A Caching and Streaming Framework for Multimedia

Shantanu Paknikar
Wipro Technologies
72, Electronic City,
Bangalore 560001
+91-80-3464851
shantanu.paknikar@wipro.com

Mohan Kankanhalli
School of Computing
National University of Singapore
Kent Ridge,
Singapore
(65) 874-6597
mohan@comp.nus.edu.sg

K.R.Ramakrishnan
Department of Electrical Engineering
Indian Institute of Science, Bangalore
+91-80-3092441
krr@ee.iisc.ernet.in

S.H.Srinivasan
Department of Computer Science
University of California,
San Diego
shs@cs.ucsd.edu

Lek Heng Ngoh
Kent Ridge Digital Labs
21 Heng Mui Keng Terrace
Singapore 119613
lhn@krdl.org.sg

ABSTRACT

In this paper, we explore the convergence of the *caching* and *streaming* technologies for Internet multimedia. The paper describes a design for a streaming and caching architecture to be deployed on broadband networks. The basis of the work is the proposed Internet standard, Real Time Streaming Protocol (RTSP), likely to be the *de-facto* standard for web-based A/V caching and streaming, in the near future. The proxies are all managed by an 'Intelligent Agent' or 'Broker' - this has been designed as an *enhanced RTSP proxy server* that maintains the state information that is so essential in streaming of media data. In addition, all the caching algorithms run on the broker. Having an intelligent agent or broker ensures that the 'simple' caching servers can be easily embedded into the network. However, RTSP does not have the right model for doing broker based streaming/caching architecture. The work reported here is an attempt to contribute towards that end.

Keywords

Caching, Streaming, Proxies, Broker, Layered coding, Replacement Policy, Hit Ratio, Quality Hit Ratio.

1. INTRODUCTION

High-speed local area networks (LANs) are now being widely deployed all over the world. Users on such LANs usually access the Internet (the web) through a proxy server, which also caches, or stores a copy of, popular objects on the local disk(s). The advantages of web caching have been discussed in a number of papers; these have been listed in [1]. If a media file is being retrieved, the download delay can be minimized by means of the *streaming* paradigm, in which *a media file is played out while it is being received over the network.* For an introduction to streaming, refer [1].

ACM Multimedia 2000 Los Angeles CA USA

We envisage the future of the World-Wide-Web as one involving a large number of streaming transfers of A/V content. Most such transfers would take place with the streaming data passing through one or more of the transparent proxy servers caching the streaming data as it passes through. Effective Web caching techniques will be critical for the successful deployment of streaming multimedia services over the World-Wide-Web. This should be obvious, because of the huge latencies involved, and the requirements of real time play out. However, existing web proxy caching systems have been designed for web pages (HTML documents). Such systems need to be modified for retrieval of streaming A/V data.

In this paper, we describe the design of a media caching framework in which the local broadband network has a number of co-operating caching servers for A/V content, all managed by an 'Intelligent Agent' or 'Broker'. We believe that such a system will use the proposed Internet Standard, Real Time Streaming Protocol (RTSP) in place of the Hypertext Transfer Protocol (HTTP), to implement the streaming of the media data. Thus, not only the server and clients, but the caching proxy servers too, will all 'talk' RTSP. Information on RTSP is available in [2]. This paper describes a framework for the working of such a distributed caching and streaming system. Earlier web caching concepts and algorithms have been modified for the proposed framework. Several novel issues have been identified, and new approaches to tackle them have been proposed.

2. PROPOSED ARCHITECTURE[1]

The architecture consists of a high-speed local network, which contains a number of caching proxy servers. One of these functions as the broker or central controlling agent and the others act as sibling caching proxies. The broker is an *enhanced RTSP proxy server*, and performs the standard caching functions as well as handling the streaming issues. All client requests are transparently routed through the broker. The broker maintains the state information that is so important in all the RTSP sessions. Having an intelligent agent or broker ensures that we can embed the 'simple' caching servers into the network. Another new feature of the proposed architecture is that a server

[1] This architecture has been earlier described in [7]. The work reported here is a collaborative effort with the author of [7], and is to be incorporated into that framework.

is allowed to source the content into the caches ahead of any client requests. Thus, the source may not be on-line all the time but its *content* can always be assumed available and hosted in one or more of the caching proxy servers. Besides being used for the obvious purpose of load balancing, the sibling caching proxy servers are individual RTSP proxies in their own right. This enables the broker to transfer control to them efficiently whenever needed. For example, in case of a sibling 'hit', the broker can send an RTSP REDIRECT control command to the client with a pointer to the appropriate caching proxy, which has the clip. The streaming would then take place in an RTSP session from the caching proxy to the client. The sibling proxies do not communicate with each other, it is the broker which manages all the interactions.

As a representative case of caching of multimedia objects, we consider video object caching. We also show how our system can be optimized for the caching of scalably encoded or *layered* video objects. Such objects will have a 'base' layer containing essential information, and one or more 'enhanced' layers containing higher level information. We believe that the framework described in this paper will take advantage of layered coding when the number of layers is between 5 and 10. Such coding schemes have already been proposed [3], [10]. We use RTSP as the basis of our work here as we believe that it is likely to be the de-facto standard for web-based A/V streaming. However, RTSP does not have the right model for doing broker based streaming/caching architecture. Our work is an attempt to contribute towards that end. Additionally, our work is probably applicable to the manipulation of non-live video clips only.

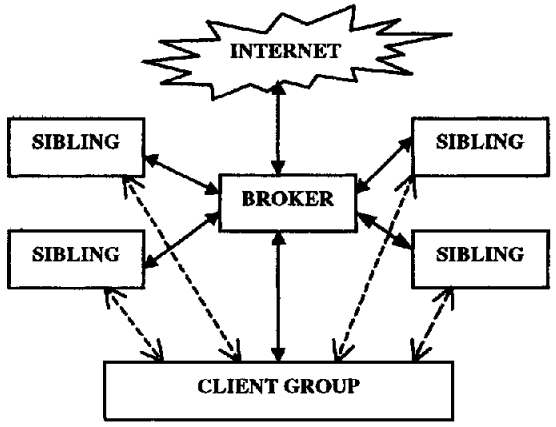


Figure 1. The proposed caching and streaming architecture. The initial interaction is always through the broker. The broker may transfer control to the siblings. The dashed lines indicate that a regular RTSP session can exist between a sibling and a member of the client group.

3. PROPOSALS

3.1 The Auxiliary Cache

As described by the architecture in section 2, the media caching system will be distributed over the broker and it's siblings. The main cache will be a single logical entity physically spread over the disks of the broker and it's siblings. The broker will need to carry out the indexing and retrieval of the information distributed across the proxy servers in an efficient manner. To aid the broker in this task, we use a data structure containing only the *information* related to the cached objects. We call this the 'auxiliary cache', and it is located on the broker (in memory). The idea is adapted from [6]. The auxiliary cache aids the broker in indexing the entries, and allows it to locate the required A/V clip easily. Its use ensures that the broker needs to access the disk caches only for the actual data writing tasks.

3.2 Replacement Policies

The 'weight' of the clip is a measure of its relative importance as compared to the other clips. Thus, higher the weight, lower should be the probability of the clip being replaced. The standard replacement policies are Least Recently Used (LRU) and Least Frequently Used (LFU) policies. In LRU, the weight is inversely proportional to the time since the last access of the object. In LFU, the weight is directly proportional to the number of times the object is accessed. We also have policy based on the size of the objects (SIZE), in which the weight is proportional to the size of the clip raised to some exponent. The value of the exponent determines whether the policy is biased towards smaller or larger objects, as explained in the last portion of this section. In addition, we have a policy called LFU_admin, which is the LFU policy combined with an admission control policy. [Section 3.4] Finally, we experiment with a hybrid policy (HYBRID) proposed in [9], and compare the results with the standard policies. In the hybrid policy, the weight w is computed as

$$w = F^f S^s R^r$$

where F is the number of times the clip is accessed (frequency), S is the size of the clip (size) and R is the time since the last access for the clip (recency). The three exponents f, r, and s are chosen by trial and error. The value for f should be a positive number, meaning that more frequently accessed objects are more likely to be cached. The value of r should be a negative number, such that more recent objects (i.e. those with a smaller value of R) are more likely to be cached. The value of s can be either positive or negative. A positive value would favor caching large objects over small ones. If recency is determined to be more important than frequency, the absolute value of the exponent r should be greater than that of the exponent f.

3.3 New Replacement Policies

The framework proposed takes advantage of clips that are scalably encoded into layers to provide a better Quality-of-Service (QOS) to the clients. The typical number of layers could be between five and ten. Layers higher up (upper enhancement layers) in general, will contain less important information as compared to the lower layers, including the base layer. For the replacement policy, we propose a new *layered* approach: The system caches clip layers as separate objects. These are sorted according to their weights, as computed below:

$$w_l = w(\frac{1}{n})$$

where w_l is the weight of the layer, w is the weight of the clip,

and n is the number of the layer (for the base layer, $n = 1$). The layers are then successively deleted in decreasing order of their weights until enough space is created for the incoming object. Thus, the highest (least significant) layer of the lowest weight object is deleted first. This policy ensures that the caching system is very robust to transients in the workload, because clips will be deleted from the cache gradually rather than at once. This will increase the chances of at least the base layer of clips being present on the caching system. Clips could also be admitted into the cache one layer at a time, however our experiments [5.2] show that the best results are obtained when the entire incoming object is cached.

3.4 Admission Control

The idea of having admission control is that a clip should be cached only provided it can offset the loss of the clip(s) it replaces. This will make the caching scheme less sensitive to transients in the workload. The outcome of the admission control is either positive or negative. We use the algorithm proposed in [6]. This algorithm checks whether the incoming clip has greater weight than the least weight clip already cached, and has size less than the least weight clip. The result is positive if both conditions are satisfied. A detailed explanation may be found in [1].

3.5 A novel Performance Metric

The system uses the standard performance metrics of 'Hit Ratio' and 'Byte Hit Ratio', defined in several papers [1]. In addition, we now propose a new performance metric, which we call the *Quality hit Ratio*.

3.5.1 Quality Hit Ratio

The proposed proxy caching system is designed to handle layered video clips in addition to standard non-layered ones. Therefore, we cannot do with performance metrics as simple as Hit Ratio and Byte Hit Ratio, because they do not reflect on the *quality* of the video clip that is being returned to the client. In general, more the number of layers of a clip present on the cache, better the quality. Whenever a request for an object is received, the broker converts it to a request for the necessary layer(s) and forwards the request to the source. Whenever a new object (layer) comes into the cache, older objects must be purged from it. We have proposed in section 3.3 that the purging of unnecessary objects be done on a layer by layer basis. Thus, at equilibrium, the system is made up of a number of stored objects, which may or may not have all of their layers present. A cache 'hit' for such a system would occur whenever at least one layer of the requested object is present on the cache.

The system performance will then be reflected not only by a high Hit Ratio; but also by the *quality* of the objects being returned to the clients; that is the quality of the hits. A *high quality hit* implies that almost all layers of that object are present, because a larger number of layers ensure better quality for users. Thus, we state that:

"Not only should any such system maximize the probability of a Hit, but it should also simultaneously maximize the probability of the Hit being of as high quality as possible".

We therefore introduce the twin concepts of *quality hit* and *quality hit ratio*. We define the *quality hit* as a number between 0 and 1 indicating how many out of the total number of layers of that object, are present on the cache. A quality hit of 1 implies that all layers of that object are present. Correspondingly, a quality hit of 0 implies a cache 'miss'. (That is, not even the base layer is present).

Ensuring that at least the base layers of the most popular objects are stored on the system maximizes the hit ratio. Now, if as many layers of the cached objects as possible are stored, the quality of these hits is maximized. Finally, we define the *Quality Hit Ratio* as

$$Q_h = \frac{\sum_{i=1}^{H} \frac{n_i}{L}}{H}$$

where H is the total number of hits, L is the total number of layers for each object, and n_i is the number of layers present for object i.

3.6 Simultaneous Request Policy

Standard caching proxies deal with small[2] objects. For any object, the session between the proxy and the client lasts for a very short time. Thus, staggered requests for the same object can be easily dealt with. By staggered requests we mean that a new client requests an object while that is being served to another client from the source. In this case, the proxy can either ask the client to wait until the present object is cached, or start a separate session with the source for the new client.

However, the above approach will be highly inefficient when the proxy is streaming A/V content to the client. This is because each session in this case will last for a much longer time, and the cost associated with setting up a new connection to the source would be quite high. We propose a new approach, which we call the *simultaneous request policy*, for the case of clients requesting the same clip at staggered intervals. In particular, the proposed scheme takes advantage of the bandwidth mismatch between the local network and the external link to give each successive client (after the first) a slightly better Quality of Service. The bandwidth mismatch is such that the bandwidth between the broker and the client B_{bc} is greater than or less than the bandwidth between the broker and the source, B_{sb}. Thus

$B_{bc} = k(B_{sb})$

Where k is any positive real number. For campus networks and corporate intranets, $k > 1$ as the local network is much faster than the external link. In this case, consider the case when a client is being streamed a clip from the external link, through the broker. The bit rate available to the client is limited by the bit rate on the external link, B_{sb}. Let the file size of the clip be denoted by *FILE_SIZE*. Also, the broker caches the clip as it passes through to the client – this is the 'cut-through' caching concept explained in the RTSP draft. Let the portion of the clip cached at time t be denoted by *AVAIL_VIDEO*. Now, if at this time t, another client requests the same clip, it is possible to

[2] 'Standard' web objects include html files, small inline images, etc. which have sizes of the order of a few KB.

serve the clip to him at a speed $k(B_{sb})$. The value for k can be shown to be [1]

$$k < \frac{FILE\_SIZE}{(FILE\_SIZE - AVAIL\_VIDEO)}$$

The second client can thus avail of a higher bit rate and a better QOS.

4. IMPLEMENTATION DETAILS

The caching algorithms have been simulated using C. The simulation is driven by a 'trace file', a log file of an actual proxy cache showing the access pattern of requests for some period. *What we really need is a trace file showing the RTSP requests, however such a trace file is not yet available anywhere.* Instead, we use trace files of HTTP requests taken from two proxy servers in IISc. A justification for using these traces is given in the next section. All the experiments are performed in some 'window size.' This is essentially how far back into the trace files to look. For a particular cache size, after the cache is full, the algorithms will stabilize (in terms of hit ratio or any other performance metric) after some time. For the cache sizes and clip sizes that we have used in our experiments, we have found that a window size of 30,000 to 40,000 requests is enough to study the cache behavior.

4.1 Justification of the traces used

The trace-driven simulation approach is the standard one in web caching experiments. For media clips, trace files of this nature are not yet available anywhere, as a proxy caching system exclusively for streaming of A/V clips has not yet been implemented, to the best of our knowledge. Given this situation, we use HTTP traces as the inputs to our simulation. We believe that to some extent, the HTTP traces will reflect user access patterns for media clips too, provided the local network is a broadband one. A justification for this is presented below.

The broker receives requests for A/V clips from a number of clients. For our problem, these A/V clips will invariably be embedded objects in web pages. Even if they are independent presentations, the user will invariably be 'led' to them through a link from some web page. The justifications are:

- RTSP has been designed for the web and is a proposed *Internet* standard.
- It is likely that web pages of the near future might have tags, which point to an RTSP URL instead of an HTTP URL.
- RTSP will not replace HTTP totally. In fact, they will *coexist*. HTTP servers will continue to serve web pages as always. If the web page has a link to a media resource, the client request for that resource is transferred by the HTTP server to the RTSP (media) server. From that point onwards, the client interacts with the RTSP server.

We therefore believe that an overwhelming majority of requests to RTSP servers will be a result of 'transfer of control' from HTTP servers, and a negligible number of requests will be direct requests. Thus, there will be a 'mapping' of HTTP requests to RTSP requests. Thus, if a web page is very popular, then it is quite likely that a clip(s) to which there is a link on that page is also very popular. This is valid considering the three points mentioned above.

The above means that the request *distributions* for RTSP, that is the user access patterns should be similar to those for HTTP. The differences will be in the mean object size, the standard deviation, and the minimum and maximum object size. In any case, these factors do not influence the performance of the newly proposed layered caching policies [5.2] – they only help determine the maximum cache size necessary for the system to reach equilibrium.

Generating the access patterns has the difficulty that we still do not know what will be the request distribution. In the absence of such data, it appears that using the HTTP patterns is the best option available. An important point is that the only condition for the newly proposed caching algorithms to work well is that the accesses follow a zipf – like distribution[3]. It is well known that HTTP trace files follow a zipf-like distribution [8].

4.2 Analysis of the input trace file

It is important to know the nature of the trace data (the access patterns) used for the simulation. We plot the user access patterns as a rank v/s frequency diagram (the zipf distribution) where the most popular or most frequently accessed document has rank 1. The frequency versus rank plot for the trace file under consideration is shown below.

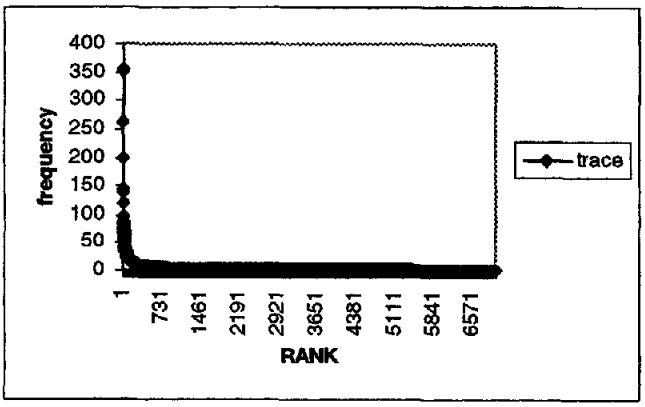


Figure 2. Frequency versus Rank plot for the trace file, indicating object popularity

Similar plots have been obtained for other trace files[4]. The plots confirm the general observations that a few objects are accessed extremely often, while a large number of objects are accessed relatively infrequently.

[3] Video-On-Demand literature such as [11] suggests that the popularity of video objects can be modeled using the well-known *zipf distribution*.

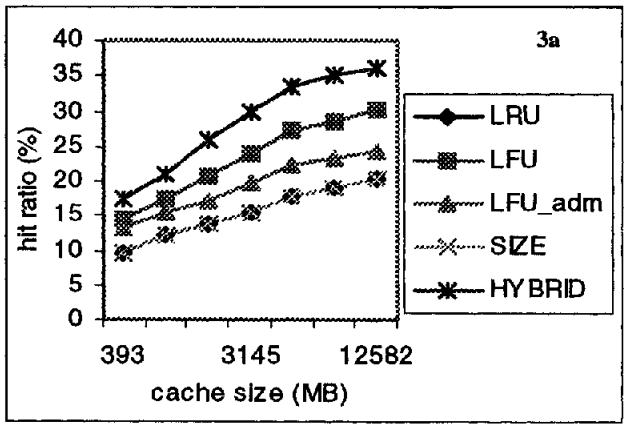
[4] These have not been included because of space constraints.

5. RESULTS

5.1 Modified Replacement Policies

The standard caching policies that have been compared are Least Recently Used (LRU), Least Frequently Used (LFU), Size (SIZE), LFU with admission control (LFU_adm) and a hybrid policy (HYBRID) based on LFU, LRU and SIZE [1], [9]. The comparisons are made on the basis of hit ratio and byte hit ratio. In all cases except LFU_adm, the incoming object is always cached. This is to take into account the recency factor, as the incoming object is the most recent. The results are shown in figure 3.

It is interesting to note that the LFU policy outperforms the LRU policy in all the tests, suggesting that for proxy caching, frequency is of greater significance than recency. Also, the importance of the clip size cannot be discounted. Possibly, the best algorithm would be one that would take into account all three factors. The HYBRID policy serves this purpose, and outperforms the others as shown below.



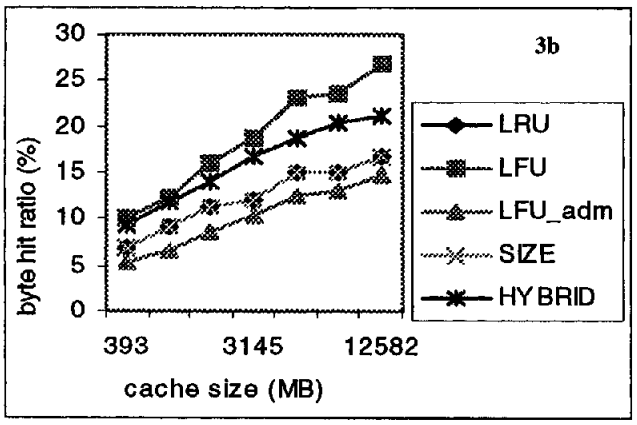


Figure 3. Performance comparison of modified replacement policies with a) hit ratio and b) byte hit ratio plotted against cache size

Our experiments show that the optimum values of the parameters of the HYBRID policy are r = 0, f = 2 and s = -1.5. Thus, the r exponent must be set to 0 for best results. We thus conclude that frequency and size are major factors that determine which objects should be purged from the cache. The incoming object is always cached, and it is here that the temporal locality or recency of requests comes into play. Thus, the object that is admitted into the cache is the incoming one (since there is a high probability that it will be requested again soon). However, the object(s) that is (are) purged from the cache are those with the least weight as determined by the respective replacement policy. Another observation is that byte hit ratio is usually smaller than standard hit ratio. We speculate that this is so because the most profitably cached objects are small ones[5]. We also see that the admission control does not improve the performance. The possible reason is this. The incoming object is the most recent object. If the admission control result is negative, this object is prevented from coming into the cache. Thus, if the traces exhibit the property of recency, then the system performance could deteriorate. Since this is the case in our experiments, it is likely that the traces have some degree of recency.

An important observation from the results is if hit ratio is the performance metric to be maximized, then the HYBRID policy is the best, as shown in the first plot. However, as can be seen from the second plot, the byte-hit ratio is maximized for the LFU policy.

5.2 Layered Replacement Policies

Based on the layered replacement approach mentioned earlier [3.3], we introduce two new layered caching policies: Object-In-Layer-Out (OILO), and Layer-In-Layer-Out (LILO). We compare these with the standard approach used in web caching, which we call Object-In-Object-Out (OIOO). These three policies are then combined with the standard ones [5.1], to give new policies.

Table 1. Layered Replacement Policies

Policy	Incoming Object	Outgoing Object
OIOO	FULL CLIP	FULL CLIP
OILO	FULL CLIP	LAYER
LILO	LAYER	LAYER

In general, whenever the broker in our layered caching system receives a request for a video clip, either a full or partial HIT (or a full or partial MISS) may occur. Here, a 'full' hit implies that all layers are present. In case of a partial HIT, the broker translates the request into one for the 'missing' layer(s) and forwards it up to the source. Now, in a non-layered caching system, the objects stored are complete clips. However, in our type of a layered video caching system, the objects stored by the broker are *layers* of the clips, not the clips themselves.

[5] A frequency versus size plot confirms this for HTTP traces. Whether this is true for RTSP traces remains to be seen.

This point is important because, in the first case, for a given cache size, the system's maximum capacity will be to store some *N* objects. However, in the second case, for the *same* cache size, the system's maximum capacity will be to store ***k*N*** objects, where $(1/k)*clip\_size$ is the size of the base layer. Thus, we would expect that for a layered video caching system, the hit ratio for a given cache size would be higher, but the quality hit ratio would be lower. (A 'non-layered' caching system would have a quality-hit ratio of 1).

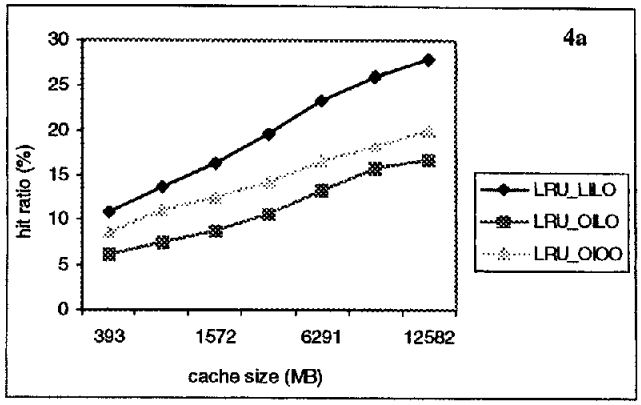
The implications of the new and simple concept of 'Quality hit ratio' that we have proposed, are worthy of notice. Assuming that clips are available in layered form, a simple paradigm shift in the manner of their retrieval, storage, and removal from the caching system results in the following distinct benefits.

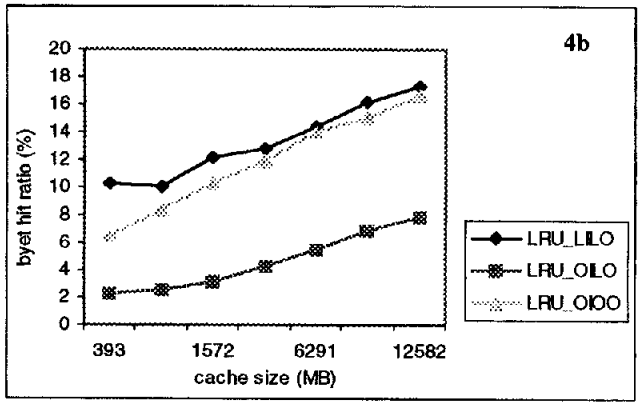
Retrieving one layer at each object request results in an implicit form of admission control. Initially, only one layer is retrieved, the other layers are successively retrieved provided the clip is requested often enough. The layered retrieval increases the probability of providing streaming access to the clip, since the individual layers will require a lower bit-rate as compared with the full clip. Another point to be noted is that retrieving one layer at a time instead of the whole clip results in better load balancing of the external ***Bsb*** link – a larger number of requests can simultaneously handled.

Secondly, the hit ratio goes up. This is because, it is possible to tune the system to result in a Quality hit ratio of close to 100% while showing a significant improvement in hit ratio. Again, this is possible mainly because of the fact that some clips are much more likely to be accessed than other ones. It would then be possible to select a 'popularity' threshold above which clips are cached at full quality (all layers) and below which the quality degrades with decreasing clip popularity. (Fewer and fewer layers of the clip are stored). In our simulated system, no such 'threshold' is selected – the system dynamically purges the least significant layers of the least weight objects until enough space is created for the incoming layer.

The plots in figure 4 show the results obtained with the LRU policy. We see that for the hit ratio and byte hit ratio, the LILO scheme outperforms the other two, while having a mean Quality Hit ratio of 89%. Future work would involve minimizing this tradeoff between lowering of the Quality hit ratio and raising of the hit ratio and byte hit ratio.

The plots in figure 5 show the results for the LFU policy. As far as the hit ratio is concerned, the two layered schemes give a much better performance than the OIOO non-layered one. A comparison of the two layered schemes indicates that the hit ratios are roughly the same for both. However, the byte hit ratio in case of LILO is much more than that in case of OILO. In addition, the Quality hit ratio is much higher in case of OILO than in case of LILO. This suggests that a choice between the two layered schemes can be made in the following way. For the LFU policy, to maximize the byte hit ratio; we must use the LILO policy whereas to maximize the Quality hit ratio we must use the OILO policy.





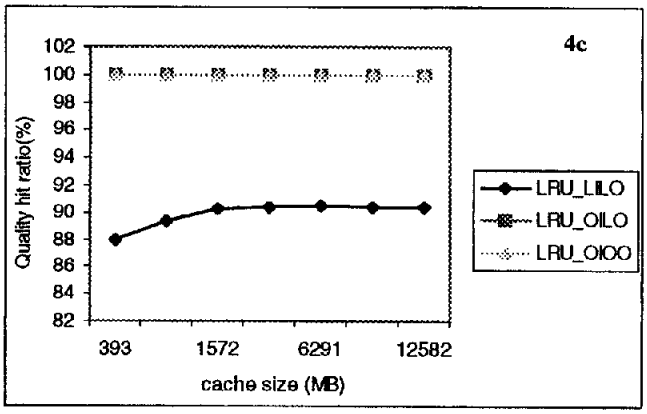
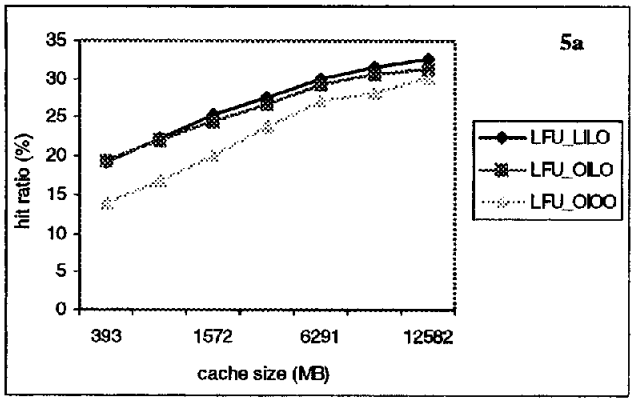


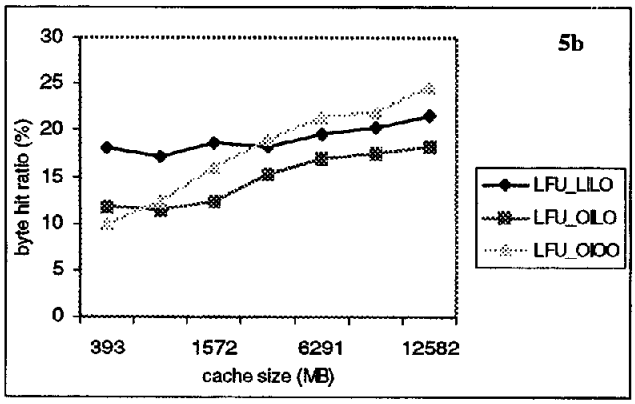
Figure 4. The plots above show the results obtained for the LRU based layered caching schemes, for the metrics hit ratio (4a), byte hit ratio (4b) and quality hit ratio (4c).

The plots in figure 6 show the results obtained for the HYBRID policy. The general behavior is similar to the previous two plots.

Across the four sets of plots, the highest hit ratio is 38.8%, for the HYBRID_LILO scheme. The highest byte hit ratio is 24.6% for the LFU_OIOO scheme. Between the two layered schemes, the highest byte hit ratio is 21.6% for the LFU_LILO scheme.

If only hit ratio was the criterion, then HYBRID_OILO appears to be the best scheme overall, with hit ratios almost the same as LILO and with a Quality hit ratio of almost 100%. However, if byte hit ratio is the criterion, then the LFU_OIOO scheme is the best.





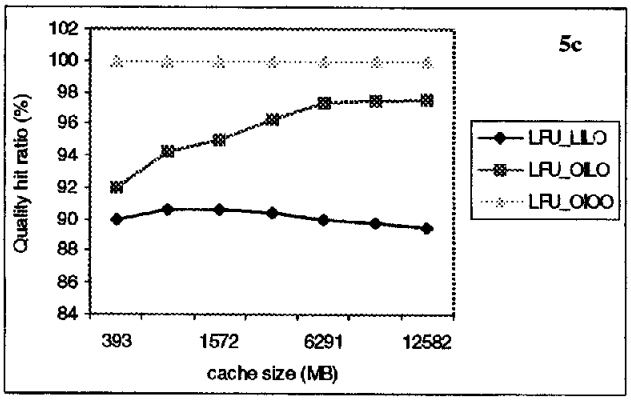
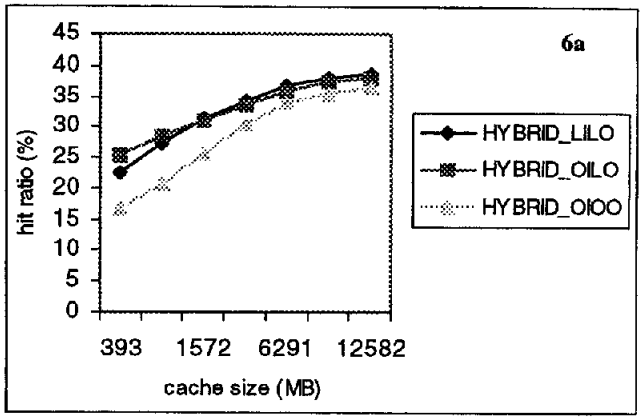
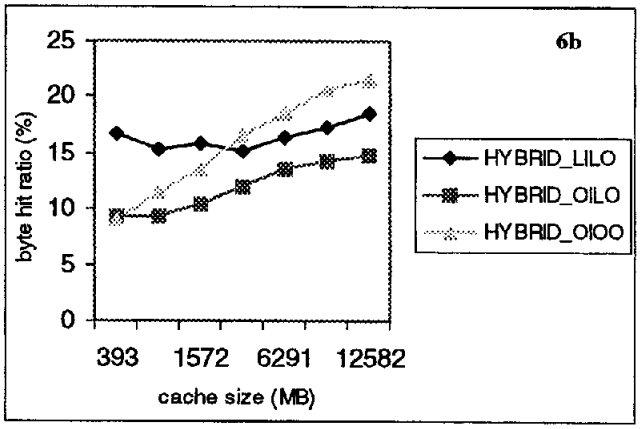


Figure 5. The plots above show the results obtained for the LFU based layered caching schemes, for the metrics hit ratio (5a), byte hit ratio (5b) and quality hit ratio (5c).





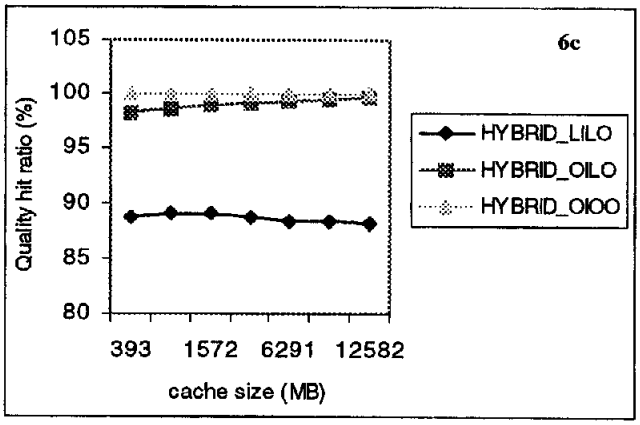


Figure 6. The plots above show the results obtained for the HYBRID based layered caching schemes, for the metrics hit ratio (6a), byte hit ratio (6b) and quality hit ratio (6c).

6. CONCLUSIONS

In this paper, we have proposed a new caching and streaming framework for multimedia objects. The framework has a broker-based architecture and uses the proposed Internet Standard Real Time Streaming Protocol (RTSP). The problem and the area of research itself being a novel one, we hope that the work reported in here proves to be of great value in the near future. It is hoped that the proposed design of a broker based caching and streaming architecture for multimedia will serve as a generic framework. Among the important conclusions drawn through the experiments is that a hybrid caching policy based on frequency, size and recency usually gives the best results. We have also seen that the novel layered replacement approach proposed gives better results than standard object-based replacement schemes. The novel performance metric, Quality Hit Ratio appears to be adequate as a measure for the evaluation of the new layered caching policies.

It appears that caching and streaming are henceforth going to be the major factors influencing the successful deployment of Internet Multimedia systems. However, the convergence of these two technologies is a recent development and has thrown up a number of new challenges. Our work here addresses some of the issues. However, many remain unresolved, and thus open a large number of areas for future work.

The design of the caching and streaming architecture is already in place. The system performance has also been verified through simulation. Therefore, the next logical step in this direction is the implementation of the proposed architecture in an actual network scenario. Also, the integration of the proposed framework into a multicast scenario is extremely important, as multicast is one of the essential technologies of Internet Multimedia. A 'layered multicast' approach needs to be investigated. Stream control issues, particularly related to layered video and RTSP, must be looked into. It is important to add functionality to RTSP to implement 'trick modes' such as FF, REW, etc.

As the Quality hit ratio goes closer and closer to 1, the hit ratio and byte hit ratio drop. This tradeoff between the raising of the Quality hit ratio and lowering of the hit ratio and byte hit ratio needs to be minimized.

Since we are dealing with streaming media, a thorough investigation into QOS related issues, is necessary. This problem involves 3 party QOS negotiation with three cases possible. One, the broker simply 'relays' the requested QOS parameters upward to the source. Two, the broker 'maps' the requested QOS parameters to a new set based on the external resources available. Three, the broker acts as the source itself (this is in case of a cache 'hit') and responds to the request.

Among other issues that require further research, is the generation of the 'sum_clip'. The broker must be able to combine the layers already present with the incoming layer(s) without introducing additional latency. The layered encoding scheme being used will have to provide for this. Secondly, an investigation into the feasibility and advantages, if any, of video interpolation and transcoding schemes is also required. These are possible additional functions of the broker. Thirdly, a performance comparison between a single broker-based architecture and a multiple broker-based architecture is needed. Finally, if possible, a trace file of accesses to a *media server* (audio or video) should be used instead of the HTTP file. This will make the results of the trace driven simulation much more realistic.

7. REFERENCES

[1] S. Paknikar. "A Caching and Streaming Framework for Multimedia". *Masters thesis,* IISc, Bangalore, December 1999.

[2] H. Schulzrinne, et al. "Real Time Streaming Protocol (RTSP)." *Proposed Internet Standard,* RFC 2326, April 1998.

[3] S. McCanne. "Scalable Compression and Transmission of Internet Multicast Video." *Ph.D. Thesis,* University of California, Berkeley, December 1996

[4] M. Abrams, et al. "Caching Proxies: Limitations and Potentials." *Fourth International World Wide Web Conference,* Boston, 1995.

[5] S. Williams, et al. "Removal Policies in Network Caches for World Wide Web Documents." Proceedings of the ACM SIGCOMM, 1996.

[6] Aggarwal, et al. "On Caching Policies for Web Objects." *IBM research report* RC 20619 11/08/96

[7] L.H.Ngoh, et al. "A Multicast Connection Management Solution for Wired and Wireless ATM Networks", *IEEE Communications Magazine,* Vol. 35, No. 11, pp. 52-59, Nov 1997.

[8] P. Cao, et al. "Web Caching and Zipf-like Distributions: Evidence and Implications." *IEEE Infocom,* 1999.

[9] D. Wessels. "Intelligent Caching of World-Wide-Web Objects." *MS Thesis,* University of Colorado, 1995.

[10] A. Swan, S. Mccane and L. Rowe, "Layered Transmission and Caching for the Multicast Session Directory Service", Proc. Sixth ACM Multimedia Conference, Bristol, Sep 1998.

[11] S. Carter and D. long, "Improving Bandwidth Efficiency of Video-On-Demand Servers", Computer Networks 31 (1999) 111-123.