ACCESS EMULATION AND BUFFERING TECHNIQUES FOR
STREAMING OF NON-STREAM FORMAT VIDEO FILES

Sung-I Kang, Junehwa Song, Sang-Kyu Bae and Heung-Kyu Lee
Department of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology (KAIST)
373-1 Kusong-dong Yusong-gu, Taejon 305-701, Republic of Korea

Abstract

Videos recorded in non-stream format are widely used in
the Internet. However, they are not directly utilized in video streaming. For streaming without download, a time consuming format conversion is usually required. To make such videos to be directly utilized in video-on-demand, we have developed a new service model and implementation techniques. In this paper, we describe the techniques in detail along with a prototype movie-on-demand system. With a low-end Linux server, the system successfully supports over 30 clients connected at the same time.

1. Introduction

Nowadays, video streaming [1,2] is one of popular buzzwords in the Internet world. However, on the other hand, the quality of services is not very satisfactory yet. Many video-on-demand systems serving videos on-line assume degradation of video quality when unexpected packet loss occurs by the unstable Internet. However, this packet loss becomes a direct reason for the unsatisfactory services, and the quality of the services can not meet the expectation of the paid clients. Thus, commercializing on-line video services becomes difficult. We argue that lossless and continuous playback as well as interactivity are very important for commercial movie-on-demand services.

Based on this consideration, we have developed a new service model for interactive movie-on-demand systems. In the model, video data saved in a remote server are reliably delivered to clients. In addition, they are handled interactively just as saved in the local system. This model is very different from that of current video streaming systems which focus on the recovery of once lost data. In order to realize the service model, we have developed two base implementation techniques. The first is an access emulation technique which makes remote files to look like local ones. The other is a buffering technique for continuous playback of high-quality movies. As a result, even video files of a non-stream format can be used in streaming.

To show the effectiveness of the techniques, a prototype interactive movie-on-demand system has been built. The system directly serves non-stream format videos files. The system has successfully supported over 30 clients connected via various access networks such as Ethernet, ADSL (Asymmetric Digital Subscriber Line), and Cable Modem [4].

This paper is tightly related with formatting of multimedia data [3]. We define some terms related with formatting. Firstly, compression format means a video coding technology, such as MPEG-1/2, M-JPEG, and DV, to reduce video size effectively. That is, it is interchangeable with the term called codec (coder and decoder). Secondly, file format means a data structure of a file to record multimedia data, which are usually compressed by a codec. Finally, media format means a data layout of physical recording media, such as Video-CD and DVD-Video [10]. Transmission channel [13] for streaming can also be considered a kind of delivery media.

![Diagram](image_url)

Figure 1. Formatting of multimedia data

Fig.1 shows a hierarchy (relationship) of multimedia formats with corresponding instances. The downward arrowed flow on the left side is the case of safe delivery...
via recording media in the current video rental services. The right side is the case of streaming, i.e., video transmission over network. If the formatting flow is changed from left to right as noted with the dotted arrow, it may need one time of format conversion. However, if our techniques are used, such format conversions will not be needed any longer. This paper focuses on this dotted flow.

The rest of the paper is laid out as follows: Section 2 surveys multimedia file formats for video delivery via various media. Two types of file formats for multimedia data are compared and analyzed. Section 3 describes implementation techniques to directly serve non-stream format video files without format conversion or data loss. Section 4 describes a prototype system implemented with the techniques and experimental results performed with the system. Finally, in Section 5, we conclude with future works.

2. Multimedia File Format and Transmission

There are many multimedia file formats storing audio or video data into a file. Among them, we consider only file formats that can include audio and video together because we target the movie-on-demand service. In this section, we analyze and compare two types of multimedia file formats, and describe reliable network transmission for non-loss delivery.

2.1. Store and Streaming Formats

Multimedia file formats can be largely categorized into two types: one for local playback from safely delivered media and one for remote playback via network transmission. We call the former store(non-stream) format and the latter streaming format.

The store format is usually an old one developed before streaming technology becomes popular. It is mainly for interactive playback of videos stored on reliable local media such as CD, DVD, and computer disk. The store format videos generally assume fast random access and no data loss during playback. Because the video files of this format are not directly served via streaming, they should be downloaded into a local system before playback. Downloading of video files takes a long time due to their sizes and may cause copyright problems. The main reasons that downloading are necessary is that store formats have non-sequential layouts and that the data are sometimes accessed randomly. In this format, AVI(Audio/Video Interleaved), DAT of MPEG-1 Video-CD, VOB of MPEG-2 DVD-Video are included.

Among those three store file formats, AVI does not depend on a specific compression format and its related specifications are opened to public. Thus, it is still widely used in the PC world as a file format for testing new compression formats, i.e., codecs. Although MPEG is currently considered as the main standard in compression format, various new codecs, enhanced from MPEG or based on new algorithms such as wavelet and fractal, are continuously developed. Therefore, to use codec-independent file formats is very important in designing video systems. Thus, we select AVI as the main file format to show the validity of the proposed techniques. More detailed analysis about AVI is given in the next section.

The streaming format, as one for transmitting multimedia data over network, resolves the streaming problem of the store format. This format considers recovery from data loss caused by network congestion or channel errors. In addition, structurally, it has a packet-based layout considering transport protocols. Although the streaming format has more functions than the store format, it has additional structural overheads for those additional functions. Various streaming file formats are currently developed and used by streaming solution providers with their proprietary streaming technologies. For example, ASF(Advanced Streaming Format)[7,8] of Microsoft’s Windows Media, RM of Real Networks’ Real Media, and MOV of Apple’s QuickTime are included in this format.

2.2. Analysis of AVI Store Format

As mentioned, AVI is widely used due to codec independence. However, it is difficult to directly use in video streaming because of its non-sequential layout, as shown in Fig.2.

![Figure 2. Standard AVI structure](image-url)

The standard AVI structure [1] consists of three parts: header, movie data, and index. The header includes descriptive information for movie data in the second part, such as frame rate, video screen size, audio bit rate, and related AVI codecs. Codecs are identified with a four character code called FOURCC. The movie data part includes audio and video data blocks in an interleaved fashion. Finally, the index part is an optional one, but almost all AVI videos have it because by using the index, fast replay is possible when users move playback positions. It consists of a set of entries one for each frame, as shown in Fig.3.
AVI is a kind of RIFF (Rich Interchange File Format) format [3], which consists of data blocks called chunks. It has a unique RIFF code string, ‘RIFF/AVI’, which is used to identify if the file is formatted in AVI or not. DAT of Video-CD is also a RIFF format with code string ‘RIFF/CDXA’. However, practically the file formats are directly identified with their file extensions, such .avi, .dat, and .wav.

2.3. Format Comparison

In video-on-demand systems using a streaming format, AVI videos should be first converted to the streaming format. However, this conversion usually takes a long time and needs a huge storage. In addition, interactivity may be decreased due to this conversion. Because our service model assumes no data loss during transmission and our techniques emulate remote files into local ones while sustaining interactivity, it is not necessary to use a stream format with additional overheads. To see the overheads, we compare the size of videos formatted in AVI and ASF, which are two formats widely used in the PC world.

Table 1. Size comparison of AVI and ASF files

<table>
<thead>
<tr>
<th>Movie</th>
<th>AVI (KB)</th>
<th>ASF (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>StarWars EP1(trailer)</td>
<td>18,068</td>
<td>18,921</td>
</tr>
<tr>
<td>Mission To Mars(trailer)</td>
<td>16,892</td>
<td>18,037</td>
</tr>
<tr>
<td>Matrix (trailer)</td>
<td>17,910</td>
<td>18,545</td>
</tr>
<tr>
<td>Final Fantasy (trailer)</td>
<td>4,804</td>
<td>5,706</td>
</tr>
<tr>
<td>ConAir (full)</td>
<td>685,588</td>
<td>687,397</td>
</tr>
</tbody>
</table>

Table 1 shows the sizes of AVI and ASF files for the same movie. As noted earlier, the ASF has additional information not appearing in the AVI. So, the ASF-formatted video is larger than the AVI-formatted one by over 1%, or sometimes up to 20%. If AVI videos can be reliably sent over network, bandwidth requirement is reduced. In addition, the requirement of the format conversion from the AVI to the ASF is removed.

In summary, the differences between store and streaming formats are as follows. As shown in Fig.4, the store format keeps the descriptive information for movie data, such as a header and an index, at the separate parts of a file. The movie data of a stream format including the descriptive information are divided into packets. As a result, for the same video data using the same codec, the stream format is larger than the store format because it has additional information, such as packet headers and timestamps for resynchronization, and padding data for rate control.

2.4. Reliable Transmission

For reliable transport in the TCP/IP based Internet, TCP (Transmission Control Protocol) can be used. In addition, there are protocols obtaining reliability by adding ARQ (Automatic Repeat reQuest) or FEC (Forward Error Correction) mechanisms to unreliable UDP (User Datagram Protocol) [11,12,13]. FEC without retransmission is proper to real-time video streaming. However, it does not always guarantee reliability. Thus, to increase the level of reliability, it is used combined with ARQ.

Although TCP is slower than UDP-based protocols due to reliable sequenced delivery and window-based slow start mechanism, it has advantages in that it is simple to use and quite efficient with kernel-level implementation. That is, applications do not need to consider miscellaneous functions for reliability, such as sequenced delivery, retransmission, and flow control. And also, with a bi-directional connection, control and data transmissions can be easily integrated into a connection. Even when UDP-based protocols are used in streaming, a separate TCP-based connection is used for stream control.

Because a client sees a remote video as a local file, it itself controls the file access, i.e., data flow. In other words, unlike other video streaming systems using sender-oriented transmission, our movie-on-demand system is based on receiver-oriented transmission. Thus, TCP is used as a reliable transport protocol of our prototype system.
3. Access Emulation and Buffering

In this section, we describe in detail two implementation techniques: access emulation and buffering used to provide reliable streaming of store format video files.

3.1. Access Emulation

The access emulation is a technique for streaming store format videos by showing remote files as local files to clients. This is somewhat similar to the network-based file system in that it emulates data accesses. However, it is different in that only internal data of specific files is shown to clients. Therefore, remote files themselves could not be accessed via normal file system access.

To show how the access emulation operates, we explain with three file access methods for video playback as shown in Fig.5. First, consider the local file access of Fig.5(a). Because high-speed random access is possible to the local files, high-quality video playback and interactivity can be provided. Of course, this is not proper for on-line service of copyrighted full movies except freely downloadable movie trailers.

![Diagram of file access methods]

See Fig.5(b). From the technical point of view, a simple method that can share videos among clients without download of files is to utilize currently available network-based file systems, such as CIFS(Common Interchange File System), and NFS(Network File System). In this case, it is not necessary to develop any software to provide video-on-demand functions. Clients connect and share video files saved in a file server. Although this method is the simplest to share videos, from the service point of view, it is not proper to use in an Internet-based on-line movie service. It may be used in an intranet-based systems if related legal rights to all contents have been acquired. In addition, the video may not successfully be played back when large data blocks are suddenly accessed because network-based file systems consider synchronous processing of accesses for all types of files. This phenomenon can be easily observed even in LAN systems connected via a 100Mbits switching hub. Also, it is difficult to serve many clients simultaneously because network-based file systems designed for general file service are not optimized for video services.

Finally, a stream buffer access method is possible as shown in Fig.5(c). In this case, local file accesses are processed in a local buffer and the buffer keeps the data to be displayed ahead of their playback time. This method solves both the technical and the service problems described above at the same time.

![Diagram of processing methods of file access]

Fig.6 shows the operational differences between the file access methods and the stream buffer access method. The stream buffer method can smoothly process sudden access of large blocks in the fast local buffer by predicting such accesses. In this case, the problem is how to create a virtual file for access emulation.

![Diagram of DirectShow filter and file access emulation]

Windows includes a well-defined API(Application Programming Interface) for multimedia data processing, called DirectShow [7]. We have used the API to implement the access emulation technique in Windows. As shown in Fig.7, the asynchronous file source filter of DirectShow is first changed into a virtual file source filter. The virtual file information to show remote files like local
files is obtained from the server at the connection time.
For other multimedia processing functions such as parsing
and rendering, we use those in DirectShow.

3.2. Buffering

Due to differences in file formats, playback of videos
in different file formats show different access sequences
in accessing data blocks within video files. Therefore, the
access pattern should be analyzed in advance for a
buffering to effectively support the access emulation.

As described Section 2, almost all AVI videos include
an index for fast restart after moving playback position.
However, this index makes AVI videos difficult to do
streaming because it is needed before the startup but lays
at the end of files. According to this access pattern, the
index is transmitted right after the header. The index is
used for normal playbacks as well as for replay upon
playback position movement. To buffer AVI video
effectively, two types of buffers are used. One is a
circular stream buffer for short time buffering of video
data. The other is a static buffer for the index.

![Figure 8. Access pattern and buffering](image)

Fig.8 shows an example of the access pattern of AVI
videos in a local file and its buffering sequence in
streaming. Because audio and video blocks are
interleaved in AVI, accesses can be locally done in a
non-sequential fashion. Thus, the stream buffer is moved
slower than the current access point.

4. Implementation

Using the two techniques described in this paper, we
have developed an interactive movie-on-demand system.
The system directly serves video files of store formats
without format conversion into a streaming format. The
system is designed to support a community concept, as
shown in Fig.9. That is, clients can connect any server
within the same community.

![Figure 9. System structure](image)

4.1. Video Client

The video client provides the playback of store format
videos. It has been implemented as a Windows program
using DirectShow API. Fig.10 shows three screen
snapshots of the remote player. As shown in Fig.10(a),
the player provides the basic functions for playback control
such as play, pause and stop, as well as the functions for
screen control such as screen resizing and full screen, as
shown in Fig.10(b). If the computing resources are
sufficient in the client machine, multiple players can run
at the same time on a single computer, as shown in
Fig.10(c).

![Figure 10. Video client screenshots](image)
In addition to AVI videos, the remote player can playback other file formats supported by DirectShow. For example, DAT video of Video-CD and MPEG-I audio/video system streams. Different from the AVI with a non-sequential layout, the streaming of these MPEG-based video files is simpler because they have a sequential layout.

4.2. Video Server
The video server is implemented as a multi-threaded Linux program using Pthreads API [5]. It supports multiple video clients and is controlled by a proprietary remote control client.

With a server running on a PC with a 400Mhz Pentium II processor with 128MB memory, over 30 clients connected via LAN, ADSL, and Cable Modem [4], successfully played back their requested high-density videos of about 640x270 screen size. We consider the reason which a low-end PC can support so many clients at the same time is that the server has just a simple passive data transmission function. That is, because transmission control is totally done by clients, the server logic becomes very simple. This is a different design approach from that of current real-time video streaming systems with complex scheduling and rate control logics [14].

5. Conclusion
In this paper, we have presented two implementation techniques for realization of our movie-on-demand service model. The techniques focus on making remote video files to look like local ones and guaranteeing non-loss data transmission. They do not depend on a specific compression or file format. Therefore, movie-on-demand systems using the techniques can utilize video files of various existing store formats in streaming. In addition, it can easily embrace new codecs being continuously developed. We have also shown that the store format is more proper to our service model than the streaming format. That is, the store format provides superior interactivity with index and does not need format conversion after editing. Also, it does not have additional overheads, such as the packetization, which is necessary in streaming formats.

Finally, the presented techniques utilize a video server passively. They do not need a proprietary streaming server. Therefore, they can be used in progressive streaming without a specific streaming server. That is, in the case that contents distribution does not incur any legal problems, e.g., a movie trailer service, general web servers or ftp servers can be utilized in streaming service of store (non-stream) format video files. In the future, we plan to support progressive streaming and to include other fast reliable transport protocols.

Acknowledgement
This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc).

References


Biography

Sung-Ill Kang received the BS and MS degree in computer science from Pusan National University, Korea, in 1986 and 1988, respectively, and is currently a PhD candidate in computer science, Advanced Institute of Science and Technology (KAIST), Korea. His research interests are multimedia systems, operating system, and real-time computing.

Junehwa Song joined KAIST as a professor in 2001. His research interests include Internet computing and distributed multimedia systems.

Sang-Kyu Bae received the BS degree in computer science in KAIST, Korea in 1999. He is currently a BS candidate in KAIST. His research interests are video watermarking, video compression, multimedia systems, and operating system.

Heung-Kyu Lee received the BS degree in electronics engineering from Seoul National University, Korea in 1978 and the MS and PhD degree in computer science from KAIST, in 1981 and 1984, respectively. Since 1990, he joined in KAIST as a professor. He is a member of IEEE and ACM. His major interests are digital watermarking, fault-tolerant and real-time computing, and multimedia systems.