A More Aggressive Prefetching Scheme for Streaming Media Delivery over the Internet

Junli Yuan*, Qibin Sun, Susanto Rahardja
Institute for Infocomm Research (I^2R), 21 Heng Mui Keng Terrace, Singapore 119613

ABSTRACT

Efficient delivery of streaming media content over the Internet becomes an important area of research as such content is rapidly gaining its popularity. Many research works studied this problem based on the client-proxy-server structure and proposed various mechanisms to address this problem such as proxy caching and prefetching. While the existing techniques can improve the performance of accesses to reused media objects, they are not so effective in reducing the startup delay for first-time accessed objects. In this paper, we try to address this issue by proposing a more aggressive prefetching scheme to reduce the startup delay of first-time accesses. In our proposed scheme, proxy servers aggressively prefetch media objects before they are requested. We make use of servers' knowledge about access patterns to ensure the accuracy of prefetching, and we try to minimize the prefetched data size by prefetching only the initial segments of media objects. Results of trace-driven simulations show that our proposed prefetching scheme can effectively reduce the ratio of delayed requests by up to 38% with very marginal increase in traffic.

Keywords: Streaming media, delivery, startup delay, proxy, prefetching

1. INTRODUCTION

The desire for an on-demand media streaming over the Internet has grown rapidly in recent years, and streaming media is expected to become one of the most popular types of Internet traffic in the future [1]. Due to the large sizes of media objects and the continuous streaming demand of clients, delivering multimedia contents over the Internet is a challenging problem since the Internet is only a best-effort network which does not provide a guaranteed quality of service. To improve the performance of media streaming over the Internet, many researchers have been actively looking into the proxy caching approach, which has been successfully used for improving the performance of the delivery of traditional web content such as HTML and image files. However, because of the distinct characteristics of streaming media, the extension of proxy caching techniques to streaming media applications must be handled in a different fashion than that of traditional web content. For example, to handle the large sizes of media objects, the objects may need to be partitioned into small segments for caching.

To address the problem of media streaming over the Internet, special-purpose content delivery networks (CDN) have been proposed. CDN networks try to help with the delivery of media objects by replicating their content from original servers to dedicated servers which are put geographically close to clients [2] [3] [4]. Although CDN is effective in improving the performance of media content delivery, its implementation and deployment are very expensive. So, researchers are actively looking into other alternative ways such as proxy caching and prefetching etc. Sen et al proposed a prefix caching technique which stores the initial portion of a media object, called the prefix, at a proxy [5]. Chen [6] and Wu [7] etc. generalized the prefix caching paradigm and proposed various segment-based caching algorithms. These algorithms partition a media object into a series of segments and make caching decision for them accordingly. These schemes are effective in reducing the playback delay, especially when the cache size is limited and the media object size is large. Zhang et al proposed another caching scheme which partitions the data of a media object along the rate axis instead of the time axis [8]. This type of partitioning is attractive for variable-bit-rate (VBR) media streaming since only the lower part of a nearly constant rate needs to be delivered across the backbone network during user's access. Caching schemes do not always guarantee the continuous delivery because the portions to be viewed may not be cached in the proxy on time. To further improve the performance, some prefetching schemes are proposed, such as look-ahead

* junli@i2r.a-star.edu.sg; phone +65 6874-8810; fax +65 6774-4998; www.i2r.a-star.edu.sg

Visual Communications and Image Processing 2007, edited by Chang Wen Chen, Dan Schonfeld, Jiebo Luo,
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 6508, 650818, © 2007 SPIE-IS&T · 0277-786X/07/$18

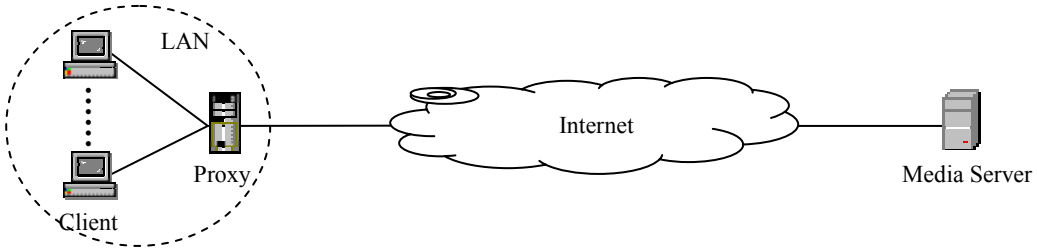

Fig. 1. The client–proxy–server structure of streaming media delivery systems

window based prefetching and active prefetching methods [9], and cumulative prefetching [10] etc. These mechanisms try to avoid the discontinuity of streaming media delivery by prefetching the portions to be viewed while clients are viewing current portions.

Although existing prefetching techniques have shown some effectiveness, they still possess some deficiency. The current prefetching schemes only perform prefetching for the currently accessed object and the prefetching action is only triggered when a client starts to access that object. For a first-time accessed object, its initial portion will not be fetched by both current caching and prefetching schemes. So, clients will still suffer startup delay for the first-time accesses. To address this problem, we propose a more aggressive prefetching scheme which prefetches media objects before they are requested. Note that there is a concern about performing such "beforehand prefetching", which is that it is usually difficult to predict clients' future requests correctly. If the prediction is incorrect and wrong objects are prefetched, some important resources such as network bandwidth and cache space will be wasted. It even becomes worse for media objects because they are usually very large. In traditional web content delivery, Markatos et al proposed a Top-10 approach to prefetching [11], which addressed this concern quite successfully. We would like to apply similar approach to prefetching streaming media in the Internet. Because user access pattern for streaming media objects also follows Zipf-like distribution [1] [12] which is the generally accepted access pattern for traditional web objects, we expect this idea would work well in the context of streaming media delivery. Our work differs from the prefetching schemes for traditional web content delivery in that it concentrates exclusively on streaming media object delivery while other approaches are optimized for traditional web content such as HTML documents and images.

We evaluated the performance of our proposed prefetching scheme by trace-driven simulations. The results show that our scheme can effectively reduce the startup delay of first-time accessed objects while maintain the cost at very low level.

The rest of this paper is organized as follows. In Section 2, we describe the details of the proposed aggressive prefetching scheme. Section 3 gives our evaluation methodology. In Section 4, we present the performance results. Finally, Section 5 concludes the paper.

2. AGGRESSIVE PREFETCHING

In this study, we assume streaming media delivery systems have a three-level structure: client-proxy-server, as shown in Fig. 1, although our scheme also works in multiple-level structured systems. Adopting proxy servers is a common way used by Internet-based media streaming systems to improve their delivery performance. Proxy servers are also widely used in the multimedia content delivery networks (CDN). Therefore, in our study, we also assume the usage of proxy servers. We assume a proxy and the clients behind it are in the same local area network (LAN) where the bandwidth is enough for in time delivery of streaming media. Each proxy knows its clients' access profile. The media server keeps access records, based on which a popular-object-list can be generated. In our new prefetching scheme, the prefetching action takes place at the proxy level with hint from the server, and it is transparent to the clients.

Our aim is to reduce the startup delay for first-time accessed objects by aggressively prefetching them in advance. There are special difficulties when doing such aggressive prefetching, which are related to the accuracy of prediction and the

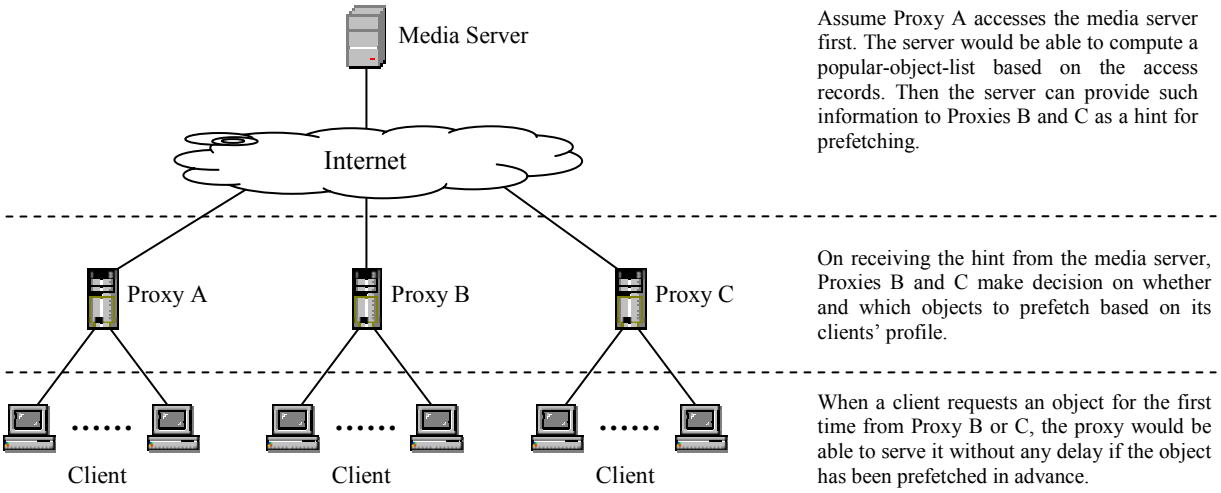

Fig. 2. The operation of server-assisted prefetching

large sizes of media objects. We make use of server's knowledge about access patterns to improve the accuracy of prediction and use partial prefetching to handle the large size problem.

Generally, not all users access a media server at the same time. For example, a media server may receive a lot of requests from users in time-zone A where it is in the daytime, while it may receive almost nil requests from users in time-zone B which is in the night. Although these two groups of users access the media server at different times, their requests to the same server usually share similar pattern which follows Zipf-like distribution as shown in many research works [1] [12]. The Zipf's law indicates that users tend to frequently access a small number of popular objects on a server while other unpopular objects are seldom accessed. A server usually knows what objects are most popular based on its knowledge about recent access records. We want to make use of the server's knowledge to generate hints and use such hints to improve the prefetching performance of proxy servers.

Our proposed prefetching scheme introduces the assistance of the media servers. In the scheme (see Fig. 2), the media server will compute a popular-object-list periodically based on its knowledge of recent access records. Then, when a proxy sends a new request to the server, the server can piggyback the popular-object-list onto the response and sends it back to the proxy. Upon receiving such popular-object-list, the proxy will check the content of the list and the profiles of its clients, and then it may decide to prefetch some objects which are currently not in its cache. For any object to be prefetched, our scheme will prefetch only the initial segment of it under the same consideration as other caching and prefetching schemes, i.e. media objects are usually very large and caching them in full size will soon deplete the space of proxy caches. Furthermore, prefetching operation should have lower priority than normal request. So, the proxy should perform such prefetching only when its workload is not heavy. Later on, new requests will arrive at the proxy. According to Zipf's law, a considerable percentage of the new requests will most likely be requesting those prefetched popular objects. Thus, most of them can be served directly by the proxy within the LAN, so those requests will not suffer startup delay.

3. EVALUATION METHODOLOGY

We conducted trace-driven simulations to evaluate the effectiveness of our proposed prefetching scheme. We generated a synthetic trace using the MediSyn toolset developed by HP Labs [13]. The default parameters of the MediSyn toolset were used, which produced a trace contains 101,464 requests to 1,000 unique media objects. We use the first 25% of the trace as server's history records (i.e. training trace) based on which the server computes a popular-object-list as the hint for prefetching. The rest of the trace (i.e. test trace) is used to run simulations to study the performance of our prefetching scheme. The object sizes in the generated trace are relatively small. We have proportionally increased them to make the overall average size 60 minutes long. The specifics of the trace are summarized in Table 1.

The performance metrics we use in our evaluation are the *ratio of delayed request* and *traffic increase*. The ratio of delayed request is defined as how many requests among the total suffer from startup latency since the initial segments of the requested objects are not cached in the proxy. If the initial segment of an object is under fetching but not finished yet, any requests that demand the same object will be considered as delayed requests. The traffic increase is defined as the increase in traffic due to unsuccessfully prefetched objects. It is measured in the number of objects prefetched but never being used. The traffic incurred by prefetching such objects should be considered as a waste since the prefetched objects are not used to serve users' requests.

Table 1. Trace specifics

Trace type	Synthetic
Trace generator	MediSyn [13]
Training trace size	25,366 requests
Test trace size	76,098 requests
Total unique objects	1,000 unique media objects
Average object size	60 minutes (play time)

As stated in Section 2, media objects are usually very large and it is often inappropriate to prefetch or cache them in full size. So, for any object to be prefetched, we assume only the initial segment of 60-second long will be prefetched or cached. For the caching system, we assume the common LRU caching replacement algorithm is used.

We study the performance of our proposed prefetching scheme under different circumstances by varying the cache size and prefetching size (i.e. the number of objects prefetched). The cache size varies among 6 sizes, which are set to be able to hold segments of 5%, 10%, 15%, 20%, 25% and 30% of the total unique objects, respectively. The size of prefetching varies from 0 to 256, where the size of 0 stands for the normal no-prefetching situation. The parameters for the caching and prefetching systems are summarized in Table 2.

Table 2. Parameters of caching and prefetching systems

Prefetching	Data type prefetched	Initial segments of objects
	Data size prefetched	60-second (play time)
	Prefetching size	Vary among 0, 2, 4, 8, 16, 32, 64, 128, 256 objects
Caching	Cache replacement policy	Normal LRU algorithm
	Cache size	Vary among the following 6 sizes: Sizes of the segments of 5%, 10%, 15%, 20%, 25% and 30% of the total unique objects

Like the normally fetched objects, the prefetched objects will be stored in the common cache space as well. So, prefetched objects will contend with other normally fetched objects for cache space. There are basically two ways to handling such contention according to the priority given to the prefetched objects for being kept in cache:

1) Equal priority

   This method treats prefetched objects the same as normal objects, i.e. prefetched objects are given the same priority as normal objects for being kept in cache. When the cache is full, the prefetched objects in the cache may be evacuated from the cache as any other normal objects would be.

2) Higher priority

   In this method, prefetched objects are given higher priority to stay in the cache than other objects. In other words, the prefetched objects will always be kept in cache as long as the cache space is enough for keeping them. The

rationale behind this idea is that, since the objects prefetched by our scheme are popular objects, the potential of them being requested is high, so they should be kept in cache longer than other objects.

In our study, we consider both of the above situations and study the difference of their performance under our proposed prefetching scheme.

4. EXPERIMENTAL RESULTS

In this section, we report the results of our trace-driven simulations, which show the effectiveness of our proposed aggressive prefetching scheme.

4.1 Ratio of Delayed Request

The ratio of delayed request measures the percentage of the requests among the total that suffer from startup latency because of the initial segments of the requested objects being absent from the cache. Fig. 3 and Fig. 4 plot the ratio of delayed request for the "Equal Priority" and "Higher Priority" situations, respectively. We normalize the results against the situation of "Cache size = 5%, Prefetching size = 0". From Fig. 3, we see that increasing cache size would reduce the ratio of delayed request significantly, while increasing prefetching size only has small impact on the ratio. This indicates

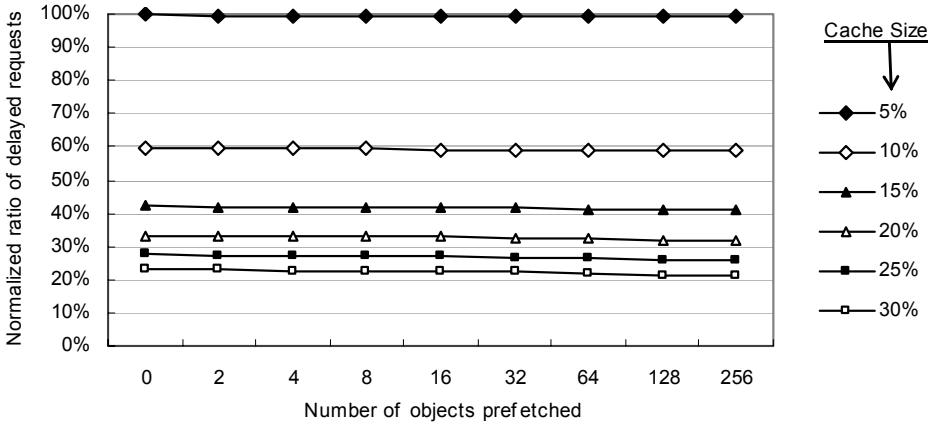

Fig. 3. Ratio of delayed request under the "Equal Priority" situation

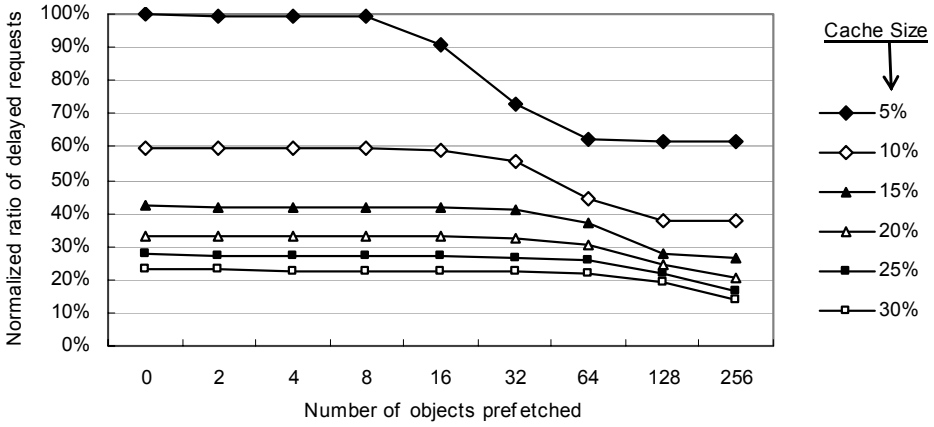

Fig. 4. Ratio of delayed request under the "Higher Priority" situation

that the cache size plays a more important role in reducing the ratio of delayed request than prefetching size does under the "Equal Priority" situation. The reason for it is that, when the prefetched popular objects are given equal priority as other objects, they may be evacuated from the cache before they are actually used to serve users' requests. This trend is more obvious when the cache size is small. As we can see from Fig. 3, when cache size is small, increasing prefetching size barely has any impact on the ratio of delayed request. With larger cache sizes, the impact of increasing prefetching size becomes more noticeable. This is because prefetched objects could stay longer in larger caches, so the chances for them being used to serve users' requests become higher.

From Fig. 4, we see that increasing prefetching size will also have considerable impact on the ratio of delayed request when prefetched objects are given higher priority to stay in the cache. This impact is more significant when cache size is small. For example, when the cache size is set to be able to hold segments of 5% of the total unique objects, increasing prefetching size to 64 and above could reduce the ratio of delayed request by up to 38%. This phenomenon can be explained as follows: The prefetched objects are very popular objects, so the chances for them being requested by users are very high. When such objects are given higher priority to stay in the cache, more user requests could enjoy reduced delay. From another point of view, these results reflect that the cache space will be utilized more efficiently if those prefetched objects are given higher priority to stay in the cache.

4.2 Traffic Increase

If an object is prefetched but never being used, then the traffic incurred by prefetching such object should be considered as a waste. The metrics "traffic increase" is employed to measure the traffic incurred by prefetching such objects. In our study, we use "the number of objects prefetched but never being used" to represent the traffic increase.

Fig. 5 and Fig. 6 show the traffic increase against the prefetching size for the "Equal Priority" and "Higher Priority" situations, respectively. When the prefetching size is small, e.g. $\leqslant$ 16, there is almost no traffic increase for all situations. However, when the prefetching size is greater than 16, traffic increase starts to appear, and the increment grows quickly as the prefetching size increases. This is understandable since the more objects are prefetched, the more of them may never be used before they are evacuated from the cache. This trend is more obvious when cache size is small. When cache size is bigger, the traffic increase becomes smaller as there are more cache space to hold the prefetched objects.

From the graphs, we also see that the traffic increases in the "Equal priority" situation are generally higher than that in the "Higher priority" situation. This indicates that giving the prefetched objects higher priority to stay in cache is beneficial since these objects are popular objects and they will be requested sooner or later. If the prefetched objects are not given higher priority, they may be evacuated from the cache much faster. So, there will be more prefetched objects being evacuated from the cache before they are actually requested. This will not only introduce increase in traffic, but also worsen the ratio of delayed requests, as suggested in Fig. 3 and Fig. 4.

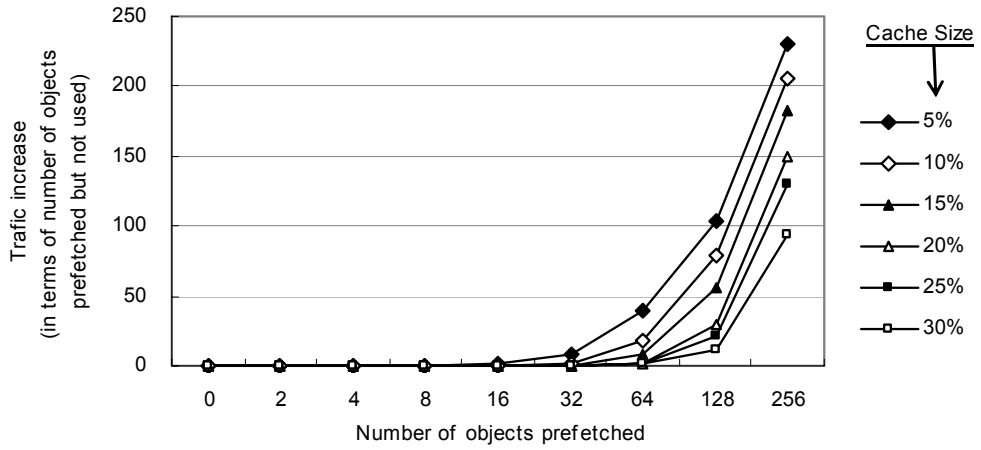

Fig. 5. Traffic increase under the "Equal Priority" situation

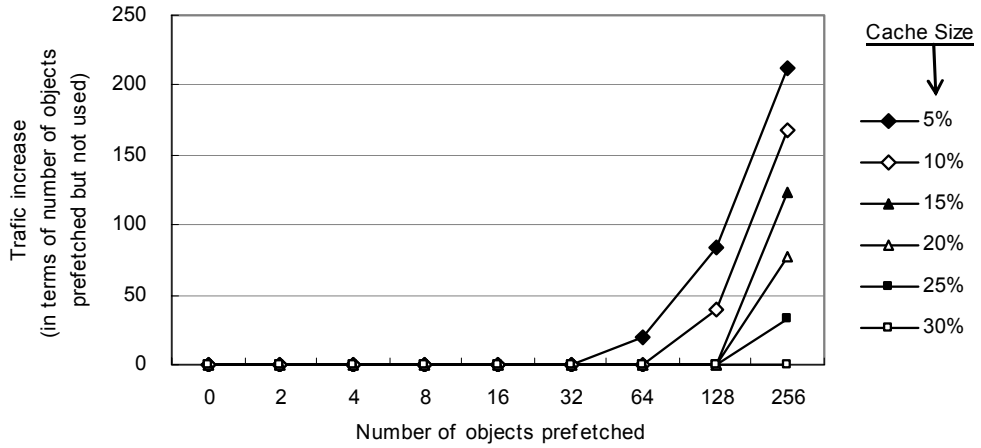

Fig. 6. Traffic increase under the "Higher Priority" situation

5. CONCLUSIONS

In this paper, we proposed a more aggressive prefetching scheme for streaming media delivery. This scheme aims to reduce the startup delay for first-time accessed objects by aggressively prefetching them in advance. To ensure the prefetching accuracy, we introduced the assistance of the media servers by having the servers to locate the most popular media objects and provide such information to proxies as hint for prefetching. To handle the large size problem, we adopted the segment prefetching mechanism. Trace-driven simulation results show that our scheme can effectively reduce the ratio of delayed requests by up to 38% while introduce very marginal increase in traffic.

REFERENCES

1. Xueyan Tang, Fan Zhang, Samuel T. Chanson, "Streaming Media Caching Algorithms for Transcoding Proxies", In the *Proceedings of the 2002 International Conference On Parallel Processing (ICPP 2002)*, Vancouver, British Columbia, Canada, August 18-21, 2002.
2. John G. Apostolopoulos, Tina Wong, Wai-tian Tan, Susie J. Wee, "On Multiple Description Streaming with Content Delivery Networks", In the *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, USA, June 23-27, 2002.
3. Jian Ni and Danny H.K.Tsang, "Large-Scale Cooperative Caching and Application-Level Multicast in Multimedia Content Delivery Networks", *IEEE Communications Magazine*, May 2005.
4. Susie Wee, John Apostolopoulos, Wai-tian Tan, Sumit Roy, "Research and Design Challenges for Mobile Streaming Media Content Delivery Networks (MSM-CDNs)", In the *Proceedings of the 2003 IEEE International Conference on Multimedia and Expo (ICME 2003)*, Baltimore, Maryland, USA., July 6-9, 2003.
5. Subhabrata Sen, Jennifer Rexford, Don Towsley, "Proxy Prefix Caching for Multimedia Streams", In the *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 1999)*, New York, USA., 21-25 March 1999.
6. Songqing Chen, Bo Shen, Susie Wee, and Xiaodong Zhang , "Adaptive and lazy segmentation based proxy caching for streaming media delivery", In the *Proceedings of 13th ACM International Workshop on Network and Operating Systems Support for Design Audio and Video (NOSSDAV'03)*, California, USA, June 1--3, 2003.
7. Kun-Lung Wu, Philip S. Yu, Joel L. Wolf, "Segment-Based Proxy Caching of Multimedia Streams", In the *Proceedings of the Tenth International World Wide Web Conference (WWW)*, Hong Kong Convention and Exhibition Center, Hong Kong, May 1-5, 2001.

8. Zhi-Li Zhang, Yuewei Wang, David H.C. Du, Dongli Su, "Video Staging: A Proxy-Server-Based Approach to End-to-End Video Delivery over Wide-Area Networks", *IEEE/ACM Transactions on Networking*, 8(4):429--442, August 2000.
9. Songqing Chen, Bo Shen, Susie Wee, Xiaodong Zhang, "Streaming Flow Analyses for Prefetching in Segment-based Proxy Caching to Improve Media Delivery Quality", In the *Proceedings of the Eighth International Workshop on Web Content Caching and Distribution (WCW'03)*, IBM T.J. Watson Research Center, Hawthorne, NY USA, 29 September - 1 October 2003.
10. Jaeyeon Jung, Dongman Lee, and Kilnam Chon, "Proactive Web caching with cumulative prefetching for large multimedia data", *Computer Networks*, Vol. 33, pp. 645-655, June 2000.
11. E.P. Markatos and C. Chronaki. "A Top-10 Approach to Prefetching on the Web". In the *Proceedings of the INET 98 Conference*, 1998.
12. Shudong Jin, Azer Bestavros, Arun Iyengar, "Accelerating Internet Streaming Media Delivery Using Network-Aware Partial Caching", In the *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002)*, page 153, Vienna, Austria, July 2-5, 2002.
13. Wenting Tang, Yun Fu, Ludmila Cherkasova, and Amin Vahdat, "MediSyn: A Synthetic Streaming Media Service Workload Generator", In the *Proceedings of 13th ACM International Workshop on Network and Operating Systems Support for Design Audio and Video (NOSSDAV'03)*, California, USA, June 1--3, 2003.