

Content Delivery Network with Hot-Video Broadcasting and Peer-to-Peer Approach*

JULIAN LIU¹, SU-CHIU YANG¹, HSIANG-FU YU^{1,2} AND LI-MING TSENG¹

¹*Department of Computer Science & Information Engineering*

²*Computer Center*

National Central University

Chungli, 320 Taiwan

E-mail: yu@dslab.csie.ncu.edu.tw

With the growth of bandwidth, video/audio streaming services have become popular, and are considered a future killer application on the Internet. Many researches proposed several possible technologies for streaming services, such as hot-video broadcasting, caching, content delivery network (CDN), and peer-to-peer communications. In the paper, we propose a new streaming architecture combining the above technologies. We construct our CDN using two-level hashing, which the first level maps user requests to video servers, and in the second level, the selected video servers choose proxy cache servers for the requests. Once receiving the user request, the cache server first checks whether it has the requested video data. If the data are available, the server transmits them to the user over hot-video broadcasting. Otherwise, the server connects to the original video server for the data, which are then broadcast to the user. In addition, the server saves the user information in its hash table. When a user request for the same video arrives, the server can direct the request to the former user with the data, so as to save bandwidth. Finally, we have realized our architecture using Java language.

Keywords: hot-video broadcasting, content delivery network, cache, streaming, peer-to-peer communications

1. INTRODUCTION

With the advancement of broadband networking technology and the increase in processor speed and disk capacity, video-on-demand (VOD) services have become possible. A VOD system is typically implemented by a client-server architecture, and may easily run out of bandwidth because the growth in bandwidth can never keep up with the growth in the number of clients. This results in tremendous demand for computing power and communication bandwidth on the system. To alleviate the stress on the bandwidth and I/O demands, many alternatives have been proposed by sacrificing some VCR functions, also known as near-VOD services. One way is to broadcast popular videos. According to [9], 80% of demand is for a few (10 or 20) very popular videos. Because the server's broadcasting activity is independent of the arrivals of requests, the approach is appropriate for popular or hot videos that may interest many viewers over a certain period of time. One way to broadcast a popular video is to partition the video into segments,

Received March 30, 2004; accepted June 30, 2004

Communicated by Don-Lin Yang.

* The authors would like to thank the National Science Council of the Republic of China for financially supporting this research under Contract No. NSC 92-2213-E-008-004.

which are broadcast on several streams periodically. The schemes share a similar arrangement. A video server divides a video into segments that are simultaneously broadcasted on different data streams. One of these streams transmits the first segment in real time. The other streams transmit the remaining segments according to a schedule predefined by the scheme. When clients want to watch a video, they wait first for the beginning of the first segment on the first stream. Thus, their maximum user waiting time equals the length of the first segment. While the clients start watching the video, their set-top boxes (STB) or computers start downloading enough data from the other streams so they will be able to play the segments of the video in turn.

We can also place a cache server between video servers and clients. The hot videos can be cached so the load on the video servers can be reduced. By caching the prefix of the videos, this approach also significantly decreases the start-up time. An alternative to distribute multimedia content is content delivery networks (CDN), which duplicate video data on replica servers in advance, and then directs user requests to the nearest one according to some decision-making strategies. Such technologies can offer scalable video services. With the fast growth of peer computing, a possible approach is to offer video services over peer-to-peer networks. Without central servers each node can share its resources, and obtain data from peer nodes. In theory, such an architecture avoids the bottleneck caused by central servers, and further provides better scalability and fault tolerance.

In this paper, we propose a CDN, which combines hot-video broadcasting, cache and peer-to-peer communications, as indicated in Fig. 1. Initially, we widely deploy proxy cache servers, which store video segments in advance, and distribute the segments to users via hot-video broadcasting technologies. Meanwhile, caching hot video data reduces the bandwidth requirements and the load on video servers. Due to the broadcasting technologies, for a single video, our system load and bandwidth consumption do not increase with the growth of viewers. Particularly, we apply the content routing technologies

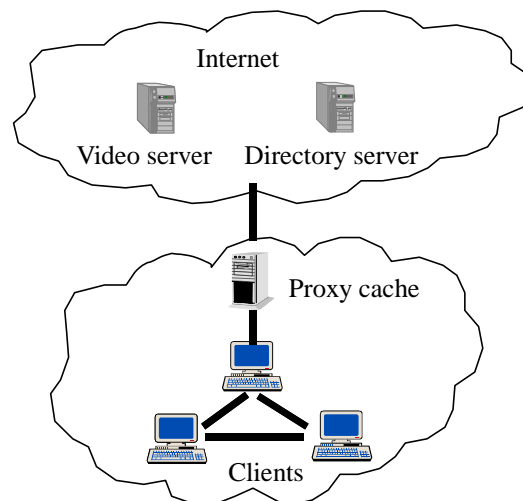


Fig. 1. The three-tier architecture based on hot-video broadcasting.

originally from CDN to address peer viewers, and then exchange their video data. Such an approach can further increase our system capability. By patching over peer, viewers can obtain missed video segments so that start-up time and backbone bandwidth consumption are reduced. Also, multiple data sources could avoid the video playing interruption caused by server/peer failure.

The rest of this paper is organized as follows. Related work about hot-video broadcasting, video cache, and CDN are introduced in section 2. In section 3, we present our system architecture. The realization is described in section 4. Finally, we make brief conclusions.

2. RELATED WORK

2.1 Hot-video Broadcasting Schemes

The simplest broadcasting scheme is staggered broadcasting [8]. The server allocates K streams to transmit a video. The maximum for viewers waiting time is $\frac{L}{K}$, where L is the video length. Pyramid [18] broadcasting partitions a video into increasing size of segments and transmits them on multiple streams of the same bandwidth. This requires less bandwidth than staggered broadcasting under the same maximum waiting time. Fast broadcasting (FB) [13] divides a video into a geometric series of 1, 2, 4, ..., 2^{K-1} . Its maximum waiting time is $\frac{L}{2^{K-1}}$. In comparison with staggered broadcasting and pyramid broadcasting, the FB scheme requires a shorter waiting time. An implementation of the FB scheme on IP networks was reported in [20].

Based on the pagoda broadcasting scheme, the new pagoda broadcasting (NPB) scheme [14] partitions a video into fixed-size segments and maps them into data streams of equal bandwidth at the proper decreasing frequencies. Accordingly, the NPB scheme has a shorter waiting time than the FB scheme. The recursive frequency splitting (RFS) scheme [16] further decreases the NPB scheme's waiting time by using a more complex segment-to-stream mapping. The harmonic broadcasting (HB) [12] scheme first divides a video into several equal segments, and further divides the segments into sub-segments according to the harmonic series. Yang, Juhn, and Tseng [19] proved that the HB scheme requires the minimum bandwidth under the same waiting time. The staircase broadcasting (SB) scheme [11] was proposed to reduce buffer requirements at the client end. The scheme requires a client to buffer only 25% of a playing video.

2.2 Content Delivery Networks (CDN)

CDN is a particular network which is designed for transmitting some kind of contents, such as Web pages, video, or audio. To approach to users as near as possible, such networks deploy replica/cache servers widely. When a user request arrives, the CDN first searches for a server which can transmit data fastest, and then redirects the request to that server. The redirection can decrease the transmission delay, bandwidth usage, and waiting time. There are two possible ways to build a CDN, network and overlay. Network

method requires the devices or nodes (such as routers or switches) on networks to identify data content type and to address user location. In addition, the devices are responsible for request redirection. In this case, the CDN is directly based on the physical level. The way of overlay builds the CDN on current networks. The programs on servers and clients are responsible for the content switching.

Fig. 2 indicates the seven parts of a CDN. The request routing system is responsible for request routing which redirects clients' requests to the best replica servers according to client location and demand content. The server selection is based on three criteria: hub number, round-trip time, and server load. There are three possible redirection methods:

- HTTP redirection. This method is only for Web browsing. A user request first connects to the origin server. The server then redirects the request to a replica server by returning HTTP status code 304.
- DNS indirection. When clients issue domain-name queries, their resolvers will look up the domain server for the corresponding IP address. If multiple IP addresses are matched, the domain server further analyzes the user source IP, and returns the one associated with the nearest replica or cache server.
- Anycasting. In this method, all replica servers use one anycast address which is the destination of all user requests. Then, an anycast-support router is applied to dispatch the requests to the replica servers.

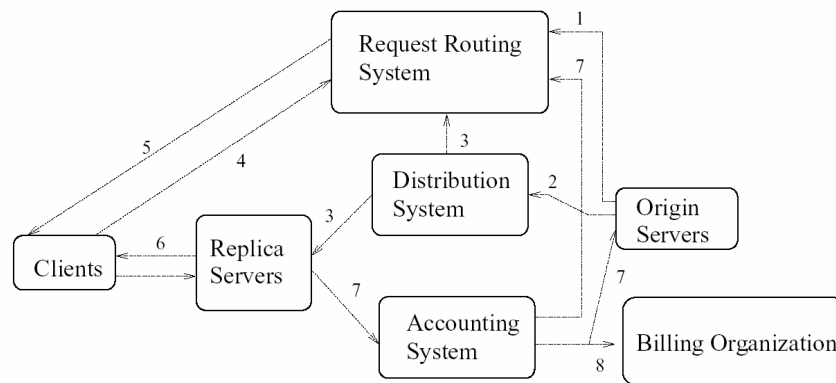


Fig. 2. The operation of a CDN.

The distribution system is responsible for transmitting contents on origin servers to replica servers. It determines which contents are put on which replica servers. In addition, it affects the locations of the replica servers. The factors affect the system load and bandwidth consumption.

2.3 Peer-to-peer Communications

Peer-to-peer communications are considered to be a distributed system without a central server or a fixed topology. Each node executes the same program, offers identical

functions, and shares resources with each other. Such an architecture provides better fault tolerance and load balance.

One important challenge to peer-to-peer communications is locating the requested resource. There are three possible methods – central routing, hierarchical routing, and distributed hash routing. In the first approach, a central server routes all user requests and maintains information about network topology and available resources. Thus, this approach uses peer-to-peer communications at data transmission. The main advantages are low routing cost and easy system construction. However, a single server easily causes a performance bottleneck. Furthermore, if the server goes down, the entire system fails. Systems based on this approach include Napster [1], Audiogalaxy [2], SoulSeek [3].

In hierarchical routing a group of nodes elects a supernode, which handles request routing, and maintain resource information. Furthermore, the supernodes exchange information with each other. The supernode assignment is based on multiple criteria, such as uptime, available bandwidth, and computing power. When the load exceeds the capability of a supernode, the orphan nodes will hold a new election, which generates a new supernode. The main advantage of the approach is a reduction in the numbers of messages for supernode election. Kazaa/Fasttrack [4], WinMX [5], eDonkey [6], Gnutella [7] used this approach.

Distributed hash routing distributes routing the load on all nodes by means of a distributed hash table. For example, suppose there exist n nodes on networks. Each node keeps $\log n$ routing records. When a node receives a user request, the node looks up at most $\log n$ other nodes for the location of the requested resources. Due to the distribution of the routing load, such an approach has good scalability. In comparison with hierarchical routing, the messaging traffic is significantly less. The main drawback is that a requested routing requires $\log n$ look-up steps. Once the distance between two nodes is very large, the routing will take much time.

Due to the advantages of load balance and fault tolerance, peer-to-peer communications can be employed to transmit media streams, thus avoiding the overload on video servers. However, the application still encounters the following challenges. The distance between peer nodes is probably large so that the transmission time is very large. Furthermore, the connection quality between them may be unreliable, and video playing is thus broken. Another challenge is the fast change of node status. In peer-to-peer environments, nodes probably live this time, and die that time. The uncertainty causes difficulty in data transmission. Finally, in comparison with regular servers, a peer node has limited resources. Thus, how to achieve a load balance on each node is an important issue.

3. SYSTEM ARCHITECTURE

Our system consists of the directory service, cache service and video service. The directory service locates the requested video and redirects the request to the nearest cache. Also, it stores the video accesses. When a new request arrives, the service can offer a list of peer nodes which have accessed the same video so that the user can obtain video data accordingly. The cache service includes two parts – proxy cache server and peer cache. Once a request is received, the proxy cache server checks whether the requested video

was cached or not. If it is available, the server directly forwards it to clients by some hot-video broadcasting scheme. Otherwise, the server obtains the video from the original video server. The video is then transmitted to clients, and stored on disk. In the peer-cache, a client that has accessed videos can become a cache, offering video segments for other clients interested in the same videos.

3.1 Directory Service

The directory service is responsible for locating the video server for rapidly incoming requests. The service also finds the nearest cache server for a client to reduce startup latency, transmission time, and bandwidth usage. The last function is to let clients share their video data. When a video request arrives, the directory service searches not only the cache server but also for peer clients, which can provide the requested video data.

One possible approach to the directory service is to use a central server which maintains information about video accesses. The system adopting the approach includes the domain name system (DNS), and LDAP. Another approach uses the distributed hash table, in which each node maintains a part of a hash table. When a node receives a request, it checks its hash table for requested objects. If available, the node returns the object directly. Otherwise, the node forwards the request to the next node. The process repeats until the requested objects are found. A famous technology for this approach is chord [15].

Our system employs both approaches. Suppose there are n video servers, denoted S_1, S_2, \dots, S_n . We also assume there are V videos, which are mapped to n hash tables. Each server thus has V/n videos on average. Using a hash table, a user can locate the requested video rapidly, and the maximum table look-up times are n , regardless of the number of videos.

After locating a video server for an incoming request, we then select the proxy cache server nearest to the client. Suppose there are m proxy cache servers, denoted P_1, P_2, \dots, P_m . We then map all IP addresses to the cache servers. In IP v4, each server thus responds to $2^{32}/m$ nodes. Each client uses his/her address as the key of a particular hash table to match a cache server. Our hash function transforms a client's IP address to a double word format. For example, suppose a client's address is 206.191.158.55. Its hash value is $((((206 * 256) + 191) * 256 + 158) * 256 + 55 = 3468664375$. For example, assume there are 10000 proxy cache servers. The first four numbers, 3468, represents the identifier of the assigned proxy cache server.

We use a two-level DNS to realize the directory service. The first level is the top-level domain (TLD), which could be any legal domain server. It contains location information for video servers and delegates its sub-domain to each video server. The second level is located at a video server, which contains location information of the proxy cache servers. Fig. 3 illustrates the domain transmission. For example, suppose a user request C_j wants to access a video R_i . In our directory service, the client first uses the video name as a key to compute a hash value $H(R_i)$, and then connects to the TLD serve to get video server location S_j . Once the client request has been received, the video server computes the table index of $H(C_j)$, and returns P_j , which is the cache server closest to the user. The user thus obtains the requested video from the cache server.

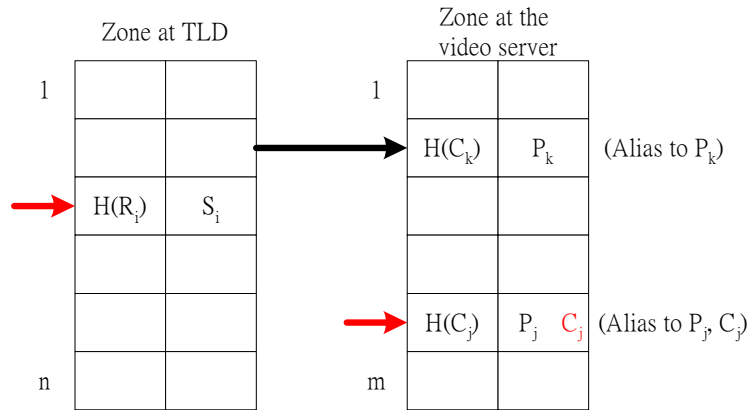


Fig. 3. Two-level hashing.

In addition, we wish that users could share video segments with each other while they are watching videos. To achieve this goal, we apply the dynamic DNS scheme. In Fig. 3, the video server adds the client’s IP, C_j , to the record of $H(C_j)$. Suppose another user C_{j2} with the same location as C_j (i.e., $H(C_j) = H(C_{j2})$) wants to watch the same video. Because $H(R_i)$ is identical, the user will find there are two video sources P_j and C_j . Thus, user C_{j2} will be able to download video data from either the proxy cache P_j or the peer user C_j .

3.2 Cache Service

The service contains two parts. One is the proxy cache server, which accepts user requests, delivers video data using hot-video broadcasting schemes, and caches video data. The other is peer cache, which a client becomes a temporal cache after receiving video data. The users that want to watch same movie can obtain data from their peer cache.

3.2.1 Proxy cache server

Fig. 4 illustrates the operation of the proxy cache. For example, assume a user requests a video with 15 segments. In the beginning, the cache is empty so the cache server must fetch segments from the video server according to the segment schedule of fast broadcasting, as indicated in Fig. 4 (a). In the figure, the segments lined vertically are downloaded at the first time slot, and segments drawn horizontally are obtained at the second time slot. Note that the data transmission between the proxy cache server and the video server is via unicast while the transmission between the proxy cache server and the clients is via multicast. If a later request requires the same video, the cached segments colored gray can be broadcast directly, as indicated in Fig. 4 (b). Only the missed segments are downloaded from the video server.

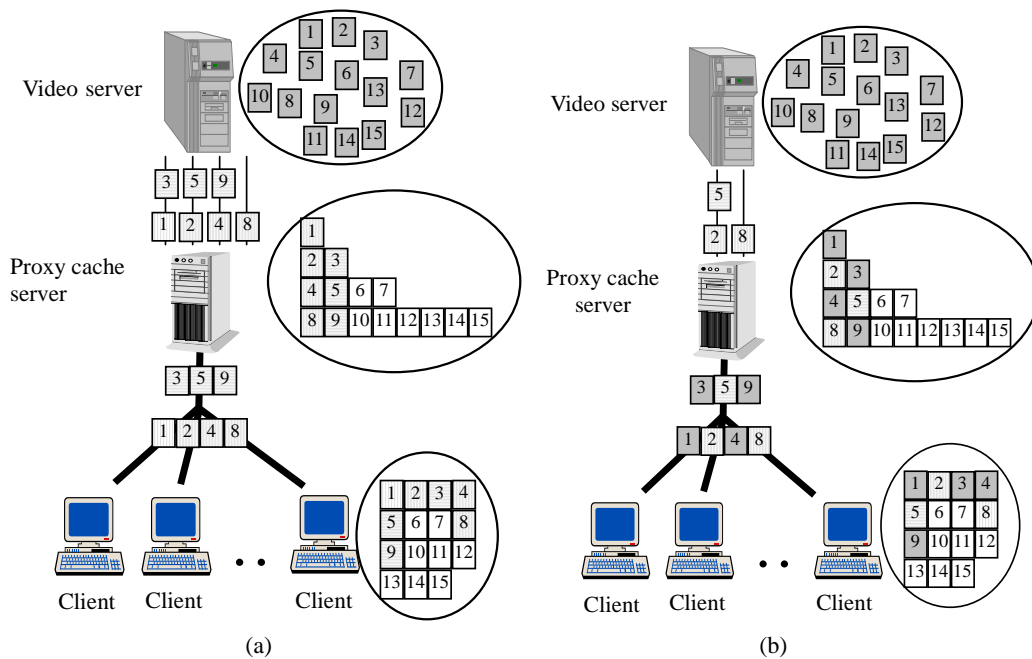


Fig. 4. An illustration for cache operation on four streams.

Due to limited cache size, segment replacement is necessary. The LRU is a basic approach. In addition, we consider the broadcasting frequency of a video segment. This is because the smaller the segment number, the higher the access frequency. Accordingly, if we can give the segment with a small number a higher priority, the hit ratio might increase.

3.2.2 Peer cache

There are three situations when a new user request arrives in our system. They are

- The requested video was not accessed before so the user must connect to the proxy cache server for video data.
- A peer user has already accessed the requested video. The user can download video data either from the proxy cache server or the peer user.
- A proxy cache server is currently broadcasting the requested video. The user can thus join the broadcasting session until finishing downloading all video data. Alternatively, the user receives the segments on broadcasting streams, and downloads the missed segments from the peer user.

Fig. 5 illustrates a peer cache. Initially, there are three users downloading video data by fast broadcasting. They have received segments 1, 2, 4, and 8. A new user then arrives when the proxy cache server begins transmitting segments 3, 5, and 9. Thus, the user

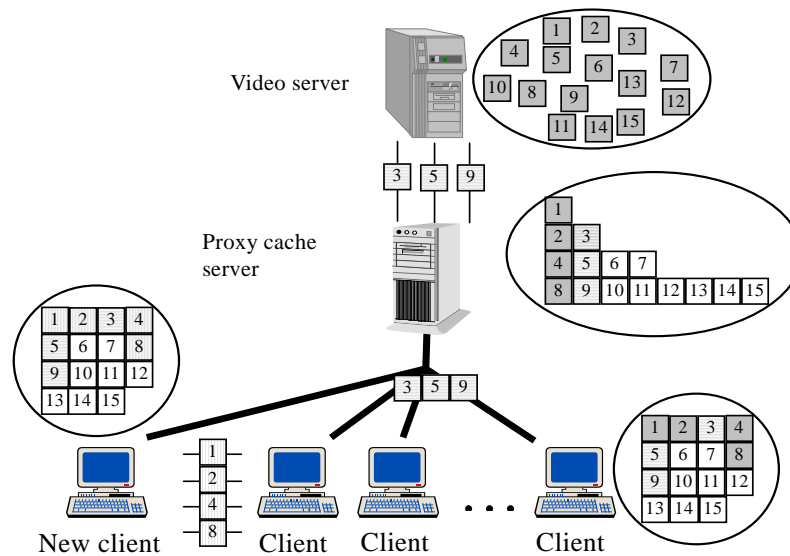


Fig. 5. An illustration for peer caching.

obtains segments 1, 2, 4, and 8 from the peer, and receives segments 3, 5, 9 from the proxy cache server. Accordingly, the proxy cache server can finish broadcasting data earlier, saving the bandwidth. The peer cache can also increase the fault tolerance. When a dedicated proxy cache server fails, a client can find a peer cache as a backup.

4. IMPLEMENTATION AND EXPERIENCE

4.1 Function Modules

Our system contains three parts – a video server, a proxy cache server, and a client, as indicated in Fig. 6. The video server offers three main functions which partition video into segments, deliver segments to the proxy cache server, and maintain the domain name service. The proxy cache server receives user requests, and transmits video data using the hot-video broadcasting scheme. Additionally, the cache server is also responsible for segment replacement. The client consists of four function modules. The first queries and updates DNS records dynamically. The next receives and reassembles video segments. The third function obtains segments from peer clients, and the last one is for video decoding and playing.

4.2 Demonstration

We implemented our system using the Java language. In particular, the client is built on the JMF library. Fig. 7 shows a video list once a user executes the client program. Assume that the user chooses the second video. The client program then computes the hash value according to the video name, and obtains the value 1. Suppose our TLD is

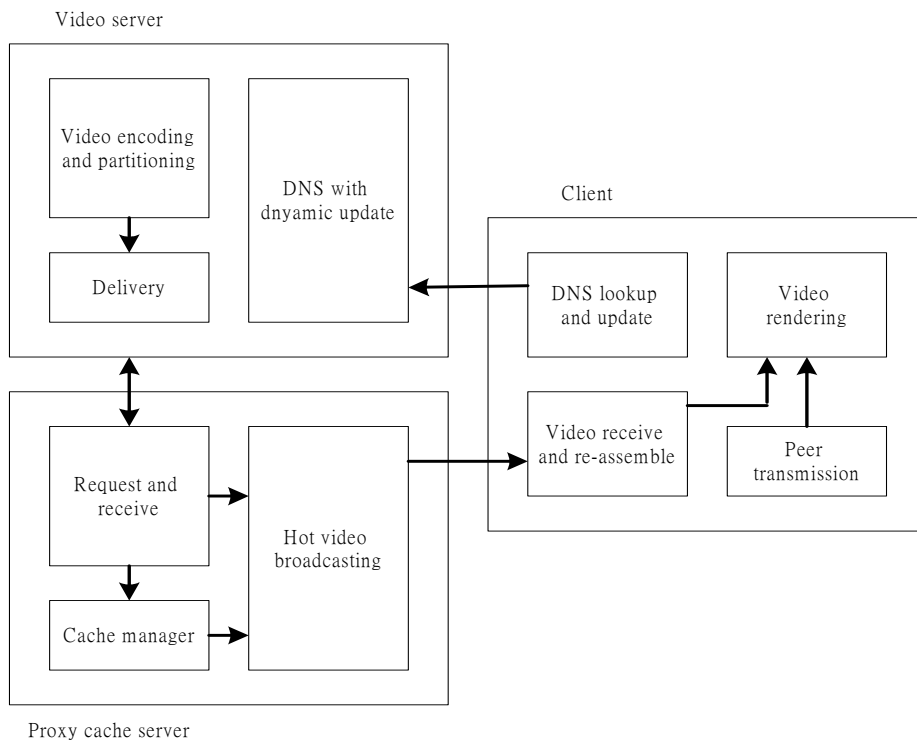


Fig. 6. The system function blocks.

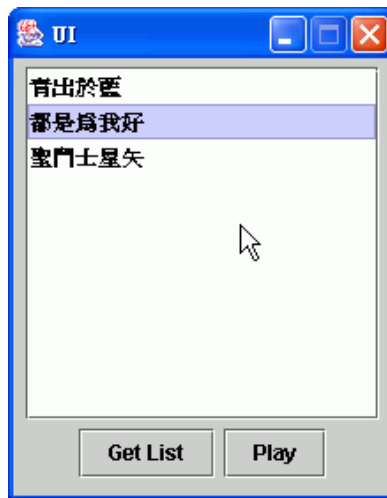


Fig. 7. The video list.

“dslab.ncu.tw”. Thus, the video server that stores the requested video is located at “1.dslab.ncu.tw”. Assume the user’s IP address is 140.115.50.58. We then compute the

hash value of this address, and obtain the value 2356359738. Suppose we use the first five numbers to index the proxy cache server. Thus, the found cache server is “23563.1.dslab.ncu.tw”.

In our demonstration, the IP of the proxy cache server is 140.115.50.71, as indicated in Fig. 8. Because it is the only video source, the user connects to it for data downloading. Once receiving the user’s request, the proxy cache server transmits the requested video by the FB scheme. Fig. 9 depicts the content change in cache. At the beginning, the proxy cache server only stores segments 1, 2, and 4. After receiving the request, it fetches the remaining segments 3, 5, 6, and 7 from the video server. Fig. 10 shows that the client first buffers the received data, and then plays the video. Finally, the client registers his/her address on the domain of “23563.1.dslab.ncu.edu.tw”, as indicated in Fig. 11.

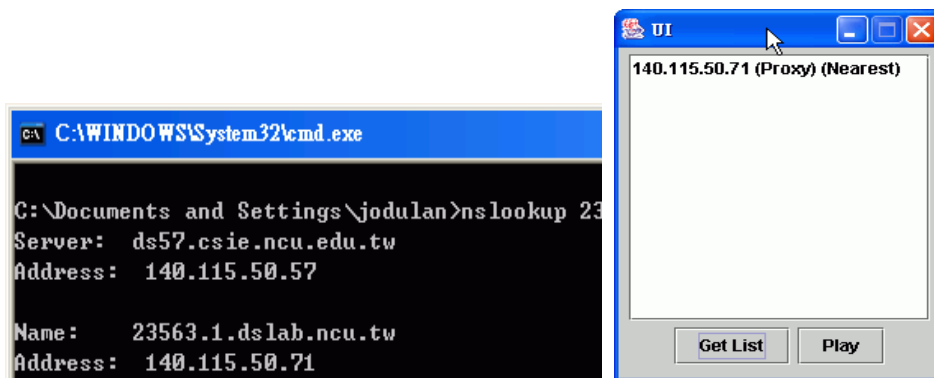


Fig. 8. The location of the proxy cache server.



Fig. 9. The cache change before and after broadcasting video data.

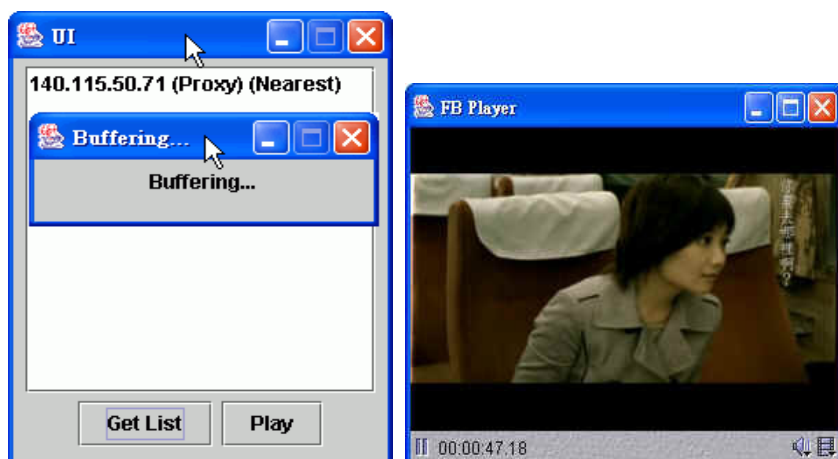


Fig. 10. Video playing on the client side.

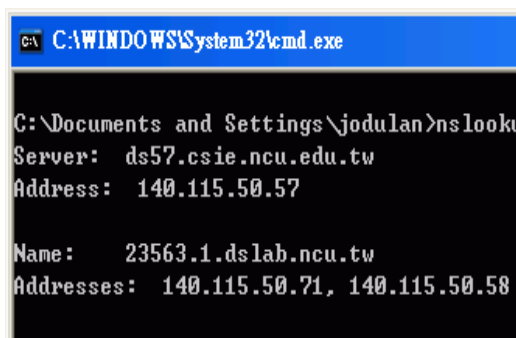


Fig. 11. The domain name update on the video server side.

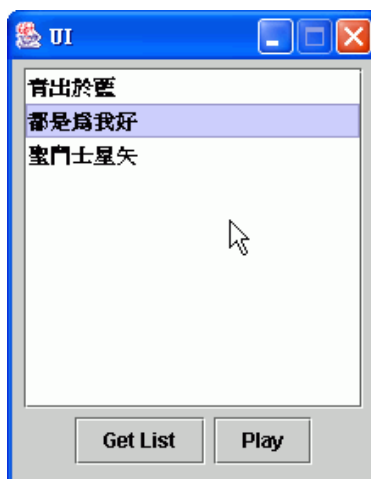


Fig. 12. The list of the proxy cache and peer nodes with the same video data.

Assume another client from the same location wants to watch the same video. The client will obtain the same domain name “23563.1.dslab.ncu.tw” for the proxy cache server through the same procedure. However, the domain name maps two IP addresses, 140.115.50.71 and 140.115.50.58, because of the registration of the former client, as indicated in Fig. 12. The client thus can select one source for downloading video data.

5. CONCLUSIONS

In this paper, we have proposed a new CDN, which combines hot-video broadcasting, cache and peer-to-peer communications. We widely deploy cache servers, which store video segments in advance, and distribute them via hot-video broadcasting technologies. By request routing, users can obtain video data from the nearest server. In addition, users probably obtain the data from their peer. Such peer patching can reduce start-up time and save backbone bandwidth. In addition, multiple data sources could avoid the video playing interruption caused by server/peer failure. Finally, we implemented a prototype for our architecture by two-level DNS with dynamic update of video information.

Our system did not fully address some issues. For example, the connection quality between peers may be unreliable, and video playing can be broken. Also, the node status may change very fast, e.g., from a live node to a dead one in a very short time. And in comparison with regular servers, a peer node has very limited resources. In the future, we plan to seek some approaches, such as QoS, for these problems.

REFERENCES

1. <http://www.napster.com/>.
2. <http://www.audiogalaxy.com/>.
3. <http://www.slsk.org/>.
4. <http://www.kazaa.com/>.
5. <http://www.winmx.com/>.
6. <http://www.edonkey2000.com/>.
7. <http://gnutella.wego.com/>.
8. K. C. Almeroth and M. H. Ammar, “The use of multicast delivery to provide a scalable and interactive video-on-demand service,” *IEEE Journal on Selected Areas in Communications*, Vol. 14, 1996, pp. 1110-1122.
9. A. Dan, D. Sitaram, and P. Shahabuddin, “Dynamic batching policies for an on-demand video server,” *Multimedia Systems*, Vol. 4, 1996, pp. 112-121.
10. H. Deshpande, M. Bawa, and H. Garcia-Molina, “Streaming live media over a peer-to-peer network,” in Work at CS-Stanford, Submitted for publication, 2002.
11. L. S. Juhn and L. M. Tseng, “Staircase data broadcasting and receiving scheme for hot video service,” *IEEE Transactions on Consumer Electronics*, Vol. 43, 1997, pp. 1110-1117.
12. L. S. Juhn and L. M. Tseng, “Harmonic broadcasting for video-on-demand service,” *IEEE Transactions on Broadcasting*, Vol. 43, 1997, pp. 268-271.

13. L. S. Juhn and L. M. Tseng, "Fast data broadcasting and receiving scheme for popular video services," *IEEE Transactions on Broadcasting*, Vol. 44, 1998, pp. 100-105.
14. J. F. Paris, "A simple low-bandwidth broadcasting protocol for video-on-demand," in *Proceedings of International Conference on Computer Communications and Networks*, 1999, pp. 118-123.
15. I. Stoica, R. Morris, D. Karger, M. F. Karsch, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of ACM SIGCOMM '01*, 2001, pp. 149-160.
16. Y. C. Tseng, M. H. Yang, and C. H. Chang, "A recursive frequency-splitting scheme for broadcasting hot videos in VOD service," *IEEE Transactions on Communications*, Vol. 50, 2002, pp. 1348-1355.
17. P. Vixie (ed.), S. Thomson, Y. Rekhter, and J. Bound, "Dynamic updates in the domain name system," RFC 2136, 1997.
18. K. L. Wu, P. S. Yu, and J. L. Wolf, "Segment-based proxy caching of multimedia streams," in *Proceedings of the 10th International World Wide Web Conference*, 2001, pp. 36-44.
19. Z. Y. Yang, L. S. Juhn, and L. M. Tseng, "On optimal broadcasting scheme for popular video service," *IEEE Transactions on Broadcasting*, Vol. 45, 1999, pp. 318-322.
20. Z. Y. Yang, "The tele – presentation system over internet with latecomers support," Ph.D. Dissertation, Department of Computer Science and Information Engineering, National Central University, Taiwan, 2000.
21. B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, 2001.



Chih-Yang Liu (劉之揚) received the B.S. and M.S. degrees in Computer Science from National Central University, in 2001 and 2003. He currently works as a software engineer in developing digital TV. His interests are embedded system, network programming, and distributed multimedia system.



Su-Chiu Yang (楊素秋) received the M.S. degree in Electronics Engineering from National Taiwan Institute of Technology in 1980. Currently she is working towards her Ph.D. degree in Computer Science and Information Engineering at National Central University, Taiwan. Her research interests are in the area of internet traffic measurement, intrusion detection, and network management.



Hsiang-Fu Yu (游象甫) received his B.S. degree in electrical engineering, and his M.S. and Ph.D. degrees in computer science from National Central University, Taiwan in 1993, 1995, and 2004, respectively. He currently works as a senior programmer at the Computing Center in National Central University. His research interests include proxy cache, information retrieval, and multimedia streaming.



Li-Ming Tseng (曾黎明) received the B.S. degree in Engineering Science, the M.S. and the Ph.D. degrees in Electrical Engineering from National Cheng Kung University, Taiwan in 1970, 1974 and 1980, respectively. Dr. Tseng is currently a professor in the Department of Computer Science and Information Engineering in National Central University, Taiwan. His research interests include distributed systems, computer networks, multi-lingual domain name systems, and stream delivery. He is a member of IEEE and Chinese Computer Association.