Preservation of Digital Media based on Embedded Context and Provenance Information

Byoung-Dai Lee[1*], Sungryeul Rhyu[2], Kyungmo Park[2] and Jaeyeon Song[2]

[1]*Department of Computer Science, Kyonggi University, Suwon 443-760, Korea*
[2]*Multimedia Global Standard Group, Samsung Electronics, Co., Ltd., Suwon 443-742, Korea*
[*]*blee@kgu.ac.kr, {suzz.rhyu, kyungmo_partk, jy_song}@kgu.ac.kr*

Abstract

Recently, with accessible media, tools, and applications, a user has been able to create multimedia files without professional knowledge. A wide variety of multimedia file formats are available to meet different requirements. From the perspective of digital preservation and version management, however, most existing multimedia file formats have several shortcomings. For instance, there is no support for recovering damaged multimedia files using their derived files or a systematic comparison of content among derived files. In this paper, we propose a method to preserve multimedia files in such a way that individual files maintain preservation information along with multimedia data to keep track of change history. In order to show the feasibility of our approach, we extend the ISO base media file format on which various well-known media formats such as MP4 are based and analyze our approach in terms of storage consumption.

Key Words*: ISO base media file format, Context and provenance information, Digital media preservation*

1. Introduction

Due to the rapid advances in information technology and the wide deployment of fast network infrastructure, the production and consumption of multimedia has been growing tremendously around the globe. As the User Generated Content (UGC) culture has proliferated, the role of the user has shifted from a passive audience member who simply consumes multimedia services, to an active and creative prosumer, who designs, edits, and publishes content to share with others. For instance, YouTube [14], a representative video-sharing website, has eight hundred million unique users a month, and roughly 60 hours of new videos are uploaded to the site every minute [15].

With accessible media, tools, and applications, a user can create media files without professional knowledge. A typical UGC production scenario for video content is shown in Figure 1. At the acquisition phase, a user acquires the base media content from various sources. For instance, a user may create homemade video, with a high quality capture device or download multimedia files from the Internet. Then, the user may edit the base content using video editing software in the editing phase. Finally, in the publishing phase, the user-created multimedia file may be hosted on the user's website, traded on a P2P network, or hosted by a video-sharing website. During the entire UGC production cycle, several

[*] Corresponding Author

multimedia files are created and used. For instance, a user might download a recorded video in 3GP [7] format from her smart phone, create a new MP4 [10] video by editing the downloaded content, and upload the MP4 file to a video-sharing website. In addition, a user often stores those files, possibly within a hierarchical folder structure, for later use or backup.

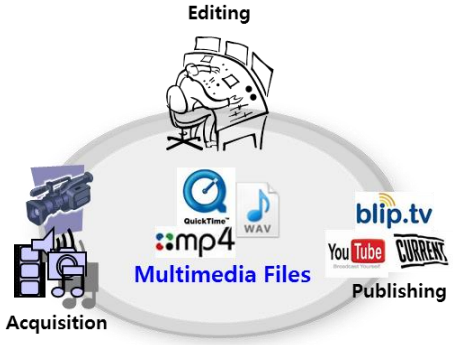

Figure 1. A Typical UGC Production Scenario

Existing multimedia file formats focus mainly on effective representation of content and efficient packaging of data. As a result, no information for media preservation, in particular context and provenance information [8], is included in the multimedia files. Context and provenance information describes the relationship of the multimedia file and/or the contents of the file to other information objects. Examples of such information include the origin of the multimedia file, any changes that have taken place since it was created, and how it relates to other multimedia files. Therefore, multimedia files without embedded context and provenance information can have several shortcomings. First, when a multimedia file is damaged, it is not possible to recover the file using those multimedia files that have been derived from it. Replicating the multimedia file to several locations is the frequently used solution to address this problem. If derived files can also be used for restoration, the preservation of multimedia files can be further enhanced by employing both strategies. Second, there is no systematic way to compare multimedia content stored in related files. For instance, in order to find differences between a multimedia file and its modified version, a user must watch both programs and spot the differences by eye.

To address the abovementioned shortcomings, we present a method of preserving multimedia files in such a way that individual files maintain metadata for preservation information based on their change history to describe their relationships to other multimedia files. Our approach is applicable to a wide range of multimedia file formats as it does not need to consider the internal semantics of individual formats due to its bitstream-level manipulation of files. In order to show the feasibility of our approach, we extend the ISO base media file format [9]. The ISO base media file format defines a general structure for time-based multimedia files. Several well-known formats such as MP4 and 3GP are extended from it.

The paper is organized as follows: Section 2 presents the related work, Section 3 describes the proposed media preservation method in detail and Section 4 presents an implementation and performance analysis of our approach. Finally, Section 5 gives the conclusions.

2. Related Work

Digital preservation is a set of intentions, strategies, and activities aimed at ensuring the continuing usability of digital objects over time [1]. The OAIS reference model [8] is a conceptual framework for the long-term preservation of digital information; as such, it addresses all the major activities of an information-preserving archive in order to define a consistent and useful set of terms and concepts. Many existing digital preservation systems [1, 3, 4, 5, 13] are based on the OAIS reference model. In this paper, however, we focus on standard-based file formats to deal with media preservation and version management.

MPEG-A Professional Archival Application Format (PA-AF) [11] provides a standardized packaging format for digital files. It specifies metadata formats 1) to describe the original structure and attributes of digital files archived in a PA-AF file, 2) to describe context information related to a PA-AF file and digital files archived in it, and 3) to describe necessary information to reverse the pre-processing process applied to digital files prior to archiving them in a PA-AF file [6]. It also specifies a file format for the carriage of the metadata formats and digital files. In particular, because it assumes that the content for archiving is already in an appropriate format, PA-AF can be used to package any kind of multimedia content.

Self-contained Information Retention Format (SIRF) [12] is a logical container format for a storage subsystem appropriate for the long-term storage of digital information. A SIRF container consists of three components: a magic object; preservation objects; and a catalog. The magic object identifies whether this is a SIRF container and its version. The preservation object is a digital information object that includes the raw data to be preserved and additional embedded or linked metadata. The container may include multiple versions of preservation objects and multiple copies of each version. The catalog contains the metadata needed to make the container and its preservation objects portable into future storage systems without relying on functions external to the storage subsystem.

BagIt [2] is a hierarchical file packaging format for the storage and transfer of arbitrary digital content. It is currently in the process of becoming an IETF standard. The BagIt specification is organized around the notion of a bag. A bag is a named file system directory that contains at minimum a data directory (called the payload), and metadata files (called tags). The payload contains any arrangement of files or folders, whereas the tags are metadata files that are intended to facilitate and document the storage and transfer of the bag. The tags include information such as the listing of the payload files and corresponding checksums, and the organization transferring the content

Packaging is the common characteristic of the abovementioned approaches. That is, different versions of a media file are packaged into a container and metadata is used to provide information to locate and relate individual media files within the container, along with other information required for media preservation. In particular, metadata is present as an independent entity from the stored media files. This approach is flexible in that it requires no knowledge of the internal semantics of the media files to be preserved. For the same reason, only limited sets of relationship information are provided.

The core concept of our approach is to provide as metadata the change history between a source media file and a modified version of it without considering the internal format of the file. By applying this, we believe that existing packaging formats can provide richer information sets.

3. Digital Media Preservation based on Change History

In order to provide capabilities to recover a damaged multimedia file using its derived files and to support systematic comparison of multimedia content among derived files, the approach that we have taken is that each multimedia file contains its change history and the minimum data to reproduce the source file from which the multimedia file is derived. The integral parts of our approach are as follows: 1) Multimedia files are compared at the bitstream-level and the differences are maintained as change history. Therefore, there is no need to consider the internal semantics of individual multimedia file formats. 2) Each multimedia file contains, as preservation information, only the direct relationship to its source file that it has modified. Therefore, the file size can be significantly reduced. Other related multimedia files can be restored by incremental restoration. In other words, an older version of a multimedia file restored using a current multimedia file also contains preservation information for its source file. This process continues until the designated version of the multimedia file is restored.

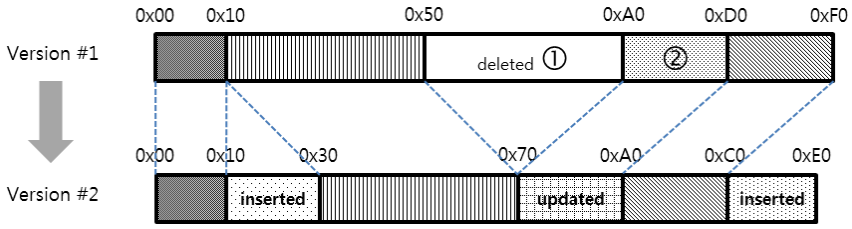

Figure 2. An Example of Bitstream-level Comparison of Multimedia Files

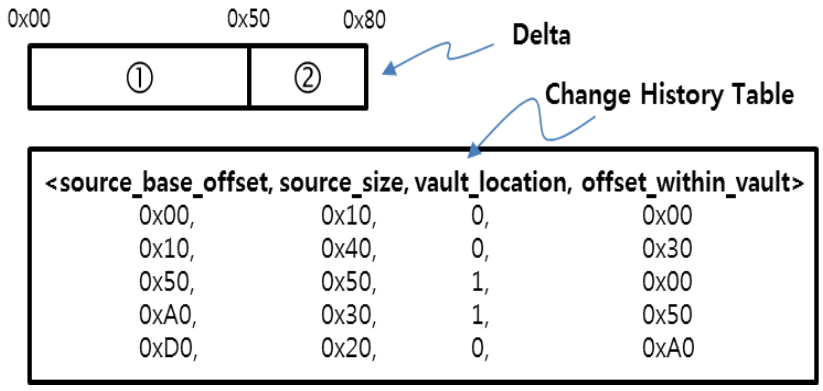

Figure 3. Metadata for Preserving Change History

Figure 2 illustrates the differences in bitstream-level between a multimedia file (*e.g.*, Version #1) and a modified version (*e.g.*, Version #2). For instance, the content of Version #1 corresponding to the block from file offset 0x50 to file offset 0xA0 is not present in Version #2 due to deletion. Similarly, the content from file offsets 0xA0 to 0xD0 in Version #1 has been updated and is located in the block from 0x70 to 0xA0 in Version #2. The blocks from offsets 0x10 to 0x30 and from 0xC0 to 0xE0 are newly inserted into Version #2.

Figure 3 shows the important metadata about the change history and minimum media data, which we call the *delta*, to be embedded into Version #2. The delta contains media data that is required to reproduce Version #1 from Version #2. Therefore, the blocks that are not present in Version #2 due to deletion or update are stored in the delta. The change history contains four different types of information required to restore the source media file: "source_base_offset" and "source_size" represent the file offset and size of the data block starting from the offset in the source multimedia file; "vault_location" and "offset_within_vault" represent where bitstream data should be retrieved to restore the data block indicated by source_base_offset and source_size. If the contents of the data blocks of Version #1 were not modified, then those blocks would be present in Version #2, potentially with different locations. On the other hand, if the data blocks of Version #1 were deleted or modified, those blocks would not be present in Version #2; therefore, they must be stored in the delta. For instance, the second row in the change history tells that the data block corresponding to file offsets 0x30 to 0x70 of Version #2 must be used to restore the data block corresponding to 0x10 to 0x50 of Version #1. The fourth row of the change history tells that the data block corresponding to block offsets 0x50 to 0x80 in the delta must be used to restore the data block corresponding to file offsets 0xA0 to 0xD0 of Version #1.

4. Implementation and Analysis

4.1. Overview of ISO Base Media File

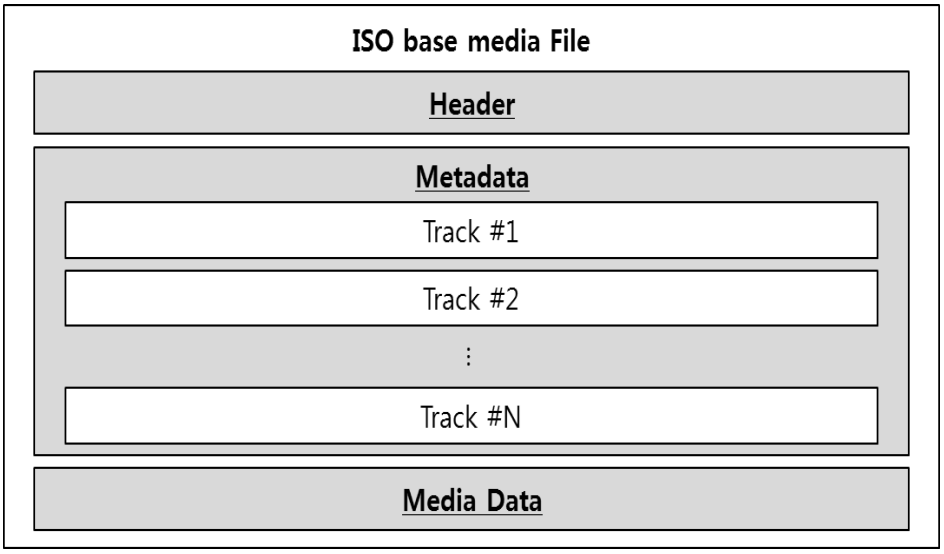

Figure 4. A Logical Structure of ISO Base Media File Format

In order to implement our approach, we extended the ISO base media file format. The ISO base media file format was specified as ISO/IEC 14496 (MPEG Part 12). It defines a general structure for time-based multimedia files, such as audio and video, that facilitates interchange, management, editing, and presentation of the media. Due to its flexibility and extensibility,

the ISO base media file is used as the basis for other formats in the family, such as MP4 and 3GP. As depicted in Figure 4, the ISO base media file format consists of three logical components: header, metadata, and media data. The header contains the general information for the media contained in the file. Examples of the information are a content identifier, content provider, and the creation date of the content. If the media file consists of multiple tracks, each of which represents a timed sequences of media (*e.g.*, frames of video), the header also contains the track configuration information. The primary information contained in metadata includes the information about the placement and timing of the media components and the profile information required for decoding the components. A media component, called samples, represents the timed unit within each track; it might be a frame of video or audio. Media data contains the actual media data. It may be in the same file or in other files.

Files conforming to the ISO base media file format are formed as a series of objects, called *boxes*, which are defined by a unique type identifier and length. All data is contained in the boxes and there is no other data within the file. For instance, individual tracks in Fig. 4 are represented by the track boxes. As a container box, the track box only contains several sub boxes for storing the track header information, the layout of the media data represented by the track, and the time ordering of the media. The sub-boxes, in turn, may contain their own sub-boxes, if necessary.

4.2. Extension of the ISO Base Media File Format for Media Preservation

Figure 5 shows the preservation-related boxes proposed in the paper: **prsv** and **pdat**, and their relationship to existing boxes. The prsv box is located at the top-level and contains pdat as its sub-box. Table 1 shows the detailed syntaxes for both boxes. Both boxes conform to the box format specified by ISO/IEC 14496 standard. Explanations of the data types used in the paper can be found in [9]. The primary role of the prsv box is to indicate whether the current multimedia file contains preservation information for its source file, whereas the pdat box contains the actual data for the change history and the delta required to restore the source file. Figure 6 illustrates an example of the ISO base media file after preservation information has been embedded.

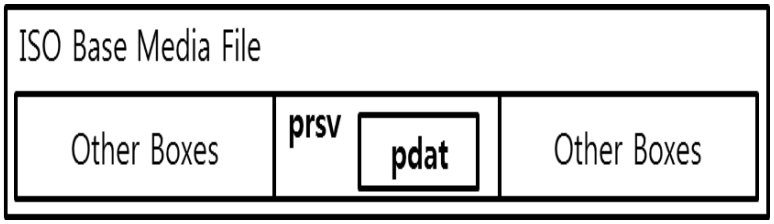

Figure 5. Logical View of the ISO Base Media File Extension

Table 1. Detailed syntaxes for proposed boxes

Box	Syntax

prsv
 	
(Preservation Box)
	aligned(8) class Preservation extends Box ('prsv') {
	  unsigned int(32) preservation_date;
	  bit(1) is_dirty;
	  if (is_dirty = '1')
	  PreservationData()
	}

pdat
 	
(Preservation Data Box)
	aligned(8) class PreservationData extends Box ('pdat') {
	  bit(8) delta[];
	  unsigned int(4) offset_size;
	  unsigned int(4) length_size;
	  unsigned int(16) change_history_entry_count;
	  for (int i = 0; i < change_history_entry_count; ++i) {
	  unsigned int(offset_size*8) source_base_offset;
	  unsigned int(length_size*8) source_size;
	  unsigned int(8) vault_location;
	  unsigned int(offset_size*8) offset_within_valut;
	  }
	}

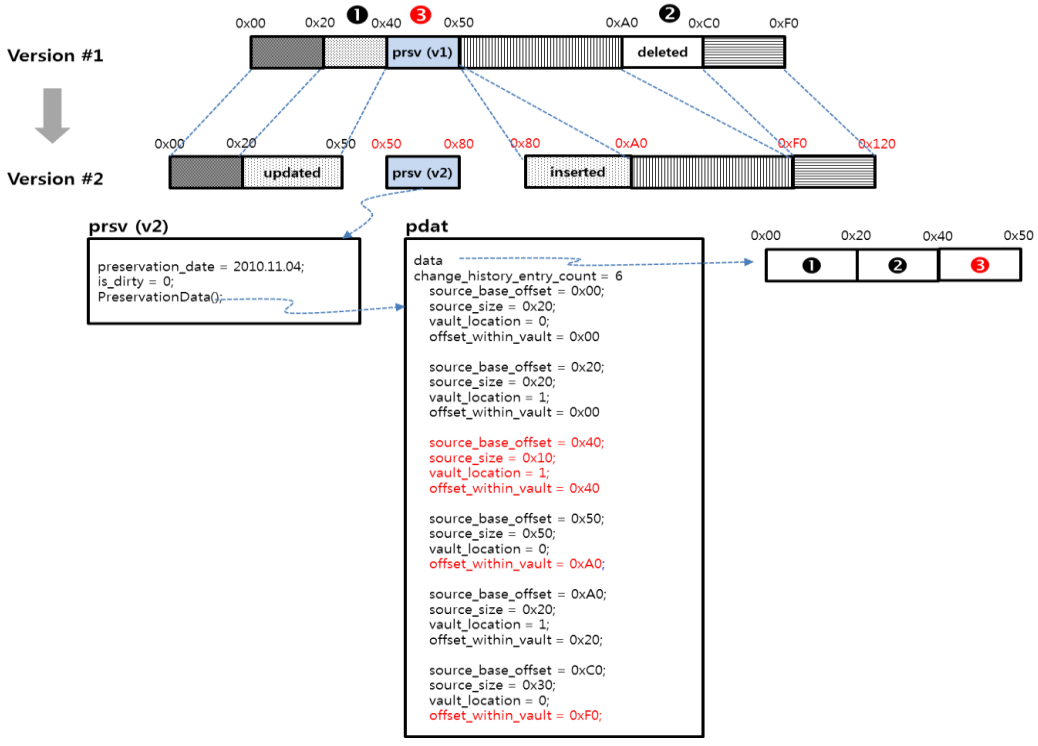

Figure 6. An Example of an ISO Base Media File with Preservation Information

4.3. Analysis

Suppose that $f_{no\_preservation}$ and $f_{preservation}$ are ISO base media files that were created by modifying a source media file, $f_{original}$, and, therefore, $f_{no\_preservation}$ and $f_{preservation}$ have the same multimedia content. In addition, let $f_{no\_preservation}$ be a typical ISO base media file, whereas $f_{no\_preservation}$ contains the preservation information proposed in this paper. Then, the file size of $f_{preservation}$ (*e.g.*, $|f_{preservation}|$) can be expressed as follows:

$$\left|f_{preservation}\right| = \left|f_{no\_preservation}\right| + \left|box_{prsv}\right| + \left|box_{pdat}\right| \quad (4.1)$$

where $|box_{prsv}|$ and $|box_{pdat}|$ represent the sizes of the prsv and the pdat boxes, respectively. Note that the size of the prsv box is constant, whereas that of the pdat box is variable in that the size of the change history table and the size of the delta determine the size of the pdat box. Thus, (4.1) can be further decomposed into (4.2).

$$\left|f_{preservation}\right| = \left|f_{no\_preservation}\right| + change\_history\_entry\_count * \left|entry\right| + \left|delta\right| + \alpha \quad (4.2)$$

where $|entry|$ represents the size of an entry in the change history table, and $|delta|$ is the size of the delta field. α is the sum of sizes of all constant fields defined in the prsv and pdat boxes. Typically, $|entry|$ and α are less than 200 bits long. Figure 7 shows the performance of our approach in terms of storage consumption. For experiments, we used a video file of 500MB size. As shown in (4.2), the factors that determine the size of $f_{preservation}$ are the number of entries in the change history table and the size of deleted or modified parts of $f_{original}$. Therefore, we measured the size of $f_{preservation}$ while varying these two factors. Table 2 shows the configurations used for the experiments. Note that for simplicity, we assume that the size of $f_{no\_preservation}$ is the same as that of $f_{original}$.

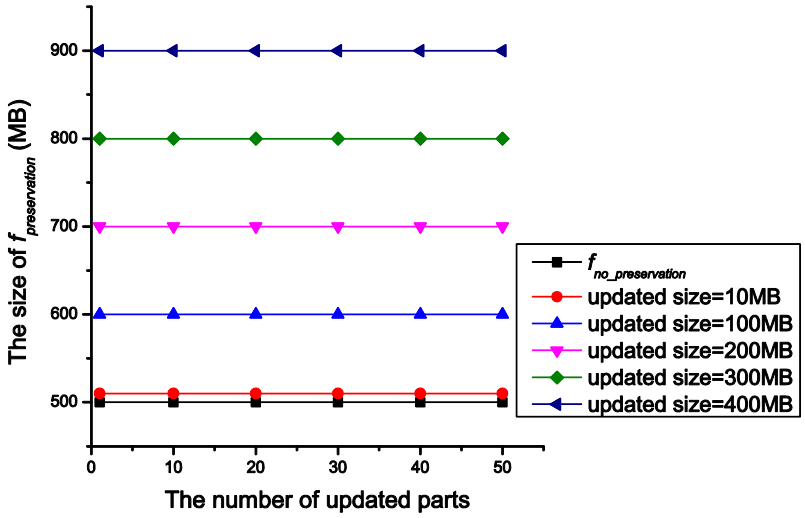

Figure 7. Storage consumption

Table 2. Configurations for experiments

Parameters	Values
The size of $f_{original}$	500MB
The size of $f_{no\_preservation}$	500MB
The number of updated parts	1, 10, 20, 30, 40, 50
The total size of updated parts (e.g., delta)	10MB, 100MB, 200MB, 300MB, 400MB

As shown in Figure 7, for each of the updated size (*e.g.*, delta), the sizes of $f_{preservation}$ are virtually identical, regardless of the number of updated parts in $f_{original}$. According to (4.2), when the size of updated parts is fixed, the number of entries in the change history is the major factor that determines the size of $f_{preservation}$. However, because the size of an entry in the table is negligible, the number of updated parts does not significantly affect the size of $f_{preservation}$. Figure 8 shows the impact of the number of updated parts in the source file to the size of the preservation-related boxes (*e.g.*, the prsv and the pdat boxes). For this experiment, we fixed the number of updated parts as 50. This figure shows that the size of delta and the box size are almost the same and, therefore, it backs up the result shown in Figure 7.

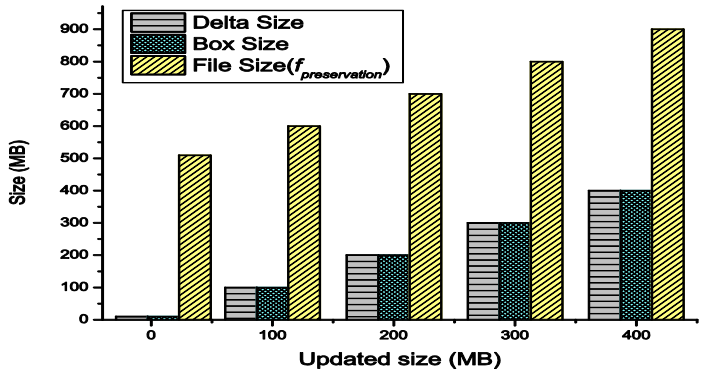

Figure 8. The impact of the size of delta

Figure 9 illustrates the overheads incurred by embedding preservation information. We define the overhead as the ratio of the preservation information to the file size (*e.g.*, $\left(\left|box_{prsv}\right| + \left|box_{pdat}\right|\right) / \left|f_{preservation}\right|$). As demonstrated in Figure 8, the size of the preservation information is determined primarily by the size of delta. Therefore, as the size of updated parts increases, the overhead due to the preservation information also increases. This phenomenon suggests that reduction in the size of the delta by applying compression techniques is strongly required to minimize the overhead, thus allowing widespread use of this technique.

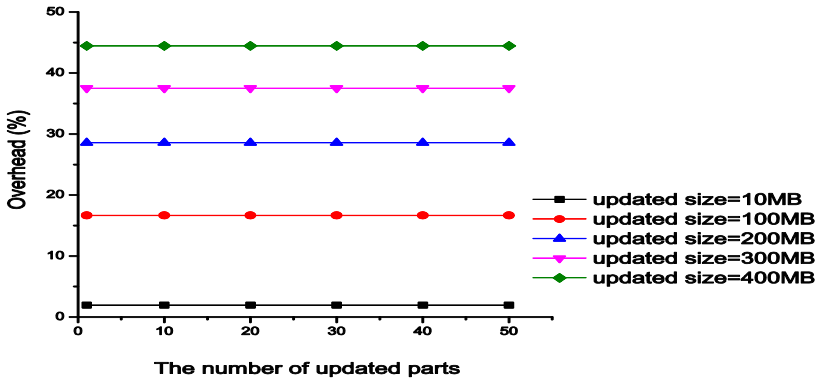

Figure 9. Overheads of Preservation Information

5. Conclusions

Thanks to accessible media, tools, and applications, users can now create multimedia files without professional knowledge. A wide variety of multimedia file formats are available to meet different requirements. Existing multimedia file formats focus mainly on effective representation of content and efficient packaging of data. From the perspective of digital preservation and version management, however, most existing media formats have several shortcomings. For instance, there is no support for recovering a damaged multimedia file using its derived files or for the systematic comparison of content among derived files. In this paper, we proposed a change history based multimedia file preservation technique. Due to bitstream-level manipulation, our approach does not require knowledge of the internal semantics of a file format. Furthermore, due to incremental restoration, multimedia files are able to provide preservation information with little overhead.

Acknowledgments

This research was supported by Samsung Electronics, Co., Ltd. (No. 2011-0278).

References

[1] S. Abrams, S. Morrissey and T. Cramer, "What? So What?: The Next Generation JHOVE2 Architecture for Format-Ware Characterization", International Journal of Digital Curation, vol. 3, no. 4, **(2009)**.
[2] A. Boyko, J. Kunze, J. Littman, L. Madden and B. Varga, "The BagIt File Packaging Format", http://www.ietf.org/internet-draft/draft-kunze-bagit-07.txt, **(2012)**.
[3] P. Caplan, "DAITSS, an OAIS-based Preservation Repository", 2010 Roadmap for Digital Preservation Interoperability Framework Workshop, **(2010)**.
[4] M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, P. Reshef and J. Satran, "Preservation DataStores: Architecture for Preservation Aware Store", 24th IEEE International Conference on Mass Storage Systems and Technologies, San Diego, USA, **(2007)**, pp. 3-15.
[5] D. Giaretta, "CASPAR Overview", 2nd International Digital Curation Conference, Glasgow, Scotland, **(2007)**.
[6] N. Harada, Y. Kamamoto, T. Moriya, Hendry, H. Sabirin and K. Munchurl, "Archive and Preservation of Media Content Using MPEG-A", IEEE Multimedia, vol. 17, no. 4, **(2010)**, pp. 94-99.
[7] ETSI 3GPP TS 26.244, Transparent End-to-End Packet Switched Streaming Service (PSS): 3GPP File Format (3GP), The 3rd Generation Partnership Project (3GPP), **(2009)**.
[8] ISO 14721:2003, Pink Book, Issue 1.1, CCSDS 650.0-P-1.1, Reference Model for an Open Archival Information System (OAIS), CCSDS, **(2009)**.
[9] ISO/IEC 14496-12:2008(E), Information Technology – Coding of Audio-Visual Objects – Part 12: ISO Base

Media File Format, ISO/IEC, **(2008)**.
[10] ISO/IEC 14496-14, Information Technology – Coding of Audio-Visual Objects – Part 14: MP4 File Format, ISO/IEC, **(2003)**.
[11] ISO/IEC 23000-6:2009(E), Information Technology – Multimedia Application Format (MPEG-A) – Professional Archival Application Format, ISO/IEC, **(2009)**.
[12] Self-contained Information Retentioon Format (SIRF) – Use Cases and Functional Requirements, Storage Networking Industry Association (SNIA), **(2010)**.
[13] R. Tansley, S. Morrissey and T. Cramer, "DSpace as an Open Archival Information System: Current Status and Future Directions", Lecture Notes in Computer Science, vol. 2769, **(2003)**, pp. 446-460.
[14] Youtube, http://www.youtube.com.
[15] Wikipedia, http://wikipedia.org/wiki/YouTube.