Design and Performance Studies of an Adaptive Scheme for Serving Dynamic Web Content in a Mobile Computing Environment

Zhigang Hua, Xing Xie, *Member*, *IEEE*, Hao Liu, *Student Member*, *IEEE*, Hanqing Lu, and Wei-Ying Ma, *Senior Member*, *IEEE*

Abstract—Currently, people gain easy access to an increasingly diverse range of mobile devices such as personal digital assistants (PDAs), smart phones, and handheld computers. As dynamic content has become dominant on the fast-growing World Wide Web [24], it is necessary to provide effective ways for the users to access such prevalent Web content in a mobile computing environment. During a course of browsing dynamic content on mobile devices, the requested content is first dynamically generated by remote Web server, then transmitted over a wireless network, and, finally, adapted for display on small screens. This leads to considerable latency and processing load on mobile devices. By integrating a novel Web content adaptation algorithm and an enhanced caching strategy, we propose an adaptive scheme called MobiDNA for serving dynamic content in a mobile computing environment. To validate the feasibility and effectiveness of the proposed MobiDNA system, we construct an experimental testbed to investigate its performance. Experimental results demonstrate that this scheme can effectively improve mobile dynamic content browsing, by improving Web content readability on small displays, decreasing mobile browsing latency, and reducing wireless bandwidth consumption.

Index Terms—Mobile computing, adaptive content delivery, dynamic content, small form factors, Web content adaptation, fragment caching.

1 Introduction

Nowadays, mobile computing devices are becoming very popular in people's daily lives. Through such portable devices, the mobile users can easily access the World Wide Web (WWW) anytime and anywhere. As dynamic content becomes dominant on the Web [24], it is highly essential to enable the mobile users to effectively browse such prevalent Web content on these devices. To let the mobile users really enjoy the convenience and ease of browsing dynamic content in a mobile computing environment, there still exist several hurdles to be crossed [17]. The main difficulties can be ascribed to three constrained factors on mobile devices, namely, limited bandwidth, small screen, and thin computing capacity:

- Limited network bandwidth will cause considerable latency for the mobile clients to access dynamic Web content as such content has to be dynamically generated by remote Web server and transmitted through limited wireless bandwidth to reach the mobile clients.
- A small screen will degrade the content readability on mobile devices as current Web content is always designed with desktop PCs in mind [6].
- Additionally, thin computing capacity will further increase the browsing latency for the mobile clients.

A number of studies have been carried out with varying foci concerning Web content adaptation for small displays or Web caching for content delivery. However, none of these existing approaches has comprehensively considered improving dynamic content access in a mobile computing environment. In this paper, we proposed an adaptive scheme called MobiDNA, aiming to comprehensively improve the dynamic content browsing experience for mobile users. By using fragment information widely available in dynamic Web content, we developed a novel content adaptation algorithm to improve Web content readability on small displays and designed an enhanced caching strategy to reduce the browsing latency for mobile clients.

The rest of this paper is organized as follows: Section 2 presents related work. Section 3 introduces a framework of the MobiDNA system. Section 4 presents the MobiDNA system implementation. Section 5 describes the experimental testbed, including the selection of a dynamic Web application (i.e., IBuySpy). In Section 6, we present the experimental results and corresponding analysis. Finally, we conclude this paper and introduce future work in Section 7.

- *Z. Hua is with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280. E-mail: hua@cc.gatech.edu.*
- *X. Xie and W.-Y. Ma are with the Web Search and Mining Group, Microsoft Research Asia, 5/F Sigma Center, No. 49 Zhichun Road, Beijing, 100080, China. E-mail: {xingx, wyma}@microsoft.com.*
- *H. Liu is with the Computer Science Department, Stanford University, Stanford, CA 94305. E-mail: haoliu@stanford.edu.*
- *H. Lu is with the Institute of Automation, Chinese Academy of Sciences, No. 95 East Road, Zhong Guan-Cun, Beijing 100080, China. E-mail: luhq@nlpr.ia.ac.cn.*

Manuscript received 30 Apr. 2005; revised 27 Oct. 2005; accepted 6 Feb. 2006; published online 16 Oct. 2006.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-0123-0405.

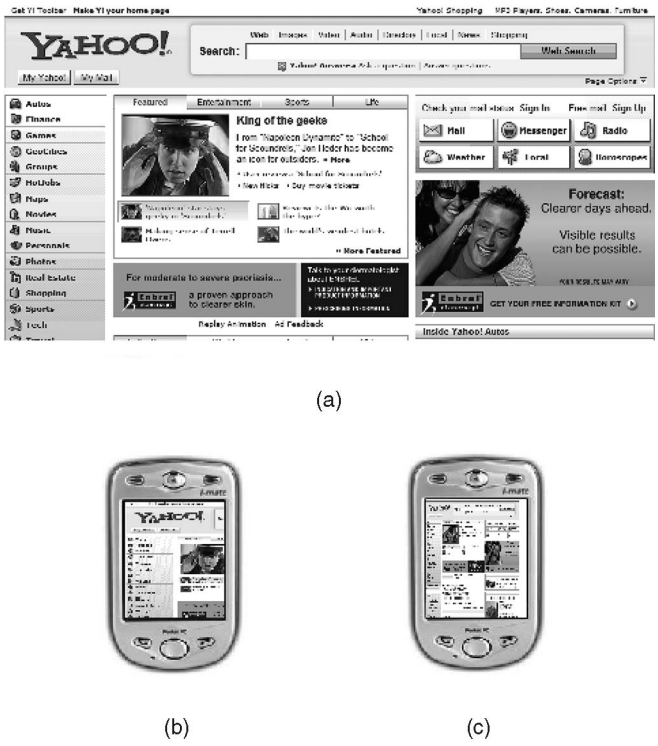


Fig. 1. (a) Screenshot of Yahoo! Page. (b) Original display on a Pocket PC. (c) Thumbnail presentation on a Pocket PC device.

2 Related Work

There are a number of studies concerning various foci on Web content adaptation or dynamic content delivery. We look into these related works in the following sections.

2.1 Web Content Adaptation

As shown in Fig. 1, the original Yahoo portal page (Fig. 1a) is too large to display on a Pocket PC device with a 240×320 resolution (Fig. 1b). In this case, the number of user interactions will be heavily increased to scroll through various parts of the page. To generate tailored display of large Web content on small devices, Su et al. [20] proposed a method that presents a thumbnail view for Web content, as shown in Fig. 1c. Although this presentation style allows users to easily access an overview of a large Web page on small screens, manual interactions like panning and scrolling are also largely required in a mobile browsing session, which are still difficult for the mobile users to browse.

To reduce resource consumption in a mobile computing environment, previous work [11] proposed a transcoding method that changes data quality in order for applications to use the minimum amount of energy when processing it. Nobel et al. [13] designed application-aware adaptations to retrieve and present data at varying degrees of fidelity for mobile clients. Apart from content quality adaptation, some studies [2], [3], [4], [5], [6], [8], [9], [10], [16], [19] focused on Web page layout modification techniques to meet the restrained capability and limited bandwidth on mobile devices. The reauthoring technique [2] required Web pages to have sections and section headers, which, however, are rarely used in Web page authoring today because large manual maintenance efforts are needed. According to a previous study [9], the Web page is reformatted on the basis of page annotation. However, this approach requires a practical solution to facilitate the creation of annotations for existing Web pages.

Although the content transcoding and page layout modification technologies can reduce wireless resource consumption, they change the original Web content quality or layout and do not consider improving Web content readability and interaction on small screens. In our work, we focus on developing a new and feasible content adaptation method to improve dynamic Web content readability and enhance user interaction on small-screen devices.

One significant method to increase Web content readability on small screens concerns extracting fine-grained blocks from large Web pages. Currently, the page segmentation technique has been widely used to segment large Web pages into small blocks that can fit into small displays. Yu et al. [23] proposed a vision-based page segmentation (VIPS) that makes use of page layout features such as font, color, and size, etc., which can efficiently keep related content within a page together while separating semantically different blocks from each other. However, the VIPS algorithm considers maintaining content integrity prior to generating tailored content display, consequently failing to ensure a tailored display of the segmented blocks on small screens. Based on a visual analysis of HTML elements, Chen et al. [6] presents a useful page splitting algorithm that can effectively partition large Web pages into a series of tailored content blocks. However, there are still several problems for these page segmentation techniques. First, it remains a great challenge to achieve satisfactory precision for page segmentation that is mainly based on a pure analysis of HTML elements. Second, mobile client latency will also be heavily increased by executing such content adaptation functionality, which is usually time-consuming, especially for mobile devices that are usually empowered with poor computing capacity.

2.2 Dynamic Content Caching

The caching strategy has been widely used to reduce bandwidth usage and accelerate dynamic content access in a mobile computing environment. Generally, the conventional page-level caching cannot function effectively for dynamic content, as a small change in a page will lead to renewed Web content generation and transmission. A number of new caching strategies [1], [7], [12], [22], [24], [25] have been proposed. Among them, the fragment-level caching strategy has been proven to be the most practically effective. Commonly, the fragment units are widely available in dynamic Web content, which has been sufficiently demonstrated and verified in previous work [15].

Conceptually, a fragment unit is a portion of a dynamic page that has a distinct theme or functionality and is distinguishable from other parts of the page [15]. In current fragment technologies like Edge Side Includes (ESI) [7] and Proxy+ [25], dynamic Web content is encoded with markups indicating cacheable characteristics. Generally, the template of a dynamic page is designed to contain references to its included fragments. Each fragment is treated as a separate object in a dynamic page and has its own cache and access profile described in a configuration file. For instance, the content providers may want to cache the template of a page for several days or even several months, but only cache a particular fragment (e.g., a stock

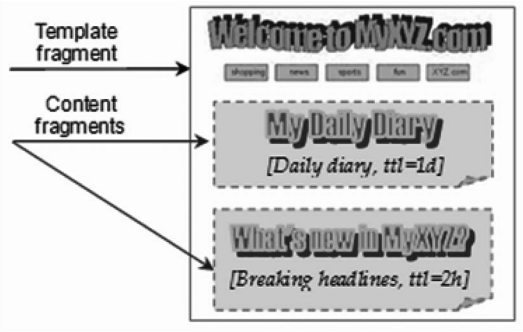


Fig. 2. An example for a dynamic page comprising a template fragment and two content fragments.

quote) for a matter of seconds or minutes. If a dynamic page is considered as an individual, its lifetime will be limited to the fastest changing fragment. Therefore, the page-level caching will be less useful because a page's cache lifetime is determined by the fastest changing fragment.

As for an example of a dynamic page comprising cacheable fragments, consider the dynamic page shown in Fig. 2, which includes a template fragment and two content fragments. The template fragment usually consists of common elements of a page, such as the logo, navigation bars, and other "look and feel" elements. Generally, the content fragments represent dynamic subsections of the page, and each one has its own lifetime period. In the example, the cache lifetime of the template fragment is set to three months, while the lifetimes of the *diary* and *news* fragments are set to one day and two hours, respectively.

The advantages of the fragment-level schemes are obvious and have been conclusively demonstrated. A previous study [24] showed that about $2 \sim 3$ folds of latency reduction can be achieved and over 70 percent client Web requests can be processed from cache. This work has also investigated different offloading and caching options and concluded that simple augmentation at Web proxies enabled with fragment caching and page composition could achieve most of the benefits. Yuan et al. [25] developed an approach to enable fragment caching at Web proxies to speed up dynamic content delivery, which only requires simple modifications to existing Web applications.

However, in a mobile computing environment, mobile clients cannot benefit from fragment caching at proxy side because the download time of Web content in a mobile environment is determined by the last mile, i.e., the slow dial-up link for mobile devices to access the Web. Rabinovich et al. [14] proposed to push the well-known ESI [7] fragmentation scheme to the ultimate wireless network edge—the mobile client side—and its experimental results demonstrated that response time can be greatly speeded in this way. We found that only speeding up dynamic content delivery is still insufficient to reduce the dynamic content browsing latency in a mobile computing environment. The reason is that the Web content adaptation process usually required in the mobile client side will also cause considerable delay. However, this delay has not yet been considered for optimization.

As above, current approaches are mainly focusing on one limited aspect, while none of them has comprehensively considered improving dynamic Web content access in a mobile computing environment. In this paper, we introduce an adaptive scheme for serving dynamic content to the mobile users.

3 System Framework

This section presents the framework of the MobiDNA system. We first state the benefits by integrating the content adaptation and caching functionalities in our system to improve the overall performance. We then describe the content adaptation algorithm and the enhanced content caching strategy.

3.1 Integration of Web Content Adaptation and Caching

Currently, Web content is primarily designed with desktop PCs in mind, which is usually too large to display on the small screens of mobile devices. Although a number of studies have been conducted to improve Web content readability on small displays, none of them has taken dynamic content into special consideration.

Consider a process of accessing dynamic content on a mobile device. First, the requested content is dynamically generated by a remote Web server and then transmitted over a wireless network to reach the mobile client. Second, content adaptation is employed to adapt Web content for small displays. This process comprises both network transfer and page adaptation, which greatly increase browsing latency for the mobile clients. Generally, content adaptation is used to improve Web content readability, and caching is used to speed up content delivery. Two such functionalities are always treated separately. However, we found that this separation will degrade the overall performance in a mobile computing environment:

1. As stated previously, current Web content segmentation methods are mainly based on a pure analysis of HTML elements to acquire content blocks. Consequently, the precision of a content structure analysis based on the HTML level is always challenging. Differing from static Web content, dynamic content usually includes fragment units [15] indicating page structure. However, such indicative fragment information in dynamic content has not been exploited for use in a content adaptation process.
2. Though fragment caching can reduce latency by decreasing data transmission, it is still insufficient to reduce mobile client latency since the additional Web content adaptation process also leads to considerable delay. If the caching functionality does not consider saving Web content adaptation results, the content adaptation process has to be repeatedly executed, which will still cause considerable latency and impose processing load for mobile client devices.

Based on the drawbacks of separating content adaptation and caching functionalities, we propose an adaptive scheme called MobiDNA that integrates two functionalities by utilizing the widely available fragment information in dynamic content. In the following, we introduce a modified dynamic content adaptation algorithm and an enhanced content caching strategy.

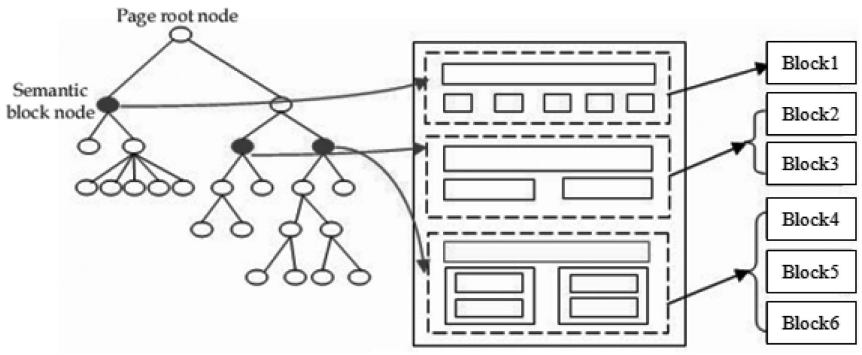


Fig. 3. (a) Two-step dynamic content adaptation process: Three semantic blocks are identified according to the fragment information. (b) The detected semantic blocks are further partitioned to generate tailored blocks that fit into the small screens.

3.2 A Modified Web Content Adaptation Algorithm

Conventionally, a fragment is designed with a distinct theme and is distinguishable from the other parts of a dynamic page [15]. Consequently, one fragment is assumed to be an integrated content unit that indicates informative cues for Web page content structure. In light of this evidence, we develop a novel dynamic Web content adaptation algorithm by utilizing fragment information.

Before describing our Web content adaptation algorithm, we first introduce a new notation named *semantic block*, which is defined as a continuous content unit that does not include two or more fragments within its content scope. Accordingly, the template fragment of a dynamic page is usually not a semantic block as it always contains content fragments, while the content fragments (e.g., *news* and *diary* fragments in Fig. 2) are usually candidate semantic blocks. Specially, a continuous HTML unit that doesn't include any fragment is usually referred as a semantic block (e.g., the logo and navigation bar unit in Fig. 2).

We describe our modified content adaptation algorithm in two steps, namely, semantic block detection and semantic block partition. Fig. 3 presents an example for this algorithm that works on the dynamic page presented in Fig. 2. First, all the semantic blocks are detected in a dynamic page based on the identification of fragment units (Fig. 2a). Second, the detected semantic blocks are further analyzed to generate fine-grained tailored content blocks based on the HTML layout analysis (Fig. 2b). We detail the two processing steps in the following.

Semantic block detection. We developed a simple yet effective algorithm for semantic blocks detection in dynamic content, which can be summarized in Fig. 4. In the figure, *is-sem-block()* is a function to justify whether an HTML node T_i is a semantic block according to the predefinition. If the node T_i satisfies the condition specified by the predefinition, it is inserted into a semantic block set that is initialized as null. In this way, we can obtain a set of semantic blocks from a dynamic page. Note also that page decomposition in this level considers content integrity prior to tailored presentation on small displays. As a result, the generated semantic blocks cannot insure tailored display on small screens. We carry out the second step to partition semantic blocks into tailored blocks to fit into small displays.

Semantic block partition. We summarize the semantic block partition algorithm in Fig. 5. As shown in the figure, *fit-for-display()* is a function to examine whether a semantic block fits into the small screen of a mobile device, and *no-child-node()* is a function to verify whether a semantic block contains children nodes for further splitting. Line 5 specifies that a semantic block does not need further splitting if the block itself fits into a small display or has no children nodes. If a semantic block does not meet this condition, its structure will be analyzed for further decomposition.

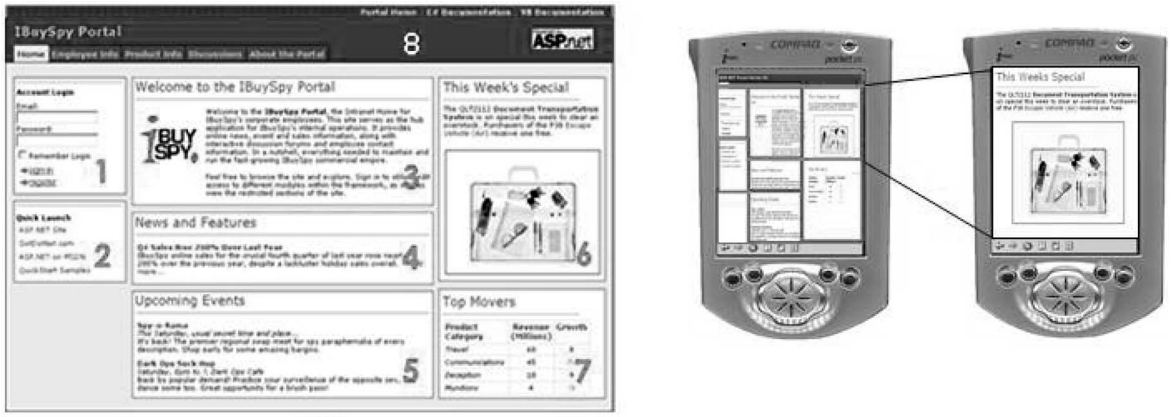
To further split semantic blocks into tailored blocks, we adopted the Web content adaptation algorithm [6] in our MobiDNA system, which was proven to be effective in Web page splitting based on HTML element analysis. In

Algorithm 1: Semantic Block Detection
1: INPUT: T (the root of the HTML DOM tree)
2: OUTPUT: SemBlockSet (a semantic block set initiated as null)
3: DetectSemanticBlock (T)
4: **for** (each child node $T_i \in T$)
5: **if** (is-sem-block(T_i) =TRUE)
6: SemBlockSet $\cup$ $T_i \rightarrow$ SemBlockSet
7: **else** DetectSemanticBlock(T_i)
8: **end for**

Fig. 4. The semantic block detection algorithm.

Algorithm 2: Semantic Block Partition
1: INPUT: SemBlockSet (the set of semantic blocks in a page)
2: OUTPUT: BlockSet (the set of tailored blocks initiated as null)
3: PartitionSemanticBlock(SemBlockSet)
4: **for** each semantic block $S_i \in$ SemBlockSet
5: **if** (fit-for-display(S_i) **or** no-child-node(S_i))
6: BlockSet $\cup$ $S_i \rightarrow$ BlockSet
7: **else** SplitHTML(S_i) $\rightarrow$ BlockSet
8: **end for**

Fig. 5. The semantic block partition algorithm.



(a) (b)

Fig. 6. (a) Tailored blocks acquired from the homepage of IBuySpy application. (b) Tailored representation of the IBuySpy homepage on a small-screen Pocket PC device.

Fig. 5, the function *SplitHTML()* in line 7 represents a function that is implemented according to a previous study [6], which identifies content blocks from the structure of a semantic block in an iterative manner. At the beginning, the the processed semantic block is regarded as a single content block. At each iteration, the content adaptation algorithm finds the best way to partition a content block into smaller ones. In this way, a set of content blocks will remain at the end of the process, which serves as the final result for semantic block splitting.

Practically, we observed that semantic blocks generated in the first step of our algorithm are always de facto tailored blocks. Only a minority of detected semantic blocks require a further partition for generating tailored blocks on small displays, hence resulting in a significant reduction of time complexity of page adaptation. As shown in Fig. 6a, we present an example for the adapted result generated by our content adaptation algorithm. This figure displays the partition results for the homepage of the IBuySpy Web application. Specially, all the generated blocks labeled from 1 to 8 are actually the content fragments included by that page.

Representation style. There exist various styles to represent these generated tailored blocks on small screens [6], such as single-subject and multisubject representation. Generally, single-subject representation is suitable for Web pages that contain the content of a particular topic, such as a news story on the BBC news site. Multisubject representation is more appropriate for the Web site containing multiple topics, such as the homepage of a site. In our approach, we adopted the multisubject representation to illustrate our content adaptation technique. In this case, a Web page is organized into a two-level hierarchy, which presents a global thumbnail overview at the top level that contains index to detailed tailored blocks at the bottom level.

Fig. 6b presents an example for the representation style used in the MobiDNA system. This figure shows the adapted result of the IBuySpy homepage on a Compaq Pocket PC with a display resolution of 240×320. When a user clicks a tailored block (the left part in Fig. 6b) on the top-level thumbnail, the window is navigated to its detailed content (the right part in Fig. 6c), which fits well into the small screen on the Pocket PC device. Furthermore, the user can freely use the back and forward buttons to switch between the top-level thumbnail overview and the bottom-level block to browse the content detail of each block.

3.3 An Enhanced Fragment Caching Strategy

Generally, in fragment caching strategy, the original content of a fragment is saved to cache for releasing the transmission load in the late requests and reducing the accessible latency for end users. However, as we stated above, in the mobile computing environment, only caching the original fragment content is not sufficient for reducing the latency caused by the additional content adaptation process on mobile devices.

To reduce the mobile client latency of accessing adapted dynamic content, we design an enhanced fragment caching strategy that is featured by saving the adapted content instead of the original content to cache. By directly fetching the adapted content of validated fragments stored in cache, this new caching strategy is capable of saving both the transmission and adaptation costs.

In our case, we save the adapted result of dynamic content to cache by adding additional markups that identify the tailored blocks generated by our adaptation algorithm. In Fig. 7, we present an example for the cached content of the template fragment that displays in Fig. 2. As shown in Fig. 7a, the template includes an indicative markup $<\text{block1}/>$ that points to the tailored block labeled *block1*, and two fragment markups that indicate two fragments labeled *frag1* and *frag2*. In Fig. 7b, the fragment *frag1* contains two markups that represent two tailored blocks labeled *block2* and *block3*. In this way, all the tailored blocks generated by our content adaptation algorithm can be directly acquired by identifying these predefined markups. As a result, our method can reduce the repeated processing effort and browsing latency.

3.4 Summarization

As shown in Fig. 7, through an iterative manner that starts from the template fragment of a dynamic page, all cached tailored blocks can be identified through the predefined markups, which indicate the tailored blocks previously

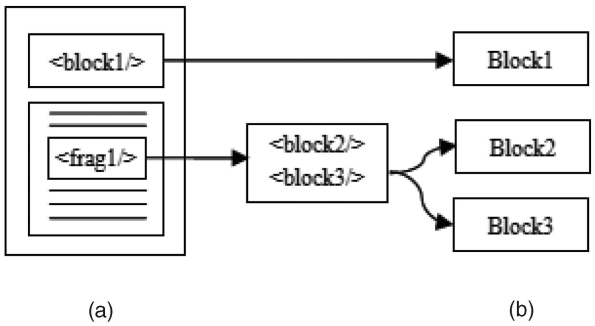


Fig. 7. An example for the cached content labeled with markups. (a) Template fragment. (b) Tailored block.

generated by our algorithm. In our case, only the invalidated or changed fragments in a dynamic page need to be renewedly processed, while tailored blocks of cached fragments can be directly identified without the renewed content processing or analysis effort.

In comparison with the existing Web content adaptation techniques, our approach is characterized as two aspects. First, fragment information widely available in dynamic content is exploited for use in Web content structure analysis instead of pure HTML analysis, which is beneficial to improve the accuracy of Web content structure analysis. Second, the adapted content version saved in cache can be easily analyzed to identify tailored blocks, which can avoid repeated transmission and processing of cached fragments, consequently resulting in a significant reduction of time complexity and processing load. Our experiments verified that this content adaptation method can effectively improve content readability on small screens and help reducing mobile client latency.

4 System Implementation

This section presents the MobiDNA system implementation. We first discuss different deployment options and conclude that mobile client side deployment can achieve the best performance. We then describe the workflow of the client-side implementation.

4.1 Deployment Options

As pointed by previous work [14], there are a number of possibilities to deploy fragment caching, such as Web server, network edge server, or client side. In the following, we look into these three options for deploying our MobiDNA system.

The most common way is to deploy the system at the Web server. In this case, the Web server responds with an adapted version of dynamic content to the mobile clients. Thus, with the adapted content version in cache, the Web server can avoid dynamic content generation and adaptation of the unchanged fragments. However, the outbound traffic from the Web server to the mobile clients has not been reduced since the entire Web content has to be transmitted over a wireless network. As a result, the mobile client latency cannot be reduced. Moreover, the Web server load will be heavily increased by this additional content adaptation service.

The second possibility is to deploy the MobiDNA system at network edge server that resides close to the client side. Generally, this scheme is significantly useful for reducing content delivery when the edge server is enabled with the quality transcoding functionality to meet the client-side capacities. In our case, the edge server is assumed to be capable of assembling a dynamic page by combining cached and changed fragments, and adapting the page as a tailored version to serve the mobile clients. This scheme can save the bandwidth usage for the link from the Web server to the edge server and also can lighten the Web server load. However, in a mobile computing environment, the access to Web content can hardly be accelerated in this case, because the main bottleneck of Web content delivery in a wireless network lies in the last mile, i.e., the slow dial-up link that connects mobile devices to Internet [14]. Additionally, the security issue also becomes a concern in this case, since the edge server is empowered with the ability of substituting or modifying the original content downloaded from content providers.

Furthermore, both the Web server and the edge server options may lead to cache explosion problem, because various characteristics of heterogeneous client devices (i.e., screen size) may result in different cache versions even for the identical Web content. For example, desktop PC and PDA acquire various responses from the Web servers for the same fragment or page. Therefore, the Web server or edge server needs to store various tailored versions in the cache to meet the different client characteristics. This will explosively increase the cache storage volume especially with the fast-growing types of heterogeneous computing devices. Additionally, the Web server or edge server should be implemented to be aware of client capacities for providing an appropriate adapted cache content version.

Therefore, similarly to a previous study [14], to better serve the mobile clients, we deploy our adaptive MobiDNA system at the real network edge, i.e., the mobile client side. In this case, mobile clients only request and download invalidated fragments, which are combined with local cached fragments to assemble an entire page. The page is handled by our modified content adaptation method, which can generate a tailored display to the mobile users. Moreover, the adapted content of these invalidated fragments are saved to client cache.

Although this client-side deployment scheme causes additional resource consumption on mobile devices (e.g., CPU, memory, and battery power), we think it is worthy as it can effectively improve content readability and reduce user latency for dynamic content access in a mobile computing environment. As we will see in our experiments shown in Section 6, the additional overhead caused by the MobiDNA on the mobile clients can be effectively offset by the reduction of transfer time and bandwidth consumption. To our knowledge, deploying the MobiDNA scheme at the mobile client side can achieve at least three main advantages:

- Wireless bandwidth and mobile client latency can be greatly reduced because less Web data is transmitted over the last mile in a wireless network.

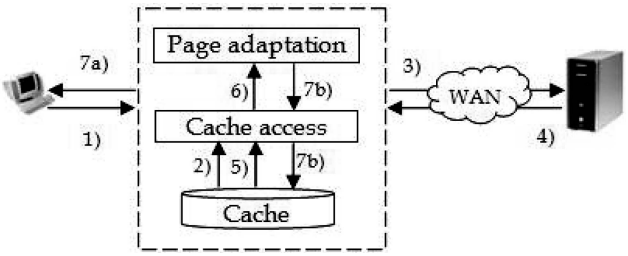


Fig. 8. The workflow of the MobiDNA system on the mobile client side.

- Cache explosion can be effectively avoided because adapted content cache versions in one mobile device are always fixed for the identical Web content.
- The edge server commonly used for speeding up dynamic content delivery is not a required facility, resulting in a significant reduction of network deployment complexity.

4.2 Client-Side Implementation

In this section, we detail the MobiDNA implementation in the mobile client side. The system should be able to request the invalidated fragments instead of the whole dynamic content, assemble all fragments to assemble an entire page, and adapt the page as a tailored presentation on small screens. As shown in Fig. 8, the workflow can be formulated as:

1. A mobile client sends an HTTP request to a dynamic page.
2. MobiDNA examines cache status of the requested page and its included fragments. If all items are validated in the local cache, go to Step 5 and a tailored presentation is returned immediately to the user; otherwise, go to Step 3.
3. MobiDNA attaches a list of notations identifying validated cache fragments to the HTTP request header and forwards it to the Web server.
4. The Web application generates a response containing additional markups that indicate cacheable fragments. Specially, the Web application does not generate and respond the validated fragments indicated by the HTTP request header.
5. MobiDNA parses the received content and assembles an entire page by fetching the adapted fragment content from local cache.
6. The assembled page is adapted by our adaptation algorithm to generate tailored blocks on small screens.
7.
   a. A complete tailored display is presented to the mobile user.
   b. Simultaneously, the adapted content of downloaded fragments are saved to cache.

Practically, we implemented a browser application capable of MobiDNA functionality, which was tuned to 240×320 in size. We set cache storage to 4MB, which is automatically maintained by the most used least-recently-used (LRU) strategy. Moreover, we provided this client cache with convenient APIs for write and read access.

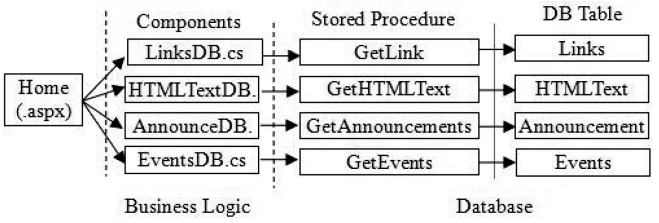


Fig. 9. An example of the three-tier architecture for the portal page of the IBuySpy application.

5 Experimental Design

To validate the practicality of the MobiDNA system, we selected the IBuySpy application as a test benchmark and constructed an experimental testbed for system evaluation.

5.1 The IBuySpy Benchmark

Logically, most of the dynamic Web applications today can be roughly partitioned into three tiers in architecture: presentation, business logic, and back-end database. Whether the users create an application for conducting online commerce or accessing data in legacy systems, etc., the design always follows such a three-tier architecture.

We select a typical three-tier Microsoft.Net application, i.e., IBuySpy, as a benchmark. Web sites like this are among the most common dynamic applications on the Web. It is a representative dynamic Web application and contains pages with complicated modular HTML structure. The IBuySpy application is implemented by ASP.Net, and its source code is available in [21]. We look at the portal page of the IBuySpy application as an example for illustrating the interaction between the three tiers of the IBuySpy application in Fig. 9. The *presentation tier* communicates with client browsers directly. It contains *Web Forms pages* (*aspx* files), *Web Forms user controls* (*ascx* files), and their *code-behind* classes. Web Forms pages represent dynamic pages, and Web Forms user controls represent independent content units (i.e., fragments) of Web Forms. When a request arrives, the specified Web Forms page and Web Forms user controls are dynamically loaded. The objects in the *business logic tier* accept invocations from the presentation tier. The *database tier* consists of application data and *stored procedures*. Using the ASP.Net fragment technology [25], the fragments in the IBuySpy application can be marked automatically by simply overriding two basic classes in IBuySpy namespace, namely, *Page* and *UserControl*.

5.2 Experimental Testbed Construction

Fig. 10 shows the experimental testbed we used for our experiments. The network testbed consists of a client machine, an edge server,[1] and a Web server. In the testbed, we used a low-performance desktop PC to simulate a mobile computing device, and Shunrasoftware [18] as an emulator to simulate a wireless network. This simulated wireless network was used to enable flexible experimentation settings with different network characteristics in our study.

1. Here, we still adopted an edge server to keep aligned with the real wireless network case, although the edge server is not a required facility for our system.

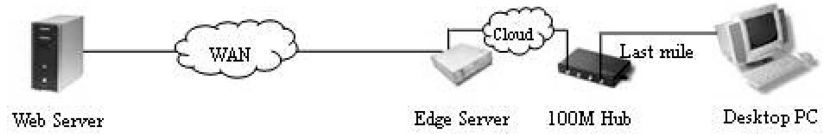


Fig. 10. The experimental testbed.

TABLE 1
The Configurations of the Web Server and the Edge Server Machines in the Testbed

	Web server	Edge server
CPU	Intel Xeon CPU 3.06GHz CPU	Intel Xeon CPU 2GHz
Disk storage	120GB	80GB
RAM memory	2.0GB	1.0GB
Operation system	Microsoft Windows Server 2003	Microsoft Windows Server 2003
Service	IIS6.0 + SQL Server + ASP.Net IBuySpy	Shunra\Cloud

TABLE 2
The Configurations of the Simulated Client Machine in the Testbed

	Compaq iPAQ 3830/3850 PocketPC	Simulated client machine
CPU	206 MHz Intel ARM RISC Processor	200MHz Intel Pentium Processor
RAM memory	64MB	64MB
Operation system	Microsoft Pocket PC 2002	Microsoft Windows2000 Professional
Resolution	240x320	240x320 (tuned window size)

As shown in Table 1, we adopted two server machines with sufficient power so that they never become processing bottlenecks in test runs. We deploy the IBuySpy application on the Web server to provide dynamic content for the client side. The adopted Web server is a powerful machine with an Intel Xeon 3.06GHz CPU and 2GB RAM that runs the Microsoft Windows Server 2003 operation system. The database that supports the IBuySpy backend data operation is SQL Server 2000, and Microsoft Internet Information Service (IIS6.0) is used to publish the IBuySpy service. The edge server is also a powerful machine with an Intel Xeon 2GHz CPU, 80GB disk SCSI disk, and 1GB RAM memory.

The simulated client system is summarized in Table 2. The adopted client desktop PC is quite modest by current desktop PC performance standards, and its configuration was selected to simulate a modern PDA device. Generally, a simulated machine should satisfy several restrained conditions in mobile devices like limited bandwidth, small display, and thin-computing capacity. The configuration of this simulated machine refers to Compaq iPAQ 3830/3850 Pocket PC, a popular and representative PDA. To simulate the small display size of a PDA, we tuned the client browser window to 240×320 in size.

For simplicity and good network performance, the testbed was configured using 100BaseT full duplex switched network connections between all wired machines. The client is also connected to the edge server through a 100Mbps Ethernet switch. In order to avoid contentions between client–edge communication and edge–server communication, two network adapters are installed on the edge server machine. The two portions of traffic will go through different network adapters and switches. Furthermore, the added latency of routing through the machines was neglected practically. Because all tests with the network were conducted within a limited physical scope, we considered the amount of packet loss to be negligible in our experiments.

To provide similar network conditions in a wireless environment, we used a network emulator to limit the accessible bandwidth and specify network latency for the client machine. Cloud [18] is a minor and effective software to control bandwidth limitation and network latency, which was deployed in an edge server and only imposes minor additional loads. As the experimental testbed is prepared, we set out to estimate the effects of the proposed MobiDNA system.

6 Performance Studies

To evaluate the effectiveness of our MobiDNA scheme, we carried out experiments over two typical wireless networks (25kpbs and 56kpbs[2]) under three latencies set by Cloud (500, 2,000, and 5,000 ms). Every time a request is started from the client, it travels through the delayed link and so does its response. Therefore, the response time is always influenced by the network latency.

We built a general browsing sequence in the IBuySpy site, which corresponds to a typical user browsing pattern for such Web applications. In Table 3, we present the size and the number of the included fragment for each dynamic page in this sequence. We developed a program that runs on the client machine to simulate a client Web browser for accessing dynamic content. The actual behavior of the program is controlled by a test script. In each test, the program repeats the following steps until the test duration is over. First, it opens a persistent HTTP connection to the Web server. Second, it starts a request of dynamic content to the Web server. Third, after receiving the response and

2. 25kbps is a typical upload bandwidth for the GPRS network, and 56kbps is a typical downstream bandwidth for the dial-up links.

TABLE 3
The Bytes and Fragment Numbers of IBuySpy Pages

Requested Web pages in IBuySpy site	Page size (byte)	Number of included fragments
Homepage	25,371	9
Employee	13,263	6
Product	24,750	8
Discussion	16,945	6
About	49,950	9
More details	6,717	3
Register page	7,108	2

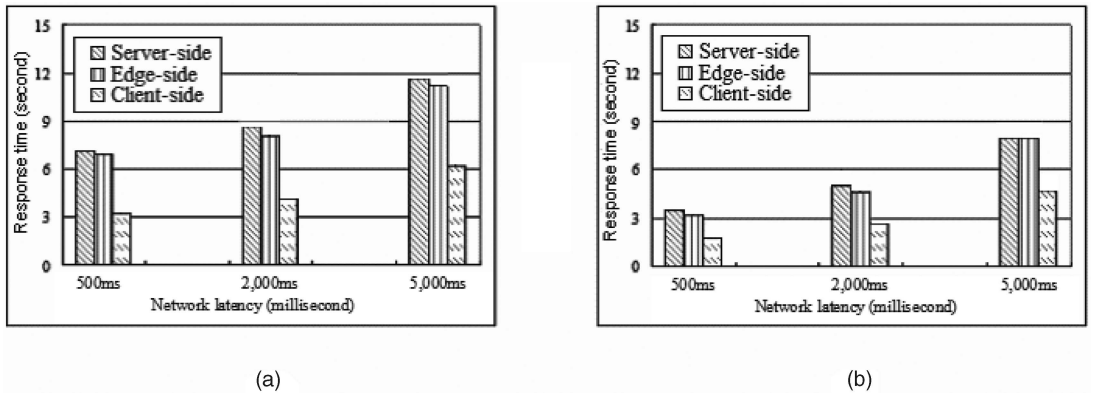


Fig. 11. Response time under two networks: (a) 25kbps and (b) 56kbps.

adapting the page for a tailored display, it waits for some thinking time (60 milliseconds in our tests) and chooses the next request and so forth. If the program chooses to exit, it stops sending requests and closes the connection.

In a traditional page-level caching strategy, the cache hit ratio (*CHR*) is widely adopted for evaluation, which is defined as the ratio of the number of pages found in cache and the total pages requested. However, *CHR* is no longer a valid criterion for fragment-level caching, in which the Fragment Cache Hit Ratio (*FCHR*) is used instead. FCHR is defined as the ratio of the number of fragments validated in the cache and the total fragments in all the pages requested.

In the following, we first provide the comparison of response time in three different deployment options. Second, we investigate the reduction of the mobile client latency with the MobiDNA system. Third, we present the network bandwidth reduction in the MobiDNA system. Finally, we conduct a user study to investigate the Web content readability in our MobiDNA system.

6.1 Comparison of Deployment Options

As has been described in Section 4.1, there exist three possibilities to deploy the MobiDNA system: Web server, edge server, and client side. In the following, we look into the response time of accessing the dynamic pages shown in Table 3 under these three deployment options.

Fig. 11 shows the average response time versus various latencies in the two network bandwidths. In this experiment, we tuned the fragment cache hit ratio to 60 percent for requesting dynamic Web pages in the IBuySpy site. This figure demonstrated that the response time of the deployments on the Web server and edge server are in the similar level, which are much higher than that of the client side. The reason is evident that the response time of downloading Web content is determined by the last mile link in the wireless network.

As we have expected, the client-side deployment can achieve the most significant reduction of response time, since less data needs to be transferred on the last mile in the wireless network, while the other two schemes do not save the last mile traffic. In the example, the client-side scheme has achieved about two times the response time reduction compared to the other two schemes. This experiment verifies the analysis we presented in Section 4.1, that the client-side deployment of our system can achieve the best performance in a mobile computing environment.

However, some may argue that the additional download of fragment markups and dynamic page composition at the mobile client side would increase latency, especially when the client accesses the page for the first time and, hence, no fragments are validated in cache. Practically, this experiment shows that this overhead can be effectively offset.

6.2 Latency Optimization

The client latency is a very important factor that has a key impact on users' Web browsing experience, which becomes especially important in a mobile computing environment due to the limited accessible bandwidth and constrained computing capacity on mobile devices.

Different from desktop PCs in which the display time of Web browsing is mainly caused by network transmission, the mobile browsing latency comprises both the delays

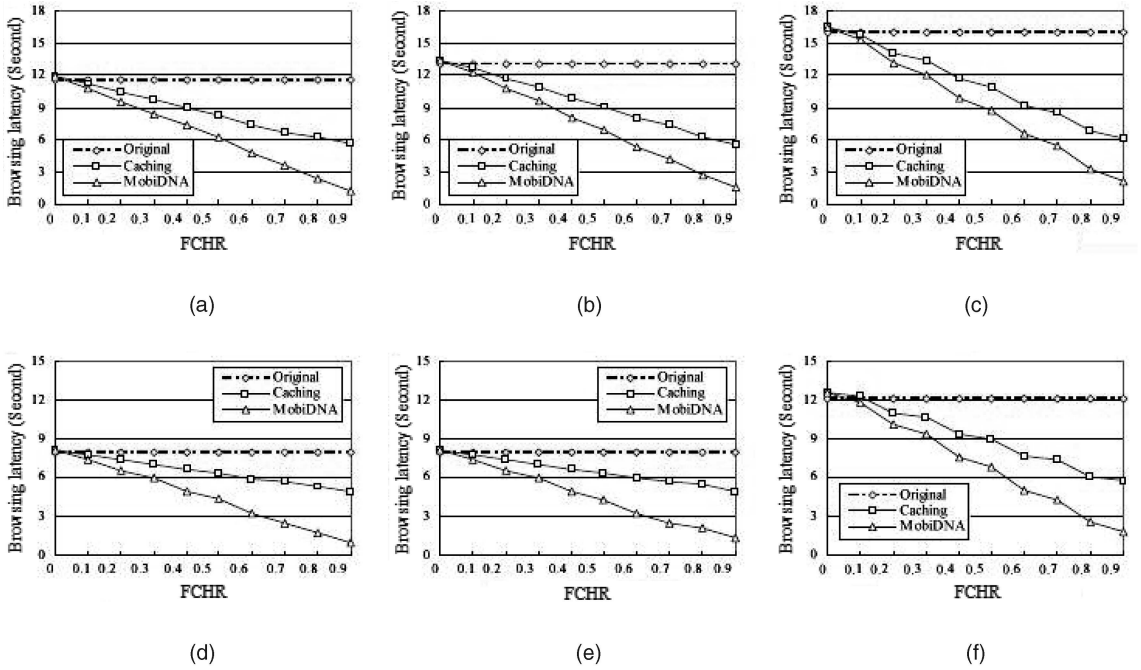


Fig. 12. The mobile browsing latency under two networks: (a)-(c) 25kbps and (d)-(f) 56kbps. (a) Network latency = 500ms. (b) Network latency = 2,000ms. (c) Network latency = 5,000ms. (d) Network latency = 500ms. (e) Network latency = 2,000ms. (f) Network latency = 5,000ms.

caused by network transmission and Web content adaptation. Therefore, pure caching of original content does not suffice to reduce the latency for mobile clients, since content adaptation also causes considerable delay.

We carried out experiments to compare the client latency in three approaches. They are: 1) common cases in which no Web content is saved for cache (*common*), 2) the fragment caching strategy that is deployed in the mobile client side (*caching*), and 3) the adapted content caching approach in the MobiDNA system (MobiDNA). Fig. 12 displays the mobile client browsing latency under 25kpbs and 56kbps bandwidths in various FCHR values.

It is not a surprise to find that a higher FCHR results in a higher level of latency reduction for both fragment caching and MobiDNA. Furthermore, it can also be seen from the figures that, even with the same case of FCHR, our enhanced caching can achieve more latency reduction than the fragment caching strategy. The reason is clear that the MobiDNA system can save both content adaptation and network transmission costs, while only network transmission can be saved in the fragment caching strategy. Moreover, the results also demonstrate that, with the increase of FCHR, the reduction of display time of the MobiDNA system is more pronounced versus the fragment caching strategy. These results clearly indicate the effectiveness of the latency optimization achieved by the MobiDNA system.

6.3 Bandwidth Reduction

Similarly to the fragment caching strategy, MobiDNA can reduce the network transfer load. This experiment was conducted to validate the reduction of network load through the MobiDNA system. We present the transmission bytes by varying FCHR values in Fig. 13. Note that two cases labeled "with markup" and "without markup" in the figure are distinguished by whether to count the fragment markup bytes in network transfer stream. This setting is designed to explore the additional traffic load that the fragment markups actually cause on the network. The result is straightforward: The higher FCHR can lead to a higher reduction of bandwidth consumption. Furthermore, this benefit is more pronounced when FCHR gets higher.

The bandwidth reduction in the MobiDNA system should be in proportion to the FCHR generally. However, we found that the actual saving of transmitted bytes in practice is slightly lower than FCHR. We ascribe the reason to the extra bandwidth consumed by the fragment markups carried by responses. We estimate that this adds about 200 bytes for each fragment that is invalidated in the client cache. However, the slight additional overload caused by the fragment markups can be effectively offset by the saved bytes through our approach.

6.4 User Study

One of the significant effects of the MobiDNA system is its improvement of dynamic content readability on the small

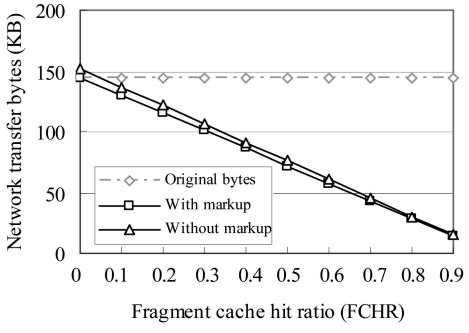


Fig. 13. The network transfer bytes.

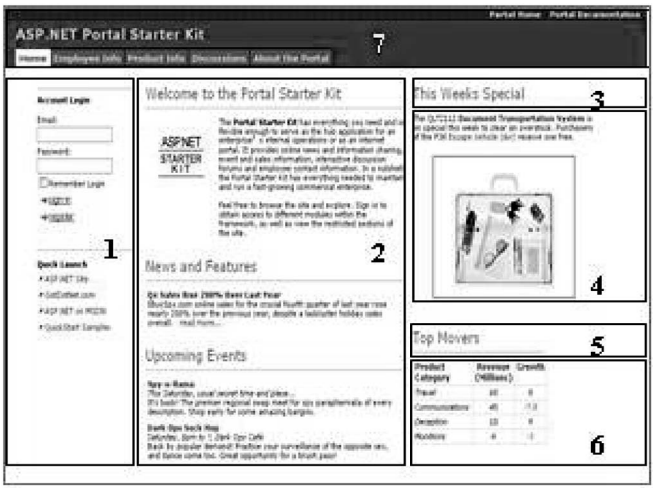


Fig. 14. The adapted result of the IBuySpy homepage by Chen et al. [6].

displays of mobile devices. Judging Web content readability is a subjective task. One way to obtain statistically meaningful results is to average the judgments of different users. We carried out a user study to explore the dynamic content readability on small displays with various display styles.

Different display styles. In this test, we compare three display styles of Web content on small screens. They are: 1) original presentation without Web page adaptation (*common*), 2) adapted display through page adaptation based on pure HTML analysis (*adaptation*) [6], and 3) the adapted display generated by the MobiDNA system. Specially, Fig. 14 presents an example for previous adaptation work [6]. Compared to the MobiDNA system (Fig. 6a), this result fails to maintain information integrity of the content fragments. For example, blocks 3 and 4 are split separately, while they actually constitute an integral fragment. The same problem also occurs with blocks 5 and 6. Additionally, blocks 1 and 2 are too large to display on small screens, which both require further partitioning. As this method does not use fragments for dynamic Web page structure analysis, its adaptation may result in inaccuracy.

User study setting. Our study involved 10 participants comprising six males and four females. All the testers were computer science graduate students who were skillful Web users and were familiar with operations on PDA or Smart Phone. All the participants never had any knowledge of the MobiDNA previously.

The participants were first asked to browse Web pages using a small browser of a 240×320 window size. All the subjects felt the readability was poor and raised the common need for improving Web content readability on small displays. The subjects were then presented an easy instruction on how to operate in the MobiDNA system. We asked the users to browse Web pages over a course of 15 minutes. This exercise would help users get familiar with the MobiDNA operations. We asked the testers to do evaluations which reflect the readability of Web content in various display styles. The users gave their feedback toward these styles based on a simple rating principle ranging from 1 to 5, which reflects how "pleased" the user was with display result:

TABLE 4
The Mean Score of Readability Evaluation in Various Display Styles

Display Styles	Mean Evaluation Score
Common	1.93
Adaptation	3.13
MobiDNA	3.50

- The highest level 5 means that content readability is very good and allows users to easily browse Web content on a small display without much manual effort.
- The middle level 3 means that Web content readability is ordinary and generally does not improve or impair Web content reading on a small display.
- The lowest level 1 means that content readability is very poor and causes a serious browsing problem or information loss on a small display.

All the users were left to decide on the rating level of each display style. The order was varied for different subjects for each dynamic page: Five users were given the order of "*common, adaptation, MobiDNA*" and the other five saw the display results in an order of "*common, MobiDNA, adaptation.*" This balanced ordering could allow the subjects to use *adaptation* and *MobiDNA* equally, consequently removing the bias of use evaluations of different display styles among users.

User study result. In total, we collected 121 evaluations for each display style on the pages in IBuySpy, averaging about 12 pages per user. In Table 4, we present the average rating score for each display style. On average, the adapted result generated by MobiDNA receives the highest score, i.e., 3.50. Generally, the readability of the original page display on small screens was poorly rated by users with the lowest score, indicating that it is very necessary to adapt Web content for improved readability on mobile devices. This result verified that the MobiDNA's dynamic content adaptation algorithm is more effective by utilizing fragment information than the general page adaptation algorithm based on HTML layout analysis. Furthermore, we noticed that the average rating score of the readability on our MobiDNA system was less than 4.0, a reasonable level of satisfaction, reminding us to further improve our method for generating higher quality Web content readability on small screens.

7 Conclusions

Based on the pervasiveness of mobile devices and their easy access to the dynamic content on the Web, in this paper, we proposed an adaptive scheme called MobiDNA for serving dynamic Web content in a mobile computing environment. First, dynamic content is adapted for small displays through a modified content adaptation algorithm by utilizing fragment information. Second, the adapted content is saved to the mobile client cache for reducing network transmission and Web content adaptation costs. We provided detailed implementation for the MobiDNA solution and constructed an experimental testbed to evaluate its performance. The experiments showed that our approach

can improve dynamic content readability on small displays, decrease mobile browsing latency, and reduce wireless network consumption.

For future work, we are planning to improve our system in three directions. First, we will integrate the fragment identification technology into the MobiDNA system that allows the MobiDNA to automatically identify fragments in dynamic Web content. This will enable the MobiDNA to be independent of any specific fragment technology, hence allowing for a scalable use in the heterogeneous Web. Second, we consider integrating the transcoding technology to compress Web content quality, which is believed to be able to further reduce content delivery in a mobile computing environment. Third, with the satisfactory results in our experiments, we will incorporate the MobiDNA scheme into real Web browser applications on mobile computing devices, such as Pocket Internet Explorer on the Pocket PC device.

ACKNOWLEDGMENTS

The authors give thanks to Ming-Yu Wang, Simon Goumaz, and Yusuo Hu for their generous help and insightful suggestions. Special thanks are given to the anonymous referees who have provided very indicative and helpful comments.

REFERENCES

[1] K. Amiri, S. Park, R. Tewari, and S. Padmanabhan, "DBProxy: A Self-Managing Edge-of-Network Data Cache," *Proc. IEEE Int'l Conf. Data Eng. (ICDE),* Mar. 2003.

[2] T.W. Bickmore and B.N. Schilit, "Device-Independent Access to the World Wide Web," *Proc. Sixth World Wide Web Conf.,* Apr. 1997.

[3] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, "Accordion Summarization for End-Game Browsing on PDAs and Cellular Phones," *Proc. SIGCHI Conf. Human Factors in Computing Systems '01,* Apr. 2001.

[4] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, "Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices," *Proc. 10th World Wide Web Conf.,* May 2001.

[5] J.L. Chen, B.Y. Zhou, J. Shi, H.J. Zhang, and Q.F. Wu, "Function-Based Object Model towards Website Adaptation," *Proc. 10th World Wide Web Conf.,* May 2001.

[6] Y. Chen, W.Y. Ma, and H.J. Zhang, "Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices," *Proc. 12th Int'l World Wide Web Conf.,* May 2003.

[7] Edge Side Includes (ESI) Official Site, http://www.esi.org, 2004.

[8] A. Fox, S.D. Gribble, Y. Chawathe, and E.A. Brewer, "Adapting to Network and Client Variation Using Infrastructural Proxies: Lessons and Perspectives," *IEEE Personal Comm.,* vol. 5, no. 4, p. 10-19, 1998.

[9] X.D. Gu, J.L. Chen, W.Y. Ma, and G.L. Chen, "Visual Based Content Understanding towards Web Adaptation," *Proc. Second Int'l Conf. Adaptive Hypermedia and Adaptive Web Based Systems,* May 2002.

[10] M. Hori, G. Kondoh, K. Ono, S. Hirose, and S. Singhal, "Annotation-Based Web Content Transcoding," *Proc. Ninth World Wide Web Conf.,* May 2000.

[11] E.D. Lara, D.S. Wallach, and W. Zwaenepoel, "Puppeteer: Component-Based Adaptation for Mobile Computing," *Proc. Third Usenix Symp. Internet Technologies and Systems,* Mar. 2001.

[12] W.S. Li, W.P. Hsuing, D.V. Kalashnikov, R. Sion, O. Po, D. Agrawal, and K.S. Candan, "Issues and Evaluations of Caching Solutions for Web Application Acceleration," *Proc. 28th Int'l Conf. Very Large Data Bases (VLDB),* Aug. 2002.

[13] B. Noble, M. Satyanarayanan, and M. Price, "A Programming Interface for Application-Aware Adaptation in Mobile Computing," *Proc. Int'l Symp. Mobile and Location-Independent Computing '95,* Apr. 1995.

[14] M. Rabinovich, Z. Xiao, and F. Douglis, "Moving Edge-Side Includes to the Real Edge: The Clients," *Proc. Fourth USENIX Symp. Internet Technologies and Systems '03,* Mar. 2003.

[15] L. Ramaswamy, A. Iyengar, L. Liu, and F. Douglis, "Automatic Detection of Fragments in Dynamically Generated Web Pages," *Proc. 13th Int'l Conf. World Wide Web,* May 2004.

[16] N. Milic-Frayling and R. Sommerer, "SmartView: Flexible Viewing of Web Page Contents," *Proc. 11th Int'l Conf. World Wide Web,* May 2002.

[17] W.Y. Ma, I. Bedner, G. Chang, A. Kuchinsky, and H.J. Zhang, "A Framework for Adaptive Content Delivery in Heterogeneous Network Environments," *Proc. Multimedia Computing and Networking Conf. '00,* Jan. 2000.

[18] Shunra/Cloud Software, http://www.shunra.com/cloud.htm, 2004.

[19] G. Stuary, T. Rag, and K. Sreedhar, "ATTENUATOR: Towards Preserving Originally Appearance of Large Documents when Rendered on Small Screen," *Proc. Int'l Conf. Multimedia Expo '03,* July 2003.

[20] N.M. Su, Y. Sakane, M. Tsukamoto, and S. Nishio, "Rajicon: Remote PC GUI Operations via Constricted Mobile Interfaces," *Proc. Eighth Ann. Int'l Conf. Mobile Computing and Networking,* Sept. 2002.

[21] The Official Microsoft ASP.NET Site for IBuySpy Application, http://www.asp.net/Default.aspx?tabindex=5&tabid=42, 2005.

[22] K. Yagoub, D. Florescu, P. Valduriez, and V. Issarny, "Caching Strategies for Data-Intensive Web Sites," *Proc. 26th Int'l Conf. Very Large Data Bases (VLDB) '00,* Sept. 2000.

[23] S. Yu, D. Cai, J.R. Wen, and W.Y. Ma, "Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation," *Proc. 12th World Wide Web Conf.,* May 2003.

[24] C. Yuan, Y. Chen, and Z. Zhang, "Evaluation of Edge Caching/Offloading for Dynamic Content Delivery," *Proc. 12th Int'l World Wide Web Conf.,* May 2003.

[25] C. Yuan, Z. Hua, and Z. Zhang, "Proxy+: Simple Proxy Augmentation for Dynamic Content Processing," *Proc. Int'l Web Caching Workshop '03,* Sept. 2003.

Zhigang Hua received the BS degree in computer science from the Huazhong University of Science and Technology in 2002 and the MS degree from the Institute of Automation, Chinese Academy of Sciences, in 2006. He worked as a visiting student at Microsoft Research Asia from 2002 to 2005, where he focused on adaptive media for mobile devices, interfaces for Web search and navigation, and Web image mining. He is currently a graduate student in the College of Computing, Georgia Institute of Technology. His research interests include human computer interaction, mobile computing, and Web search.

Xing Xie received the BS and PhD degrees in computer science from the University of Science and Technology of China in 1996 and 2001, respectively. He is currently a researcher at the Web Search and Mining Group, Microsoft Research Asia (MSRA). He joined MSRA in July 2001, working on adaptive content delivery, mobile multimedia applications, and mobile Web search. He is a member of the IEEE and the ACM.

Hao Liu received the BS degree in electrical engineering from Central China Normal University in 2000 and the MS degree from the Institute of Electronics, Chinese Academy of Sciences, in 2003. He worked as a visiting student at Microsoft Research Asia from 2003 to 2005, where he focused on photo and Web image search and browsing on mobile devices. Currently, he is a graduate student in the Department of Computer Science, Stanford University. His research interests include new Web search and browsing facilities and mobile search applications. He is a student member of the IEEE.

Hanqing Lu received the BS degree in computer science and the MS degree in electrical engineering from the Harbin Institute of Technology in 1982 and 1985, respectively, and the PhD degree in electronic engineering from the Huazhong University of Science and Technology in 1992. He is currently a professor and vice director in the Institute of Automation, Chinese Academy of Sciences. He directs the Image and Video Analysis Research Group at the National Laboratory of Pattern Recognition. His research areas include image retrieval, mobile computing, face detection, and video coding. He has published over 50 international journals and conference papers.

Wei-Ying Ma received the BS degree in electrical engineering from the National Tsing Hua University in Taiwan in 1990 and the MS and PhD degrees in electrical and computer engineering from the University of California at Santa Barbara in 1994 and 1997, respectively. From 1997 to 2001, he was with HP Labs, where he worked in the field of multimedia adaptation and distributed media services infrastructure. He joined Microsoft Research Asia in 2001 and is currently a senior researcher and research manager at Microsoft Research Asia, where he has been leading a research group to conduct research in the areas of information retrieval, Web search, data mining, mobile browsing, and multimedia management. He currently serves as an editor for the *ACM/Springer Multimedia Systems Journal* and as associate editor for the *ACM Transactions on Information Systems* (TOIS). He has served on the organizing and program committees of many international conferences, including ACM Multimedia, ACM SIGIR, ACM CIKM, WWW, ICME, CVPR, SPIE Multimedia Storage and Archiving Systems, SPIE Multimedia Communication and Networking, etc. He is also the general cochair of the International Multimedia Modeling (MMM) Conference 2005 and the International Conference on Image and Video Retrieval (CIVR) 2005. He has published five book chapters and more than 100 international journal and conference papers. He is a senior member of the IEEE.