ADAPTING WEB CONTENT BASED ON USER CONNECTION SPEED TO ENSURE RESPONSE TIME

Reza Rostami[1] and Thiam Kian Chiew[2]

[1, 2] Faculty of Computer Science and Information Technology
University of Malaya, Kuala Lumpur, Malaysia

Email: [1]reza_lg13@yahoo.com; [2]tkchiew@um.edu.my

ABSTRACT

Response time to download a Web page is mainly determined by network bandwidth and the size of the page. With the notion of tolerable waiting time (TWT), this response time should be confined below a threshold value for user satisfaction. Nevertheless, users connect to the Internet with different connection speeds, causing variable response times in relation to different connection speeds. This paper presents a Web content adaptation and delivery technique based on user connection speed in order to download the page within a configurable time threshold. The technique not only ensures Web page response time meets the desired threshold value, but also preserves overall page quality by preventing Web page items to have inconsistent quality.

Keyword: Web Content Adaptation, User Connection Speed, Page Size

1.0 INTRODUCTION

Websites used to be simple collection of documents normally in the text format. As the use of the Internet gets more popular, the Web has evolved into various applications. People do almost everything such as playing games, shopping, paying bills and many other activities online. In parallel with this phenomenon, Web applications make use of servers that generate dynamic Web pages to serve different users with different needs. Generating dynamic pages effectively reduces tedious work of developing many static pages but needs more server side processing, and makes the Web developers to face tougher performance challenges.

On the other hand, Internet users are not normally patient enough to wait for a long time for a Web page to download and will easily switch to the competitor company's website, simply because of low download speed rather than the services that the website offers. Therefore, the performance (in terms of download speed known as response time) of a website is one of the key factors for customers to either hang on to or leave it [1].

In addition, users connect to the Internet with different bandwidths and connection speeds. A Web page can be downloaded in one area in less than three seconds while another user cannot see the same page within the same time because of the lack of bandwidth and low connection speed. All of these raise the need of servers that can adapt to different user connection speeds and deliver customized Web content based on the speeds in order to ensure desirable response time is achieved.

To overcome this problem some context-aware techniques have been developed [2, 3]. According to [2], a system that uses context to provide relevant services for the users, where the relevancy depends on user's task is a context-aware system. Content adaptation techniques are developed to effectively adapt the Web content and improve the performance of the websites. Examples of the techniques include constraining the page size under the situation of low bandwidth or slow user connection speed; recommending relevant contents to the users; etc.

The content adaptation technique developed by [4] is one that focuses on ensuring response time. For a Web page comprising of multiple Web content items (e.g. images), and each item has multiple candidate versions to be selected to compose the resulting page, the technique chooses the best possible version for each Web content item based on user connection speed. The selection algorithm does well, but some issues remained unsolved:

- When there are many Web content items in a Web page and there are many versions available for each Web content item, the computation becomes complicated using their technique.
- The earlier items on the page have the opportunity to have their best versions chosen while the later items are likely to get lower quality versions. This leads to having a page that includes some Web content items with very high quality and the rest of items with very low quality. This results in a page with overall quality that does not reflect the adaptation criteria, i.e. user connection speed.

In this paper, we developed a technique that adapts the Web content based on user connection speed, since the user connection speed is one of the key factors that affects the performance of websites. We used the idea of multiple versions for each Web content item suggested by [4], but developed a new selection algorithm to overcome the limitations of their algorithm. We also designed a folder structure and a reusable software component that can effectively handle the act of updating the information of different versions for Web content items. Since the focus of this research is on Web content adaptation technique rather than the detection of user connection speed, we have used the user connection speed detection technique developed by [5]. This technique is chosen because it is easy to be implemented using JavaScript and yet is effective enough for this research.

2.0 BACKGROUND

Context-awareness can be defined as *"presentation of information and services to a user, automatic execution of a service, and tagging of context to information for later retrieval."* [2]. The work is useful to recognize the scope of context-aware computing and select appropriate context and context-aware techniques to use. [3] developed a context-aware adaptation framework (FAÇADE) to fulfill the existing variable and incongruous environments. They used three different contexts for their adaptation methods, namely device, network, and user. [6] introduced additional contextual variables such as state of activities, roles, collaborators' location, and available resources in addition to other typical variables of working groups such as platform, to design and implement a framework that allows application developers to specify the adaptability of groupware systems. It can be noted that context-aware system are designed to serve users with different needs in different situations in order to achieve some specific goals. For Web-based systems, content adaptation is a typical approach to realize the systems' context-awareness.

Web content adaptation can be implemented through different techniques, taking into account different factors, and the adaptation can be done at the server side or by a third party agent. For example, [7] developed a server based content adaptation technique. Their method includes multiple versions of each multimedia Web content items, one of which will be chosen based on the client device capability by a customizer. [8] developed a framework for Web content adaptation based on the user device's capability. They offered designs of business objects, logical Web contents, and content adaptation separately. [9] developed a solution to the Web server overload problem that relies on adapting the delivered content. They used two versions (the full and degraded versions) for each item and a program that can switch between the two versions based on server state.

Meanwhile, [10] discussed some useful transformation for image contents of a website and the way it can affect the website performance. They have developed an HTTP protocol extension for server directed transcoding (SDT) and architecture for SDT applets and SDT proxies. [11] developed a decision engine to adapt Web content when the server receives a request from a mobile device by considering the user's preferences, the device's rendering capability and the network conditions. [12] offered a service-based content adaptation architecture that facilitates the use of third-party adaptation services and enables a novel content negotiation and adaptation model. Their technique allows users to select appropriate adaptation services through a negotiation with the side that provides the content.

In addition, [13] developed a content adaptation technique for wireless Web services. They used Document Object Model API to generate dynamic content based on pre-stored user preferences and user's device capabilities. In another research, [14] investigated content adaptation issues for mobile devices by considering the client context and the environment that receives requests. Adaptation is performed at the presentation layer, based on a critical point of view related to the semantics of content as a hierarchy of items.

A Web content adaptation and delivery mechanism based on application-level quality of service (QoS) policies was presented by [15]. Two kinds of application-level QoS policies were introduced: transmission time and transmission order of inline objects. They used two abstract terms namely *low* and *high* to prioritize images and this causes the adaptation to be limited with two situations only, a limitation that is similar to one faced by [9]. [16] proposed an e-learning system that can adapt its content based on each learner's environment. Their system first detects the environment conditions of each user, and then delivers the format and type of data for e-Learning based on the conditions. They considered user connection speed and user operating system as user environment.

The underlying motivation for content adaptation techniques mentioned above is to enhance overall user satisfaction which is achieved through delivering Web contents appropriate to users' tasks, platforms (e.g. operating systems), devices (e.g. mobile) as well as to reduce transmission time or response time. Response time is an important factor to determine the success of a website. As coined by [1], there is a *tolerable waiting time* beyond which the users may leave regardless quality of contents or services provided by the website. However, among the researches mentioned above, very few used response time as an explicit measure for the effectiveness of adaptation techniques proposed, except [4] and [15] while [9] addressed response time issue indirectly through reducing server load. These researches adapt Web contents to control the resulting page size in order to constrain response time below a threshold value.

The main drawback of [9] and [15] has been mentioned above. In this paper, we focus on the adaptation technique proposed by [4]. In [4], multiple versions of images are used and the best version for each image is chosen based on user connection speed. The different versions of a component item have different data sizes. According to their work, suppose a multimedia Web page, P, consists of a number of component items d_i where $P = \{d_1, d_2. . . d_n\}$. A component item d_i can be computed by transcoding into versions, d_{i1}, d_{i2}, …, d_{ij} with different resolutions. The problem to be solved is choosing the best version of each Web content item in a way that the sum of their sizes is maximized subject to the Web page size constraint. The Web page size constraint can be calculated by multiplying the user connection speed and a time threshold, within which the Web page is supposed to be loaded. They formulated this problem as a linear multi-choice knapsack problem (LMCKP). Then they mapped the LMCKP problem to a knapsack problem (KP) and finally offered a dynamic programming method to solve the knapsack problem. The selection of the best version for each item is a computation intensive mechanism. It is not efficient when there are many Web content items in a Web page and there are many versions available for each Web content item. In addition, as the Web content items must be sorted in descending order in terms of their size, the maintenance/updating of the database that stores data about the items and their versions is difficult and tedious. In addition, the mechanism favors earlier items over later items that are selected, resulting larger quality difference among the items. We thus propose a better Web content items selection algorithm to overcome the limitations.

3.0 ARCHITECTURE AND DESIGN

Figure 1 shows the architecture of our content adaptation technique. First, the user connection speed is detected. Then the Web page size constraint is calculated using the detected user connection speed. The Web page size constraint is the maximum size that the Web page can have so that it could be downloaded within a configurable time threshold. Once all Web content items that construct the Web page are identified, the best possible version for each Web content item could be selected by using the selection algorithm. The algorithm is encapsulated in a software component called Adapter. This process is done using the information about different versions of the Web content items, which is stored in a database. The content of the database needs to be initiated or updated in offline mode. This process is automatically done using another software component called Scanner, which scans a folder structure that is defined to accommodate different versions of the Web content items in the hard disk. Finally, the Web page can be rendered and delivered using the selected versions of the Web content items.

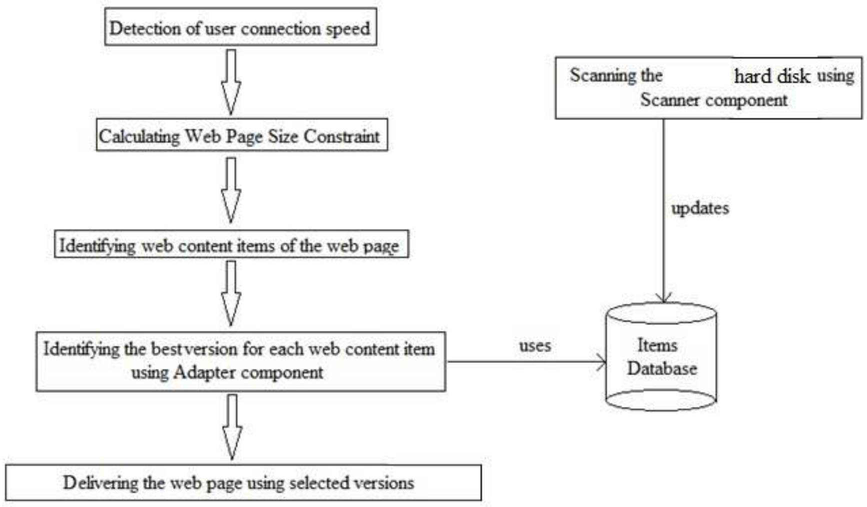
Detection of user connection speed
Calculating Web Page Size Constraint
Identifying web content items of the web page
Identifying the best version for each web content item using Adapter component
Delivering the web page using selected versions
Scanning the hard disk using Scanner component
updates
uses
Items Database

Fig. 1: System architecture

3.1 Folder Structure to Organize Web Content Items

In this research, a term called Web content item is used. A Web content item is any single item that constructs a Web page (e.g. image, video and animation). In other words, content of a page is composed of all Web content items of the page. In our adaptation method, multiple versions for each Web content item are considered. Imagine each page consists of n Web content items, w_1, w_2, w_3, …, w_n , where $n>0$.

For each w_i there are m versions, v_1, v_2, …, v_m , where $m>0$. For each Web content item, v_1 is the original version while v_2 is a degraded version of v_1 and in turn, v_i is a degraded version of v_{i-1}. This means for each Web content item:
Size of v_1> Size of v_2>…….> Size of v_m

The problem to be solved is to choose the best version of each Web content item in a way that the sum of their sizes is maximized subject to the Web page size constraint. The Web page size constraint can be calculated by multiplying the user connection speed and a time threshold, within which Web page is supposed to be downloaded. To manage Web content items, there is a need to store the information of Web content items and their different versions in a database. There are some issues arising through working with the mentioned database. First, to initialize the data stored in the database, the information of any single version of any Web content item should be entered separately and the version number of each version should be calculated based on its size.

For large websites, doing all these things manually needs a tedious effort and is very error prone. Furthermore, there is always a need to update the Web content item database in relation to updating the website. The action of updating is as difficult as initiating the database. These difficulties cause the need to develop a component that can be used in offline mode, in order to avoid runtime overhead that deteriorates website performance. The component automatically scans the hard disk in order to record the information of Web content items and their corresponding versions in the database.

To manage Web content items effectively, there is a need to design a folder structure and record the information, including folder for each item, the name of each version of that item (the name includes the type of file as well, e.g. image1.jpg), and the version number for each file in the database. The developed folder structure needs to be flexible enough to fulfill the system requirements.

To do so, a set of rules should be defined such that the address path of each Web content item should be the URI of a folder. The folder contains files that represent the Web content item's versions. For example, consider a Web content item that is an image called 'image1.jpg'. This item has a storage address and this address can be used to locate the image. If the image is planned to be used in a Web page in the normal way, the URI of the image can be like '/subcategory/image1.jpg'. Nevertheless, in our proposed folder structure, since an image has multiple versions, the Web content item (image1) will be stored in a folder with the URI of '/subcategory/image1', and this folder contains some files that are in fact different versions of image1 (e.g. version1.jpg, version2.jpg, and version3.jpg). Therefore, in any Web applications that use the proposed method, instead of recording the path to each Web content item and its versions, there is only a track to the folder (through its URI) that contains different versions of the item. Different versions of the item are identified based on their sizes in ascending order rather than their names. This ensures that the system is independent from file names and types.

Each folder should contain at least one version of a Web content item. It can alternatively contain any other number of versions. For Web content items that are not supposed to be delivered with a degraded version, there should only be one version in the corresponding folder. This allows the Web developer to specify important Web content items such as company logo that must be delivered with the highest quality. A database is needed to keep track of Web content items and their version' information. This information includes the URI of a folder for each item, the name of each version of that item, and the version number for each file.

3.2 Scanner Component and Detection of User Connection Speed

We developed a software component called Scanner that scans the hard disk, given a root folder with structure mentioned in the previous section, and initiates or updates the corresponding database that stores information about Web content items. This component automatically calculates the version numbers based on size (the best version assigned with version number 1, second best version with 2, and so on) and stores the version numbers in the database.

An existing user connection speed detection method proposed in [5] was used to detect user connection speed in this research. The method uses the capability of Java Script to detect user connection speed. A configurable time threshold for downloading Web pages was also defined. This time threshold indicates the duration within which Web pages are expected to be downloaded. We also defined Web page size constraint as the maximum size that a Web page can have in order to be downloaded within the time threshold. It can be calculated as below:

$$\boldsymbol{S_c = C_s * T_t} \qquad \textbf{(Equation 1)}$$

where S_c is the Web page size constraint; C_s is the user connection speed detected; and T_t is the time threshold defined. The Web page size constraint is later used by the adapter component to adapt the Web content accordingly.

3.3 Adapter Component

We developed a new Web content item selection algorithm and encapsulate it in the Adapter component. This algorithm first calculates the original size of the Web page, S_o as:

$$\mathbf{S_o} = \sum_{i=1}^{n} S_{wi1} \qquad \textbf{(Equation 2)}$$

where S_{wi1} is size of the best version of Web item i and n is the number of Web content items that comprise the page. If $S_o \leq S_c$, no content adaptation is required. Otherwise the degraded rate, D_r will be calculated as:

$$\boldsymbol{D_r} = \frac{S_c}{S_o} \qquad \textbf{(Equation 3)}$$

Then the maximum size for each Web content item i, S_{wimax} is calculated as shown in **Error! Reference source not found.**4:

$$\boldsymbol{S_{wimax} = S_{wi1} * D_r} \qquad \textbf{(Equation 4)}$$

The version with its size equals or less than S_{wimax} will then be chosen. Two situations should be considered here, namely iterative improvement, and selection of the best case in the worst scenario

3.4 Iterative Improvement

The size of the selected version for a Web page item *i* is normally less than S_{wimax} and thus some capacity will be wasted. The wasted capacity for Web content item i, S_{ei} is calculated as shown in Equation 5:

$$S_{ei} = S_{wimax} - S_{wisel} \qquad \textbf{(Equation 5)}$$

where S_{wisel} is the size of selected version for Web content item *i*. The sum of wasted capacity for all Web content items, S_{eall} can then be defined as:

$$S_{eall} = \sum_{i=1}^{n} S_{ei}$$

S_{eall} might be a considerable amount that can be used to improve the quality of some selected versions. For example, consider a page that has 3 Web content items namely Item 1, Item 2, and Item 3, and consider each item has some versions as follows:

Item 1: v1 (20kB), v2 (16kB), v3 (13kB), v4 (10kB)
Item 2: v1 (14kB), v2 (11kB), v3 (8kB)
Item 3: v1 (23kB), v2 (21kB), v3 (19kB), v4 (17kB), v5 (15kB)

Assuming the connection speed C_s is 6kB/s and time threshold T_t is 7s, page size constraint S_c will be 42kB and degraded rate D_r will be 0.735. S_{wimax} for Items 1, 2, and 3 are 14.7kB, 10.29kB, and 16.09kB; while the selected versions are v3, v3, and v5, respectively. The total wasted capacity S_{eall} is 5.08kB.

With this wasted capacity, Item 1 can be checked for possible quality improvement. v3 with the size of 13kB is chosen for Item 1 and the next better version (version with bigger size) is v2 with the size of 16kB. Since the size difference between the two versions (3kB) is less than 5.08kB, the better version v2 will be selected for Item 1. Total wasted capacity S_{eall} will thus be reduced too, i.e. 5.08kB – 3kB= 2.08kB.

With the new S_{eall} of 2.08kB, Item 2 is then checked for possible quality improvement. As v3 with the size of 8kB is chosen for Item 2 and the next better version is v2 with the size of 11kB, and the size difference between the two versions (3kB) is more than 2.08kB, v3 is remained selected for Item 2.

Finally, Item 3 is checked for possible quality improvement. As v5 with the size of 15kB is chosen for Item 3 and the next better version is v4 with the size of 17kB, and the size difference between the two versions (2kB) is less than 2.08kB, v4 will be selected instead while the new S_{eall} will now become 0.08kB.

This process will be repeated for all Web content items until none of the selected versions can be further improved. Doing so resulted in the selected versions of v2 (16kB) for Item 1, v3 (8kB) for Item 2, and v4 (17kB) for Item 3. Thus, the resulting Web page size is 41kB, which is less than but very close to 42kB (Web page size constraint, S_c).

3.5 Selection of the Best Case in the Worst Scenario

If S_{wimax} of a Web content item is less than the size of its last version (the version with the smallest size), the last version must be chosen. In this case, S_{ei} will have a negative value and eventually, there is a potential for S_{eall} to be a negative value as well. This means that the sum of selected versions exceeds the Web page size constraint, S_c. In this case, instead of improving the chosen versions, they should be iteratively degraded one by one until the total wasted capacity S_{eall} becomes a non-negative number or as close to zero as possible.

4.0 EVALUATION

We evaluated our system in two parts:

- Evaluating the effectiveness of our technique in real world situations.
- Evaluating the efficiency of our selection Algorithm against the algorithm developed by [4].

4.1 Evaluating the Effectiveness of Our Technique in Real World Situations

We tested and evaluated our method by taking the following steps. First we created a folder called 'images' that contained all versions of Web content items used in a Web application, in the way explained earlier in this paper. Then a desktop application was developed to use the Scanner component to scan the folder, and initiate the corresponding database. The Web application used the Adapter component to adapt the Web content, based on user connection speed detected using the technique developed by [5]. This Web application was also equipped with the ability of recording response time to download each Web page. The response time will be submitted to the server and is used to evaluate the system. Furthermore, another Web application that used the original Web content items without adaptation but would submit response time to the server was developed to compare its performance with the one that supported our adaptation technique.

We uploaded both Web applications to the same server and asked 20 users from 5 different cities to use the applications in real environment, and submit the response time to the server. We set the time threshold to load the Web page to 10 seconds and asked the users to browse the homepage of both applications at the same time and submit the performance data to the server. This ensures that the users were connected to both applications with the same, or at least similar, user connection speed. The submitted data were stored in a database for analysis and comparison. The size of the original homepage is 1267kB. Table 1 shows the data that were submitted to the server by the 20 users. The first column shows user connection speed. The second column is Web page size constraint, S_c. The third column is the size of the adapted Web page which is the sum selected versions for all Web content items. The fourth column is the time to load the adapted Web page. Finally, the last column is the time to load the original Web page.

Table 1: Data collected from 20 users

User connection speed (kB/s)	**Web page size constraint (kB)**	**Size of adapted Web page (kB)**	**Time to load adapted Web page (s)**	**Time to load original Web page (s)**
37	370	367	10.409	33.432
38	380	380	10.996	34.713
39	390	389	9.907	31.988
43	430	429	10.889	29.092
58	580	580	10.343	23.039
44	440	434	9.678	29.909
45	450	443	9.990	27.055
52	520	512	10.345	26.003
59	590	589	10.458	22.136
40	400	399	10.512	30.380
39	390	386	10.056	34.403
41	410	409	10.404	30.043
49	490	477	9.951	27.103
46	460	458	10.898	27.301
120	1200	1199	10.777	11.030
125	1250	1244	10.878	10.908
98	980	968	9.309	13.432

123	1230	1208	9.012	10.870
89	890	879	10.173	14.942
98	980	968	9.680	13.044

Figure 2 graphically compares the behavior of the homepage of the adapted and original websites for the 20 users. To understand whether or not the homepage of the adapted website was loaded within the expected time (10 seconds), we need to compare the blue line (expected time) and the red line (time taken to download the adapted homepage). It can be seen that the red line and the blue line are very close to each other for all the users. The response times to download the adapted homepage were between 9 to 11 seconds. It is not always equals to 10 seconds, however, the 1 second difference (+/- 10%) is still acceptable. Furthermore, for 10 out of the 20 users (50%), the difference between the expected and actual response time was less than 0.5 seconds (+/- 5%). This means that the adapted homepage was loaded within the expected time range and we have successfully achieved our goal for controlling Web page response time.

Comparing the green line (time to load the original homepage) and the red line, we can see for all users the adapted homepage was loaded in shorter time than the original homepage. The time to load the original homepage was depended on the user connection speed (higher user connection speed led to shorter response time) and exceeded the expected time. For example, for the user numbered 15, 16 and 18, time to load the original homepage was very close to time to load the adapted homepage because the user connection speed was relatively high in these cases. On the contrary, the time to load the adapted homepage was almost equal to the expected time all time and was independent from the user connection speed.

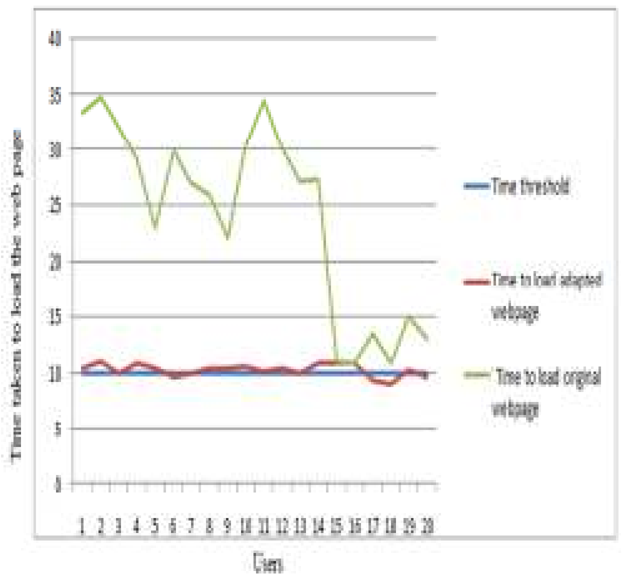

Fig. 2: Performance chart for downloading the original and adapted Web pages

4.2 Evaluating the Efficiency of Our Selection Algorithm Against the one Developed by [4]

In the algorithm developed by [4], the earlier items that were evaluated by the algorithm have the higher opportunity to have their best versions chosen while the later items are likely to get lower quality versions. This leads to having a page that includes some Web content items with very high quality and the rest of items with very low quality. This is apparently an undesired result as the overall page quality does not appropriately reflect the user connection speed and the page is segregated into two or even more sections in terms of Web content items' quality. In contrast, with our selection algorithm, the version that better reflects the user connection speed is chosen initially based on the degraded rate for every item. Then in terms of version improvement, our method uses in-stage improvement iteratively causing smaller quality difference between the selected versions for all Web content items.

In relation to maximization of Web page size, neither algorithm is superior to the other one. It really depends on the distribution of sizes and the number of versions for each Web content item. Here are three examples that show three different situations.

Example 1: Web page size constraint, S_c = 120kB. The versions of Web content items are depicted in Table 2. Versions selected by both algorithms are shown in the table as well.

Table 2: Web content items and their versions for Example 1

	Ver.1 (kB)	Ver. 2 (kB)	Ver. 3 (kB)	Selected by [4]'s algorithm	Selected by our algorithm
Item1	90	40	10	Ver. 1 = 90kB	Ver. 2 = 40kB
Item2	90	30	10	Ver. 3 = 10kB	Ver. 2 = 30kB
Item3	90	30	10	Ver. 3 = 10kB	Ver. 2 = 30kB
Total	-	-	-	110kB	100kB

[4]'s algorithm outperforms ours in relation to page size maximization subject to the page size constraint but their algorithm clearly suffers from the selected Web content items having larger quality difference; where the best version is chosen for Item 1 while the worst versions are chosen for the rest. With our algorithm, the medium versions are chosen for all the three items.

Example 2: Web page size constraint, S_c = 120kB. The versions of Web content items and the selected versions are shown in Table 3.

Table 3: Web content items and their versions for Example 2

	Ver.1 (kB)	Ver. 2 (kB)	Ver. 3 (kB)	Selected by [4]'s algorithm	Selected by our algorithm
Item1	90	60	10	Ver. 1 = 90kB	Ver. 2 = 60kB
Item2	90	30	10	Ver. 3 = 10kB	Ver. 2 = 30kB
Item3	90	30	10	Ver. 3 = 10kB	Ver. 2 = 30kB
Total	-	-	-	110kB	120kB

With only one out of nine versions having a different size as compared to Example 1, our algorithm outperforms [4]'s and shows closer quality among the selected versions of Web content items.

Example 3: Web page size constraint, S_c = 120kB. The versions of Web content items and the selected versions are shown in Table 4.

Table 4: Web content items and their versions for Example 3

	Ver.1 (kB)	Ver. 2 (kB)	Ver. 3 (kB)	Selected by [4]'s algorithm	Selected by our algorithm
Item1	90	50	10	Ver. 1 = 90kB	Ver. 2 = 50kB
Item2	90	30	10	Ver. 3 = 10kB	Ver. 2 = 30kB
Item3	90	30	10	Ver. 3 = 10kB	Ver. 2 = 30kB
Total	-	-	-	110kB	110kB

Again, with just one difference in the size of a version as compared to the previous examples, now both algorithms perform equally in terms of page size maximization subject to page size constraint.

5.0 CONTRIBUTIONS, LIMITATIONS AND FUTURE WORK

We have developed a technique for Web page content adaptation based on user connection speed. Different user connection speeds will cause a same page to be downloaded in different duration of time. Thus, in order to ensure the response time being controlled under a desired time threshold regardless of user connection speed, the page size should be adjusted accordingly.

To adapt Web content, we designed a selection algorithm that does not cause much overhead and can effectively be used when there are many Web content items in a Web page and there are many versions available for each Web content item. Furthermore, our algorithm can select the versions for Web content items in a way that better reflects the user connection speed. This is an improvement over [4]'s algorithm. We have also shown that our algorithm performs as good as [4]'s algorithm in terms of page size maximization. It is a matter of versions' sizes and distribution that will determine total size of the adapted page.

We have also designed a folder structure and a reusable software component that can effectively handle the act of updating the information of different versions for Web content items, with very minimal human intervention. This supports the automation of content adaptation to a higher degree.

Nevertheless, since we used one session variable to store the detected user connection speed, and the speed may change during a session, the user connection speed used for determining page size constraint may differ from the real user connection speed. It is, however, beyond the scope of our research to design a more effective technique to detect user connection speed dynamically and correctly. In addition, our proposed technique only takes the user connection speed into consideration to adapt the Web content, while sometimes, there is a situation where the server is overloaded and causes long download delay rather than slow user connection speed. User device capability is also one of the important factors that affects website performance as perceived by end users and this was not considered in our research as well.

To overcome the limitations, there is a need to consider more factors to adapt the Web content such as user device capabilities and server overload state. Doing so shall produce a better Web content adaptation technique that improves overall result of content adaptation. This is a necessary step leading to convincing researchers and practitioners in related fields to adopt our algorithm.

REFERENCES

[1] Nah, F. F.-H. " A Study on Tolerable Waiting Time: How Long are Web Users Willing to Wait?", in *Proceedings of American Conference on Information Systems (AMCIS)*, Tampa, 2003.

[2] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. "Towards a Better Understanding of Context and Context-Awareness", in *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC'99)*, Karlsruhe, 1999.

[3] Kurz, B., Popescu, I., & Gallacher, S. "FACADE - A Framework for Context-Aware Content Adaptation and Delivery", in *Proceedings of the 2nd Annual Conference on Communication Networks and Services Research*, Fredericton, 2004.

[4] Jan, R-H, Lin, C-P, & Chern, M-S. "An Optimization Model for Web Content Adaptation", *Computer Networks*, Vol. 50, No. 7, 2006, pp. 953-965.

[5] Skinner, J. JavaScript Speed Detection, 2005, Available: http://www.thefutureoftheweb.com/blog/javascript-speed-detection, Accessed: 15 February 2012.

[6] Decouchant, D., Mendoza, S., Sánchez, G., & Rodríguez, J. "Adapting Groupware Systems to Changes in the Collaborator's Context of Use", *Expert Systems with Applications*, Vol. 40, No. 11, 2013, pp. 4446-4462.

[7] Mohan, R., Smith, J.R., & Li, C-S. "Adapting Multimedia Internet Content for Universal Access", *IEEE Transactions on Multimedia*, Vol. 1, No. 1, 1999, pp. 104-114.

[8] Kitayama, F., Hitose, S., Kondoh, G., & Kuse, K. "Design of a Framework for Dynamic Content Adaptation to Web-Enabled Terminals and Enterprise Applications", in *Proceedings of Asia-Pacific Software Engineering Conference (APSEC'99)*, Takamatsu, 1999, pp. 72-79.

[9] Abdelzaher, T & Bhatti, N. "Web Content Adaptation to Improve Server Overload Behavior", *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol. 31, No. 11-16, 1999, pp. 1563-1577.

[10] Knutsson, B., Lu, H., & Mogul, J. "Architecture and Pragmatics of Server-Directed Transcoding", in *Proceedings of the 7th International Workshop on Web Content Caching and Distribution*, Boulder, 2002.

[11] Lum, W. Y. & Lau, F. C. M. "A Context-Aware Decision Engine for Content Adaptation", *IEEE Pervasive Computing*, Vol. 1, No. 3, 2002, pp. 41-49.

[12] Berhe, G., Brunie, L., & Pierson, J-M. "Modeling Service-Based Multimedia Content Adaptation in Pervasive Computing", in *Proceedings of the 1st Conference on Computing Frontiers*, Ischia, 2003, pp. 60-69.

[13] Pashtan, A., Kollipara, S., & Pearce, M. "Adapting Content for Wireless Web Services", *IEEE Internet* Computing, Vol. 7, No. 5, 2003, pp. 79-85.

[14] Lemlouma, T. & Layaida, N. "Context-Aware Adaptation for Mobile Devices", in *Proceedings of IEEE International Conference on Mobile Data Management*, Berkeley, 2004, pp. 106-113.

[15] Harumoto, K., Nakano, T., Fukumura, S., Shimojo, S., & Nishio, S. "Effective Web Browsing Through Content Delivery Adaptation", *ACM Transactions on Internet Technology*, Vol. 5, No. 4, 2005, pp. 571-600.

[16] Pravalpruk, B., Supnithi, T., Tummarattananont, P., & Mekpiroon, O. "Environment Model for Adaptive e-Learning", Digital Learning, 2006, Available: http://digitallearning.eletsonline.com/2006/11/environment-model-for-adaptive-e-learning/, Accessed: 16 February 2012.