Dynamics and Cachability of Web Sites: Implications for Inverted Capacity Networks

Sebastian Zander[1] , Grenville Armitage, Clancy Malcolm
Center for Advanced Internet Architectures. Technical Report 030405B[2]
Swinburne University of Technology
Melbourne, Australia
zander@fokus.fraunhofer.de
{garmitage, cmalcolm}@swin.edu.au

***Abstract*- The traditional Internet access model involves low bandwidth last-mile circuits and high bandwidth backbones. Imagine that in the future the last-mile becomes a high bandwidth service. In such an inverted capacity network content caching in the access network becomes essential to avoid backbone congestion and improve user experience but on the other hand the high access bandwidth also offers opportunities for new caching mechanisms. We focus on the Web as the most important and well established content service. With respect to caching the question is how much of the web content is cachable and what is the dynamic behavior?**

In this paper we analyze the cachability and dynamic behavior of a number of web sites and the implications for an inverted capacity network. In contrast to previous work we use an active approach for collecting the measurement data to be able to analyze complete web sites instead of subsets accessed by a specific user group over a certain time period.

Keywords- Inverted capacity network, World Wide Web, WWW, cachability, dynamics.

I. INTRODUCTION

The traditional Internet access involves low bandwidth last-mile circuits aggregating into higher bandwidth metropolitan, regional, and international backbones. Consumer market last-mile access typically involves 56K dial-up, ISDN, cable modem, or ADSL technologies. Regional backbones are often measured in gigabits per second, and many international backbones have capacities in the hundreds of megabits per second. Imagine that in the future the last-mile becomes a high bandwidth service in the order of multi-megabits or even gigabits per second (for example, using emerging passive optical networking techniques for fibre to the home or business). The result could be an inverted capacity network. In such a network the core is the bottleneck in terms of bandwidth making caching mechanisms in the access network essential. However a highly increased bandwidth in the access network also provides opportunities for new caching concepts as any customer would have enough bandwidth and storage capacity to act as a possible content cache for other nearby customers. Every town library could run content caches for their neighborhoods, revitalizing their roles as 21st century information repositories. Being able to push the content close to the user the bandwidth usage in the backbone network is decreased because the users can access most content from their local cache. The reduction of user traffic injected into the backbone network could also help to reduce the burstiness of the backbone traffic which would decrease the risk of congestion. Local or neighborhood content caches would also exhibit significantly reduced latency and packet loss rates at the IP level, leading to improved HTTP over TCP performance and shorter download times.

Despite the recently growing popularity of peer-to-peer networks the World Wide Web (WWW) is still one of the most popular Internet services used and web traffic is a large fraction of the overall Internet traffic. Therefore this paper focuses on the Web as the most important and well established content service. The question is to what extend the existing web service could benefit from an inverted capacity network. From a different perspective one could also ask whether mechanisms such as caching and content pushing can help to avoid congestion in the core network. In this paper we analyze the cachability and dynamic behavior of existing web content. As a first step towards answering these questions the results of this analysis will show upper bounds for caching mechanisms. They will also show whether the existing mechanisms are efficient enough or must be improved for future networks.

Chapter II discusses related work. Chapter III describes the approach used for the data collection. The results and findings are presented in chapter IV. Chapter V concludes and outlines future work.

II. RELATED WORK

A couple of studies of web traffic have been performed especially in recent times. These studies analyze web access traces either from the perspective of web browsers, web proxies or web servers [1, 2, 3, 5, 7]. Active monitoring of web sites was used by [4, 6] to measure the changing of web content. Furthermore a number of trace-driven simulations have been performed [2].

[1] Visiting researcher from the Measurement and Network Technologies Group, Fraunhofer FOKUS, Berlin, Germany
[2] A revised version of this paper will be presented at the 11th IEEE International Conference on Networks (ICON 2003) in Sydney, Australia, September 2003

Similar to us [2] also analyses the cachability of web objects as seen by a web proxy. In this work the cachability analysis depends on the traffic generated by the users over the measured time period. In contrast to this work we have focused on investigating the cachability of whole Web sites and comparing the cachability with the real dynamic behavior of the web objects (e.g. the time interval between changes).

In this paper we take the approach of active monitoring of web sites where most recent research focuses on the analysis of traffic traces (passive measurement). Using an active approach has the big disadvantage of injecting a large amount of synthetic traffic into the network. It is however the only way to collect data on whole web sites whereas in contrast the analysis of a traffic trace only allows the investigation of the specific part of web sites accessed by a certain user group. This is desired by the existing papers because most are focused on optimizing cache behavior and strategies but it does not provide a complete picture of web sites. With our approach it is also possible to collect data over long time periods unbiased by user accesses. As shown in [1] the stability of the popularity of web pages may change completely within two months except for the very top 10-50 pages and we have seen no investigation on whether cachability or the dynamic behavior changes with the user access patterns.

III. DATA COLLECTION

The data collection is based on active web crawling as used by search engines. We use a modified version of a web spider [10] originally used for indexing documents for a search engine. The spider can be highly configured in terms of the URLs to investigate, the content types to investigate, the scope (e.g. follow links with a given path, site or any) and the visit interval. The set of investigated sites selected are the most popular for a specific user group as indicated by local web proxy logs. For our first measurements the spider is configured to only investigate objects with the same server name as the homepage. The spider accesses the Internet through a web proxy which helps to improve performance by limiting the requests which actually go out into the Internet and to compare the cachability based on the HTTP protocol definition [8] with the cachability as determined by a real cache implementation (see chapter IV).

For each server defined the spider scans the site starting from the entry point given (usually the index page). Each HTML document accessed is scanned for outgoing links which are inserted into an URL list. Each URL is composed of server, directory, file and parameter parts. Thus, from the viewpoint of the spider URLs differ even if they differ only by their parameter part. Therefore links to a dynamic script where many URLs point to with different parameters lead to a large number of different URLs. For each object the spider uses a hash function to generate a unique ID for the content, determines the cachability and whether the object has changed since the last visit. This information together with the visit timestamp, response status and other relevant information is inserted into a database [11]. From there the data can be accessed for later analysis. The overall architecture is shown in Fig. 1.

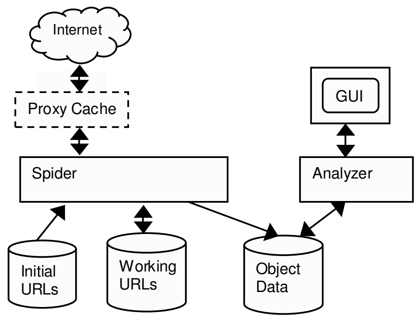

Fig. 1: Data Collection Architecture

The behavior of the spider upon arrival of an HTTP response is determined by the response status code and network errors as encountered at the socket layer. The behavior is shown in Table 1. ID generation refers to the process of generating the ID over the response body (content). Dynamics update refers to the process of updating the per-visit information, whereas information update refers to the update of the per-URL information (such as content type, size and headers etc.). Since network errors are assumed to be of short duration such URLs are rescanned after a shorter time interval than the usual visit interval.

The spider uses object validation based on the Last-Modified header and the Etag header [8]. For all cachable objects all subsequent requests after the initial will result in 304 "Not Modified" responses which means the content has not changed and is not sent again. Even if an Expires or Cache-Control header indicates no changes over a certain time period the spider will revisit the particular object again after the configured visit interval. Currently the spider has the following shortfalls as compared to passive analysis:

- It can not make any POST requests.
- It does not send cookies (although it receives them).
- It can not handle HTTP authorization.

We believe that the above does not have a major influence on our analysis because we focus on information/news sites which do not involve a high level of user interaction.

Status Code	**ID generation**	**Dynamics update**	**Information update**
OK (200, 206)	yes	yes	yes
Redirect (300-303, 307)	yes	yes	yes
Not Modified (304)	no	only visit time, status	only specific headers [8]
Error (305,400-415, 501, 502, 505)	no	only visit time	delete existing headers but keep the rest
Network Error (500, 503, 504, network errors)	no	only visit time	keep existing

Table 1: Spider behavior based on response status

We have compared the use of CRC32 and MD-5 hash functions for generating the IDs. An MD-5 ID uses 32 bytes (ASCII encoded) whereas a CRC-32 ID uses only 4 bytes and the computation of the MD-5 ID takes substantially more time. We analyzed a dataset containing roughly 575,000 different URLs. With MD-5 we got 509,546 distinct IDs whereas with CRC32 we got only 509,508 distinct IDs. Obviously using CRC32 we are experiencing some collisions. Although the fraction is only 0.007% it may become larger for larger URL sets. Since the additional resources needed for MD-5 are not a problem (neither in terms of CPU performance for the spider nor in terms of space for the database) we have decided to use MD-5 IDs.

Our cachability analysis is based on the definitions of [8] and some knowledge about the implementation of the well known Squid proxy [9]. There are several reasons why a document is not cachable. Since more than one reason can apply for a certain URL we have ordered the reasons and report only the highest reason found for an URL as reason for non-cachability:

1. Method – The request method is not GET or HEAD (because we use an active approach this reason never applies as the spider uses only GET requests).
2. No Freshness – The object has no freshness information attached i.e. there is no Last-Modified, Expires or Cache-Control header present
3. Stale – The response is expired according the Expires and/or Cache-Control headers.
4. Cache-Control – The response is marked as not cachable by a Cache-Control header. We distinguish between Cache-Control public and private.
5. Pragma – The response is marked as not cachable with a Pragma "no-cache" header. Pragma should only be used for HTTP 1.0 responses.
6. Uncachable Response – certain responses are not cachable e.g. 302 responses are not cachable unless explicitly allowed by a Cache-Control header.
7. Cookie – The response includes a Set-Cookie or Set-Cookie2 header (Squid 2 does allow the caching but Squid 1 does not).
8. Dynamic URL – The URL contains a question mark "?" or "cgi-bin"

Furthermore we investigate whether an object can be validated if Last-Modified and/or Etag headers are present in the response. A future version of our software will be capable of indicating more than one reason per object and will also cover authorization headers. Responses containing authorization headers are usually not cachable except if explicitly allowed by a Cache-Control header.

IV. RESULTS

In this chapter we present the results of the measurements we have done. We have investigated six (three commercial and three university/government) popular web sites (as indicated by a local web proxy log) for a time period of 14 days. The total number of distinct URLs is approximately 500,000 while the overall content size is approximately 14.5 GB. The visit interval for all URLs was set to 24 hours. In case of a short term network error the URL was visited again in only 6 hours.

Fig. 2 shows the overall response status distribution. In this figure 2xx and 304 response status have been summarized as OK. Redirects are summarized under redirects. 500, 503 or 504 responses status as well as network errors on socket layer are shown as network errors. All responses with status code 401, 402, 403 or 407 are counted as other while the remaining response status codes are errors. The figure shows that over 95% of the URLs have an OK status while 2% are temporary network errors, 2% are errors and 1% other and redirects.

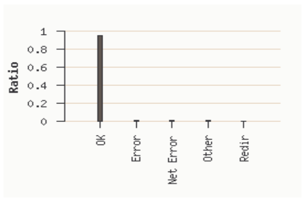

Fig. 2: Response status distribution

Fig. 3 shows the content type distribution both as object count and summarized content size. The content types are ordered by object count from left to right. Not surprisingly the most popular content types are HTML, GIF and JPEG objects. Interestingly these are followed by PDF documents which would actually be in second position if the data was ordered by summarized content length. Probably this would not be seen in passive trace analysis because users will not download PDF documents as frequently as HTML objects because of their size.

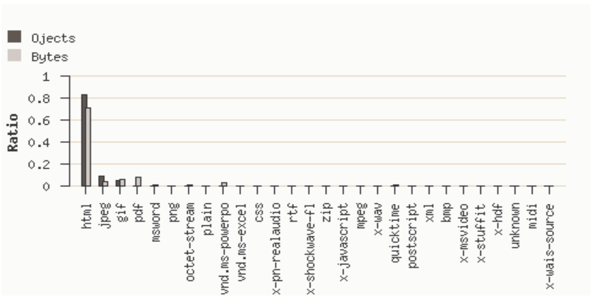

Fig. 3: Content type distribution

Table 2 shows the result of the cachability analysis. Although we have observed the cachability over a time period we here show only the cachability statistics from the latest visit. Although new objects have been created and some objects seem to have been disappeared the overall cachability ratios remained almost constant during our measurement period. Averaging the cachability over the measurement period provide the same result. As shown in the table a slight amount of GIF and JPEG is not cachable while almost all HTML documents are not cachable. All other content types are cachable.

Content Type	Cachable objects	Cachable bytes
application/pdf	100%	100%
image/gif	93%	89%
image/jpeg	95%	92%
text/html	3%	1%
Other	100%	100%
Overall	20%	32%

Table 2: Cachability by content type

Fig. 4 shows the main reasons for objects not being cachable. As mentioned before usually there is more than one reason. We show only the reason with the smallest number (highest priority) as explained in chapter III. The most common reason is either we know that the object is a dynamic script (92%) or the response contained no freshness information (7%) which is the case if the content has been dynamically generated or the server does not send the headers because of inability or wrong configuration.

Since newer web caches (e.g. Squid version 2) are able to cache URLs containing "cgi-bin" we further investigated the objects classified as uncachable because of dynamic URLs. First we found that 99.9% of the URLs containing "cgi-bin" also contain a "?" and URL parameters.

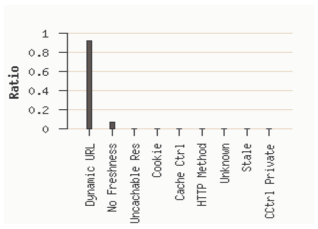

Fig. 4: Reasons for being uncachable

As stated in [8] these URLs should not been cached except explicitly allowed. We then investigated how many of these URLs explicitly allow caching (Expires, Cache-Control header) or at least contain implicit caching information (Last-Modified header). We found that:

- 0.033% of the URLs we have classified as uncachable above are explicitly allowed to be cached by a Cache-Control and/or Expires header,
- 0.007% of the URLs we have classified as uncachable have a Last-Modified header and
- 63% of the uncachable URLs have an Expires header which is set to the past to prevent caching (these URLs also have Cache-Control set to "no-store").

The conclusion is that 99.9% of the URLs are still uncachable because they lack any kind of freshness information.

We also investigated the content length of cachable objects and the content length of uncachable objects. As already suggested by the results given in Table 1 the uncachable objects are smaller having an average size of 26kB while the cachable objects are 40kB on average. As expected the tail of the distribution is much longer for cachable objects with a maximum object size of 61MB versus 1MB for the uncachable objects. Fig. 5 shows the cumulative density functions over a logarithmical x-axis. Almost all uncachable objects are between 2 and 63kB in size. Most cachable objects are smaller than 32kB. However a small fraction is very large.

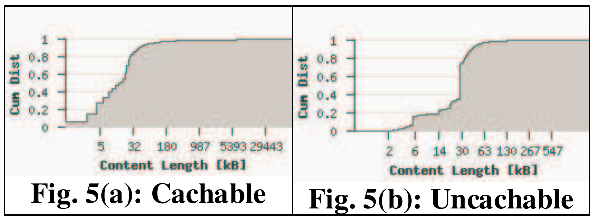

Fig. 5(a-b): Cachability and content length dependency for cachable and uncachable objects

Since all of the requests are going through a local proxy cache we can compare the theoretical amount of cachable objects with the amount actually cached by the proxy. This is interesting because assuming an optimal proxy with virtually unlimited storage space we would assume that all objects cachable in theory were cached by the proxy. However our measurements show that only a small fraction of the cachable content is actually served from the cache (see Table 3). The actual numbers vary over time but the overall percentage never exceeded 20%.

Content Type	Cachable objects cached	Cachable bytes cached
application/pdf	9%	8%
image/gif	36%	49%
image/jpeg	9%	19%
text/html	7%	7%
Other	0%-43%	0%-43%
Overall	12%	13%

Table 3: Cachability vs. real cache behavior

In contrast to the cachability of the objects we have also examined the real dynamics by detecting changes of the content using the MD-5 IDs generated for each object. Due to our active measurement approach we had to restrict the visit time interval to 24h. This means we were not able to detect changes within that 24h period. However in our investigation we found that the most of the objects have not changed at all and a significant amount of objects changed less than once per day on average. Of the uncachable objects however a large number change at least once in 24 hours. Probably a large amount of this content changes much more often as shown in previous work [1,2,5].

Table 4 shows the average rate of change observed over 14 days for the most popular content types and some other faster changing content types. It also shows the percentage of objects which have not changed at all during our measurement period. The change rate is defined as number of changes divided by the number of visits. Fig. 6(a-d) shows the cumulative density distributions of the number of changes and the minimum time between changes per object for the four most frequent content types (HTML, GIF, JPEG, PDF) over a logarithmic x-axis. We show the minimum time between changes instead of the average time because the minimum time represents the worst case.

Content Type	**Change rate objects**	**Change rate bytes**	**Unchanged objects**
application/pdf	1%	8%	96%
application/postscript	6%	7%	80%
application/vnd. ms-powerpoint	7%	23%	80%
image/gif	6%	11%	92%
image/jpeg	1%	1%	98%
image/png	4%	0.5%	96%
text/html	23%	28%	43%
video/x-msvideo	14%	15%	75%
Other	<1%	<1%	-
Overall	20%	22%	52%

Table 4: Average rate of change over 14 days

Comparing the dynamics with the cachability we find a big discrepancy between the cachability and dynamic behaviour of HTML documents. Only 20% are cachable but 43% have not changed at all within 14 days and of the documents that have changed 20-40% have a minimum change time interval larger than 24 hours. The percentage of PDF, GIF and JPEG objects that does not change at all is larger than 92%. For GIF images it can be seen that for almost all changing objects the change interval is one day or less (see Fig. 6(b)) while for PDF and JPEG the curve is more rounded (see Fig. 6(c,d)) meaning a larger percentage (20%, 40%) of the objects have minimum change intervals of two days or more. We believe the reason for the different behaviour of GIF and JPEG images is the use of the different image types. While GIF is used for a lot of images which are updated very frequently such as counters JPEG images are more used for photographs which are less frequently or never updated.

For some sites nearly all uncachable objects changed frequently. However for some other sites a large number of uncachable objects changed only very infrequently or not at all during our measurement period. Depending on user access patterns this may lead to a large amount of data which is unnecessarily transferred over the Internet. This situation could be improved by finding a clever scheme for generating validation information for dynamic content which would enable caches to at least be able to validate these objects.

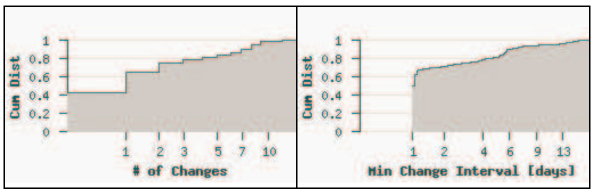

Fig. 6(a): Number of changes, minimum change time interval (HTML)

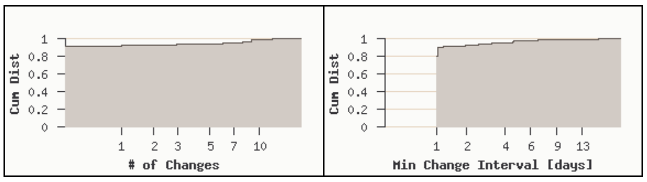

Fig. 6(b): Number of changes, minimum change time interval (GIF)

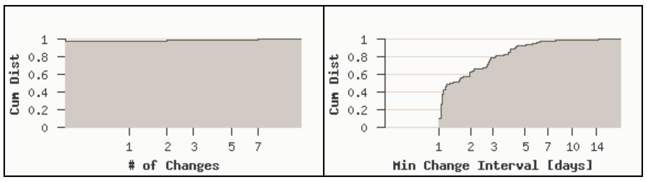

Fig. 6(c): Number of changes, minimum and change time interval (JPEG)

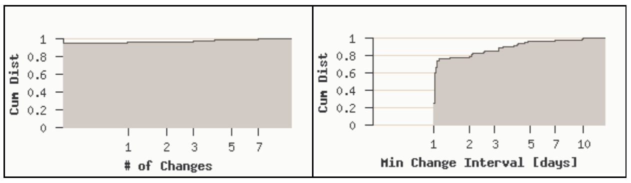

Fig. 6(d): Number of changes, minimum change time interval (PDF)

We also investigated whether there is a correlation between the object size and the minimum time interval between changes. Fig. 7 shows a scatter plot with the minimum change interval over the content size (for content sizes smaller then 150kB). The figure shows that small objects (<30kB) have a smaller minimum change interval (lower part on the left side of the dashed line) while for larger objects the minimum change interval is more equally distributed. It clearly shows that large objects do not have a larger minimum change interval.

Finally we have looked at duplication of objects. Duplication means objects have a different URL but the same content as indicated by the MD-5 hash. We found that 93% of the objects in our data set have no duplicates. 7% of the object content occurs at least twice under a different URL whereas in the extreme case one document was found having over 1,800 different URLs pointing to it. Actually this document is generated by a dynamic script which obviously generates the same content for a large number of different input parameter combinations.

Duplication is a potential problem if content is highly duplicated, different duplicates are accessed by a local user population and the content is substantially large. Existing caching could be improved by generating unique IDs such as a MD-5 hash for the objects. Although a MD-5 hash would experience collisions, combined with more information such as content type and size it could be sufficiently unique. Each object with a unique ID would need to be stored only once in the cache. Furthermore download times of large objects could be improved because a cache can supply a cached duplicate as soon as it would see the content type, size and unique ID in the header of the server response.

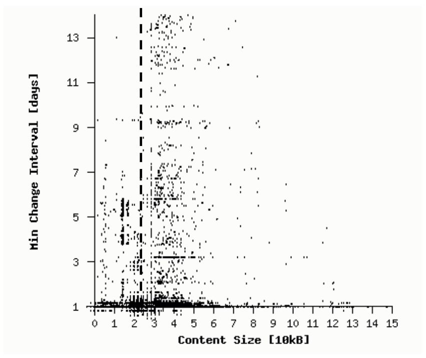

Fig. 7: Objects size and change correlation

With respect to a future inverted capacity network we find that the poor cachability of HTML documents might be problematic because a large amount of content can not be cached. We believe however that the cachability could be improved by new caching mechanisms. The possibility that each user has a large network bandwidth may enable new ways of web caching e.g. peer-to-peer distribution of web content.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have analyzed the cachability and dynamic behavior of a number of web sites and the implications on a capacity inverted network. Our study is done using an active measurement approach investigating complete web sites. The goal is to get a complete picture of those sites rather than seeing only a snapshot chosen by a specific user group. From our analysis we reach the following conclusions:

- Only 20%/32% of the investigated objects/bytes is cachable while the rest is not cachable by the current HTTP protocol. Most of the uncachable objects probably have been dynamically generated. The main reason for being not cachable are URLs containing either an "?" or "cgi-bin" and lack any freshness information. Uncachable objects are smaller on average and the tail of the distribution is much shorter compared to the distribution of cachable objects. The amount of cachable objects would probably be even smaller for a passive trace assuming that each user has a browser cache enabled.
- 52% of the objects were completely static and have not changed at all during our measurement period. Depending on the content type 10%-40% of the objects are dynamic but seem to be updated more infrequently in intervals at least 48 hours. Small objects (<30kB) tend to have smaller minimum change intervals while large objects do not necessarily have large minimum change intervals.
- A large discrepancy between the amount of cachable and changing objects has been observed for HTML documents. The reasons for this are objects which are generated dynamically but whose content does not change at all or does not change with a high frequency.
- Only 7% of the investigated URLs were at least duplicated twice while the majority of 93% has no duplicates.

Future work will include a more detailed statistical analysis of various other aspects not yet covered such as the growth of the sites observed. We will also extend our analysis to more sites observed over a longer time period. We plan to conduct a passive proxy trace based measurement for a user group where the above investigated web sites are popular. This will allow us to compare the future trace-based analysis results with the current analysis results and to find out whether the statistics gathered passively are representative or not. The combination of data about user specific access patterns and data about the cachability and dynamic behavior of web sites is a basis for future work which will investigate the efficiency of different caching architectures and mechanisms for capacity inverted networks.

We will also work on a more efficient active probing mechanism. We think that such a mechanism can be realized by using sampling techniques. Using sampling it would be possible to get representative statistics for complete sites by only looking at a small subset of objects. The sampling interval could be adapted according to the change interval of the content so that it is possible to more accurately measure the dynamic behavior of the sampled objects. Another very important research direction is to find solutions for improving the cachability of dynamically generated content.

REFERENCES

[1] V. N. Padmanabhan, L. Qiu, "The Content and Access Dynamics of a Busy Web Server: Findings and Implications", *Technical Report MSR-TR-2000-13*, Microsoft Research, Feb. 2000.

[2] A. Wolmann, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, H. Levy, "Organization-Based Analysis of Web-Object Sharing and Caching", *In Proc. of the 2nd USENIX Symposium on Internet Technologies and Systems*, Oct. 1999.

[3] T. Kroeger, D. Long, J. Mogul, "Exploring the bounds of web latency reduction from caching and prefetching", *In Proc. of the USENIX Symposium on Internet Technologies and Systems*, Dec. 1997

[4] W. Koehler, "Digital libraries and World Wide Web sites and page persistence", *Information Research, Volume 4 No. 4*, Jun. 1999

[5] F. Douglis, A. Feldmann, B. Krishnamurty, J. Mogul, "Rate of Change and other Metrics: a Live Study of the World Wide Web", *In Proc. of the USENIX Symposium on Internet Technologies and Systems*, Dec. 1997

[6] B. E. Brewington: "Observation of changing information Sources", *PhD Thesis, Thayer School of Engineering, Darthmouth College*, Jun. 2000

[7] T. Kelly, J. Mogul, "Aliasing on the World Wide Web: Prevalence and Performance Implications", *WWW2002, ACM 1-58113-449-5/02/0005*, May 2002

[8] R. Fielding et al. "Hypertext Transfer Protocol -- HTTP/1.1", IETF RFC2616, Jun. 1999

[9] Squid Cache, http://www.squid-cache.org, Mar. 2003

[10] MnoGoSearch Engine, http://search.mnogo.ru, Mar. 2003

[11] MySQL Database, http://www.mysql.com, Mar. 2003