THE AFFECTS OF CACHING IN BROWSER STAGE ON THE PERFORMANCE OF WEB ITEMS DELIVERY

Waheed Yasin, Hamidah Ibrahim, Nor Asilah Wati Abdul Hamid, Nur Izura Udzir
Faculty of Computer Science and Information Technology,
Universiti Putra Malaysia,
43400 UPM, Serdang, Selangor Darul Ehsan, Malaysia
Email: waheedos80@yahoo.com, [hamidah, asila, izura]@fsktm.upm.edu.my

ABSTRACT

Network congestion remains one of the main barriers to the continuing success of the Internet. Caching is a way to reduce traffic load on the server and network backbone, which improves the efficiency and scalability of web items delivery. Caching in computer networks might be performed in different stages. In this article, we investigate the load that web pages can put on a network and how caching can reduce the bandwidth requirements. This article concludes that caching in browser stage improves the delivery of web items.

KEYWORDS

Caching, Web contents delivery, Web browsing

1 INTRODUCTION

Network bandwidth and the traffic load are the main concerns for network administrators [1]. Moreover, repeated traffic that are being sent to web servers waste the bandwidth of the network, which might lead to more delays for web users due to the increase in traffic on the Internet [2], [3]. On the other hand, they might push network administrators to extend networks' gateways at rates that are out of proportion to the end-users' growth [4]. These extensions of bandwidth are immediately consumed without reaching the expected improvement of network performance that makes web users unsatisfied.

Caching is one of the popular approaches that is considered as a way to reduce traffic load on the server and network backbone, which improves the efficiency and scalability of web items delivery.

Caching in web systems might be performed in different stages. The aim of this article is to show the affects of caching in browser stage on the performance of web items delivery using three different web browsers.

This article is organized as follows: A background of caching is presented in section 2. Section 3 presents the methodology. Results and discussion are presented in section 4. Related works are presented in section 5. Section 6 presents the conclusion and future works. In section 7, all the references are listed.

2 CACHING

The term caching has a French origin, which means storing. In computer engineering, caching is the process of storing data in an intermediate media which is called a cache and it is used for serving future queries instead of fetching data from original sources. This procedure will speed up the throughput of the system by answering these queries. The data which is stored in the storage of cache is a replication of the original data that belongs to the main source. In this section the stages of caching is presented in addition to caching benefits and limitations.

2.1 Caching Stages

Caching approach should be beneficial for computer network administrators in order to reduce web user delays caused by repeated traffic.

Thus, web caching might be performed in different stages [3], which are listed as follows:

- *Browser stage.* In this level caching is performed close to the web user such as caching using computer's hard disk.

- *Proxy stage.* Proxy caches or sometimes called proxy servers are located at the network edges in client-server systems. Proxy servers can store files and directly serve users' requests in the network. Thus, it reduces the traffic which needs to be transmitted across the network, which improves the responsiveness for web users.

- *Web server stage.* In this stage, caching is performed at the source of web content, which is usually at the network of Internet Service Providers (ISPs), which is a web cache that is shared among all users of that network.

Figure 1 illustrates caching levels in computer networks.

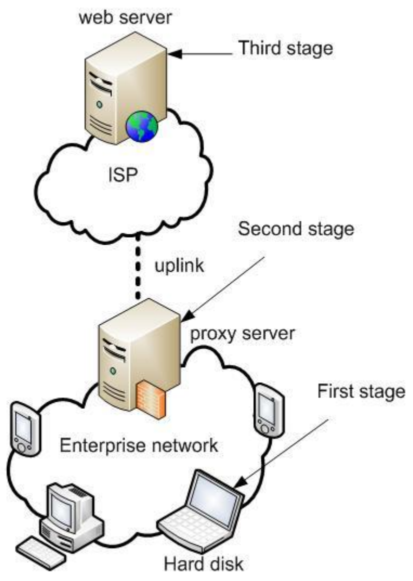


Fig. 1 Caching Stages in computer networks

2.2 Benefits of Caching

Many benefits can be gained by deploying caching in the computer systems, some of these benefits are listed below.

- *Reducing the system load*. When the system has to answer a query, it is not required to generate new information; rather a consistent copy of this information can be delivered from the cache.

- *Reducing the latency*. Using caching allows answering queries from the cache storage that results in reducing the latency, because the response time which is taken to deliver the content from the original server is often more than the response time taken when delivering the content from the cache.

- *Reducing the bandwidth consumption.* Caching is used to save the bandwidth. That is because contents do not need to be retrieved many times; rather they need to be retrieved once by the cache and only need to be maintained for consistency reasons. For example, if the cache is located in a Local Area Network (LAN), the distance between this cache and clients is too small compared to the distance between the origin server and these clients. Thus, retrieving the content from the original server many times waste the outgoing link (uplink). Furthermore, it is known the uplink is often considered as the bottleneck of any enterprise. Thus, caching allows retrieving contents from the storage with higher bandwidth capacity. Also, caching often used compression techniques that can results in saving bandwidth.

2.3 Caching Limitations

Although there are a lot of benefits that a cache can provide, caching has limitations that have to be considered when a cache is deployed in the system. Some of these limitations are discussed in this section.

- *Data Consistency*. The consistency of data in the cache is one of the most important issues that should be considered. Web contents might be updated in the original servers which results in data invalidity in the cache. Thus, when there is a request for data which has just been updated, the cache has to deliver this updated data. Some researches called the copy of data as a replica. Many researches have been conducted in the area of cache consistency [5].

- *Data Privacy.* Another issue should be considered in caching is data privacy. That means sometimes there is some data that are considered as sensitive data. These kinds of data should be maintained in a different way. One technique might be used is to categorize data into two groups that are non-cacheable data and cacheable data. A non-cacheable data is the data that is considered as a sensitive data; while a cacheable data is the data that is not considered as sensitive as the previous category [5]. Thus, the cache might only keep cacheable data in its storage.

- *Data Replacement.* Replacement mechanism is a mechanism which refers to the process that happens when the cache becomes full and there is no enough space for new items, which leads to the task of removing old items to make spaces for new ones. Thus, replacement mechanism has to decide which items are needed to be stored, and which items have to be removed. Many factors should be considered when the cache has to take a replacement decision [6]. Some of these factors are listed below.
  - Recency: time since the last using of a web object.
  - Frequency: total number of requests of a web object.
  - Size: size of the web object.
  - Cost of fetching a web object: cost to fetch a web object from the origin source including processing, bandwidth and other resources.
  - Modification time: time since the last modification.
  - Expiration time: time as soon as the web object becomes useless and it is able to be replaced.

3 METHODOLOGY

A web page normally consists of many portions that figure out its final appearance. A portion is sometimes called a fragment and it has some characteristics that are used to distinguish it from other fragments. Thus, a web object may contain more than one fragment. These fragments might be static or dynamic, and as it is known caching these fragments improves the performance of the network. However, caching static fragments does not cause problem to the system, the challenge of data consistency appears when dynamic contents have to be cached, furthermore contents' privacies are not able to be cached.

On the other hand, some of these fragments might have different Time-To-Live (TTL), thus web page fragmentation allows the system to only fetch fragments that have already expired. For example, a news web site may contain different fragments such as news provider's name, breaking news, advertisements, etc. Fragments might be reassembled in proxy stage. One of the standards that is recognized to perform the reassembly is called Edge Side Includes (ESI). A cache, which implements ESI standard for fragments reassembly, may figure out the web page from cached fragments or fetch them from any other authoritative resource such as other caches or original server. A drawback of implementing ESI standard for fragments reassembly is the traffic bottleneck imposed by this standard specially when clients are connected to the Internet via dial-up lines. Thus, another approach has been proposed in [7] in order to reassemble fragments at the client's side which is called Client Side Includes (CSI). An issue about fragment based web caching that should be considered is the level of fragmentation and how that will provide optimal performance for caches. On the other hand, the techniques that can be used for publishing fragments also have to be considered.

The most common technique to create these fragments could be performed by the web designer. In other words, web designers are responsible for dividing the web page into fragments, and the cache has to have a full knowledge about this fragmentation. Automatic web pages' fragmentation is another technique which can be used to perform web page fragmentation. In this technique, the proxy cache should detect the fragments that form the web page.

In our experiment, we use three different web browsers to show the benefits of caching on the first stage of caching that are Mozilla Firefox version 14.0.1, Windows Internet Explorer version 9.0.8112.16421, and Google Chrome version 21.0.1180.83 m.

For Mozilla Firefox we had to allow a plug-in which is called Firebug version 1.10.2. Similarly, for Windows Internet Explorer from Tools menu we have chosen Developer Tools then Network Tab; however, for Google Chrome we had to allow Page load time add-in.

A laptop with the characteristics that are shown in Table I has been used in our experiment. This laptop has a connection to a LAN with 100 Mbps speed. This LAN is connected to the internet core switch of the ISP which is called YEMENNET.

Table 1. Laptop characteristics

Description	**Item**
TOSHIBA	System Manufacturer
Windows Vista Home Basic	Operating System
Intel(R) Core(TM) Duo CPU T850 @ 2.16 GHz 2.17 GHz	Processor
2.00 GB	Memory (RAM)
32-bit Operating System	System Type

4 RESULTS AND DISCUSSION

Three different web sites have been browsed, two of them are not frequently modified that are *www.gso.upm.edu.my* and *www.fsktm.upm.edu. my*, while the third one is news website which is *www.bbc.com* and is frequently modified. A key point has to be mentioned is these websites have been chosen randomly and have been assigned to different web browsers that have been mentioned earlier.

Firstly, we used Mozilla Firefox to open the *www.gso.upm.edu.my* web site. From Fig. 2-a, it has been observed that for the first time it took total time 1 minute and 55 seconds to open the page.

On the other hand, It took only 8.91 seconds to open the page and all the web page contents were loaded from the cache as illustrated in Fig. 2-b.

It can be observed that *www.gso.upm.edu.my* web page has the size of 5.9 MB, and is not frequently modified, thus all the 5.9 MB has been loaded from the cache.

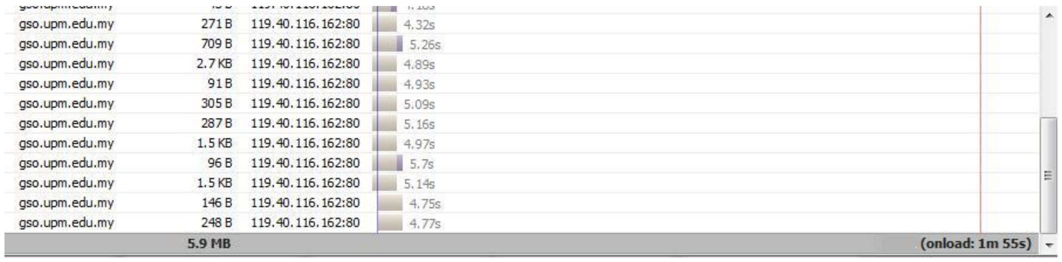
That means Mozilla Firefox has to fetch all the objects of the web site from an origin source for loading for the first time; however, it fetches all web objects from the browser cache for loading the second time. In other words, web pages put a load on a network and caching reduces the bandwidth requirements.

Secondly, we used Google Chrome to open a frequently modified web site which is *www.bbc.com*. It has been observed that Google Chrome took 24.6 seconds to open the page for the first time as shown in Fig. 3-a.

On the other hand, Google Chrome took 9.66 seconds to open the web page for the second time because it loads some components from the cache as shown in Fig 3-b. That means Google Chrome has to fetch all the objects of the web site from an origin source for loading for the first time; however, it fetches some web objects from the browser cache for the second time loading.

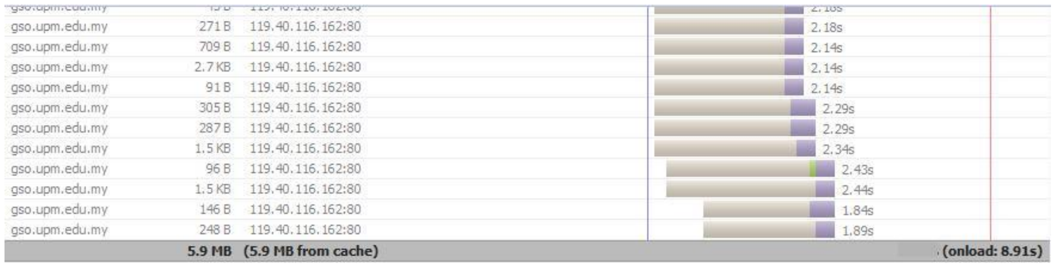
Thirdly, for Windows Internet Explorer, it can be observed that the web page of *www.fsktm.upm.edu.my* that is not frequently modified has been loaded for the first time in 45.38 seconds as shown in Fig. 4 -a.

On the other hand, Windows Internet Explorer took only 3.37 seconds to open the web page for the second time because it loads some components from the cache as shown in Fig. 4-b. That means Windows Internet Explorer fetch all the web items from the origin source for the first time loading, however; it fetches some objects from the browser cache for loading for the second time. Thus, cache



gso.upm.edu.my	271 B	119.40.116.162:80	4.32s
gso.upm.edu.my	709 B	119.40.116.162:80	5.26s
gso.upm.edu.my	2.7 KB	119.40.116.162:80	4.89s
gso.upm.edu.my	91 B	119.40.116.162:80	4.93s
gso.upm.edu.my	305 B	119.40.116.162:80	5.09s
gso.upm.edu.my	287 B	119.40.116.162:80	5.16s
gso.upm.edu.my	1.5 KB	119.40.116.162:80	4.97s
gso.upm.edu.my	96 B	119.40.116.162:80	5.7s
gso.upm.edu.my	1.5 KB	119.40.116.162:80	5.14s
gso.upm.edu.my	146 B	119.40.116.162:80	4.75s
gso.upm.edu.my	248 B	119.40.116.162:80	4.77s
	5.9 MB		(onload: 1m 55s)

Figure 2-a. Mozilla Firefox opening a web page for the first time



gso.upm.edu.my	271 B	119.40.116.162:80	2.18s
gso.upm.edu.my	709 B	119.40.116.162:80	2.14s
gso.upm.edu.my	2.7 KB	119.40.116.162:80	2.14s
gso.upm.edu.my	91 B	119.40.116.162:80	2.14s
gso.upm.edu.my	305 B	119.40.116.162:80	2.29s
gso.upm.edu.my	287 B	119.40.116.162:80	2.29s
gso.upm.edu.my	1.5 KB	119.40.116.162:80	2.34s
gso.upm.edu.my	96 B	119.40.116.162:80	2.43s
gso.upm.edu.my	1.5 KB	119.40.116.162:80	2.44s
gso.upm.edu.my	146 B	119.40.116.162:80	1.84s
gso.upm.edu.my	248 B	119.40.116.162:80	1.89s
	5.9 MB (5.9 MB from cache)		(onload: 8.91s)

Figure 2-b. Mozilla Firefox opening a web page for the second time

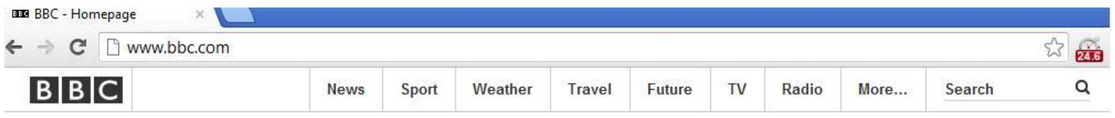


Figure 3-a. Google Chrome opening a web page for the first time

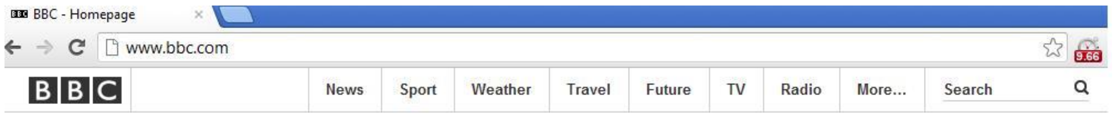
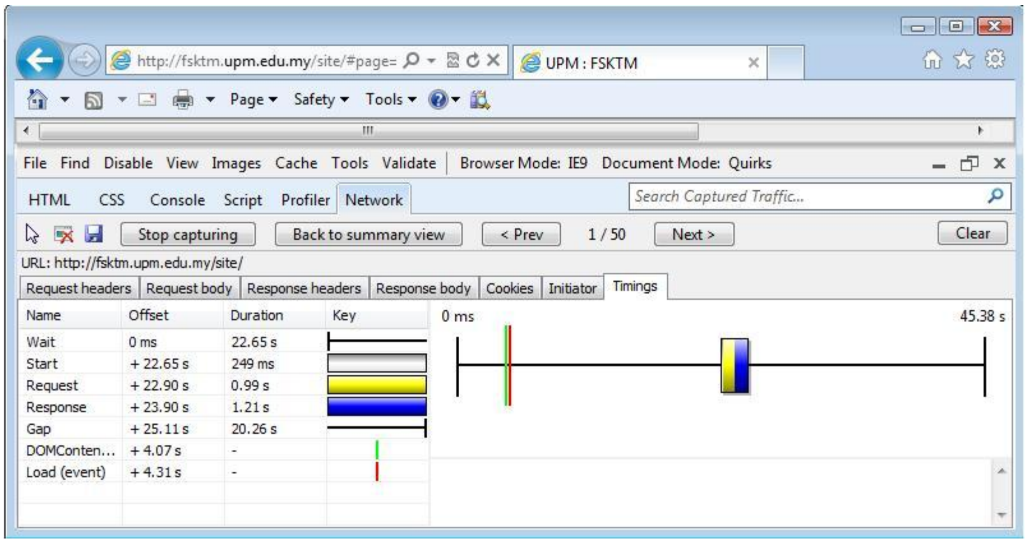


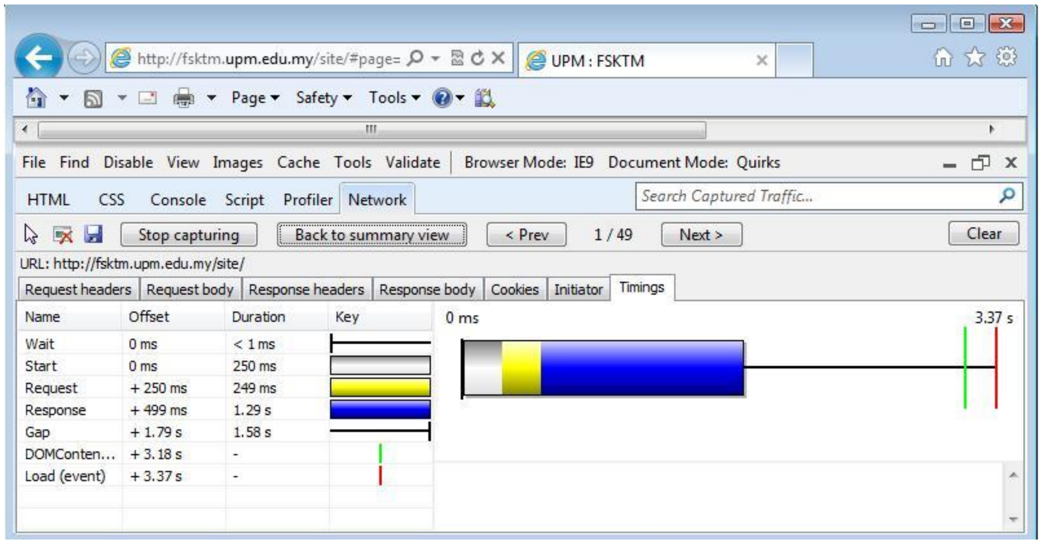
Figure 3-b. Google Chrome opening a web page for the second time


http://fsktm.upm.edu.my/site/#page=
UPM : FSKTM
Page
Safety
Tools
File Find Disable View Images Cache Tools Validate
Browser Mode: IE9 Document Mode: Quirks
HTML CSS Console Script Profiler Network
Search Captured Traffic...
Stop capturing
Back to summary view
< Prev
1 / 50
Next >
Clear
Request headers
Request body
Response headers
Response body
Cookies
Initiator
Timings
0 ms
45.38 s

URL: http://fsktm.upm.edu.my/site/

Name	Offset	Duration
Wait	0 ms	22.65 s
Start	+ 22.65 s	249 ms
Request	+ 22.90 s	0.99 s
Response	+ 23.90 s	1.21 s
Gap	+ 25.11 s	20.26 s
DOMConten...	+ 4.07 s	-
Load (event)	+ 4.31 s	-

Figure 4-a. Windows Internet Explorer opening a web page for the first time



URL: http://fsktm.upm.edu.my/site/

Name	Offset	Duration	Key
Wait	0 ms	< 1 ms	
Start	0 ms	250 ms	
Request	+ 250 ms	249 ms	🟨
Response	+ 499 ms	1.29 s	🟦
Gap	+ 1.79 s	1.58 s	
DOMConten...	+ 3.18 s	-	🟩
Load (event)	+ 3.37 s	-	🟥

Figure 4-b. Windows Internet Explorer opening a web page for the second time

has positive effects on the performance of web items delivery.

5 RELATED WORKS

In this section we report some of the recent researches on utilizing the network bandwidth based on the caching strategy.

In [8], caching has been adopted to reduce the overhead imposed by flooding. First, a principle which is called temporal locality has to be understood that its idea is data which has been recently requested is requested again. The proposed approach is called interest based shortcuts and it relies on the principle of interest based locality where peers are organized logically according to their interests over the existing network. The assumption of interest based locality is a peer, which has a web item that other peers are interested in, may have other web items that are also interested in. Thus if a peer's request exhibits a cache miss, it is forwarded to another peer that has the same interests. It has been concluded that keeping query results in the cache reduces the overhead and network traffic that are imposed without caching.

Static caching algorithm is a way of caching where the most frequently requested items are stored in the cache. This algorithm may work optimally if web items have same sizes. Furthermore, requests are independent and identically distributed. In [1], it has been shown that the static caching algorithm still performs close to the optimal results for large sizes such as media items, when requests are strongly correlated.

In [9], an admission-control technique which is called screening has been proposed to improve proxy caching performance by reducing the delay that is experienced by web users. This technique can be applied in web server and proxy stages. Screening classifies web items into two categories, which are cacheable and noncacheable based on loading times, and it uses LRU algorithm for caching web items. The variation in web item size has a negative impact on the performance of screening, which is considered as a drawback.

In [2], algorithms for implementing a network of nodes' caches have been developed for centralized and decentralized environments in order to address the problem of wasting storage space by storing unnecessary repeated objects in caches without taking into account that these objects have already been cached by neighbouring nodes. An experiment has been performed, and its results show that the implementation of these proposed algorithms improve the performance of a network when nodes taking into account web items that have already been cached in their neighbours' caches.

In [10], a scalable proxy caching algorithm has been proposed in order to minimize channel bandwidth and buffer size of a client. This algorithm is based on the fact which is changing a video frame depends on the relative frame position in the time axis as well as the frame size. An experiment has been performed, and its results show that the performance of the proposed algorithm is better than Selective Caching for QoS networks (SCQ) and Prefix algorithms [11].

In [8], a study on the properties of peer-to-peer live streaming data requests has been conducted, which shows that peer-to-peer live streaming traffic can be answered from the cache. Also, a caching algorithm which is based on sliding window has been proposed which is called SLW algorithm that has been evaluated and compared with traditional caching strategies. An experiment has been performed, and its results demonstrate that SLW algorithm performs close to off-line optimal algorithm which has full knowledge of future requests.

In [12], a pre-fetching caching or sometimes called proactive caching protocol has been proposed to be applied at a client side. This proposed protocol uses a threshold which might be set dynamically in order to optimize pre-fetching gain. Although, pre-fetching systems improve responsiveness for web users, they might increase network traffic load caused by prediction algorithms inaccuracy. Pre-fetching caching protocols might succeed in special cases under certain circumstances. Furthermore, the variation in the size of a web item has negative effects on the performance of this approach.

In [13], the problem of determining the appropriate variant of a web item has been addressed in addition to which variant has to be cached at a particular node in a network. Furthermore, a model for caching web items has been proposed by considering transcoding and transmission costs cooperatively. Also, an algorithm to determine the optimized location has been presented. An experiment has been performed, and its results show that a same web item is treated as a different item. The proposed algorithm is based on symmetric routing paths only; thus, it might not work properly in asymmetric cases.

One of the recent caching strategies has been presented in [14] which proposed a cooperative caching by overlying a peer-to-peer network on client-server network for push-based broadcast. This strategy also has some limitations such as each node has to know the broadcast program. Moreover, signaling overhead is ignorable such as the time and the bandwidth consumed by query and reply messages.

From these related works it can be concluded that caching is considered as a way to improve the delivery of web objects.

6 CONCLUSION AND FUTURE WORK

Network congestion remains one of the main barriers to the continuing success of the Internet. Caching is a way to reduce traffic load on the server and network backbone, which improves the efficiency and scalability of web items delivery. On the other hand, full caching for objects is not a practical solution and leads to consume cache storage in keeping few objects because of its limited capacity. Furthermore, repeated traffic that are being sent to web servers waste network bandwidth. Caching in computer networks might be performed in different stages. In this article, an experiment is carried out to show the affects of caching in browser stage using three different web browsers. This article concluded that caching in browser stage improves the delivery of web items.

Our plan for the next step in this work is to show the effects of caching on the performance in *Proxy stage.*

7 REFERENCES

1. Jelenkovic, P. R. and Radovanovic, A.: Asymptotic optimality of the static frequency caching in the presence of correlated requests. Operations Research Letters 37, 307--311 (2009).

2. Kumar, C.: Performance evaluation for implementations of a network of proxy caches. Decision Support Systems 46, 492--500 (2009).
3. Yasin Waheed, Ibrahim Hamidah, Nor Asila Wati Abdul Hamid, Nur Izura Udzir: A Systematic Review of File Sharing in Mobile Devices Using Peer-To-Peer Systems. Computer and Information Science 4, 28--41 (2011).
4. Abhari, A., Dandamudi, S. P., and Majumdar, S.: Web object-based storage management in proxy caches. Future Generation Computer Systems 22, 16--31 (2006).
5. Karlsson,Magnus: Replica Placement and Request Routing. In: Zhang, Yanchun Tang, Xueyan Xu, Jianliang Chanson, Samuel T. (eds.) Web Content Delivery. Web Content Delivery. Web Information Systems Engineering and Internet Technologies, vol. 2, pp.23--43. Springer, USA (2005).
6. Hala ElAarag, S. R.: A quantitative study of web cache replacement strategies using simulation. Simulation 88, 507--541 (2012).
7. Michael Rabinovich, Zhen Xiao, Amit Aggarwal: Computing on the edge: A platform for replicating internet applications. In: Proc. International Workshop on web Caching and Content Distribution, pp. 55--77, Kluwer Academic Publishers, Massachusetts (2003).
8. Sripanidkulchai, K. and Zhang, H.: Content Location in Peer-to-Peer Systems: Exploiting Locality. In: Zhang, Yanchun Tang, Xueyan Xu, Jianliang Chanson, Samuel T. (eds.) Web Content Delivery. Web Content Delivery. Web Information Systems Engineering and Internet Technologies, vol. 2, pp.73--97. Springer, USA (2005).
9. Kaya, C. C., Zhang, G., Tan, Y., and Mookerjee, V. S.: An admission-control technique for delay reduction in proxy caching. Decision Support Systems 46, 594--603 (2009).
10. Oh, H. R. and Song, H.: Scalable proxy caching algorithm minimizing clients buffer size and channel bandwidth. Journal of Visual Communication and Image Representation 17, 57--71 (2006).
11. Sen, S., Rexford, J., and Towsley, D.: Proxy prefix caching for multimedia streams. In: Proc.The Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 1310--1319, IEEE, New York (1999).
12. Balamash, A., Krunz, M., and Nain, P. Performance analysis of a client-side caching/prefetching system for web traffic. Computer Networks 51, 3673--3692 (2007).
13. Qu, W., Li, K., Kitsuregawa, M., and Nanya, T.: An optimal solution for caching multimedia objects in transcoding proxies. Computer Communications 30, 1802--1810 (2007).
14. Hara, T., Maeda, K., Ishi, Y., Uchida, W., and Nishio, S. Cooperative caching by clients constructing a peer-to-peer network for push-based broadcast. Data and Knowledge Engineering 69, 229--247 (2010).