The Franco-Polish School of New Information and
Communication Technologies in Poznań, Poland

Northeast Parallel Architectures Center (NPAC)
Syracuse University, USA

Video-on-Demand System: Compression and Codec Independent Video Client in ActiveMovie Architecture

Bartłomiej Winnowicz
Master Thesis
Thesis Supervisor:
prof. Czesław Jędrzejek

Consultant:
dr Marek Podgórny

Syracuse, September 1996

Abstract

This document presents Video-on-Demand (VOD) system being developed in Northeast Parallel Architectures Center (NPAC) in Syracuse, USA. The VOD system allows to search for and stream movies over the computer network, with immediate displaying of the movies on the Client side.

A special emphasis is given to Video Client component of the VOD system. The author of the paper was responsible for design and implementation of the Video Client component working with PC stations.

ActiveMovie, a brand new architecture and technology for building multimedia applications in Windows 32 bit environment, is also described. The implementation of Video Client for

Windows 95/NT as a network ActiveMovie source filter is presented.
The new solution is compared with Video Client for Windows 3.1, being developed using Video for Windows architecture.

Streszczenie

Poniższy dokument zawiera opis systemu *Wideo na Żądanie*, skonstruowanego w Northeast Parallel Architectures Center (NPAC) w Syracuse, w USA. System ten pozwala na przesyłanie sekwencji wideo poprzez sieć komputerową, z natychmiastowym wyświetlaniem tych sekwencji poprzez użytkownika.
Najdokładniej przedstawiony został element systemu znajdujący się na komputerze użytkownika - *Wideo Klient*. Autor dokumentu był odpowiedzialny za zaprojektowanie i zaimplementowanie *Wideo Klienta* na komputery klasy PC.
Dokument opisuje także ActiveMovie - nową architekturę i technologię do tworzenia aplikacji multimedialnych w 32-bitowym środowisku Windows. *Wideo Klient* dla systemów Windows 95 i Windows NT został zaimplementowany jako filtr sieciowy działający w schemacie ActiveMovie.
Nowe rozwiązanie zostało porównane z *Wideo Klientem* dla systemu Windows 3.1, który bazuje na wcześniejszej technologii multimedialnej - Video for Windows.

Résumé

Ce dossier présente un système de Vidéo-sur-Demande (VSD) qui est en développement à NPAC à Syracuse, USA. Le système VSD permet de rechercher et de transmettre des films sur le réseau de l'ordinateur, avec immédiate retransmission des films du côté du client qui veut visionner.
Une importance toute particuliculière est donnée au composant *Vidéo Client* du système VSD. L'auteur du dossier a été responsable pour la création et l'instauration du composant *Vidéo Client* travaillant avec des stations PC.
FilmActif (ActiveMovie), une nouvelle architecture et technologie pour créer des applications multimédiatiques dans le milieu de Windows 32 bit est aussi décrite. L'instauration de composant *Vidéo Client* pour Windows 95/NT entant que filtre source dans l'architecture du FilmActif est présentée.
La nouvelle solution est comparée avec *Vidéo Client* pour Windows 3.1, utilisant l'architecture précédente, Vidéo pour Windows.

I.Introduction

Multimedia - the combination of any two or more different media types (text, graphics, images, audio, video), has shown great potential for enriched communication, bringing images and spoken words directly to our eyes and ears. A decade ago, personal computers (PCs) were able to present data to an user only in a textual form. But the significant progress in technology allowed to add graphics and sound cards to average PC station , fully supporting presenting of audiovisual information stored on a local hard disk. New applications, with incorporated video and audio streams, appeared on the market. They included computer-based training and computer-aided design applications, games. Authoring tools were developed as means to build multimedia applications. A very large size of files containing video and audio data made a diskette or even a hard disk unsuitable for new multimedia applications. Therefore, a CD-ROM drive was added as a standard equipment to the multimedia PC station, and a CD-ROM disk became an easy way of providing the new multimedia applications to an end user. The idea of using media-specific compression techniques in order to minimize the amount of requested storage space led developers to incorporate video and audio compressors and decompressors into multimedia PC stations. Simultaneously, a theoretical research of best compression algorithms became a basis for the codec implementations.
Along with the emergence of a new generation of multimedia applications has come a need to take maximum advantage of those applications by running them over computer networks. The phenomenon of very fast growing World Wide Web (WWW) proved how important and how useful the global networked information service can be. At the beginning of WWW, it was

designed to handle hyperlinked text, and in this way to allow to browse through textual information made available on the global network. The WWW became a means to search for and receive information that consists of various types of media. New functionality was added to WWW servers and browsers, enabling interchange of multimedia data. Amount of information and data in digital format, freely accessible over the global network, increased dramatically. Whole branch of new services came under discussion: distance learning, interactive TV, information kiosks, audio/video conferencing, digital libraries, multimedia news services, home shopping. They have to use the power of new multimedia-oriented desktops and new network solutions, because the traditional computer systems were not designed to transport and transform real-time constrained video and audio streams. For these services to succeed, computers have to be able to handle and transmit larger amounts of data at faster rates for lower costs.

This is where Video-on-Demand (VOD) technology comes in, as a basis for the new multimedia services. VOD technology relates the important problems lying under development of the new services: video data search and access, video streaming over the network, video processing and displaying, user interaction. Interactive, stream-oriented network solutions are developed as custom protocols between two major VOD components: VOD server and VOD client. The accessibility of multimedia information is provided by VOD servers, having large storage capacities, very fast CPUs and very broad network bandwidth at their output. VOD clients are responsible for receiving video from the network, video decompression and video displaying.

Video-On-Demand (VOD) could be seen itself as an innovative TV service that gives users the power to interact through their video-enabled desktops. At the touch of a button and in the comfort of their home, they can enjoy movies, keep up with the latest news, learn from educational programs and documentaries, shop, book concert tickets and do much more. The unique feature of VOD is that programs and services can be available day and night. Viewers do not have to wait for scheduled screening times or watch whatever program is being shown. Program selection is user-friendly and easy as there is a screen menu where users indicate their choice. An additional protection feature could prevent unauthorized access.

NPAC VOD system was designed to be a base service in the scope of the interactive multimedia services. The functionality it provides include:

- a searching capability for particular video material; the search can be category- or content-based,
- end user interaction with the VOD system, allowing him/her to manage a flow of video data,
- preprocessing, network streaming and post-processing of multimedia data (video, audio, text),
- video content production,
- video storage and access management.

NPAC VOD system architecture differentiate three major components: Video Server, Database Server and Video Client. This model can be found in other VOD system designs [], and is well suited to perform and divide the tasks of VOD system.

The Video Client is very important component of NPAC VOD system. A functionality of the Video Client includes:

- real-time networking, decompressing and displaying of video data
- enabling user interaction
- user interface for searching and displaying of video material.

There is one very important reason to build the Video Client for PC stations - namely, the significant number of potential users (either Internet or intranets) of VOD systems owns or has access to the PC stations.

A goal of the author of this thesis was to build the Video Client working on PC platforms. Major challenges in building of the Video Client for PC was:

- achieve compression and codec hardware and software independence,

- provide a design independence of networking and decompression functionalities,
- merge the Video Client with existing NPAC Video-on-Demand architecture, as well as extend functionality of the architecture,
- finally, build the Video Client in the most standardized way, using existing multimedia development architectures.

The need for the compression and codec independence can be illustrated by an example of former NPAC Video Client. The Video Client is working in Windows 3.1 environment and is limited to PC stations with specific hardware MPEG-1 decoder. When it was built, the requirement was that the Video Client had to handle MPEG-1 streams. At the time, there was no software MPEG-1 decompressor with sufficient quality available. Therefore, there were few hardware decoders taken into consideration to provide video decompression functionality to the Video Client. Unfortunately, the hardware decompressors provided incompatible Application Programming Interfaces (APIs), resulting in a necessity of rewriting the Video Client application code for each such a codec. To provide the VOD service to possibly the largest number of end users there was a need for more generic solution. There was an architecture for developing multimedia applications in Windows 3.1 environment, Video for Windows (VfW). This architecture provides some means to build applications using various video and audio decompressors installed in the Windows operating system, but it has following disadvantages. Firstly, it was designed to build local, file-oriented applications. The VfW API provides functions that work only with local files. Secondly, it wraps up video and audio streams into Microsoft's proprietary file format, Audio Video Interleaved (AVI), which is totally incompatible with MPEG-1 file format. Next, the VfW architecture provides a limited access to installed codecs, making it impossible to access their buffers directly. Finally, the overall VfW architecture does not take network real-time constraints into account. Those all disadvantages made building of a generic Video Client using Video for Windows architecture impossible, and the final version of the Video Client was limited to PC stations having PCMotion hardware MPEG decoder card installed. The relying on one proprietary decoder is a feature common for all multimedia applications using Video for Windows architecture, what limits severely number of PCs able to host these applications.

The possibility to overcome all difficulties with building Video Client, related to Video for Windows architecture, came with a new development environment from Microsoft, ActiveMovie. Microsoft changed its orientation from providing only local solutions to ship whole game of solutions for Internet and intranet networks. These solutions are called ActiveX technology. The ActiveMovie gives a technology to stream multimedia data, as well as provides an architecture to build flexible and easy-to-configure multimedia applications. The architecture is never again oriented on local applications. Multimedia data can be streamed from HTTP server, and this functionality is being incorporated into ActiveMovie run-time.

From the point of view of the Video Client developer, ActiveMovie allows to extend its functionality by custom network solutions. The architecture is well designed object oriented technology, basing on Component Object Model (COM). Additionally, it provides fast software MPEG, AVI and QuickTime decompressors, as well as means to replace them with custom decompressors in overall streaming scheme. The modular architecture of ActiveMovie allows direct access to video and audio codecs. The ActiveMovie API exists as a set of COM interfaces, and these interfaces represent the standard way of incorporating new codecs (hardware and/or software) to the system. ActiveMovie uses power of Windows 32bit multithreading and preemptive multitasking, which makes it possible to run only on Windows 95/NT operating systems, with no chance of doing that on Windows 3.1 system.

The Video Client for Windows 95/NT was designed to use maximum capabilities of ActiveMovie, including Microsoft's software MPEG-1, AVI and QuickTime decoders. It also took advantage of DirectX support for video displaying that ActiveMovie uses. The Microsoft decoders for these three compression formats have slightly different APIs and all the cases has been implemented. The design of the Video Client focused on finding a point of network connection to the ActiveMovie components responsible for various decompression formats that will make required changes influencing the least a behavior of the architecture. There was a possibility to replace more of ActiveMovie functionality by custom solutions but the probability of incompatibility of the Video Client with next ActiveMovie releases was very high, as ActiveMovie development kit is still a beta software. The Video Client replaced only one major component of ActiveMovie streaming architecture: file source filter, with new, network source filter.

Thanks to ActiveMovie and Video Client design, new Video Client for Windows 95/NT is independent from hardware or software MPEG-1 decoders. Moreover it works with various compression formats: AVI, QuickTime and H263. Thus there is no need for additional hardware supporting decompression. The high-end solutions could still make use of custom hardware compressors, what requires writing of an ActiveMovie transform filter, but incorparating of the hardware decompressors is totally independent from the Video Client application itself. The Video Client will utilize custom decompressors, such as hardware MPEG decoders, if only they are installed in ActiveMovie architecture. An example of using a custom decompressor is H263 decoder implemented and registered as a transform filter independently from the Video Client. Regarding low bitrate requirements of H263 it makes sense now to use VOD system on low bandwidth networks.

Therefore the new Video Client for Windows 95/NT is much more flexible, extensible, and provides more functionality to NPAC VOD system than its version for Windows 3.1.

The outline of this thesis is as follows: **...............**

I.Video-On-Demand Systems

A.Functionality

The general goal of Video-on-Demand system is to allow end users to request and receive various kinds of video material from remote repositories available in the system to the users home or workplace, through network connection. The video material, for example movie pictures or news, are being displayed at user desktops, as if they were residing in users local file system, while in reality there are streamed from the remote source.

There has to be several conditions satisfied to call a system a Video-on-Demand one.

The most important is that displaying of video material has to start a non-significant portion of time after an user has requested it. The delay may vary (from 1 to 30 seconds) but it should be independent from the file size of the movie being displayed. And then it should play continuously as long as the user requests it or until the movie ends. It means that Video-on-Demand system has to stream large files over the network and process them incrementally at the user desktop as soon as possible. This is the contrary to downloading of video material and apart from small delays it has another advantage over downloading - it does not use large amounts of local storage on the users computer system.

Next, the Video-on-Demand system has to provide an end user with capabilities to search for the video material they are interested in. They should be able to browse through the system in order to get information about the movies actually available, based on a category, producer, content, etc.

The result of such a search should lead users to the interfaces allowing them finally to display the video material they have found and chosen. The Video-on-Demand system should guide the user through the searching and displaying process by a commonly-accepted and easy-to-understand User Interfaces.

The other functionality that a Video-on-Demand system should provide to an end user is its interactivity with the user while displaying video material. It means that there should be the ways in which the end user can dynamically change parameters of the movie being displayed. The level of that interactivity may vary depending on the specification of the system from simple Play and Stop capability to full VCR-like functionality extended by additional features like changing

displaying window size, adding video marks or introducing special effects.
The presented functionality relates to the end-user point of view on the Video-on-Demand system. Additionally, from the point of view of the administrator of the system it should provide tools for automated contents production, content editing, content management (i.e. arrangement and rearrangement of the digitized video material in and between repositories of the system), statistics generating and dynamic tuning of the system based on those statistics.

A.Core Technologies

1.Internet and World Wide Web

Internet as it is known today is really a number of records which map out how the communication between computers attached to the net proceeds. Internet (i.e. the laying down of the records) initially started in the USA as a military developing project. One of the objectives was to develop and construct a network so reliable that it is functional also in war situations. Today Internet is controlled and developed by civilian organizations and the military no longer plays a determinative role as regards Internet.
At first primarily American universities were attached to Internet and until recently very few private users were attached to Internet. Currently Internet is clearly the "largest" in the USA. Estimates indicate that 50% of all resources attached to Internet are situated on the American continent. The rest is mainly in Europe and Japan though Internet exists on all continents. Nobody owns Internet. Internet was traditionally financed by universities and research institutes and only recently private companies are starting to attach and thus finance Internet on a larger scale.
Internet is unparalleled the world's largest and most frequently used network. Approximately 30 million is an estimate of users attached to Internet. Should the present growth rate continue (which of course will not happen) the population of the entire world will be attached to Internet by 2001.
Internet has developed from offering remote logins (Telnet) only to now comprising e.g. file transfers (FTP), electronic mail, discussion groups (Usenet), information retrieval and much more. The most recent facility is the World Wide Web (WWW), connecting all the above facilities with even more on a user friendly graphic platform. In order to obtain access to WWW a Web-browser (NetScape, Internet Explorer, Mosaic a.o.) is applied. With these programs it is easy to surf the Internet in an effective and entertaining fashion. The required service or information is obtained by clicking. World Wide Web is prophesied to become the absolute most popular and most used service on Internet. By now all larger companies in the computer line and many others have their own Web-server to which information, various services, and support on the companies' products is offered.
WWW is actually a number of documents placed on different servers around the world. The documents comprise both text and graphics. Often they comprise hyperlinks which means references to the Internet resources. Hyperlinks are often indicated by a different colour than the rest of the text in the document. Using the mouse to click a hyperlink the document in question will be shown or moved to the particular resource - this may be Telnet, Usenet News, FTP, sound files among many other facilities. The resources in WWW are indicated unambiguously by means of a URL - Uniform Resource Locator.
WWW clients do not directly support synchronized playback of continuos media. Audio and video objects are treated as files, and as such they must be transferred completely before they can be played, and they cannot be played in parallel. The development of new WWW servers and browsers takes multimedia streaming aspects in consideration.

1.High bit-rate compression (MPEG)

MPEG is an acronym for the Motion Picture Expert Group (MPEG) of the International Standards Organization. The group generates standards for digital video (sequences of images in time) and audio compression. In particular, they define a compressed bit stream, which implicitly defines a decompressor. However, the compression algorithms are up to the individual manufacturers, and that is where proprietary advantage is obtained within the scope of a publicly available international standard.

There are actually two MPEG standards, MPEG-I and MPEG-2. A third, MPEG-4, is currently under development.

MPEG-1 is a small picture format mode geared to a resolution of 352 by 240 pixels at 30 frames per second, with full CD quality audio. MPEG-I was originally optimized at single speed CD-ROM transfer rates or 1.416 Mb per second. MPEG-2 offers a resolution of 720 by 480 pizels at 30 frames par second with full CD quality audio. MPEG-2 can incorporate a range of compression ratios which trade off economies of storage and transmission bandwidth against picture quality. At compression ratios of 30:1 and smaller. MPEG-2 offers the perception of broadcast quality TV. For greater economy, MPEG-2 supports up to 200:1 compression ratios. At 25 or 30 frames per second, an MPEG-1 encoded movie provides a quality similar to what might be obtained when watching a VHS movie recording. The corresponding transmission rate is 1.5 Mbit/s. This is not a very high speed for most local area networks. A 10 Mbit/s Ethernet should be able to support 4-5 concurrent transmissions. An FDDI ring or an 155 Mbit/s ATM network should be able to support anywhere between 20 and 100 concurrent transmissions. On a wide area network, it might however not be possible to support one single transmission. Many of the links in the current Internet have only a bandwidth of 1.5Mbit/s or less. Another obstacle is the storage size of movies. At 1.5 Mbit/s, one hour will require 675 Mbyte, which equals 1.35 Gbyte per feature length film. Storing a reasonable collection of films will require either large arrays of expensive disks, or some, possibly complicated, system based on tertiary storage such as tape stations.

Yet another problem is the processor speed. 1.5 Mbit/s corresponds to approximately 188 kByte/s. If data is sent as 1500 byte packets, which is the maximum permitted content of Ethernet packets, this yields a packet intensity of 125 packets per second. That is to say, one packet will be sent every 8 ms. Depending on how long it takes the server to generate one packet, this may severely limit the number of concurrent transmissions that can be sustained.

The presented limitations made a use of MPEG compression in multimedia applications (local and network) a high-end solution. MPEG streams cannot be played over Internet or ISDN networks, when they can be played over LANs or ATM networks. For the last few years the real-time decompression of MPEG streams was impossible on PC stations without installing hardware MPEG decoders. However, the CPU speed increase and the optimization of MPEG decompression software implementations made the installation of MPEG decompressing hardware unnecessary. The major problem remains the network bottleneck, and unsuitable existing routers and protocols to transport high-bitrate multimedia data. Therefore the MPEG streaming over the network is a solution for intranets, not for the present Internet.

1.Low bit-rate compression (H263)

H.263 is a ITU-T video compression standard. It was designed for low bitrate communication, early drafts specified datarates less than 64 Kbits/s, however this limitation has now been removed. It is expected that the standard will be used for a wide range of bitrates, not just low bitrate applications. It is expected that H.263 will replace former low bit-rate video compression standard, H.261, in many applications.

The coding algorithm of H.263 is similar to that used by H.261, however with some improvements and changes to improve performance and error recovery. The coding algorithm is a hybrid of inter-picture prediction, transform coding, and motion compensation. The differences between the H.261 and H.263 coding algorithms are listed below. Half pixel precision is used for motion compensation whereas H.261 used full pixel precision and a loop filter. Some parts of the hierarchical structure of the datastream are optional in H263, so the codec can be configured for a lower datarate or better error recovery. There are four optional negotiable options included to improve performance: Unrestricted Motion Vectors, Syntax-based arithmetic coding, Advance prediction, and forward and backward frame prediction similar to MPEG called P-B frames.

H.263 supports five resolutions. In addition to QCIF and CIF that were supported by H.261 there is SQCIF, 4CIF, and 16CIF. SQCIF is approximately half the resolution of QCIF. 4CIF and 16CIF are 4 and 16 times the resolution of CIF respectively. The support of 4CIF and 16CIF means the codec could then compete with other higher bitrate video coding standards such as the MPEG standards.

H.263 is better than MPEG-1/MPEG-2 for low resolutions and low bitrates. H.263 is less flexible than MPEG, but therefore requires much less overhead. Another difference is the negotiable options in H.263. MPEG has B-frames, but H.263 has PB-frames which are almost as good for moderate amount of movement, but require much less overhead. H.263 has overlapped block motion compensation, motion vectors outside the picture and syntax-based arithmetic coding. These options are not in MPEG at all. Unlike in MPEG, H.263 does not take care of compressing audio streams and audio/video synchronization.

H263 is still preserving quite good quality while requiring low bitrates. Thus H263 video compression can be adapted as a well-suited video streaming solution for the present Internet

1.Real-Time Networking

When it comes to the delivery of bursty data, TCP is the protocol that guarantees it. However, TCP is not well suited to handle multicast and real-time data. That's where RTP (real-time transport protocol) comes in.

RTP works alongside TCP, providing end-to-end delivery of such data as video broadcasting and multiparticipant interactive audio and video. Feedback on reception quality and optional identification of the receivers of the multicast stream are provided by the real-time transport control protocol (RTCP), which is an integral part of RTP.

Although most implementations of RTP run in IP environments on top of the user datagram protocol (UDP), the protocol actually was developed to function independently of the underlying transport and network layers. This means that RTP can be used with other protocols, such as ATM or IPX. Because multimedia applications vary in their requirements, RTP is by design a protocol framework rather than a simple protocol. Unlike TCP, which is implemented as a distinct layer, RTP was designed to be tightly coupled to the application.

A key factor in achieving real-time quality of service is a reservation setup protocol, a mechanism for creating and maintaining flow-specific state information in the end-point hosts and in routers along the data flow path. The resource reservation protocol (RSVP) was developed specifically for the packet-switched multicast environment. It was designed to meet following requirements: support for heterogeneous service needs, scalability to large multicast groups, ability to preempt resources to accommodate advance reservations.

Summarizing, in a multimedia session, each medium is carried in a separate RTP session, with its own RTCP packets controlling the quality of the session. Routers communicate via RSVP to set up and manage reserved-bandwidth sessions.

The current problem is that Internet routers generally do not support RSVP protocol. The process of replacing old routers with new, RSVP-enabled ones, will take few years.

1.IP Multicasting and MBone

Compressing data is one way to make better use of bandwidth, but multicasting also is important in allowing a stream to be sent efficiently to many receivers. The concept of multicasting is simple: Rather than sending a separate stream of packets going to each intended user (unicasting) or transmitting all packets to everyone (broadcasting), it involves the simultaneous transmission of traffic to a designated subset of network users.

A key to multicasting is the Internet Group Management Protocol (IGMP). IGMP enables users to sign up for multicast sessions and allows these multicast groups to be managed dynamically, in a distributed fashion.

The MBone is a logical network that is built on top of the Internet. It provides multicast IP connectivity between several hundred LANs on the Internet. Multicast IP is an extension to the IP protocol that makes it possible to denote a group of recipients by one single IP address.

Essentially, when an IP datagram is sent to such a multicast group, it will be delivered to all hosts that are members of that group. The routing of multicast packets on the MBone is implemented so that multicast packets are only duplicated when the path to group members branch at a router. When the transmission media of a link supports multicast, such as Ethernet, the IP multicast datagrams are transmitted using link level multicast. The efficient routing of multicast packets on the MBone has made it tractable to use it for experiments in multimedia conferencing. Such experiments have been conducted for about three years using various tools for transmitting audio, video and whiteboard data.
IP Multicasting can be used in Nearly Video-on-Demand (NVOD) systems that provide TV program services over the network. The example of utilization of IP Multicating over MBone network is IP/TV video-multicasting application under development by Precept Software, Inc.

1.Asynchronous Transfer Mode

Asynchronous Transfer Mode (ATM) is a set of standards and specifications for a networking technology. The key features that makes ATM well-suited in Video-on-Demand systems include:

- ATM is connection-oriented such as the telephone system. Data transfer takes place after an end-to-end virtual circuit has been set up across a network. Since the circuit is virtual, connectionless applications may use ATM as well. The ATM switch provides dedicated bandwidth; that is, each connection across a switch can provide throughput of, say 100 megabits per second, rather than the whole network providing 100 megabits per second throughput that must be shared by all users. Unlike leased lines as those provided by the phone companies, the dedicated bandwidth is only available for the duration of the connection.
- ATM technology uses fixed sized cells which makes it possible to guarantee service to time-dependent data such as audio and video data. The fixed sized cells also allow for fast switching hardware.
- ATM allows multiple parallel (logical) connections to be multiplexed over a single line.
- Error detection and correction is not performed at each node in the network but rather at the end nodes only. This takes advantage of the reliability of todays network links.
- ATM can use networks that are currently installed; therefore, costly rewiring is not necessary for users to use ATM.
- ATM is scalable. It provides the same lower protocol layers for local, metropolitan, and wide area networks.

ATM uses small fixed length packets (5 bytes of header plus 48 bytes of payload). ATM transmits data at the speed of 150 Mbps to Gigabits. ATM provides such services as: bandwidth on demand , constant bit rate service, Quality of Service guarantees. This is why ATM is suitable for video and audio.
Admission control in ATM is responsible for accepting or denying a connection bandwidth request. It is based on description of traffic flow and the current amount of resources (bandwidth, buffers, and processing). The resource allocation in ATM is a tradeoff between network utilization and user satisfaction.
ATM could be a great networking technology for Video-on-Demand and other multimedia services. Unfortunately, the use of ATM in global networks is not technically possible during next few years. However, the technology is already coming to intranets.

1.Other emerging technologies

A strong need for multimedia application caused many developers to create much more new technologies, better suited to the new requirements that the traditional technologies were. They include:

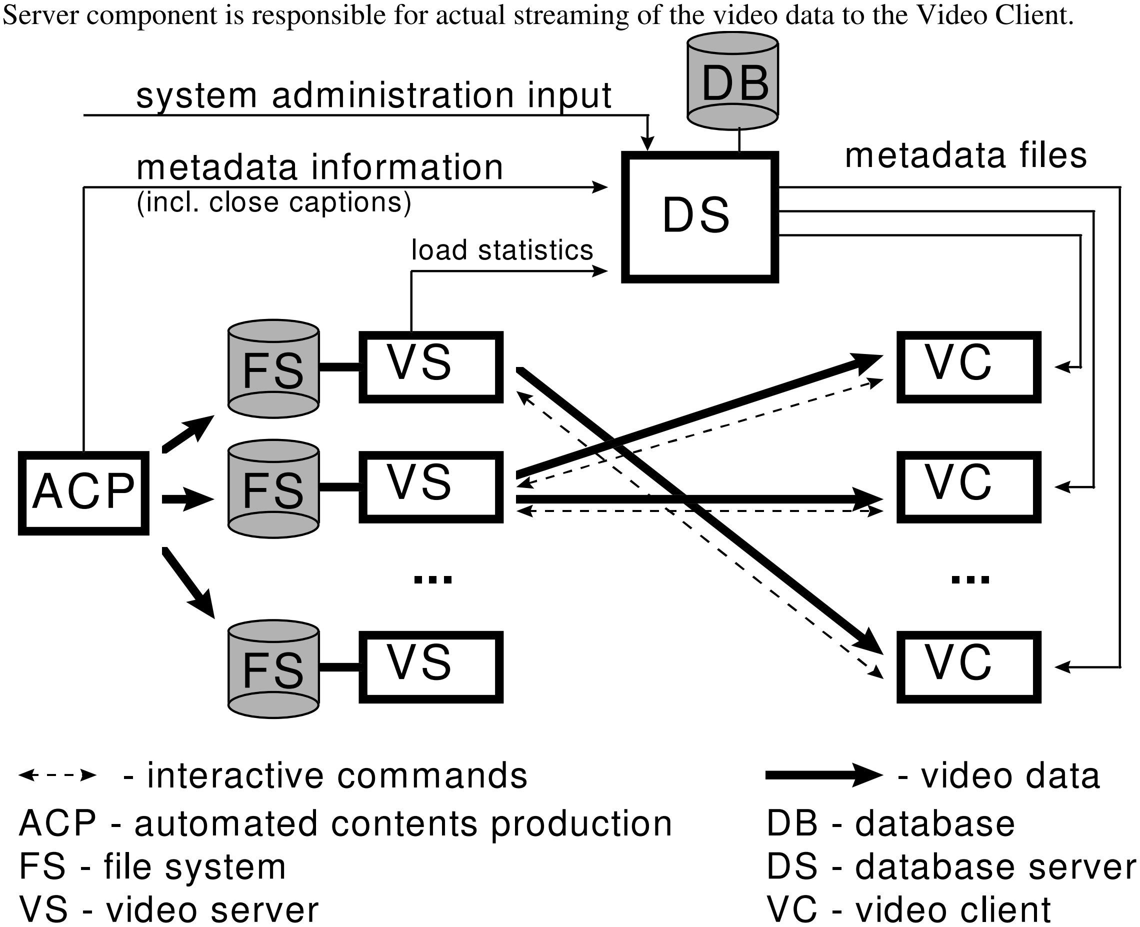
- continuous media and real-time file systems, with guaranteed rate I/O support (an example of real-time file system is Silicon Graphics XFS file system)
- mutimedia databases, merging capabilities of relational and object oriented databases

- set-top-boxes which represent specialized set of hardware and software on an end user desktop, serving as an interface between the user and the multimedia system.

A.NPAC Video-on-Demand System

1.Component view

The following model was designed to address all requirements for Video-on-Demand system presented in previous paragraph. Each of the components address some of the Video-on-Demand functionality. The functionality of the system residing on an end-user desktop is realized by Video Client component. The Video Client component communicates with two server components: Database Server and Video Server. The Database Server is responsible for providing to the Video Client the capability of searching for video material. The searching will not involve receiving the video data itself, instead it will provide the Video Client with the information where the video files are stored, that the end user is interested in. Based on that information, the Video Client may now communicate with the Video Server, which resides on the computer storing the requested video material locally in its filesystem. Therefore the Video Server component is responsible for actual streaming of the video data to the Video Client.



The designed model allows two consecutive phases of user interaction with the Video-on-Demand system: searching for video material and actual displaying of that video material. First phase is realized in communication between Video Client and Database Server, second - in communication between Video Client and Video Server. There is not much dependence of either phase to the other, the major one is that the result of the first phase invokes the second phase. By selecting from the search results, the user decides which video to watch, i.e. which file on which Video Server has to be accessed by the Video Client. The two user interaction phases have to be preceded by contents production tasks. As a result of them the video data will be loaded on Video Servers filesystems, and the information about the video data will be stored in the

Database Server.

1.Metadata concept

The independence of the searching and displaying phases is possible due to the metadata concept. The searching for video material is based on the metadata about the Video Servers existing in the system, and the metadata about particular video clips that are being stored on these servers. The metadata about the Video Server may include name, IP address, type of operating system, performance, and others. The metadata about video clips includes the name of the clip, Video Servers on which one can find the clip, the clip filename on the Video Server, generally speaking the name and remote locations of the video clip. This information passed to the Video Client from Database Server will enable establishment of the connection between the Video Client and the appropriate Video Server, as well as the streaming of required video file to the Video Client. There is another part of metadata about movie clip that is of utmost importance in process of video material search. This is the information on which an end user performs the search operations, and that kind of metadata about movie clip is strictly depending on the kind of search capabilities that the Video-on-Demand system specifies. For example, if an end users is going to be given an option of selecting video clips based on their producer, there must be a producer name associated with all of these video clips. It means that the information about producer will be included into metadata about these video clips. Similarly, if there are some video clips that their producer is unknown, the metadata information related to those clips will not contain the producer name, and in effect these clips will be excluded from the basis for the example kind of search. Making searching capability useful requires creating of metadata information for every piece of video material that is added to the system, as well as for every new Video Server added. The metadata information stored in database should be consistent with the actual overall state of the system, and dynamically updated while the Video Servers or video clips are added or removed, as well as metadata information about video clips should change in the database respectively to changes of video clip properties.

1.Video material search

There are two types of video material search that an end user could request to: category-based and content-based. The example of the search based on the video clip producer belongs two the first type. This is the search for a property associated with the entire video clip. Database Server does not need to perform an extensive database search in this case, as this is simply checking of the value of the single property of metadata information related to video clips. The result of such a search is a list of metadata files about the clips that have that property satisfied. The content-based type of search represents the need not only for the particular video clip, but also for the particular position in that clip that satisfies user query. For example, the end user looks for the video clips where the specific person was shown. In case of lengthy news program or a 2-hour-long movie picture the user could be very interested in displaying of the video material from the moment the person is being shown or starts talking. To allow the end user to have such a capability the searching part of Video-on-Demand system has to provide as a result both video clips and video clip positions. Therefore the Database Server must produce search results that will consist of the list of metadata files for video clips, and additionally for each clip there must be a list of positions satisfying users query. The Video Client must be able to seamlessly start playing from the position the user will choose. It requires constructing of Video Client with having that in mind. As to the Video Server, it must be able to stream the video data from any random position. The Video Client must inform the Video Server from what position it should start streaming and it means that random access must be included into the set of interactive commands being sent between Video Client and Video Server.

The reason why content-related metadata is so important in searchable Video-on-Demand system is that any content-based search made on raw multimedia data, for example speech recognition or image recognition, has very high computational cost. Moreover, the large number of video material that has to be searched (on several Video Servers) makes it impossible to perform search through raw video data on-the-fly, and this is out of the question. Thus, to perform any content-capable search on the Video-on-Demand system, there must be metadata produced describing a video content, in such a way it can be stored efficiently in database, and the Database Server will be able to perform quick search operations on this type of metadata. The question remains what is the best way of creating metadata for content search, as well as how this metadata should look like. The NPAC solution will be presented later in the next paragraph.

The process of creating metadata, storing it into database, and final search through the metadata database by sending queries, with retrieving and displaying it on the Client side could be realized by various search systems capable of searching through multimedia data, in existence of unchanged Video Client and Video Server components. The only condition would be a standardized form of metadata files invoking the Video Client.

1. Video Server and Client communication

While the content production does not have real-time constraints, and searching phase introduce real-time constraints but they are without the influence on the robustness of the Video-on-Demand system, the Video Server and Video Client communication has very high real-time requirements. In Video-on-Demand system the large amount of data for video stream has to be sent over the network in continuous fashion without delays. It creates network bandwidth requirements and problems to solve like what is the best way to deal with burstiness of network traffic, as well as how to diminish amount of data sent over the network, without loss of the video quality. There are solutions aiming in multimedia oriented network protocols, there are other solutions aiming in very efficient compression scheme for video data. The combining of both methods can give tremendous improvement in the quality of the video being sent over the network with the network characteristics unchanged.

The communication between Video Server and Video Client may be divided into two categories: video data stream and interactive commands. The video data stream is the actual video material that was requested by the end user, being sent from the Video Server to the Video Client. This stream may require a very high bit rate, up to 6 Mbits/s, and Video Client may be very sensitive to delays in receiving of that stream. Interactive commands are the means of adding interactivity to the Video Client while displaying a movie material. The interactive commands are being sent to the Video Server in order to change the state of video data stream, e.g. to pause are to resume streaming. The commands do not have bandwidth requirements but similarly to video data stream they request small delays. The interactive commands should never be lost over the network, or be reordered.

1. System management and administration

The Video-on-Demand system performs multiple management and administration tasks, such as automated contents production, Video Server load statistics, updating system database. The contents production is very time consuming, as it consists of such tasks as digitizing and compressing of video data, metadata creation based on video content, or copying video files on remote filesystems of Video Servers. The number of additional task related to the contents production is also significant. The automation of the process of video content production speeds up the lengthy process as well as provides continuous way of keeping Video-on-Demand system consistent with its database. The automation should begin from the capturing of the video material from live source: satelite or TV feed, video camera or VCR. However, finding the way of marking analog broadcast video is not an easy task. Automated Contents production updates the database and a Video Server filesystem each time new video clip is being digitized. It will digitize, compress and copy video file to a Video Server, as well as may parse this video file to obtain additional metadata information, allowing content-based search. All created metadata information about the video clip will be put into the database. Even with totally automated process of video content production there will still be needed some manual update to the database. The Database Server should provide means for updating the database, e.g. in case of adding or removing the Video Server from the system. The Database Server would be also responsible for gathering statistics from the Video Servers (e.g. which video clips are accessed the most) and basing on that information the Database Server should decide about changing the state of the Video-on-Demand system (e.g. make copy of the most accessed video files on other Video Servers).

1.Database Server

NPAC Database Server is build upon an HTTP server. Therefore the integration of Video Client and Database Server components during the searching phase is implemented by Web interfaces. Database Server extends the functionality of the HTTP server by providing the access to its local database. Database Server gives the means to translate user queries into the database SQL queries, as well as to translate SQL queries results into search results being sent back to the Video Client. The end user queries and search results are sent between the Database Server and the Video Client using standard HTTP protocol. The user queries have to invoke searching engine, i.e. parsing of metadata information about video clips existing in the database, and this is done using Common Gateway Interface, being standard way of adding new logic to the basic functionality of an HTTP server.

NPAC Video-on-Demand system uses an Oracle database to store metadata information. The CGI scripts that have to process users category and content search for video clips use scripting languages with database API, such as OraPerl.

NPAC Database Server was designed to perform management tasks of whole Video-on-Demand system. Therefore, apart of metadata information, the database stores also a system management information.

1.Video Server

NPAC Video Server was designed to process requests from multiple Video Clients simultaneously. Video Server works as multithreaded application, but depending on the platform on which is being implemented uses different type of multithreaded architecture. NPAC Video Server is provided on two platforms: NT workstation and SGI workstation. The SGI version uses the model where there are two separate threads of execution for each Video Client. The first thread is responsible for video data streaming to the Video Client. The other thread handles interactive commands sent to the Video Server from the Video Client. This includes also sending of acknowledgments back to the Video Client. The model works well with small number of simultaneous client requests. The problem arises when the number of requests augments because in this case the number of threads grows proportionally to the number of client requests. The model that is better suited for large number of simultaneous client requests is implemented in Video Server for Windows NT workstation. The Video Server for NT has constant number of execution threads, independently from the number of client requests being actually processed. This solution is possible due to a new concept of I/O completion ports, being a part of Windows NT operating system. In the Video Server there are two types of asynchronous I/O operations: network and file asynchronous I/O. The events related to processing of these operation like a completion of asynchronous operation are queued and handled by I/O completion ports. This allows processing of the simultaneous Video Client request in the organized fashion without need for separate execution threads for each client request. The second model does not loose the robustness of the first model, being much more efficient in case of large number of simultaneous client requests.

The NPAC Video Server is configured to talk to ATM and Ethernet interfaces. The ATM interface increases the number of Video Clients that could be handled simultaneously. In case of MPEG video streams, the number of simultaneously handled requests goes up to 20. In the most common situation, video data is being streamed from the ATM interface to an ATM switch, which sends it further to an Ethernet switch, and from there it is sent to Ethernet card of multimedia PC station running Video Client.

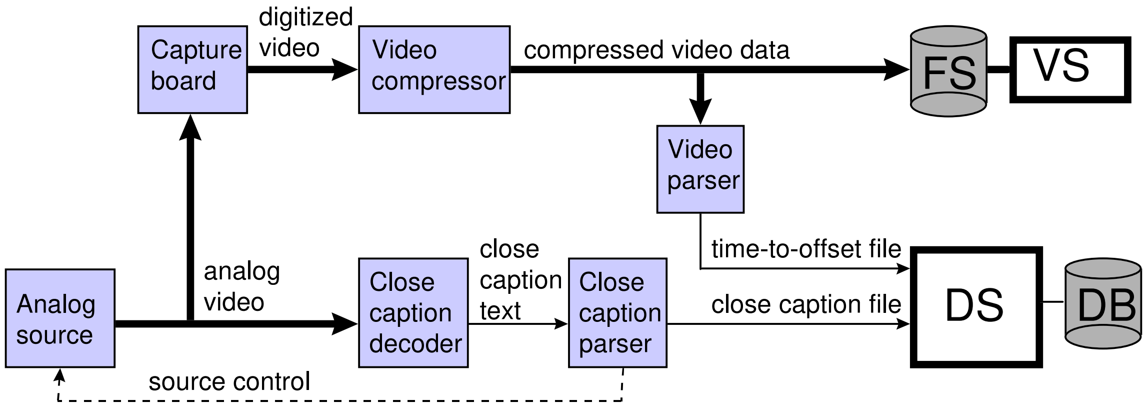
1.Video Client

The NPAC Video Client is being presented in detail later in the paper. The front-end functionality of the Video Client is providing end users with interfaces to search for particular video material, as well as to display the video material and interact with the display process. To support the front-end functionality the Video Client has to enable communication with Database Server and Video Servers, perform real-time network operations, real-time video and audio streams decompression, finally render multiple streams in synchronized way. Each of these operations may be CPU-high-consuming. Real-time constraints in every part of Video Client execution, and strong dependability between these parts make the issue of interconnection between networking, decompression and display critical for the performance of whole Video-on-Demand system. There could be Video-on-Demand system without Database Server - it would loose its search capabilities. The Database Server would not be so important in case of small Video-on-Demand systems with one Video Server and limited number of video material. There are also examples of Video-on-Demand systems without Video Servers where role of streaming of video data is undertaken by HTTP servers. Video Client is a least common denominator for all the types of Video-on-Demand systems, and the tasks that the Video Client has to perform are very similar in all these systems.

NPAC Video Client performs video material search tasks using interface of a WWW browser, with HTML pages automatically generated by Database Server. A result of these operations is one metadata file. The browser launches Video Client application responsible for video data streaming and sends the metadata file as a parameter to the Video Client. Some level of communication between the browser and the Video Client is sustained, using CGI scripts. An example of such a communication is invoking from the HTML page related to a video clip seeking messages to the Video Client actually displaying the clip.

1.Automated Content Production

NPAC video content production process consists of several tasks. The source of the video content can be video tapes, video camera or satelite feed. The portions of analog video stream are being captured and digitized by the encoding workstation. The encoding workstation compresses the digitized video material to the appropriate video format such as MPEG, AVI, H263 using appropriate hardware or software compressors. The compressing process requires a lot of computational power and the hardware support may speed up the process significantly. For example, compressing of a raw digitized video data on a computer with 100MHz Pentium processor may take two hours for a 1-minute-long video clip, if this is done in software. Instead, hardware MPEG compressors are able to compress the same data real-time, while capturing the video material from a live source. In fact, also the first element of content production process, digitizing, cannot be performed on the encoding workstation without hardware support, i.e. specialized capture boards. The simultaneous, hardware supported process of video media digitization and compression removes one problem in automation of content production. The problem that still remains is how to automate the process of selecting interesting parts of video material from the live analog source. The efficient way of image recognition is not possible yet. The solution could be the extended close caption information included into broadcast video signal, such as start and end of television programs, start and end of an advertisement, change of scene. The extended close caption service is one of the US standards for broadcast TV, but the standard is not used by major broadcasters.



Apart from digitization, the analog video material has to be processed also in order to retrieve the close caption text from the video signal. The additional hardware, the close caption text decoder, is needed. The close caption text is a textual description of the video content. As the close caption text is included in the video signal itself, the video content and close caption text are well synchronized. The close caption text decoding is performed real-time and allows to interleave the close caption text with the time stamps. The interleaved information is put into close caption file and this file is sent to Database Server to be stored into metadata database. The close caption text is a part of the metadata information about video clip, enabling user queries to the Video-on-Demand system to be content-based.

Another metadata information that is put into the database during contents production allows an end user to efficiently access any position within the video material. The video data is stored on Video Servers in a compressed way. The compression process can change sizes of various portions of the same video clip very differently. There is a need for a tool that produces a relation between position (time or frame) in an uncompressed video material and its compressed representation (file offset). This relation information is just the part of metadata information about video clip. If provided, the Database Server sends it to the Video Client that wants to display particular movie. Basing on this information, the Video Client may request from Video Server various portions of video material and the Video Server will start streaming of compressed video data from the specified file offset. Without that information, the Video Client may not be able to translate the time to the file offset itself, as it may require parsing of the video data file, not existing on the Video Client side. The video parser tool is a component of contents production process that generates the required metadata information. The input of the video parser is compressed video data. Parsing of the compressed video, sometimes including its decompression, produces the time-to-offset relation file that is inserted into the metadata database. The compressed video file is copied to a Video Server and information about the video file name and location is also added to the metadata database.

1.Close captioning

Captioning is an electronic (or manual) process which converts the audio portion of a television program into written words. These words usually appear at the bottom of a television screen with decoding capability or through a peripheral decoder. Unlike subtitles, captions are designed specifically for the deaf and hearing-impaired: in addition to on-screen dialogue, sound effects and off-screen nuances that are significant to a particular scene may also be described.

The US standard, NTSC TV broadcast signal consists of 525 lines at 30 frames per second. Individual images forming the motion picture are divided by a 21-line-wide horizontal vertical bar called Vertical Blanking Interval (VBI). VBI can carry additional information usually for the synchronization purposes. Electrical Industries Associate (an American standard body) defines a standard for transmission of the closed-caption characters in the 21st line of the Vertical Blanking Interval. This service is also known as the line 21 service.

There are 4 closed-caption channels, 4 text channels and one Extended Data Services channel

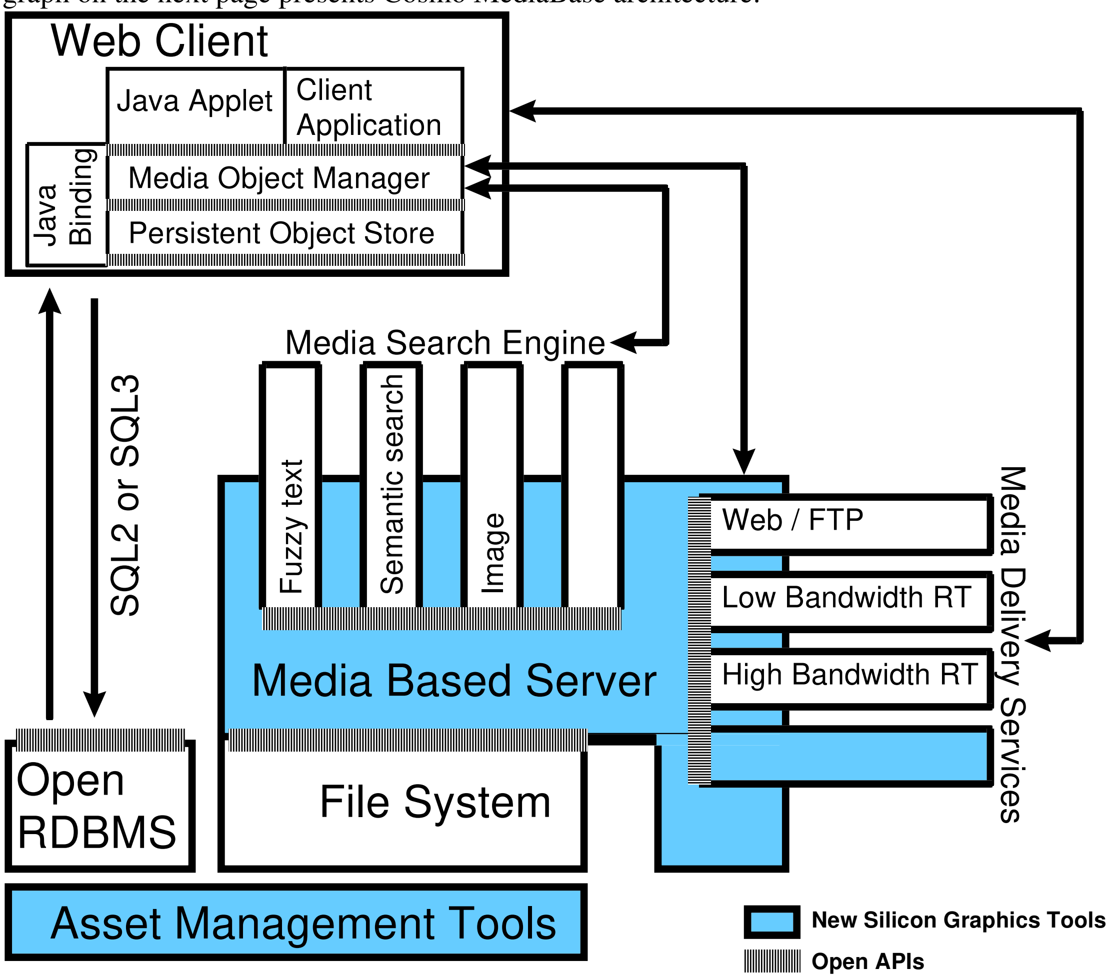
(XDS). Closed-caption channels as well as text ones may carry characters for 4 different channels each in different language. Extended Data Services is supposed to carry various kinds of information such as weather condition, general information about the current television program (summaries, time to end etc.). However, currently only one closed-caption channel is being used by broadcasters and during the contents production that channel is decoded in order to add close caption information to metadata about video clip.
Closed-captioned video material carries itself a textual information about the content of the video. The audio component which is directly correlated to the image is translated into the text. The NPAC Video-on-Demand system extracts closed-caption stream from the TV signal and assigns time stamps to it. In such a way search for key word would return a time value in the video stream at which the key word appeared, and would allow an end user to display the video material from the moment the keyword has been said.

A.Market VOD Architectures

Actually there are only a few existing architectures supporting fully VOD functionality. The short description of two such architectures is presented here, namely:

- Cosmo MediaBase by Silicon Graphics
- FlashWare by Precept.

Cosmo MediaBase was chosen as an example because its architecture is very similar to NPAC Video-on-Demand one, while FlashWare was chosen because it provides a client solution for PC stations in Windows multimedia environment, similarly as the NPAC Video Client does. The graph on the next page presents Cosmo MediaBase architecture:

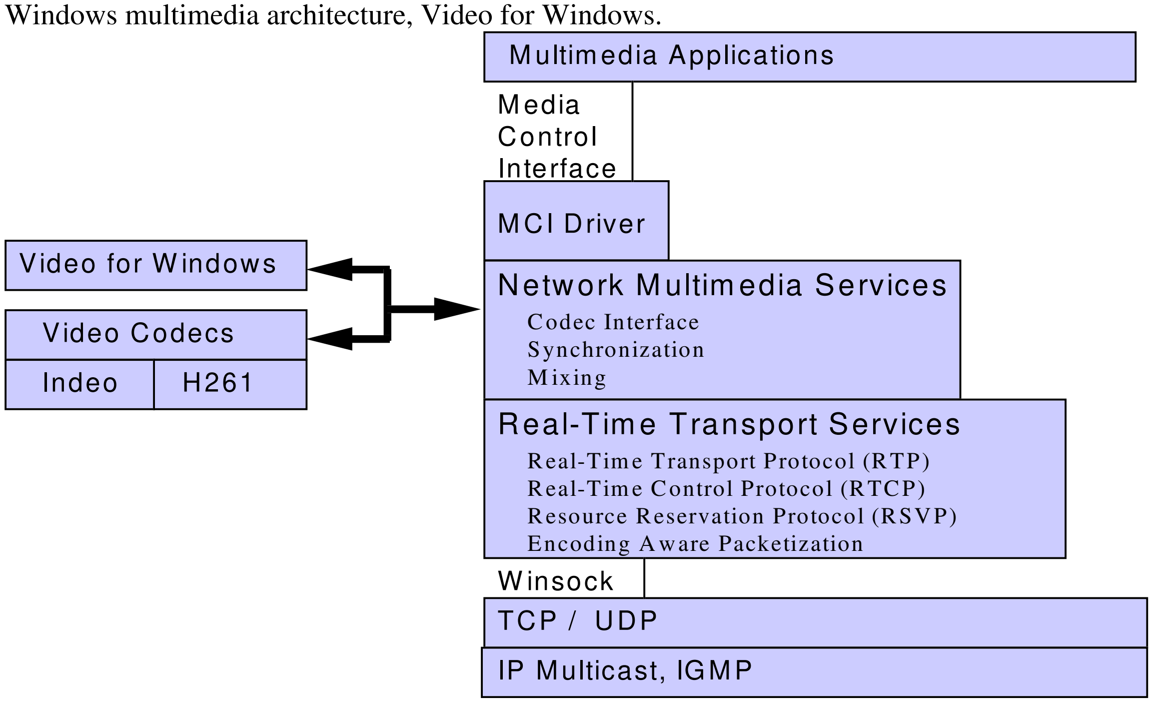


Cosmo MediaBase enables search for multimedia data using the metadata concept, which can be

found in NPAC VOD system. The MediaBase Web Client performs the search i.e. retrieves the metadata information by communicating with metadata database. Like in NPAC VOD system, searching and video data streaming tasks are separate on both server and client side. Web interfaces enable search for video material in Media Based Server which is a host for the metadata database. Client Application is responsible for actual streaming of video data from the Server. The Media Based Server exposes to separate APIs: one for search, another for video delivery. Finally, Media Based Server performs management tasks.
The differences between MediaBase and NPAC VOD architectures are following:

- in MediaBase, Web Client search interface and the Client Application are communicating using Java, while in NPAC system this communication uses only simple CGI scripts
- both Video Server and Database Server components of NPAC VOD architecture are merged in one Media Based Server, an HTTP server with search and video delivery functionality added. This the major design difference, and NPAC VOD architecture seems to be less redundant (Database Server is not duplicated).

FlashWare is another multimedia networking software having Video-on-Demand functionality. Flashware is devoted to Windows operating system, making use on both server and client side of Windows multimedia architecture, Video for Windows.

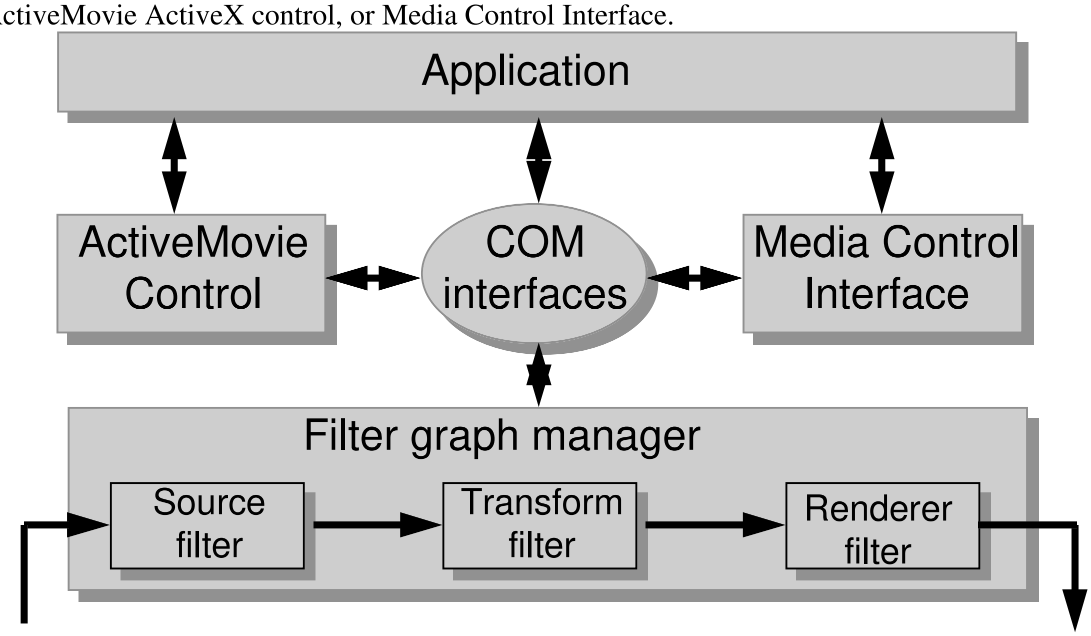


The graph presents a layer architecture of both Video Server and Video Client components of FlashWare system. The system is symmetric, in the contrary to both systems described before in this chapter. The FlashWare has few unavoidable limitations because it uses Video for Windows. Namely, all compression formats that are supported have to be wrapped up to Audio Video Interleaved (AVI) format. It means there can be no support for MPEG-1 or MPEG-2 format without redesigning of the architecture. The architecture has to rely on own proprietary codecs because the codecs available in Video for Windows architecture are generally inefficient. The advantages of the FlashWare solution relate first of all to the networking scheme. FlashWare has implemented Real-Time Transport Protocol as well as a TCP/IP stack with IP multicast. The stack was optimized for multimedia traffic.

I.ActiveMovie Technology

A.General Description

ActiveMovie is an architecture that controls and processes streams of multimedia data. It is also a runtime that uses this architecture to allow users to play digital movies and sound encoded in various formats, including MPEG-1.
The playback capability makes use of video and audio hardware cards that support Microsoft DirectX set of application programming interfaces (APIs). ActiveMovie also plays movie files in audio-video interleaved (.avi) or Apple QuickTime (.mov) format.
The ActiveMovie architecture defines how to control and process streams of time-stamped multimedia data by using modular components called filters connected in a configuration called a filter graph. An object called the filter graph manager is accessed by applications and controls how the filter graph is assembled and how data is moved through the filter graph.
The filter graph manager provides a set of Component Object Model (COM) interfaces to allow communication between the filter graph and the application. Applications can directly call the filter graph manager interfaces to control the media stream or retrieve filter events. Or they can use the ActiveMovie ActiveX(.ocx) control for higher-level programming.
ActiveMovie is the architecture to use for most new multimedia applications for Windows 95 or Windows NT. With a few exceptions, it replaces multimedia playback services, APIs, and architectures provided by Microsoft in earlier versions of the Windows Software Development Kit (SDK). The first release of ActiveMovie does not provide a corresponding replacement for every solution found in the previous multimedia technology. For example, there is no video capture capability built into the runtime. In these cases, ActiveMovie gives developers an opportunity to use its architecture to provide custom solutions.
ActiveMovie can be accessed through the following interfaces: the COM interface, the ActiveMovie ActiveX control, or Media Control Interface.



Because of the flexible, modular design of ActiveMovie architecture, there are many potential uses and applications for filter graphs. Some examples include filter graphs that implement video capture, control of remote devices such as VCRs, animation sequencing, and MIDI recording and editing.

A.Component Object Model in ActiveMovie

All components of the ActiveMovie filter graph architecture are implemented as COM objects. This includes the filters through which data is passed, and filter components that serve as a connection between filters or allocate memory. Each object implements one or more interfaces, each of which contains a predefined set of functions, called *methods*. An application calls a method, or other component objects, to communicate with the object exposing the interface. Filter graph architecture uses COM interfaces because they have the following properties:

COM interfaces are publicly defined. This means that any filter that implements the correct predefined interfaces will work in a filter graph without any knowledge about the other filters, because all filters are built with the same interface specifications.

COM interfaces do not change after definition. A base set of interfaces are guaranteed to work; additional interfaces may be introduced to cover additional services. This definition prevents version problems.

COM interfaces must have all methods implemented by any object that exposes. This assures that calling a method on the interface of an object will not generate an error.

COM interfaces are discoverable. All COM objects support a method called **QueryInterface** that allows an external component to discover if an interface is present and retrieve a pointer to it.

COM interfaces are implemented by the object that exposes the interface (they do not contain an implementation themselves). The interface is essentially a contract for the functionality. Objects like the filter graph manager, or Microsoft filters, have implemented interfaces that can be accessed. A developer must implement the interfaces on custom-build filters.

Microsoft ActiveMovie provides a framework that simplifies the creation of Component Object Model (COM) objects.

ActiveMovie components are supplied as inprocess servers (that is, servers that run in the same address space as your application). They are packaged in a single dynamic-link library (DLL), quartz.dll. Developers can use the COM framework of ActiveMovie to build their own inprocess COM servers which they can package in their own DLL(s).

Typically, a single C++ class implements a single COM class. The ActiveMovie COM framework requires that C++ classes that implement COM objects conform to a few simple rules, and that the developer provides a class factory template for each such class. The *class factory template* contains information about the class that is vital to the framework.

COM objects are created by their class factories, are reference counted during their lifetimes, and self-destruct when their reference count drops to zero. COM objects may be created in isolation, or may be aggregated with an already existing COM object. In this second case, the existing object (referred to as the outer object) maintains the reference count. The created object (referred to as the inner object) is not reference counted, but will be destroyed by the outer object during the destruction of the outer object. (The application cannot directly manipulate COM objects; an application can only invoke the methods that the object chooses to expose through its interfaces. Typically, COM objects make several interfaces available. All COM objects must support the **IUnknown** interface that is responsible for the reference count.

The concept of a class factory is not specific to ActiveMovie; it is a common design that appears when the underlying type of the object being created is not known to the client that requests its creation. With COM objects, clients request interface pointers but know little about the underlying objects that implement that interface.

A.Filter Graphs

A filter graph is composed of a collection of filters of different types. Most filters can be categorized into one of the following three types:

A source filter, which takes the data from some source, such as a file on disk, a satellite feed, an Internet server, or a VCR, and introduces it into the filter graph.

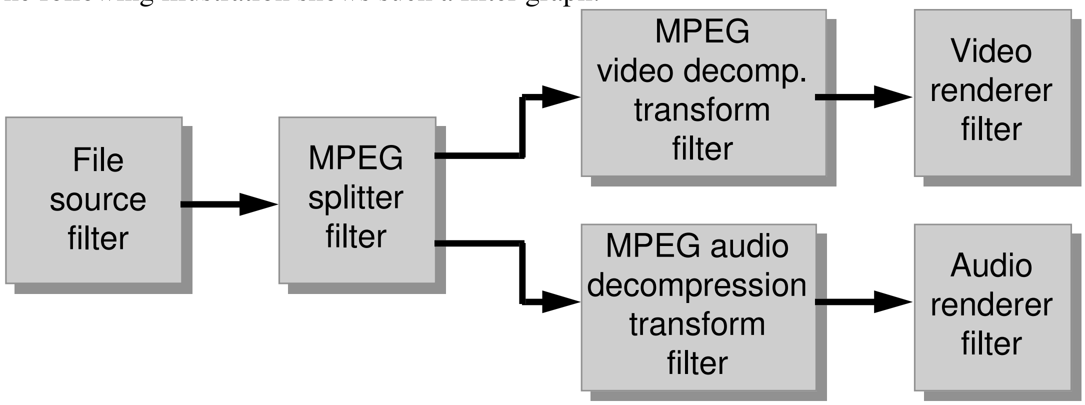
A transform filter, which takes the data, processes it, and then passes it along.

A rendering filter, which renders the data; typically this is rendered to a hardware device, but could be rendered to any location that accepts media input (such as memory or a disk file).

For example, a filter graph whose purpose is to play back an MPEG-compressed video from a file would use the following filters:

A source filter to read the data off the disk.
An MPEG filter to parse the stream and split the MPEG audio and video data streams.
A transform filter to decompress the video data.
A transform filter to decompress the audio data.
A video renderer filter to display the video data on the screen.
An audio renderer filter to send the audio to the sound card.

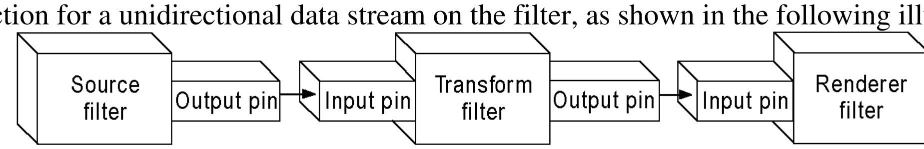
The following illustration shows such a filter graph:



Filter graphs stream multimedia data through filters. In the media stream, one filter passes the media *downstream* to the next filter. An *upstream filter* describes the filter that passes data to the filter; a *downstream* filter describes the next filter in line for the data. This distinction is important because media flows downstream, but other information can go upstream.
To make a filter graph work, filters must be connected in the proper order, and the data stream must be started and stopped in the proper order. The filter graph manager connects filters and controls the media stream. It also has the ability to search for a configuration of filters that will render a particular media type and build that filter graph. Filter graphs can also be preconfigured, in which case the filter graph manager does not need to search for a configuration.

A.Stream architecture

Stream architecture defines objects and interfaces that exchange streams of time-based data. In particular, it defines interfaces for the following requirements:
Connecting filters to other filters
Negotiating data types
Transporting data between filters
Synchronizing presentation of data
Graceful degradation of rendering in cases of insufficient resources (that is, *quality-control management*)
The two basic components used in the stream architecture are filters and pins. A *filter* is a COM object that performs a specific task, such as reading data from a disk. For each stream it handles, it exposes at least one pin. A *pin* is a COM object created by the filter, that represents a point of connection for a unidirectional data stream on the filter, as shown in the following illustration:



Input pins accept data into the filter, and *output pins* provide data to other filters. A source filter provides one output pin for each stream of data in the file. A typical transform filter, such as a compression/decompression (codec) filter, provides one input pin and one output pin, while an audio output filter typically exposes only one input pin. More complex arrangements are also possible.
Reference clock synchronization is accomplished by implementing the **IReferenceClock**

interface on any filter that has a reference clock. For example, because sound cards are predominantly used for reference clocks, the audio renderer filter implements this interface, which essentially allows any caller to register for the receipt of time notifications.
The ActiveMovie stream architecture provides for graceful adaptation of media rendering to overloaded or underloaded media streams. The **IQualityControl** interface is used to send quality-control notifications from a renderer filter either upstream, eventually to be acted on by some filter in the graph, or directly to a designated quality control manager. The base classes implement the passing of quality control notifications upstream by providing the **IQualityControl** interface on the output pins of filters. Quality control notification uses a **Quality** structure, which indicates whether the renderer is overloaded or underloaded. A filter capable of, say, discarding samples to relieve and overloaded condition, can then act on this notification. This is typically done by a source filter but could be done by other filters. For example, the ActiveMovie AVI Decoder filter skips samples until the next key frame when it receives a quality control notification.
The stream architecture allows applications to communicate with the filter graph manager; it also allows the filter graph manager to communicate with individual filters to control the movement of the data through the filter graph. Using the stream architecture, filters can post events that the application can retrieve, so an application can, for example, retrieve status information about a special filter it has installed.
The filter graph manager exposes media control and media positioning interfaces to the application. The media control interface, **IMediaControl**, allows the application to issue commands to run, pause, and stop the stream. The positioning interface,**IMediaPosition**, lets the application specify what section of the stream to play.
Individual filters expose a media control interface so that the filter graph manager can issue the run, pause, and stop control commands. The filter graph manager is responsible for calling these methods in the correct order on all of the filters in the filter graph. (The application should not do this directly.)
The position commands are handled less directly. The filter graph manager gets called by the application to, for example, play a specified length of media stream starting at some specified stream time. However, unlike the media control interface, only the renderer filter exposes a media position interface. Therefore, the filter graph manager calls only the renderer filter with positioning information. The renderer then passes this position control information upstream through media position interfaces exposed on the pins, which simply pass it on. The positioning of the media stream is actually handled by the output pin on the filter that is able to seek to a particular position (for example, a file source filter) because pins are responsible for the data transport mechanism.
Position information is passed serially upstream because there may be filters between the renderer and the source filter that require position information. Consider a transform filter that is written to perform some video or audio modification only during the first 10 seconds of a video clip (for example, increasing the volume or fading in the video). This filter probably needs to have information about where the stream is starting so that it can determine its correct behavior. For example, it should not perform if the start time is after the first 10 seconds, or it should adjust accordingly if the start time is within this duration.

A.COM Interfaces

The ActiveMovie™ COM interfaces comprise the schematic of an architecture for streaming time-stamped media. The filter graph, through which media flows, is composed of objects, such as filters, pins, media samples, allocators, and enumerators, that work together. COM interfaces are implemented on these objects and are called by other objects with which they interact. The filter is the only filter graph COM object that has a CLSID; all other objects in the filter graph support COM interfaces and are created as needed by the filter. Filters and their supporting object must implement their COM interfaces and a class library is available for help in that task.

The filter graph manager, on the other hand, has a CLSID and supports several fully implemented interfaces.
The ActiveMovie COM interfaces can be categorized as follows:

Filter graph manager interfaces, which are fully implemented and used by applications to create, connect, and control filter graphs and by filters within the filter graph to post event notifications and to force reconnections when needed.

Filter and pin interfaces, which must be implemented by the filter. They comprise the methods exposed by filters for communicating with the filter graph manager, connecting with other filters, passing data downstream (from source filter to renderer filter) and passing quality control and media positioning information upstream (from renderer to source).

Enumerator and media sample interfaces, which are interfaces on objects created temporarily for passing information.

Control interfaces, which are exposed by filters and the filter graph manager to enable the starting, stopping, and positioning of media in the stream. The control interfaces on the filters must be implemented when writing a filter, whereas they are already implemented on the filter graph manager.

The filter is the main COM object and has a class ID (CLSID) and name registered in the registry. Filters must provide access to their pins and otherwise communicate with the filter graph. They must also allow the filter graph manager to manage the data flow by accepting state change messages.

At a minimum, a filter exposes the **IFilter** interface. This interface provides methods that allow the enumeration of the pins on the filter and return filter information. It also provides the inherited methods from **IMediaFilter**; these methods allow control of state processing (for example running, pausing, and stopping) and synchronization and are is called primarily by the filter graph manager.

Pins do not normally have registered class identifiers and are usually created by the filter object on which they reside. They are exposed externally by the filter, which includes a method (**IFilter::EnumPins**) to hand out pointers to the **IPin** interfaces of its pins, normally to the filter graph manager. The filter graph manager is responsible for connecting pins by calling an **IPin** method on one of the pins with a pointer to the other pin. Once pins are connected, each pin holds a pointer to the pin to which it is connected.

Media sample and enumerator interfaces are temporary objects created to pass information or data between objects. They do not have class identifiers.

The media sample interface is created from the memory allocator, which uses the media sample object as its unit of exchange. It has no class identifier. It is the unit of media data that is passed from one filter to the next via the memory allocator shared by two connected pins.

Enumerators in ActiveMovie are based on the OLE **EnumXXX** interfaces. They include the Next and Prev methods, which tell the enumerator what item or items to return; the Skip method, which skips one or more items; and the Clone method, which makes a copy of the enumerator. Enumerators are used to present lists of items such as filters in a filter graph, pins on a filter, or media types that are preferred by a pin.

Control interfaces allow the filter graph manager to coordinate the activities of the data stream with filters. They are exposed by the filter graph manager, filters and pins.

A.Data flow

Data flow in the filter graph occurs in the following ways:

Media sample data flows from one filter to the next—originating at a source filter and terminating, eventually, at a renderer filter.

Control information, such as end-of-stream and flushing notifications, travels with the media data stream from filter to filter.

Notification events flow from the filters to the filter graph manager and, optionally, to the application.

Filter graph control data flows from the application to the filter graph manager and finally to the filters themselves.

Quality control data originates in the renderer and flows upstream through the filters until it finds a filter capable of cutting back or increasing the media data flow. It may also flow directly to a quality control manager if one is registered.

ActiveMovie filters pass media data downstream, that is, from the output pin of one filter to the input pin of the next filter. The flow and control of the data is effected by the interfaces on those pins and the filters themselves. Data streaming activity is serialized by the filters; all data streaming calls for a given pin are explicitly serialized and usually originate from a single thread.

Data is passed from the output pin of one filter to the input pin of the next. The two connected pins agree upon a common method of exchanging data, called a transport. The most common transport is the local memory transport. This transport is implemented if the input pin supports the **IMemInputPin** interface. The ActiveMovie class library base classes assume this transport.

Filters must follow protocols in order to pass and receive media samples. The connected pins must agree upon the allocator to be used, must have a means of passing the data, and must follow the correct procedure for holding onto a sample or releasing it back to the sender. Media samples are data objects that support the **IMediaSample** interface. They are obtained from an allocator, most likely represented by an object supporting the **IMemAllocator** interface.

An output pin typically exposes the following interfaces:

IPin methods are called to allow the pin to be queried for pin, connection, and data type information, and to send flush notifications downstream when the filter graph stops.

IMediaPosition allows information about the stream's duration, start time, and stop time to be relayed from the renderer. The renderer passes the media position information upstream to the filter (typically the source filter) responsible for queuing the stream to the appropriate position.

IQualityControl passes quality-control messages upstream from the renderer to the filter that is responsible for increasing or decreasing the media supply.

An input pin typically exposes the following interfaces:

IPin allows the pin to connect to an output pin and provides information about the pin and its internal connections.

IMemInputPin allows the pin to propose its own transport memory allocator, to be notified of the allocator that an output pin is supplying, to receive media samples through the established allocator, and to flush the buffer. This interface can create a shared memory allocator object if the connected pin does not supply a transport memory allocator.

The standard transport interface, **IMemInputPin**, provides data transfer through shared memory buffers, although other transport interfaces can be used. For example, where two components are connected directly in hardware, they may connect to each other by using the **IPin** interface, and then seek a private interface that can manage the transfer of data directly between the two components.

Control data originates at the application and is passed to the filter graph manager. At the COM level, this is handled by filter graph manager interfaces in the control.odl type definition library. Examples of control data are calls to the **IMediaControl** interfaces, such as **IMediaControl::Run**, **IMediaControl::Pause**, and **IMediaControl::Stop**. The **IMediaPosition** and **IMediaSelection** interfaces provide access to moving forward or backward in a media stream.

The most important thing to understand about the flow of control data is that it should always pass through the filter graph manager first. This is because there is usually an order that must be followed in controlling the filters in the filter graph to avoid deadlocks and other problems. The filter graph manager is dedicated to handling these conditions.

A.ActiveMovie versus Video for Windows

A subset of Media Control Interface (MCI) commands is available for backward compatibility with Microsoft Video for Windows 1.x and compliance with the OM-1 MPEG MCI specification.

The ActiveMovie model of calling COM interfaces to perform multimedia services is more efficient than the Video for Windows model, which uses a driver entry point. The latter requires the overhead of a lengthy **switch** statement to decode a message identifier, whereas calls to COM methods are direct.

I.Video Client for PC

A.General Functionality

Video Client is the component of Video-on-Demand system which resides on an end user computer. Therefore the Video Client has to provide an end user with an User Interface to VOD system. If the VOD system is simple, for example it consists only of one Video Server and few Video Clients, as well as number of the video material available is non-significant, the Video Client has to expose only displaying interface. This interface could be reduced to the display window where the video clip is shown, if the system was not interactive. On the other hand, in VOD system with full functionality, the Video Client has to provide at least two interfaces: first to allow an user to search for a particular video material, second to allow the user to interact with the video being already displayed.

Having network bandwidth limitations in concern, VOD systems are making use of video compression. The reason is that raw, pixel by pixel digitized video represents very large amount of data. Fortunately, there is very strong temporal correlation between consecutive video frames, as well as the space correlation between adjacent pixels in one video frame. These correlations indicate that there is a large redundance in raw video data, and this data can be very sufficiently compressed using appropriate compression schemes. An use of the best compression schemes proved compression ratio for average video sequences equal to 100:1, 200:1. Therefore VOD systems are using video compression extensively.

For the Video Client it means that it has to provide one more functionality: decompression on-the-fly of video data received from the network. The decompression can be a CPU-consuming process. Thus there is a need of incorporating fast video decoders (software or hardware) into Video Client or the Video Client has to be able to drop some frames during playback. The best solution would be if the Video Client sent notification messages to its Video Server, making the Video Server to drop frames that will not be displayed anyway.

It is important that Video Clients support various video formats, for various network conditions. First of all it relates to required data rate of such a video format. An end user should be given an option of choosing from a few compression formats, which format suits the best his/her network connection. Required bandwidth influences a quality of the video and an end user should be aware of that. Then the user could decide to choose worse quality video, to be able to watch it without any breaks.

Displaying of video clips should be able to take advantage of all additional hardware support, such as full screen support of some video cards.

The Video-on-Demand system should be interactive, and it means that Video Client should provide an User Interface for this interactivity, as well as low level functions processing interactive commands and sending them to Video Server. A scope of the interactivity for the Video Client should be VCR-like functionality, i.e. it should be able to manage commands such as Play, Stop, Pause, Resume, Seek To, Step Forward, Step Back, etc.

A.NPAC Video Client - user interaction

NPAC VOD system allows search and delivery of multimedia data. Firstly, an end user is going to perform the search operations. The WWW browser is used to provide an end user with the search interface. The user completes a form by choosing the category of a video clip, the way the video clip is encoded, as well as a keyword that will be used to perform content-based search. Each of these form inputs are optional - the form could be left empty, what would mean that user is going to be informed about all movies available in the VOD system. Completing of the search form is the first stage of user interaction with the VOD system. This stage using Netscape as WWW browser is shown in Appendix A1.

Normal consequence of a user query to the VOD system is returning back results of performed search. The results, generated by CGI script on Database Server talking to the metadata database, are displayed again using WWW browser. These results consist of list of video clips that satisfied the user query (category), they consist also of positions in the video clips that are related to the search keyword (content). Now the end user may choose from those lists and make the Video Client to display the chosen video from the appropriate starting point. This is the second stage of user interaction, and is shown in Appendix A2.

Now comes the moment the Video Client starts actually display the video clip. The display window appears, as well as the VCR-like interface is provided with Play, Stop, Pause buttons and Seek slidebar. Additionally, the WWW browser displays a close caption text associated with the video clip. The close caption text is correlated to displayed video, and it gave an idea of a very useful and accurate seeking interface to the Video Client. Namely, all sentences of close caption text are time-stamped and by clicking on any of those sentences we may move the displaying of the video clip to the position the sentence is synchronized with. The video displaying stage is shown in Appendix A3.

A. Video Client for Windows 3.1

The implementation of Video Client for Windows 3.1 made use of Video for Windows (VfW) architecture, multimedia application development kit designed few years ago by Microsoft for Windows 16bit environment. Unfortunately, it could not use much of it, because this architecture was designed to decompress and display video files existing in a local filesystem. Also decompressors available through the VfW architecture were not presenting sufficient quality.

The implementation of the Video Client for Windows 3.1 used only the high level API of Video for Windows architecture, Media Control Interface (MCI). The MCI interface represents a set of control funcions such as Play, Stop, Seek To Offset, and allows to have a common user interface to control various types of multimedia resources (hardware and software): video, wave or midi files, CD audio, VCR, camera and others. Because it was designed for such different types of resources the Media Control Interface does not include any low-level, video data related interfaces. The MCI does not provide any functionality itself. The implementation of the MCI interface must be provided by the developer, what means that the developer must build a MCI driver that will support MCI API, and install it in Video for Windows environment.

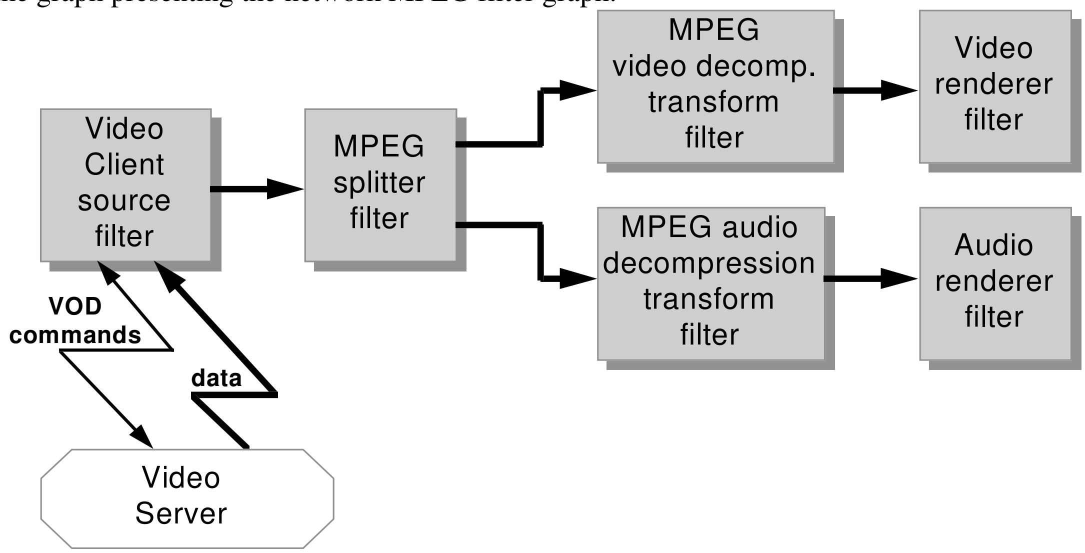
Because of inflexibility of the Video for Windows architecture the Video Client for Windows 3.1 supports only one video format, MPEG-1, and works only with the PCMotion hardware MPEG decoder. Still, it has an excellent quality that cannot be achieved with software MPEG decoders.

The Video Client for Windows 3.1 has 3-level architecture. The top level is an end user application providing the user with Graphics User Interface (GUI). The standard Microsoft's Media Player provides sufficient GUI and the application is used as the top of the Video Client. This application translates mouse events into MCI messages that are sent to the intermediary level of the Video Client architecture: MCI driver. The MCI driver is basically needed only to gain the free GUI from the Media Player. The MCI driver performs some initialization procedures, launches the real Video Client application, and from that moment resends MCI messages to the Video Client application. The Video Client application is the bottom level of the Video Client architecture for Windows 3.1. The Video Client application is responsible for actual networking and decompressing of video stream. The networking is done by implementing of custom VOD protocol between Video Server and Video Client, which is described in section related to Video Client in ActiveMovie. Processing of MPEG stream is done by accessing PCMotion decoder API. An user space buffer is designed to minimize bad effects of bursty network traffic and give the source of data for PCMotion decoder.

A. Video Client in ActiveMovie

1.Filter graph design

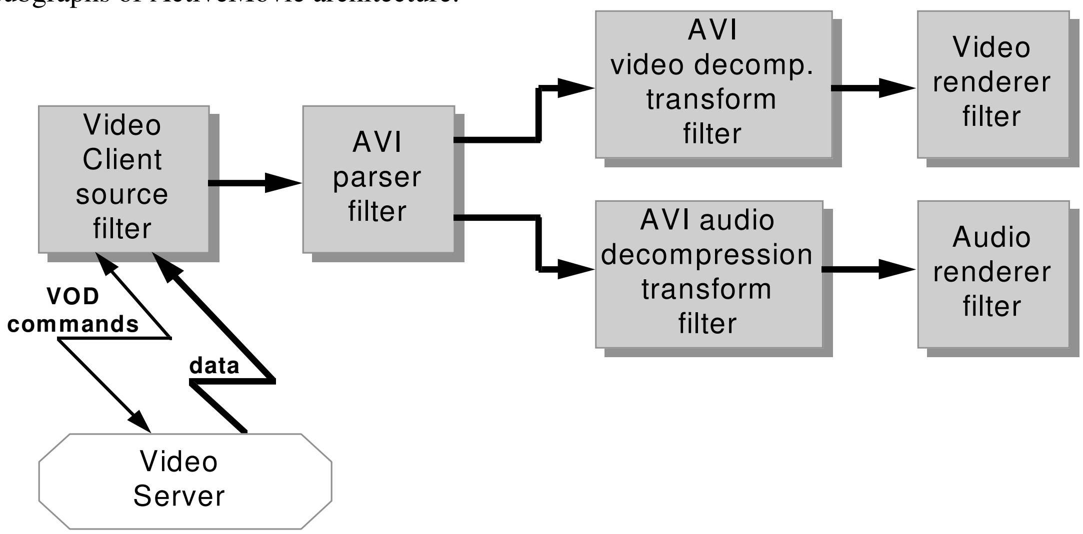
The Video Client for Windows 95/NT was designed to use extensively ActiveMovie architecture. The part of the Video Client encompassing most of its functionality is Video Client filter graph. The Video Client replaces functionality of ActiveMovie file source filter, with new functionality of source filter reading video data from the network. The first filter graph built using Video Client source filter was working with Microsoft software MPEG-1 decoder. This is the graph presenting the network MPEG filter graph.



In this filter graph all filters except of the source one are provided by ActiveMovie architecture. The Video Client source filter connects to Video Server, initiate video data streaming and interacts with Video Server using custom VOD protocol. The MPEG-1 stream is then copied to video data buffer, which major goal is to eliminate effects of "burstiness" of network streaming in displaying video data (video or audio breaks). Video data buffer can store few seconds of video. The video data buffer is being read by the filter that Video Client source filter connects to. In this filter graph this is MPEG splitter filter. MPEG splitter filter is responsible for demultiplexing of MPEG-1 System stream coming from the Video Client source filter. Then two separated streams: MPEG-1 Video and MPEG-1 Audio are being transmitted from the parser filter to their respective decompressor filters. Finally, video and audio renderer filters perform tasks of video and audio playback. In this model, the Video Client filter is connecting to two other components, in totally different ways. It connects to Video Server which has nothing to do with ActiveMovie architecture - the custom VOD protocol support control and data flow. To connect to an ActiveMovie parser filter, Video Client source filter has to implement ActiveMovie COM interfaces, and conform to connection and data flow mechanisms of ActiveMovie (MPEG splitter filter is actually such a parser filter as it does parsing of MPEG-1 System stream to retrieve video and audio streams).

As every ActiveMovie filter, Video Client may own pins that will be responsible for communication between neighbor filters. The Video Client has one neighbor filter, a parser filter, therefore it has to provide one output pin to connect with a input pin of the parser filter. The major COM interface enabling communication between the parser filter and the Video Client source filter is IAsyncReader interface exposed by the output pin of the Video Client source filter. Parser filters are requesting multimedia streams from a source filter using the IAsyncReader interface. The MPEG splitter invoke IAsyncReader interface methods on the Video Client source filter. The same interface is invoked on the source filter by AVI parser filter as well as QuickTime parser filter. The last two use only another subset of IAsyncReader methods than MPEG, and another communication model. The Video Client source filter implements whole IAsyncReader interface, as well as implements both communication models.

Therefore, Video Client source filter can be connected and work with AVI and QuickTime filter subgraphs of ActiveMovie architecture.



The presented graph is very similar to the previous one. The Video Client source filter uses the same network communication process. The communication of the source filter with the AVI parser filter is slightly different, it uses other methods of IAsyncReader interface. The MPEG stream processing functionality in downstream filters of the network MPEG filter graph is duplicated by AVI stream processing functionality in the network AVI filter graph.

1.Video Client Source Filter classes

Two objects have to be constructed in order to incorporate networking functionality in ActiveMovie filter graphs. First object is Video Client source filter, the other is Video Client output pin. The Video Client source filter is responsible for initialization tasks, state management and handling media control messages sent to it by the filter graph manager. The source filter must be registered in the Windows 95/NT system as an inprocess server. The Video Client output pin is not registered in the system, because its existence is strictly related to existence of the Video Client source filter. When filter graph manager constructs the designed network filter graph, it adds all required filters to the graph, including the Video Client source filter. Then, the source filter creates Video Client output pin object. The Video Client source filter is a COM object and a class implementing it must conform to COM specifications. The Video Client source filter has to handle its reference count and provide interface to other objects to do it. In such a way, Video Client output pin can exist as an internal part of the source filter, with no own reference count, but using reference count of the Video Client source filter.

When the Video Client source filter has to provide functionality of the COM object, the Video Client output pin is responsible for data streaming.

An implementation of the Video Client source filter and output pin provides following classes:

- CMemReader, the Video Client source filter class
- CAsyncReader, base class for CMemReader
- CAsyncOutputPin, the Video Client output pin class, implementing IAsyncReader interface
- CAsyncIO, a class responsible for queuing and processing of asynchronous requests for multimedia data from parsers
- CAsyncRequest, a class representing a single parser request for block of data
- CMemStream, a class representing mutimedia stream, and providing methods to access the stream.

1.IAsyncReader interface

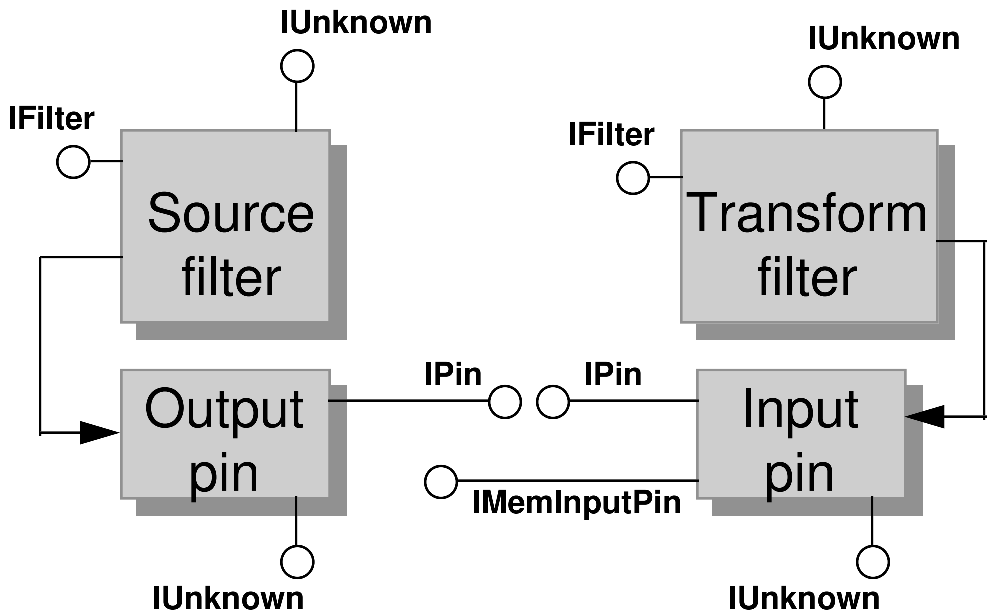
IAsyncReader interface is the most important interface supported by Video Client ActiveMovie

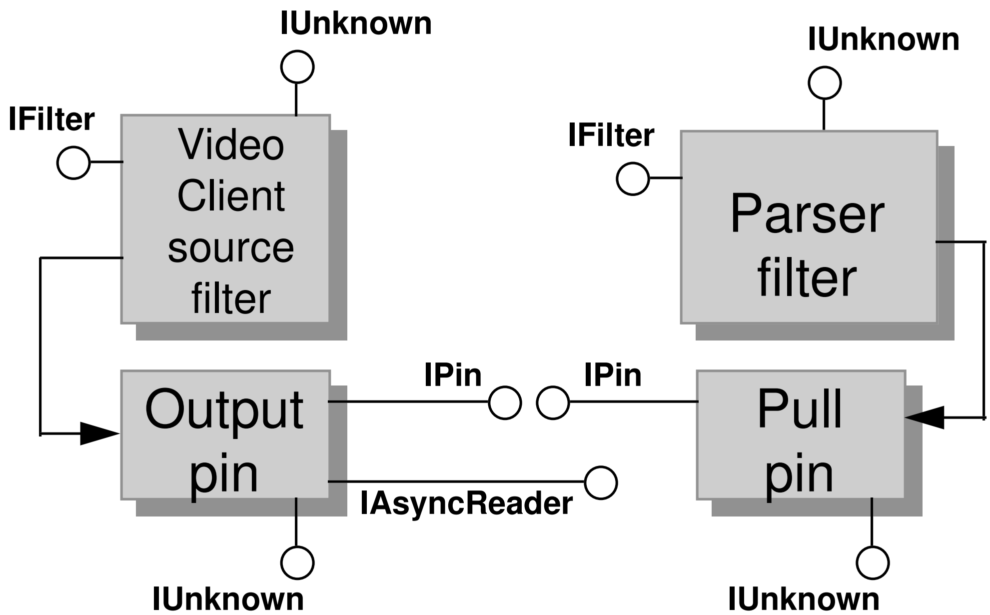
objects. It enables the connection and communication between the asynchronous source filter and parsers. The Video Client output pin implements this interface while parsers are calling messages of the interface to read multimedia data from it. The IAsyncReader interface allows multiple overlapped reads from different positions in the media stream. The IAsyncReader interface has to be implemented if a filter reads data of media type MEDIATYPE_Stream from some source. The Video Client source filter is such a filter.
IAsyncReader interface consists of the following methods:

IUnknown methods	**Description**
QueryInterface	Returns pointers to supported interfaces.
AddRef	Increments the reference count.
Release	Decrements the reference count.

IAsyncReader methods	**Description**
RequestAllocator	Retrieves the actual allocator to be used.
Request	Queues a request for data.
WaitForNext	Blocks until the next sample is completed or the time-out occurs.
SyncReadAligned	Performs an aligned synchronized read.
SyncRead	Performs a synchronized read.
Length	Retrieves the total length of the stream, and the currently available length.
BeginFlush	Causes all outstanding reads to return.
EndFlush	Ends the flushing operation.

1.Connection process





1.Video-On-Demand protocol

The custom Video-on-Demand protocol was implemented to provide an end user with interactive Video-on-Demand. The protocol handles data and control flow over the network. Data flow is unidirectional, i.e. the multimedia data streaming goes only from Video Server to Video Client. Data flow can have high network bandwidth requirements. In case of MPEG-1 stream sent over the network, the required bandwidth is 1.5 Mbit/s of MPEG-1 stream data rate plus overhead of network protocol. A research has shown 15-20% overhead of TCP/IP protocol used to transport multimedia data. Contrary to the data flow, control flow in VOD protocol is bi-directional and does not have high bandwidth requirements. In case of control flow, the most important is its reliability - the control commands sent from and to Video Client cannot be lost or reordered.
The current implementation of data and control flow uses sockets. Because both Video Server and Video Client are implemented on Windows platforms, they make use of Windows sockets. There are two sockets created: one to handle the data flow, another one to handle the control flow. There is a video data buffer in Video Client supporting the data flow. As for the control flow, currently there are following control commands:

- from the Video Client: Load Movie, Play, Stop, Seek to Offset, Terminate Connection
- from the Video Server: Stop Acknowledgments, End of File notifications.

As it can be seen, more control capabilities reside on Video Client side. It gives an end user possibility to interact with movies being sent from Video Servers.
The video protocol on Video Client side is implemented using asynchronous mode of Windows Sockets. There is a callback window created that is responsible for handling of all network (socket) events. The handle to the window is passed to Windows Sockets DLL, together with set of message values that the window would like to obtain as a result of some network events. Generally, there is two types of events the Video Client would be interested in:

- a portion of multimedia data has come on data socket, and can be read
- Windows Sockets are able to send some command to Video Server, control socket can be written
- an acknowledgment or notification has come on control socket, and can be read.

While initializing sockets, the Video Client informs Windows Sockets what events it is interested in, and what message Windows Sockets should send to the callback window. In this way, I/O operations on data and control sockets can be performed asynchronously. The callback window procedure will handle network events when they occur, but the Video Client does not have to poll for them.

1.Video data buffer

The video data buffer was built to minimize video and audio blocking effects when network delivery service is bursty. A size of the buffer may differ depending on datarate of video compression, i.e. video data buffer should be smaller for H263 video streaming than MPEG video streaming. For MPEG-1 system stream, the size of video data buffer is 1-2 Mbytes, what corresponds to 5-10 seconds of MPEG-1 compressed movie. The buffer of the same size could store up to 10 minutes of H263 compressed video clip. There is no need for such a large buffer in case of H263. Moreover, for video clips less than 10-minutes long, the Video Client could download all the movie before actually displaying it. As this is Video-on-Demand system, we are not going to do that.

The video data buffer is a circular buffer, with read and write position pointers.

The video data buffer has few thresholds, indicating how much the buffer is full or empty. There are three important thresholds:

- video pending threshold - the threshold represents number of bytes that have to be read from the network and written to the video data buffer before video displaying is activated. The higher the threshold is the longer an end user has to wait for video display started. The lower the threshold is the higher probability is that Video Client will have to stop displaying video due to lack of new data (remember bursty character of traditional network traffic). Therefore the threshold must be a tradeoff between these two conditions.
- network inactivating threshold - if the number of bytes in video data buffer exceeds the threshold, the Video Client should stop reading video data from Video Server. The situation could happen if network connection was faster than required. The threshold eliminates danger of the video data buffer overload. The threshold has to be bigger than video pending threshold. Otherwise, video would never start playing.
- network reactivating threshold - it should be little less than network inactivating threshold. The difference should not be too small, otherwise it will result in often network activating and inactivating procedures, what could impact the CPU (utilize CPU time for initialization and uninitialization functions).
- video pausing threshold - the threshold is needed when video data is not received on data socket for a long time, and video data buffer is almost empty. The threshold should cause Video Client to stop video displaying before video data buffer is empty. The Video Client should still try to read from sockets for some time, and if video pending threshold is reached before a timeout occurs, the Video Client should resume displaying of video material. This threshold is smallest from all video data buffer thresholds.

The thresholds can be adjusted according to network or CPU conditions. They could be changed dynamically basing on some history showing utilization of the video data buffer. However, in current implementations of Video Clients for PC they are fixed.

There is one important difference between the video data buffer implementations for Video Client source filter in ActiveMovie and for Video Client with PCMotion MPEG hardware decoder. Namely, the ActiveMovie architecture requires often small readjustments of video position, especially while pausing video streaming. It is not very efficient to send Seek message to Video Server each time the architecture wants to move few video frames back or forward, especially that it can be improved by redesigning the video data buffer. The Video Client for Windows 3.1 has the network inactivating threshold at 95% of buffer size. The Video Client in ActiveMovie has increased the buffer size but lowered the threshold to 60%. By this operation, the circular video data buffer has 40% of 'memory', i.e. the video data that was already read by decompressing and rendering filters is still being stored in the buffer. Now, when ActiveMovie architecture moves several frames back (for 1.2 Mbyte buffer it can move back up to 2.5 seconds), the Video Client is able to restore old state of video data buffer, and it does not need to perform the lengthy seek operation.

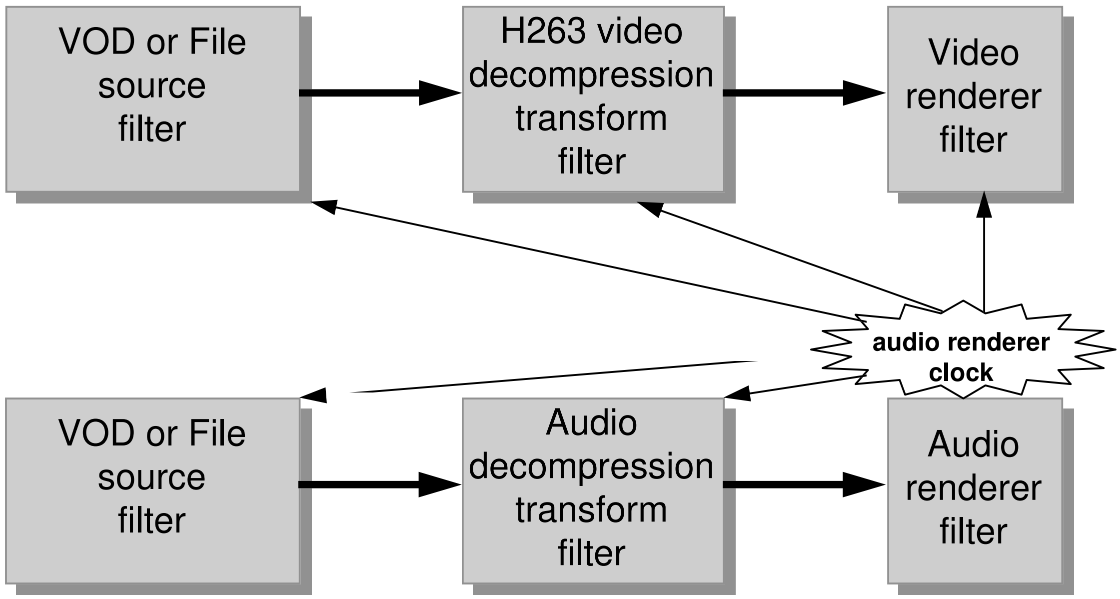
1.Video Client control flow

1.Working with Internet browsers

sdfsdfsdfsdfsdf

1.Designed Extensions to Video Client

sdfsdf



I.Conclusions

ActiveMovie is still a beta software. NPAC VOD project group has became a beta tester of ActiveMovie technology. The documentation is incomplete, especially for the Microsoft MPEG and AVI parser interfaces, which had to be communicated with in order to build multiformat Video Client. Some APIs are still under development. From the other hand, Microsoft has provided its ActiveMovie developers with newsgroup service, which the author found very helpful.

The new Video Client for PC has another advantage: a few Video Clients can work simultaneously on end user desktop (network limit of 4 MPEG streams, more clients if H263, CPU limit of 2 on Windows 95 with Pentium 75Mhz).

The debugging feedback from the work on new solution (in ActiveMovie) allowed to improve functionality and stability of the Video Client for Windows 3.1.

In the United States, the challenge of handling increasing power of new technologies is fueled by public awareness and demand for applications that take advantage of these technological advances. In Poland, the public awareness of new possible multimedia services is very limited.

In future: IP multicasting - new architecture, RTP protocol.

Video Client for Windows 3.1 is a high-end solution and thanks to the MPEG decoder hardware support provides better quality. The Video Client for Windows 3.1 is used in NPAC as a demonstration platform, while Video Client in ActiveMovie can become a non-expensive market solution.

I.Acknowledgments

Sincere appreciation is expressed to the following persons for their help and support:

- Czesław Jędrzejek (professor of EFP, thesis supervisor)

- Marek Podgórny (associate director of NPAC, thesis consultant, VOD project leader)
- Kemal Ispirli (member of VOD group, Video Server developer)
- Avneesh Pant (former member of VOD group, Video Server developer)
- Grzegorz Lewandowski (member of VOD group, ActiveMovie developer)
- all other helpful people of Northeast Parallel Architectures Center.

The work on NPAC Video-on-Demand system including developing of NPAC Video Client for Windows 3.1, Windows 95, and Windows NT described in this paper was founded by American Department of Defense grant, Rome Lab. Contract No. F30602-95-C-0273, PR No. C-5-2293/4

I.Glossary

API	Application Programming Interface
ATM	Asynchronous Transfer Mode
AVI	Audio Video Interleaved
CD-ROM	Compact Disk Read Only Memory
CGI	Common Gateway Interface
CIF	Common Intermediate Format
CPU	Central Processing Unit
DCT	Discrete Cosine Transform
DLL	dynamic link library
FTP	File Transfer Protocol
GOP	Group of Pictures
GUI	Graphics User Interface
H.263	video compression standard
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IGMP	Internet Group Management Protocol
IP	Internet Protocol
ISDN	Integrated Service Digital Network
ISO	the International Standards Organization
ITU	International Telecommunication Union
kbps	kilobit per second
LAN	Local Area Network
Mbps	Megabit per second
MCI	Media Control Interface
MIME	Multipurpose Internet Mail Extensions
MPEG	Moving Pictures Experts Group
NPAC	Northeast Parallel Architectures Center
NTSC	National Television System Committee
PC	Personal Computer
QCIF	Quarter CIF
QoS	Quality of Service
RIFF	Resource Interchange File Format
RSVP	Resource Reservation Protocol
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SDK	Software Development Kit
SGI	Silicon Graphics Inc.
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UI	user interface

URL Unique Resource Locator
VCR Video Cassette Recorder
VfW Video for Windows
VOD Video-on-Demand
WWW World Wide Web

I.Bibliography

1. "Microsoft ActiveMovie Software Development Kit - Developer's Guide", Beta Release - June 14, 1996, Microsoft Confidential
2. Don Gilbert, "ActiveMovie", Microsoft Interactive Developer, Summer 1996
3. "Video for Windows Programmers Guide", Windows 3.1 Software Development Kit
4. "The Challenges of Networking Video Applications" - Starlight Technical White Paper, *http://www.starlight.com/starlight/html/wp.html*
5. P. Shenoy, P. Goyal, and H.M. Vin, "Issues in Multimedia Server Design", ACM Computing Surveys, Vol. 27, No. 4, Pages 636-639, December 1995
6. D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications", Communications of the ACM, Vol. 34, No. 4, Pages 46-58, April
7. 6.G. Blakowski and R. Steinmetz, "A Media Synchronization Survey: Reference Model, Specification, and Case Studies", IEEE Journal on Selected Areas in Communications, Vol. 4, No. 1, Pages 5-35, January 1996
8. Ted Eilley, Fred Snijders and ChaoYing Ma, "Specification of the Components Systems Architectures", DIAMOND VOD project documentation, 30 June 1994, *http://goole.octacon.co.uk:80/proj/diamond/*
9. Ted Eilley, Phil Lloyd, David Walker, Irek Defee, Padraig Ryan and Kieran Cleary, "VoD Application User Interface Design", DIAMOND VOD project documentation, 31st August 1994, *http://goole.octacon.co.uk:80/proj/diamond/*
10. Mark Handley and Jon Crowcroft, "The World Wide Web - Beneath the Surf", UCL Press, 1994, http://www.cs.ucl.ac.uk/staff/jon/book/book.html
11. Bartłomiej Winnowicz, "Video Client for PC, Internship Report", January 1996, http://www.npac.syr.edu/projects/internship/index.html

II.Appendixes

Appendixes A1-A3 show consecutive stages of user interaction with NPAC Video-on-Demand system, with use of Netscape browser.
Appendix B1 shows the last stage of user interaction with the system - during video displaying, with use of Internet Explorer browser.