Cooperative Popularity-Aware MPEG-4/MPEG-7 Streaming Proxy System

Ching-Ju Lin, Ying-Hao Ni, Hsien-Ping Suen, and Cheng-Fu Chou

Abstract —*Real-time multimedia services over network greatly arouse users' interests in multimedia objects. To decrease the influence of instable network upon the quality of real-time multimedia services and to reduce the traffic loads of servers, we've implemented an MPEG-4/MPEG-7 streaming proxy system that can reduce client perceived startup latency and efficiently improve the utilization of the storage resources in streaming proxies. By distributing information of cached data between neighboring proxies in the same cluster, our cooperative streaming proxy system can redirect a client's request to another suitable proxy, or fetch-and-relay media objects if all proxies in the same cluster do not cache the requested object. Moreover, a popularity-aware caching policy is designed to make better use of the cache capacity among a cluster of neighboring streaming proxies; consequently, cache hit ratio can be further improved. The performance evaluation of our system is presented to demonstrate the improvements mentioned above*[1].

Index Terms —**Cooperative Streaming Proxies, Popularity-aware Caching Policy, MPEG-4/MPEG-7, RTSP/RTP.**

I. INTRODUCTION

With the rapid growth of multimedia services over network, multimedia objects can be distributed through the network to worldwide users who are interested in them. In recent years, streaming has been widely used when multimedia objects are requested by the clients. Without the support of streaming services, users have to completely download the multimedia objects on their computers before enjoying the objects. By streaming, a user can enjoy a multimedia object during the process of downloading it, which can greatly reduce the perceived startup latency.

However, in streaming systems, network congestion or heavy traffic loads on servers will greatly degrade the quality of multimedia services. In order to provide better streaming services and to reduce the loads of streaming servers, streaming proxies are built up locally to provide streaming services to nearby clients. By setting up proxies in local networks, multimedia objects can be cached in proxies so that the clients can directly get streamed objects from nearby proxies without accessing the faraway servers, which are usually in worse network condition regarding the clients.

Unlike the web proxies, streaming proxies cache multimedia objects that are much larger in size than web objects. The storage capacity of a single proxy may not be sufficient enough to meet a desirable cache hit ratio; therefore, in our work, the key idea is to develop a cooperative mechanism among a cluster of neighboring proxies to efficiently utilize available storage resources in these neighboring streaming proxies. By adopting the concept of storage sharing, a more flexible caching strategy can be applied to a cluster of neighboring proxies; meanwhile, a higher cache hit ratio can be achieved and the loads of streaming servers are reduced as a result.

By efficiently utilizing storage resources among proxies, our *"Cooperative Popularity-Aware MPEG-4/MPEG-7 Streaming Proxy System"* can offer better streaming services to the clients. In this system, proxies located in the same local network form a cluster and one of them is chosen as the cluster head proxy, which periodically decides the cache contents in each member proxy in the cluster based on objects' popularity information collected from all proxies and distributes its decision to all member proxies. With this distributed information, each proxy in the cluster can know other proxies' current cache contents, which enables the redirection functionality. When a client requests a certain object, it first sends the request to its default proxy. The default proxy provides the requested object to the client if a cache hit occurs in the default proxy; otherwise, it will attempt to redirect the client's request to one neighboring proxy that can meet this request. If no proxy in the cluster has a cached copy of the requested object, the default proxy will fetch the object from the server and relay it to the client.

On the other hand, the popularity-aware caching policy proposed in this work can help improve the performance of the proxy system. Note that the popularity of multimedia objects would be greatly different in the real world. If a very popular object is only cached in a single proxy, the traffic loads of this proxy will be extremely heavy and numerous clients who are interested in this object will be forced to share the limited bandwidth on the link of this proxy. Also, the loads of proxies in the cluster will be highly imbalanced. According to the design of our proposed popularity-aware caching policy, the number of cached copies of an object among a cluster of neighboring streaming proxies depends on the object's popularity in the history. Popularity metrics

[1] This work was partially supported by the National Science Council and the Ministry of Education of ROC under the contract No. NSC92-2622-E-002-002 and 89E-FA06-2-4-8.

Ching-Ju Lin, Ying-hao Ni, Hsien-Ping Suen, and Cheng-Fu Chou are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan (emails: {cjlin, yhni, hpsuen, ccf}@cmlab.csie.ntu.edu.tw)

0-7803-8920-4/05/$20.00 © IEEE

of objects will be periodically calculated by the cluster head proxy to react to current preferences of clients. Therefore, popular objects will have more distributed cached copies in the cluster; therefore, the loads of the proxies can be more balanced and a higher cache hit ratio can be realized at the same time. We'll show that our developed system can yield better performance in terms of perceived startup latency and cache hit ratio in the performance evaluation section..

Furthermore, to meet the next-generation, global multimedia standard, our system supports MPEG-4 [3] streaming services and also MPEG-7 [4] multimedia content query functionality. In the following, this paper is organized as follows. Related works of streaming techniques and streaming proxies are shown in section II. Our *Cooperative Popularity-Aware MPEG-4/MPEG-7 Streaming Proxy System* will be described in detail in section III. The performance evaluation of our schemes via simulations is presented in section IV. Finally, conclusions are drawn in section V.

II. Related Works

In recent years, several techniques, such as batching [9] [12], patching [7][16], periodical broadcasting [6][14][17], piggybacking [13][19], and chaining [15][20], have been proposed to improve the utility and the scalability of streaming systems. The common goal of these techniques is to make the content providers serve more requests with less overhead. We'll give a brief literature survey of these techniques in the following.

Using batching techniques, the content providers, instead of sending the requested content immediately, waits for a certain period to receive more prospective requests, and forms a multicast group of the requested content if there are some requests received during the waiting time.

Patching is a technique that can improve the performance of batching. After forming a multicast group, if the content provider receives a request for the same content, the provider may let the new comer join the previously formed multicast group and send an additional patch-stream which composes the prefix part of the content to the new comer.

Similar to batching techniques, a content provider with periodical broadcasting will defer new requests until the next periodical broadcasting is performed. It can reduce the overhead to provide the functionality such as fast forward and fast reverse.

Piggybacking is a technique that can merge two or more individual streams into one. When a content provider with piggybacking is serving two individual streams of the same content, it sends slightly faster to the later one or sends slightly slower to the early one. With proper control, these two streams can be merged into one at some future time.

Chaining is an efficient peer-to-peer scheme that can reduce the load of the content providers. Every receiver in the chaining system may be the content provider of other receivers. When receiving a new request, if the content of the requested object is still in the buffer of some receivers, the original content provider will inform the new comer that the content can be received from another receiver. In this way, the buffered data of receivers can be well utilized.

Another idea borrowed from web cache systems is the streaming proxy servers [10][11]. If clients can retrieve the requested contents from local streaming proxies instead of from the original content provider, the loads of the original provider will be reduced, and the scalability of the service can also be greatly improved. However, the considerations of multimedia streaming proxy systems are far different from those of web proxy systems since the contents cached at the web proxies and streaming proxies are not the same in essence.

Several mechanisms about streaming proxy server systems are proposed, such as prefix caching [8] and prefetching [18]. Multimedia contents are usually much larger than the web contents and can easily eat up proxies' storage space. Prefix caching means the proxies cache the prefix parts of the multimedia contents only. When serving requests, the proxies send the prefix parts of the contents and start to retrieve the remaining parts of the contents from the server at the same time, which is called prefetching. Besides, the cache replacement scheme is also an important issue for proxies. Several widely used web cache replacement schemes, such as LRU (Least Recently Used), and LFU (Least Frequently Used) may not be suitable for multimedia proxy systems. The most important difference between our work and previous works is that the system is formed by multiple cooperative proxies. Meanwhile, the utilization of the proxies' storage resources is enhanced by implementing the popularity-aware storage management. In the following section, we'll present our popularity-aware caching policy, which can raise the cache hit ratio by 60% to 110% when compared with the LRU caching policy in different scenarios.

On the other hand, MPEG-4 is an ISO/IEC standard for multimedia for the fixed and mobile web. Every component in the MPEG-4 media content is treated as an object, which means the interaction between the users and the components in the media content is better than before. MPEG-7 is an ISO/IEC standard for description and search of audio and visual contents. Our streaming system is designed for MPEG-4 multimedia contents and support MPEG-7 query functionality. The ultimate goal of our streaming system is to provide users not only good quality of services but also favorable interfaces for enjoying the multimedia contents.

III. The Streaming Proxy System

Our RTSP/RTP [1][2] streaming proxy system will be elaborated in this section. Fig. 1 depicts our streaming proxy system. Subsection A illustrates the supports for MPEG-4 and MPEG-7 standards in our system, and the cooperation

mechanism between proxies and the popularity-aware caching policy will be discussed in subsection B and C, respectively.

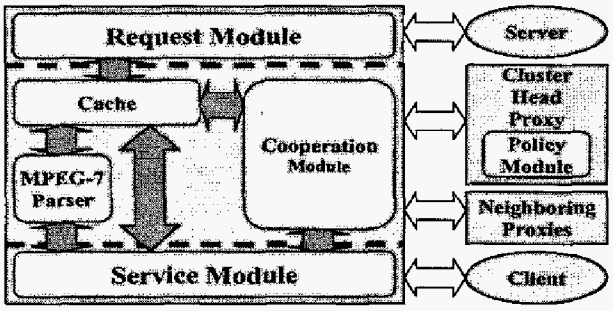


Figure 1. The working modules in our streaming proxies: the internal interactions between these modules and the external interactions with servers, clients, neighboring proxies and the cluster head proxy.

A. MPEG-4 Streaming Services with MPEG-7 Query Functionality

Audiovisual sources will play an increasingly pervasive role in our lives, and there will be growing needs to have these sources further processed. This trend makes it necessary to develop forms of audiovisual information representation that go beyond the simple waveform or sample-based, compression-based (such as MPEG-1 and MPEG-2) or even object-based (such as MPEG-4) representations. MPEG-4 [3] is showing its viability as a next-generation open multimedia standard for content authors, service providers and end users alike; consequently, it's crucial to build a streaming proxy system that supports this object-based standard: synchronized delivery of MEPG-4 streaming is implemented in our system. As shown in Fig. 2, during the streaming service, each streamed object is divided into two separate elementary streams -- audio and video streams, respectively. Besides, the MPEG-7 standard [4], formally named "Multimedia Content Description Interface", offers standardized tools to describe multimedia contents and can form a great couple with MPEG-4. MPEG-7 is not aimed at any one application in particular; rather, the elements that MPEG-7 standardizes support as broad a range of applications as possible. In our system, MPEG-7 multimedia content query functionality is also implemented in our streaming proxies: when sending a request to our streaming proxies, clients can also provide object information to activate an MPEG-7 query. As shown in Fig. 3, when receiving a request with the description of some requested object, the MPEG-7 parser in our proxies will find the multimedia object that matches this description and the streaming data is subsequently transmitted to the client.

B. Cooperative Streaming Services

In our cooperative streaming proxy system, multiple proxies in the same local network are organized as a cooperative cluster, i.e., all proxies in such cluster -- like a

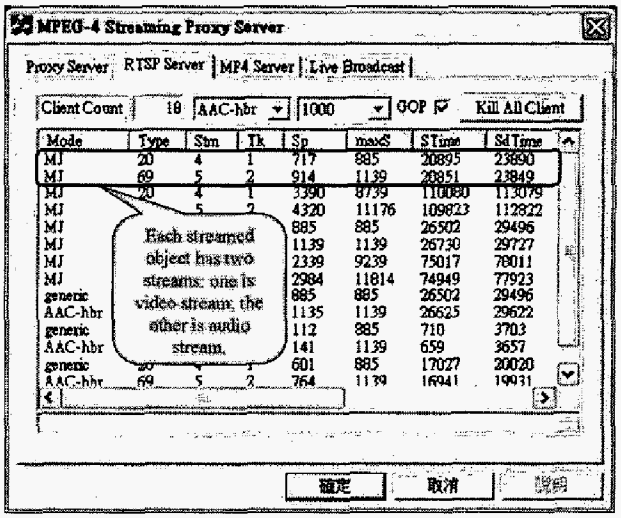


Figure 2. A snapshot of MPEG-4 streaming services.

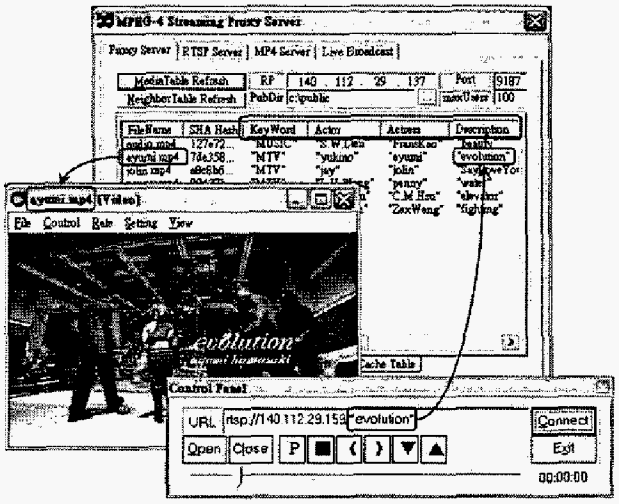


Figure 3. An example of request by using MPEG-7 content description is shown. Information items enclosed by the blue line are MPEG-7 content descriptions of a multimedia object.

clustered local server -- are available for clients to get streaming services. One of the proxies in the cluster is chosen as the cluster head proxy, which periodically collects popularity information of objects from all other member proxies in the cluster and determines the updates for cache contents at all member proxies by its caching policy module (see Fig. 1). The details of how the policy module works will be given in the next subsection. Next, the cluster head proxy will distribute its decision to all member proxies by sending *cache update notification messages*, which includes the cache updates for all proxies and will afterward invoke the distributions of multimedia objects between the proxies and even from the server. By this way, all proxies can be aware of one another's current cache contents and therefore the redirection functionality can be realized.

When a client joins the streaming system, it is requested to select a proxy as its *default proxy*, which can deal with its

requests and exchange control messages with the client. When the client sends a request to its *default proxy*, the client is able to get the streaming service by one of the following three ways: (1) if the *default proxy* had cached the requested object, the client can immediately receives the streaming data from the *default proxy*, (2) if the *default proxy* doesn't have the requested object and if, by the redirection mechanism, the *default proxy* finds another neighboring proxy (*suggested proxy*) that has the object, the client will be asked to redirect its request to this *suggested proxy*, and (3) if the *default proxy* finds that no proxies in the cluster cache the requested object, the *default proxy* will fetch the object from the streaming server by its request module (see Fig. 1) and simultaneously relay the fetched object to the client.

C. Popularity-aware Caching Policy

To enhance the cache hit ratio of our system, the limited storage resources should be carefully managed to cache as many popular objects as possible. The key concept of our caching policy is that the number of cached copies of an object among all proxies in the cluster depends on its popularity. Therefore, when an object becomes more popular than others to a certain degree, clients can more easily get this object because more proxies in the cooperative cluster can provide this object. Meanwhile, the workload of proxies can be balanced, and the clients can also get streaming services sooner. In the following, we illustrate the popularity-aware caching policy adopted among our cooperative streaming proxies.

The caching policy is implemented in the policy module in the cluster head proxy (see Fig. 1). Since users' interests always change over time, the popularities of objects should be periodically evaluated to reflect users' current preferences. During the period interval T, when a proxy receives a request from a client, it updates the number of requests for the object. At the expiration of the period, each proxy will distribute this popularity information to the cluster head proxy, which calculates the *popularity index* (*P.I.*) of all objects. An object's *P.I.* is defined to be the ratio of the accumulated count of requests for this object and the total sum of requests received by proxies in the cluster. Count of requests for every object during all periods within a meaningful history is maintained by the cluster head. After all objects' *P.I.*'s are obtained, the policy module can decide how many cached copies of every object should have in the cluster. Let s represent the total number of proxies in the cluster. In our implementation, the number of cached copies of an object o is given by (1),

Object o's number of cached copies in the cluster

$$= \lceil s/2^n \rceil, \quad if \quad P.I.(o) \geq (1/2^n)\% \quad and \quad P.I.(o) > 0 \quad for \quad n = 0,1,2.... \tag{1}$$

However, not every object can have its designate number of cached copies at the same time because of the limited storage capacity in the cluster. The priority rule is that the caching requirements of objects with higher *P.I.* values should be fulfilled first. If the deletion of an old cached copy is required during the cache update process, the cached copy with the lowest *P.I.* should be replaced first. Once the cluster head proxy finishes arranging the cache updates in all proxies, it sends *cache update notification messages* to its member proxies as mentioned in the previous subsection.

IV. Performance Evaluation

By applying a synthetic workload that obeys Zipf-like distribution [5] in our simulation experiments, we compare the average client perceived startup latency and cache hit ratio between our streaming proxy system and the baseline proxy system in which streaming proxies don't cooperate and the LRU cache policy is adopted.

A. Simulation Setup

In the network topology of our ns2 simulation experiments, there are one server that stores all multimedia objects for providing streaming services to clients and three clusters of proxies that serve three separate groups of clients. Regarding the clients, the values of bandwidth on the links to their default proxies and other neighboring proxies are 10Mbps and 8Mbps, respectively. The data rate of connection between the server and the proxies is 5Mbps. Some nodes are connected to the server and transmit background traffic with data rate 10Mbps to simulate the more congested network condition in core network area. The time interval T for updating the popularity information is set to be 1hr. The random variable x represents interarrival time of two adjacent requests received in a cluster and it is exponentially distributed with mean 36(sec). The sizes of multimedia objects are assumed to be identical in our simulations.

In the following subsections, we consider different scenarios and evaluate the cache hit ratio of our cooperative proxy system and the baseline system with LRU caching policy by varying (1) the total number of objects, (2) the storage size of a single proxy, and (3) the number of proxies in a cluster, which is called *cluster size*. Lastly, the average client perceived startup latency will also be presented to demonstrate that our cooperative proxy system can perform better.

B. The Effect of Total Number of Objects on Cache Hit Ratio

Fig. 4 illustrates the impacts of variation in total number of multimedia objects. There are two kinds of cache hit in our cooperative popularity-aware system: one is cache hit in the default proxy; the other is by redirection. CP-cache hit ratio (default proxy) and CP-cache hit ratio (redirection) represent these two cache hit scenarios respectively; CP-cache hit ratio (total) stands for total cache hit ratio regardless of what kind of cache hit occurs. In this

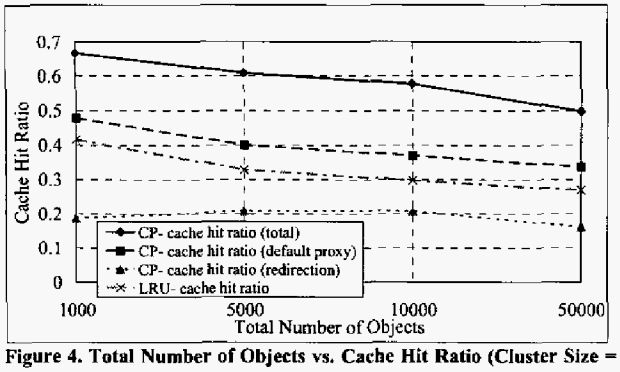


Figure 4. Total Number of Objects vs. Cache Hit Ratio (Cluster Size = 6, Cache Size Per Proxy = 50)

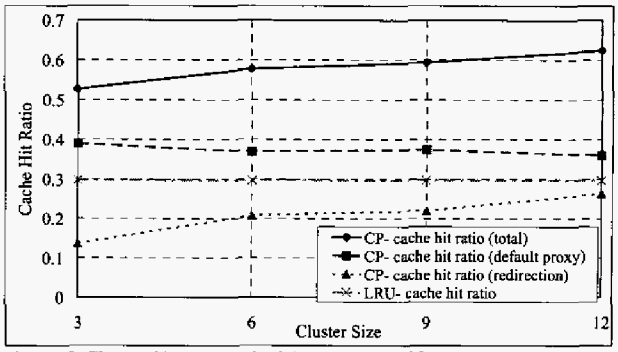


Figure 5. Cluster Size vs. Cache Hit Ratio (Total Number of Objects = 10000, Cache Size Per Proxy = 50 objects

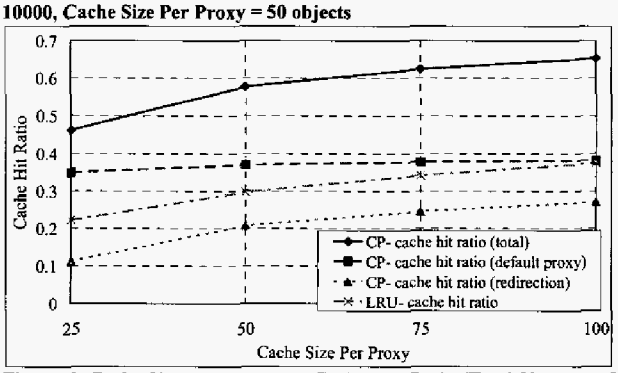


Figure 6. Cache Size Per Proxy vs. Cache Hit Ratio (Total Number of Objects = 10000, Cluster Size = 6)

simulation, cluster size is 6 and cache size per proxy is 50 objects.

With increasing number of objects, the cache hit ratio of both systems decreases since the storage size of the proxies is fixed. However, we can observe from Fig.4 that the CP-cache hit ratio (default proxy) consistently outperforms the LRU system, which means that our popularity-aware caching policy can beat LRU even without the redirection mechanism. Moreover, the total cache hit ratio of our system is much higher than LRU system. The CP-cache hit ratio (redirection) verifies that the cooperative mechanism indeed improves the total cache hit ratio.

C. The Effect of Cluster Size on Cache Hit Ratio

In this simulation, the cache size per proxy is 50 objects and the total number of objects is set to be 10000. The cluster size varies from 3 to 12. Fig. 5 shows that the cache hit ratio in LRU proxy system cannot be raised by increasing the cluster size because there's no cooperation between the LRU proxies. Consequently, for LRU systems, the cache hit ratio keeps constant even when the system deploys additional proxies. As expected, our cooperative proxy system could yield better cache hit ratio as cluster size increases because (1) the system can use popularity-aware cache policy to cache more popular objects in the bigger cluster and (2) it's easier to successfully find one proxy for redirection. Therefore, when the cluster size increases, the total cache hit ratio in our cooperative system is increased as well.

D. The Effect of Storage Size of Single Proxy on Cache Hit Ratio

The impact of cache size of single proxy is evaluated in this simulation. The cluster size is set to be 6 and the total number of objects is 10000. From Fig. 6, the CP-cache hit ratio (default proxy) can perform better than LRU system when the cache size is small, since the cache policy in our cooperative system can always keep the popular objects in the cache. However, the cache hit ratio of LRU system increases when each proxy has bigger storage size for caching multimedia objects. When each proxy can cache 100 objects, the cache hit ratio in default proxies in our system and that of LRU is almost identical. This is because the caching policy is less important when the storage resources are abundant. Generally speaking, our system, which utilizes both popularity-aware cache policy and cooperative mechanism, can improve cache hit ratio by 60% to 110% when compared with LRU system in different scenarios.

TABLE I
COMPARISON OF STARTUP LATENCY AND CACHE HIT RATIO BETWEEN OUR STREAMING PROXY SYSTEM AND BASELINE LRU PROXY SYSTEM

	Our System	Baseline System
Average Client Perceived Startup Latency	3.38SEC	4.93 SEC
Cache Hit Ratio	57.76%	29.74%

E. Average startup latency and hit ratio

The data from TABLE I is from the scenario with 10000 objects, 6 proxies in a cluster and cache size of 50 objects in each proxy. Our system reduces the average client perceived startup latency by about 31.44% and improves cache hit ratio by about 94% when compared with the baseline LRU proxy system, which verifies that our system can provide better streaming services.

V. CONCLUSION

In this paper, we discussed the cooperative popularity-aware scheme among streaming proxies, and the main contributions of our streaming proxy system are as follows.

First, we've implemented a RTSP/RTP streaming proxy system that supports MPEG-4/MPEG-7 audiovisual source standards. Second, we've developed a cooperative mechanism that adopts popularity-aware caching strategy to provide better quality of streaming services. By using the cooperation mechanism, the considerations of caching is not restricted by the size of storage in a single proxy and a more flexible caching policy can be implemented to balance the loads and increase the cache hit ratio. The popularity-aware strategy is proven to be able to cleverly utilize the storage resources in the cluster. As shown in the performance evaluation, our system decreases perceived startup latency and improves cache hit ratio, which leads to better streaming services and justifies the guidelines of our design.

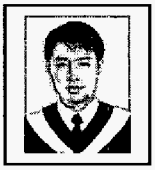
ACKNOWLEDGMENT

We'd like to appreciate Dr. Meng-Jyi Shieh's efforts in building the prototype of the streaming server and clients.

REFERENCES

[1] H. Schulzrinne et. al., "Internet Draft: Real Time Streaming Protocol (RTSP)", Section 10 of RFC2026, February 16, 2004.
[2] H. Schulzrinne et. al., "RTP: A Transport Protocol for Real-Time Applications", RFC3550, July 2003.
[3] MPEG-4 (ISO/IEC 14496).
[4] MPEG-7 (ISO/IEC 15938).
[5] http://www.hpl.hp.com/research/idl/papers/ranking/ranking.html.
[6] T. C. Chiueh and C. H. Lu, "A periodic broadcasting approach to video on-demand service," Oct. 1995.
[7] W. Liao and V. O. K. Li, "The split and merge protocol for interactive video-on-demand," Apr. 1997.
[8] Z. L. Zhang, Y. Wang, D. H. C. Du, and D. Su, "Video staging: a proxy-server-based approach to end-to-end video delivery over wide area networks," Aug. 2000.
[9] W. F. Poon, K. T. Lo, and J. Feng, "Adaptive batching scheme for multicast video-on-demand systems," Mar. 2001.
[10] W.-H. Ma and D. H. C. Du, "Reducing bandwidth requirement for delivering video over wide area networks with proxy server," Dec. 2002.
[11] V. Kahmann and L. Wolf, "A proxy architecture for collaborative media streaming," Dec. 2002.
[12] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on demand video server with batching," in Proc. ACM Multimedia, 1994, pp. 15–23.
[13] L. Golubchik, J. C. S. Lui, and R. R. Muntz, "Reducing i/o demand in video-on-demand storage servers," in Proc. ACM SIGMETRICS Joint Int. Conf. Measurement Modeling of Computer Syst., May 1995, pp. 25–36.
[14] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for meteropolitan video-on-demand system," in Proc. ACM SIGCOMM'97 Conf. Applications, Technologies, Architectures, and rotocols for Computer Communication, Cannes, France, Sep. 1997, pp. 89–100.
[15] S. Sheu, K. A. Hua, and W. Tavanapong, "Chaining: A generalized batching technique for video-on-demand systems," in Proc. Int. Conf. Multimedia Computing Syst. (ICMCS'97), 1997, pp. 110–117.
[16] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand," in Proc. ACM Multimedia, Sep. 1998, pp. 191–200.
[17] J. F. Paris, S. Carter, and D. D. E. Long, "Efficient broadcasting protocols for video on demand," in Proc. 6th Int. Symp. Modeling, Anal. Simulation Computer Telecommun. Syst., Jul. 1998, pp. 127–132.
[18] L. Fan, Q. Jacobson, P. Cao, and W. Lin, "Web prefetching between low-bandwidth clients and proxies: potential and performance," in Proc. 1999 ACM SIGMETRICS Int. Conf. Measurement Modeling of Computer Syst, 1999, pp. 178–187.
[19] N. L. S. Fonseca and R. A. Facanha, "Integrating batching and piggybacking in video servers," in in Proc. of IEEE GLOBECOM'00. vol. 3, 2000, pp. 1334–1338.
[20] T.-C. Su, S.-Y. Huang, C.-L. Chan, and J.-S. Wang, "Optimal chaining and implementation for large scale multimedia streaming," in Proc. IEEE Int. Conf. Multimedia Expo, ICME vol. 1, 2002, pp. 385–388.

Ching-Ju Lin received the B.S. degree from Department of Computer Science, National Tsing-Hua University in 2003. She is currently a Ph.D. student at Graduate Institute of Networking and Multimedia in National Taiwan University and also a member of Network Group in Communication and Multimedia Laboratory. Her research interests include streaming systems and wireless network.



Ying-Hao Ni received the B.S. degree from Department of Computer Science and Information Engineering, National Taiwan University in 2003. He is currently a master student at Department of Computer Science and Information Engineering in National Taiwan University and also a member of Network Group in Communication and Multimedia Laboratory. His research interests include streaming systems and location-aware services.

Hsien-Ping Suen received the B.S. degree from Department of Computer Science, National Tsing-Hua University in 2003. He is currently a master student at Department of Computer Science and Information Engineering in National Taiwan University and also a member of Network Group in Communication and Multimedia Laboratory. His research interests include streaming systems and topology control in ad hoc networks.

Cheng-Fu Chou received the M.S. and Ph.D. degree from University of Maryland, College Park, in 1999 and 2002, respectively. In 2002, he joined the Department Computer Science and Information Engineering, National Taiwan University. His current research interests are in wide-area network applications, distributed multimedia systems, heterogeneous wireless communication systems, and wireless sensor network and their performance evaluation.