A Novel Decoder-downloadable System for Content-oriented Coding

Naoto SHIMIZU[†], Toshinori MIYAZAWA[‡], Wataru KAMEYAMA[†], Hiroshi WATANABE[†], Hideyoshi TOMINAGA[†,‡]

†Waseda Univ.
Graduate School of GITS
Nishiwaseda 1-3-10, Shinjuku-ku,Tokyo,Japan

‡Waseda Univ.
Dept. of Elec., Info. and Comm. Eng.
Okubo 3-4-1, Shinjuku-ku, Tokyo, Japan

***Abstract*—**

In this paper, a new system architecture called decoder-downloadable system is described. The purpose of this system is to provide a uniform platform for the multimedia world based on image compressions using characteristics of the contents. This system enables dynamic decoder downloading by negotiation between servers and clients for seamless and minimum delay playback. This negotiation scheme is implemented by Java and CORBA (Common Object Request Broker). Thus, servers and clients in this system are independent from OSes and hardware specifications. This system is suitable for TV broadcasting as well as Internet streaming.

We improve a decoder architecture in order to make the system more flexible. A decoder has some functions such as bitstream parsing, image compression algorithms (DCT, Wavelet, Motion estimation and so on). However, conventional decoders are monolithic software. Thus, it is impossible to share parts of decoders. The proposed method enables a module-based decoder and sharing some modules among some decoders. Consequently, the flexible decoder can make downloading time shorter by avoidance to download redundant parts of decoders. It also achieves scalability enabling this system to be used in some multimedia applications, such as a multimedia content search system, as well as a multimedia player.

I. Introduction

Discrete Cosine Transform (DCT) based coding, such as MPEG and JPEG, has been widely used for image compression. However, DCT-based approach may not be suitable for all kinds of images. Some types of images may be encoded efficiently by a method taking the characteristics of the image content into account. We call it "Content-oriented coding scheme". As its example, we have been studying animation image coding [1]. From this study, an open system architecture that we can distribute our animation image decoder on Internet Streaming and TV broadcasting is desired. However, no open system fulfilling the requirement has been reported. Thus, we implement the system called "decoder-downloadable system". In this paper, we present the system architecture.

In multimedia world based on Content-oriented coding scheme, it is anticipated that the number of image compression algorithm increases. Conventional decoders are developed as monolithic software. However, there are parts that decoders can share. Thus, their redundancies are eliminated by realizing module-based decoders, which are composed some modules and share them among decoders. We call these module-based decoders "flexible decoder". Designed decoders in this system as flexible decoder provide several advantages.

These proposal concept model is shown in Fig.1. The proposed system architecture provides a new way for all multimedia applications to share a variety of modules and functionality in a platform-, vendor-, and location-neutral environment.

II. Decoder-downloadable system

A. Problem of Plug-in system

Multimedia systems on the Internet, such as Microsoft Media Player, RealPlayer, provide functions to deal with new types of media by installing the appropriate software when it is required. Such optionally downloaded software is called Plug-in. However, Plug-in system has problems as follows.

1) Time delay

   The time delay happens when a new content starts and a client does not have a suitable decoder, because it takes

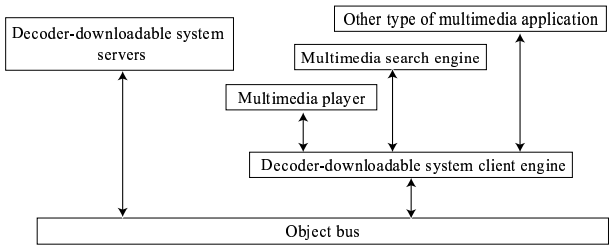

Fig. 1. Proposed concept model

additional time to download it. This interrupts seamless playback of contents and causes a serious problem especially in case of TV broadcasting.

2) Installation problem

Plug-in system requires users a little knowledge about image coding. As TV broadcasting has been universal service, and Internet streaming has been same situation, too. The system in the universal service should not expect clients about service installation knowledge. This point is very important for applying Content-oriented coding scheme, because it will require more knowledge about image coding than the conventional scheme.

3) Closed specification

The specification of Plug-in and the way to transfer Plug-in depend on each multimedia software. There is no compatibility among them. Therefore, developers of decoders are needed to create ones for each multimedia system. Each developer of multimedia system is also urged to create the architecture for downloading decoders.

B. Proposed system architecture

In this section, we propose the system architecture that enables to download decoders dynamically and to play back without delay.

The proposed system architecture is shown in Fig.2.

A client is composed of five parts, which are “Core engine”, “Schedule management engine”, “Decoder component management engine”, “File I/O”, “User Interface”. Each interface between them is shown as a number of circles in Fig.2. Their details are described below.

At a server-side, there are “Content server” storing actual bitstreams of contents, “Decoder server” storing binary codes of the decoders and “Information server” storing information describing a profile for each content (here after called content information). To detect the corresponding content information easily by information server, this information has a hierarchical structure. Content information is shown in Fig.3. Each server may be on the same machine or the different machines, which enable to achive load balancing.

We implement a client system by using Java. The part of media processing is developed with JMF (Java Media Framework)[3]. This is because the proposed system should work in the various environments. At the present time, it throws a little doubt to construct the part of media processing by using Java. Therefore, we design the system architecture in such a way that the module for media processing depending on JMF can be detached from other parts entirely. However, decoders developed

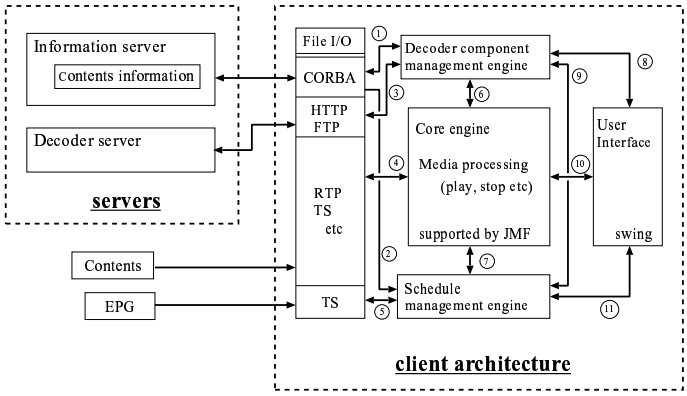

Fig. 2. Decoder-Downloadable System Architecture

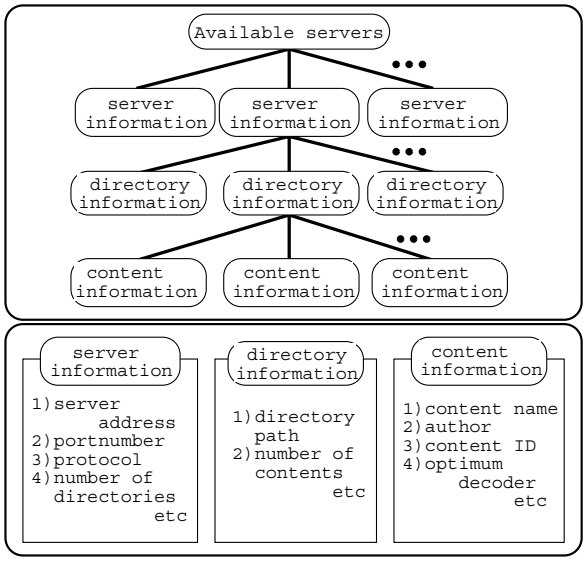

Fig. 3. Content information

by Java give a merit that server do not have to take care of the type of the client operating system. They are also useful for flexible decoder[1].

CORBA (Common Object Request Broker) [4] is used to define its interface between servers and clients being independent from the hardware specification. Moreover, the system is independent from any transport protocol by CORBA GIOP (General Inter-ORB Protocol). Therefore, the system can be employed on the Internet, TV broadcasting, and mobile system and so on.

The procedure in the system is executed in the following order:

(1) The scheduler management engine in the client system starts to get the content information. It parses the information and confirms the decoder component management engine whether the decoder has already existed in

[1]This detail is described in section 3.

the client's side or not.

(2) If a client dose not have the corresponding decoder, the scheduler management engine instructs the decoder component management engine to get the decoder URI. Decoder component management engine requires content server where the decoder is stored.

(3) Information server starts to find where the requested decoder is stored using decoder information. It returns decoder URI to the client.

(4) The scheduler management engine decides when to start downloading the decoder from decoder server and instructs the decoder component management engine. The timing to give the instruction is very important, because decoder downloading interrupt continuous media playback in case, such as low network bandwidth.

The role of each part in the system is as follows:

- Core engine

  This part provides the implementation of media play, trick mode and so forth. The decoders are actually installed in this part.

- Decoder component management engine

  This part manages decoders in the client's system, queries about what decoder is needed, downloads decoders, and deletes decoders in the stock.

- Schedule management engine

  This part controls when to start media play, to download or to delete decoders. It instructs to each part that provides actual above functions in timely manner.

- User interface

  This part is the interface for user input.

- File I/O

  This part selects a protocol based on its usage: 1) CORBA, Sending and Receiving Content information, 2) HTTP / FTP, Downloading the binary data of decoders, or 3) RTP / MPEG2-TS, Receiving media data.

Each interface in the system is described below:

interface 1: Exchanging the information about decoders.

interface 2: Exchanging the information about the contents.

interface 3: Downloading binary data of decoders.

interface 4: Downloading media data of contents.

interface 5: Downloading EPG (Electronic Program Guide) data that is used in the client's scheduler.

interface 6: Providing of decoder modules to Core Engine.

interface 7: Indicating when decoder's memory load starts, providing the information about what state the memory load is.

interface 8: Indicating whether the decoder is added or deleted to the users.

interface 9: Providing the information about the contents, indicating when to start downloading the decoder and deleting the decoder in the client system, confirming for existence of decoders.

interface 10: Accepting of the input information by the users.

interface 11: Indicating the information of contents that can be played back, choosing the contents by the users.

C. Result of system experiment

It is important to evaluate the scheduling ability of downloading decoders. Thus, the relation between the number of downloading decoders and time delay until a new content starts should be evaluated.

The experimental parameters are as follows. In this experiment we used a simulation program providing functions such as rotation effect instead of the decoders. The special effect programs work just like decoders in this system. The allowable number of them is defined as 12. The size of each decoder is 4.25KB. Client's environment is CPU : Pentium III 1GHz, RAM : 256MB, OS : Windows 2000, Version of Java : JDK1.3.1 .

Waiting time to play a content back with downloading decoders is evaluated by several scheduling methods, which are 1) "no download" meaning that a client has already necessary decoders, 2) "parallel download" meaning that a client downloads in parallel, 3) "serial download" meaning that a client downloads decoders in serial according to the order of playback. The result is shown in Fig.4. The horizontal axis shows the number of stored decoders. The vertical axis shows the actual time delay.

The result shows that the serial download is better than the parallel download. The reasons are:

1. The bandwidth of the network

   When there is heavy media streaming traffic, the parallel download consumes more bandwidth than the serial download. Therefore, the parallel download causes the more delay of downloading the decoders than the serial download.

2. The local resources

   Media processing occupies processing power. In this sense, the local resources are getting fewer by increasing the sessions for downloading the decoders. This reduction of the local resources affects media processing.

In case of Push service such as TV broadcasting, the optimum number of downloading decoders that achieves the minimum time delay is given by the probability that a user switch the content, such as channel hopping. Although this probability may obey a specific distribution, it is difficult to assume that this probability in practice.

This probability list may be created by storing history of changing contents. However, we have not implemented this function yet.

III. Flexible Decoder

A. Features

Flexible decoder enables the architecture that is not treated as the monolithic program but that is as the aggregation of the modules that provide functions. Flexible decoder was once considered in the standardization work in MPEG-4 Systems [5]. MPEG-4 was designed to establish a flexible and extensible architecture that provides some functions such as users' content-based manipulation as well as the efficient compression. Although flexible decoder architecture is not included in the standard of MPEG-4 Systems, it enables that clients construct decoders by downloading only necessary modules when a new type media content arrives. It reduces network bandwidth and clients' / servers' resources for downloading decoders. Therefore, it is efficient to apply flexible decoder in decoder-download system.

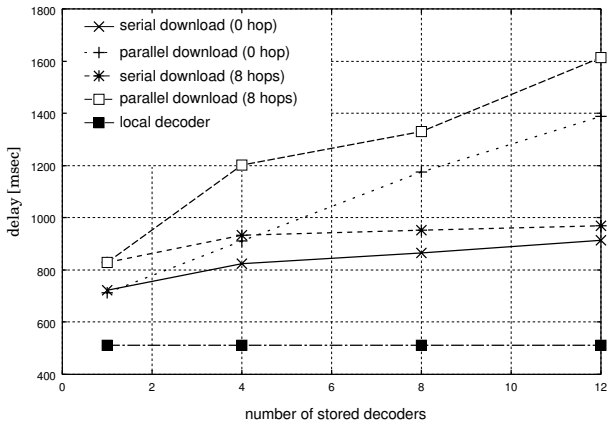

Fig. 4. The relation between the number of downloading decoders and time delay

B. Approach for Flexible decoder

Previous work [6] shows a method for Flexible decoder. In this work, users can change the image processing algorithms, such as DCT and Wavelet, by developing decoders based on the [6], while decoding process is still in progress. Separation of decoder's functions is achieved by three components model. Three components model means that a decoder is separated into 1) bitstream parsing block, 2) data restorer block and 3) image reconstrutor block shown in Fig.5. Bitsream parsing block provides parsing bitstream in accordance with the one's syntax. Image reconstrutor block provides each actual image decoding algorithm. Data restorer block has a key role for the separation between bistream parsing block and image reconstrutor block. Data restorer block provides data alignment for image reconstrutor block. It can be viewed as a conversion process between them. Consequently, the existance of data restorer block ensures that the modification to one side will not influence the other side.

This decoder architecture can be regarded very useful for Flexible decoder. However, it becomes more difficult to develop them in accordance with the architecture compared with the conventional way, which means developing decoders as a monolithic program.

C. Detach image processing from bitstream parsing

In this proposal, the modules for bitstream parsing are entirely detached from the ones for image decoding. This enables decoders to use various modules for the same image decoding algorithms. For example, it is possible to select an IDCT module suitable for clients' hardware specification (MMX, 3DNow!, DSP, ASIC and so forth), to share same bitstream parsing. Decoder-downloadable system can be used for the various multimedia system such as the search engine [7] as well as the media processing. This flexibility can be realized simply by replacing the modules for image decoding algorithms with the ones for the other processes. Client-independent decoder software such as JAVA is exploited to make the separation more flexible.s

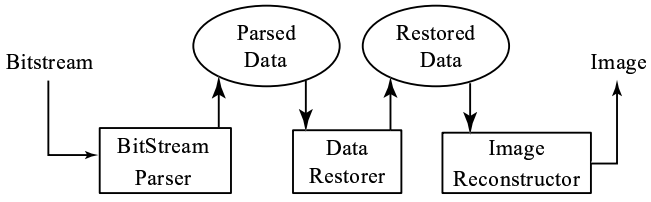

Fig. 5. Flexible decoder structure

D. Proposed structure

We propose the method for separation between bitstream parsing block and data restorer block. This is achieved by the support of MSDL-S (MPEG4 Systems and Description Language - Syntactic description language) and Flavor (Formal Language for Audio-Visual Object Representation) [8].

1) MSDL-S and Flavor: MSDL-S is defined in the standard of MPEG4 Systems for description of biststream syntax shown in Fig.6. MSDL-S is used for generating bitstream parsing modules as well as describing bitstream syntax shown in Fig.7.

Flavor is a representative MSDL-S compiler, which is proper that the code generator instead of the compiler because Flavor provides the only function generating standard Java and C++ source code automatically at the present time. It is possible to simplify and speed up the development of software that processes multimedia information by MSDL-S and Flavor. Bitstream parsing is required to develop the modules that deal with the bitstream-oriented nature of the data, although general-purpose microprocessors are strictly byte-oriented. MSDL-S enables bit-oriented. Thus, MSDL-S provides the functions to cope with the bitstream data easily.

2) Actual process: MSDL-S and Flavor provide only the bitstream parsing modules with ease. Thus, decoder developers must implement their own interfaces with data restorer block. This approach cannot realize the fixed interface. Thus, it is impossible to share the bitstream parsing modules. Consequently, we modify the Flavor to generate the base models for the data restorer block as well as the bitstream parsing block (Fig.8). This modification is explained as follows. A set of related data, in other words a data layer, is represented as Class in MSDL-S. Flavor generates each Java and C++ source code corresponding to the data layer. Thus, we improve to enable that the base models for data restorer block corresponding to the data layer is created at the same time. In short, MSDL-S source file is used as IDL (Interface Definition Language) for the data restorer block

```
class single_pixel {
    int(8) pixel;
}
```

Fig. 6. Representation with MSDL-S

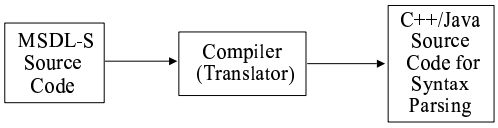

Fig. 7. Generate bitstream parsing from MSDL-S Source code

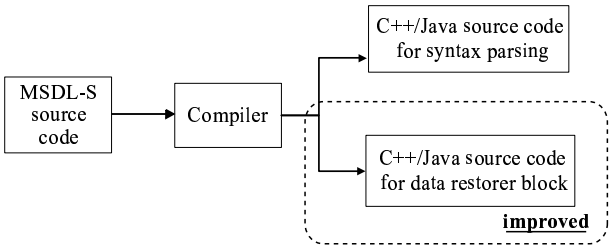

Fig. 8. Proposed flexible decoder generator

modules as well as the bitstream parsing modules. Each decoder developer implements the data restorer block modules for own use, while these decoders share the same bitstream parsing modules.

IV. Conclusion

In this paper, we proposed a new system architecture for content-oriented coding scheme. This system enables dynamic decoder downloading by negotiation between clients and servers. Therefore, clients can decode the contents without the knowledge about image coding scheme and can play back media contents seamlessly. This system is superior for the portability, because it is developed by Java and CORBA, It is addressed that development decoders as flexible decoder are useful for decoder-downloadable system. The method to separate bitstream parsing from the other processes using MSDL-S is also proposed.

References

[1] O. Nakagami, T. Miyazawa, H. Watanabe, H. Tominaga, “A Study on two-layer coding for animation images,” IEEE International Conference on Multimedia Expo (ICME) 2002, August 2002.

[2] D. Wu, T. Hou, W. Zhu, Y.-Q. Zhang, J. M. Peha, “Streaming Video over the Internet: Approaches and Directions,” IEEE Trans. on Circuits and Systems for Video Technology, February 2001.

[3] Sun Microsystems, “JavaTM Media Framework API,” http://java.sun.com/products/java-media/jmf/

[4] Object Management Group, “CORBA/IIOP Specifications,” http://www.omg.org/technology/documents/

[5] ISO/IEC JTC1/SC29/WG11, “ISO/IEC 14496-1, Information technology - Coding of audio-visual objects - Part 1: Systems,” 2000.

[6] H. Arakawa, T. Maeda, M. Etoh, “Software architecture for flexible and extensible image decoding,” Signal Processing Image Communication 10, pp.235-248, 1997.

[7] J. R. Smith and S. F. Chang, “VisualSEEk: a Fully Automated Content-based Image Query System, ” In Proceedings of the 1996 ACM Multimedia Conference, pages 87-98, Boston, MA, 1996.

[8] A. Elftheriadis, “Flavor: A Language for Media Representation,” ACM Multimedia 97 Conference, Seattle, WA, November 1997.

[9] ISO/IEC JTC1/SC29/WG11, “ISO/IEC 13818-2, Information technology - Generic coding of moving pictures and associated audio information: Video,” 1996.