Table 1: Connection Setup Primitives for the VOD Server

check_no_connections()	Check to see if a new connection can be supported.
indicate_connection_setup_failure()	Indicate status of connection setup.
start_QOS_negotiation(video_name, client_name)	Begin QOS negotiation.
retrieve_video_chars(video_name)	Retrieve video characteristics for QOS negotiation.
indicate_QOS_needs(max_rate, min_rate,...)	Indicate QOS Requirements.
increment_no_users()	Reserve resources.
setup_connection(video_name, client_name)	Spawn a child to setup dedicated UDP connections.

Table 2: Connection Setup Primitives for the VOD Client

query_video_database(database_name, video_name)	Query the VDB for the specified video.
examine_video_statistics(video_params)	Check to see if the connection can be sustained.
acknowledge_conn_acceptance()	Start the connection.
close_connection()	Close the connection.
setup_playout_and_wait_for_connection()	Initialize the playout hardware and wait for the connection call from the remote server.

adjustment of volume are dealt with locally via the playout mechanism at the client. The connection management flow is illustrated in Fig. 4. Some of the functions that are required for connection management and maintenance are summarized in Table 3.

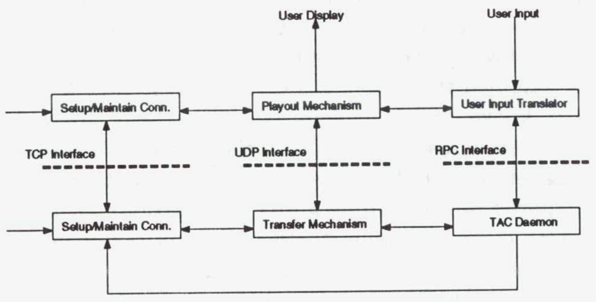

Figure 4: Connection Management for VOD

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 designed to allow the browsing of a database of movies and the subsequent playout of individual movies [6]. 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 in which 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.

When a VVB session is started, a *Category Screen*

Table 3: Connection Management Primitives for the VVB

playout_video()	Indicate to both the client and the server to ensure that frames are played out at the proper rate.
forward_video()	Indication to the server to playout every nth frame in lieu of consecutive frames.
reverse_video()	Analogous to forward_video().
pause_video()	Pause video stream. Directed toward the playout mechanism and the server.
change_volume()	Change volume at the client.

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.

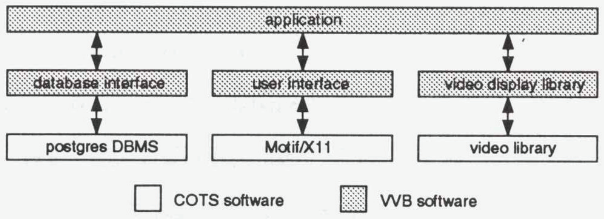

Figure 5: Software Architecture for the VVB

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 integrates 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.

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. In the current VVB implementation, we do not distinguish between the resource and metadata servers. 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 (details of the metadata models used for the VVB database are described extensively elsewhere [6]).

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