Estimating and Eliminating Redundant Data Transfers Over the Web: A Fragment Based Approach

Christos Bouras Agisilaos Konidaris
[1]*Computer Engineering and Informatics Department, University of Patras, 26500 Patras, Greece.*
[2]*Research Academic Computer Technology Institute – CTI, Riga Feraiou 61, 26221 Patras, Greece.*
E-mail : {bouras, konidari}@cti.gr

Abstract

Redundant data transfers over the Web, can mainly be attributed to repeated transfers of unchanged data. Web caches and Web proxies are some of the solutions that have been proposed, to deal with the issue of redundant data transfers. In this paper we focus on the efficient estimation and reduction of redundant data transfers over the Web. We first prove that a vast amount of redundant data is transferred in Web pages that are considered to carry fresh data. We show this by following an approach based on Web page fragmentation and manipulation. Web pages are broken-down into fragments, based on specific criteria. We then deal with these fragments as independent constructors of the Web page and study their change patterns independently and in the context of the whole Web page. After the fragmentation process we propose solutions for dealing with redundant data transfers.

Keywords: Frequency of change estimation, Proxy servers, Web Fragments, Client Side Includes

1. Introduction

Redundant data transfers have mostly been related to the resource change estimation problem. Many technologies and techniques have been adopted, in order to reduce redundant data transfers. Client Caching, proxy servers, specific HTTP headers, are only some of them. Redundant data transfers are mainly attributed to re-transmissions of unchanged Web pages. Only when a proxy or a cache can efficiently estimate the change frequency of a Web page, can it accurately decide whether to request a new copy of a page or use a locally cached copy.

In this paper we look at the Web page change estimation problem in relation to Web data transfers. Let's assume that a Web client cache is able to accurately compute the change frequency of a Web page. This means that the client cache "knows" when to actually request a "fresh" copy of the page from a Web server and when to use a locally stored copy. We argue that even in this case, the client causes the transfer of redundant (or unchanged) data over the Web. This argument is based on the observation, that even when Web pages actually change, they do not change completely. In fact, as we will show in our study, most of the data in a page does not change, even when the page can be identified as "changed".

2. Related Work

A lot of related work can be found in the fields of Web page change estimation and the reduction of Web data transfers. The issue of statistical change estimation has been a popular problem [1]. The fragment approach that we present here is based on our previous work in [17]. The rate and frequency of change of Web resources has also been a popular problem with many interesting and significant studies such as [2,6,7]. The work in [4,6] analyses the change frequency estimation problem very efficiently and has been a very helpful basis for our approach. We must also refer to the View and Dynamic Web page materialization problem as a related field to ours [14,10,18], since our approach can have applications to dynamic page materialization policies. The work in [16] refers to Web proxy cache replacement policies and is related to our work since the experimental study in this paper is carried out under a client/proxy/Web server scheme. Finally the work in [9,8,15] analytically presents the Web latency problem that we also attempt to efficiently address in this paper through the reduction of redundant data transfers. Some very important work on fragments and their significance to Web performance has been presented in [11,12,13]. A fragment approach has also been proposed in the Edge Side Includes framework [19].

3. Methodology and approach

In this paper we base our results on HTML files collected from popular Greek and International Web sites. The sites that we selected, were portals and news sites. The reason

for this selection was that we intended to monitor sites with frequent changes on their pages. After collecting HTML files, we measured several change parameters in order to define the following:

- If a change had occurred in the Web page
- Where (topologically) the change occurred
- How changes were related in a Web page

In this paper we assume that the only material that we have at our disposal in order to identify and "predict" changes in a Web page is the HTML code of previous occurrences of the whole page. Our methodology has two steps:

- An experimental phase which is actually a content-gathering and manipulation step and
- A computation phase where content processing is carried out

4. Experimental phase

During the first step of our study we collected data. Through a Java Web client/HTML parser that we implemented, we issued repeated requests for the home pages of 10 popular portal and news sites around the world. The experiment lasted for 24 hours. The requests to the home pages of the sites were executed at a rate of 1 request per 3 minutes. At every request, our client requested the home page of the site under review and saved the returned HTML to a text file. After collecting all the HTML files we passed over to the second step of this phase. The second step included HTML parsing, analysis and manipulation. The specific analysis that we performed relied on a HTML fragmentation algorithm.

The fragmentation algorithm can be viewed as an HTML filter. This filter, fragmented every occurrence of a Web page, based on the <BODY>, </BODY>, <IMG>, <TABLE> and </TABLE> tags. The <TABLE> and </TABLE> tags were selected as fragment delimiters because they are the most popular data structuring tags used in "contemporary" HTML pages. The filter was able to break-down the HTML of the page, initially into two parts:

- The HTML outside the <BODY> tag and
- The HTML inside the <BODY> and </BODY> tags.

Then, the HTML inside the <BODY> and </BODY> tags was broken down recursively into all the tables that it was made up of. The HTML inside the <BODY> and </BODY> tags was also stripped of all the <IMG> tags and their contents. The results of this manipulation technique were:

- The whole HTML file
- The contents of the < BODY > and </ BODY > tags
- The contents of the < BODY > and </ BODY > tags, stripped of all <IMG> tags and their contents
- All the individual tables contained in the < BODY > and </ BODY > tags. These tables were called fragments

All nested tables were treated as shown in Figure 1:

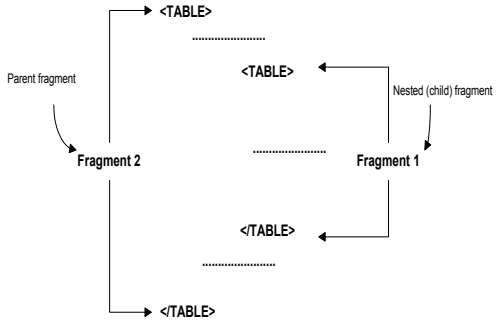

Figure 1. Example of fragment nesting identification

After the fragmentation of the HTML, an ascii conversion process was carried out for every fragment of every page. The ascii equivalents of every character in a fragment were added and stored in a new text file (one for every site), thus creating a numerical "signature" value for every fragment.

5. Computation phase

In this paragraph we present the computation phase of our methodology. The input of this phase are the fragment "signature" values that were the output of the previous phase. The text files will be used to extract results, related to the change probability and change pattern of every page. For space saving reasons we will use a specific Web site home page from now on in this paper. The selected home page was that found at europe.cnn.com. This page was selected because the results that we will extract from it are representative of all other Web site home pages that we worked on.

5.1. The fragment change probability and relative change probability matrixes

The fragment change probability matrix contains the change probabilities for all the identified fragments of a Web page. The relative change probability matrix contains the relative change probabilities for all the fragments. The change probability identifies the probability with which a fragment changes according to our experimental results. The notation $\mathbf{F_1}$, $\mathbf{F_2}$ … $\mathbf{F_n}$ is used to identify the fragments in the matrix.

The change probabilities were derived with the use of the following simple probability definition: The

probability of change for fragment $\mathbf{F_N}$ observed $\mathbf{N_{F_n}}$ times, during which $\mathbf{K_{F_n}}$ changes were observed is equal to:

$$P(F_n) = \frac{K_{F_n}}{N_{F_n}} \quad (1)$$

The relative change probability of the fragments, was computed with the use of equation 2, that provides the relative change probability of fragment F_B, when fragment F_A changes:

$$P(F_B/F_A) = \frac{P(F_A F_B)}{P(F_A)} \quad (2)$$

With the use of equation 1 and 2 we constructed the change probability and the relative change probability matrixes shown in Tables 2 and 3 respectively, for the home page of europe.cnn.com.

5.2. Page anatomy identification

The goal of this step was to identify the "anatomy" of a Web page. This step intended to analyse the Web pages in order to give us an idea of the fragments and their placement inside the pages. In this paragraph we, once more, use the home page of europe.cnn.com as an example. During this step we measured the following values:

- Number of fragments identified in the Web page
- Number of images
- Number of total fragments changing
- Percent of data outside the <BODY> and </BODY> tags

Table 1. The page anatomy parameters for the home page of europe.cnn.com

	Number of fragments	Number of images	Number of total fragments changing	Percent of data not in body tags
europe.cnn.com	29	82-85	17	~3%

The number of fragments remained the same during the whole experiment for europe.cnn.com. This is a clear indication of Web page structure stability. The same indication comes from the number of images that only vary from 82 to 85 during a whole day.

Out of 29 total fragments, changes were observed in 17. This number included fragment nestings. This means that a change to a nested fragment was also recorded as a change to the parent fragment. By taking a closer look to the structure of Figure 2 we intuitively expect that the fragments that actually change will be less than 17. We will prove this in following paragraphs. Finally, only 3% of the total data of the page was data that could be found outside the <BODY> and </BODY> tags. This indicates the page is data rich and that data outside the body tags can not cause significant redundant transfers even if it changes frequently (it was actually never observed to change).

5.3. The "reason of change" identification

This step aimed at identifying the "reason of change" of a Web page. We look at every request that was identified as "changed" from the previous request. The goal was to identify why the page was identified as "changed" (meaning what actually changed in the page). The delimited file that holds the results of the experimental phase also holds the fragment dependencies. The fragment dependencies represent table nestings and helped us construct the fragment tree for the Web page under consideration. The fragment tree for the europe.cnn.com site is shown in Figure 2.

After creating the fragment tree we, first eliminate all fragments that have not changed at all during our experimental phase. These are: F1, F2, F10, F6, F7, F14, F16, F24, F29. All fragments nested in fragments that never change (e.g. F28 in F29) are also eliminated. We then follow a "conservative" approach in the detection of totally dependent fragments. A totally dependent fragment is one whose changes can be safely attributed to a fragment nested in a lower level in the fragment tree. Our conservative approach states that a parent fragment can be eliminated as totally dependent on a nested fragment(s), only when, after removing the nested fragment(s), no HTML remains inside the parent fragment. In any other case we eliminate the nested fragment and keep the parent fragment, that is a data super-set in every case. For example in our study F20, F19 and F18 have shown 6 changes, in total, each. After a closer examination we found that after removing F18 and F19 HTML actually remained in F20. This means that a change in F20 could be attributed to:

- A change inside F18 or F19
- A change in the HTML outside F18 and F19 but inside F20

Thus, changes to F20 could not be explicitly attributed to either F19 or F18. In this case we eliminated F18 and F19 and kept F20 as their "representative" fragment. After this second step we ended up with the smallest possible sub-set of fragments that could change when a change was observed in the Web page. These fragments are: F3, F12, F9, F26, F25, F22, F21, F20 and F17. (They are less than 12 because of child fragment elimination, as described in this section)

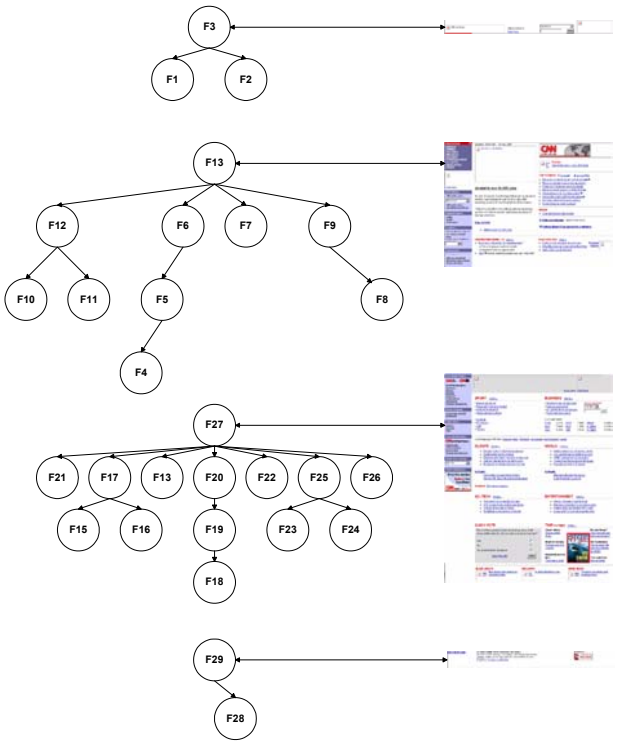

Figure 2. The fragment tree of the home page of europe.cnn.com and the corresponding visual representations. It is clear that the page is constructed with the *header, main body, footer* approach

Our goal here is to identify the redundant data transferred, as part of the unchanged fragments and the static HTML parts, inside the Web page, every time a change was observed. The following equation provides this percentage:

$$Data_{red} = Data_{all} - (Data_{fnc} + Data_{fcc}) \quad (3)$$

In equation 3 $Data_{red}$ is the percent of the redundant data transferred, $Data_{all}$ is all the data in the page (100%), $Data_{fnc}$ is the data contained in data fragments that never change and $Data_{fcc}$ is the data in fragments that have been observed to change but have not changed in the current change.

Equation 3 was used for every occurrence of a change in the page and the results are shown in Figure 3.

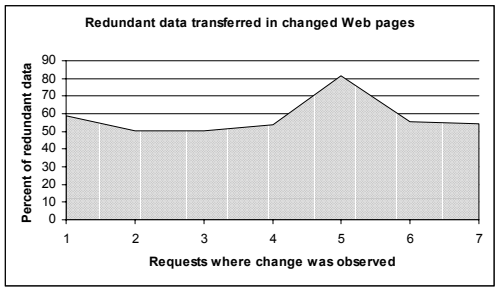

Figure 3. Redundant data transferred in Web pages that have changed

It is clear from Figure 3 that the data transferred inside a Web page of europe.cnn.com when the page actually changes, has not changed on average more than 43%. This means that the amount of redundant (or useless, or unchanged) data transferred together with the changed data inside a changed page, amounts to 57% of the total data on the page. This fact confirms our intuitive approach and provided us with enough motivation to find a solution that could prevent this phenomenon.

6. Proposed solution based on Client Side Includes

In this section we propose techniques that could help solve the problem of redundant data transfers over the Web. The basic cause of redundant transfers is the absolute binding of redundant data to useful (changed) data, in the context of Web pages. The problem can be solved if all portions of data (fragments) could, in some way, be treated (and requested) independently, and not under a unified page request scheme. The proposed solution is a straightforward one, and can be immediately put to work, since it utilises current Web techniques and practices. Our solution consists of two techniques and is based on the concept of Client Side Includes. The basic idea is that a client requests a page that is actually a page template. This template consists of many Client Side Includes that represent fragments. Each time the client requests the page a fragment change determination algorithm is executed. With the use of these techniques, the client will be able after a "slow-start" period, possibly caused by insufficient initialisation data, to accurately identify the Client Side Includes that should be requested because they have changed. The rest will be requested by local cache since they won't have changed. The basic problem of this approach is that all pages must be coded with the Client Side Includes format. This would require major changes to current Web page design techniques and is totally unacceptable. But this is true only for pages on Web servers. Let's consider the case of an intermediate server (e.g. a proxy server) that is able to fragment and re-build Web pages originating from different Web servers.

The clients would request pages from different Web servers through a proxy server. The proxy server has the ability to fragmentize each page (in the way that we have described in this paper). The fragmented page is then re-constructed with the use of Client Side Includes at the proxy server and sent to the requesting clients together with the change probability or the relative change probability matrixes. The whole procedure is transparent to the client. There are several approaches and client/proxy/web server architectures that could be followed under the scenario described here. Some of them are:

- The fragment change determination algorithms are executed on the proxy servers and the clients receive HTML files with Client Side Includes in the place of fragments that have been determined as "changed" by the proxies. This way the clients immediately request the included Client Side Includes upon the reception of the templated response from the Proxy servers
- The fragment determination algorithms are executed on the proxy servers and the proxy servers request the fragments that are determined as "changed". This way, the clients receive the final HTML document and do not have to further request any Client Side Includes.

The techniques that we will describe in the following paragraphs are alternative to the two approaches described above and can be implemented at the clients and at the proxy server. Both techniques are based on a matrix (change probability matrix or relative change probability matrix). These matrixes are initially constructed at the proxy server. If the techniques are implemented at the clients, the appropriate matrix must be transferred to the client before execution. Both techniques presented is this paper are implemented at the clients after the transfer of the matrixes from the proxy servers. This approach does not exploit the maximum potential of the Client Side Includes technique, since the proxy server itself must request the whole page from the originating server when needed with the use of the if-modified-since header. The actual gain is in the transfers between the proxy server and the clients. In the following paragraphs we will thoroughly explain and quantify the procedure.

In order to show the efficiency of our approach we will work on a real example. The site that we will use in the example will be europe.cnn.com. We will be using the home page of the site as sample data. The first problem that we will work on is the transformation of the home page HTML to an equivalent that uses Client Side Includes. In order to achieve this we recursively replace every table in the HTML with a Client Side Include. The procedure is clearly shown in Figure 4.

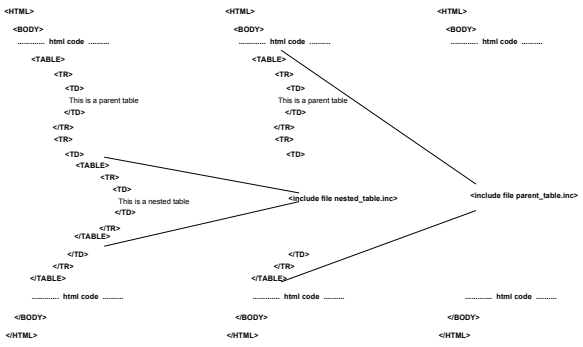

Figure 4. The recursive table replacement process

It is clear that the nested table structure will be replaced by a Client Side Include structure, but the content will remain unchanged. Having the new page structure in mind we present two techniques that can significantly improve data transfer over the Web.

6.1. Technique 1: Dependent on the change probability matrix

This first technique depends on the fragment change probability matrix. This matrix holds the change probabilities of every fragment and the standard deviation of every fragment change probability. The standard deviation is computed with the use of equation 4.

$$\sigma = \sqrt{Var[P(Fn)]} \quad (4)$$

Equation 4 computes the standard deviation of the change probability of fragment Fn as the square root of the variance of the change probability.

As shown in previous paragraphs, after the fragment elimination phase we have come-up with 9 fragments that are responsible for all changes observed at the europe.cnn.com site home page and can be considered independent. These are the fragments that we will use in the change probability matrix. This technique uses the change probability matrix to "discover" if a fragment has changed or not, at every client request.

The fragment probability matrix for europe.cnn.com is shown in Table 2.

Table 2. The fragment probability matrix for the home page of europe.cnn.com

	F3	**F9**	**F12**	**F17**	**F20**
Change Probability - P_{Fn}	0.033	0.033	0.019	0.024	0.029
Standard deviation - σ_{PFn}	0.013	0.013	0.085	0.012	0.012
$P_{Fn+}\ \sigma_{PFn}$	0.046	0.046	0.104	0.036	0.041

	F21	**F22**	**F25**	**F26**
Change Probability - P_{Fn}	0.019	0.014	0.005	0.005
Standard deviation- σ_{PFn}	0.012	0.090	0.006	0.009
$P_{Fn+}\ \sigma_{PFn}$	0.031	0.104	0.011	0.014

In this paper we will set the request threshold probability to 1 (we can also use another threshold value. This is part of our future work). Intuitively the request threshold probability is the flag that initiates a fresh fragment request. What we are actually stating when setting it to 1 is that a fragment must be requested when its change probability is at least equal to the request threshold probability. Thus:

$P(F_n) \geq P_{Thresh}$ in order for $P(F_n)$ to be requested. (5)

The fragment probability matrix holds the current values of the fragment change probabilities. Each time the page is requested, the fragment probability matrix is updated with a new value for all the fragments in the page. The update of the fragment probability matrix is executed according to the following equation:

$$P(F_n)new = (P(F_n)old+P(F_n)current+Var[P(F_n)])/2 \quad (6)$$

Equation 6 represents a totally conservative approach. The approach is conservative because fragment "freshness" has been valued higher than redundant data transfer reduction. What's the meaning of reducing redundant data transfers over the Web if clients keep getting stale data? Equation 6 initially adds the probability variance to the value of the fragment change probability. By doing this we maximize the fragment change probability. This, intuitively maximizes the times that a fragment will be computed as "changed" and thus, maximizes the times that it is requested. The current value of the fragment probability is added in order to capture the current "trend" of change for the fragment. It is obvious that according to the time of day, fragments change in different intervals. For example a stock quote changes frequently when the stock market is open but stops changing when it closes. Our technique adopts an initialization phase that computes fragment change probabilities over a big period of time. It is clear that the overall change probability of the stock quote would not provide accurate results for all times of a day. In order to avoid this as much as possible we averaged the value of the fragment change probability with the current change probability in order to be consistent.

The basic idea of this technique is to use the fragment change probability matrix to determine if a fragment has changed or not. Let's assume that clients are connected to a proxy server. The proxy server has passed the initialization phase and has already constructed a fragment change probability matrix for the europe.cnn.com home page. The clients start requesting the page. As a response, the clients receive a templated page consisting of client sided includes and the change probability matrix of the requested page. At the first request, the clients do not have the includes in their cache and must request them. After the first request, the clients receive the same templated Web page. Only this time, the clients use the change probability matrix to determine whether to request a fragment from the proxy or use the locally cached copies. This technique requires a substantial period of time in the initialization phase in order to provide consistent results. In our experiments we have used only 20 hours as the initialization phase. The redundancy and staleness graphs are shown in Figures 5 and 6 respectively.

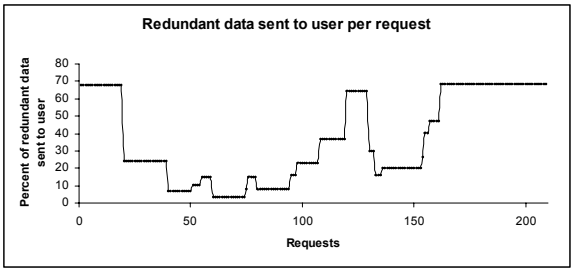

Figure 5. The amount of redundant data sent to clients

Technique 1 attempts to statistically map the results of the initialization phase to current requests. The initialization phase is used as the "history of change" and is constanly updated by current findings. In our experiments we used an initialization phase of 20 hours and tested it with 210 requests. The results showed that redundant data was still transferred to the clients. On the other hand the technique is much better than the transfer of a new copy of a Web page every time it is requested, that would cause the total (100%) transfer of fragments. We expect that this technique will show much better results under real conditions with an initialization period of more than a week, since the fragment change probabilities will be much closer to their true values.

As shown in Figure 6 certain requests initiate the transfer of stale data. The amount of stale data is small and confined to specific requests. We also expect the amount of stale data to drop for larger initialization phases.

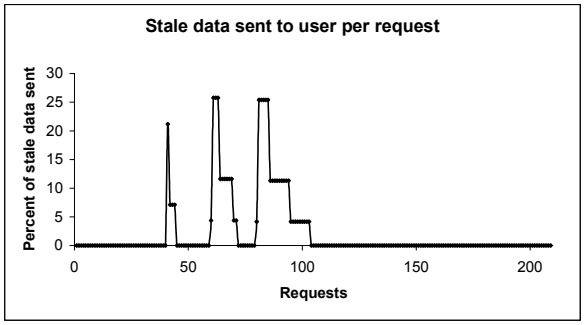

Figure 6. The amount of stale data sent to clients

6.2. Technique 2: Dependent on the fragment relative change matrix

This technique implements the same scenario as the previous, but decides to request a fragment or not, based on the fragment relative change matrix. The fragment relative change probability matrix for the home page of europe.cnn.com is shown in Table 3.

In this technique we assume that the if-modified-since header is available for every Web page. After requesting

the header, the client knows whether the whole page has changed or not. If it has not changed it uses the local copy. If the page has changed, the client uses the fragment relative change probability matrix to determine the fragments that have changed.

Table 3. The fragment relative change matrix for the home page of europe.cnn.com

Europe.cnn.com	F3	F9	F12	F17	F20
Whole page	1	1	0.57	0.71	0.86
When F3 changes	1	1	0.57	0.71	0.86
When F9 changes	1	1	0.57	0.71	0.86
When F12 changes	1	1	1	0.75	1
When F17 changes	1	1	0.60	1	0.80
When F20 changes	1	1	0.67	0.67	1
When F21 changes	1	1	0.67	0.67	1
When F22 changes	1	1	0.67	1	1
When F25 changes	1	1	0	1	1
When F26 changes	1	1	0	0	1

Europe.cnn.com	F21	F22	F25	F26
Whole page	0.86	0.43	0.14	0.14
When F3 changes	0.86	0.43	0.14	0.14
When F9 changes	0.86	0.43	0.14	0.14
When F12 changes	1	0.50	0	0
When F17 changes	0.80	0.60	0.20	0
When F20 changes	1	0.50	0.16	0.16
When F21 changes	1	0.50	0.16	0.16
When F22 changes	1	1	0.33	0
When F25 changes	1	1	1	0
When F26 changes	1	0	0	1

The fragments, whose relative change probabilities are equal to 1, when the whole page changes, are immediately requested. These are always F3 and F9. After these requests the client must decide which fragment to request next, since all other fragments may or may not have changed since their relative probabilities are less than one. Obviously the client must rank the rest of the fragments in a "smart" way. The ranking must be executed in a way that the best-ranked fragments can give a lot of information about the rest of the fragments. The ranking can be done in various ways. In this paper we examine two fragment-ranking algorithms:

The fragment probability and information based ranking. This ranking uses the fragment probability matrix described in technique 1 to rank the fragments and then, re-ranks the fragments according to the information that can be extracted by their relative change probabilities. The fragment with the greater probability value is ranked first and so on. Intuitively this can be justified by the fact that we want the next fragment that we request to have a large change probability in order for it to be useful. If the fragment has not changed we can not use its information from the fragment relative change probability matrix, and we must go on to the next ranked fragment. This ranking provides the following order for the rest of the fragments: F20, F17, F21, F12, F22, F25, F26. If we look closer at the fragment relative change probability matrix we will find that F20 and F21 do not provide any information for any other fragment except themselves. Thus, we eliminate them. Also, F17 provides information for only one fragment. We also eliminate it. Our final fragment ranking is F12, F22, F25, F26, F17, F20, F21.

The information based ranking. This ranking only considers the possible information given by a fragment change for other fragments through the relative change probability matrix. In other words in ranks the fragments according to the ones (they are valued higher) and zeros (0) they contain in their row in the fragment relative change probability matrix. According to this ranking the fragments are ranked as follows: F25, F22, F26, F12.

After ranking the fragments (other rankings may be applied) the client requests the fragment ranked first. After receiving it, it checks if it has actually changed. If it has, then its corresponding line in the fragment relative change probability matrix is used to extract information about other fragments. When this is over and a decision has not been reached on how to treat all fragments, the client requests the next ranked fragment. This goes on until a decision has been reached for all fragments in the fragment relative change probability matrix.

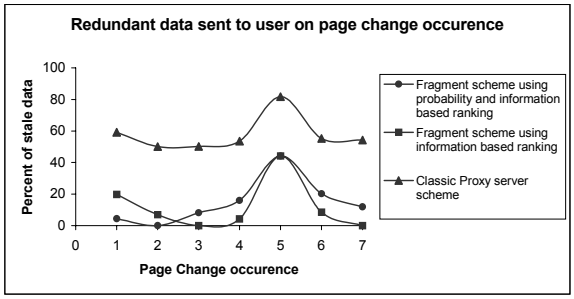

Figure 7. The redundant data sent to the user when the page changes

It is obvious that this technique also values "freshness" more than the reduction of redundant data transfers. This technique actually guaranties 100% fresh data and at the same time reduces Web data transfers. Since the if-modified-since header is provided, no data transfers are initiated when the page has not changed. The only interesting requests to look at, are those that lead to a new version of the page. In Figure 7 we show the percent of redundant data transfers, in cases that the page has changed. The reduction in redundant transfers is clear compared to the classic if-modified-since proxy server approach. It is also clear that information based ranking gives better results than the combined approach of probability and information based ranking.

7. Future work

Future work can be carried out in all aspects of the problem that we have presented in this paper. The HTTP protocol inherently treats Web pages as entities. We are already working on a scheme [20] that proposes minor changes to HTML and HTTP, in order to be able to apply our fragment framework directly to the client/server model with or without the intervention of a properly tuned proxy server.

8. Conclusions

In this paper we have proved that redundant data transfers are carried out on the Web, together with required data transfers, in the context of Web pages. Our research was mainly triggered by the observation that only some portions of Web pages change every time we request them. We have quantified the amount of redundant data transferred, through a specific example, and have showed that it is substantial. The solutions that we have proposed, have shown that proxy servers, Web servers and Web crawlers can benefit from treating pages, through a fragment approach. The techniques proposed in this paper show that by applying a purely statistical approach under a fragment scheme, a proxy server can efficiently deal with Web page changes and at the same time substantially reduce the amount of redundant data transferred over the Web.

References

[1] Howard M. Taylor and Samuel Karlin, "An Introduction To Stochastic Modeling", Academic press, 3rd edition, 1998

[2] Craig E. Wills and Mikhail Mikhailov, "Towards a better understanding of Web resources and server responses for improved caching" In Proceedings of the Eighth World-Wide Web Conference, 1999

[3] Brian E. Brewington and George Cybenko, "How Dynamic is the Web?", In Proceedings of the Ninth Worls-Wide Web Conference, 2000

[4] Junghoo Cho and Hector Garcia-Molina, "Synchronizing a database to improve freshness", In Proceedings of the 2000 ACM SIGMOD, 2000

[5] Fred Douglis, Anja Feldman, and Balachander Krishnamurthy "Rate of change and other metrics: a live study of the World Wide Web", In USENIX Symposium on Internetworking Technologies and Systems, 1999

[6] Junghoo Cho and Hector Garcia-Molina, "Estimating frequency of change", Technical report, Stanford Database Group, 2001-09-22

[7] Brian E. Brewington and George Cybenko, "Keeping Up with the Changing Web", IEEE Computer Magazine May 2000

[8] Md Ahsan Habib and Marc Abrams, "Analysis of Sources of Latency in Downloading Web Pages", WebNet 2000, October 30 - November 4, 2000 San Antonio, Texas, USA

[9] S. L. Tong and V. Bharghavan, "Alleviating the Latency and Bandwidth Problems in WWW browsing", Usenix Symposium on Internet Technologies and Systems '97, Monterey, CA. December 1997.

[10] Jim Challenger, Paul Dantzig, Daniel Dias, and Nathaniel Mills, "Engineering Highly Accessed Web Sites for Performance ", Web Engineering, Y. Deshpande and S. Murugesan editors, Springer-Verlag.

[11] J. Challenger, A. Iyengar and K. Witting., "A Publishing System for Efficiently Creating Dynamic Web Content", In INFOCOM 2000, March 26-30, 2000 Tel Aviv, Israel

[12] Jim Challenger, Arun Iyengar and Paul Dantzig, "A Scalable System for Consistently Caching Dynamic Web Data ", In Proceedings of IEEE INFOCOM'99, New York, New York, March 1999.

[13] Craig E. Wills and Mikhail Mikhailov, "Studying the impact of more complete server information on Web caching", 5th International Web caching and Content delivery Workshop, Lisbon, Portugal, 22-24 May 2000

[14] Alexandros Lambrinidis and Nick Roussopoulos "On the Materialization of WebViews", In proceedings of the ACM SIGMOD Workshop on the Web and Databases (WebDB '99), Philadelphia, Pensylvania, USA, June 1999

[15] Themistoklis Palpanas and Balachander Krishnamurthy, "Reducing Retrieval Latencies in the Web: the Past, the Present, and the Future", Graduate Department of Computer Science, University of Toronto, Technical Report CSRG-378

[16] Martin Arlitt, Richard Friedrich and Tai Jin, "Performance Evaluation of Web Proxy Cache Replacement Policies" HP Technical Report HPL-98-97R1, 991122, External, Available at http://www.hpl.hp.com/techreports/98/HPL-98-97R1.pdf

[17] C. Bouras and A. Konidaris, "Web Components: A Concept for Improving Personalization and Reducing User Perceived Latency on the World Wide Web", The 2nd International Conference on Internet Computing (IC2001), Las Vegas, Nevada, USA, June 25th - 28th 2001

[18] C. Bouras and A. Konidaris, "Run-time Management Policies for Data Intensive Web sites" International Workshop on Web Dynamics (In conjunction with the 8th International Conference on Database Theory), London, UK, 3 January 2001, pp. 1-10

[19] ESI technical specification found at http://www.esi.org/language_spec_1-0.html

[20] C. Bouras and A. Konidaris, "Improving Web Performance and Web Resource Mining through the Dynamic Resource Identification Protocol", Work in Progress.