

# A New Technique to Increase the Proxy Caching Performance in VoD Systems

**Abbas Javadtalab**  
DML Laboratory,  
AICTC Research  
center, Sharif  
University of  
Technology, Tehran,  
Iran.  
Email  
[Javadtalab@ce.sharif.edu](mailto:Javadtalab@ce.sharif.edu)

**Hamid Reza Rabiee**  
DML Laboratory,  
AICTC Research  
center, Sharif  
University of  
Technology, Tehran,  
Iran.  
Email  
[rabiee@sharif.edu](mailto:rabiee@sharif.edu)

**Mona Omidyeganeh**  
Multimedia Group,  
Iran  
Telecommunication  
Research Center  
Tehran, Iran.  
Email  
[momid@itrc.ac.ir](mailto:momid@itrc.ac.ir)

**Mohammad Ghanbari**  
Fellow IEEE, Dept.  
of Electronic  
Systems Eng.,  
University of Essex,  
England.  
Email  
[ghan@essex.ac.uk](mailto:ghan@essex.ac.uk)

## Abstract

*Congestion, delay, traffic and server overloading are the main parameters to be considered in video streaming over the Internet that affect the Quality of Service (QoS). With these parameters in mind, we propose a new caching scheme for Video on Demand (VoD) systems that improves the network throughput. To achieve our goal, a novel solution for storing video files in the proxy with scalable video file sizes, is presented. The video file size is constrained such that the quality of the smallest size video will still be subjectively acceptable. Our experimental results shows that the parameters such as throughput, hit ratio and delay are improved by 72%, 14%, and 31%, respectively, compared to the existing proxy caching algorithms.*

## Keywords

Memory space, proxy, caching, video on demand, streaming.

## 1. INTRODUCTION

In recent years, the use of video contents over the Internet in many applications such as video on demand (VoD) is on rapid increase. These applications require certain QoS performance and consume most of the bandwidth. The existing Internet protocols do not guarantee the required QoS. Proxy caching is the most efficient method to reduce bandwidth, delay and to improve network throughput, in VoD applications [1, 2].

Proxy caching was first used in the context of world wide web applications to reduce the bandwidth requirements of downloading html pages. Although some attempts have been made to extend web proxy algorithms to video proxies for multimedia applications, little success has been reported. Perhaps, the main reason for such failure is the fact that, the usual html data and video have different

characteristics and requirements. For instance, video data are delay dependent but the typical web pages have less sensitivity to delay. Moreover, the size of video data is basically much larger than the usual html data files and may require a large size caching space to deliver the desired performance.

Markus et al [2] have suggested that the video clips should be divided into parts before being stored in the cache. Therefore, each part can be replaced in the cache, more efficiently. Furthermore, they have recommended that the proxy should forward client's requests to a neighboring proxy instead of forwarding them to the server. Their algorithm has a major drawback; when a request for the whole video clip has been made, the requested clip will encounter partial hit ratio for parts that may not exist in the proxy. To overcome this problem, the video parts can be waxed as suggested in [1]. In [3], Sen et al have suggested to save only the first few minutes of a video clip in the proxy rather than the whole segments, because most viewers prefer to view the first few minutes of the clip instead of the whole video. In [4], they have introduced a memory replacement algorithm for managing the segments in the cache. In [5], authors have offered a model to predict the request time in order to enhance the hit ratio by estimating the next request time. Liu et al [6] suggested organizing each request in a class by a bandwidth priority parameter. Then the amount of video prefix size can be determined by this parameter. The variable bit rate video streaming has been suggested by Stockhammer et al [7] for variable bandwidth requirements in wireless applications. They claim that this technique can offer a better QoS within the achievable bandwidth. In [8], they have proposed to divided the video clips into two layers; The cache layer divides the incoming video stream into small segments, and the fetch layer organizes the outgoing

requests according to the priority that it assigns to them. Finally, Liu et al have suggested an algorithm that increases the bandwidth but decreases the latency [9]. In all of the previous works, a video is assumed to be a very large size data file. Hence, they have tried to increase the proxy performance by offering various proxy management techniques. However, the amount of caching space is limited in the real world. Consequently, a proxy cannot be able to save all the clips in the cache and therefore it will remove some of them in order to accept a new request.

In this paper, we have introduced a new algorithm to reduce the required space for partitions of video clips in order to serve more new requests without removing any video clip from the server. The key point in this technique is in managing video clips in the cache by exploiting the scalability property of video contents in a layered fashion. Based on experimental results, the new solution not only saves bandwidth and reduces the delay, but also enhances the system throughput. The rest of the paper is organized as follows. The proposed algorithm is introduced in section 2. The simulation results are provided in section 3. Finally, the concluding remarks are presented in section 4.

## 2. THE PROPOSED ALGORITHM

A video proxy caching system is a server that acts as an intermediary between clients and the original video server to ensure lower latency and network traffic. Basically, the cost of the link between a proxy and a client is lower than the cost of the link between the proxy and the server [5]. In general, minimizing the delay is the goal of designing a real time caching system. The second issue is reducing the required bandwidth between the server and the proxy. The following points must be considered in the design of efficient proxy caching systems. First, using an efficient algorithm in order to decrease the miss-ratio. Second, arranging a suitable management algorithm for video clips that is queued in the proxy, and finally, using video characteristics to enhance the utilization of the caching system such that more clips can be stored in the proxy. Most of the previous works only consider the first and second requirements for the design of an efficient and real time algorithm. In this paper, we have tried to exploit the scalability of compressed video signals as well.

To achieve this goal, we have used the concept of layered coding that tries to create various versions of a video clip with different qualities for different bandwidths by adding video enhancement layers to a minimum quality base layer. Therefore, in layered scheme each clip is constructed by using a base layer and one or more enhancement layers. The base layer includes the required information for displaying clips at a reasonable quality. Hence, the clients will be able

to view a low quality clip when they only receive the base layer. The enhancement layers contain information that may improve the video quality when they are added to the base layer. The Enhancement layers must be used with the base layer and they can not be used alone. These ideas have already been supported in the codec standards such as MPEG4 and H.263++. By using the layered coding idea, a clip can be produced with different qualities from a single base layer. Because clients have different connection bandwidths, they can receive the best possible video quality corresponding to their available bandwidth.

The proposed algorithm utilizes the concept of layered coding based on the resolution (size) of the video clips. Here, each clip has a pre-specified quality (depending on its size) according to the class in which it is assigned to. Essentially, there is no limit in the number of the classes, but in this work, the number of classes is confined to three (Table 1). The first class called 'Class A' refers to clips that are highly on demand. Each clip in this class has a base layer and two enhancement layers. The second class called 'Class B' refers to a class with an average demand, with a base layer and one enhancement layer. Finally, the third class named 'Class C', corresponds to clips that are low in demand with a base layer only.

**Table 1. The class labels for video clips**

Type	Base layer	Enhancement Layer1	Enhancement Layer 2
Class A	yes	Yes	Yes
Class B	Yes	Yes	No
Class C	Yes	No	No

The clips are organized into these classes, by their queuing order. When one clip is requested, its priority is increased depending on its class label. In addition, when a clip is received by the proxy, it is positioned on the top of the queue as a class A clip and the clip located at the end of the queue is selected for removal. In each class, the first clip in the queue can be upgraded to the next higher class by receiving an enhancement layer. Furthermore, each clip in the rear of the queue will be moved to a lower class by removing the enhancement layer.

Although concepts in layered coding are the key point in this algorithm, however, in order to manage clip delivery efficiently, one should obtain the best trade-off between the required memory space and the clip subjective quality. In the proposed technique a modified version of the LRU [4] algorithm with the following changes have been used.

- In each class, organize the clips by using the LRU algorithm.

- When a request is received and the clip is available, increase the clip's priority in the queue by  $k_i$ ,  $i \in 1, 2, 3$ , (where  $k_i$  is the priority factor for class  $i$ , and is determined by the administrative policy).
- If the clip exists in the proxy, decide based on the class of the clip:
  - If the clip belongs to class C, reply to the request and increase (upgrade) the clip priority to  $k_3$ .
  - If the clip is on top of the queue within a class (or by adding  $k_3$ ) move it to the top of the queue, then forward the request for the second enhancement layer to the server.
  - When the enhancement layer is received from the server, save it in the proxy and send a copy of it to the client.
  - Check the memory space for saving the enhancement layer in the proxy.
  - If the memory space is enough to save the enhancement layer, then save it in the proxy and add it to the clips. Otherwise, downgrade the last clip of the class A to class B by removing the second enhancement layer.
  - If the required space is available, terminate the conversion, otherwise perform the conversion action for the last clip of class A as well.
  - If the queue of class A becomes empty, continue the conversion by the last clip of class B. If the memory constraint still exists, and class B becomes empty too, continue the process in class C (by removing the last clip of class C).
  - Finally, if the problem still exists, return an error message (the probability of occurrence of this case is negligible since proxies are normally equipped with large storage capacity to store clips).
- On the other hand, when the clip does not exist in the proxy, forward the request to the server.
  - If the server does not have the requested clip, return an error message.
  - Otherwise, upon receiving the clip from the server, save it in the proxy and also send to the client, simultaneously.
  - Repeat the memory management similar to the previous stage (the algorithm tries to avoid removing the clip from the proxy if possible).

In addition, the system always reserves some free space to answer unexpected requests before the memory management algorithm has freed the

required space. Our algorithm can be extended similar to the prefix idea [3]. The pseudo code of the proposed algorithm is shown in the following paragraph.

```

while new request is receive in the proxy
{
  if (clip exists in the cache )
  if (clip belongs to class A)
  {
    update priority and Update list and reply the request }
  else if (clip belongs to class B)
  {
    update priority
    if (clip is at the top of the list)
    if (there is enough free space){
      convert to class A and update list
      and reply the request }
    else if(there is not enough free space )
      call space management function}
  else {
    update Priority
    if (clip is at the top of the list )
    if (there is enough free space ){
      convert clip C to class B and
      update list and reply the request }
    else if (there is not enough free space )
      call space management function}
  else if (clip doesn't exist in cache ){
    forward request to the server and
    waite for reply
    if (clip is not found in the server)
      send error
    if (there is enough free space){
      insert clip at the top of the class A and
      reply the request}
    else if no space then
      call space management function
  }
}

space management function:
{
  while not enough space do
  if (class A list is not empty)
    Convert rear list element to class B
    And update list
  else if class B list is not empty then
    Convert rear list element to class C
    update list
  else if class C list is not empty then
    delete rear list element to class C
    update list
  else
    send error
}

```

## 2.1. Mathematical Analysis

Let  $L$  be the size of a clip,  $p_i$  the size of the  $i$ th enhancement layer and  $B$  the size of the base layer, then assuming that the whole parts of a clip are saved in the proxy, the following equation holds for a single clip:

$$L = B + \sum_i p_i \quad (1)$$

We also define the throughput of the system,  $h$ , as the ratio of the amount of memory consumed in the proposed model,  $L_{\text{Proposed}}$ , to that of the base system,  $L_{\text{Base}}$ , (the base system is the system that does not discard its enhancement layer).

$$h = \frac{L_{\text{Base}} - L_{\text{Proposed}}}{L_{\text{Base}}} \cdot 100 = \frac{B + \sum_i p_i - (B + \sum_j p_j)}{B + \sum_i p_i} \cdot 100 \Rightarrow \frac{\sum_i p_i - \sum_j p_j}{B + \sum_i p_i} \cdot 100, \quad i \geq j \quad (2)$$

In this equation, if the proposed model does not remove any enhancement layer (i.e.  $i = j$ ), the system throughput is zero. However, if all the enhancement layers are removed (i.e.  $j = 0$ ) then:

$$\lim h = \frac{\sum_i p_i}{B + \sum_i p_i} \cdot 100 = 1 \quad (3)$$

As reflected in equation (3), the data rate of the enhancement layer has an important role in saving the memory consumption of the proxy. The quality of the clip is another factor that should be considered since memory compaction reduces the quality of the clip. In addition, the amount of feasible memory compaction is bounded in the real world. Therefore, we can form a dynamic trade-off between the quality of the clip and its memory footprint and adaptively change the quality of the clips according to the system load.

### 3. SIMULATION RESULTS

#### 3.1. Configuration

We have used the following configurations throughout the simulation. The simulation system is composed of three parts: the server that has the responsibility of saving and streaming clips, clients and the proxy. Darwin™ streaming server was used at the server side; QuickTime™ player at the client side and the modified proxy (the proposed algorithm).

All simulations are done with the following setup:

1. The MPEG4-FGS coded QuickTime video samples were used with equal durations.
2. The clip sizes were CIF, QCIF and Half QCIF<sup>1</sup> with 910783, 692372, 489372 bytes, respectively.
3. The size of the base layer, first Enhancement and second Enhancement layers were 489372, 203000, 218411 bytes, respectively.

4. 50 clips were considered and the proxy memory space was set equal to one fifth of the total clip size (approximately 10 MB).
5. The requests were randomly generated with uniform and weighted distribution (see the following).

In the uniform distribution case, the request numbers were produced by a uniform distribution random generator. In the weighted distribution case, the request numbers were created with a weighted distribution according to the following rules: first, 10 percent of the clips were requested with a probability of 0.6. Then, 30 percent of the clips were requested with a probability of 0.3. Finally, 60 percent of the clips were requested with a probability of 0.1. This model tries to represent the real word conditions.

All the simulations were carried out on a 100 Megabit/sec local area network (LAN) with a star architecture. The simulations were carried out in four stages with both the base and the proposed model.

#### 3.2. Simulation of the Base and Proposed Models with Uniform Distribution

In the first set of experiments, the uniformly distributed model was used to send requests to the proxy in a sequential manner. The results are shown in Fig. 1. The results have been attained after responding all of the requests for the based and the proposed models Fig1. The equations represent results. In the proposed model all the available resolutions (CIF, QCIF, Half-QCIF) have been used to enhance the quality of service (QOS). The QOS parameters for the base and proposed models are shown in Table 2.

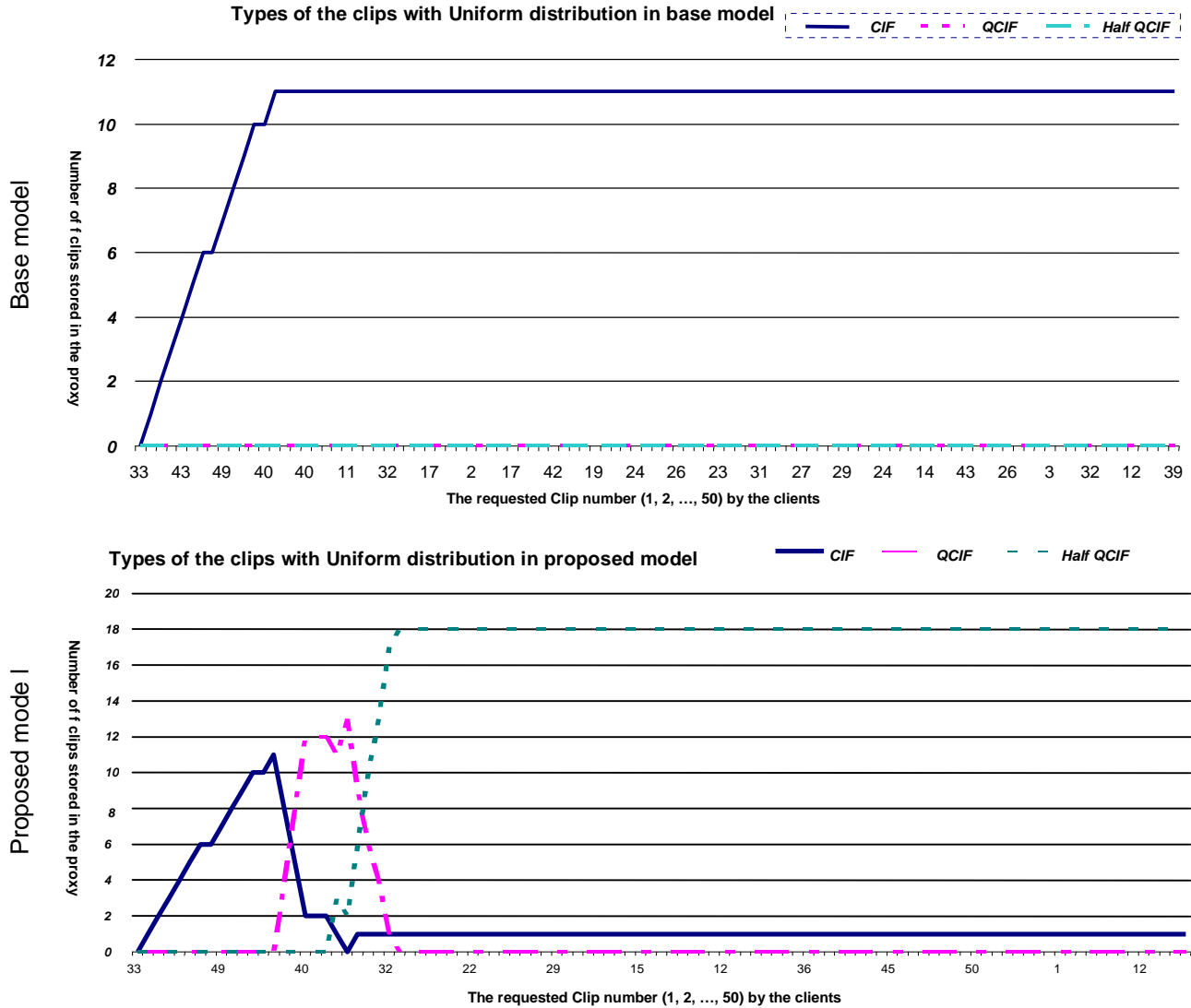
Table 2. QOS parameters in uniformly distributed for base and proposed model

QOS Parameters	Base Model	Proposed Model
Average waiting time	62.51 sec	53.36 sec
Total Server load	74684206 bytes	65191179 bytes
Hit Ratio	18.81	30.61
Throughput	11 clips	19 clips

#### 3.3. Simulation of the Base and Proposed Models with Weighted Distributions

The simulations for the weighted distribution of section 3.1 were carried out in a manner similar to the uniformly distributed case. The results are illustrated in Fig. 2 and the QOS parameters are shown in Table 3.

<sup>1</sup> HalfQCIF =  $\frac{1}{4}$  QCIF



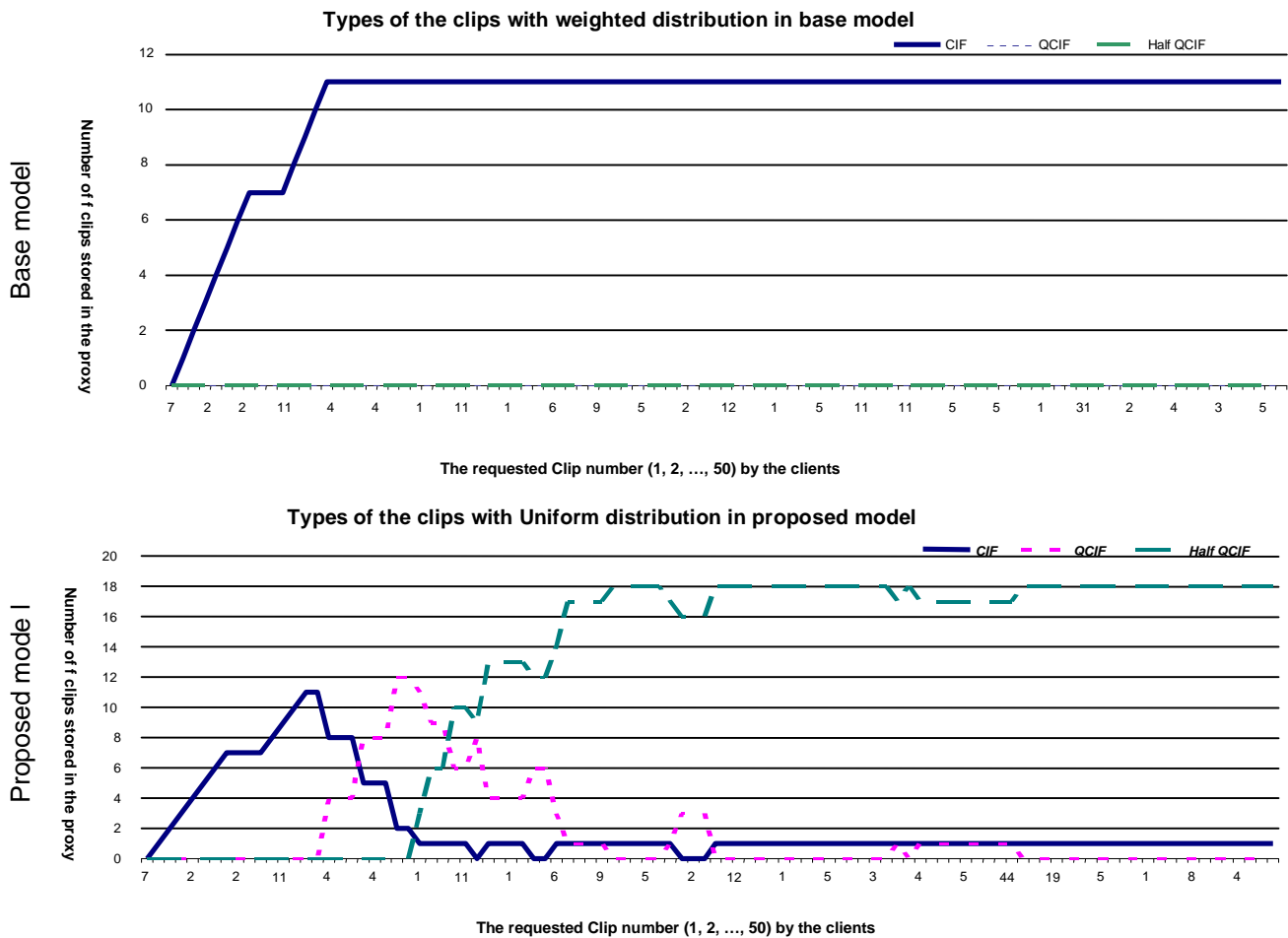
**Fig.1.** The number and the type of clips stored in the proxy for uniformly distributed requests; in the base model (top graph) all of the clips have the same resolution (CIF), in the proposed model all the available resolutions (CIF, QCIF, Half-QCIF) have been used to enhance the quality of service (lower graph).

**Table 3.** QOS parameters in weighted distribution for base and proposed model

QOS Parameters	Base Model	Proposed Model
Average waiting time	37.35 sec	25.92 sec
Total Server load	44628367 bytes	33261217 bytes
Hit Ratio	51.48	66
Throughput	11 clips	19 clips

#### 4. CONCLUSIONS

Throughput, delay, video quality and server loading are among the most important parameters in multimedia networks. By emergence of new multimedia services, the need for proxy cache servers is essential to guarantee QOS in interactive video delivery systems. In this paper, we have presented a new proxy caching algorithm for Video on Demand (VoD) systems that improves the network throughput. To achieve our goal, a novel



**Fig. 2.** The number and type of clips stored in the proxy for requests with weighted distribution; in the base model (top graph) all of the clips have the same resolution (CIF), in the proposed model all the available resolutions (CIF, QCIF, Half-QCIF) have been used to enhance the quality of service (lower graph).

solution for storing video files in the proxy with scalable video file sizes, is presented. Our experimental results showed that compared to the base model, the proposed algorithm not only decreases the delay time but also decreases the required bandwidth up to 25%. In addition, the other network parameters such as throughput, hit ratio and delay are improved by approximately 72%, 14%, and 31%, respectively.

**REFERENCES**

[1] E. Bommaiah, K. Guo, M. Hofmann, and S. Paul, "Design and Implementation of a Caching System for Streaming Media over the Internet," in IEEE Real-Time Technology and Applications Symposium (RTAS), June 2000.

[2] Markus Hofmann, T.S. Eugene Ng, Katherine Guo, Sanjoy Paul, and Hui Zhang, "Caching techniques for streaming multimedia over the internet," Bell Laboratories Technical Memorandum, May 1999.

[3] S. Sen, J. Rexford, and D. Towsely, "Proxy prefix caching for multimedia streams," in IEEE INFOCOM, Mar. 1999, pp. 1310–1319.

[4] T. P. Kelly, Y. M. Chan, S. Jamin, and J. K. MacKie-Mason, "Biased re-placement policies for Web caches: Differential Quality-of-Service and aggregate user value," in 4th Int. Web Caching Workshop, San Diego, CA, Mar. 1999.

[5] Li Zhu, Gang Cheng, Nirwan Ansari, Zafer Sahinoglu, Anthony Vetro, and Huifang Sun, "Proxy Caching for Video on Demand Systems in Multicasting Networks," Conference on Information Sciences and Systems, The Johns Hopkins University, March 12-14, 2003.

[6] J. Liu, X. Chu, and J. Xu. "Proxy Cache Management for Fine-Grained Scalable Video Streaming," in IEEE INFOCOM, Hong Kong, March 2004.

[7] Stockhammer, T. Jenkac, H. Kuhn, G, "Streaming Video over variable Bit Rate Wireless

Channels," in IEEE Transactions on Multimedia, Volume: 6, Issue: 2, PP. 268- 277, April 2004.

- [8] A. Nayyeri, M.R. Hashemi, and N. Yazdani, "A Novel Two Tiered Proxy Caching Scheme for Video on Demand Applications", 10th International Workshop on Web Caching and Content Distribution (WCW), Sophia Antipolis, French Riviera, France, 12-14 September 2005.
- [9] B. Liu, W. Zhang, and S. Yu, "Proxy Caching Based on Segments for Layered Encoded Video Over The Internet," IEEE 6th CAS Symp. on Emerging Technologies: Mobile and Wireless Comm. Shanghai, China, May 31-June 2, 2004.