Design of Android based Media Player

Nikhil S. Sakhare[1], R. W. Jasutkar[2]

[1]P.G. Student, M.E. (Mobile Technology),
Department of Computer Science & Engineering,
G.H. Raisoni College of Engineering,
Nagpur, India
nikhilsakhare.06@gmail.com

[2]Assistant Professor,
Department of Computer Science & Engineering,
G.H. Raisoni College of Engineering,
Nagpur, India
ratnaprabha.jasutkar@gmail.com

Abstract: ***Many users like to watch video by a mobile phone, but the media player has many limitations. With a rapid development of communication and network, multimedia based technology is adopted in media player. Different approaches of media player shown in this paper are plug-in extension technology, multimedia based on hierarchy, media player based on file browser, media player based on FFmpeg (Fast Forward Moving Picture Expert Group), media player based on file server.***

Keywords: media player, FFmpeg, file browser, file server.

1. Introduction

With the continuous development of science and technology, mobile phone is no longer just communication, but a multimedia platform that provides multimedia capabilities. Playing a video on media player becomes basic function, but the media player has many limitations since there're limited format supported by media player.

At present, the decode module of most of the media players is based on FFmpeg decode library which supports more than 90 kinds of decoders, such as Storm Codec, KMP Codec, etc and the display module is based on SDL (Simple DirectMedia Layer) [1], [2], [4], [20].

This paper shows different approaches for design of media player. First is plug-in extension technology on android multimedia player software platform. Second is media player based on hierarchy. Third is android media player software development. Fourth, is android media framework and fifth, is continuous media player.

Section II provides an overview of different approaches to be used. Section III describes the comparison of different approaches and finally section IV refers to conclusion.

2. Different Approaches

FFmpeg is an open source that produces libraries and programs used in audio and video areas. It support more than 90 kinds of decoders and also support protocols such as H.261/H.263/H.264 [12], [14] and so on. It provides a complete solution to transcode, record and stream audio/video [16]. It includes an audio/video codec library, an audio/video container mux and demux library, the transplantation and codec quality [1], [2], [4].

SDL is a library used to display audio and video information. Display module of most of the media player is based on SDL [1], [2], [4], [20].

2.1 Plug-in extension technology on android multimedia player software platform

2.1.1 Multimedia player software platform on android

Jin and Jiaming describes multimedia player software platform on Android with the use of the OpenCORE kernel. Packaging the kernel and providing in the form of SDK is used to develop multimedia player application in mobile terminal, such as video player and streaming media player, etc [1].

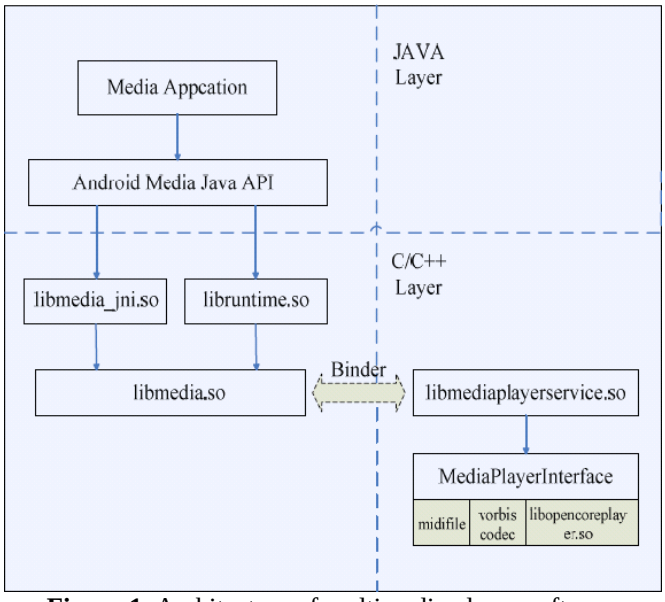

Figure 1. Architecture of multimedia player software platform

2.1.2 Architecture of plug-in software platform on FFmpeg

Wu and Xiao show the development of application needs powerful decoding and container that support more video encoding formats. It describes the upper layer is Java, which provides the application development interface (Android_FFmpeg API). The bottom layer is the C/C++ layer, which is the core layer used to process audio/video data with FFmpeg [11].

The implementation of FFmpeg consist of audio/video which is packed at the bottom layer includes the data source analysis of audio/video, the play of audio/video, paused and callback related mistakes, etc. The implementation of the bottom layer will be provided by Android_FFmpeg API through Java Native Interface (JNI) [1].

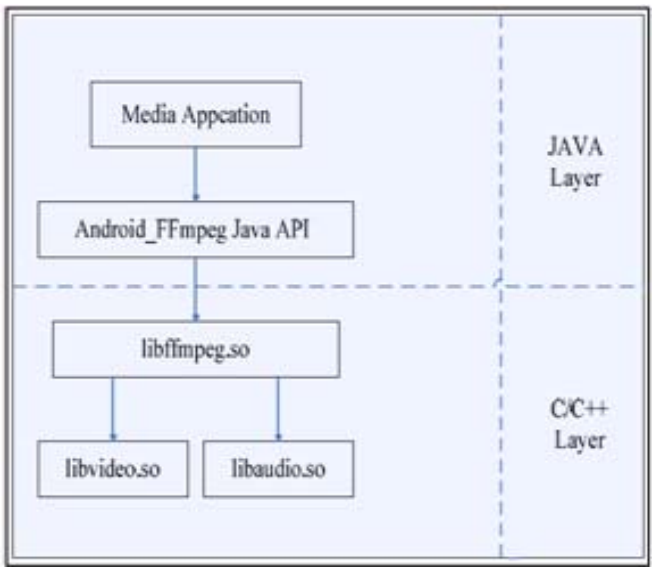

Figure 2. Architecture of plug-in software platform based on FFmpeg

The bottom layer involves the audio/video data processing core (libffmpeg.so). It packages the related audio/video function of FFmpeg. It includes decoding of all major audio/video, synchronization between multiple media streams [17], completing the display and playback features of the audio/video [20], after decoding low-level audio/video equipment libraries (libvideo.so; libaudio.so) [1].

The decoding of selected components achieves the decoding format from the header of the native file or streaming media file. It also selects corresponding decoder to decode compressed media streams [1].

2.2 Media player based on hierarchy

2.2.1 Design of media player

Song et. al., shows to play media files, the media player gather the media data first, decode the audio/video streams later, then display the data after decoding [2].

During the three steps, media player needs to parse the coding format of the media file, decode for the original data by the corresponding decode programs, put the original data to buffer queues, then display the original data after being synchronized. Design reduces the application coupling [2].

The layers of system structure of media player are UI, decode layer, pretreatment layer and data extract layer [2].

(a) UI

UI is used to display the original data on media player for users such as functions of play, pause, page down, page up, etc [10], [11].

(b) Decode layer

Decode layer is used to gather information of the media file formats, and then decodes audio/video streams by the corresponding decoder, and then synchronize the audio/video streams. It includes all kinds of decoders, the decoder choosing module and the synchronize module [11], [21].

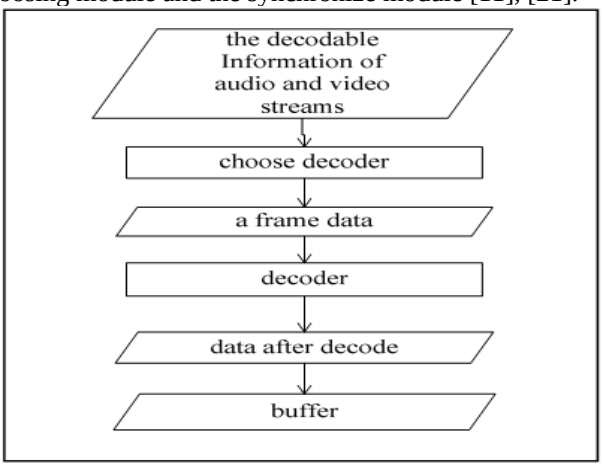

Figure 3. Flow Chart of Decode layer

Before decoding, registration of all the formats is necessary which can be decoded by the module. Then provide a link to connect the corresponding decode unit and a media format [2].

(c) Pretreatment layer

Pretreatment layer is used to demux the media file according to the available format and store the information of the media file into the buffer.

(d) Data extract layer

Data extract layer is used to read the media file.

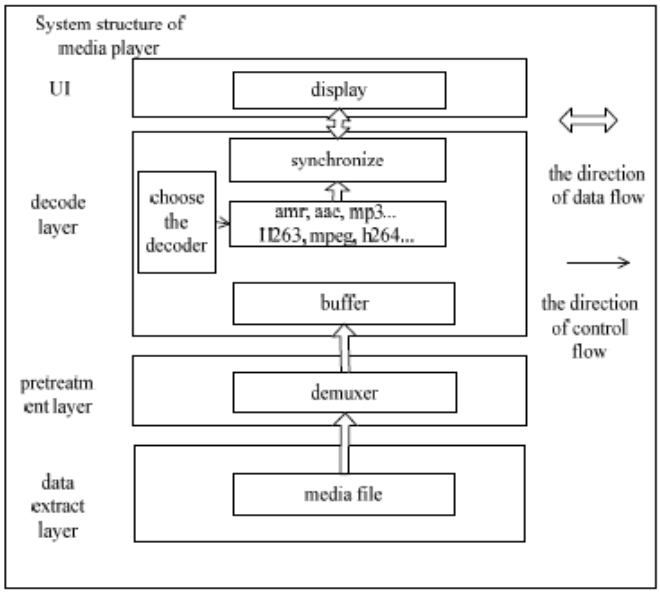

Figure 4. System structure of media player

2.3 Android media player software development

2.3.1 Android media player

Jin et. al., shows common media codec mechanism, so it is easily been integrated to multimedia files such as audio, video and pictures. Android Media Player plays audio files such as local files, resource files and network file streams from many sources. This includes media player plays audio files from SD card and displays the lyrics synchronously [3], [13].

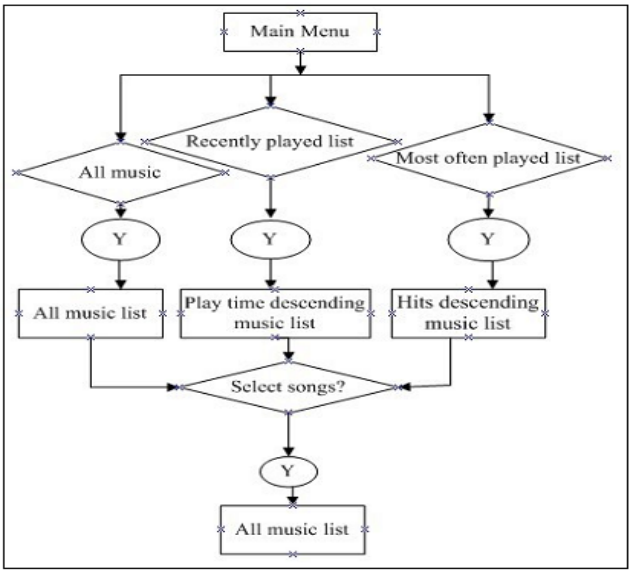

Figure 5. System processes of media player

2.3.2 Software module

(a) Main menu module:

Entering in the initial interface of media player, it consists of three options: All music, recently played list, and most often played list [3].

When Android system is start, media player automatically scan multimedia files in the SD card. Simultaneously, store the received information in a system database. To make the data in the multimedia database, broadcast mechanism is used. Then sending a broadcast in the application to update the multimedia database by scanning the SD card. Then register the scanSd Receiver, Broadcast Receiver [3].

(b) Play List module:

Play List is a major part to show the name of audio files, its singers and durations. Play List is used to create a list, when using the mouse to click one of the items in ListView, it will trigger setOnItemClickListener monitor [3].

(c) Recently played module and the most often played module:

Function onCreate() and onUpgrade() is used to create and update the database. Then invoke queryRecently() to realize recently played list by inquiring songs according to play time descending and invoke queryByClicks() to realize Most often played list by inquiring songs according to the hit descending [3].

(d) Play module:

The main function is to display the information of title, lyrics and time about the song, and some of the media player's functional keys, such as play, pause, stop, last, next, backward and forward, and then display the lyrics [3].

2.4 Android media framework

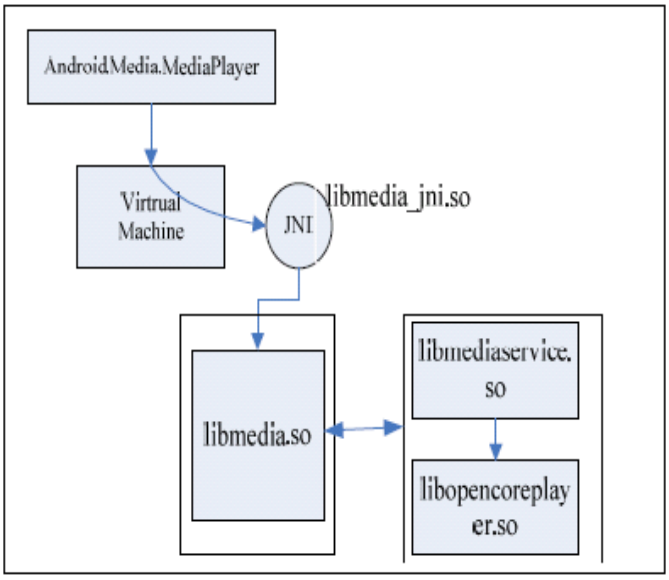

Figure 6. Android media framework

2.4.1 Android media framework architecture

Song et. al., describes the goal of Android media framework is to provide a consistent interface for all services provided by libraries. The core part of the media framework is composed of libmediaplayerservice, libmedia and libmedia_jni [4].

libmediaplayerservice implements players and the media service which manage player instances.

Libmedia defines base interfaces and the inheritance hierarchy. libmedia_jni is the centre between java application and native library. First, it implements the JNI specification so that it can be used by java application [11], [26]. Second, it implements the pattern for the convenience of caller [4].

2.4.2 Android media players

Media Player is an important part of Android media framework. It is used to control the playback option of Audio and Video [20].

Methods of Media Player are implemented in C/C++ and then compiled to .so file. JavaTM Native Interface (JNI) is a standard programming interface for writing Java native methods and embedding the JavaTM virtual machine into native applications. Once the components decode the source audio file, the decoded stream is send to audio hardware to turn into sounds [4], [11], [26].

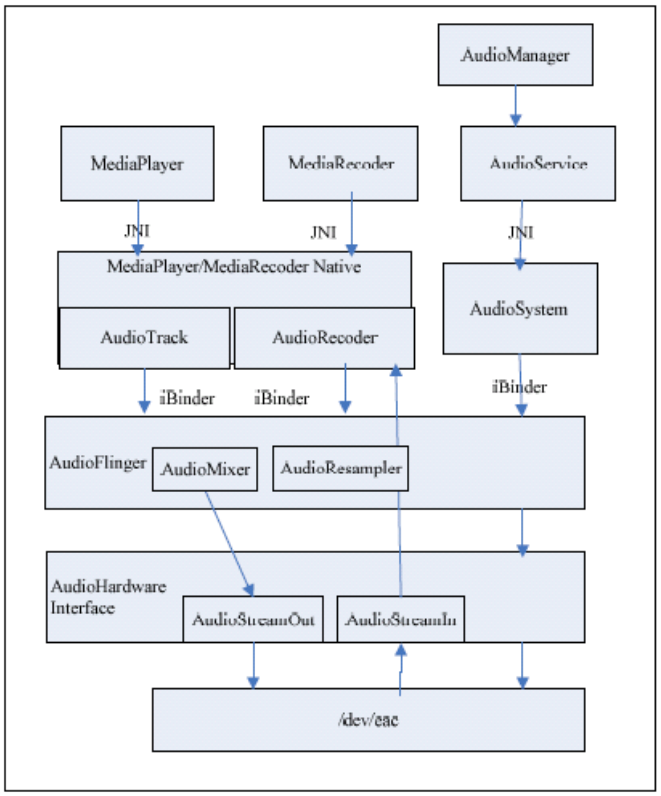

Figure 7. Android media framework layer

2.4.3 FFmpeg decode work flow

FFmpeg provides powerful media encode/decode functions. First of all register all kinds of codec defined in FFmpeg to system. So that the audio/video files can be decoded by FFmpeg. Then open the media file from local files and get the file stream information and the file stream type. After these, choose a appropriate decoder and allocate memory for data structures AVCodec, AVFormatContext, AVStream.. And then divide the file stream into audio stream and video stream [4].

AVFormatContext is used to save the input/output data. AVCodecContext save the pointer of AVCodec and data related with codec. AVCodec is used to store the codec information such as pixels format information. AVStream saves the data segment [4].

2.4.4 FFmpeg transplant

FFmpeg is a complete, cross-platform solution to convert record and stream audio/video files. The makefile Android.mk of NDK is different from normal makefile. So the precondition is modifying the makefile of FFmpeg.

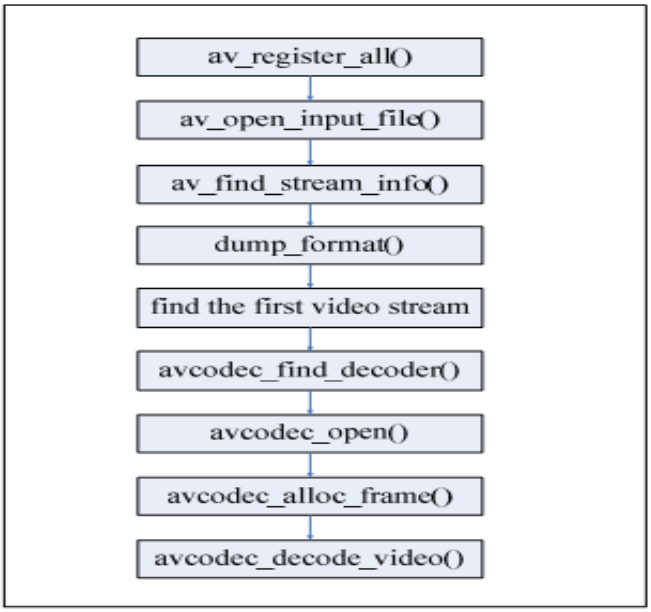

Figure 8. Decode work flow

Transplantation of other files and then, files are compiled into .so files. These .so files are as components to provide the functions. The java applications invoke these files through JNI [4].

2.5 Continuous media player

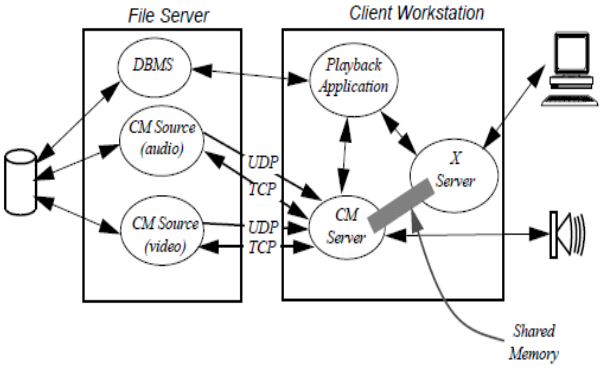

Figure 9. CM player architecture

2.5.1 CM player system architecture

Rowe and Smith shows the playback application is the interface between the user and the CM Server process. The application includes creating windows, responding to input events, and sending commands to the CM Server [5].

The CM Server receives CM data from the CM Source and sends it to the appropriate output device. The CM Server has a time-ordered play to synchronize the playing of audio and video packets [15]. It communicates with CM Source processes on the file server through inter process communication channels, and it communicates with the X server through shared memory. The system clocks on the different systems are synchronized so that actions in the CM Server and Sources can be synchronized. The CM Source processes read CM data and send it to the CM Server. CM data is sent in 8k packets on a UDP connection [5].

Meta data about scripts is stored in a database. The meta data is separated from the raw CM data so that different scripts can include overlapping clips without having to make a copy of the CM data [5].

3. Comparison

3.1 Time based audio and video synchronized algorithm

FFmpeg provides DTS and PTS parameters, DTS is decoding time stamp and PTS is time show stamp. When obtaining a data packet from the multimedia data stream.

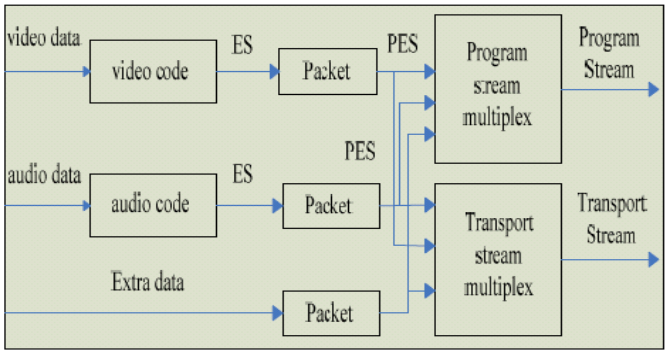

Figure 10. The relationship diagram of the transport stream. PES stream and ES stream

3.2 Time-based multimedia synchronized transmission algorithm is as follows:

$$\begin{cases} Handup(f_{curframe}) & T_{pts} > T_{clock} \\ Play(f_{curframe}) & T_{pts} = T_{clock} \\ Pass(f_{curframe}) & T_{pts} < T_{clock} \end{cases}$$

Jia-ming and Jin shows display and playback the multimedia data (audio or video), it is necessary to extract time stamp information (Tpts) from the multimedia data block, and compared to the current reference time (Tclock). If the timestamp of multimedia data block is less than the current reference time (Tpts< Tclock), play the current data as soon as possible, or even discard the current multimedia data directly; If the multimedia data blocks of time stamp is greater than the reference time stamp (Tpts> Tclock), the data block is transferring immediately to suspended state, and waiting to play [6].

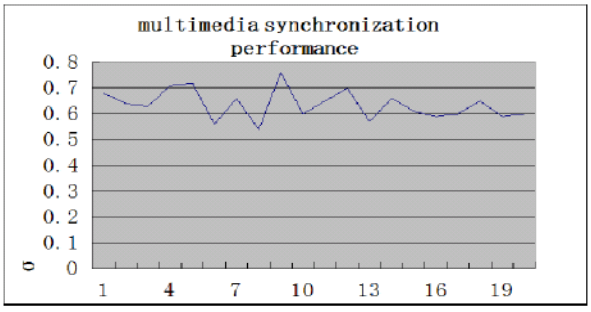

Figure 11. Multimedia synchronization performances

Selected 20 video files whose container format is flv and bit rate is 800kbps, to test multimedia synchronization performance.

Table 1: Shows algorithm, audio and video synchronization, and codec standard support for different approaches

Sr. No	*Different Approaches*	*Algorithm*	*Audio and Video synchronization*	*Codec standard support*
1.	Plug-in extension technology based on android multimedia player platform	FFmpeg transplan-tation	synchronization between multiple media streams	aac; ac3; flv4; mpeg; Audio Mpeg1; Mpeg2; divx, etc
2.	Media player based on hierarchy	FFmpeg transplan-tation, Play module, UI	Display the original data after being synchronized	amr; aac; mp3; h263; mpeg; h264
3.	Android media player software development	Main menu module, Play list module, Recently played module, Play module	Plays audio files from SD card and displays the lyrics synchronously	mp3; amr; aac;
4.	Android media framework	FFmpeg decode, FFmpeg transplan-tation	Display all music files in list view and control playback of the audio	rm; wam; mp3; h263
5.	Continuous media player	CM data model, CM server abstract-ion, CM network protocol	synchronized clock to synchronize the process that plays the CM data on a client	jpeg; mpeg;

4. Conclusion

This paper shown different approaches for design of media player. Media player should consider the improvement in scenario such as decode efficiency needs to be improved, synchronization between multiple media streams, and display of the original data. Use of FFmpeg decode library seems to be an alternative method, research shows FFmpeg supports most media formats which gives a high decode efficiency.

Different approaches shown in this paper are plug-in extension technology, multimedia based on hierarchy, media player based on file browser, media player based on FFmpeg, media player based on file server.

References

[1] Jin He and Jiaming He, “The Research of Plug-in Extension Technology Based on Android Multimedia Player Platform”, IEEE 2011 International Conference on Computer Science and Service System (CSSS-2011), pp.874-877.

[2] Maoqiang Song, Jie Sun, Xiangling Fu and Wenkuo Xiong, "Design and Implementation of Media Player Based on Android", IEEE 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM-2010), pp.1-4, 23-25 Sept. 2010.
[3] Shuangyan Jin, Haoliang Li and Yongfei Liu, "Research on Media Player Based on Android", IEEE 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD-2012), pp.2326-2329.
[4] Maoqiang Song, Wenkuo Xiong and Xiangling Fu, "Research on Architecture of Multimedia and Its Design Based on Android", IEEE 2010 International Conference on Internet Technology and Applications (ITAPP-2010), pp.1-4.
[5] Lawrence Rowe and Brian Smith, "A Continuous Media Player", Workshop on Network and OS Support for Digital Audio and Video, Computer Science Division-EECS, pp.1-11, Nov.1992.
[6] He Jia-ming and He-Jin, "Research on the Synchronized Transmission Algorithm of Embedded FFmpeg Multimedia Data", IEEE 2011 International Conference on Internet Technology and Applications (iTAP-2011), pp.1-3.
[7] Zhengzheng Zhang and Yaorong Lin, "Development on video player based on Android system", Vol. 2, Modern electronic technology, 2011, pp.5-8.
[8] Jianye Liu and Jiankun Yu, "Research on Development of Android Applications", IEEE 2011 4th International Conference on Intelligent Networks and Intelligent Systems (ICINIS-2011), pp.69-72, 1-3 Nov. 2011.
[9] Yang Fengsheng, "Android Application Development Secret", Beijin. China Machine Press, 2010, pp.188b.
[10] Chun Yuan and Wenshuo Zhou, "Design and Implementation of Embeded Streaming Media Player Based on STB", IEEE 2011 International Conference on Multimedia Technology (ICMT-2011), pp.3017-3019.
[11] Junqin Wu and Fangyang Xiao, "Design of Embedded Streaming Media Player based on J2ME", IEEE 2011 International Conference on Multimedia Technology (ICMT-2011), pp.512-514.
[12] Seungsoon Lee and Minseok Song, "Selective Frame Prefetching for Reducing Disk Energy Consumption in Scalable Video Coding (SVC) Media Players", 2012 IEEE Transactions on Consumer Electronics (TCE-2012), Volume.58, Issue.2, pp.428-434, May 2012.
[13] Deeksha Saraswat and Shashank Batnagar, "Smart Player: A New Era of Media Players", IEEE 2011 International Conference on ICT Convergence (ICTC-2011), pp.756-759.
[14] Wim Van Lancker, Davy Van Deursen, Ruben Verborgh and Rik Van de Walle, "Semantic media decision taking using N3logic", Springer Science & Business Media, LLC 2012, 15 March 2012, Springer 2012.
[15] Wim Van Lancker, Davy Van Deursen, Erik Mannens and Rik Van de Walle, "Implementation strategies for efficient media fragment retrieval", 26 March 2011 Springer Science & Business Media, LLC 2011, pp.243-267, Springer 2012.
[16] Liao Jianxin, Lei Zhengxiong, Ma Xutao and Zhu Xiaomin, "Proxy-Based Patching Stream Transmission Strategy in Mobile Streaming Media System", Journal of Electronics (China), Vol.23 No.4, July 2006, pp.515-519, Springer 2006.
[17] Heejin Ahn, Seongjin Cho, Hyunik Na and Hwansoo Han, "Access Pattern Based Stream Buffer Management Scheme for Portable Media Players", 2009 IEEE Transactions on Consumer Electronics (TCE-2009), Volume.55, Issue.3, pp.1522-1529, August 2009.
[18] Wanhyung Ryu and Minseok Song, "Design and Implementation of a Disk Energy Saving Scheme for Media Players Which Use Hybrid Disks", 2010 IEEE Transactions on Consumer Electronics (TCE-2010), Volume.56, Issue.4, pp.2382-2386, November 2010.
[19] Wenhao Wang and Mingyu Gao, "Design of Embedded Media Player Based on S3C2440 and SDL_FFMPEG", IEEE 2011 International Conference on Electrical and Control Engineering (ICECE-2011), pp.2979-2982.
[20] Jaewoo Kim, Ahron Yang and Minseok Song, "Exploiting Flash Memory for Reducing Disk Power Consumption in Portable Media Players", 2009 IEEE Transactions on Consumer Electronics (TCE-2009), Volume.55, Issue.4, pp.1997-2004, November 2009.
[21] Injae Lee, Hankyu Lee, Jinwoo Hong, and Jihun Cha, "Interactive Contents Player for Rich Media Service", 2009 IEEE Transactions on Consumer Electronics (TCE-2009), Volume.55, Issue.1, pp.112-118, February 2009.
[22] George Toma, Laurent Schumacher and Christophe De Vleeschouwer, "Offering Streaming Rate Adaptation to Common Media Players", 2011 IEEE International Conference on Multimedia and Expo (ICME-2011), pp.1-7.
[23] Han Hu, Jian Yang, Zilei Wang, Hongsheng Xi and Xumin Wu, "Scene Aware Smooth Playout Control for Portable Media Players over Random VBR Channels", 2010 IEEE Transactions on Consumer Electronics (TCE-2010), Volume.56, Issue.4, pp.2330-2338, November 2010.
[24] Jing Chen, Jiajun Wang, Chuixin Zeng, Zeyu Chen and Muhammad Jahanzaib Khan, "iPhone-based Multi-stream M-learning Platform", 2012 IEEE Symposium on Electrical & Electronics Engineering (EEESYM-2012), pp.742-746.
[25] Won-Jin Kim, Keol Cho and Ki-Seok Chung, "Stage-based Frame-Partitioned Parallelization of H.264/AVC decoding", 2010 IEEE Transactions on Consumer Electronics (TCE-2010), Volume.56, Issue.2, pp.1088-1096, May 2010.
[26] Xiangling Fu, Xiangxiang Wu, Maoqiang Song and Mian Chen, "Research on AudioVideo Codec Based on Android", IEEE 2010 6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM-2010), pp.1-4.