Client-Server Metadata Management for the Delivery of Movies in a Video-On-Demand System[1]

T.D.C. Little and D. Venkatesh

Multimedia Communications Laboratory
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877, (617) 353-6440 fax
{*tdcl,dinesha*}*@bu.edu*

MCL Technical Report 12-14-1994

Abstract–The Internet is experiencing an explosion in the number of connected users. Beyond simple mechanisms of FTP and electronic mail, distributed information services provided by WAIS, Gopher, Archie, WWW, etc. are rapidly changing the operating view of the Internet. This change is further expedited by the falling costs associated with terminal equipment that can handle continuous media such as audio and video. The support of bandwidth-intensive and storage-intensive media in a distributed environment requires rethinking both the raw data transmission infrastructure as well as information dissemination mechanisms.

In this paper we describe a mechanism for location, identification, and delivery of continuous media in the form of digital motion pictures in the context of a distributed system. The metadata mechanism and supporting protocols are based on the client-server model. The mechanism can be applied to numerous application domains including multimedia-based home entertainment, catalog shopping, distance learning, and distributed-interactive classrooms. We demonstrate the practicality of our the mechanisms though a prototype application based on home entertainment.

Keywords: Internet-based services, metadata management, client-server architectures, video-on-demand.

[1]In *Proc. 1st Intl. Workshop on Services in Distributed and Networked Environments (SDNE)*, Prague, Czech Republic, June 1994, pp. 11-18. This material is based upon work supported in part by the National Science Foundation under Grant No. IRI-9211165.

1 Introduction

The Internet is expanding at a very rapid pace, with almost a million new users being added every month [3, 11]. Many services are being added to satisfy the varying needs of the different groups of customers who use the network. Some of these services make use of continuous media like video and audio that typically consume enormous bandwidth, and strain the resources of the network as well as local data stores [2].

However, technology has kept pace with this demand, and different communication services are being developed that will be able to support the new demands being made from the network. This communication ability, coupled with the decrease in memory costs, and increase in storage device capacities will create an excellent opportunity for information providers to develop applications that will use distributed resources. Interactive multimedia applications are fast becoming the most popular applications being developed to exploit the new infrastructure. These applications which include services such as video-on-demand (VOD), distance learning, catalog browsing, etc. will soon become the prime consumers of network resources [3, 11].

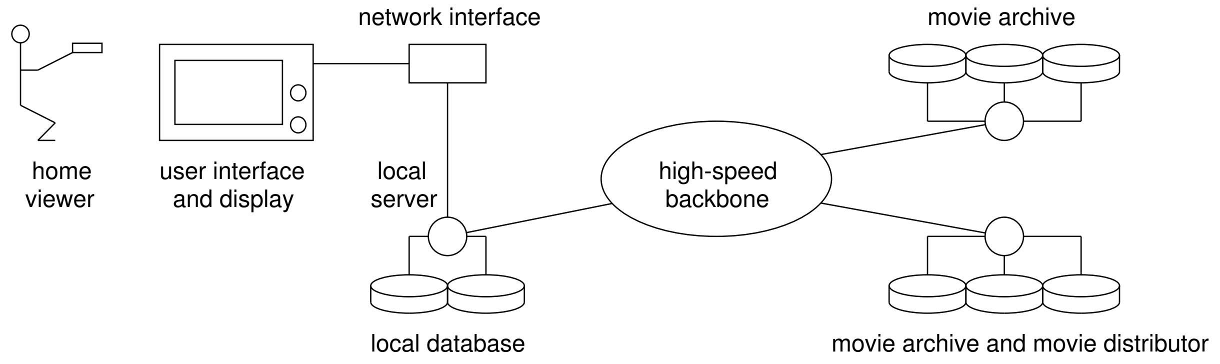

Figure 1: A Distributed Database Scenario for VOD

As these bandwidth and memory intensive applications evolve, it becomes very important to develop schemes which efficiently utilize the system resources while simultaneously providing the best service to their users [1, 4, 12, 13]. To be successful, these applications will need to make enormous volumes of data easily accessible to their users. To make the handling of data manageable, the information will need to be distributed across different sites. One view, specific to the distribution of home entertainment, is shown in Fig. 1 [7].

This model consists of a local database connected via a (generic) high speed backbone network to archives where multimedia information are stored. The local databases cache the information locally (on a very large scale), and make them available to their users on an on-demand basis [16].

The choice of data distribution is based on the desire to support many users. The support of a single VOD session/user requires substantial bandwidth [13]. To support many users, significant system communication and storage I/O are required. Local storage on a single user computer is not feasible due to the high cost per user. The distribution of data and use of multiple access points increases the system's overall I/O capacity to support interactive sessions. Furthermore, a distributed system can have more access points at which users can enter the system. Experimental testbeds and deployments to date are either simplistic in functionality, or do not address issues of scale [5, 10, 14, 15]. Future systems of this type must support thousands of users.

Using this system model we seek to support information access and delivery services for interactive multimedia applications. To this end, we investigate service mechanisms for building such applications which require location, identification, and delivery of multimedia data. More specifically, we examine services for retrieval of distributed multimedia data in a client-server architecture, in which a server coordinates different client requests. We also present an experimental prototype we have developed in our laboratory which demonstrates the aforementioned concepts.

The remainder of this paper is organized as follows. In Section 2, we develop our query architecture based on the concept of resource and metadata servers. Section 3 describes the application of these concepts in the context of a client-server VOD architecture. In Section 4, we describe the prototype testbed we have built in our lab. Finally, Section 5 concludes the paper.

2 Services for Supporting Interactive Multimedia Applications

A critical issue in the design of multimedia information systems is the data distribution architecture. Due to the enormous volumes of data involved, it is unwieldy to completely store and manage all information distribution from a single site. Furthermore, the interactive nature of the queries to the centralized database requires the database manager to provide both database management and continuous media delivery functionality.

Services required to support interactive multimedia allocations include data distribution, metadata management, and resource management. These are described in the following subsections.

2.1 Metadata Management

To simplify the query mechanism, and increase interactivity, most database systems employ a "metadata" mechanism [6]. Metadata are "data about data." In our context, this means that they contain concise information about the location and characteristics of the data to be retrieved (e.g., movie titles, where they are stored) [8]. By using such a scheme, the data delivery process can be decoupled from the database management functions.

The advantage of this metadata approach is that it allows the user examine the contents of a database without having to retrieve the large objects. The user can quickly go over the an information summary and to the select the information to be delivered. This concept is particularly appealing with respect to stored audio and video which can be costly to deliver and can consume significant portions of a data server's I/O bandwidth.

There are several requirements for the metadata management scheme necessary to support interactive multimedia applications. First, we require basic services to support object identification, location, naming, and distribution [9]. Second, application-specific models are required to support application functionality (e.g., searches on movie scene content). In the former, we can also consider the limitations of available I/O bandwidths for individual storage devices at data servers. This limitation can be managed by a central site in the form of a "bean count" of sessions in progress (e.g., movies played out off a specific server). The attributes managed by the central site include:

- location,
- availability,
- current state (e.g., busy/free), and
- characteristics (e.g., data rates, duration).

The metadata concept is ideal for interactive multimedia applications in which many objects can be involved in a single presentation. For example, a movie might involve the coordination of audio and video streams. A classroom video might involve the simultaneous presentation of audio, video, graphics and text. The metadata server must maintain the relationships between these diverse objects and inform their presence to the client whenever a query is made. In the context of a VOD server, the system must store different movie attributes. Movie-specific attributes include entities such as *actor, producer, title, shot, scene,* etc. [8].

2.2 Distributed Resource Server

For a movie database, resource attributes include the parametric loads on individual storage subsystems and subnetworks; and the number of users currently accessing the VOD system. Maintaining a record of the parametric loading and system usage time is the responsibility of the resource server via a state table. Since resource allocation decisions are based on this information, it is important that it be kept up to date.

In information delivery systems, where the primary function of the database is data retrieval, the resource and metadata servers must keep track of available movies and resources to provide their delivery. When resources are overloaded or not available, the metadata server will have to find alternative sources of information, if available, or inform the user of their non-availability. These functions need to operate transparently to the user.

In order to manage its resources, the resource server must collect periodic input from the managed systems. These can include event reports, as well as periodic answers to queries from the various data servers. The resource server also needs to keep track of changes to the system configuration which can include resource relocation, failure, and the announcements of new resources.

It is envisioned that many interactive multimedia services will require their users to pay for offered services. User data is maintained in a separate database, and updated whenever a user accesses the system. This can be used in the generation of accurate and automatic billing mechanisms.

The main attraction of using metadata and resource servers are their flexibility. When resource updates occur (resource removal, relocation, and new resource generation), only the entries in the database need to be updated for the change to be visible to the entire user community. Finally, the metadata server itself can be replicated, providing for an additional level of performance.

2.3 Service Scaling via the Resource Server

Using the metadata and resource server models provides us with a simple scheme that quite powerful. As it uses database entries to maintain the relationships between objects, the metadata can be easily extended to support different media formats by object redirection. This is particularly appealing when supporting heterogeneous terminal devices with different display and presentation devices.

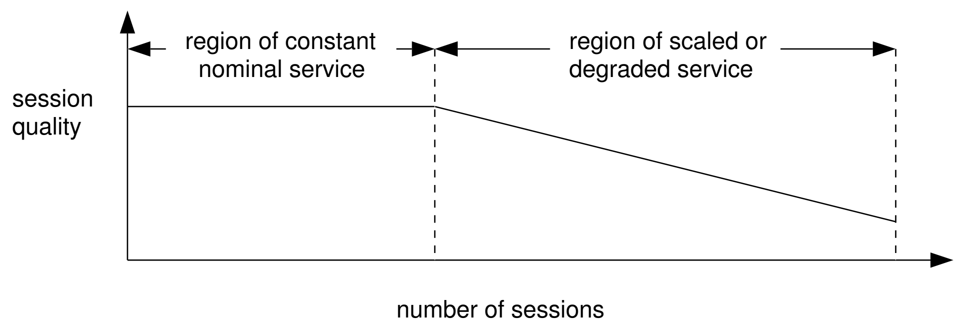
region of constant nominal service
region of scaled or degraded service
session quality
number of sessions

Figure 2: Quality vs. Number of Sessions in a VOD system

The resource server can also be used to provide graceful degradation of service when the system overloads. For example, multimedia storage servers have a limited capacity for supporting concurrent sessions. When this capacity is exceeded, the session quality can degrade abruptly (i.e., response time for interaction and bandwidth for data transmission). However, knowing these limits enables the resource server to either ensure graceful degradation, or to find alternate sources of replicated data and I/O bandwidth. This concept of graceful degradation is illustrated in Fig. 2.

3 Resource Management for a Client-Server VOD Architecture

We now describe our proposed scheme for managing metadata and resources for a VOD system. We base our discussion around an application for interaction with a database of motion pictures. A model for a distributed VOD architecture is illustrated in Fig. 3.

Different clients connect to a video database/servers (VDBs) via a network. The databases can be distributed in nature. A metadatabase server called the Query Database (QDB) pro-

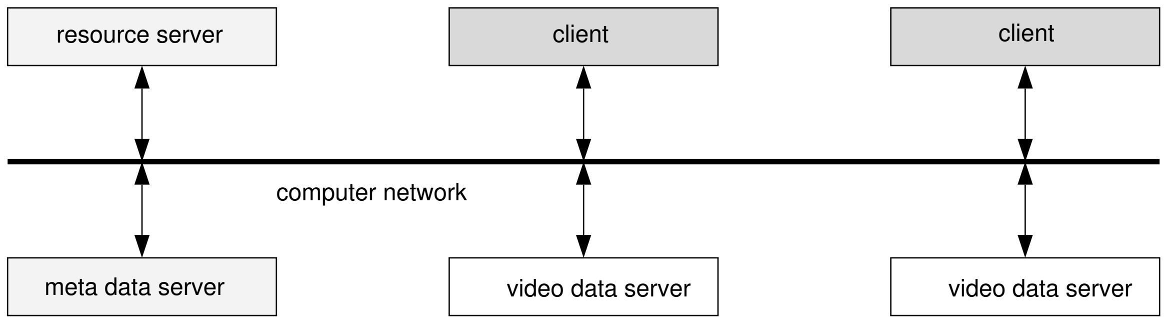

Figure 3: VOD Archtectural Model

vides interactive database access functionality for the application-specific data. A resource server maintains the availability of different resources in the system. When a user at a client wishes to view a movie, a query is made first to the QDB for interpretation of the user browsing operations. The resource server maintains tables about the availability of movies for playout. The delivery of the movie data to the user not begin until the user has browsed the selections, issued a command to “play” a movie, and has been given permission to do so from the resource server. In the remaining subsections we examine the connection establishment and maintenance functions necessary to support these processes.

3.1 Connection Establishment

At startup, the VDB maintains a process that waits for a requests from the resource server to allocate new connections to the storage resources. When a request arrives, the VDB decides if it can support a new connection. If it can, it responds to the client and resource server to start QOS (quality of service) negotiations. If not, the connection is closed.

When the QOS negotiation phase is entered, the server indicates to the client station the characteristics of the video to be played out. The server and the station then negotiate the QOS parameters.[2] Once this phase is complete, the server and the client proceed to the connection setup phase.

The connection establishment phase begins when the user decides on the movie to view (e.g., after browsing the database the user might press a “play” button for specific movie). During this phase, the user’s machine (client) has knowledge obtained from the QDB describing the the location of the VDB containing the desired movie. This information is

[2]This stage is not currently implemented on the VVB prototype.

Table 1: Connection Setup Primitives for the VOD Server

check_no_connections()	Checks to see if a new connection can be supported.
indicate_connection_setup_failure()	Indicates that the connection was unsuccessful.
start_QOS_negotiation(video_name, client_name)	Begins QOS negotiation.
retrieve_video_chars(video_name)	Retrieves the video characteristics for QOS negotiation.
indicate_QOS_needs(max_rate, min_rate,...)	Indicates QOS Requirements.
increment_no_users()	Reserves resources.
setup_connection(video_name, client_name)	Spawns a child to setup dedicated UDP connections.

transparent to the user.

When the a specific movie is requested for playout, a client process sends a request to the resource server for the establishment of a connection. The connection supports real-time video delivery and playout.[3] The session between the client and the server lasts the entire duration of playout of the movie. The client's request is a simple signal indicating the "name" of the movie to be displayed. The server checks its local status (the "bean count") to make sure that it will not be overloading the system, and can support movie playout for the entire duration of the feature presentation. To do this, the server must maintain the number of users connected at any given time. Alternatively, this information can be part of the QDB, in which case, the QDB makes sure that the database being accessed has enough residual capacity to support the new connection. If the connection cannot be supported, the server sends back a "connection refused" message.

If the connection can be supported, the server then proceeds to examine the characteristics of the required movie from the VDB. Depending on these characteristics, the server then starts a QOS negotiation procedure. If the negotiation proceeds to a successful completion, and mutually acceptable QOS parameters are agreed upon, then the connection setup phase begins. Otherwise the connection is terminated and the resources freed up. During the QOS negotiation phase, resources are reserved for the new connection, and will be released only if the connection setup fails. This ensures that there are no deadlocks at the server.

Once the QOS parameters are established, the client sends a "connection established"

[3]We do not assume a large storage capacity at the client station. Alternate architectures, in which the movie is transferred to the users site completely before playout begins are not considered here.

Table 2: Connection Setup Primitives for the VOD Client

query_video_database(database_name, video_name)	Queries the VDB for the specified video.
examine_video_statistics(video_params)	Checks to see if the connection can be sustained.
acknowledge_conn_acceptance()	Starts the actual connection.
close_connection()	Closes the connection.
setup_playout_and_wait_for_connection()	Initializes the playout hardware, and wait for the connection call to come from the remote server.

message. On the receipt of this message the server increments its state description to account for the new user. It also initiates a new process to handle the transfer of video data.

The functions that are required at the client and server machines for connection setup are summarized in Tables 1 and 2.

Table 3: Connection Management Primitives for the VVB

playout_video()	Indicates to both the client and the server to ensure that frames are played out at the proper rate.
forward_video()	This is an indication to the server to playout every fourth frame in lieu of consecutive frames. (note: forward and reverse do not really affect the playout mechanism at the clients end).
reverse_video()	Analogous to forward_video.
pause_video()	This is to both the playout mechanism, and the server, to stop information flow, as well as freeze the current frame.
change_volume()	This is only to the playout mechanism.
stop_playout()	Stop stops all data traffic, and is similar to freeze, except that in freeze, a single frame is displayed. Stop does not release the resources for a finite time, just in case the viewer decides to resume viewing. After this period, it exits the program by calling close_connection().

3.2 Connection Management

The connection maintenance mechanism manages the connection during the entire play-out duration of a movie. It ensures adherence to the QOS parameters negotiated during connection establishment. The connection management mechanism is also responsible for continuous media flow control and user temporal access control (TAC) operations such as *play, stop, pause, forward*, and *rewind.*

During the connection setup phase, a dedicated connection is set up between the network station and the VDB.[4] Flow control can be facilitated by an out-of-band feedback circuit.[5] Connection management processes are set up simultaneously at connection establishment. They are aware of the QOS parameters agreed to at the time of connection setup and enforce this behavior from the system. User TAC inputs which include typical VCR-like controls are translated and sent as control signals to both the server and the playout mechanism. Controls such as adjustment of volume are dealt with locally via the playout mechanism at the client. When the user presses the "stop" button, the connection is released and all resources are freed up. The connection management flow is illustrated in Fig. 4. A summary of the functions that are required for connection management and maintenance are in Table 3.

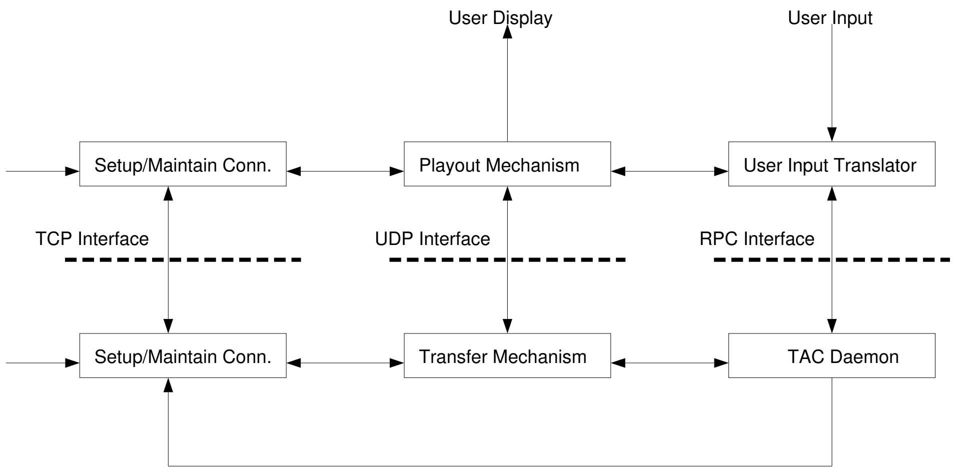

Figure 4: Connection Management for VOD

4 Application of the Services: The VVB

In this section, we describe a prototype multimedia application developed using the aforementioned framework. The Virtual Video Browser (VVB) is an interactive VOD prototype

[4]UDP for our implementation.
[5]TCP for our implementation.

designed to allow the browsing of a database of movies and the subsequent playout of individual movies [8]. It incorporates a simple query interface which lets users specify their preferences to the system to retrieve the appropriate video. The VVB is designed to work in a distributed environment where movies are stored in different databases interconnected via a network. It is therefore an ideal testbed for running distributed multimedia applications.

4.1 The VVB Interface Functionality

The basic VVB interface consists of four screens or menus: the *Category Screen*, the *Video Shelf Screen*, the *Query Shelf Screen* and the *Text-Output Screen*. In addition, the VVB allows a movie to be played-out in a video window (see attached color figure).

When a VVB session is started, a *Category Screen* appears, which allows the user to choose from a set of predefined categories (western, action, comedy, etc.). A user's selection results in a *Video Shelf Screen* to be displayed.

The *Video Shelf Screen* represents "virtual" shelves that one might find in a video rental store. These shelves display the movie titles resulting from a query to the movie database. The user can browse the shelf and playout a selected movie. The user has the additional option of querying the database for some specific attributes.

When the user decides to formulate a query, the *Query Shelf Screen* is invoked. The user can customize a query by specifying movie-specific attributes of the desired movie (producer, director, actor, scene, etc.).

After applying the query, all movies that conform to the requirements are displayed on a *Text-Output Screen* and an updated *Video Shelf Screen*. The *Text-Output Screen* provides an additional level of detail with respect to the query. This screen allows the identifying and browsing individual scenes of the movies.

4.2 Software Architecture

The VVB software architecture is designed in a layered fashion using object-oriented techniques with three application programming interfaces (APIs) as shown in Fig. 5. These APIs provide the functionalities required for database management, the video display, decompression, and user interface. The main program of the VVB is very simple as it forms the "glue" among the API functionalities. The advantage of this approach is the ability to rapidly create additional applications based on the same core software components. The VVB software modules consist of both commercial, off the shelf (COTS) software and software developed in the Multimedia Communications Lab.

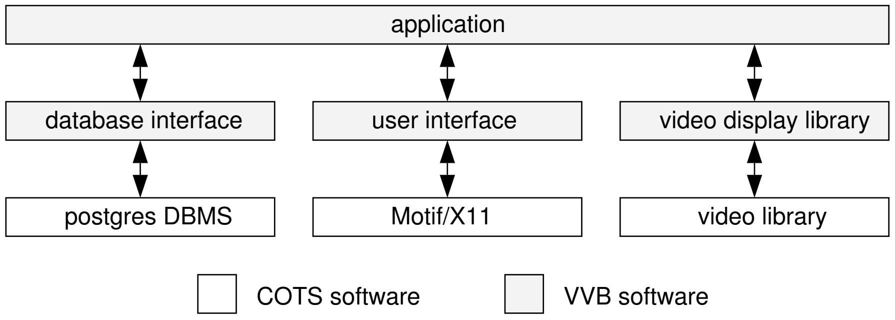

Figure 5: Software Architecture for the VVB

4.3 The VVB Mechanics

The system model for the VVB is consistent with the proposed distributed systems services of Section 3. It consists of many movie-terminals, or clients, and video-databases, or servers, interconnected via a computer network (Fig. 3). No particular assumptions are made about the underlying network in its design. A single central database (QDB) contains the information about the availability of movies and their locations within the VOD system. It acts as a *name-server* for the mapping of movies to the video databases. The VDBs contain the video data (movies) necessary for playout. The clients are provided with the necessary hardware/software to support the VVB user-interface and movie playout. For the VVB implementation it is assumed that while the QDB is located at a single site known to all network stations, the VDB is distributed across many sites in the network. The client queries the QDB to identify the VDB containing the required movie, and uses this information to set up a video communication channel between the client and the server.

The mechanics of the VVB operation can be described by three phases corresponding to a user query, connection establishment, and connection management (Fig. 6).

- **Query Phase:** The query phase involves the querying of the QDB to get all information that might be required by the user to obtain a selected movie or scene. The query mechanism is implemented using a remote procedure call (RPC) between the video-playout client and the QDB.[6] The user begins the query by selecting a category from the *Category Screen*. The client-system then queries the remote database (the QDB) for movies belonging to that particular category. The user can get more detailed information of the movies including a summary, poster, or textual description of a scene by further querying the QDB for movie-specific data.

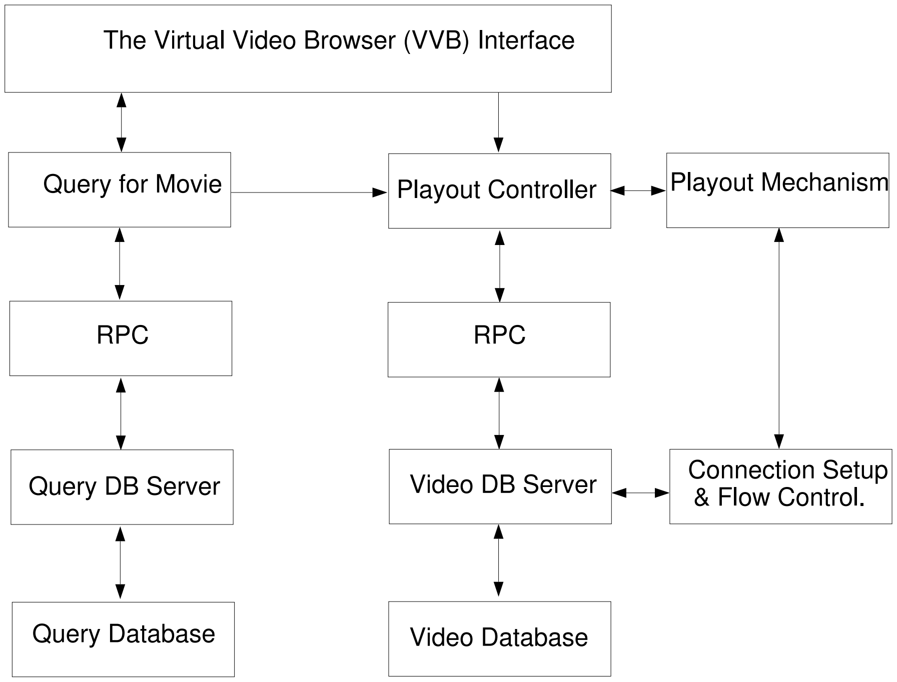

Figure 6: Mechanics of the VVB Operation

After the query phase, a movie can be selected for playout. The video playout phase is divided into two parts. The connection setup phase, and the connection management phase. During connection setup, the network station checks with the VDB to see if it can support a new connection. If a connection can be set up, it begins a negotiation phase, where QOS parameters are negotiated. If the negotiation proceeds to a successful conclusion, the connection setup and maintenance phases are initiated.

- **Connection Setup Phase:** The connection-setup phase establishes a dedicated client-server UDP connection which facilitates data transfer with a continuous media, synchronized flow control through the use of a separate dedicated TCP control channel.

[6]POSTGRES is the DBMS used in the VVB implementation of the QDB.

- **Connection Maintenance Phase:** The connection maintenance mechanism ensures that a satisfactory performance is delivered to the user. It ensures timely delivery of video frames to the client by monitoring a control signal returned via the TCP connection. The system thus adapts itself to load changes that can affect playout.

Video data are then delivered to the client via a dedicated UDP connection established when the session is invoked. This process requires the use of flow control mechanisms to ensure continuous playout at the client station. For this purpose we use UDP rather than TCP. The choice of UDP was based on the need to support data delivery in real-time. Even though TCP is a more reliable protocol, it uses retransmission of lost packets, which is usually not acceptable for real-time/video communication. In addition, video data are tolerant to losses, and we exploit this characteristic in our design. However, further analysis of flow control is beyond the scope of this paper.

5 Summary and Conclusion

In this paper, we have presented a distributed database organization for multimedia information systems based on the use of metadata and resource servers. We have illustrated a connection management scheme in context of a video-on-demand application. To this end, we identify the service primitives required for the management of continuous-media sessions in a distributed system. The concepts presented are illustrated by example of a prototype distributed application called the Virtual Video Browser (VVB). The VVB employs a client-server architecture to let its users browse a video database and to set up video sessions to remote video databases.

The VVB employs a two phase process to serve its users. These are (1) a query phase during which user queries are sent to the metadata server, and (2) a connection establishment and maintenance phase, during which connections are set up between the video server and the client machine. The VVB is currently in use on a testbed of Unix workstations interconnected via Ethernet. To be used as a viable alternative to broadcast TV, the VVB needs to function on a system supporting a large number of movies and users. We are currently considering the effort required to scale-up to such a scenario.

The general nature of the VVB interface and query mechanism provides us with a powerful tool to support other interactive multimedia applications. Additional projects are underway at the MCL. These projects include the deployment of a live, interactive distance learning environment that includes time-independent retrieval of instructional video.

Acknowledgment–The development of the VVB prototype, including data collection, was made possible by the significant efforts of the following individuals: G. Ahanger, R.J. Folz, J.F. Gibbon, P. Lumba, J.D. Marshall, A. Mol, G. Orlando, M. Ramanathan, F.W. Reeve, and D.H. Schelleng.

References

[1] Anderson, D.P., Homsy, G., "A Continuous Media I/O Server and its Synchronization Mechanism," *Computer*, Vol. 24, No. 10, October 1991, pp. 51-57.

[2] Berra, P.B, C.Y.R. Chen, A. Ghafoor, T.D.C. Little, "Issues in Networking and Data Management of Distributed Multimedia Systems," *Symposium on High-Performance Distributed Computing*, Syracuse NY, September 1992.

[3] Elmer-Dewitt, P. "First Nation in Cyberspace," *Time*, December 6, 1993, pp. 62-64.

[4] Gelman, A.D., H. Kobrinski, L.S., Smoot, S.B., Weinstein, "A Store-and-Forward Architecture for Video-On-Demand Service," *Proc. ICC*, 1991, pp. 27.3.1-27.3.5.

[5] Keller, R., and W. Effelsberg, "MCAM: An Application Layer Protocol for Movie Control, Access, and Management," *Proc. ACM Multimedia 93*, August 1993, pp. 21-29.

[6] Korth, H.F., and A. Silberschatz, "Database System Concepts," *McGraw Hill Ltd.*, 1986.

[7] Little, T.D.C., and D. Venkatesh, "Probabilistic Assignment of Movies to Storage Devices in a Video-On-Demand System," *Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video*, Lancaster, UK, November 1993.

[8] Little, T.D.C., Ahanger, R.J. Folz, J.F. Gibbon, F.W. Reeve, D.H. Schelleng, and D. Venkatesh, "A Digital Video-on-Demand Service Supporting Content-Based Queries," *Proceedings of the 1st ACM International Conference on Multimedia (ACM Multimedia'93)*, Anaheim CA, August 1993, pp. 427-436.

[9] Little, T.D.C., and A. Ghafoor,"Spatio-Temporal Composition of Distributed Multimedia Objects for Value-Added Networks," *Computer*, Vol. 24, No. 10, October 1991, pp. 42-50.

[10] Loeb, S., Hill., R., Brinck, T., "Lessons from LyricTime: A Prototype Multimedia System," *Proc. 4th IEEE ComSoc Intl. Workshop on Multimedia Communications (Multimedia '92)*, Monterey, CA, April 1992, pp. 106-113.

[11] Press, L. "The Internet and Interactive Television," *Communications of the ACM*, Vol. 36, No. 12, pp. 19-23.

[12] Ramarao, R., and V. Ramamoorthy, "Architectural Design of On-Demand Video Delivery Systems: The Spatio-Temporal Storage Allocation Problem," *Proc. ICC'91*, 1991, pp. 17.6.1-17.6.5.

[13] Rangan, P.V., H.M. Vin, and S. Ramanathan, "Designing an On-Demand Multimedia Service," *IEEE Communications Magazine*, Vol. 30, No. 7, July 1992, pp. 56-64.

[14] Rowe, L.A., B.C. Smith, "A Continuous Media Player," *Proc. 3nd International Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, November 1992.

[15] Silvers, Singh, E., "Multimedia Communications on the NYNEX Shuttle," *Proc. COMPCON Spring 1992*, San Francisco, CA, February 1992, pp. 84-87.

[16] Sincoskie, W.D., "System Architecture for a Large Scale Video On Demand Service," *Computer Networks and ISDN Systems* Vol. 22, 1991, pp. 155-162.