Exploiting Traffic Balancing and Multicast Efficiency in Distributed Video-on-Demand Architectures [*]

Fernando Cores, Ana Ripoll, Bahjat Qazzaz, Remo Suppi, Xiaoyuan Yang, Porfidio Hernández, and Emilio Luque

Computer Science Department – University Autonoma of Barcelona – Spain
{Fernando.Cores,Ana.Ripoll,Remo.Suppi,Porfidio.Hernandez, Emilio.Luque}@uab.es
{Bahjat,Xiaoyuan}@aows10.uab.es

Abstract. Distributed Video-on-Demand (DVoD) systems are proposed as a solution to the limited streaming capacity and null scalability of centralized systems. In a previous work, we proposed a fully distributed large-scale VoD architecture, called Double P-Tree, which has shown itself to be a good approach to the design of flexible and scalable DVoD systems. In this paper, we present relevant design aspects related to video mapping and traffic balancing in order to improve Double P-Tree architecture performance. Our simulation results demonstrate that these techniques yield a more efficient system and considerably increase its streaming capacity. The results also show the crucial importance of topology connectivity in improving multicasting performance in DVoD systems. Finally, a comparison among several DVoD architectures was performed using simulation, and the results show that the Double P-Tree architecture incorporating mapping and load balancing policies outperforms similar DVoD architectures.

1 Introduction

Video on Demand (VoD) has been gaining popularity over recent years with the proliferation of high-speed networks. Distributed continuous media applications, are expected to provide service to a large number of clients often geographically dispersed over a metropolitan, country-wide or even global area. Employing only one large centralized continuous media server to support these distributed clients results in a high cost and non-scalable system with inefficient resource allocations.

To address this problem, researchers have proposed distribution of the service in order to manage client dispersal. Systems based on this approach are termed Distributed VoD systems and they have demonstrated the ability to provide minimum communication-storage cost for distributed continuous media streaming applications [1]. A DVoD system requires the arrangement of those servers that offer the video retrieval and playback services in a distributed system, in order to support a large num-

[*] This work was supported by the MCyT-Spain under contract TIC 2001-2592 and partially supported by the Generalitat de Catalunya- Grup de Recerca Consolidat 2001SGR-00218.

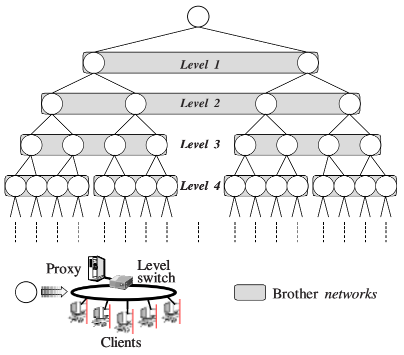
ber of concurrent streams. In the literature, these approaches range from: 1) the use of Independent servers, 2) one level proxies, to 3) hierarchical distributed systems. The initial approach is based on replicating VoD servers close to clients' networks so that these users do not need to access the main server [2]. One-level proxies try to reduce the size of local servers in such a way that they only store those videos with a higher access frequency; these servers are managed as main-server caches and are called proxies [9]. Hierarchical DVoD systems are based on a network with a hierarchical topology, with individual servers on the nodes and network links on the edge of the hierarchy. The nodes at the leaves of the hierarchy, termed head-ends, are points of access to the system where clients are usually connected [3][9][12][14].

In [6] we proposed an architecture for a fully distributed VoD system (called Double P-Tree) which, in addition to supporting a large number of concurrent streams, allows for the distribution of network traffic in order to minimize the network's bandwidth requirements. This is achieved by distributing both the servers as well as the clients throughout the topology, avoiding the concentration of communication traffic on the last level of the hierarchy (head-end). It is demonstrated through an analytical study that this distributed architecture is fault-tolerant and guarantees unlimited and low-cost growth for a large-scale VoD system.

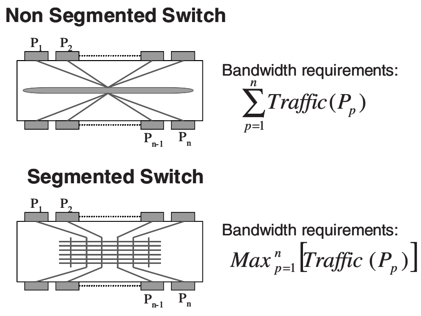
In this paper, we focus on the design aspects of the Double P-Tree architecture with the view to optimizing its performance and to supporting a greater streaming capacity. We concentrate particularly on two aspects: incorporating a video-mapping mechanism to minimize service distance, and the proposal of traffic-balancing policies that allow a reduction in network bandwidth requirements for the system. These proposed policies have been evaluated through several simulation experiments and the results have shown significant improvement in the Double P-Tree architecture's performance. In addition, on the one hand we study the influence of the architecture's connectivity in improving the efficiency of multicast policies in distributed systems, and on the other hand we analyze the proxy storage capacity.
The remainder of this paper is organized as follows: in section 2, we first give an overview of the Double P-Tree architecture and we describe some topics related to its implementation. In section 3, we propose some techniques related to video placement and traffic balancing. Performance evaluation is shown in section 4 and, finally, in the last section, we indicate the main conclusions to be drawn from our results.

2 Distributed VoD Architecture

Fig.1a depicts the architecture of the proposed DVoD system. This architecture is designed as a network with a tree topology, with individual small servers (proxies) as the nodes, and network links as the edges of the hierarchy. Nodes are assumed to be able to store a limited number of videos and stream a finite number of videos. Meanwhile, networks links are expected to guaranteed the specific QoS requirements of video communications. A brief description of the system architecture is given below.



(a) Topology



(b) Network infrastructure

Fig. 1. Double P-Tree Architecture

2.1 Network Topology

For the network topology we have selected a fully distributed topology based on proxies. The structure of this topology consists of a series of levels, in accordance with the number of local networks and the order of the tree. Each hierarchy level is made up of a series of local networks with its local-proxy and clients that forms the following tree level. To improve topology connectivity, several local networks (named brothers networks) from the same level are interconnected, increasing the number of adjacent local networks without changing the topology size or last level width. This new architecture is named Double P-Tree because the brother networks are joined in a second tree within the original tree topology [6].

In order to reduce network bandwidth requirements the network infrastructure is designed using segmented switches in local networks. Fig 1b, shows network bandwidth requirements for non-segmented and segmented switches. This selection is based on the fact that in segmented switches, every port (P_i) has an Independent-bandwidth, and therefore, it is only necessary to have enough bandwidth in order to support the maximum traffic from all ports. Segmented switches allow the reduction of switch-bandwidth requirements if traffic is distributed among different ports.

Double P-Tree architecture can make better use of segmented switches due to its network traffic being distributed among different sources. A possible drawback of this utilization is that topology-port traffic (ports used to implement the topology) and server-port (port used to connect proxy-server) could be unbalanced when the proxy load is centralized in one server-port. In order to solve this unbalance (increasing the bandwidth requirements), the architecture connects the proxies to local-networks using several ports.

2.2 Proxy Server

The simple inclusion of a hierarchical system with proxies does not, in itself, obtain improvements in the system's scalability or efficiency: since, as all the proxies are caching the same videos, if a proxy cannot serve a request from its client, then it is also very probable that none of the other proxies will be able to serve this request, and the solution will then require accessing the main server. We therefore need to use a new proxy organization and functionality to increase the hit probability as the request climbs the various levels on the tree. This proposal lies in dividing the storage space available within proxies into two parts: one of these will continue functioning as a cache, storing the most requested videos; the other proxy space will be used for making a distributed mirror of system videos [6].

In order to provide True VoD we have concentrated on multicast transmission techniques [7][10]. These techniques can greatly reduce server I/O and network bandwidth. But with them, it is difficult to implement VCR functions since a dedicated channel is not allocated to each user. Whenever a user tries to play the VCR functions he will disjoin the multicast channel, and some new resources that have not been planned before must then be reserved and assigned to him. These resources will be used to meet the VCR functions and/or to provide a unicast channel so that the user can continue with the normal playback. Several ideas has been proposed to solve this problem basically reserving some channels for these specific actions [5][11] .

Our implementation for the VCR functions does not reserve specific channels and is based on the observation that there are periods of times during which network bandwidth is under-utilized. During these periods, the proxy server sends more video in advance (pre-fetching) to an appropriate client's buffer. Whenever a user invokes a VCR action, the resources that have been assigned before the peak time are recovered in favor of this VCR request.

Another important element that affects the proxy performance is the proxy file system. Proxy servers that implement conventional file systems have been designed to reduce load on servers as well as client access time [4][13]. Nevertheless for continuous media with soft real-time constraints, typical file systems based on best-effort paradigm are inadequate for achieving this new requirement. Proxy servers can provide performance guarantees to applications only in conjunction with an operating system that can allocate resources in a predictable manner.

In our case, the most representative workload is the updates and removes in the caching and mirroring subsystem; consequently, the disk broker must employ placement policies that minimize fragmentation of disk space resulting from frequent writes and deletes. In order to obtain the best performance from the disk driver, track-buffer techniques are used. These techniques eliminate rotational delays in reads and obtain maximum performance on write operations.

3 Architecture Design Issues

In order to implement an efficient DVoD architecture several challenging research problems have to be solved to allow an efficient management of network and the services. Some of these problems are related with the subjects of reducing service distance and balancing communication traffic. In this section we propose some policies to accomplish these goals.

Table 1. Performance of Video-Mapping Heuristic for Double P-Tree

	Mirror Distribution	Effective bandwidth	Mean service distance
Unicast	sequential	11.001 Mb/s	1,80
	heuristic	12.849 Mb/s	1,747
Multicast	sequential	14.913 Mb/s	1,80
	heuristic	15.951 Mb/s	1,747

3.1 Videos Mapping on Distributed Mirror

The main factor that penalizes DVoD architectures performance, measured as the number of concurrent clients supported by the system (effective bandwidth), is the over-bandwidth required due to requests that cannot be served locally. In this case, a remote service requires: local bandwidth in the remote server, bandwidth in the server-port in the switch, bandwidth in the remote switch-topology port, bandwidth in the local switch-topology port and finally bandwidth in the client switch-port. A good approximation to evaluate this over-bandwidth is mean service distance (the distance needed to reach all movies from every node in the system).

In particular, Double P-Tree mean service distance is affected by diverse factors, such as topology connectivity, proxy storage distribution between caching and mirroring, and mirror-videos mapping in proxies. The first and the second issues were analyzed in a previous paper [6], and the last one is studied below.

Given that it is too complex achieve a optimal video distribution, we have developed a heuristic to choose which videos need to be mapped in every proxy-mirror. This heuristic consists of calculating, for each proxy within the architecture, the minimum distance where we can find each of the movies of the system-repository (taking into account videos already mapped in the previous proxies). Then, in order to minimize the mean service-distance, we always choose those videos that are stored in the proxy-mirrors furthest from this proxy. In the case of there being various videos at the same distance, the most popular are then selected.
Table 1 shows the mean service distance and effective bandwidth obtained by the heuristic and a sequential distribution of videos based simply on assigning a group of videos to each one of the proxies in a sequential manner. These results use the simulation parameters given in section 4, taking unicast and multicast policies into account. As we can see, the heuristic reduces mean-service distance from 1.80 to 1.74, which allows for an improvement in system performance ("Effective Bandwidth" column) of 7% and 14% for unicast and multicast, respectively.

3.2 Traffic Balancing Policies

In architectures using segmented switches, the bandwidth requirements for a network depends on the maximum traffic supported by any of its ports. Therefore, it is very

Table 2. Performance of Traffic Balancing Policies for Double P-Tree

	Traffic Balancing Policy	Effective bandwidth	Mean Distance	Imbalance
Unicast	Unbalance	12.849 Mb/s	1,747	56,44%
	Mirror Balanced	12.240 Mb/s	1,769 (3)	55,61% (1)
	Traffic Balancing	15.653 Mb/s	1,747	30,45% (5)
Multicast	Unbalance	15.951 Mb/s	1,732	78,27%
	Mirror Balanced	15.630 Mb/s	1,769 (4)	71,70% (2)
	Traffic Balancing	19.700 Mb/s	1,747	48,07% (6)

important to balance port traffic in order to reduce bandwidth requirements and to increase system performance.

In Double P-Tree architecture, most loaded ports are of the server and topology ports. Server ports can easily be balanced because the proxy server can choose at any time through which port a request is attended.

On the other hand, balancing topology ports is more difficult because their load depends on video placement on distributed mirrors. For example, the links connecting proxy servers that map the most popular videos would be more overloaded than others, producing a traffic imbalance.

To achieve traffic-load balancing, we have studied two different approximations:

- Mirror Balanced.

  An initial approximation to avoid imbalance is by building the distributed mirror in more balanced way. This objective can be achieved tuning the previous mapping heuristic because we do not always select the most popular videos; rather, we choose a mixture of highly and less popular videos.

- Dynamic traffic balancing.

  However, balancing through mirror distribution has a very limited maneuverability and cannot easily adapt to changes in traffic patterns or video access frequency. Therefore we have proposed another more dynamic policy for traffic load balancing. As the imbalance problem only appears with remote requests (which are the only ones that use topology ports), when a request cannot be attended, the local proxy also receives information about traffic from all alternative sources (i.e. proxies that have a copy of requested video and enough resources to attend the request). Using this information, and if there are two or more alternative paths (meaning some video replication), our balancing policy always chooses the least-loaded path in order to balance traffic.

Using the simulation parameters given in section 4, in Table 2, we show the results obtained with these balance policies. As we can see, compared with imbalance, the mirror balanced policy is successful in reducing imbalance[1,2], but this reduction is not enough to compensate for the rise in service distance[3,4]. In contrast, the results obtained with the dynamic traffic load balancing policy are much better, decreasing imbalance significantly[5,6] (without affecting service distance) and increasing performance by around 24% (15.951Mbs and 19.700Mbs as against 12.849Mbs and 15.653Mbs , for unicast and multicast respectively).

4 Performance Evaluation

In this section, we show the simulation results for Double P-Tree architecture and contrast these with other distributed architectures. We conducted several experiments to 1) evaluate the effect of multicast, and 2) study the effect of proxy storage capacity on system behavior.

Table 3. Simulation Parameters

Parameter	Value	Parameter	Value
• Number of videos	100	• Multicast technique	Patching
• Video length	90 minutes	• Client buffer size	5 minutes
• Local networks	63	• Request rate (λ)	10 req/min by net
• Network bandwidth	100 Mb/s	• Poisson distribution	$p_i = \frac{\lambda_i^k}{k!} e^{-\lambda_i}$ (1)
• Server bandwidth 1server port (1Sp)	100 Mb/s		
• Server bandwidth 3server port (3Sp)	300 Mb/s	• Zipf distribution	$p_x = \frac{1}{x^z \cdot \sum_{i=1}^{Sv} 1/i^z}$ (2)
• Proxy capacity	20 videos		

4.1 Simulation Environment

To guide this objective, we have designed and implemented an object-oriented VoD simulator. The main parameters of the simulation environment are summarized in Table 3. In all studies, we use architectures with 63 local networks (6 levels in Double P-Tree topology) using 100 Mbps segmented switches. The request inter-arrival time is generated by the simulation of a Poisson distribution[1] with a mean of 1/λ, where λ is the request arrival rate in every local network in the VoD system. The selection of the video is modeled with a Zipf distribution[2], with a skew factor of 0.7 (z), which models the popularity of rental movies [1].

4.2 Effect of Multicast on Distributed VoD Architectures

In this section, we evaluate DVoD system performance, using effective bandwidth (number of users attended * 1.5Mbs) as the main metric. This study allows us to evaluate the maximum streaming capacity for different architectures: Independent servers, one level proxies and Double P-Tree (using heuristic mapping and dynamic traffic balancing policy), for both unicast and multicast techniques.

In order to obtain the results, plotted in Fig 2, we have assigned an aggregate network bandwidth of 6.300Mbs and an aggregate sever bandwidth of 6.300Mbs (with 1 server port) or 18.900Mbs (with 3 Server ports). To obtain the maximum system streaming capacity, we saturated the system using a high request ratio (10 req/min by network) and simulated the system behavior until the aggregate bandwidth is ex-

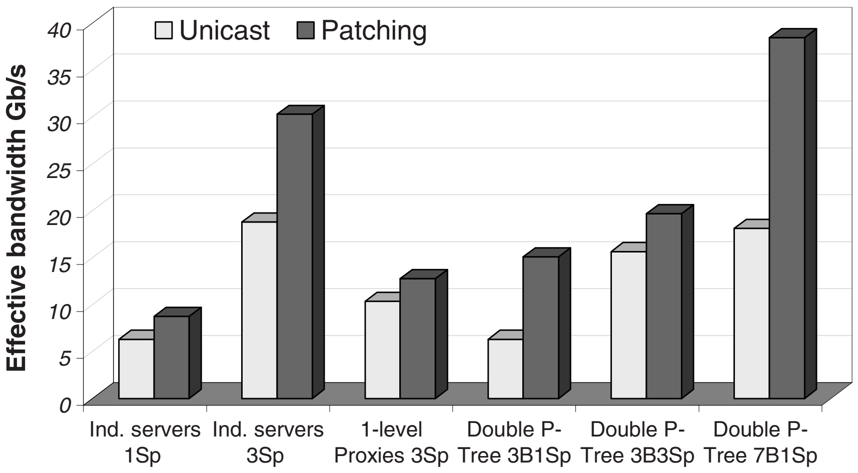


Fig. 2. Performance in Distributed VoD Architectures

hausted. When there is no bandwidth available, the system achieves its maximum streaming capacity and we evaluate the system performance (effective bandwidth using unicast and patching [12]) .

Using unicast, we can see that the Independent servers with 3Sp is the architecture that obtains the best results, achieving the theoretical maximum stream capacity of 19Gbs (63 networks * 100Mbs * 3Sp). Meanwhile Double P-Tree with 7 brothers (7B) and 3Sp obtains an effective bandwidth of 18.2Gbs (4% less), due to the additional bandwidth required to attend distributed requests and lower storage requirements (20 as against 100 videos in every proxy). However, this underperformance is less than expected according to the criteria of mean service distance between Independent servers (1) and Double P-Tree architectures (1,747 according to Table 2). This result demonstrates the strength of our architecture in distributing and balancing traffic among topology ports in order to reduce network requirements. Also, as we can see, our architecture is better than 1-level proxies architecture (improved by 75%).

Double P-Tree architecture exploits its characteristics to realize its potential advantage when client streams are shared using a multicast technique (Patching, in our case). In this more realistic scenario, Double P-Tree is the best solution, improving Independent servers by 27% (38.4Gbs as against 30Gbs) and one-level proxies by 200% (38.4Gbs as against 12.7Gbs).

The principal argument for this improvement is that the Double P-Tree has a better connectivity than the Independent server and one-level architectures. This better connectivity means that, in Double P-Tree, mirror video streams can potentially be shared among requests coming from all adjacent networks, multiplying the sharing probability by topology connectivity. Meanwhile, in low interconnected architectures, potential stream sharing is limited only to local requests.

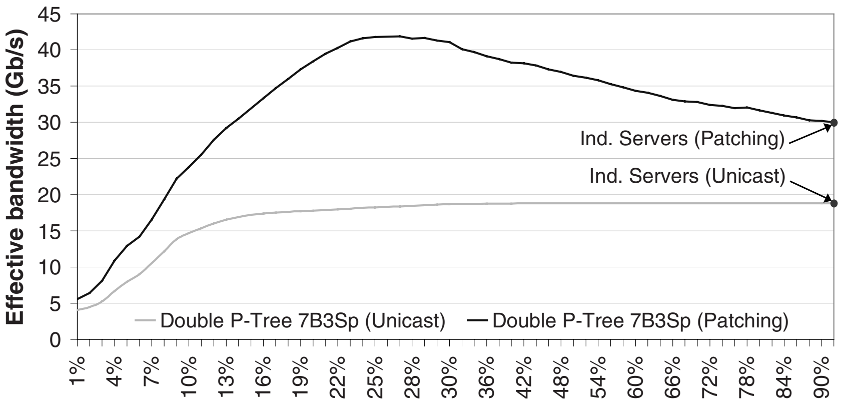


Fig. 3. Double P-Tree Effective Bandwidth & Proxy Storage Capacity

4.3 Effect of Proxy Storage Capacity

In this section, we evaluate the effect of proxy storage capacity on Double P-Tree performance. In Fig. 3, we can first see that a small proxy storage leads to low performance due to large service distance being required to attend remote requests, and to its over-bandwidth requirements.

With only storage for 15% of system videos, Double P-Tree performance (with patching) is equivalent to Independent severs performance, but uses 6 times less storage. Also, we notice that the highest performance is obtained with proxy storage of around 25%. In this case, Double P-Tree performs Independent servers in more than 38% (41.5Gbs against 30Gbs). From this point, we can observe that in the measure that storage capacity grows, performance decreases until reaching Independent server performance, in which case the proxy has enough capacity to store a full video catalog copy.

Why do more resources give less performance? The reasons for this interesting effect can be explained by the fact that, when Double P-Tree proxies have a lot of storage (more than 30%), their architectural behavior is very similar to that of Independent server systems. In this case, proxy mirrors have enough storage to reach all videos at distance-2 mirrors, therefore all remaining storage is assigned for caching.

Increasing proxy-cache size increases the number of requests attended locally, creating two consequences. First, server ports have more load, leading to network traffic unbalancing and a faster network saturation. Second, there are videos with a medium access pattern that were previously managed under mirroring scheme. Proxy-mirrors, where these were mapped, centralized all requests coming from adjacent nodes, improving stream sharing. If we now place these videos in the cache (replicating them in all proxies), we are decreasing access frequency for every video copy, reducing both the stream-sharing probability and system performance. This result clearly demonstrates the goodness of distributed mirroring as against several full mirror replications.

5 Conclusions

This paper deals with two decisive aspects for DVoD architectures performance: video placement policies in distributed mirrors and network-traffic balancing policies. The proposed policies attain a reduction in mean service distance and minimize network requirements for the Double P-Tree architecture.

Simulation results show that proposed policies substantially increase the number of concurrent clients who can be served by the system. The video mapping heuristic achieves an improvement of 7%, while the dynamic traffic-balancing policy yields an additional increase of 26%. These results clearly demonstrate the importance of network-traffic balancing policies as a fundamental instrument in diminishing the network bandwidth requirements in DVoD systems.

On the other hand, we have also shown the importance of topology connectivity in DVoD systems (in particular, in the Double P-Tree) in order to improve multicasting performance. The Double P-Tree using multicasting and similar resources clearly outperforms classical DVoD architectures, namely, Independent servers (by 38%) and one-level proxies (by 200%).

Finally, we have demonstrated that full mirror replication in every local network (as in Independent servers) not only requires more storage but also achieves a poorer performance in comparison to distributed mirroring (Double P-Tree).

References

1. S.A. Barnett and G. J. Anido, "*A cost comparison of distributed and centralized approaches to video-on-demand*," IEEE Journal on Selected Areas in Communications, vol. 14, pp. 1173–1183, August 1996.
2. S.-H. G. Chan and F. Tobagi, “Caching schemes for distributed video services”, in Proc. of IEEE Int’l Conference on Communications (ICC'99), Canada, June 1999, pp. 994–1000.
3. S.-H. G. Chan and F. Tobagi, “Distributed Servers Architecture for Networked Video Services”, in IEEE/ACM Transactions on Networking, Vol. 9, No. 2 April 2001.
4. A. Chankhunthod, P.B. Danzing, C. Neerdaels, M. F. Schwartz, and K.J. Worrell. “A Hierarchical Internetwork Object Cache". In Proceeding of the 1996 USENIX Technical Conference, San Diego, CA, January 1996.
5. J-M Choi, S-W Lee, K-d Chung, “A Multicast Scheme for VCR Operations in a Large VOD system”, ICPADS 2001:555–561.
6. F. Cores, A. Ripoll, E. Luque, “Double P-Tree: A Distributed Architecture for Large-Scale Video-on-Demand”, Euro-Par 2002, LNCS 2400, pp. 816–825, Aug. 2002.
7. A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," Multimedia Systems 4, pp. 112–121, June 1996.
8. H. Fabmi, M. Latif, S. Sedigh-Ali, A. Ghafoor, P. Liu, L.H. Hsu, “Proxy servers for scalable interactive video support” , IEEE Computer , Vol. 34 Iss. 9, pp. 54–60 Sept. 2001.
9. C. Griwodz. “Wide-Area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure.” PhD dissertation, Darmstadt Univ. of Technology, Germany, Apr. 2000.
10. K. A. Hua, Ying Cai and S. Sheu, Patching: A multicast tecnique for true video-on-demand services, ACM Multmedia’98, pages 191–200.

11. J. Y. B. Lee, "On a Unified Architecture for Video-on-Demand Services", IEEE Transactions on Multimedia, Vol. 4, No. 1, March 2002.
12. Cyrus Shahabi, Farnoush Banaei-Kashani, "Decentralized Resource Management for a Distributed Continuous Media Server" , IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 7, July 2002
13. "Squid Internetwork Objet Cache Users Guide". Available on line at http://squid.nlanr.net, 1997.
14. Xiaobo Zhou, R. Luling, Li Xie, "Solving a Media Mapping Problem in a Hierarchical Server Network with Parallel Simulated Annealing", Procs. 2000 International Conference on Parallel Processing, pp. 115–124, 2000.