Design and Implementation of Key Frame Sharing Application in Android Using Chord Protocol

K.W. Kim^{*}

ABSTRACT

Due to the popularity of mobile phones, Peer-to-Peer (P2P) content sharing in mobile phones is highly desirable, especially in a mobile data connection. In addition, there has been an increasing demand for content sharing applications for P2P communication in Android-based smartphones as ubiquitous mobile devices. However, due to wireless network limits, unstable characteristic and restricted storage space, mobile users confront challenges in establishing connections and sharing video frames with other users. To solve this issue, we propose an architectural design and implementation of key frame sharing application that uses Chord, a distributed lookup protocol to establish a content sharing platform for mobile devices, combined with the Chord SDK based on Android. Our proposed system enables users easily to share key information on video data in real-time over a wireless network without manual processing of devices which join or leave the group and cost associated with the 3G/4G network.

Key words: Key Frame, Chord Protocol, Android, Sharing

1. INTRODUCTION

With the high growth of mobile technology, Android-based smartphones and tablets have advanced rapidly in recent years. The number of smartphone users has also increased sharply because they are powerful multimedia devices which are equipped with a low power high performance processor and high resolution touch screen with cameras and sensors. Already, mobile devices are not just used for making a call or sending an SMS. They can provide us with the multimedia functionalities such as taking pictures, recording video, playing games, and searching the Internet. Wireless networks will be more popular both at work and home in the near future and the convergence of mobile or wireless devices significantly change

people's everyday lifestyle and become the main means of communication. In addition, a lot of users are still deeply interested in sharing their multimedia content. For instance, with the development of smartphones with high-quality cameras, there is an intention of users to share their own content such as photos or videos with nearby smartphones or tablet devices. However, such users still face the problem related to high costs when using the mobile data networks such as 3G/4G and uploading data to a certain server. Therefore, it is common that most of users are normally subjected to mobile data plans with limited usage. It is a matter of fact that the design of P2P protocol in a mobile environment mainly depends on the underlying wireless technology such as Wi-Fi and Bluetooth. One of the main advantages of wireless mobile P2P network is sharing resources in a mobile environment, and it also has many other advantages such as high scalability, full decentralization, highly dynamic, self-organizing in mobile environment, and convenience of establishing network [1]. There has been a wide use of many P2P applications in the

※ Corresponding Author : Kang Wook Kim, Address : (445-855) 148-901, Lotte Castle Apt., Seoku-dong, Hwa-seong-si, Gyeonggi-do, Korea, TEL : +82-10-2678-2501, E-mail : ekans999@gmail.com

Receipt date : Jan. 28, 2014 , Revision date : Feb. 25, 2014

Approval date : Mar. 14, 2014

^{*} Samsung Electronics Co., Ltd

wired network environment in the last few years. However, there is no efficient adaptation of these new technologies to a wireless mobile environment [2]. Currently, smartphones and tablets have become multimedia devices with high capability in the mobile environment. In addition, the introduction of Wi-Fi Direct protocol enable users to make good use of the P2P communication with mobile devices anytime and anywhere in a direct manner without any interface router or access point (AP) [3].

Typically, P2P communication systems are divided mainly into unstructured P2P system and structured P2P system or distributed hash table (DHT), such as Chord [4]. Structured P2P networks became popular for its advantages of high scalability and good performance. The main advantage of DHT-based systems is that data placement and search procedures generate less traffic. Among the DHT-based approaches, Chord is one of the most popular systems. As structured P2P networks, Chord protocol, which is the most representative network architecture protocol has not only simple design ideas, but also good distribution, scalability, stability and load balancing. For designing the P2P applications for different devices, it only depends on the operating systems (OS) running on these devices. In our research, we have a definite reason for choosing the Android-based mobile devices including the Google Android platform where the entire OS is open and accessible to developers and application programming is easily supported by software development kit (SDK). Therefore, in this paper, we propose an architectural design and implementation of key frame sharing application that uses Chord, a distributed lookup protocol to establish a content sharing platform for mobile devices, combined with the Chord SDK based on Android.

The remainder of this paper is organized as follows: In section 2, we introduce our proposed key frame extraction system and its block diagram.

Then, in section 3, we briefly review standard Chord protocol, and device connection mechanism via a Chord channel. Section 4 gives details about the architectural design and implementation of the key frame sharing application in Android, demonstrating the Chord SDK, class view and user interface of the proposed system. Compatibility test results for Android OS versions and various video sequences are presented in Section 5. Finally, section 6 concludes the paper.

2. KEY FRAME EXTRACTION

The use of key frames to represent the content of a video has been previously discussed in many research papers as an efficient way of preserving the temporal information of sequence based on a small amount of data [5,6]. The underlying assumption is that if these frames are extracted using an appropriate video sampling method, the visual content of each segment of the sequence can be easily understood by looking at given samples. Accordingly, such compact video representation can be suitable for the purpose of video browsing. Furthermore, for query processes involving the search for video parts containing specific objects, the concept of key frames can also be useful. In our application, we adapted a three-step key frame extraction method [6] for efficient video content representation, which was proposed in 2012.

3. CHORD PROTOCOL

3.1 Description of Chord Protocol

In this section we briefly describe the standard Chord mechanism. For a more detailed description, we are referred to [4]. The Chord protocol is one solution for connecting the peers of a P2P network. Chord can simplify the design of P2P and applications based on it by solving the touch issues such as load balancing, decentralization, scalability and flexibility. Chord will be an essential component for

P2P and large-scale distributed applications. Excellent features of Chord include its simplicity, correctness, and good performance even in the case of concurrent node arrivals and departures.

Among the DHT-based approaches, Chord is one of the most popular systems. As structured P2P networks, Chord protocol which is the most representative network architecture protocol has not only a simple design idea, but also good distribution, scalability, stability and load balancing. There is a lot of research and improvement on Chord needed to meet the actual demand [7,8]. For this reason, Chord has been recently adopted as mandatory by the IETF P2PSIP (Peer-to-Peer SIP) working group [9].

3.2 Mobile device connection via Chord channel

Chord allows you to easily develop local information-sharing applications. Devices running Chord-based applications find the position of each other using UDP-broadcast-based discovery, and then use a TCP/IP-based protocol stack to create a reliable, local, P2P communications network. Chord-based applications use this network to share data, including text messages, binary messages and files, with selected members of the network. We can use the Chord to discover de-

vices, join or leave private channels and share information by sending and receiving data and files. Fig. 1 shows the system architecture of node architecture of and illustrates the scenarios of devices connected via a Chord channel. A node is a device that is connected to other devices through the Chord protocol. The Chord public channel includes all the nodes in the local Chord network, while individual applications on nodes use private channels to interact with each other. A node is always a member of the public channel, and can also be a member of multiple private channels. Devices running Chord-based applications join the Chord public channel automatically and become nodes. However, the Chord public channel is not visible to users. This means that users must use private channels to share data, conversations, and files. A private channel only includes nodes that are running the same application. Fig. 1 shows five devices connected through a Chord public channel. Nodes 2, 3, and 4 are also connected to each other in a private channel (Channel B) while nodes 4 and 5 are connected through another private channel (Channel A). Note that node 4 is connected to both Channel A and Channel B, and that node 1 is not connected to any private channels.

Fig. 1. Example scenario of device connection via a Chord channel.

4. DESIGN AND IMPLEMENTATION OF PROPOSED APPLICATION

4.1 Design of Key Frame Extraction Engine

The functions packaged in the form of library (DLL, SO) or executable file, such as Assembly, C and C++ can be called on Java layer through JNI (Java Native Interface) in Android. JNI comes from the following reasons: First, the application has to use the system-related functions, while Java does not support or is hard to implement. Second, there are many useful libraries written in other languages. Java programs can reuse them. Third, for higher performance issues, the developer has to use assembly or C/C++ code to implement some

specific program modules [10,11]. For these requirements, Android platform supports the JNI method. In this paper, we use JNI because of the second reason, for reusing already implemented C codes for key frame extraction as explained in section 2. JNI layer is exchanging the key frame data between Application UI and MPEG decoder library. Meanwhile, it provides the interface for controlling the DC images decoding.

Normally, native C code executes faster than Java code [11]. In view of the efficiency requirements of key frame extraction application, and the characteristics of Android hierarchy, the DC image extraction engine is located between Linux kernel layer and applications layer and realized by C/C++ programming language. In the DC image extraction engine, the function of Linux kernel and libraries are called to decode DC image from video stream and calculate the differences between DC images. Functions in Android application layer call the service provided by DC image extraction engine using JNI interface. The architecture of key frame extraction is designed into four layers as shown in Fig. 2. In Android, applications are developed with Java programming language based on Android SDK, but key frame extraction engine is based on C programming language.

In this paper, we develop dynamic linked library based C programming language (.so) by JNI, and

then pack the “.so” file and the Java application as a “.apk” file by Android NDK. The advantage of this approach is we can upgrade and reuse each layer because only changing the common library allows us to develop new applications. Combining hierarchical and modular design, the key frame extraction engine consists of mainly four layers, including the user interface, scene change detection, key frame allocation, and key frame distribution. This type of design approach can simplify video information processing. It is useful to develop and maintain video processing application using the key frame extraction engine. It is also easy to add a new functionality to our proposed design scheme. There is a mapping table between native functions and Android Java functions, which is registered to Dalvik VM.

The user interface layer is the interface of key frame extraction engine, and it is a JNI interface package of key frame extraction engine. Java applications can call the corresponding key frame extraction engine functions through JNI interface. The layer controls the flow of command from one layer to another. Related APIs implement the control of key frame number, widow size setting for scene change detection, and view options for display on screen. The main flow of the functional call is shown in Fig. 3. Every step's function is

Fig. 2. Structure of Android key frame extraction module.

Fig. 3. The flow chart of key frame extraction functions.

as follows:

- `mm_decode_DC( )` : Decode DC image from compressed video
- `mm_video_seg_from_DC( )` : Segment video into shots using extracted DC images
- `mm_kframe_num_alloc( )` : Allocate the number of key frames to a shot
- `mm_kframe_distributor( )` : Distribute key frames over a shot

4.2 Chord SDK on Android platform

The Samsung Chord [12] is a very effective way to share content and user events in real-time between devices. Samsung is fully committed to making Chord the useful sharing protocol for application developers. Chord is supported by Android 4.0 (Ice Cream Sandwich), API level 14 or above. Chord enables simple real-time sharing without the cloud. The Samsung Chord SDK allows application developers to develop local information-sharing applications without a detailed knowledge of networking. For this reason, we have concluded that the Samsung Chord SDK is the most efficient way to implement the prototype application for key frame sharing. We are able to implement a large variety of features with the Chord SDK. The Chord SDK is a network framework that makes it easy for users without professional networking skills to discover many devices connected to the same subnet. It helps them group the devices through Chord channels, and exchange files and data between the devices. There is the difference between Chord and other short-distance network connections such as Wi-Fi direct and Bluetooth. In general, Wi-Fi Direct and Bluetooth refer to a physical network interface connection. Chord is a top layer messaging protocol that uses the TCP/IP network. Fig. 4 shows the Chord classes and interfaces that we can utilize in our application. The Chord classes and interfaces are described as follows:

Fig. 4. Chord classes and interfaces.

- **ChordManager** : Creates a node and manages the node's connection to channels.
- **IChordManagerListener** : Listens for the connection status of the node.
- **INetworkListener** : Listens for the status of network, even if Chord has not started.
- **IChordChannel** : Manages node names and IP addresses, and data and file transfers.
- **ChordChannelImpl** : Access the channel instance by using IChordChannel.
- **IChordChannelListener** : Listens for nodes joining and leaving channels, and data transfers.

4.3 Implementation of User Interface

We need to use a fragment structure for application UI which is provided by Android 3.0 APIs as shown in Fig. 5 because we can easily compose different functionality for each layout in activity. An activity in Android OS represents a single screen with a user interface. In a multiple activities application, generally, an activity is defined as the "main" activity, which is presented to the user when user first executes the application program. A main activity of key frame list view is an object of activity type and it provides interface to users and communicates with the common library. View pager contains content providers provided by system to get key frame information from the common

Fig. 5. Fragment structure for key frame sharing application UI.

library. These components need to cooperate with each other in order to extract and show key frames on Android platform.

To extract and share key frames, the video common library should collect key frames then display them after decoding DC image, scene change detection, key frame allocation and distribution. According to the four steps, this paper designs the key frame extraction application based on this hierarchy. In the Java layer of key frame sharing application, the relationship of function classes and context view structure is shown in Fig.

Fig. 6. Relationship of the classes and context view structure.

6. Besides the Chord classes for implementation of the key frame sharing module, we are required to make use of the Media Libraries provided in the Android SDK. It involves extending the SurfaceView class, creating a SurfaceHolder class and a MediaRecorder class, and implementing a SurfaceHolder callback function. The SurfaceView class provides a dedicated drawing surface for the picture captured, which can be displayed on the LCD screen. SurfaceHolder provides the user with the interface to control the surface size and format, to edit the pixels in the surface, and to monitor changes to the surface. SurfaceHolder callback will be called if there is any change to the surface.

We installed the application on each of these phones. User interface is shown in Fig. 7. Fig. 7(a) shows a screen shot of the user interface of the application and its menu tabs. It has a simple and clear user interface with three tap buttons in the view layout. When we touch the key frame, video is played from the position of corresponding key frame. Fig. 7(b) shows the example scenario of the application for devices connected with each other. we can support the status icon on the top right side of UI layout to conveniently check if mobile devices join and leave the group.

Key frames are delivered from a master device to a slave via HTTP server-client mechanism. Considering the bandwidth of Wi-Fi, a maximum 10 devices are connected at the same time through the wireless AP and are sharing key frames in real-time. Our key frame sharing application is based on P2P communication between mobile phones, i.e. without the use of video processing servers or network infrastructure.

5. SOFTWARE COMPATIBILITY TEST

Since there are various formats of video files, our system need to parse most of universal video formats. The formats we support are as list in table 1. So, we modify the original media player in

Fig. 7. Implemented Android key frame sharing application. (a) Screen shot of key frame display on Android phone, (b) Example service scenario of key frame sharing application

Android to support various kinds of video files listed in table 1 and then use Android API to run media scanner service, which reads meta-data from the file and adds the file to the media content provider.

We tested our system using various kinds of smartphones with different video formats, audio format and resolution. There are hundreds of devices with Android system. It is not easy to test application compatibility for all of the devices. So we choose several phones of major Android mobile phone manufacturers such as Samsung, HTC, and Google, which have different systems and hardware. We verified our application for various video formats and resolution to see if there are performance or compatibility issues using several Android phones. Test results showed that our proposed application is fully compatible for android 3.0, android 4.0 and android 4.1 and later versions. Due to the use of the standard Android develop-

ment kits, the application can be easily built on all of these mobile phones without modifications to the engine source code. So far, section 4 and 5 have illustrated all the design, implementation, and testing. Application test results on target devices confirm the validity, availability and usefulness of the proposed method. In addition, the proposed key frame sharing framework demonstrates it can provide a sufficient software platform for many Android-based P2P applications.

6. CONCLUSIONS

This paper described a prototype application in Android for sharing key frames in real-time using the Chord SDK. Using our prototype application, users are capable of creating, consuming and sharing rich media on mobile devices without complicated settings for discovering, joining and leaving

Table 1. Supported video file formats

supported video files			
video/mp4	video/3gp	video/3gpp	video/3gpp2
video/x-ms-asf	video/x-ms-wmv	video/x-ms-wma	video/divx
video/avi	video/flv	video/mkv	-

the network.

REFERENCES

- [1] Erlend Larsen, *TCP in MANET: Challenges and Solutions*, Research Report: FFI-rapport 2012/01289, 2012.
- [2] S. Kirsimäe, U. Norbistrath, G. Singer, S. Narayana Srirama, and A. Lind, "Extending Friend-to-Friend Computing to Mobile Environments," *Proceeding of International Conference on Mobile Services, Resources, and Users*, pp. 75–80, 2011.
- [3] N. Ishikawa, H. Sumino, T. Kato, J. Hjelm, S. Murakami, K. Kitagawa, and N. Saito, "Peer-to-Peer Networking Platform and Its Applications for Mobile Phones," *Mobile Peer-to-Peer Computing for Next Generation Distributed Environment*, pp. 374–396, 2009.
- [4] I. Stoica, R. Morris, D.L. Nowell, D.R. Karger, M.F. Kaashoek, F. Dalek et al., "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Trans. on Networking*, Vol. 11, No. 1, pp. 17–32, 2003.
- [5] G. Liu and J. Zhao, "Key Frame Extraction from MPEG Video Stream," *Proceeding of International Symposium on Information Processing*, pp. 423–427, 2010.
- [6] K.W. Kim and S.G. Kwon, "New Framework for Automated Extraction of Key Frames from Compressed Video," *Journal of Korea Multimedia Society*, Vol. 15, No. 6, pp. 1109–1116, 2012.
- [7] Z.L. Kis and R. Szabo, "Chord-Zip: A Chord-Ring Merger Algorithm," *IEEE Communications Letters*, Vol. 12, No. 8, pp. 605–607, 2008.
- [8] G.N. Hoang, H.N. Chan, K.N. Van, T.L. Xuan, T.N. Manh, and V.V. Thanh, "Performance Improvement of Chord Distributed Hash Table under High Churn Rate," *Proceeding of International Conference on Advanced Technologies for Communications*, pp. 191–196, 2009.
- [9] C. Jennings, B. Lowekamp, E. Rescorla, S.A. Baset, and H. Schulzrinne, *REsource LOcation And Discovery (RELOAD) Base Protocol <draft-ietf-p2psip-base-22>*, 2012.
- [10] Developer Resources for Google Android, <http://developer.android.com/>, 2014.
- [11] D. Gavalas and D. Economou, "Development Platforms for Mobile Applications," *IEEE Software*, Vol. 28, No. 1, pp. 77–86, 2011.
- [12] Samsung Mobile SDK, <http://developer.samsung.com/chord>, 2014.

Kang-Wook Kim

He received the B.S., M.S., and Ph. D. degrees in Electronics Engineering from Kyungpook National University, Korea in 1996, 1998 and 2002 respectively. He is currently a principal engineer in R&D Group, Mobile Communication Division, Samsung Electronics Co., Ltd. His research interests include visual communication, mobile embedded system, and Android application.