An Experimental Dynamic RAM Video Cache

Nicholas J. P. Race, Daniel G. Waddington and Doug Shepherd
Distributed Multimedia Research Group,
Computing Department, Lancaster University,
Lancaster, UK
e-mail: [race, dan, doug]@comp.lancs.ac.uk

Abstract

As technological advances continue to be made, the demand for more efficient distributed multimedia systems is also affirmed. Current support for end-to-end QoS is still limited; consequently mechanisms are required to provide flexibility in resource loading. One such mechanism, caching, may be introduced both in the end-system and network to facilitate intelligent load balancing and resource management. We introduce new work at Lancaster University investigating the use of transparent network caches for MPEG-2. A novel architecture is proposed, based on router-oriented caching and the employment of large scale dynamic RAM as the sole caching medium. The architecture also proposes the use of the ISO/IEC standardised DSM-CC protocol as a basic control infrastructure and the caching of pre-built transport packets (UDP/IP) in the data plane. Finally, the work discussed is in its infancy and consequently focuses upon the design and implementation of the caching architecture rather than an investigation into performance gains, which we intend to make in a continuation of the work.

1. Introduction

The delivery of real time continuous media such as digital audio and video is becoming increasingly important in today's computing environment. However, the high data rates and strict delivery constraints that continuous media imposes, have proven to be difficult to meet in high demand situations. A wealth of research has been carried out over the past ten years to solve these problems, combining developments in efficient filing systems, highly optimised scheduling policies, admission control and resource management [1]. Whilst research has led to high performance servers, there are still complex issues surrounding the end-to-end delivery of audio and video across the Internet. The large data size not only places a substantial load on the network, but also represents a high cost for video distribution, particularly if expensive backbone links are involved in the delivery process. One approach to reducing this cost is the use of caching. By inserting cache nodes in the local network, popular videos or clips can be serviced from local caches, resulting in a reduced network load over the greater distance. Caching also brings additional benefits, in that videos streamed from the cache decrease the load on the server, allowing it to service other requests. As caches are located close to clients, there is also reduced latency in establishing connection set-ups.

Advances in hardware have brought about increased dynamic RAM capacities at a reduced cost; a pattern that is expected to continue for years to come. Coupled with the move to 64-bit architectures, there is now more potential than ever to use RAM as a medium for video storage. RAM brings many advantages to the real-time delivery of continuous media; not least the elimination of disk I/O bottlenecks and increased overall bandwidth. This provides the potential for supporting a very large number of concurrent accesses, and true Video On Demand (as opposed to Near Video On Demand) streaming. When serving a large number of streams, it is suggested that RAM provides a more economic solution than disks due to its higher bandwidth capabilities [2].

In this paper we present the design and preliminary implementation details of a cache node using main memory as the primary caching medium. In order to maximise throughput performance between RAM and the network interfaces, the node uses pre-built IP, which stores video data in memory in a pre-packetised format ready for immediate transmission by the network layer, thus avoiding the need for fragmentation or packet/header assembly. Much of the implementation focuses upon the use of MPEG-2 as a streaming media type. This is an internationally standardised format for broadcast quality digital video that is generic in its deployment capabilities.

The paper is divided into the following sections. Section 2 provides an overview of the caching system from an architectural perspective, describing envisaged deployment scenarios and integration into the data and control environment. Continuing, section 3 discusses some of the implementation aspects and the realisation of

the caching node in Windows 2000. Finally, sections 4 and 5 overview some related work in the area and outline some directions of interest for future work.

2. Caching architecture

This section provides an overview of the caching architecture, describing envisaged deployment scenarios and integration into the control and data environment.

2.1. Transparent caching

Traditional web caching is achieved through the use of a proxy server, physically located between a client web browser and a server. The proxy intercepts all packets, and examines each one in order to determine whether it can service the request itself, or whether an additional request has to be made to the server. However, proxies generally need to be explicitly configured within the web browser for each client, which presents a large deployment cost and a non-scalable solution with respect to service provision. A fundamental objective of the caching node is that it should be completely transparent to both video client and server so that no additional cost of ownership is incurred, hence making the node suitable for wide-scale deployment.

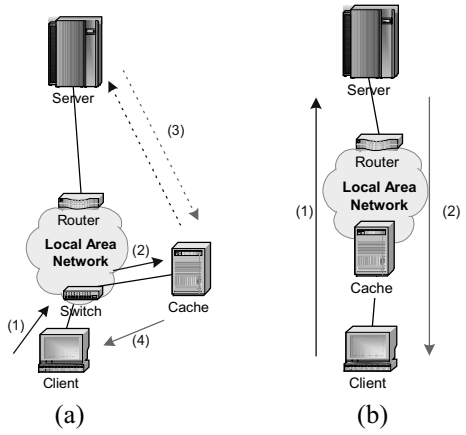

Figure 1. a) Transparent caching using L4 switches
b) Proposed transparent approach

A number of approaches to transparent caching were considered, the most common of which use L4 switches or policy-based routers. In this case, user requests for web pages are diverted by a router/switch to a local cache, and all other network traffic forwarded as normal. If the requested data is not available in the local cache, a separate TCP connection is established to the web server in order to retrieve (and store) it. The data is then returned to the client, with any subsequent requests for the same data serviced from the local storage of the cache.

Our proposed approach incorporates IP routing functionality within the cache node itself. It is assumed that cache nodes are deployed within the local area network close to clients (rather than in the core) and therefore do not perform the intensive routing operations associated with core routers. The inclusion of routing functionality means that the cache node must perform filtering of the appropriate video control and data packets, which could be considered to be an avoidable overhead. Nevertheless, the advantage to this approach is that both the server and client are unaware of the presence of the cache, and that no configuration changes are required by the end-systems in order to forward requests to the cache. In a commercial realisation of the architecture we would expect to take advantage of router hardware filtering to carry out some level of traffic redirection (such as Layer 4 switching or policy-based routing) in order to divert traffic towards the cache node.

2.2. Topology

The basic networking architecture is shown in Figure 2. It consists of a continuous media server (able to support a number of concurrent stream accesses) connected via a high-speed interconnect to a backbone router. The cache node is installed in each LAN, acting as an IP router and is directly attached to the backbone router connecting the LAN to the WAN. The proposed architecture uses UDP/IP over an ATM-based IntServ/DiffServ infrastructure, and assumes negligible packet loss on such connections. We believe that UDP is the preferable choice over TCP since it does not provide error correction and control, and is connection-less and therefore ideally suited to interception and masquerading.

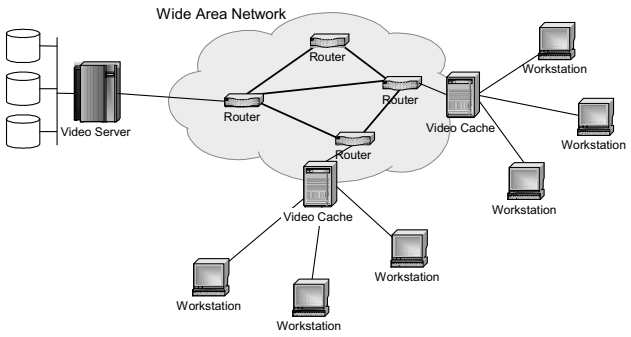

Figure 2. Network topology

The caches are able to use pre-built transport units, in this case UDP packets, as a unit of caching. This avoids both the need for user level processing and the need for re-assembly of data leaving the node. Because of the simplistic and connection-less nature of UDP, packets can be easily sent from the cache node in a form of

masquerading whereby the client is unable to determine the originality of the data and thus the caching appears to be transparent.

2.3. Integration with multimedia data architecture (MPEG-2)

The MPEG-2 (Moving Pictures Experts Group) ISO/IEC 13818 standard [3] is designed to provide high quality video by exploiting spatial and temporal redundancies in the input source in order to achieve compression. MPEG-2 has been widely adopted as the standard for digital video, and is used by DVB (Digital Video Broadcast), DAVIC and DVD.

The MPEG-2 systems layer (ISO/IEC 13818-1) specifies two mechanisms for the multiplexing and synchronisation of elementary video and audio streams to form a single data stream that is suitable for storage or transmission. Each mechanism is tailored for a different operating environment. The first scheme, known as a *Program Streaming*, uses large packets of variable sizes and is designed for use in a largely error-free environment. The scheme is similar to the MPEG-1 multiplexing standard and can support only one program (a number of elementary video/audio streams with a common time base). The second scheme, known as a *Transport Streaming*, can combine multiple programs with independent time bases into a single stream. Transport Streams use fixed length 188-byte packets, with additional error protection and incorporate timestamps within the packets to ensure correct synchronisation. They are intended for use in error-prone environments.

The caching architecture is designed around the use of MPEG-2 Program Streams as the principal data format. This format, unlike Transport Streams, does not provide features for error correction and control. However, in our experimental environment, this is not an essential requirement.

Program Streams are constructed from one or more Packetised Elementary Stream (PES) packets. A PES packet consists of a header and a payload. The header contains important timing information in the form of a *Presentation Time Stamp* (PTS) and a *Decoding Time Stamp* (DTS), which is used to ensure correct synchronisation at the decoder. The header also contains a *stream_id* field in order to distinguish one elementary stream from another within the same program. The payload consists of the video and audio data bytes that have been encoded from the original source stream. In a Program Stream, PES packets are arranged into logical groups known as 'packs'. A pack consists of a pack header, an optional system header and any number of PES-packets. The pack header also contains important timing information in the form of a S*ystem Clock Reference* (SCR). The PES stream structure is shown in Figure 3 below.

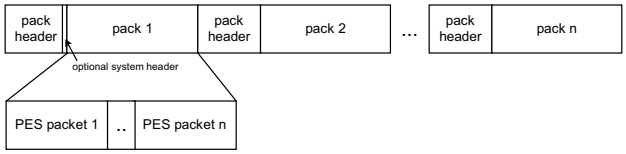

Figure 3. Structure of MPEG-2 Program Stream

2.4. Integration with multimedia control architecture (DSM-CC)

In designing a caching architecture, integration into the media control architecture is essential. In this work we have adopted the ISO standardised Digital Storage Media – Command and Control (DSM-CC) protocol [4]. This protocol is a specific application protocol, intended to provide the basic control functions and operations to manage digital storage bit streams akin to MPEG-2. It is designed for the command and control of retrieval/storage applications such as video-on-demand, interactive video services and electronic publishing.

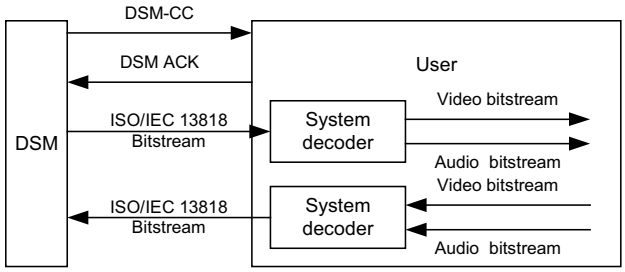

Figure 4. Configuration of DSM-CC

DSM-CC can be used in one of two modes, either as a stand-alone control protocol which is used in parallel to the transmission of the bit streams, or alternatively embedded in the data stream itself. Both approaches packetise the DSM requests and acknowledgements as PES packets. The formal specification does support a richer command and control model through the use of high level RPC (such as provided by CORBA or COM). It also defines both User-Network and User-User protocols. However, for our own purposes, consideration for the User-Network DSM-CC requests and acknowledgements is sufficient.

In the experimental platform, DSM-CC is used in stand-alone mode (i.e. it is not embedded within the data stream). It is encapsulated in PES packets and transmitted over UDP/IP. To initiate simple playback a client issues a bit stream select command to a given DSM server. This select command carries a bit stream identifier

corresponding to an ISO/IEC 13818 stream. Providing the server holds the appropriate stream, a select acknowledgement is returned (to the sender's UDP port). The client can now control the stream (play, stop, pause, resume and jump) through subsequent retrieve commands.

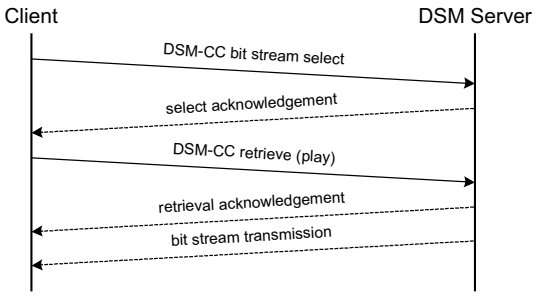

Figure 5. DSM-CC call sequence

Within the caching topology as previously described, the cache server intercepts DSM-CC requests and if it is able to provide the desired bit stream from cache, then masquerades as the server itself (this is possible because of the simplicity of the control architecture), otherwise the DSM command is forwarded to the appropriate server. In some scenarios the cache may be able to provide a portion of the bit stream (termed a *partial hit*). Further discussion of partial cache hits and their handling is given in section 2.6.

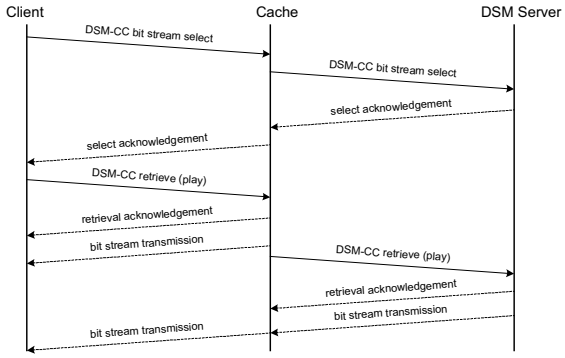

Figure 6. Example command sequence for partial hit

To determine which UDP packets make up a given bit stream, the caching node must assume that all PES packets originating from the DSM server, to a given port and address, constitute the same bit stream until an MPEG-2 end code is received. It is therefore necessary for the cache to 'snoop' the UDP payload during the caching process. Where IPv6 is deployable, the cache is able to use the flow label instead, in which case the need for snooping is avoided.

2.5. Caching behaviour and replacement policies

Because of the large size of MPEG-2 video objects (typically in the order of a number of megabits per second, per film), it is intended that popular portions of large video objects and complete video clips are cached. The behaviour of the cache is controlled by a set of policies that dictate what to do in the event of a cache miss, a cache hit or a lack of available cache memory. As the memory within the caching node becomes full, the cache must decide which data should be retained and which should be discarded. A wide range of cache replacement policies are under consideration for use within the caching architecture. Traditional caching research provides a number of alternative strategies to cache replacement. Commonly used examples include LRU (Least Recently Used), which discards the data that was accessed the longest time ago. It is based on the theory that data accessed recently is the most likely to be used again in the near future. Alternatively LFU (Least Frequently Used) discards the data that has been accessed the least over time.

Further caching policies can be brought from work on buffering and caching within multimedia servers. They exploit the fact that multimedia objects are typically accessed in a sequential manner. Blocks retrieved for one client can therefore be reused for subsequent requests within a short time interval [5, 6].

However, it is clear from an analysis of logged accesses to video data [7] that the initial portion of a video is often used to determine a users interest. The importance of a replacement policy in maintaining the initial portion of a video within the cache is affirmed in [8], where it is suggested that using a proxy prefix caching technique to store the initial frames of popular clips hides latency, throughput and loss effects between the server and the proxy cache.

2.6. Partial cache hits

Unlike traditional caching, an entire object does not have to exist within the cache for a request to be satisfied. A partial hit may occur when a request is made for data that partly exists within the cache. This can arise if a previously cached object is no longer complete, for example when a portion of the object has been replaced in accordance with a specific cache replacement policy (e.g. the "tail" of a video stream is no longer present in the cache). The cache can service the partial hit and simultaneously make a request to the server for the remainder of the stream. A transparent 'hand over' can then take place at the appropriate time. Providing the request is made at the appropriate time, the retrieved stream can be optionally cached and then forwarded.

3. Experimental design and implementation

A significant contribution of this work is in the realisation of the system within a prototype environment. Although the prototype is still in its infancy, already the design and implementation have provided a number of additional insights into the problems faced in network caching strategies. The prototype cache node is based on extensions to the Microsoft Windows 2000 Advanced Server operating system, with the caching mechanisms implemented as a kernel module (device driver), which interacts directly with the communications protocol stack. In terms of client/server networking infrastructure, the prototype cache node is connected as a gateway router between two private IP sub-networks. Consequently, communications between given client/server nodes are forced to pass through the cache. The video distribution application used within the experimental environment is a proprietary application providing simple MPEG-2 streaming over UDP/IP, with an MPEG-2 hardware decoder card running under Linux as the client.

3.1. Real-time IP interception and injection

The cache node is designed to cache pre-built IP packets. Because the core IP protocol processing is provided by the kernel, we are able to execute the necessary caching interactions without entering the upper layers. This technique helps to keep the processing carried out by the cache to a minimum and thus enable the task of caching to be executed in real-time. This is essential since the period required to filter and cache a packet must be less than the inter-packet interval. If this is not achieved the cache will act as a bottleneck and consequently back up traffic at the source (this is comparable to the notion of non-blocking network switches).

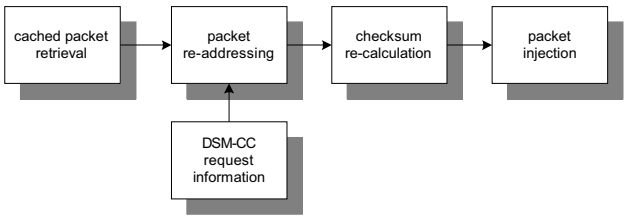

Figure 6. Packet re-send processes

Before cached packets can be sent back out into the network, a process of adaptation or 'moulding' must take place (see Figure 6). The primary purpose of this is re-addressing and the adjustment of any checksum fields. New address fields for the IP packets are determined from the intercepted DSM-CC packets. The initial prototype implementation uses IPv4, whilst adoption of IPv6 is underway.

3.2. Cache entries/indexing

An entry in the cache, known as a cache block, is made up of an informational header followed by a number of pre-built IP packets, made up of an IPv4 or IPv6 header and a series of MPEG packets within a UDP payload. Cache blocks are uniquely identified for storage and retrieval through a combination of *bitstream_ID* and *presentation time* (each 32 bits wide). The *bitstream_ID* uniquely identifies a given media stream and is assigned by the DSM-CC server which maps them to more meaningful names. The presentation time, in units of a second, is derived from the MPEG payload in the first IP packet of the cache block.

The proposed caching architecture uses a two level indexing scheme to hold the mappings between the bitstream_ID/presentation time and the physical address of the cache block. The first index, termed the cache index, provides a mapping between the stream identifier and the address (virtual) of the cache table. This index has *n* entries, where *n* is the total number of unique streams distributed by the server(s). Each cache table, per entry in the cache index, consists of an array of physical addresses corresponding to each cache block stored in the cache area for the given stream. This scheme is strikingly similar to indexing traditionally found in virtual memory management, and in fact we envisage the future exploitation of such hardware support in a 64-bit platform.

Because the scheme provides direct mappings, the resources required by the indices may be potentially large. Nevertheless, we are making the assumption that the reducing factor is the total number of unique streams concurrently handled by the system (in a more dynamic media environment, where content is frequently changed and therefore a large number of different streams may exist, a process of re-mapping would be required). Consequently, the cache node need only maintain a cache index capable of addressing the total number of unique streams that may exist in the cache at any given time. Furthermore, hardware and cost limitations means that this is constricted by the 'capacity' of the node. Our initial prototype provides ~2Gb of caching memory, capable of storing approximately 68 minutes (4096 seconds) of media. Thus the maximum total overhead of the cache tables, assuming an average stream rate of 4Mbps, is only 32Kb.

3.3. Caching memory sub-system

The nature of the caching architecture and its demands of memory management are significantly different than the

requirements found in general purpose operating systems. Allocations, corresponding to individual cache blocks are relatively large (ranging from 256K – 1024K). Blocks may exist outside the range of the virtual address space, at least within 32-bit architectures, which only provide 232 virtual addresses (4Gb). Furthermore, this problem is exacerbated by the operating system's use of reserved areas, which within the Windows 2000 operating system, only leaves somewhere in the region of 512Mb of virtual addresses available for kernel modules. To address this problem, the proposed design incorporates a specialised memory sub-system that offers memory management services specifically tailored to the task of video caching. The subsystem includes features such as Large Scale Addressing (beyond 4Gb of physical memory), Dynamic Address Mapping (mapping virtual addresses to cache blocks on demand) and partial frees (enable previously allocated blocks to be resized in a zero copy manner).

4. Related work

Research into buffering and caching within multimedia servers exploits the fact that multimedia objects are typically accessed in a sequential manner, which means that blocks retrieved for one client can therefore be reused for subsequent requests within a short time interval. Such techniques include *interval caching* [5] whereby intervals between successive streams are cached, and *distance caching* [6], which replaces blocks of data based on the distance between successive clients.

Research into caching multimedia streams within the network stems from the work in managing a distributed hierarchical video-on-demand system. The Berkeley VOD System [9] is designed to provide transparent access to large amounts of video material. Continuous media objects are stored on tertiary storage devices, and only copied to a file server when required. The MiddleMan architecture [7] is a collection of cooperative proxy servers that, as an aggregate, cache video files within a local area network. The design incorporates a central coordinator that is responsible for managing the video files stored at each proxy, controlling the storage and replacement of files and redirecting requests accordingly. An architecture for caching streaming media (known as SOCCER) is defined in [10]. It is based around the concept of cooperative helpers that are situated within the network and collectively perform caching.

In addition to minimising start up latency, caching multimedia streams can also smooth the playback of variable bit rate (VBR) video streams. Video streams exhibit burstiness due to the encoding scheme and variations within and between frames, which can be a problem in terms of buffer management and network utilisation. Proxy prefix caching [8] overcomes this by using an intermediate proxy that caches the initial frames of popular audio/video clips and performs work ahead smoothing of variable bit-rate streams in order to reduce the resource requirements from the proxy to the client. A similar technique known as video staging [11] pre-fetches and stores selected portions of VBR streams in a proxy. The aim is to reduce backbone bandwidth requirements by storing the bursty portions of a VBR stream within the proxy and combining them with a constant bit rate (CBR) video stream from the server for playout. Another prefix caching scheme [12] also caches intermediate frames based on the encoding properties of the video stream and the users buffer size. It attempts to store frames within the cache that are more critical to maintaining the robustness of the stream. [13] considers an end-to-end architecture for the delivery of layered-encoded streams in the Internet using proxy caches to smooth out variations in quality by pre-fetching segments that are missing from the cache.

Much of the work on proxy cache replacement strategies is tailored to HTML documents and images, and does not consider the impact of multimedia streams [14]. However, [15] provides an investigation into multimedia streaming and cache replacement policies, introducing a caching algorithm based upon the resource requirements of an object. Finally, [16] provides one of the few investigations into the use of main memory for caching, using trace-driven simulations to evaluate the performance benefits of main memory caching for web documents.

5. Further work

Work to date focuses on the use of large-scale RAM as a caching medium. It is proposed that this be extended to incorporate fast access disks as an additional level to a more hierarchical caching approach. However, the cache node's disk storage will only be used as an intermediary before cache content is totally dropped. It will not be used as a source medium for streaming.

Future work will also examine the implications of other media types, in particularly ISO/IEC MPEG-4 [17]. This is a multimedia format that is object-oriented and lends itself to partial caching. We envisage using media 'objects' as a unit of caching, and the distributed gather of such objects to form a scene. The adoption of such media types also opens up other areas of research interest. One such area is co-operative caching, where by caching nodes within the network use some proprietary inter-nodal protocol to handoff cache requests to other nodes in the event of a local cache miss.

Finally, the ISO standardised DSM-CC protocol has been adopted within the caching architecture for media control and playback. Future work will investigate extending this to include support for RTSP [18].

6. Concluding remarks

This paper presents ongoing work examining the initial design and implementation of a network caching architecture for high quality video, and more specifically ISO's MPEG-2. The discussed architecture is based on the notion of a transparent caching node, situated at the edge of the network, which is able to masquerade as a remote video server to its clients. This transparency lends itself to simple management and straightforward integration into the majority of existing network topologies.

The cache node, based on the Windows 2000 operating system, exploits high-speed main memory to cache pre-built UDP/IP packets containing MPEG-2 data packets. This technique of using transport/network level data units avoids the transport layer re-fragmenting the video data, thus increasing performance. In order to deploy such a cache, it is essential that the cache is able to integrate and cooperate with the video control architecture. In our prototype implementation, we have chosen to support DSM-CC as the basic control protocol. The cache behaves as a DSM proxy to the unaware client, intercepting control requests and forwarding/handling this as necessary.

Acknowledgements

We acknowledge the kind support of the Microsoft Research Labs, Cambridge, UK in funding this research under the LandMARC Research Project.

References

[1] D. Gemmell, H. Vin, D. Kandlur, P. Rangan and L. Rowe, "Multimedia Storage Servers: A Tutorial", *IEEE Computer*, Vol. 28, No. 5, May 1995.

[2] M. Kumar, "Video-server designs for supporting very large numbers of concurrent users", *IBM Journal of Research and Development*, 1998, Vol. 42, No. 2, pp. 219-232.

[3] ISO/IEC 13818-1, "Generic Coding of Moving Pictures and Associated Audio Information: Part 1 - Systems", Information Technology Specification, International Standard, 1996.

[4] ISO/IEC 13818-6, "Generic Coding of Moving Pictures and Associated Audio Information: Part 6 - Extension for DSM-CC", Information Technology Specification, International Standard, 1996.

[5] A. Dan, D. Sitaram, "Multimedia caching strategies for heterogeneous application and server environments", *Multimedia Tools and Applications*, pp.279-312, 1997.

[6] B. Ozden, R. Rastogi, A. Silberschatz, "Buffer replacement algorithms for multimedia storage systems", in Proceedings of the International Conference on Multimedia Computing and Systems, pp. 172-180, June 1996.

[7] S. Acharya, "Techniques for improving multimedia communication over wide area networks", Ph.D Thesis, Department of Electrical Engineering, Cornell University, January 1999.

[8] S. Sen, J. Rexford and D. Towsley, "Proxy prefix caching for multimedia streams", in Proceedings of the IEEE Infocom, 1999.

[9] D. W. Brubeck and L. A. Rowe, "Hierarchical Storage Management in a Distributed VOD System", *IEEE Multimedia*, Fall 1996, Vol. 3, No. 3.

[10] M. Hofmann, T.S. Eugene Ng, K. Guo, S. Paul, H. Zhang, "Caching Techniques for Streaming Multimedia over the Internet", Bell Labs Technical Memorandum, April 1999.

[11] Y. Wang, Z.-L. Zhang, D. Du and D. Su, "A network-conscious approach to end-to-end video delivery over wide area networks using proxy users", in Proceedings of the IEEE Infocom, April 1998.

[12] Z. Miao and A. Ortega, "Proxy caching for efficient video servers over the Internet", in Proceedings of the 9th International Packet Video Workshop (PVW '99), New York, April 1999.

[13] R. Rejaie, M. Handley, H. Yu and D. Estrin, "Proxy caching mechanism for multimedia playback streams in the Internet", in Proceedings of the 4th International Web Caching Workshop, San Diego, California, March 31-April 2, 1999.

[14] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, E. A. Fox, "Caching Proxies: Limitations and Potentials", in Proceedings of the 4th International World-Wide Web Conference, pp. 119-133, Dec. 1995.

[15] R. Tewari, H. Vin, A. Dan and D. Sitaram, "Resource based caching for web servers", in Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking, San Jose, 1998.

[16] E.P. Markatos, "Main Memory Caching of Web Documents", *Computer Networks and ISDN Systems*, 1996, Vol. 28, No. 7-11, pp. 893-905.

[17] ISO/IEC 14496-1, "Coding of Audio-visual Objects - Part 1: Systems", Information Technology Specification, International Standard, 1999.

[18] H. Schulzrinne, A. Rao and R. Lanphier, "Real Time Streaming Protocol (RTSP)", Request for Comments (Proposed Standard) 2326, Internet Engineering Task Force, April 1998.