Improving Web Performance by a Differencing and Merging System

Aleksandar Jevremovic[1], Ranko Popovic[1], Dejan Zivkovic[1], Mladen Veinovic[1] and Goran Shimic[2]

[1] Department of Informatics and Computing, Singidunum University, Belgrade, 11000, Serbia

[2] Military Academy, Belgrade, 11000, Serbia

Abstract

In this paper we consider the problem of improving Web performance and propose an efficient differencing and merging system (DMS) based on an HTTP protocol extension. To provide for faster information exchange over the Web, the system tries to transfer only computed differences between requested documents and previously retrieved documents from the same site. Analysis and experimental results prove the effectiveness of DMS, but also show bigger processor and memory load on servers and clients. DMS is compatible with most of the existing solutions for improving Web performance. Moreover, SSL security system may be used to provide Web privacy and authenticity. The DMS model is simple to use and can be relatively easily integrated in Web servers and browsers.

Keywords: **Web performance, HTTP protocol, HTML differencing.**

1. Introduction

Web performance is the key ingredient to an effective use of the Internet. In case of static Web pages and information portals, better Web performance exerts positive influence on the user's impression about the content. Also, dynamic Web pages and Web applications directly depend on Web performance, because speed of information exchange ultimately determines the kind of functionality that is implementable in a Web application. More generally, Web performance has a great impact on smooth migration from desktop-based to Web-based applications, as well as on success of the cloud-computing model.

Improving Web performance has been the goal of many different attempts. Popular approaches include both direct and indirect methods. The direct methods are based on contents cashing, contents compression, and AJAX calls, while the indirect methods are based on new network protocols such as SCTP, MUX/SMUX, and SPDY. The next section presents some advantages and drawbacks of these popular approaches to improve Web performance.

Main contribution of this paper is a model for a novel approach regarding the Web performance improvement. The proposed model is based on a differencing and merging system, hence it is called DMS. In a nutshell, DMS makes use of structural similarities between HTML pages. Consequently, for a client to get a page from a server, it makes sense that the client requests the smallest differences between the desired page and previously retrieved pages from the same server. If the server's difference computation procedure is not time intensive, the desired page in this way may be delivered . much faster to the client because the server transfers smaller amounts of data. The client on its part, however, needs to merge received differences to reproduce the correct version of the page.

2. Related Work

In the last few decades there has been a growing interest in improving Web performance. Many existing solutions to the problem basically try to reduce the amount of data transfered between servers and clients. Basic technique to reduce the network traffic is contents cashing, i.e., reuse of previously retrieved Web pages that are kept locally for a fixed time interval. The main drawback of contents cashing is possibility of using stale contents or retrieving unchanged pages. Among many attempts to improve efficiency of contents cashing [22], [18], [20], [21], [11], the most prominent approach is a farm of proxy servers that create a network of cached pages. This relieves the burden on central servers and speeds up contents distribution to clients. However, maintaining consistency of such a network requires specific architecture based on agents [11].

Another standard technique to reduce the network traffic is data compression. Support for data compression is built into the HTTP protocol from version 1.1 and it is supported by all modern Web browsers. Compression of textual Web contents may save up to 75% of the contents size, while the overall savings figure for general contents amounts to about 37% [12]. Reducing the network trafic by means of data compression is compatible with the DMS model.

A more complex method for improving Web performance is contents fragmentation and asynchronous transfer of fragments based on JavaScript calls (AJAX). Advantages of this method lie mainly in its compatibility with the service oriented architecture, while drawbacks include necessity to redesign existing applications.

Two leading attempts to improve Web performance by protocol replacement are SMUX protocol [5] and HTTP over SCTP protocol [16]. SMUX is an attempt of the W3C consortium to design a lightweight application protocol and to make migration to future Web protocols simpler. However, SMUX protocol is still in an experimental phase. HTTP over SCTP protocol works by replacing traditional TCP transport protocol with more efficient SCTP protocol [4]. Using HTTP over SCTP is particularly relevant to the DMS model, since HTTP over SCTP gives good results in developing regions [15], i.e., regions with poor communication infrastructure. In such an environment DMS could be very useful and greatly speed up the information exchange.

In an attempt to improve Web performance, Google is making an important step forward with its SPDY protocol [6]. Goal of this protocol is to enable multiple data streams within one HTTP connection. In addition, this protocol compresses data during data transfer and excludes the “host” field from the request header. Another interesting method for decreasing contents lag, which is used in Google’s Chrome browser, is so called DNS prefetching. This function actively performs domain name resolutions for the domain name references found in hyper links of each document.

Opera’s attempt to improve Web performance is based on its technology called Opera Turbo [2]. This is a commercial technology that uses specialized intermediary servers and data compression to reduce the network traffic.

The most common system for tracking and viewing changes on the Web is probably WebVigiL [9]. This system automates the change detection of HTML/XML documents and timely notifies users about changes based on user-specified changes of interest. In WebVigiL, two separate change detection algorithms are used to detect changes occurring to HTML and XML documents.
The problem of comparing two HTML documents to find the differences between them can be simply solved by comparing textual content of the documents, e.g. using traditional UNIX diff algorithm. However, comparing HTML documents at the lexical level, as if they were ordinary text documents, is neither informative nor intuitive. VDiff algorithm [13] gives better accuracy by taking documents’ internal tree structure into account. Still better results can be obtained in case of well-structured XML documents by using their tree representation and identifying common nodes and subtrees [7]. The hierarchical structure of XML documents has been exploited recently to detect their differences by using semantic analysis of such documents [8], [19], [14].

3. DMS Model

This section describes the proposed system and its role in the client-server communication. The central idea around DMS is that transferring differences between documents instead of full documents would reduce the network traffic. Upon receiving request for a document from a client, a server computes the differences between the document it needs to send and documents that the client already retrieved from the server. The server then selects the smallest size difference and sends it to the client. The client finally merges received difference with the corresponding local document to reconstruct the correct version of the document.

A DMS enabled server must keep track of served documents (and differences) and assign a unique ID to each document it serves. Similarly, a client must keep track of retrieved documents (and differences). Figure 1 illustrates DMS role in the client-server networking model in the simplest case of static Web setting.

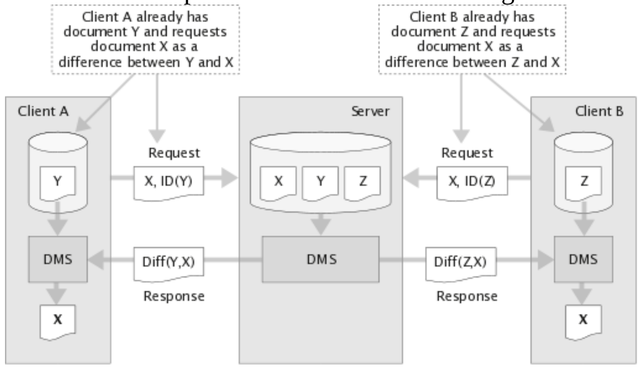

Client A already has document Y and requests document X as a difference between Y and X
Client B already has document Z and requests document X as a difference between Z and X
Client A
Server
Client B
Request
X, ID(Y)
Request
X, ID(Z)
Y
X
Y
Z
Z
DMS
Diff(Y,X)
DMS
Diff(Z,X)
DMS
Response
Response
X
X

Fig. 1. DMS model in the static Web setting

Figure 1 depicts the state when clients A and B already requested some documents from the same server in the past and retrieved and stored them locally: client A retrieved document Y and client B retrieved document Z. Now both clients in Figure 1 make request for document X to the server. Along with the request for X, client A also sends to server the document ID of Y it has locally as a reference document for computing differences. Server as its response computes differences between the documents X and Y and sends them to client A. Client A then merges received differences with its local copy of Y to reconstruct the document X. Similarly, to request the same document X, client B sends along the document ID of Z it has locally, and the server response are differences between the documents X and Z.

More interesting environment for DMS is the dynamic Web setting and Web applications. In this case, URL address is not sufficient resource identifier, since various requests with the same URL address may produce different results. Moreover, Web applications often dynamically customize user interface as a function of the user account type and user activities. Thus, different users can get different results for the same request.

This specific features of dynamically created pages and Web applications imply that a DMS enabled server should limit computation of differences between pages transferred only within user sessions. Such a server also needs to assign a special unique ID to each page obtained as a result of the dynamic document creation procedure. This means that a DMS enabled server should reserve separate space for each client to maintain already transfered pages per session.

Figure 2 illustrates DMS role in the dynamic Web setting. For each client, a DMS enabled server keeps history of pages delivered within current session. A client in subsequent requests sends IDs of some of the pages as a reference for difference computation. In Figure 2, client A retrieved three pages P1, P2, and P3 within current session, while client B retrieved two pages. Subsequently, client A requests fourth page P4 and sends along IDs of P1, P2, and P3. The server then computes differences between P4 and each of P1, P2, and P3, and returns the smallest difference as its response. The pages Q1 and Q2 delivered to client B within current session are kept separately, and client B's request for page Q3 is serviced with the smallest difference between Q3 and Q1.

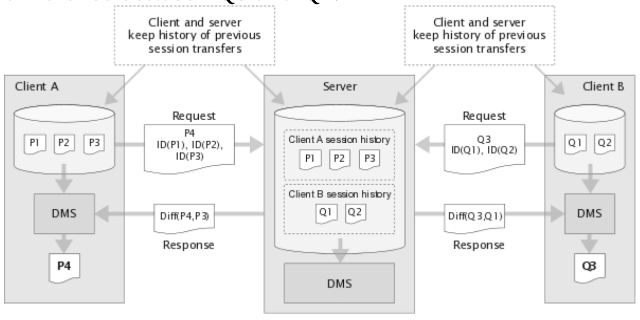


Fig. 2. DMS model in the dynamic

Web setting Note that DMS functionality is not limited to HTML pages. DMS may be also used with other textual resources (CSS, JavaScript...), as well as with binary resources (pictures, sounds...) using appropriate difference computation algorithms.

2.1 DMS Implementation

To implement DMS, it is necessary to modify the HTTP protocol itself [3] and both sides of the client-server networking model. Changes to the HTTP protocol include support for the document IDs inside HTTP messages. A new field "Accept-Diff", for example, may be added to the header of a HTTP message. This field in a HTTP request informs server that client supports DMS and contains values of document IDs as parameters for computing differences. This field in a HTTP response contains ID of the document that server has used to compute (the smallest) returned differences.

In case that server does not contain any of the documents specified in an HTTP request, it sends the requested document in its fullness. This also applies if server does not support DMS at all.

The server side implementation of DMS may take two basic forms: direct integration in the server software, or indirect support through a special service. The first approach is compatible with data compression and SSL security, but requires redesign of the Web server software. On the other hand, an independent service (which listens to a port, forwards received requests to the Web server, and sends computed differences to a client) does not necessitate changes to the original software, but it is not compatible with data compression and SSL security.

Web browser on the client side must support sending HTTP requests with the "Accept-Diff" field whose values are documents IDs that the client has stored locally. Thus, the client needs to select one or many (perhaps all) of the local documents and include their IDs in a HTTP request. Also, the client from a HTTP response needs to extract received differences and merge them with the document whose ID is specified in the "Accept-Diff" field.

A server or a client may choose to store full pages or only page differences for later use. The choice is independent and depends on the particular server and client processing power, storage capacity, and other factors that determine response time and storage requirements. For example, storing only page differences favors storage to response time, because it might be necessary to apply the merging procedure repeatedly to get the correct page.

4. DMS on the Web

This section investigates two applications of DMS on the Web, besides the direct client-server communication. First, if the client-server communication goes indirectly through a proxy server, then the proxy server must support both the client and server side of

DMS functionality. Figure 3 shows the general information flow between a client and a server in the presence of one or more proxy servers.

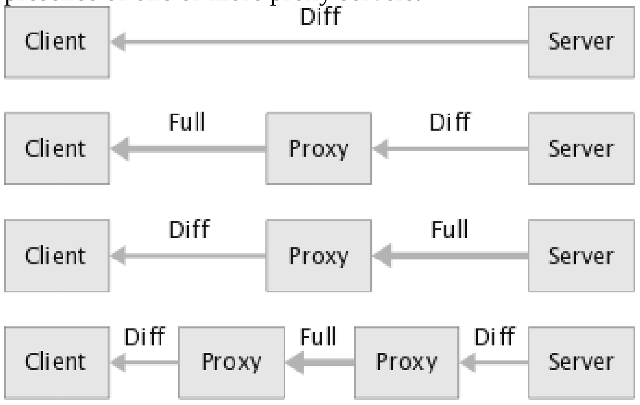


Fig. 3. DMS in the proxy server setting

Using a DMS enabled proxy server may prove especially efficient when retrieving documents from a Web server on behalf of clients. A proxy server in this case can use many already retrieved documents to request the smallest difference from the server. Moreover, to fulfill a request for one client, the proxy server may use retrieved documents received for other clients.

Second important application where DMS may find its use is the distributed contents delivery to users in geographical proximity through close distribution servers. Figure 4 illustrates DMS usage in the distributed contents delivery setting.

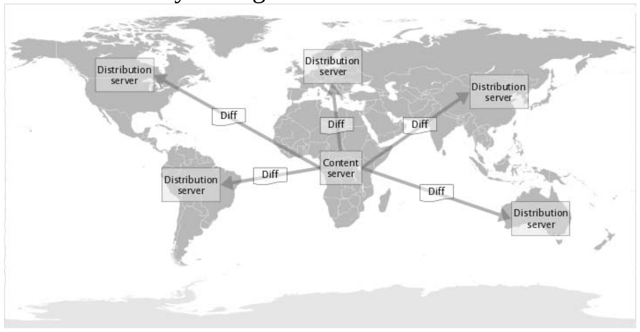


Fig. 4. DMS in the distributed contents delivery setting

Aim of using DMS in the distributed contents delivery setting is not taking the load off of the central content server, since distribution servers still forward client requests to the central server. Instead, DMS usage can result in lower datatraffic volume because of the transfer of smaller document differences from the central server to distribution servers. Similarly to a proxy server, a distribution server may request differences based on already received documents from all sessions.

Main benefit of this solution is central processing of HTTP requests with efficient distributed contents delivery. Moreover, this efficiency is achieved simply without complex schemes involving load balancing among distribution servers and their contents synchronization.

5. DMS Performance

This section analyzes DMS efficiency by evaluating its impact on the data-traffic volume. This is done by means of two experiments—one for a Web site and the other for a Web application.

It is clear that DMS usage is justified if the total time for computing, transferring, and merging document differences is less than the time for transferring a full document. DMS efficiency thus depends mostly on the similarity of requested and local documents, as well as on the efficiency of an algorithm for computing and merging differences. However, an algorithm for merging differences is determined by the corresponding algorithm for computing differences, so we only consider the latter algorithm efficiency.

If an HTTP request contains only single local document ID, then DMS efficiency on the server side is dominated by the efficiency of an algorithm for computing differences. In general, this algorithm efficiency can be measured using two opposing criteria: the time for computing differences and the size of computed differences. However, an algorithm for computing differences may choose a trade-off between the two criteria, since larger difference computation time can be made up for smaller transfer time, and vise versa. Moreover, an adaptive algorithm may be further tuned up to base its optimal decision on the processor load and the communication bandwidth.

On the other hand, if an HTTP request contains many local document IDs, then DMS performance is a function of the total time needed for computing differences with respect to all given local documents. In this case, better performance is still possible using an advanced method to select the best document that gives the smallest difference.

5.1 Experimental Results

Since the primary goal of this paper is to show soundness of the DMS concept, we have conducted two basic experiments to confirm DMS effectiveness. To get a first glimpse at the DMS performance, in the experiments we have not tested any of the real DMS functionality needed in a client and a server. Instead, we have restricted ourselves to analyze only the DMS impact on the amount of transfered network data.

To this end, we have used two of the most common contents publishing models on the Web—a Web site and a Web application. For the Web site example we have chosen the Times Online site www.timesonline.co.uk. Since we could not use DMS with commercial servers and clients, testing is performed by first retrieving some pages using ordinary browser, and then computing the page differences using the GNU diff implementation of the basic difference computation algorithm [1]. On the other hand, for the Web application example we have developed custom chat application in PHP language.

5.2 The Web Site Example

To estimate DMS impact on the speed of data transfer from a Web site, we have taken ten randomly selected documents from the science section of the Times Online site at the address http://www.timesonline.co.uk/ tol/news/science/:

1. physics/article7121202.ece
2. physics/article7131244.ece
3. medicine/article7121219.ece
4. medicine/article7125803.ece
5. eureka/article7110849.ece
6. genetics/article7120516.ece
7. biology_evolution/article7132299.ece
8. biology_evolution/article7127753.ece
9. genetics/article7128357.ece
10. eureka/article7115855.ece

These documents were taken from the site in this order on May 23, 2010 in time frame of two minutes. As each document's HTML page was stored in a local file, differences between the page and all previous pages were computed. Table I summarizes the page sizes and computed differences in bytes, where the smallest difference is highlighted in bold.

Table 1: Sizes of ten pages and mutual differences

-	Size	1	2	3	4	5	6	7	8	9	10
1	180238	-	-	-	-	-	-	-	-	-	-
2	179681	**83186**	-	-	-	-	-	-	-	-	-
3	156594	**83140**	84726	-	-	-	-	-	-	-	-
4	162835	82595	83613	**68327**	-	-	-	-	-	-	-
5	148804	91125	91527	**67453**	74039	-	-	-	-	-	-
6	187365	92658	96554	**89766**	94969	97095	-	-	-	-	-
7	185722	**89562**	91740	92577	90015	96531	99034	-	-	-	-
8	179537	86922	87551	88563	**86143**	89534	96001	86520	-	-	-
9	150343	87284	87229	61810	70055	**56766**	88474	92523	85916	-	-
10	146289	83352	88486	63694	72074	**54365**	94110	90900	87517	56801	-

Figure 5 gives the smallest difference obtained for each page in terms of the percentage of the original size.

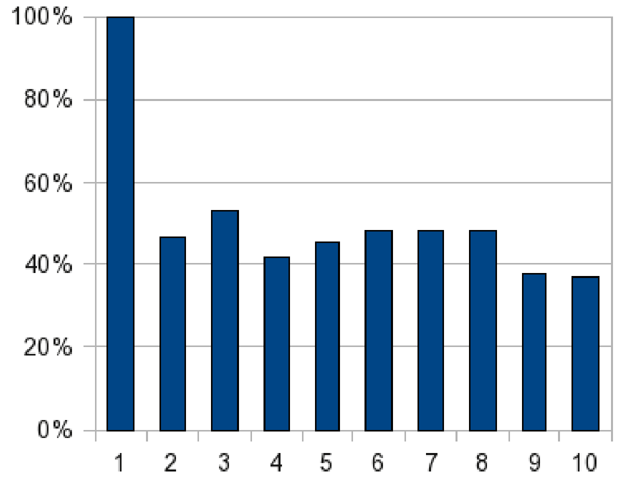


Fig. 5. The smallest differences with respect to the page sizes

The test results show that using DMS even for relatively unrelated documents would reduce the network data traffic more than 50% in most cases (except the first). Thus, taking into account that

- the Times Online site has over three million visitors each day (source: www.statbrain.com)
- the visitors of this site open 2.5 pages on average per visit (source: www.alexa.com)

and extrapolating the results obtained in the testing scenario that

- the average page size is about 160 KB
- the expected savings in the data transfer is about 50%

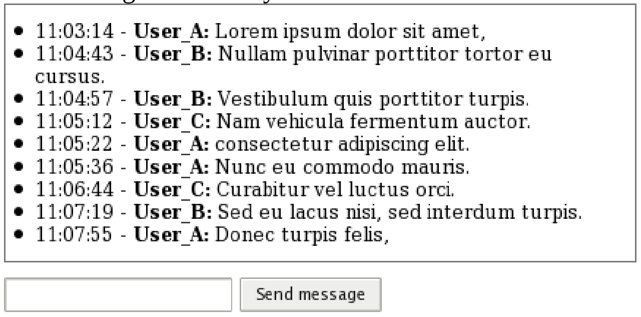
we may conclude that DMS would overall save about 360 GB in the data-traffic volume each day in case when the first page is fully transfered, or 600 GB in case when pages from previous visits are used as references to difference computations.

5.3 The Web Application Example

To estimate DMS efficiency in the context of a Web application, we have chosen an application for short message exchange between users. Motivated by the Lift Web framework [17], this minimalistic chat application has been easily developed in PHP.

The main window of the chat application contains at most 10 messages. When a server adds a new message to the full window, it removes the oldest message from the window. A client sends messages using an HTML form, and the server embeds them into a page using PHP. For the sake of simplicity, a client retrieves a complete page each time it checks whether a new message appeared in the meantime.

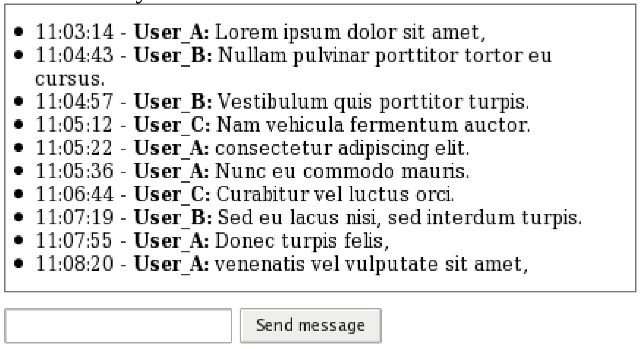
Testing scenario for the application has been devised to comprise three steps—in each step a new message is added to the application and then the relevant HTML pages are compared. Initially, there have been nine messages exchanged as shown in Figure 6, and size of the the corresponding HTML page that contains the nine messages is 1063 bytes.



- 11:03:14 - **User_A:** Lorem ipsum dolor sit amet,
- 11:04:43 - **User_B:** Nullam pulvinar porttitor tortor eu cursus.
- 11:04:57 - **User_B:** Vestibulum quis porttitor turpis.
- 11:05:12 - **User_C:** Nam vehicula fermentum auctor.
- 11:05:22 - **User_A:** consectetur adipiscing elit.
- 11:05:36 - **User_A:** Nunc eu commodo mauris.
- 11:06:44 - **User_C:** Curabitur vel luctus orci.
- 11:07:19 - **User_B:** Sed eu lacus nisi, sed interdum turpis.
- 11:07:55 - **User_A:** Donec turpis felis,

Fig. 6. Initial nine messages in the chat application

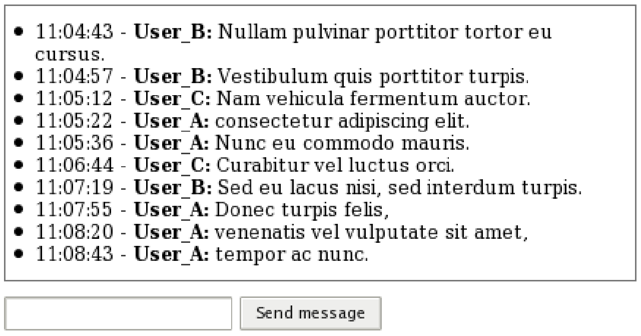
In the second step, a new message is added to the application for the total of ten messages (Figure 7). The size of the corresponding HTML page has increased to 1148 bytes. However, by using DMS only differences between the first and the second page would be transferred. The size of the differences in the diff format is mere 93 bytes.



- 11:03:14 - **User_A:** Lorem ipsum dolor sit amet,
- 11:04:43 - **User_B:** Nullam pulvinar porttitor tortor eu cursus.
- 11:04:57 - **User_B:** Vestibulum quis porttitor turpis.
- 11:05:12 - **User_C:** Nam vehicula fermentum auctor.
- 11:05:22 - **User_A:** consectetur adipiscing elit.
- 11:05:36 - **User_A:** Nunc eu commodo mauris.
- 11:06:44 - **User_C:** Curabitur vel luctus orci.
- 11:07:19 - **User_B:** Sed eu lacus nisi, sed interdum turpis.
- 11:07:55 - **User_A:** Donec turpis felis,
- 11:08:20 - **User_A:** venenatis vel vulputate sit amet,

Fig. 7. The chat application after the second step

In the third step, once a new message is added, the oldest message is removed (Figure 8). The size of the corresponding HTML page has been decreased to 1136 bytes. The sizes of the differences between the third and the second page and the third and the first page are 161 bytes and 251 bytes, respectively.



- 11:04:43 - **User_B:** Nullam pulvinar porttitor tortor eu cursus.
- 11:04:57 - **User_B:** Vestibulum quis porttitor turpis.
- 11:05:12 - **User_C:** Nam vehicula fermentum auctor.
- 11:05:22 - **User_A:** consectetur adipiscing elit.
- 11:05:36 - **User_A:** Nunc eu commodo mauris.
- 11:06:44 - **User_C:** Curabitur vel luctus orci.
- 11:07:19 - **User_B:** Sed eu lacus nisi, sed interdum turpis.
- 11:07:55 - **User_A:** Donec turpis felis,
- 11:08:20 - **User_A:** venenatis vel vulputate sit amet,
- 11:08:43 - **User_A:** tempor ac nunc.

Fig. 8. The chat application after the third step

The test results show that using DMS in the chat application would reduce the network traffic to almost 90%. Thus, it is fair to expect that DMS can be successfuly applied in other Web applications that do not use service calls and do not get page changes as service messages (SOAP, JSON, . . .).

6. Web Security and Privacy

The DMS model is compatible with standard systems that provide HTTP security and privacy. For example, the SSL security system can be used with DMS without modifications, because SSL is part of the message transport layer, while DMS acts on the application level above HTTP.

There is one situation, however, where basic DMS can be abused by a malicious user to get hold of a private document. Consider the scenario when client requests a private document protected by the SSL system and assume that server sends full version of the document as response. Client then leaves protected communication and makes a new non-SSL request for some other public document. Assume further that, for some reason, client in the request specifies ID of the protected document received during secure communication. As a response then client receives clear differences between the private document and the pubic document.

An eavesdropper can thus learn two items: URL address of the public document and clear differences between the private document and the public document. In this case, an attacker can independently request full version of the public document and use the differences in a reverse-merging procedure to obtain the protected document.

This scenario reveals a potential problem of DMS abuse that should be kept in mind when implementing DMS for a secure environment. More specifically, a DMS enabled client implementation should

1. distinguish IDs of unsecured and secured documents, i.e., those received using HTTP and HTTPS protocols, and should not mix them;

and

2. limit document IDs in a HTTPS request to those received only in the current session.

7. Conclusion and Future Work

This paper proposes DMS as a solution for an effective use of the Web. Since Web pages are well-structured and contain many common HTML language constructs, an intuition is that differences between Web pages are small despite their content. Thus, to reduce the network traffic, it pays off to transfer differences between pages instead of full pages. This is the main idea behind DMS whose soundness is justified by experimental results.

DMS is compatible with most of the existing solutions for improving Web performance. For example, clients may use ordinary document caching methods before turning to DMS, or servers may compress differences before sending them. Moreover, SSL security system may be used to provide Web privacy and authenticity.

The DMS model is simple to use and can be relatively easily integrated in Web servers and browsers. It is also general in the sense that its main idea with small modifications can be used in other problem domains such as routing, load balancing, and computer forensics [10].

Obvious subject of future work is a software implementation of DMS and evaluation of its efficiency in production environments. The software design will certainly include heuristics and algorithms for selection of documents being compared and for computation of document differences. An efficiency analysis of different design decisions that are made in this regard is also necessary for broad DMS adoption on the Web.

Acknowledgments

This work has been supported by the Serbian Ministry of Education and Science (projects III44006 and TR32054)

References

[1] GNU Diffutils. Available from: http://www.gnu.org/software/diffutils
[2] Opera Turbo. Available from: http://www.opera.com/business/solutions/ turbo
[3] RFC2616: Hypertext transfer protocol – HTTP/1.1. Available from: http://www.w3.org/Protocols/rfc2616/rfc2616.html
[4] RFC4960: Stream control transmission protocol. Available from: http: //tools.ietf.org/html/rfc4960
[5] SMUX protocol specification. Available from: http://www.w3.org/TR/ WD-mux
[6] SPDY protocol. Available from: http://www.chromium.org/spdy/ spdy-protocol
[7] Al-Ekram, R., Adma, A., Baysal, O.: diffx: An algorithm to detect changes in multi-version XML documents. In: Conference of the Centre for Advanced Studies on Collaborative Research (2005)
[8] Al-Namiy, A.Q., Majeedl, F.S.: Towards automatic extracted semantic annotation (ESA) for Web documents. In: APCIP 2009, vol. 2 (2009)
[9] Chakravarthy, S., Hara, S.: Automating change detection and notification of Web pages. In: 17th International Workshop on Database and Expert Systems Applications (DEXA 2006) (2006)
[10] Chen, L., Wang, G.: An efficient piecewise hashing method for computer forensics. In: Proceedings of the First International Workshop on Knowledge Discovery and Data Mining (2008)
[11] Kannammal, A., Padmanabhan, R., Iyengar, N.: Web cache consistency maintenance through agents. In: Proceedings of the Second International Conference on Communication Software and Networks, pp. 329–333 (2010)
[12] King, A.B.: Speed Up Your Site: Web Site Optimization. New Riders (2003). Available from: http://www.websiteoptimization.com/speed/18/ 18-2t.html
[13] Mikhaiel, R., Stroulia, E.: Accurate and efficient HTML differencing. In: 13th IEEE International Workshop on Software Technology and Engineering Practice (2005)
[14] Moon, H.J., Yoo, J.W.: Free-traversing syntactic and semantic comparison on semi-structured languages. In: Convergence and Hybrid Information Technology (ICHIT 2008) (2008)
[15] Natarajan, P., Amer, P.D., Stewart, R.: Multistreamed Web transport for developing regions. In: The second ACM SIGCOMM workshop on Networked systems for developing regions (2008)
[16] Natarajan, P., Iyengar, J.R., Amer, P.D., Stewart, R.: SCTP: An innovative transport layer protocol for the Web. In: Proceedings of the International World Wide Web Conference (2006)
[17] Pollak, D., Vinoski, S.: A chat application in Lift. IEEE Internet Computing 14(3), 88–91 (2010)
[18] Ramaswamy, L., Liu, L., Iyengar, A.: Cache clouds: Cooperative caching of dynamic documents in edge networks. In: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005) (2005)
[19] Santos, R.C., Hara, C.S.: A semantical change detection algorithm for XML. In: SEKE 2007 (2007)
[20] Sharman, R., Ramanna, S.S., Ramesh, R., Gopal, R.: Cache architecture for on-demand streaming on the Web. ACM Transactions on the Web (TWEB) 1 (2007)
[21] Sheu, T.L., Yang, C.H.: A novel hierarchical cache architecture for WWW servers. In: Proceedings of the 15th International Conference on Information Networking (ICOIN 2001), p. 863 (2001)
[22] Yin, J., Alvisi, L., Dahlin, M., Iyengar, A.: Engineering Web cache consistency. ACM Transactions on Internet Technology 2(3) (2002)