Double P-Tree: A Distributed Architecture for Large-Scale Video-on-Demand

Fernando Cores, Ana Ripoll, Emilio Luque[1]

Computer Science Department - University Autonoma of Barcelona – Spain
Fernando.Cores@uab.es Ana.Ripoll@uab.es Emilio.Luque@uab.es

Abstract. In order to ensure a more widespread implementation of video-on-demand (VoD) services, it is essential that the design of cost-effective large-scale VoD (LVoD) architectures be able to support hundreds of thousands of concurrent users. The main keys for the designing of such architectures are high streaming capacity, low costs, scalability, fault tolerance, load balance, low complexity and resource sharing among user requests. To achieve these objectives, we propose a distributed architecture, called double P-Tree, which is based on a tree topology of independent local networks with proxies. The proxy functionality has been modified in such a way that it works at the same time as cache for the most-watched videos, and as a distributed mirror for the remaining videos. In this way, we manage to distribute main server functionality (as a repository of all system videos, server of proxy-misses and system manager) among all local proxies. The evaluation of this new architecture, through an analytical model, shows that double P-Tree architecture is a good approach for the building of scalable and fault-tolerant LVoD systems. Experimental results show that this architecture achieves a good tradeoff between effective bandwidth and storage requirements.

1 Introduction

Video-on-demand (VoD) refers to video services in which a user is able to request any video content at any time from a server. This technology is important for many multimedia applications such as distance learning, digital libraries, videoconferences, Internet, TV broadcasting, and video-on-demand systems.

The service of a video request involves a high volume of information with real-time requisites, very strict quality of service (QoS) levels and great disk bandwidths. These requirements imply that a multimedia server can support only a limited number of users depending on its capacity. Moreover, VoD system capacity is also restricted by available network bandwidth, which requires a massive investment in infrastructure.

Therefore, in order to provide VoD services to accommodate hundreds of thousands of concurrent users, the design of large-scale VoD architectures is required. In addition to high streaming capacity, the main keys for these LVoD systems are:

– Low-cost. An LVoD system may require a high investment; therefore, it is imperative to reduce server, network and storage costs.

[1] 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.

- Scalability. It is essential that the system can adjust initial system-size to user requirements, but maintaining the possibility of easy expansion to accommodate more users or new services.
- Low complexity. LVoD architecture components should not be too complex as this can make system implementation, design and management more difficult.
- Fault-tolerance. VoD systems have to continue giving service to users even if one or more architecture components crash.
- Load distribution is important due to the nonuniform distribution of user requests, which leads to load imbalances among servers and poor utilization of the overall system resources.
- Resource sharing. Nowadays, resource sharing (broadcast, multicast, etc.) is the key for the design and implementation of cost-effective VoD systems.

In order to attain an LVoD system and to accomplish the previous requirements, we proposed a full-distributed architecture based on a tree topology of independent networks with proxies. This architecture, called Double P-Tree, modifies the typical proxy functionality in such a way that it works at the same time as cache for the most-watched videos, and as a distributed mirror for the remaining system videos. To enhance topology connectivity and to improve architecture efficiency and fault tolerance, we group several local networks (brother nets) from the same level.

This paper is organized in the following way. In section 2, we will first undertake an overview of the solutions proposed in the literature for the construction of LVoD systems. Following this, in section 3, we will describe the Double P-Tree architecture. In section 4, we will describe an analytical model for the architecture and in section 5 we show its evaluation. Finally, in the last section, we will indicate the main conclusions to be drawn from our discussion, and will suggest future lines of research.

2 Related Work

In recent years, research into VoD systems has mainly been focused on policies that attempt to improve available bandwidth efficiency. These techniques basically aim at increasing the number of users that can be served with a limited bandwidth. Such approaches are grouped into two broad policy groups: broadcasting (pyramid [14] and skyscraper [7]) and multicasting techniques (batching [4], patching [8], merging [5] and chaining [12]). Nevertheless, all of these techniques aim to improve the performance of available bandwidth within the system, but do not increase it.

Currently, there are several alternatives that can be used to implement LVoD systems:

- Centralized systems are the simplest approach to an LVoD system, but require high costs, very complex servers and are not scalable [1].
- Independent servers [2][13]. In these systems, users are grouped into network segments known as local networks; in such a way that system bandwidth is able to be that accumulated by each one of the individual nets. The key to the success of these systems is based on placing VoD servers close to clients' nets so that these users do not need to access the main server, and thereby create a system of hierarchical servers. These systems have an unlimited scalability (adding new servers) but with high storage costs, load imbalance and poor resource sharing.

- Proxy-based systems [2][6]. Previous architecture involves considerable storage cost; as a result, certain proposals have opted for reducing the size of local servers in such a way that they do not store all system videos, but rather, only those with a higher access frequency. These servers are managed as main-server caches and are called proxies, just as their Internet counterparts. The main problem with these systems is that requests that cannot be served locally end up in the main server, which becomes both a bottleneck and a growth-limiting factor.

In spite of these architectures being able to obtain a high streaming capacity, they do not successfully achieve all LVoD requirements, which limit their implementation.

Current research on scalability and distributed systems are focused on server design and implementation [10][11], more than in the entire system. Therefore, they do not consider net bandwidth costs and scalability as main goal. Recently studies about distributed VoD architectures are oriented in the system management [9] or how to map the media assets onto hierarchical architectures to improve QoS in [15].

3 Designing a fully distributed VoD system

In this section we present the different elements that integrate our architecture. First, we show the basic tree topology and the new proxy functionality; we then extend the tree topology, obtaining our final proposed architecture and its implementation.

3.1 Basic Topology (P-Tree) with mirroring

Our first approach consists of an expansion of the proxy architecture, currently restricted to a single level, using a tree topology that provides the system with unlimited scaling capacity, as well as greater flexibility when deciding on its size and form of growth. This topology, called Proxy-Tree (or P-Tree) presented in [3], consists of a series of levels, in accordance with the number of local networks and the order of the tree (binary, tertiary, etc.). Every hierarchy level is made up of a series of local networks with its local-proxy and clients that forms the following tree level. Subsequent networks are successively hung from each one of the previous levels of local networks until reaching the final level.

To decentralize the architecture we remove the main server, distributing is functionality (as a repository of all system videos, server of proxy-misses and system manager) among all local proxies. To do this, we modify proxy functionality, dividing the proxy storage into two parts: one of these will continue being managed as a cache, storing the most requested videos; and the remainder will be used for making a distributed mirror of system videos.

However, this distributed architecture, with mirroring, does not achieve as much performance as similar systems such as one-level proxies, because it has a smaller proxy-hit probability and a bigger average request-service distance. This smaller proxy-hit probability is due to our having to distribute proxy storage into two different schemes (caching and mirroring), reducing the total popularity of proxy videos and affecting proxy efficiency.

On the other hand, total net traffic also has an important influence on VoD system performance. In proxy systems, net traffic depends on request-service distance. For

example: In one-level proxy when a local-proxy cannot serve a request, this only has to cross one network (level) to reach the main server, and its miss penalty is restricted to twice the bandwidth required for a video stream. With the mirroring scheme, the proxy-miss service-distance is affected by distributed-mirror capacity, which depends on the number of proxies situated at the same distance from the local network. If distance-1 distributed mirror does not have enough capacity to store all system videos, then some requests have to cross two or more levels to be attended, having a penalty of 3 or more times the bandwidth required for a stream.

Moreover, both factors are dependent: increasing proxy-storage dedicated to caching implies a better proxy-hit probability but increases the average-distance needed to serve the proxy-misses from distributed mirroring. On the other hand, if we use more storage for mirroring, we reduce average mirror-service distance, but increase proxy-misses.

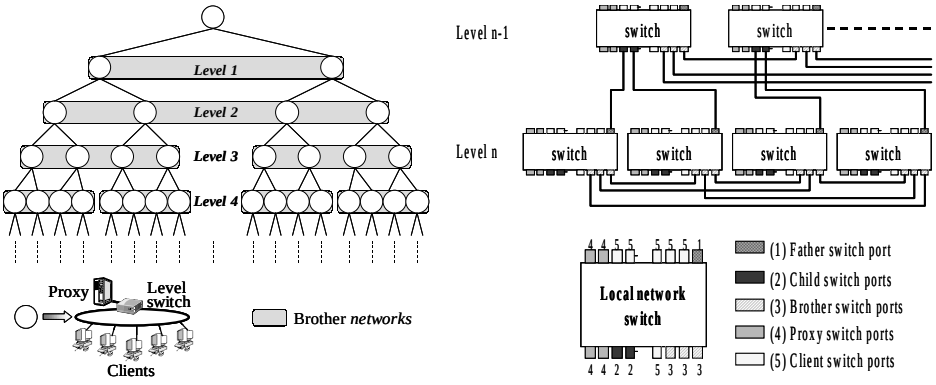
The only way to enhance distributed-mirror capacity, without modifying caching performance or proxy capacity, is by augmenting the number of adjacent proxies from every local network.

3.2 Improving connectivity (Double P-Tree architecture)

One way to increase connectivity is to use a bigger tree order, such as a tertiary-tree. However, we have realized that bigger tree-order topologies are not a good solution because they have a broader last-level that is proportionally far larger than in the binary tree topology. These last-level networks have poor connectivity, as they have no child networks, and this increases service-distance. Moreover, their effect on system efficiency is worse because the last level is the largest of the tree topologies.

A better solution is to increase topology connectivity by grouping several local networks (brothers nets) of the same level. In this way, we can increase the number of adjacent local networks without changing the topology size or last level width.

Using the concept of brother networks, we have designed a new topology shown in Fig 1a. This topology is called Double P-Tree due to brother networks forming a second tree within the original proxy tree topology.



(a) Topology. (b) Implementation.

Fig. 1. Double P-Tree architecture.

The main advantage of this topology is that it considerably reduces the service-distance of user requests. The most evident enhancement is the reduction of distance-2 traffic. Upgrading topology connectivity, the number of distance-1 proxies is augmented, and increases distributed-mirror capacity and efficiency. In this way, it is more feasible that the distributed mirror is able to store all the system videos by using only distance-1 proxies, and therefore, all requests would be satisfied without the need to access upper levels. Double P-Tree architecture not only increases the efficiency of the mirroring, it also improves caching. Since the distributed mirror is larger, this scheme does not need as much proxy-storage as previously required. This unneeded proxy-capacity can then be assigned for caching.

This topology also has a better fault-tolerance. In a simple binary-tree topology, a network crash can create a sub-tree that is totally isolated from the rest of the system, which can cause request reneging if this sub-tree has insufficient proxies to store a full copy of the system videos. This is more unlikely to occur with the new topology.

3.3 Double P-Tree Architecture Implementation

Double P-Tree topology implementation is performed in local-network switches. In Fig 1b, we show the reserved ports to build the topology and how they are interconnected. Every local-net has a port (father-port) to connect the local net with the upper topology-level, several ports (child-ports), depending on tree order, to connect all the local children nets (above level), some ports (brother-ports) to connect the net with its brothers in the same level and finally one or several ports for the local-proxy. The remaining switch-ports are used to connect clients, usually using several switches that form a tree hierarchy.

Distributed architectures usually have a performance penalty compared with centralized approaches. This efficiency reduction is the sacrifice required in order to obtain a scalable and fault-tolerant system. An important characteristic in distributed systems with respects to centralized ones is that load is distributed between different sources. Meanwhile in a centralized approach, only one source supports all load.

In order to take advantage of this feature, so as to reduce the penalty imposed on distribute architectures, we propose the use of segmented switches in local nets. Using a non-segmented switch, we need enough local-net bandwidth to support the total of local traffic. Instead, with a segmented switch, every port has an independent-bandwidth, and therefore, local nets only needs enough bandwidth to support the maximum of all port traffic.

However, with this scheme, topology-port traffic and local-proxy traffic are too unbalanced, because proxy load is centralized in only one port. To solve this unbalance that increases the bandwidth requirements for local nets, we connect proxies to the local-net using several ports.

4 Double P-Tree analytical model

In order to study the effectiveness of the proposed architecture, we have to demonstrate primarily: that it can scale without causing network or proxy saturation and that its effectiveness is at least equivalent to similar architectures.

The system scalability is estimated evaluating the growth of traffic generated by the system itself and the evolution of bandwidth requirements in architecture

components when the system grows. For our study, we are going to use the number of independent-streams supported by the system (effective bandwidth) as a main performance-metric. Other measurements such as total system bandwidth, local network bandwidths, server/proxies streaming capacity and storage requirements will provide us with an idea of the system's limitation with respect to the number of users that it can admit, its costs, and its grade of scalability.

All these parameters are evaluated using an analytical model of architecture behavior. Also, to contrast Double P-Tree system with similar approaches, we will study one-level proxies architecture performance. Table 1 shows the notation used during this analysis. In Table 2 we show the analytical model for one-level proxy-based architectures (which can also be used to model centralized and independent severs systems). Table 3 shows the expressions used for Double P-tree analytical model.

In the following models, we assume a unicast policy, i.e., each user is assigned to its own dedicated stream. This assumption is valid since our study is directed at evaluating the system streaming-capacity (effective bandwidth) and its scalability. These results will be independent of whether bandwidth management policies can later be used to increase the efficiency and number of clients managed by the system.

Table 1. Notation used in the analytical model.

Symbol	Definition	Symbol	Definition
B_T	Total system bandwidth	T_p	Proxy switch-ports
B_e	Effective system bandwidth	C	Proxy cache size (number of videos)
B_p	Main net bandwidth	M	Proxy mirror size (number of videos)
B_c	Local net bandwidth	P_{mc}	Proxy's cache miss probability
B_u	Local net user bandwidth	P_{hc}	Proxy's cache hit probability
L	Number of topology levels	P_{mm}	Proxy's cache+mirror miss probability
N	Number of local networks	P_{mc}	Proxy's cache+mirror hit probability
O	Tree-order	Lf_i	Load received by net *i* from father net
B	Number of brother-nets	Lc_{ib}	Load received by net *i* from brother net *b*
V	System videos	Ls_{is}	Load received by net *i* from son net *s*

Table 2. One-level proxy-based architecture analytical model

Measure	Expression	Nº.
Total bandwidth	$B_T = B_p + B_c \cdot n$	(**1**)
Main network required bandwidth	$B_p = B_u \cdot n \cdot p_{mc}$	(**2**)
Local networks required bandwidth	$B_c = B_u \cdot Max\left(\frac{1-p_{mc}}{T_p}, p_{mc}\right)$	(**3**)

Table 3. Double P-Tree Architecture analytical model.

Measure	Expression	Nº.
Effective system bandwidth	$B_e = B_u \cdot n$	(**4**)

Total system bandwidth	$B_T = \sum_{i=0}^{L} o^i \cdot Bc_i$	(**5**)
Bandwidth required by local net *I*	$Bc_i = Max(\frac{Lp_i}{Tp}, Lf_i, Lc_i 1, \cdots, Lc_i o, Ls_i 1, \cdots, Ls_i b)$	(**6**)
Load managed by local proxy	$Lp_i = [Bu \cdot (1 - p_{mm}(i,0))] + \left[Bu \cdot \sum_{d=1}^{L*2} Ne(i,d) \cdot \frac{p_{mm}(i,d)}{r_{\ln}(i,d)} \right]$	(**7**)

The traffic managed by topology ports (*Lf, Lc, Ls*), in expression (6) of table 3, is the sum of outgoing (local proxy-misses served from other proxies), incoming (proxy misses from remainder of system served by the local proxy) and passing traffic (proxy misses from the remainder of the system served by other proxies). Total outgoing traffic consists of the proxy-misses, and the total incoming traffic to the network can be calculated in the same way as in (7). However, its distribution among different ports and the passing traffic have to be evaluated by using an analytical simulator.

5 Double P-Tree evaluation

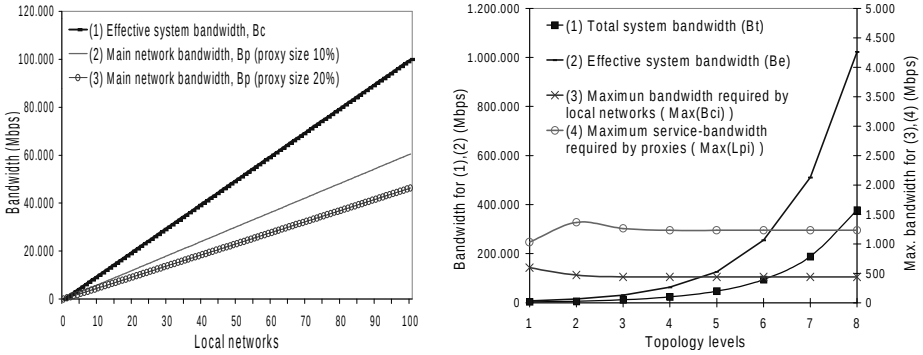
In order to evaluate Double P-Tree architecture and to compare its performance with other systems we assume a total system bandwidth of 127.000 Mbps (every 127 local net has 1.000 Mbps of bandwidth), that proxies/servers are connected using 4 switch-ports in all architectures (without taking topology ports into account), a proxy-capacity for 20% of system videos and in Double P-Tree a binary topology with 7 levels.

5.1 Scalability

Fig. 2, illustrates the bandwidth requirements for most critical scalability elements in distributed architectures when the system size grows. In one-level proxies (Fig. 2a), the most critical element is main network bandwidth and in the Double P-Tree, it is the bandwidth required by local networks and proxies (Fig. 2b).

With the results in Fig. 2a, we confirm our previous statements about the limited scalability of one-level architectures. In these architectures, system growth is obtained at the expense of increasing bandwidth requirements for the main network independently of proxy capacity (charts 2 and 3). Consequently, its scalability is limited by its centralized components (main network and server).

In the Double P-Tree, Fig. 2b, we notice that even when exponentially increasing the system capacity (chart 2), the maximum bandwidth required for local networks and its proxies (charts 3 and 4) is stable and small (400 and 1200 Mbps respectively). These results allow us to conclude that Double P-Tree architecture has an unlimited scalability, even when using small architecture components.



(a) One-level proxy-based system. (b) Double P-Tree system.

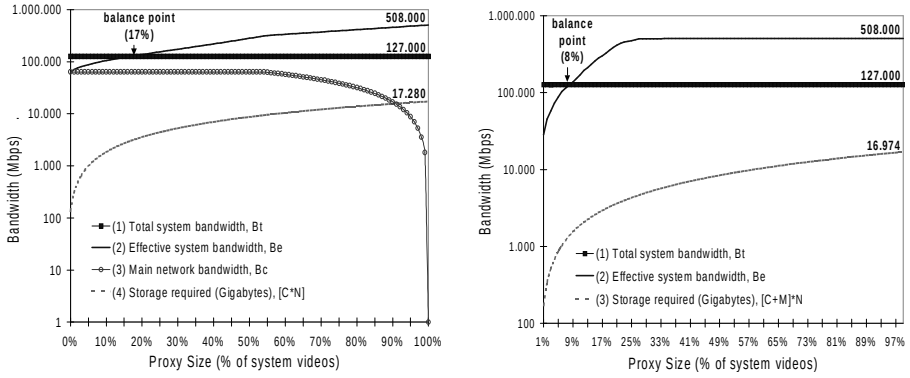
Fig. 2. Scalability of distributed architectures.

5.2 Proxy-storage requirements

In Fig. 3, following our comparison between proxy architectures, we study the proxy-storage requirements for both systems.

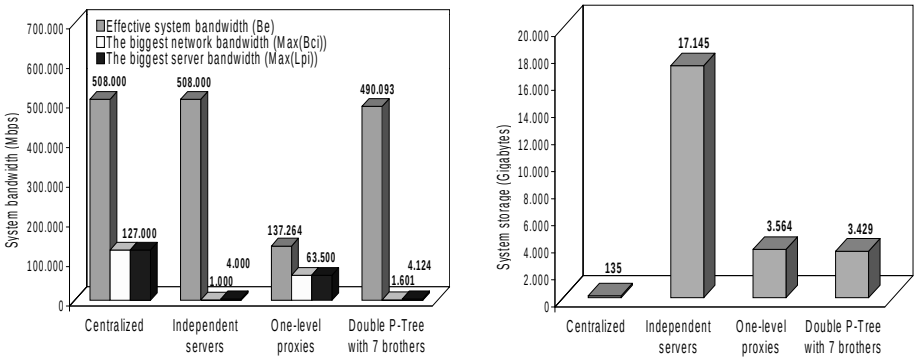
It is important to emphasize that, by using the same analytical model, in Fig. 3a we can evaluate the 3 main approaches for LVoD architectures: centralized systems (when proxy capacity is 0%), independent server systems (when proxy capacity is 100%) and one-level proxy systems in the remaining cases.

Comparing both proxy-based architectures, we can see that double P-Tree effective bandwidth (Fig. 3b) has a bigger growth gradient, achieving its maximum peak with a proxy capacity of only 25%. Meanwhile in a one-level system, the same peak is only achieved with a proxy capacity of 100% (the system then becomes an independent server architecture). In the same way, the balance-point between system and effective bandwidth is achieved with a proxy capacity of 8% as against the 17% (more than double) in the one-level proxy approach. These results also show that without the need for a main server, Double P-Tree architecture can operate by using proxies with a capacity of only 1% of system videos.



(a) One-level proxy-based system. (b) Double P-Tree system.

Fig. 3. Storage requirements in proxy-based architectures.



(a) LVoD systems bandwidth requirements. (b) LVoD systems storage requirements

Fig. 4. Performance and requirements in Large-scale VoD architectures.

5.3 Efficiency

We now compare Double P-Tree with the main current approaches for scalable LVoD systems. The centralized approach is only used as a reference, due to its null scalability and high costs, which are not consistent with our design goals.

In Fig. 4a, we observe that Double P-Tree improves one-level effective bandwidth more than 350%. Moreover, the biggest Double P-Tree network and server are 15 times smaller (4000 against 63.500), requiring similar proxy-storage (Fig. 4b) of around 3.500 Gigabytes.

Additionally, by now comparing results with the independent servers, it can be seen that this architecture obtains 4% more effective bandwidth than in our approach (508.000 against 490.000 Mbps), but uses 5 times more storage (17.200 against 3.500 Mbps), as shown in Fig. 4b. We believe that this perfomance gap would be resolved and even overcome by our approach when the effect of multicast techniques (the key to VoD system performance) is taken into account. Our optimism is due to the fact that multicast efficiency broadly depends on the number of users accessing the same server. In independent server systems, this number is limited by local-network size and, therefore, resource sharing (network streams, service bandwdith, etc..) is limited. Meanwhile, in Double P-Tree, distance-1 local nets can be served by the same proxy, increasing the number of users that can share system resources. In a topology with 7 brothers, sharing probability can be 10 times larger (number of neighbors) than in independent servers.

6 Conclusions

To achieve a full distributed system we proposed a hierarchical-tree topology of independent networks with proxies. This architecture distributes the main server functionality among proxies and modified proxy functionality, dividing its capacity between two schemes: caching and mirroring. The caching scheme stores the most requested movies, whilst mirroring is used for making a distributed mirror. Moreover,

to improve topology connectivity and system performance, we have modified the basic-tree topology by adding the concept of brother networks to local group nets.

The double P-Tree guarantees an unlimited and low-cost growth, high fault tolerance, a better load-balance (due to larger local-network connectivity), and a large streaming capacity independent of the technology available, and without requiring high storage requirements or complex and costly servers/networks.

The results show that new architecture outperforms streaming capacity (effective bandwidth) by more than 350%, compared with similar architectures (one-level proxies). In comparison with independent server systems, the effectiveness is similar but has 5 times less storage and allows for better resource sharing between users.

Our future research will focus on the study of proxy management policies that allow proxy size reduction. In addition, we intend to study the performance and changes required by policies such as prefix caching, chaining, patching and other classical policies for their subsequent incorporation into our architecture.

7 Bibliography

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, "Distributed Servers Architecture for Networked Video Services", in IEEE/ACM Transactions on Networking, Vol. 9, No. 2 April 2001.
3. F. Cores, A. Ripoll, E. Luque, "A fully scalable and distributed architecture for video-on-demand", in Procedings of PROMS'01, Twente, Holland, Oct 2001
4. A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," Multimedia Systems 4, pp. 112-121, June 1996.
5. D. L. Eager, M. K. Vernon, and J. Zahorjan, "Minimizing bandwidth requirements for on-demand data delivery", Proc. MIS'99 , pp. 80-87, Indian Wells, CA, Oct. 1999.
6. 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.
7. K. A. Hua and S. Sheu, "Skyscraper broadcasting: a new broadcasting scheme for metropolitan VoD systems," in SIGCOMM 97, pp. 89--100, ACM, Sept. 1997.
8. K. A. Hua, Ying Cai and S. Sheu, Patching: A multicast tecnique for true video-on-demand services, ACM Multmedia'98, pages 191-200.
9. F. Kon, R. Campbell, and K. Nahrstedt. "Using Dynamic Configuration to Manage a Scalable Multimedia Distributed System." Computer Communication Journal, Special Issue on QoS-Sensitive Distributed Network Systems and Applications, 2000.
10. A. Krikelis, "Scalable Multimedia Servers", IEEE Concurrency, pp. 8-10, Oct-Dec, 1998.
11. D.N Serpanos, A. Bouloutas, A, "Centralized versus distributed multimedia servers", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10 Issue: 8, pp. 1438-1449, Dec. 2000.
12. S. Sheu, K. A. Hua, and W. Tavanapong "Chaining: a generalized batching technique for video-on-demand systems", In Proc. IEEE Int'l Conf. On Multimedia Compunting and Systems (ICMCS)'97, Ottawa, Canada, June 1997, pp. 110-117.
13. F. A. Tobagi, "Distance learning with digital video," IEEE Multimedia Magazine, pp. 90-94, Spring 1995.
14. S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," Multimedia Systems 4, pp. 197--208, Aug. 1996.
15. 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.