(54) Title: A SYSTEM AND METHOD FOR ACCESSING WEB PAGES

(57) Abstract: A method and system for reducing the bandwidth required in transmitting web pages to a browser is presented. The method includes a browser requesting from a proxy a first web page having a first content. A web page interface loads and transmits this first web page to the proxy and the proxy stores a copy of the first web page in its local memory. If the browser requests a second web page that is similar in content to the first web page, the proxy determines the differences between the content of the first web page and the content of the second web page after the proxy receives the second web page from the web page interface. The proxy then transmits these differences to the browser and the browser incorporates these differences with the first web page to create the second web page.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.
A SYSTEM AND METHOD FOR ACCESSING WEB PAGES

FIELD OF THE INVENTION

The invention relates in general to accessing web pages and more specifically to a system and method used to reduce the bandwidth required in transmitting web pages to a browser.

BACKGROUND OF THE INVENTION

A user may request several web pages in sequence from a web browser. In such a case, typically the browser requests the first web page from a server and the server loads this first web page into memory from a persistent storage device. The server then compresses the first web page before sending it back to the browser for decompression and display. When the user selects a second web page, the browser usually discards the first web page from its local memory and requests the second web page from the server. The actions performed by the server and browser for the first web page are then repeated.

This method of accessing web pages occurs for each web page that the user selects to access, even if the web pages requested are similar. Therefore, the bandwidth between the server and browser may be higher than required when transmitting a web page that is
similar to the currently displayed web page. The present invention overcomes this waste of bandwidth.

SUMMARY OF THE INVENTION

The invention features a system and a method that reduces the amount of data sent over a network when a client computer accesses a web page that is similar to a previously accessed web page. A user utilizes a browser to request from a proxy a first web page having a first content. The first content includes a first web link that invokes a request for a second web page having a second content. The proxy sends the request for the first web page to a web page interface. The web page interface loads the first web page into memory and transmits the first web page back to the proxy. The proxy determines using a predetermined criteria if the first content is sufficiently similar to the second content of the second web page. If the proxy determines that the first content is sufficiently similar to the second content, the proxy modifies the first web link to point to a script routine. The proxy then stores the modified first content of the first web page in its local memory for future use and sends the modified first web page to the browser.

If the user then utilizes the browser to request the second web page, the browser invokes the script routine. The script routine first transmits the request for the second web page to the proxy. The proxy forwards this request to the web page interface and
the web page interface returns the second web page having the second content to the
proxy. The proxy scans the second web page for web links that point to similar web
pages and modifies these web links to point to the script routine. The proxy then stores
the content of the modified second web page in its local memory for future comparisons.

The proxy then obtains the differences between the first and second web pages and
transmits the differences to the browser. The browser then uses the transmitted
differences and the content of the currently displayed first web page to produce
substantially a copy of the content of the second web page.

DESCRIPTION OF THE DRAWINGS

The aspects of the invention presented above and many of the accompanying
advantages of the present invention will become better understood by referring to the
included drawings in which:

FIG. 1 is a block diagram of an embodiment of the system used to access two
similar web pages by transmitting the differences between the two web pages; and

FIGS. 2A and 2B are sections of a flow diagram illustrating an embodiment of the
steps for accessing a second web page that is similar to a first web page.
DESCRIPTION OF EMBODIMENTS

In brief overview and referring to Fig. 1, the network system, in one embodiment, includes a server computer 50 (or server) in communication with a client computer 10 (or client). A user wishing to access a first web page performs an action on the client 10 to request the first web page from the server 50. For example, the user may use a browser 20 to request the first web page. The server 50 loads the first web page into its memory from a persistent storage device 60 and subsequently transmits the first web page to the client 10. The client 10 then displays the first web page to the user.

If the user requests a second web page, the browser 20 of the client 10 similarly requests the second web page from the server 50. In one embodiment the server 50 then compares the first and second web pages and, if the second web page is similar to the first web page, determines the differences between the two web pages. If the differences between the first web page and the second web page satisfy a predetermined criteria, as described below, the server 50 compresses the differences between the two web pages and transmits only the differences to the browser 20. The browser 20 then displays the second web page on the client 10 by updating the first web page with the transmitted differences.
In more detail, the server 50 is in communication with a persistent storage device 60. The server 50 further includes a web page interface 40 in communication with a proxy 30. The proxy 30 is in communication with the browser 20 over a communication channel 15, and the browser 20 is accessed by the client 10.

In one embodiment, the client 10 uses the browser 20 to make a first request to the proxy 30 for a first web page over the communication channel 15. The proxy 30 then forwards the first request to the web page interface 40. The web page interface 40 loads the first web page into the web page interface’s 40 memory from the storage device 60 and provides the first web page to the proxy 30. In order for the proxy 30 to identify differences, the proxy 30 must receive the web page content in clear, that is, not encrypted. Also, in another embodiment (shown in phantom), the proxy 30' is located on another server 50' and is remotely located from the server 50 having the web page interface 40. The proxy 30' communicates with the server 50 over a second communication channel 45.

After the proxy 30 obtains the first web page, in one embodiment the proxy 30 modifies at least one reference in the first web page to another web page so that selecting the modified reference calls a script routine. The script routine is software that the proxy 30 embeds within the first web page. Then the proxy 30 stores a copy of the modified first web page in its local memory. In another embodiment, the proxy 30 stores the first
web page in its unmodified state. The proxy 30 then sends the modified first web page, which includes the embedded script routine, over the communication channel 15 to the client 10. The client 10 then displays the first web page.

The client 10 then poses a second request to the server 50 for a second web page. As in the first request, the web page interface 40 loads from the storage device 60 the second web page corresponding to the second request and sends this second web page to the proxy 30. The proxy 30 then determines the differences between the first web page and the second web page. If the two web pages satisfy a predetermined criteria, the proxy 30 compresses the differences and transmits the compressed differences to the client 10 over the communication channel 15. As described in more detail below, the client 10 decompresses the differences and displays a web page corresponding to the second web page by incorporating the differences between the first web page and the second web page into the previously displayed first web page.

Looking more closely at the steps performed by one embodiment and also referring to FIGS. 2A and 2B, the user selects (step 200) a first web page P1 that the user wants displayed on the client 10 using the browser 20. The client 10 sends (step 205) a request for the first web page P1 to the proxy 30 and the proxy 30 forwards (step 210) this request to the web page interface 40. The web page interface 40 loads (step 215) the
first web page P1 into its memory from the storage device 60. In another embodiment, the web page interface 40 creates (step 215) the first web page P1.

Once the first web page P1 is loaded, the web page interface 40 transmits (step 220) the first web page P1 back to the proxy 30. The proxy 30 then modifies (step 225) the first web page P1 to enable difference comparisons between the first web page P1 and similar web pages. In modifying the web page, the proxy 30 initially scans the first web page P1 and searches for web links or other calls to other web pages (referred to generally as web links) which, if selected, result in the first web page P1 being replaced by a second web page. For each of these web links, the proxy 30 determines if it is likely that the web page referred to by the web link is similar to the first web page P1 using a heuristic program. The heuristic program uses a predetermined criteria to determine whether two web pages are similar. Examples of the predetermined criteria include the compressibility of the two web pages, the page names of the two web pages, and a meta tag associated with the two web pages.

When using the compressibility criteria, the heuristic program computes the differences between the two web pages. If the size of the differences between the two web pages is substantially less than the size of the second web page, then the heuristic program determines that the two web pages are similar.
In another embodiment, the heuristic program uses the page names of the two web pages as the predetermined criteria. The heuristic program compares the pathname of the two web pages and considers similarly named web pages to be similar. Examples of similar pathnames of two web pages include web pages in the same directory or web pages generated by the same program executing on a web server (e.g., a servlet or Active Server Page (ASP)).

In yet another embodiment, the heuristic program uses a meta tag criteria as the predetermined criteria. Meta tags are a general mechanism for specifying attributes of web pages and are typically used by web browsers 20 and readers of HTML source code. For example, a meta tag can be added to a web page denoting whether a web page is cacheable. A programmer can add meta tags to web pages manually or to the scripts that generate the web pages. In this embodiment, the proxy 30 uses meta tags to denote a new attribute of web pages (e.g., the similarity between web pages). For example, a web page has an added tag “<META isSimilarToLast>” which denotes similarity to the previous web page. As another example, tags are added to sets of web pages, such as a tag “<Meta name=‘oneOfSet’ contents=‘ShoppingBasket’>”. This meta tag includes the keyword attribute ‘oneOfSet’ and the value ‘ShoppingBasket’ of the meta tag to describe the web page. By using meta tags to denote similarity, a programmer or web
page designer overrides the decision regarding similarity between two web pages normally made by the proxy 30.

In a further embodiment, the proxy 30 determines similar web pages by keeping a database of pairs of web pages found to be similar or different. For instance, a certain meta tag, such as `oneOfSet`, is included within the web pages to indicate to the heuristic program that the web pages are similar. In such a case, the proxy 30 maintains two databases 48, 49, both of which are initially empty. The first database 48 includes information on the previously loaded web pages that the proxy 30 loaded and the value of the specific meta tag included within the web pages that can indicate to the heuristic program that two web pages are similar (e.g., `ShoppingBasket`). The second database 49 contains information relating two or more web pages (e.g., similar / dissimilar). In the alternate embodiment described above, the remote proxy 30 determines similar web pages by keeping a first database 48' and a second database 49'.

Specifically, if the heuristic program determines that an initial web page A has the `oneOfSet` meta tag, then the proxy 30 maps the initial web page A to the value of the `oneOfSet` meta tag (e.g., initial web page A → ShoppingBasket). It should be noted that the value of the meta tag may be a null value. If the initial web page A has a web link to a reference web page B, the proxy 30 first consults the second database 49 to determine if the proxy 30 has previously deemed the initial web page A and the reference
web page B to be similar. If the second database 49 contains information indicating that
the initial web page A is similar to the reference web page B, then the proxy 30 modifies
the web link of the initial web page A referencing the reference web page B so that the
script routine is invoked when the browser 20 requests the reference web page B. If the
second database 49 contains information indicating that the initial web page A is
dissimilar to the reference web page B, then the proxy 30 does not consider the initial
web page A to be similar to the reference web page B. The proxy 30 does not modify
the web link of the initial web page A referencing the reference web page B so that the
script routine is not invoked when the browser 20 requests the reference web page B.

If the second database 49 contains no information on the reference web page B, the
proxy 30 consults the first database 48. If the first database 48 has no information on the
reference web page B, the proxy 30 makes no decision regarding similarity between the
reference web page B and the initial web page A based on the meta tag heuristic and/or
the database entries. Instead, the proxy 30 employs one of the other previously described
heuristics (e.g., compressibility and/or page names) to determine whether the initial web
page A is similar to the reference web page B.

If the first database 48 contains the same value of the meta tag for the reference
web page B that is associated with the initial web page A (e.g., ‘ShoppingBasket’) and
the values are not equal to null, then the proxy 30 deems the initial web page A similar to
the reference web page B. Two web pages having meta tag values that are equal to null are not considered similar by the proxy 30 in order to ensure that only specific meta tag values are considered equivalent. In another embodiment, the proxy 30 considers web pages similar when each web page has a meta tag value equal to null. The proxy 30 then modifies the web link of the initial web page A referencing the web page B so that the script routine is invoked when the browser 20 requests the reference web page B.

If the first database 48 contains different values of the meta tags associated with the initial web page A and the reference web page B, then the proxy 30 does not consider the initial web page A to be similar to the reference web page B. Therefore, the link of the initial web page A referencing the reference web page B is not modified. It should be noted that a modified initial web page A can have some modified web links to web pages and some unmodified web links to other web pages. Besides traditional databases, the proxy 30 can alternatively use memory data structures or files stored on a local disk to keep the first database 48 and the second database 49. Although the proxy 30 employs a first database 48 and a second database 49 to maintain the previously described information on the web pages, the proxy 30 can alternatively use a single database or multiple databases to store the meta tag information and the similarity information.

To increase efficiency, the heuristic program can be optimistic; that is, the heuristic program on the proxy 30 assumes that a web link results in a similar web page.
For example, if the heuristic program uses the page name criteria, the heuristic program can assume that any web pages within the same directory are similar. If the assumption made by the heuristic program is incorrect (i.e., the two web pages are not similar), the browser 20 still displays the correct second web page because the proxy 30 in this situation (incorrect non-similarity determination) transmits the second web page to the client 10.

During operation of a further embodiment, the heuristic program employs the page name criteria to guess whether the two web pages are similar. If the proxy 30 has previously guessed that the two web pages are similar using the page name criteria and then follows the web link to the second web page, the proxy 30 retrieves the second web page from the web page interface 40 and applies one or a combination of the other criteria (e.g., compressibility and/or meta tag criteria) to determine whether the proxy 30 should transmit the second web page or the differences between the two web pages. As described in more detail below and in a further embodiment, the proxy 30 updates the second database 49 when the proxy 30 makes its final decision on whether to transmit the second web page or the differences between the two web pages. To check that the heuristic was helpful in determining similarity, the proxy 30 can employ the compressibility criteria even if no meta tags exist in the web pages.
In contrast, if the proxy 30 follows a web link to a second web page and the proxy 30 has previously determined that the two web pages are not similar, then the proxy 30 retrieves the second web page from the web page interface 40 but does not compare the two web pages. The proxy 30 at this point can examine the second web page to determine if the second web page contains a meta tag indicating similarity with the first web page. If such a meta tag is found, the proxy 30 can store this information in the previously described first database 48 for future comparisons.

If the proxy 30 uses the heuristic program and determines that a web link refers to a web page similar to the first web page P1, the proxy 30 modifies (step 225) the first web page P1 so that the activation of that web link within the first web page P1 calls a script routine that executes on the browser 20. When the user clicks on the web link, the browser 20 invokes the script routine.

In one embodiment, the script routine is software written in JavaScript, a scripting language used to develop client-side Internet applications. It should be understood by those skilled in the art that the script routine can be written in any computer language so long as the browser 20 can interpret and execute the script. An example of a script routine is to replace a reference "<a href="foo">click here</a>" with "<span onClick="goGetIt(‘foo’)">click here</span>.” In this example, goGetIt() is a JavaScript function added to the first web page P1 which sends the second web page P2 request to
the proxy 30. The proxy 30 responds with either the second web page P2 or the
differences between the first web page P1 and the second web page P2. The JavaScript
function then performs additional processing, as described below, to recreate the second
web page P2 before the browser 20 displays it. Other references, such as form submits
(the button or method used by the user of the browser 20 to submit a form to the server
50) can be treated in the same way.

As a more specific example with a form submit, a Submit button (used for
searches, etc.), may call the script routine when the user invokes the function. An
element of a script routine for a Submit button is to replace the JavaScript line “<input
id=GoBtn type=submit value="Go">” with “<input id=GoBtn type=button
value="Go" onClick="goGetForm()">.” In this example, goGetForm() is a JavaScript
function provided in the script routine to call the proxy 30. Furthermore, if the software
code for activating other browser 20 functions (e.g., a Refresh button) is accessible to the
proxy 30, then the proxy 30 can modify these web page buttons as described above.

15

Because the proxy 30 will need the contents of the first web page P1 later, the
proxy 30 then stores (step 230) a copy of the modified first web page P1 in its local
memory. By storing the first web page P1 after modification, future comparisons of the
first web page P1 with a second web page are more accurate because the proxy 30 does
not deem the same modifications on both web pages as differences. Additionally, the
proxy 30 marks its copy of the first web page P1 to indicate to which client 10 the proxy
30 sent the first web page P1. This is necessary in a system with more than one client 10
requesting the same web page P1.

The proxy 30 then sends (step 235) the first web page P1 to the browser 20 over
the communication channel 15 and the browser 20 displays (step 240) the first web page
P1 on the client 10. In another embodiment the proxy 30 compresses the first web page
P1 prior to transmitting it to the browser 20. If the user then selects (step 245) a second
web page P2 from a web link that had been modified by the proxy 30, the selection
invokes the script routine (phantom 250) on the client 10. Once invoked, the script
routine transmits (step 255) the second request to the proxy 30. The second request for
the second web page P2 transmitted by the script routine is a different request than the
first request for the first web page P1. For example, a first request transmitted by the
browser 20 is “HTTP GET /some/page.” For the second request, the script routine uses a
special name (e.g., “special name”) to invoke a servlet or other software to calculate the
differences on the proxy 30. An example of a second request transmitted by the script
routine is “HTTP GET /special name? ’some/page’.”

The script routine also notifies the proxy 30 to compare the currently displayed
first web page P1, which the proxy 30 indexed, with the requested second web page P2
by including the “special name” in the second request. The script routine also notifies
the browser 20 to open a non-displayed window in which the differences between the first web page P1 and second web page P2 are stored. In this way, the displayed first web page P1 is left intact. The browser 20 can then recreate the second web page P2 from the transmitted differences stored in the non-displayed window and the displayed first web page P1.

The proxy 30 again forwards (step 260) the request (e.g., the second request for the second web page P2) to the web page interface 40. The web page interface 40 creates or loads (step 265) the second web page P2 and transmits (step 270) the second web page P2 back to the proxy 30. The proxy 30 next modifies (step 275) the web links in the second web page P2 to invoke the script routine (phantom 250) using the same heuristic program the proxy 30 used to modify the web links in the first web page P1. The proxy 30 then stores (step 280) the modified second web page P2 and deletes the previously stored web page. As previously described with respect to the first web page P1, by storing the second web page P2 after modification, future comparisons of the second web page P2 with another web page are more accurate because the proxy 30 does not deem the same modifications on both web pages as differences. In another embodiment, the proxy 30 modifies (step 275) the second web page P2 after storing (step 280) the second web page P2 in its local memory.
In one embodiment, the proxy 30 calculates the differences between the first web page P1 and the second web page P2 by treating the contents of both web pages as sequences of characters and comparing the two pages on a character by character basis.

In another embodiment, the proxy 30 considers the contents of the two web pages as trees of HTML elements. Examples of HTML elements are web links and characters. A few specific examples of HTML elements are "<TEXT background=red>hello world</TEXT> and <LIST>[child tags of type <LI>] </LIST>." When data is organized in a tree-like structure, each element in a tree is referred to as a node. A parent node is a node that has one or more children nodes. Nodes that have no children are called leaf nodes. In this embodiment, the proxy 30 compares the trees for common leaves and nodes to obtain the differences between the web pages.

The proxy 30 then compresses (step 285) the differences between the first web page P1 and the second web page P2 using compression software. The proxy 30 subsequently determines if the transmittal of the differences between the first web page P1 and the second web page P2 to the browser 20 is less wasteful of bandwidth than the transmittal of the second web page P2 itself. To help in this determination, the proxy 30 compresses the second web page P2. The proxy 30 then compares the size of the compressed differences to the size of the compressed second web page P2. If the proxy 30 concludes that the compressed differences are not smaller than the compressed second
web page P2, then the proxy 30 sends the compressed second web page P2 to the client 10.

As briefly discussed above, in another embodiment the proxy 30 updates the second database 49 when the proxy 30 makes a final decision as to whether the differences between the two web pages or the content of the second web page P2 is sent to the client 10. More specifically, the proxy 30 updates (step 285) the second database 49 each time a difference between the first web page P1 and the second web page P2 is calculated. If the heuristic program initially determines that the first web page P1 is similar to the second web page P2 and the proxy 30 transmits the differences between the first web page P1 and the second web page P2 to the client 10, the proxy 30 denotes in the second database 49 that the web pages are similar (e.g., first web page P1, second web page P2 -> similar). If the heuristic program initially determines that the first web page P1 is similar to the second web page P2 and then the proxy 30 determines that the two web pages are actually dissimilar (and therefore transmits the second web page P2 to the client 10 rather than the differences), then the proxy 30 denotes in the second database 49 that the web pages are dissimilar (e.g., first web page P1, second web page P2 -> dissimilar).

If the heuristic program initially determines that the first web page P1 is not similar to the second web page P2, then the proxy 30 does not compute the differences
between the first web page P1 and the second web page P2 and therefore does not update the second database 49. In another embodiment, the proxy 30 computes the differences between the first web page P1 and the second web page P2 to update the second database 49 and thereby improve future similarity determinations by the heuristic program.

Because the heuristic program uses the first and second database 49 to check the heuristic program’s determination of similarity between web pages, the heuristic program can be optimistic; that is, the heuristic program on the proxy 30 assumes that a web link results in a similar web page. However, the heuristic program still follows the similarity decisions in the second database 49 that is updated after assuming that a web link results in a similar web page.

Otherwise, the proxy 30 sends (step 295) the compressed differences between the two web pages to the client 10. The proxy 30 also discards (step 290) the stored copy of the first web page P1. In another embodiment, the proxy 30 sends the compressed differences between the two web pages to the client 10 if the proxy 30 concludes that the compressed differences are smaller than the compressed second web page P2 by a predetermined threshold, such as by a predetermined number of bytes. In another embodiment, the proxy 30 does not compress the second web page P2 and therefore does not compare the compressed differences to the compressed second web page P2.

Instead, the proxy 30 always transmits the compressed differences to the client 10.
While the proxy 30 is implementing step 260 through step 295, the script routine executing on the client 10 awaits a response from the server 50. Once the browser 20 receives the data from the proxy 30, the browser 20 decompresses the compressed data using decompression software. In the case where the browser 20 invoked the script routine and therefore the proxy 30 transmitted the differences, the browser 20 recreates (step 297) the second web page P2 by incorporating the differences between the first web page P1 and the second web page P2 into the previously displayed first web page P1.

In another embodiment, the first web page P1 is capable of modifying itself with an embedded modifying script function and so the content of the first web page P1 is capable of changing often. Therefore, because the content of the first web page P1 changes, the browser 20 stores an original copy of the first web page P1 to allow a comparison for the differences between the original first web page P1 and the second web pages. In an embodiment in which the proxy 30 does not compute the differences between this first web page P1 and the second web page P2, the browser 20 does not store an original copy of the first web page P1 because the proxy 30 transmits the second web page P2 to the browser 20.

In one embodiment described above in which the proxy 30 calculates the differences between the first web page P1 and the second web page P2, the proxy 30 treats the contents of both web pages as sequences of characters and compares the two
pages on a character by character basis. Therefore, the differences are sent in the form of
textual modifications. For example, the proxy 30 performs a Unix "diff" command to
obtain the differences between the two web pages. More specifically, the transmitted
differences instruct the browser 20 to insert XXXX at position Y and delete AAA
characters at position ZZZ, where X and A represent characters and Y and Z represent
positions on the first web page P1 or second web page P2. Upon receiving these
transmitted differences, the browser 20 performs these modifications on the previously
displayed first web page P1 to create a new second web page P2. The browser 20 uses a
standard JavaScript function "document.setInnerHTML ("<html source>")" to redisplay
the new second web page P2. The browser 20 then discards (step 298) the unneeded first
web page P1 from its local memory and displays (step 299) the new second web page P2
on the client 10.

In another previously described embodiment, the proxy 30 considers the contents
of the two web pages as trees of HTML elements. Therefore, the differences are sent in
the form of structured differences. The browser 20 modifies the displayed first web page
P1 without removing the displayed first web page P1 from the display of the client 10.

It will be appreciated that the embodiments described above are merely examples
of
the invention and that other embodiments incorporating variations therein are considered to fall within the scope of the invention. In view of the foregoing, what is claimed is:
Claims

1. A method for accessing web pages comprising the steps of:
   receiving a request for a first web page having a first content;
   receiving a request for a second web page having a second content; and
   transmitting differences between said first content of said first web page and said
   second content of said second web page to a browser.

2. The method of claim 1 further comprising the steps of:
   storing said first web page;
   modifying said first content of said first web page to create a modified first web
   page having a modified first content; and
   obtaining said differences between said first content and said second content.

3. The method of claim 2 wherein said first content of said first web page includes a
   first web link that invokes said request for said second web page.

4. The method of claim 2 further comprising determining with a first predetermined
   criteria if said first content is sufficiently similar to said second content.

5. The method of claim 4 wherein said modifying said first web page further
   comprises modifying said first content by changing said first web link to point to a
   script routine when said first predetermined criteria is satisfied.
6. The method of claim 5 wherein said script routine is a program that causes said
proxy to obtain said differences between said first content and said second content
when said script routine is executed.

7. The method of claim 4 wherein said first predetermined criteria is amount of
compressibility of said first web page and said second web page.

8. The method of claim 4 wherein said first predetermined criteria is similarity in
pathnames of said first web page and said second web page.

9. The method of claim 4 wherein said first predetermined criteria is similarity in a
tag associated with said first web page and said second web page.

10. The method of claim 4 wherein said first predetermined criteria is a database.

11. The method of claim 10 wherein said database further comprises information
denoting similarity between said first web page and said second web page.

12. The method of claim 2 wherein said obtaining said differences further comprises
comparing each character of said first content with each respective character of
said second content.

13. The method of claim 2 wherein said obtaining said differences further comprises
comparing an element of said first content with an element of said second content.

14. The method of claim 2 wherein said obtaining said differences further comprises
maintaining a database denoting similarity between said first web page and said second web page.

15. The method of claim 1 further comprising the step of compressing said differences between said first content and said second content.

16. The method of claim 15 further comprising the step of discarding said first web page after compressing said differences.

17. The method of claim 1 wherein said step of transmitting occurs when said differences are less than a second predetermined criteria.

18. The method of claim 1 further comprising the step of maintaining a record indicating that said first web page was transmitted to said browser.

19. A method for accessing web pages comprising the steps of:

displaying a first web page having a first content;

transmitting a request to obtain a second web page having a second content;

receiving differences in content between said first and second web pages in response to said request; and

incorporating said differences into said first content of said first web page to produce substantially a copy of said second content of said second web page.

20. The method of claim 19 further comprising communicating with a proxy to obtain
21. The method of claim 19 further comprising the step of decompressing said differences.

22. The method of claim 19 further comprising the step of storing said first web page until said copy of said second web page is produced.

23. A system for accessing web pages comprising:

a browser displaying a first web page having a first content and transmitting a request for a second web page having a second content; and

a proxy in communication with said browser to receive said request for said second web page and to transmit differences between said first content and said second content to said browser in response to said request.

24. The system of claim 23 further comprising a web page interface in communication with said proxy and storage, said web page interface accessing said first and second web pages from said storage and said web page interface transmitting said first and second web pages to said proxy.

25. The system of claim 23 further comprising a compressor compressing said differences between said first content and said second content.
26. The system of claim 23 wherein said browser comprises a decompressor decompressing said differences between said first content and said second content.

27. The system of claim 23 further comprising a comparator comparing each character of said first web page with each respective character of said second web page to obtain said differences.

28. The system of claim 23 further comprising a comparator comparing each element of said first content with each element of said second content to obtain said differences.

29. The system of claim 23 further comprising a storage storing said first web page for future comparisons with said second web page and storing an indicia with said first web page indicating to said browser that said proxy transmits said first web page.