support server-side Google Gears library or not. The Re-

Publisher is designed to support the server-side functionalities of Google Gears; therefore, the offline service can be provided, even the OWSs does not support it. The Re-Publisher creates a JSON file containing offline URL list, which will be received by the browsers with Google Gears plug-in. By following the Google gears mechanism, the browsers can retrieve offline Web information, and store it in the database of Google Gears locally.

Another issue is that, the URLs of the Interest List may be distributed on different domains. For each URL domain, Google Gears will ask for a confirmation before access. If there are lots of Web pages needed to be accessed, and all the URL domains of Web pages are different, multiple pop-up windows will be created by Google Gears to be confirmed. These confirmations will make some JavaScript codes be embedded for initialization. It is considerable inconvenient to users. In order to solve this problem, all the Web cache data is re-published within a single domain, i.e., the Re-Publisher's domain.

Preference Manager

In our framework, each user can define his own "Interest List" (or called "Favorites"), which contains the URLs which he or she is interested in. The Interest List will be maintained mainly in the Offline Data Pre-Fetcher with a copy in the Local DB; therefore, users can edit their Interest Lists anytime, anywhere, and on different platforms, such as PCs at home or office, and mobile devices outdoors.

Preference Agent and Offline Web Page Viewer

When users enjoy their online Web browsing activities, they may want to add the URL of the current Web page into their Interest Lists for offline Web browsing. If they have to go to another Web page to do this edition, it will be very inconvenient. Another problem is that, since the pre-fetched data is republished by Offline Data Re-Publisher, the URLs of offline Web pages will not be the same as the ones of those pages provided by the OWSs; therefore, it is impossible for users to access the Web pages through their original URLs or remember the new URLs. In order to solve these problems, the Preference Agent and Offline Web Page Viewer are required.

The Preference Agent can be used when users want to add the current webpage URL to the Interest List immediately. In our implementation, user can execute it through the "Add page to off-line list" option in "Menu" function of the Mobile IE browser. During the process of Preference Agent, users have to input their accounts and passwords, so that the Preference Agent can append the current URL to the users' Interest Lists. Finally, users will be redirected to the Web pages where they were before executing it. In this way, users can continue to enjoy their original Web browsing activities.

The Offline Web Page Viewer is responsible for providing offline copy of Interest List for users. In our implementation, user can executes it through the "View off-line pages" option in "Menu" function of the Mobile IE browser. During the process of Offline Web Page Viewer, users only have to input their account information, so that the Offline Web Page Viewer can generate the offline URL based on their accounts. Finally, the offline Interest List can be displayed for users, which is similar to the "Favorites" function in the Mobile IE browser; therefore, users can access the offline Web pages through this list.

4.2. System Behaviour Design

We have defined four states in our framework: Online State, Subscription State, Pre-Fetch & Re-Publish State and Offline State. We will describe them respectively with sequence diagrams as follows.

Online State

Normally, the system is under the Online State, information is only exchanged between users and Web sites. By using mobile devices, users browse Web pages directly through browsers. Once the user input the URL of OWS in the address bar of browser, the browser will send request HTTP message for acquiring the Web page. After the OWS received the message, it returns the Web page of specific URL to the browser for the user. Hence, the content of that specific URL will be displayed on the browser. The sequence diagram of Online State is shown below, with information exchanged between users and Web sites.

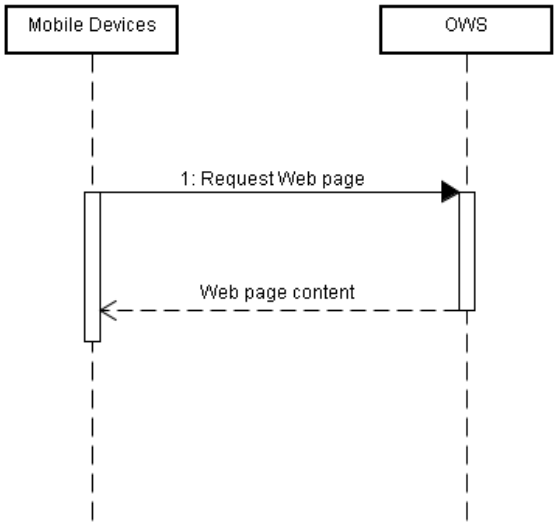

Figure 4-2 Sequence Diagram of Online State

Subscription State

When the user proceeds to subscribe his offline Web page URLs, the Subscription State begins. The information will be exchanged between the browser and Offline Data Pre-Fetcher. The user sends his ID and password to the Offline Data Pre-Fetcher through the Web browser, and then, sends the subscription message with the URL which the user wants to browse it offline. Anytime the user visits the Offline Data Pre-Fetcher, he can review which Web pages have already been subscribed in his account before. He can also append more records of URLs whenever he wants.

Pre-Fetch & Re-Publish State

When the user begins to acquire the offline Web page contents, the system state will change to the Pre-Fetch & Re-Publish one. The information will be exchanged among the browser, Google Gears on mobile device, Offline Data Pre-Fetcher, and OWSs.

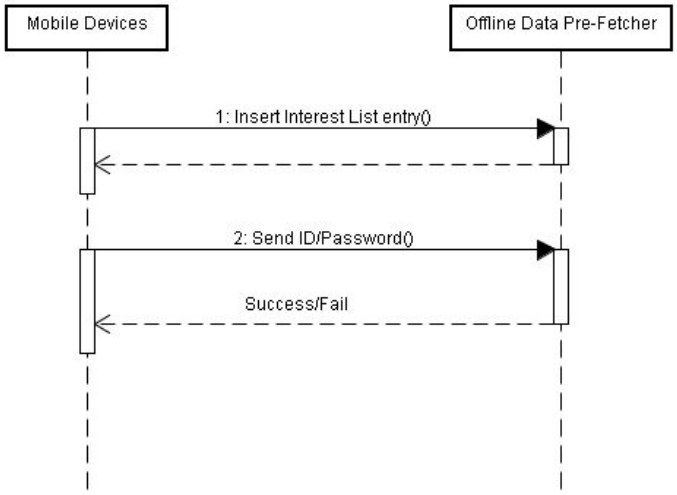

Figure 4-3 Sequence Diagram of Subscription State

First of all, the user sends his id and password to the Offline Data Pre-Fetcher through the Web browser. After he executes the "Update" option, the Offline Data Pre-Fetcher will send Web page requests to the OWS which maintains the Web pages specified in the user's Interest List. After that, the OWS will return the Web page content to Offline Data Pre-Fetcher. Offline Data Pre-Fetcher will collect all the offline Web pages based on the subscribed URLs by Web Crawler. Then, the Web Crawler arranges all the URLs of these Web pages into a file in JSON format. This file will be sent to the browser with Google Gears installed during the execution after the "Capture" option is selected. When Google Gears receives the JSON file, it receives all the URLs of offline Web pages. These URLs will be stored in one of its main component, called "LocalServer". According to these URLs, Google Gears requests the Re-Publisher of Offline Data Pre-Fetcher for offline Web page contents. Finally, LocalServer receives the Web content from the Re-Publisher S. The Database of Google Gears will keep all the offline Web page contents. The Four-way information exchange is shown in Figure 4-4.

Offline State

During the Offline State of the system, there is no external connection for mobile devices. In this state, all the information is exchanged only between the Web browser and Google Gears which stores the offline Web page contents.

In the circumstance of unavailable network connection, the Data Source Switcher, which is implemented as the LocalServer of Google Gears, still can process the URLs of offline Web pages. The LocalServer in Google Gears will parse the inputted URLs and decide whether the URLs have already been recorded in the LocalServer. Once there is no network connection, and the incoming URLs have been recorded in the LocalServer, the LocalServer will send request messages to its Database for receiving offline Web page contents. Similar to Online state, the LocalServer will respond the offline Web page to the browser. After that, the browser will display the offline Web page

contents of the offline URL. The information flow between the browser and Google Gears is shown in Figure 4-5.

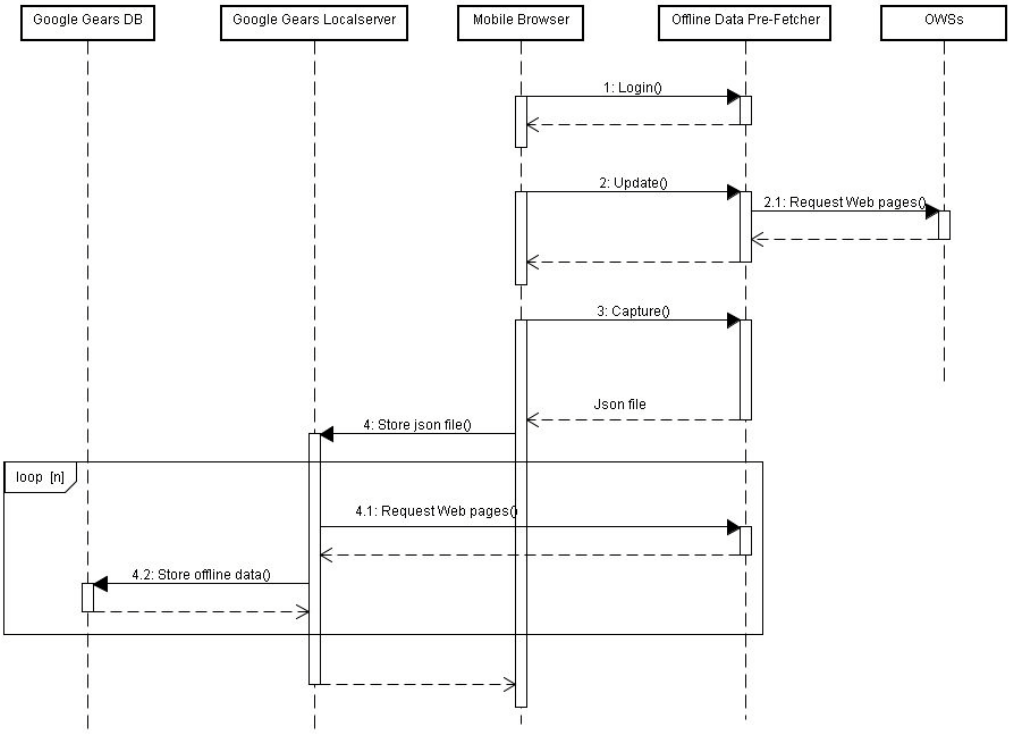

Figure 4-4 Sequence Diagram of Pre-Fetch & Re-Publish State

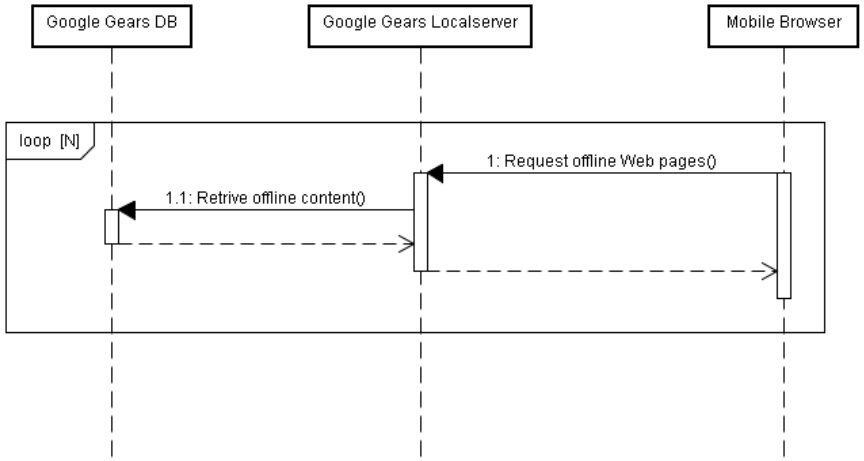

Figure 4-5 Sequence Diagram of Offline State

5 System Demonstration

In this section, we will present a demonstration for our system. In this demonstration, we will describe a general scenario with a handheld device of ASUS P535, which "Windows Mobile 5.0" is installed in it. Moreover, there will be an evaluation of the comparison between our system and related works at the end of this section.

Assume that the browser for the user in this scenario has already install Google Gears, Preference Agent, and Offline Web Page Viewer. Also, the user has already subscribed for two URLs of Web pages A and B into his Interest List. Now, the user wants to append the third URL of Web page C to Interest List, and retrieves all data of these three Web pages into his mobile device for offline browsing. All of these links are stored in the Mobile IE Favorite. The URLs A, B, C are listed in table 5-1.

Table 5-1 URLs in Mobile IE Favorite for demonstration

	Names	URLs of offline Web pages
A:	Taipei Weather	http://www.cwb.gov.tw/pda/observe/Taipei.htm
B:	National Palace Museum	http://www.npm.gov.tw/pda/new_01.htm
C:	Fox News	http://foxnews.proteus.com/content.html?contentId=25705

The operation sequence of the scenario is listed below:

1. The user logins to the webpage of Offline Data Pre-Fetcher by his account and password. He can find that there will be two URL records of A and B in his Interest List.
2. The user begins to browse the webpage C. For instance, he can select the options follow the sequence : [Start → IE → Menu → Favorite → "Fox news"].
3. The user inserts the URL of webpage C into the Interest List by using the Preference Agent. For instance, he can select the options follow the sequence: [Menu → Tool → "Add page to off-line list" → enter account/password], as shown in Figure 5-1 and 5-2.
4. The user logins to Offline Data Pre-Fetcher again to check whether the webpage C has already been inserted into the List. For instance, he can select the options follow the sequence: [Go to our Offline Data Pre-Fetcher login Web page → enter account/password].
5. The user downloads the offline data from Offline Data Pre-Fetcher to Local DB in the mobile device. For instance, he can select the options follow the sequence: [Erase → Update → Capture] , as shown in Figure 5-3, 5-4 and 5-5.
6. The user sets the mobile device into Offline status. For instance, he can select the options follow the sequence: [File → Settings → Network → confirm network shutdown], as shown in Figure 5-6.

By using the Offline Web Page Viewer, the user enters an offline Web page list, and browses the Web pages A, B and C offline. For instance, he can select the options follow the sequence: [Menu → "View off-line pages" → enter his account → "offline pages" → choose any URL link → Browse offline Web pages], as shown in Figure 5-7, 5-8, 5-9, and 5-10.

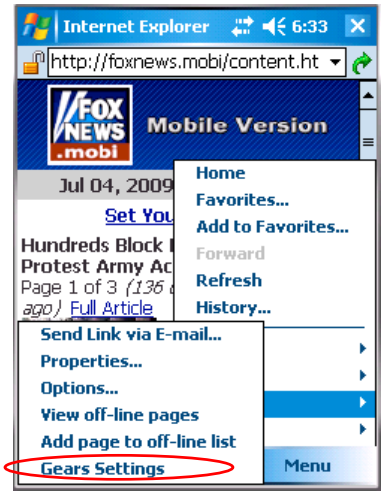

Figure 5-1 Choosing the Preference Agent

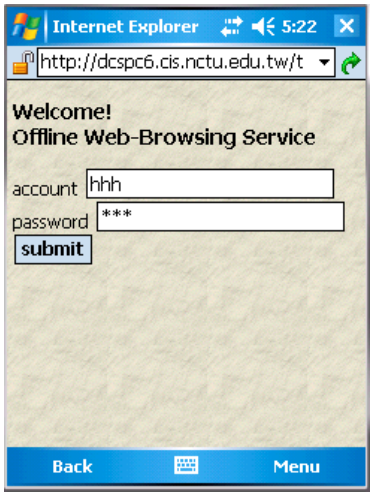

Figure 5-2 Login with ID and passwor

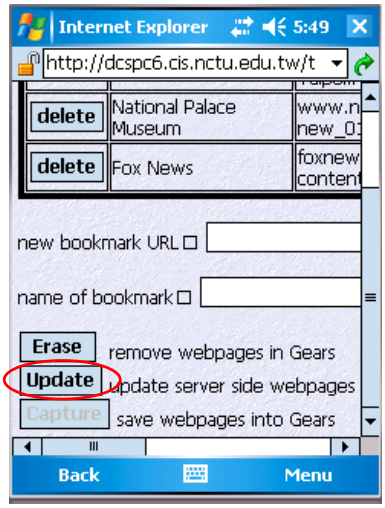

Figure 5-3 Offline Web pages update

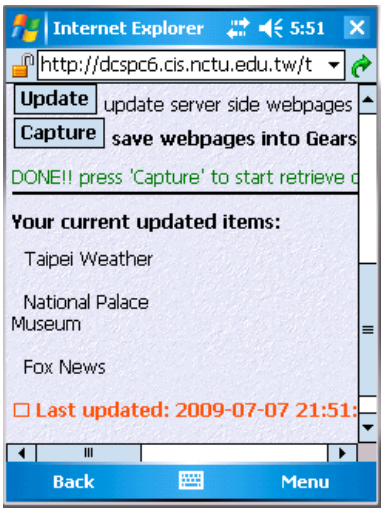

Figure 5-4 Update completion alert

Figure 5-5 Capture Web pages

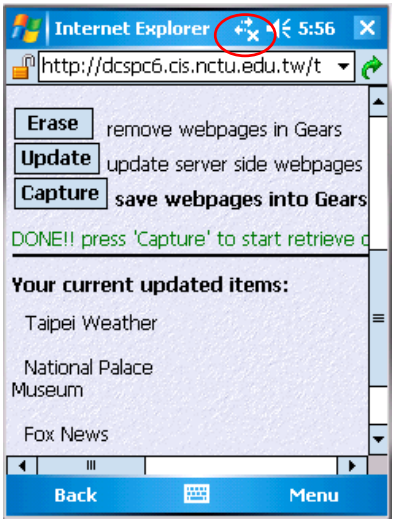

Figure 5-6 Network connection terminated

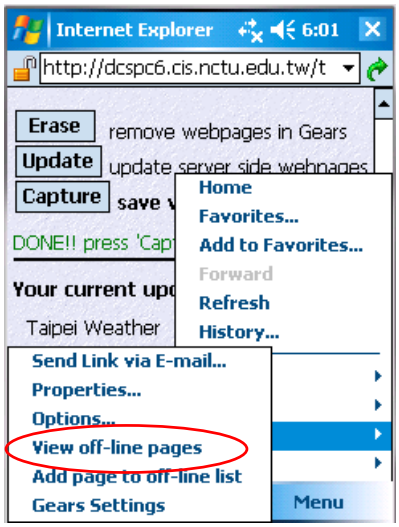

Figure 5-7 Choosing the Offline Web Page Viewer

Figure 5-8 Enter user's ID

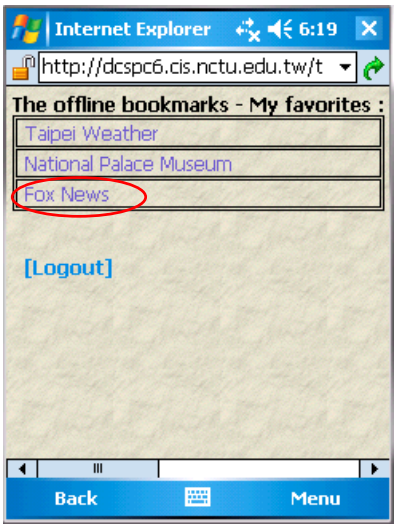

Figure 5-9 Choosing links for offline browsing

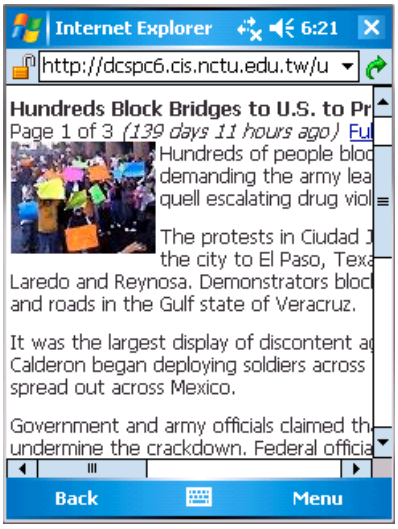

Figure 5-10 Browsing offline Web pages

6 System Evaluation

6.1. System Comparison

The comparison among our Offline Browsing System and related works is shown in Table 6-1. In order to provide Offline Browsing Service, it is important to have a friendly way for users to browse these pages. Due to the lack of OWSs' support, Google Gears and HTML5 cannot provide Offline Browsing Service for most of OWSs until now. Gears Monkey does not need any support from OWSs; however, it requires users to have specific script file prepared for each particular OWS. Teleport Pro can overcome these two problems above. It can also execute batch process for multiple OWSs, but it cannot work on mobile platform.

Table 6-1 Comparison among offline browsing technologies

	OWSs Should Support	Script File Required	Applicable on Mobile Devices	Batch Processing for Multiple Web Sites	URLs of Original Web Pages
Google Gears	Yes	No	Yes	No	Remain the same
HTML5	Yes	No	Not Yet	No	Changed
Gears monkey	No	Yes	No	No	Remain the same
Teleport Pro	No	No	No	Yes	Changed
Our System	No	No	Yes	Yes	Changed (but a plug-in has been provided to solve this problem)

The proposed Offline Browsing System has most of the advantages which are not supported by other related works. The only problem is that the URL of original Web page will be modified. It is not

user friendly, since the users have to remember what the URLs of the offline Web pages are. However, the Mobile IE plug-in, Offline Web Page Viewer, is designed to overcome this problem; therefore, users can browse offline Web pages more easily.

6.2 Efficiency Analysis

In order to evaluate the system efficiency, we focus on ten web sites, and measure the update time and the capture time, which is the time users take after they click on the "Update" and "Capture" button in the webpage provided by Offline Data Pre-Fetcher. The ten websites are listed in Table 6-2.

Table 6-2 Evaluated Web site list

Site ID	Names	URLs of Web pages
1	National Chiao Tung University	http://www.nctu.edu.tw
2	Google	http://www.google.com
3	PCWorld	http://www.pcworld.com
4	Computerworld	http://www.computerworld.com
5	Ask.com Search Engine	http://www.ask.com
6	Wikipedia	http://www.wikipedia.com
7	CNN.com International	http://www.cnn.com
8	MSN.com	http://www.msn.com
9	Baidu	http://www.baidu.com
10	ESPN	http://www.espn.com

The time consumption evaluation for the update and capture operations for each site is listed in Table 6-3. In general, the update operation is fast enough, which takes 1.27 seconds in average. As to the capture operation, the time consumption ranges from 9 to 13 seconds. Although it takes more time on mobile device, the degree of increment is not much.

Table 6-3 Time consumption evaluation for update and capture operations for each Web site

Site ID	Time consumed on PC (Sec.)		Time consumed on Mobile Device (Sec.)	
	Update	Capture	Update	Capture
1	1.2734778	9.68	1.2734779	9.71
2	1.273479	9.76	1.273479	9.80
3	1.2734781	9.89	1.2734781	9.91
4	1.2734783	10.20	1.2734783	10.23
5	1.2734784	10.26	1.2734785	10.29
6	1.2734784	10.28	1.2734787	10.33
7	1.2734786	11.34	1.273479	11.39
8	1.2734788	12.01	1.2734794	12.10
9	1.2734789	12.04	1.2734798	12.19
10	1.2734794	12.43	1.2734806	12.67
Total	120.6247857		121.3547893	

The incremental time consumption analysis for update and capture operations on PC and mobile device are shown in Figure 6-1 and 6-2. The total time consumption ranges from 10 to 15. Although it takes more time if more Web sites are selected, the update time is almost remain the same, and the capture time increases slowly. After the batch process of our system, the total time is decreased from about 2 minutes to about 14 seconds; the average decrease rate is 87.91%.

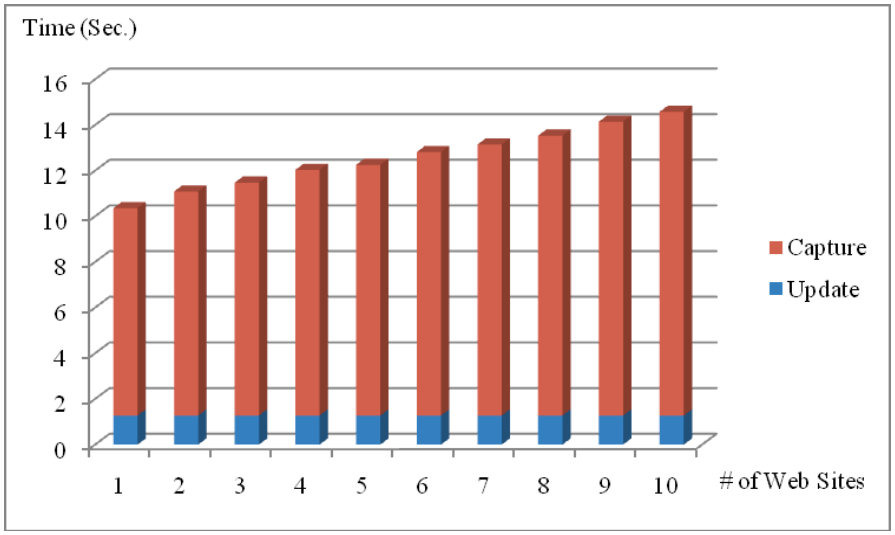

Figure 6-1 Incremental time consumption for update and capture operations on PC

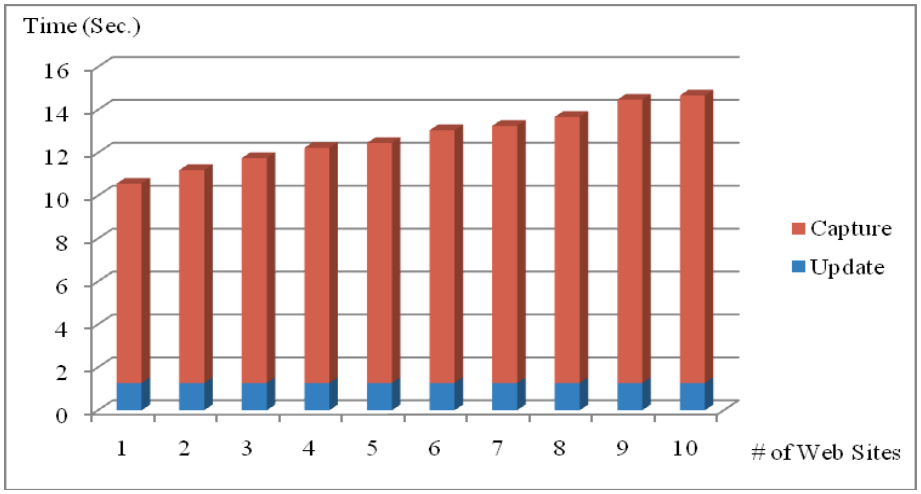

Figure 6-2 Incremental time consumption for update and capture operations on mobile device

6.3 Coverage Analysis

In order to evaluate the offlineable rate, we conduct a preliminary survey to analyze the percentage of offlinable Web pages which may be browsed by mobile users. It is not easy to identify which web

pages are possible to be browsed offline by mobile users. Browser bookmarks on mobile devices may reveal the behaviour of their owners, but few people bookmark links without offlineable service provided. In this research, we choose to analyze Web links provided by the mobile version of del.icio.us Web site [34]. The mobile version of delicious Web site is designed especially for mobile phone users; similar to del.icio.us [35], it provides suggested bookmarks with various tags. Since del.icio.us has plenty of users all over the world, we assume that Web links provided on the mobile version of it will be accessed with a high possibility by a large amount of mobile device users.

The analysis result is shown in Table 6-4. In this table, we define three categories, "R", "E", and "S", for the reasons of non-offlineable Web pages. Category "R" represents Rich Internet Applications, such as Flash video clips on YouTube or Silverlight application. "E" means search engine page, including Web portals and many home pages of Web sites. Finally, "S" includes Web pages which provide server-side services, such as Google Translate or online shopping cart. There are 100 randomly-choose Web links analyzed within more than 10 tags (including education, inspiration, money, tips, health, and so on); 82% of them are offlineable, and 18% of them are not. Among those non-offlineable pages, the most frequently happened reason is Rich Internet Applications, which reveals that Flash video clips provided by YouTube has been easily and widely embedded into many Web pages nowadays. Based on the analysis result, we can conclude that solving the problem of offline RIA browsing, especially offline YouTube video browsing, is the most important issue in our future work.

Table 6-4 Offlineable rate of Web links provided by the mobile del.icio.us Web site

# of Web pages	100
# of offlinable pages	82
Percentage of offlinable pages	82%
# of R pages	12
# of E pages	2
# of S pages	4

R: Rich Internet Applications; E: Search engine; S: Online Service

7 Conclusion and Future Work

7.1. Conclusion

Due to the limited resources supported on mobile devices, more and more applications tend to provide information to mobile users through Internet. However, the network connection is not stable and is easily interrupted by surrounding environment. Google Gears provides an Offline Webpage Storage mechanism for Web applications to run offline. However, the server-side Google Gears library should be supported on the Web servers, which is not supported on most of the Web servers nowadays. By using Gears Monkey, users can browse the offline Web pages even if the original Web servers do not support server-side Google Gears library. However, users have to write a script for each site by themselves.

In this research, we propose an offline mobile browsing framework, which enables mobile users to define their personalized Interest List and to browse Web pages offline. Especially, our framework is designed for those OWSs which do not support server-side Google Gears library; therefore, the offline Web pages can be provided without the modification of Web servers.

In the following discussion, we analyzed our system based on the objectives we mentioned in section 1. In our Offline Browsing System, we proposed a framework by client-server architecture for users to offline Web browsing, which:

Provides the offline Web browsing capability, even for those Web sites which do not support server-side Google Gears mechanism.

It is difficult for every Web site to support server-side Google Gears library. Instead, we supported the server-side Google Gears mechanism on the Offline Data Pre-Fetcher. By pre-fetching the Web pages from OWSs and republishing them, the Offline Data Pre-Fetcher provides a single entry for users to prepare offline data.

The operations of defining Interest Lists and offline browsing should be convenient for users

We provided two Mobile IE plug-ins: Preference Agent and Offline Web Page Viewer. The Preference Agent assists users in adding current Web page into their Interest Lists while the Offline Web Page Viewer assists users in browsing their own offline Web pages without remembering any new URLs or their favorite offline Web pages.

Supports batch processing for multiple specified Web pages

Users can define their own Interest Lists, which will be processed in a batch by Web Crawler. It is no longer for users to deal with repeating tasks, when they want to update their offline Web pages. This is more convenient and efficient for users to download their offline Web pages.

7.2. Future Work

In this research, we designed a framework to achieve the goal of providing offline browsing service. However, there are still plenty of aspects that should be taken into consideration. Most Web pages are designed for PC browsing. If users select these kinds of Web pages, it will consume a lot of time for waiting the download to be completed. Moreover, sometimes our system can only offer an offline webpage which is similar to, not exactly the same as, the one provided by OWS. Photos and texts are usually able to be preserved. However, some kinds of multimedia data, such as flash objects, audio and video streams (music or movie data), or interactive instant messages, are still not able to be kept or received offline. Finally, we will replace the current implementation from Google Gears to the HTML 5 standard. These issues will be considered in our future framework design.

References

1. Yi Wang , Jialiu Lin , Murali Annavaram , Quinn A. Jacobson , Jason Hong , Bhaskar Krishnamachari , Norman Sadeh, A framework of energy efficient mobile sensing for automatic user state recognition, Proceedings of the 7th international conference on Mobile systems, applications, and services, June 22-25, 2009, Kraków, Poland.
2. Ch. Borst, T. Wimb¨ock, F. Schmidt, M. Fuchs, B. Brunner, F. Zacharias, P. Robuffo Giordano, R. Konietschke, W. Sepp, S. Fuchs, Ch. Rink, A. Albu-Sch¨affer, and G. Hirzinger. Rollin’

Justin - Mobile Platform with Variable Base. In Proceedings of the IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009.

3. Yung-Wei Kao, Pin-Yin Peng, Sheau-Ling Hsieh, Shyan-Ming Yuan, A Client Framework for Massively Multiplayer Online Games on Mobile Devices, in Proc. of International Conference on Convergence Information Technology (ICCIT2007), pp. 48-53,Nov. 21~23, 2007.
4. Han, R., Bhagwat, P., LaMaire, R., Mummert, T., Perret, V., & Rubas, J., Dynamic adaptation in an image transcoding proxy for mobile Web browsing, IEEE Personal Comm., 5(6), 8–17, 1998
5. Yung-Wei Kao, Tzu-Han Kao, Chi-Yang Tsai, and Shyan-Ming Yuan, A Personal Webpage Tailoring Toolkit for Mobile Devices, in Computer Standards & Interfaces (SCI), 31(2), pp. 437-453, February 2009.
6. Y. Borodin, G. Dausch, and I. V. Ramakrishnan. TeleWeb: Accessible Service for Web Browsing via Phone. In W4A Conference, 2009.
7. Lum, W. Y., & Lau, F. C. M., A context-aware decision engine for content adaptation, IEEE Pervasive Computing, 1(3), 41-49, 2002.
8. Adnan Al-Bar and Ian Wakeman, A Survey of Adaptive Applications in Mobile Computing, IEEE International Workshop on Smart Appliances and Wearable Computing, Phoenix, AZ US, April 2001.
9. Trish Andrews, Robyn Smyth, and Richard Caladine, Utilizing Students' Own Mobile Devices and Rich Media: Two Case Studies from the Health Sciences, in Proc. of 2010 Second International Conference on Mobile, Hybrid, and On-Line Learning, 2010.
10. Google Gears, [online], Available: http://gears.google.com/.
11. Gears-Monkey, [online], Available: http://code.google.com/p/gears-monkey/.
12. Greasmonkey, [online], Available: https://addons.mozilla.org/zh-TW/firefox/addon/748 .
13. I. Hickson (Editor). HTML 5. Technical report, Web Hypertext Application Technology Working Group HTML 5, 2007. Working Draft, Available: http://www.whatwg.org/specs/web-apps/current-work.
14. James J. Kistler and M. Satyanarayanan, Disconnected Operation in the Coda File System, ACM Transactions on Computer Systems, 10(1), 1992.
15. Edgar Gonçalves, Offline execution in workflow-enabled Web applications, in Proc. of Sixth International Conference on the Quality of Information and Communications Technology, pp. 204-207, 2007.
16. Anupam Joshi, Sanjiva Weerawarana, and Elias N. Houstis, On disconnected browsing of distributed information, in Proc. of Seventh IEEE Intl. Workshop on Research Issues in Data Engineering (RIDE), pp. 101-107, 1997.
17. Ramanathan Kavasseri, Todd Keating, Mike Wittman, Anupam Joshi, Sanjiva Weerawarana, Web Intelligent Query-Disconnected Web Browsing using Cooperative Techniques, in Proc. of 1st. IFCIS Intl. Conf. on Cooperative Information Systems, pp. 167-174, 1996.
18. James E. Pitkow and Mimi Recker, Integrating Bottom-up and Top-down Analysis for Intelligent Hypertext, in Proc. of Third International Conference on Information and Knowledge Management, Intelligent Hypertext Workshop, 1994.
19. Ganesh Ananthanarayanan, Sean Olin Blagsvedt, Kentaro Toyama, OWeB: A Framework for Offline Web Browsing, Web Congress, LA-Web '06. Fourth Latin American, pp. 15-24, 2006.
20. Q. Yang and H. H. Zhang. Integrating web prefetching and caching using prediction models, World Wide Web, 4(4), pp. 299-321, 2001.
21. Swaminathan Sivasubramanian, Michal Szymaniak, Guillaume Pierre, and Maarten van Steen, Replication for Web hosting systems, ACM Computing Surveys (CSUR), 36(3), pp. 291-334, 2004.

22. Allan Heydon, Marc Najork, Mercator: A Scalable, Extensible Web Crawler, World Wide Web Journal, Volume 2, Issue 4, pp. 219-229, 1999.
23. Teleport Pro, [online], Available: http://www.tenmax.com/teleport/pro/home.htm.
24. Allaire, J., Macromedia flash MX- A next-generation rich client, Macromedia White Paper, 2002.
25. Moroney, L., Introducing Microsoft Silverlight 2.0. Microsoft Press, 2008.
26. T. Noda, S. Helwig. Rich Internet Applications, Technical Comparison and Case Studies of AJAX, Flash, and Java based RIA, 2005. http://www.uwebc.org/opinionpapers.
27. Duhl, J., Rich Internet Applications. White Paper, IDC, November 2003.
28. Daniel Peintner, Harald Kosch, Jörg Heuer: Efficient XML Interchange for rich internet applications, in Proc. of 2009 IEEE International Conference on Multimedia & Expo (ICME 2009), 2009.
29. Mike Chambers, Daniel Dura, Dragos Georgita, and Kevin Hoyt. Adobe AIR for JavaScript Developers Pocket Guide. O'Reilly Media, April 2008. ISBN: 978-0-596- 51837-0.
30. J. J. Garrett, Ajax: A new approach to Web Applications, http://www.pablolfc.com.ar/leer/Ajax.pdf, Feb. 2005.
31. A. Carreras, J. Delgado, E. Rodriguez, V. Barbosa, M.T. Andrade, H. Kodikara Arachchi, S. Dogan, and A.M. Kondoz, A Platform for Context-Aware and Digital Rights Management-Enabled Content Adaptation, IEEE Multimedia, Vol. 17, No. 2, pp. 74-89, April-June 2010.
32. Marcos Forte, Wanderley Lopes de Souza, Antonio Francisco do Prado, Using ontologies and Web services for content adaptation in Ubiquitous Computing, Journal of Systems and Software, v.81 n.3, p.368-381, March, 2008.
33. Jin Jing, Abdelsalam Helal, Ahmed Elmagarmid, Client-server computing in mobile environments, ACM Computing Surveys 31 (2), p.117–157, 1999
34. Mobile version of del.icio.us, [online], Available: http://m.delicious.com/.
35. del.icio.us, [online], Available: http://delicious.com/.