Military Technical College
Kobry El-Kobbah,
Cairo, Egypt

6th International Conference
on Electrical Engineering
ICEENG 2008

Design and implementation of a platform-independent IVOD system

By

E. Elshimy* H. Farouk** S. Ahmed*** I. Ismail****

Abstract:

Most of currently available IVOD systems open a new stream for every client. The main feature of the proposed system is Interactivity where the client takes the whole control on the streamed video. The system allows users to join already running streams, so, the user has the choice either to open a new stream or join one of the already running streams and that results in a better network utilization. The proposed system is platform-independent and it has been evaluated on two different platforms (windows – Linux). The proposed system adds strategies to detect and recover from failures and to prevent the waste of server resources by using the "Leasing" mechanism to grant the allocation of server resources for a certain period of time. The proposed system is able to manage at least 50 media streams (with audio and video content) in a 100 Mbps network.

Keywords:

Interactive video on demand, Java Media Framework

* Graduate Student, Dpt. of Information Technology, Cairo University, Cairo, Egypt
** A. Professor, Dpt. of Computer and Systems, Electronic Research Institute, Egypt
*** Professor, Dpt. of Information Technology, Cairo University, Cairo, Egypt
**** Professor, Dpt. of Information Technology, Cairo University, Cairo, Egypt

1. Introduction:

With the dramatic growth of Internet, lots of applications have been evolved. People feel that Interactive Television is the vision to their beliefs; they will be able to purchase products, view movies, play video games, browse Internet, and participate in video-conferences without leaving their houses. Of all the new things that people can do with television, video-on-demands is highly supported by Hollywood since it can lead to new markets and can bring them unpredictable profits [1].

People have been passive participants in receiving what TV service providers offer since television was introduced. Interactive Video On Demand (IVOD), unlike traditional television delivery, provides users with flexibility in choosing the kinds of information they like to receive. An IVOD system is capable of serving a large number of end users to concurrently access large number of repositories of stored data, often movies. IVOD is basically an extension of Video On Demand (VOD). In addition to the freedom of choosing movies, users can interact with movies and decide the viewing schedule [5]. In other words, IVOD system supports VCR-like functions, such as fast forward, rewind and pause. The enormous communication bandwidth and disk bandwidth required, and the Quality of Service (QoS) demanded necessitate a careful design of the system in order to maximize the number of concurrent users while minimizing the cost [6].

An IVOD system comprises of 3 major components: the "set-top box" (STB) at the client's site, the distribution network, and the server. There are many design issues to consider in building each of these components. IVOD proposes to provide subscribers who are connected through a set-top box (STB) with the possibility of ordering at any time the video of their choice and starting immediately to watch it on their television set.

An IVOD service requires real-time display of the video purchased by the client. A typical video stream consists of frames of pictures, sounds corresponding to those frames, and captioned text. The large quantity of information needed to be transmitted to the client continuously with minimal delay poses high performance requirements on the network. An IVOD network should be a high speed network with reasonable error rate as retransmission is unacceptable. Since video information is delay sensitive, the delay variation (jitter) should be kept to a minimum.

Section 2 gives a short overview of the streaming protocols; section 3 discusses related work, section 4 describes the architecture of the system on both the server and the client side, section 5 numerates the functions offered by the server and the client, section 6

explains the server and the client workflows and the interactions between the different components on each side, section 7 presents the results of testing the system with a sample media file and section 8 is the conclusion from this work.

2. Streaming Protocols:

The Internet today is recognized as the network that is fundamentally changing social, political, and economic structures, and in many ways obviating geographic boundaries. The Internet certainly provides such a national and global network infrastructure. TCP/IP has been well known as the core protocol suite in realizing the Internet. Protocols can be used in streaming services are TCP (Transmission Control Protocol), UDP (User Datagram Protocol), HTTP (Hypertext Transfer Protocol), and RTP (Real-time Transport Protocol). Java has emerged as one of the major programming languages for Internet applications in recent years. It has been used in many Internet applications ranging from database applications to multimedia applications. The main advantage of Java is that it is platform independent and it is simple to use. JMF (Java Media Framework) is a Java framework for incorporating real-time multimedia into Java applications and applets.

3. Related work:

There has been extensive research on VOD applications and their familiar features like interactivity, stream sharing, leasing mechanisms and platform independency. The majority of the proposed architectures tried to address some features explicitly and ignored the others. Zhouy and King provide architecture for a platform independent VOD system without giving the mechanisms for interactivity / joining an active stream or monitoring the existence of the server/client (leasing mechanism) [15]. Diwakar Gupta designed a focus Aware VoD without keeping the interactivity in mind [3]. Kostas. E. Psannis, Yutaka. Ishibashi and Marios Hadjinicolaou designed a novel method for supporting full interactive media stream and overlooked the stream sharing issues [10]. The proposed system tried to achieve an IVOD system architecture where the familiar features are considered core features from the ground-up.

4. System Architecture:

The proposed system is based on the client-server model which consists of three main parts:

A. Server. B. Client(s). C. Network.

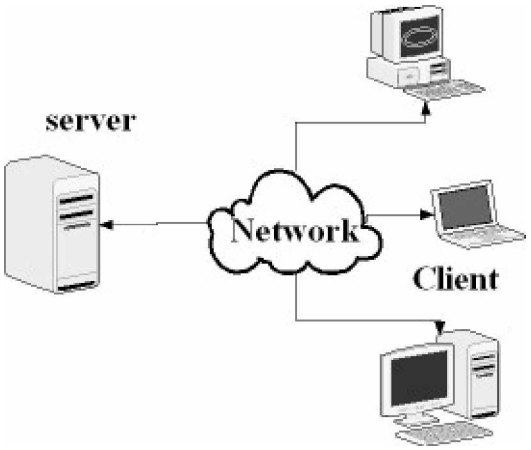

Figure (1): *System Main Component Block Diagram*

In this paper, the first two parts are demonstrated.

A. Server Side:

Figure (2) shows the block diagram of the server main components and their relationships

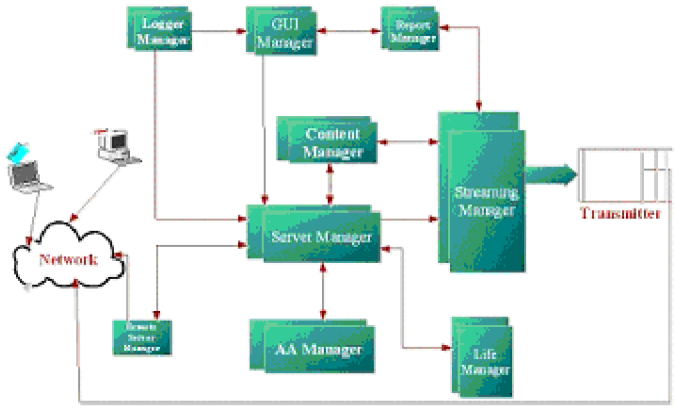

Figure (2): *Server Side Block Diagram*

The Server side consists of ten main components. The functions of each of these components are given below.

(1) Server Manager

a. The Server Manager plays the role of the coordinator between the different system components. It Instantiate and initialize the following system

components: GUI Manager, Logger Manager, Remote Server Manager, Content Manager, Authentication and Authorization Manager, Streaming Manager, Lifetime Manager and Report manager.

(2) Authentication / Authorization Manager
   a. Answer the Server Manger requests to authenticate and / or authorize a client.

(3) Content Manager
   a. Manage the available media for the Server Manager.

(4) Streaming / Resources Manager
   a. Optimize the resources usage by tracking used ports and find available ones.
   b. Maintain a data structure of the active clients.
   c. Maintain a list of active streams.
   d. Construct and initialize transmitter(s) for start actions.
   e. Find and control active transmitter(s) for all stream control actions (IVOD Events).
   f. Inform the Server Manager during the streaming state changes.

(5) Report Manager
   a. Generate reports by querying the Server Manager for the information required.

(6) Lifetime Manager
   a. Maintain the leases of active clients stored in the Server Manager.
   b. Inform the Server Manager if the client is no longer active.

(7) Logger Manager
   a. Maintain the system generated log records.

(8) GUI Manager
   a. Construct GUI interface for the server part.
   b. Inform the Server Manager for the Administrator actions.

(9) Transmitter(s)
   a. Control the active stream(s).

(10) Remote Server Manager
   a. The Server interface to the network.

B. Client Side:

Figure (3) illustrates the block diagram of the client main components and their relationships

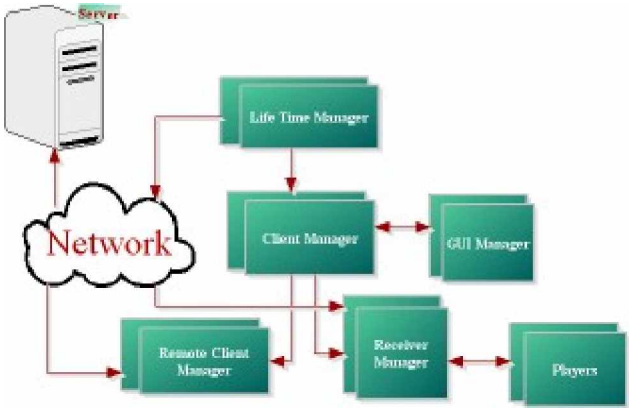

Figure (3): *Client Side Block Diagram*

The Client side consists of six main components, with the following functions:

(1) Client Manager
   a. The Client Manager plays the role of the coordinator on the client side. It Instantiate and initialize the following system components: GUI Manager, Lifetime Manager, Remote Client Manager and Receiver.

(2) GUI Manager
   a. Inform the Client Manager about the user generated actions.

(3) Lifetime Manager
   a. Monitor the status of the server.
   b. Inform the Client Manager if the server status changed.

(4) Receiver Manager
   a. Accept streamed content from the server.
   b. Construct and initialize the corresponding player(s).

(5) Player
   a. Play streamed content from the server.

(6) Remote Client Manager
   a. Manage the communication with the server.

5. System Functionality:

A. Server:

Figure (4) is a snapshot of a running VOD server.

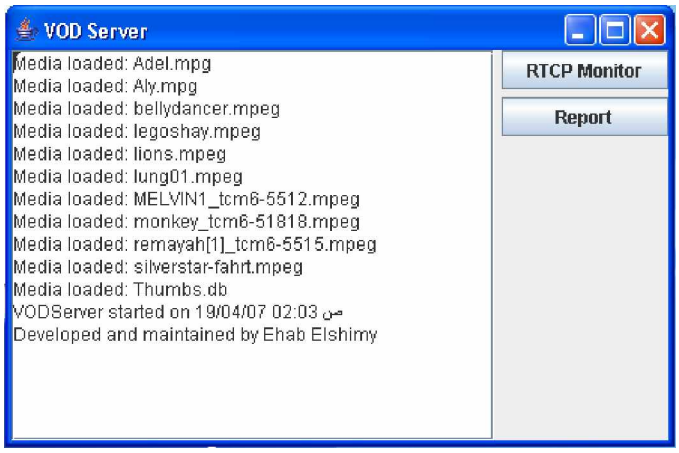

Figure (4): *Server GUI Interface*

The snapshot shows that the server has a central display area to dump significant system events; the server also provides the possibility to generate reports about active clients and streams.

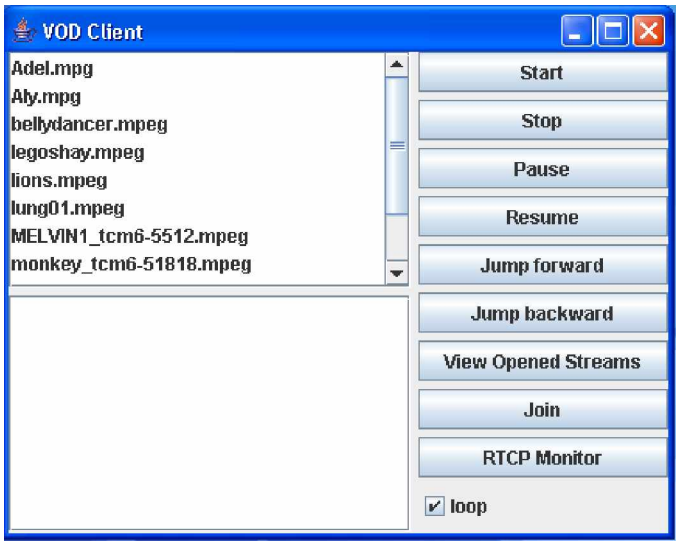

Figure (5): *Client GUI Interface*

B. Client:

Figure (5) shows a view of a running VOD client, the view shows that the client has three main parts.

The first parts is a list of available media, the second part is a list of active streams and the last part is the control part which provide the facility to control available media and active streams, the available controls are:

(1) Start
Start a new stream from one of the available media.
(2) Stop
Stop the selected media if it is already started.
(3) Pause
Freeze the selected media if it is already started.
(4) Resume
Resume the selected media if it is already frozen.
(5) Jump forward
Jump from the current position of the stream to another point in the forward direction.
(6) Jump backward
Jump from the current position of the stream to another point in the backward direction.
(7) View opened streams
Query the server of the a list of running streams
(8) Joined
Join the selected running stream.

6. Implementation:

In this section we will describe the server and clients' lifecycle.

A. The Server Lifecycle:

The Server Lifecycle begins by calling the server main method, which results in the following steps:

(1) Instantiating the Server Manager.
(2) The Server Manager constructs the Logger Manager.
(3) The Server Manager constructs the Content Manager and wait until it is ready (read the available media).
(4) The Server Manager constructs the GUI Manager.
(5) The Server Manager constructs the Remote Server Manager and initializes it with a call back object which will be called with the corresponding remote user action.

(6) The remaining components are instantiated and initialized on demand, the Authorization and Authentication manager will be instantiated with the first remote user call, the Streaming Manager will be instantiated with the first remote user call to stream content, the Lifetime Manager will be instantiated with the first need to maintain active user status, the Report Manager will be instantiated with every request to generate a report and dismissed after that.
(7) The Server Manager will be notified from different system components. From the Remote Server Manager with every user request, the request will be authenticated and authorized through the Authentication and the Authorization Manager and based on the result, it could be refused or delegated to the Streaming manager and asks the Lifetime Manager to monitor the connection. From the GUI Manager to handle an administrator action. From the Lifetime Manager to interrupt active streaming content if the client is no longer available. From the Streaming Manager to stop monitoring the connection if the streaming is already ended.

B. The Client Lifecycle:

Starting the client will result in the following steps:

(1) Instantiating the Client Manager.
(2) The Client Manager constructs the GUI Manager.
(3) The Client Manager constructs the Remote Client Manager and tries to contact the server using a known address and a hardcoded username and password.
(4) If the connection succeed, the Client Manager will query the server for the available media and populate the GUI Manager with the set of available media to display it in the available media list, during this initial contact with the server, the server will give the client a ticket which should be used in all successive interactions.
(5) If the connection fails, the Client Manager will inform the GUI Manager to notify the user.
(6) Triggered actions by the user, will be given to the Client Manager which in turn use the Remote Client Manager to communicate with the server, if the result is a streaming content, the Client Manager will inform the Receiver Manager to manage the streaming content from the server and instruct the lifetime Manager to monitor the state of the server.

The system after this initialization phase goes into idle state waiting for user actions. Starting from this point the user has the possibility to start one of the available media (open a new stream for the selected media, and assign himself as the owner of it) or query the server for the list of active streams and join one of them (join an active stream

at the current position, and assign himself as a joiner, passive monitoring, which has no control on the stream).

C. Client/Server Communication:

Start Action: When the user triggers a start action on the client, the GUI Manager will inform the Client Manager about the action, which in turn, instructs the Remote Client Manager to connect to the server using the ticket and the media name, when the Remote Server Manager gets the start call, it'll delegate it to the Server Manager which check the ticket against the list of active clients, if it isn't found, it throws an exception back to the Remote Server Manager which terminates the connection, if it is found, it will query the Content Manager for the media and give it to the Streaming Manager, the Streaming Manager looks 1st for a free multicast group and construct a transmitter to begin streaming the selected media on the given multicast group, then the Streaming Manager will return back to the Server Manager the multicast-group and another ticket to identify the active stream, the Server Manager will in turn deliver it back to the Remote Server Manager which give the Remote Client Manager this information over the network, the Remote Client Manager pass the information to the Client Manager which stores the stream ticket for successive processing and constructs a receiver to listen for the RTP-packets on the given multi-cast group, the receiver in response to receiving the initial packets from the server will construct a player to play the incoming stream.

View Opened Streams Action: The action arrives at the Server Manager in the same way like the start action but with the client ticket as the only parameter, the Server Manager will query the Streaming Manager for the list of active streams, and send it back to the Client Manager in the same way like the start action. The Client Manager will give the list to the GUI Manager which in turn display it in the list of active streams.

Join Action: The action arrives at the Server Manager with two parameters, the client ticket and the stream id, the Server Manager will ask the Streaming Manager if the stream is still active or not, if not, the Server Manager will throw an exception to the client, if it is still active, the Streaming Manager will add the client to the list of active listeners on this stream and report the multicast-group to the Server Manager, which in turn delegates it back to the Client Manager(throw the Remote Server Manager and the Remote Client Manager) over the network, the Client Manager will construct now a receiver which constructs a player as a response to the initial packets received from the server.

Pause Action: The action arrives at the Server Manager with the client ticket and the stream id as the parameters, the Server Manager will ask the Streaming Manager to pause the stream with the given stream id, the Streaming Manager will 1st check if the client with the given client ticket is the stream owner or not, if not, it will report an exception back to the Server Manager, otherwise the Streaming Manager will pause the stream, by instructing the transmitter to stop streaming and maintain the current stream position.

Jump Forward Action: Once the user triggers this action, the GUI Manager displays a popup dialog to get the requested jump interval from the user in seconds, the entered value will be transferred to the server in addition to the client ticket and the stream id in the same way like the pause action, the Streaming Manager will instruct the transmitter to change the current position in the stream with the given jump interval.

Jump Backward Action: this action is similar to the jump forward action but to jump in the backward position.

Stop Action: The action arrives at the Server Manager with the client ticket and the stream id, the Server Manager will delegate it to the Streaming Manager which will remove the client from the active listeners and check if there are remaining listeners on that stream, if any, the Streaming Manager will leave the transmitter running, if the list of active listeners is empty, the Streaming Manager will stop and close the transmitter of this stream. Now it is the responsibility of the client to close the receiver and close the player.

7. Results:

The standard method of transporting media packets through IP-based networks requires the addition of three headers. These are IP, UDP and RTP. An IPv4 header is 20 bytes, a UDP header is 8 bytes and an RTP header is 12 bytes. A total of (20 + 8 + 12) 40 bytes are therefore sent each time a packet containing media is transmitted. The additional bandwidth occupied by this header information is determined by the number of packets that are sent each second.

We used the Maximum transmission unit (MTU) for Ethernet, 1500 bytes, which results in RTP Payload size of 1460 bytes.

Our sample media file consisted of an MPEG-1 video stream of 1150 kbps and 352 x 240 (29.97 frame per second) resolution and an MPEG-1 Layer 2 encoded audio stream

of 224 kbps. Using 1460 bytes RTP payload size, the total number of video and audio packets per second are 98 and 19 packets respectively.

Each packet carries an IP/UDP/RTP header overhead of 40 bytes, meaning that 3920 (98 x 40) header bytes [video stream] and 760 (19 x 40) header bytes [audio stream] are sent each second. That means the header information will add 31.36 kbps to the bandwidth requirement of the video stream and 6.08 kbps to the bandwidth requirement of the audio stream. For our sample media file, the total bandwidth required to transmit the video and audio steams are 1181.36 kbps (1150 + 31.36) and 230.08 kbps (224 + 6.08) respectively.

The RFC specifications recommend that the fraction of the session bandwidth allocated to RTCP be fixed at five percent of RTP traffic, then the total bandwidth required for the video and audio steams are 1240.43 kbps (1181.36 + 59.07) and 241.58 kbps (230.08 + 11.5) respectively.

Ignoring the bandwidth allocated for the session control traffic, the total bandwidth occupied by our sample media file is 1482 kbps (1240.43 + 241.58) approximately.

Table (1): *Network Capacity*

	Audio Stream	**Video Stream**	**Audio Stream + Video Stream**
10 Mbps (10,000 kbps)	41	8	6
100 Mbps (100,000 kbps)	413	80	67

For classical traffic on the network we will assume an average utilization of 75% for bandwidth required for our system, so the average number of channels for video and audio streaming would be at least 50 channels.

8. Conclusion:

The paper presented a new approach for IVOD system with platform-independency and resources optimization as the system core requirements, basically the system could run on all platforms which have java runtime environment; the system was proved to be a reliable system under different network conditions, because it continued to run under heavy network loads with a large number of active clients, at least 50 media streams (with audio and video content) in a 100 Mbps network. The system enhances the

network utilization by enabling users to join already running streams and avoid opening new stream for each client. Resource optimization was achieved through stream sharing techniques and by using the leasing mechanism that allows resources to be managed without complex garbage collection mechanisms.

References:

[1] A. Albiol, L. Torres and E. Delp. "*Video Preprocessing for Audiovisual Indexing*" Fifth IEEE Southwest Symposium on Image Analysis and Interpretation. p.57, April 2002.

[2] B. Qazzaz, R.Suppi, F.Cores, A.Ripoll, P.Hernandez and E. Luqe. "*Providing Interactive Video on Demand Services in Distributed Architecture*" 29th Euromicro Conference (EUROMICRO'03), p. 215, September 2003.

[3] Diwakar, Gupta. "*Design, Implementation And Validation of a Focus Aware, Resource Management System for Distributed Multimedia Applications*". University of Delhi, 2002.

[4] H. Farouk "*Video Compression Improvements based on Camera Motion*" Ph. D. thesis, September 2001.

[5] J. Wu, J. Chen and H. Chao. "*Load Shin Protocol Design in ATM-Based VOD Systems*" 13th International Conference on Information Networking (ICION)'98), p. 222, January 1998.

[6] K. E. Psannis and M. G. Hadijnicolaou,"*A New Methodology For Implementing Interactive Operations Of Mpeg-2 Compressed Video*" Second International Workshop on Digital and Computational Video (DCV'01), p. 21 ,February 2001.

[7] L. Pinho, C. Amorin and E. Ishikawa. "*GloVE:A Distributed Environment for Low Cost Scalable VoD Systems*" 14th Symposium on Computer Architecture and High Performance Computing (SCAB-PAD'02). P.117, October 2002.

[8] N. Kamiyama. "*Renegotiated CBR Transmission in Interactive Video-on-Demand System*" 1997 International Conference on Multimedia Computing and Systems (ICMCS'97), p. 12, June 1997.

[9] O. Hadar and M. Segal. "*Models and Algorithms for Bandwidth Allocation of CBR Video Streams in a VoD System*" International Conference on Information Technology: Coding and Computing (ITCC'01), p. 148, April 2001.

[10] Psannis, Kostas E., Yutaka Ishibashi, and Marios Hadjinicolaou. "*A Novel Method for Supporting Full Interactive Media*" GVIP (2005).

[11] R. Schroeter, J.Hunter and D. Kosovic. "*FilmEd – Collaborative Video Indexing, Annotation and Discussion Tools Over Broadband Networks*" 10th International Multimedia Modelling Conference, p. 346, January 2004.

[12] S. Yamazaki, N. Nakamura, Y. Miyadera and S. Yokoyama, "*A Development of Classroom Design Simulator for Interactive video Teleconference*" International Conference on Computers in Education (ICCE'02), p.865, December 2002.

[13] S. K. Kweon and Kang G. Shin. "*Transmission of Aggrregate VoD Streams Using Playback Rate*" Seventh Real-Time Technology and Applications Symposium (RTAS'01). p. 205 May 2001.

[14] Son Vuong. "*What is a Multimedia System*". Course Notes on Communication Protocols, Computer Science Department of University of British Columbia, Vancouver, 1996. V. Li, W. Liao, X. Qiu, and E. Wong, "Performance Model of Interactive Video-on-Demand Systems," IEEE Journal on Selected Area in Communications, vol. 14, issue 6, August 96.

[15] S. Zhou and P. King, "*A Simple Platform Independent Video on Demand Application*", International Conference on Information Technologies and Applications 2002 (ICITA'2002), Bathurst, Australia, Nov 2002.

[16] W. Liao and V. Li. "*The Split and Merge (SAM) Protocol for Interactive Video-on-Demand Systems*" INFOCOM'97. Sixteenth Annual Joint Conference of IEEE Computer Communications Societies. Driving the Information Revolution, p.1349, April 1997.

[17] W. Knightly, D. E. Wrege, J. Liebeherr, H. Zhang. "*Fundamental Limits and Tradeoffs of Providing Deterministic Guarantees to VBR Video Traffic*". SIGMETRICS' 95, 1995.

[18] Y. Tanaka, T. Itamiya, T. Hagino and H. Chiyakura,"*HTTP – Proxy – Assisted Automatic Video Indexing for E-Learning*" 2004 Symposium on Applications and the Internet-Workshops (SAINT 2004 Workshops), p. 502, January 2004.