

A Method and System for Consolidating Multimedia Object Management in Heterogeneous Media Systems

Bjorn Landfeldt

*School of Information Technologies and
School of Electrical and Information Engineering
University of Sydney
NSW 2006 Australia
bjornl@it.usyd.edu.au*

Abstract

Today's multimedia systems use different types of media with different play-out devices. Currently we are heading towards a situation where for example cellular content providers, the home entertainment industry and networked multimedia over the Internet provide different content and different formats that are incompatible with devices from the other systems. This is in stark contrast with the current trend in the distributed systems and Internet communities where consolidation and interoperation of heterogeneous technologies is a prime target. If the vision of a future integration of different multimedia capable devices should become reality, we must consolidate the usage of multimedia objects and enable each player to use the different formats. This paper suggests a method and system to achieve such consolidation. We present our initial work and prototyping and aim at stirring a debate around the system to stimulate interest and future research.

1. Introduction

Multimedia delivery is becoming increasingly common over networked systems, following the trend of deployment of broadband access in both the corporate and residential markets. Today, it is becoming commonplace with activities such as listening to Internet radio stations and to supplement text and still images on web pages with audio and video content. In contrast to consumer product systems, the Internet is by nature a general-purpose enabler for any application or media format. This fundamental property of the Internet creates a new challenge in the form of "content management" or "media management" due to the inherent diversity of the media formats available.

The challenge lies in creating, distributing, obtaining, updating and installing decoders for all the available media in an integrated fashion that does not require special knowledge or actions from the users. Thus, if a user wishes to access a multimedia object from the Internet the appropriate decoder should be readily available for on-demand insertion into the user's selected

application and hence enable the play out of the object in case the decoder is unavailable on the target machine.

The current best practice is for the operating system manufacturer to maintain a database of available codecs that can be downloaded on-demand and installed as a shared resource from the operating system to all multimedia applications [1]. This technique is used by Microsoft Windows operating systems where codecs are given unique so called FOURCC codes [2]. The encoded media files include the FOURCC code so the player application can request the corresponding decoder module from the operating system during play out. Even though this approach is a step in the right direction there are two serious drawbacks, which limit its usefulness. Firstly, it requires network connectivity in order to download and install codecs on the fly and secondly it limits the set of available codecs to the offering from the central repository.

It is a clear trend that consumer devices such as DVD players, personal audio devices and stereos are capable of playing out multiple formats rather than being single format devices. These devices have very different architectures and therefore if the above solution should be used, each device would need individual codec repositories with associated maintenance. In addition, many such devices do not have continuous network connectivity but are instead synchronised occasionally for media file updates. For example, personal MP3 players are normally synchronised with a PC and modern car stereos can play CDs with MP3 or PCM encoded media.

In this paper we present an alternative method for managing the distribution of codecs, which overcomes the above mentioned problems and enables media devices to become general purpose players of any given media object regardless of encoding type. The paper aims at starting a debate of the implications of the solution and to stimulate further work in the area. We present a system and the fundamental building blocks and also outline potential applications and benefits. We also propose future research areas, primarily in optimisations of the system.

2. System Overview

In order to overcome the problems associated with distributing and managing codecs we propose to extend media objects with additional information about how to reconstruct a decoder for the object. In this way, media objects become self-contained and will not require access to external resources. Instead, any given media player will be able to decode the media format description and from there be able to reconstruct the decoder.

2.1. CDL

The key component in the proposed method is a platform independent description language that is used to describe codecs. Using such a language relocates the complexity of describing different codecs to the codec developer rather than the play out device. Instead, the device simply has to be able to understand a single format and to insert the code at runtime. This process is conceptually very similar to that of JAVA inserting code into a virtual machine at runtime.

The main problem with this technique is how to represent various codecs in a single format. Different codecs make use of very similar building blocks such as A/D conversion, filtering according to a psycho acoustic or visual model and compression. The fundamental operations in these steps are also very similar including matrix operations, building of dictionaries for compression etc. This points to the possibility to parameterise each step and to construct a language with standardised operations that accepts different parameters as input. However, our research shows that even though the fundamental components are very similar, the variations in the usage of these components and the formats of the parameters make such a solution if not impossible then at least very hard to realise.

The main reason for devising a parameter-based solution is the potential small size representation of a given codec in the header of a media object. Increased size of this representation translates to larger overheads when transferring files across networks, which in turn leads to lower network utilisation. However, as discussed below, the potential gain of such a parameterisation has to be viewed in the context of the size of the media part of the objects together with other sources of overhead in communication networks such as protocol headers.

There is no scientific way of determining what the size of a codec should be. Rather, the size depends on the coding style and efficiency of the code made by the software developer. However, we can get an indication of the order of magnitude of codec sizes by looking at currently available popular codecs. Below is a table of a number of codecs available for the Microsoft Windows platform.

Codec	Size (kb)	Audio	Video
Microsoft RLE	10		x
Microsoft video1	25		x
Radius Cinepac	108		x
Microsoft H261	180		x
Intel Indeo	194		x
Microsoft MPEG-4	248		x
Microsoft H263	280		x
Indeo Video 5.10	737		x
DSP Group True Speech	8	x	
Microsoft G711	9	x	
IMA ADPCM	15	x	
Microsoft GSM6.10	20	x	
HP Mobile Voice	56	x	
Microsoft G723	116	x	
Fraunhofer MPEG-Layer3	284	x	
Windows Media Audio Codec	288	x	

Table 1. Sizes of commonly used codecs in MS Windows

In some applications such as video playback from a disk or long time streaming audio from a web-cast station, these sizes are relatively minute to the size of the media, which in turn makes the overhead very small. However, other scenarios such as streaming media over cellular networks can show different results. In order to illustrate the overhead the system would introduce, we present the following examples:

- A short video clip of 2 minutes length
- A news video broadcast of 15 minutes length
- An audio conversation of 1 minute length
- An audio file of a song of 3 minutes length

Table 2 and Table 3 illustrate the overhead of each codec included in the media file.

Codec	Size (kb)	Video 2 size (kb)	Overhead (%)	Video 15 size (kb)	Overhead (%)
Radius Cinepac	108	18000	0.6	135000	0.08
Microsoft H261 (128kbps)	180	1920	9.3	14400	1.25
Intel Indeo	194	18000	1.1	135000	0.14
Microsoft MPEG-4	248	4000	6.2	30000	0.82
Microsoft H263 (16kbps)	280	240	117	1800	15
Microsoft H263 (64kbps)	280	960	29	7200	3.9

Microsoft H263 (256kbps)	280	3840	7.3	28800	1
--------------------------	-----	------	-----	-------	---

Table 2. Overhead of using video codecs together with media objects

Codec	Size (kb)	Audio 1 size (kb)	Overhead (%)	Audio 3 size (kb)	Overhead (%)
Microsoft G711	9	480	1.9	1440	0.6
IMA ADPCM (32 kbps)	15	240	6.3	720	2.1
Microsoft GSM6.10	20	97.5	21	292.5	6.8
Microsoft G723	116	300	38.7	900	12.9
Fraunhofer MPEG-Layer3 (128 kbps)	284	960	29.6	2880	9.9
WM Audio Codec (32 kbps)	288	240	120	720	40

Table 3. Overhead of using audio codecs together with media objects

From the tables we can see that in many cases the overheads are very small, but also in some cases they are very large. There are two ways of tackling the cases where the overhead is large. Firstly, content producers can select codecs depending on the situation so that there is a match between codec size and overall file size. Secondly, codecs can be cached on terminals so they do not have to be sent over the network with each individual file as described below in section 2.2.

We have defined a Codec independent Description Language (CDL). We base our language on the ISO C99 specification [3] since this language allows implementation of any given codec, past, present or future.

In order to make the codec size as small as possible we have identified which part of the ISO C-99 standard that is needed to realise codecs and which part can be removed. We then place this limited library of C-functions on the terminal and only include the compiled source code with references to dynamically linked libraries with the media file. The terminal in turn implements a runtime environment, which links the codec and libraries to recreate the final decoder.

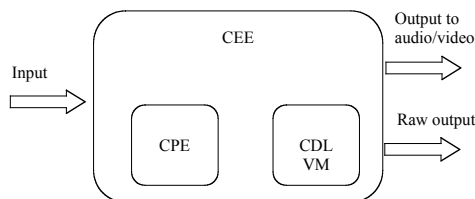


Figure 1. The CEE runtime environment

The proposed system is shown in Figure1. The runtime environment consists of three components, a Codec Execution Environment (CEE), a Codec Protocol Engine (CPE) and a Virtual Machine (VM). The CPE manages signalling for detection of already cached decoder modules as described in section 2.2. The VM accepts the codec description from the media object, links it with the libraries and executes the resulting decoder on the data part of the media object. The CEE finally coordinates the operation of the CPE and the VM. There are two types of output from the CEE, directly to an audio/video card/circuit or a raw format that can be used as standard input to an encoder as described in section 3.

2.2. OMP, Object Management Protocol

In order to minimise the overhead associated with downloading the codec description for each object we propose to use a caching scheme for codec modules. The scheme operates by allowing a terminal to query the content server or equivalent for the codec used in a media object. This simple and well-understood mechanism comprise two components, identification of codec modules and a signalling protocol.

The identification scheme can be realised in different ways, one of which is the usage of the current FOURCC codes. The main concern is the ability to uniquely identify different versions of different codecs so there is no possibility of a mismatch between a cached codec and the media object.

In order to allow for easy integration into various systems we propose to use a description format called Object Management Protocol in conjunction with an external carrier protocol borrowed from the network system. An analogy of this method is the usage of the Session Description Protocol SDP together with the Session Initiation Protocol SIP [4].

We propose to use two fields in OMP, the identifier of the codec (FOURCC) coupled with a classification code. The classification code is used for terminals to make a decision of whether or not to store the module locally in case the cache space is running low. In our current implementation we use the following classification; Audio (low, medium high bit rate), Video(low medium high bit rate). However, we acknowledge that this rudimentary classification might not be the best option and we intend to investigate what the best scheme is in future research. The classification can be very useful for terminals with limited storage space. For example, it is more likely that cellular terminals will use low bit rate codecs than high bit rate codecs and therefore these codecs should be given preference if a selection has to be made.

3. Applications

In this section we outline some of the potential applications of the proposed system and point to some of the advantages it brings.

Firstly, as mentioned before, the system will allow networked computing devices to use any media object regardless of the encoded format. This means that there is no need to develop different codecs for different platforms and to maintain repositories of these different codecs. In addition, networked devices can use stored media objects even if network connectivity is temporarily broken for example if an object is downloaded over the network but used at a later stage when network connectivity is unavailable.

Secondly, the system allows the vast market of consumer products such as CD players and TV sets to use any given media format. This in turn means that these products will become generic media players rather than dedicated to playing a few formats and in addition, will never have to be upgraded to be able to use new formats as they emerge. This in turn enables TV sets to become true interfaces to the Internet, clock radios to use Internet streaming media etc.

Thirdly, the generic design enables the CEE to operate as a universal adapter between different media formats. Since the output from the CEE is a common format regardless of the input format it is possible to direct the output into a secondary stage encoder resulting in a N-to-1 generic transcoder module. Previous research [5, 6, 7] has shown that using such a function could be very advantageous in tailoring the media to suit the operational environment, for example by reducing the resolution of the media to suit a terminal with a low bit rate network access.

There is a potentially very important role to play for this function in cellular networks. In these networks one of the critical issues is the limited spectrum and therefore also the spectrum efficiency. All current standards use specific codecs and have optimised the offered traffic classes (bearer services) to suit this codec. In the general case, the bitrates and delay requirements of other codecs will not suit the offered bearer services and therefore either the spectrum utilisation or the quality of the service will suffer when other codecs are used.

Using the universal plugin by concatenating the raw output from the CEE with an encoder of the specific codec of the network will remove this sub-optimisation and tailor each media format to fully comply with the optimal reservation classes. The universal adapter would significantly reduce the complexity of such a transcoding platform rather than using a combination of decoders and encoders.

4. Hardware implementation

It would be of great advantage to implement the CEE in hardware for many of the possible applications above. IC based solutions would bring large volumes, which in turn would push the price of components down to a level where a plethora of simple devices could be connected to networks and take advantage of the unmatched offering of media that the Internet provides. One of the key aspects of such a solution is that a

hardware-based solution does not require the same level of operating system complexity as a software based solution which can make devices cheaper.

Even though we have not yet worked on the realisation of the decoder in hardware, there are strong indications that it is feasible to realise. It is now possible to buy hardware implementations of the JAVA VM, which has to support a larger set of operations than the CDL VM [8, 9].

A key aspect is that typically a hardware based solution is far more energy efficient than a software based solution. This is a very important aspect for battery-operated devices such as PDAs, cellular phones and music players. If such a device is used to play different multimedia formats, the hardware based approach would potentially extend the battery lifetime significantly. Work has already been done on how to optimise the hardware JAVA VM to preserve battery lifetime as much as possible [10].

5. Discussion

In this paper, we have introduced a novel method and system for consolidating different multimedia formats to increase the usefulness of media and to simplify its management. It is imperative for the success of distributed systems that the availability of information and media is accessible to the widest possible audience and set of terminals and to this end; our proposal provides an enabling technology.

We have implemented a demonstration system at the University of Sydney. We chose to base the system on JAVA since it enabled us to make use of already developed virtual machines. The lessons we have learned so far are that it is possible to realise such a system, but care has to be taken in order to avoid relatively large overheads. A simple caching scheme can help alleviating this problem, but care has also to be taken not to introduce large signalling delays.

The potential advantages of using this system are many, but we are cautious about quantifying the benefits before further research has been taken in a number of areas.

Firstly, the most pressing issues lie in efficient realisation of the runtime environment for the modules. There is a need to investigate the most optimum configuration of compilation and linking steps taken in order to minimise delay when reconstructing the decoder as well as maximising power efficiency. Secondly, it is important to investigate how the signalling protocol, CPP, can be incorporated into existing carrier protocols such as SIP, and how this would affect session set-up delays. Thirdly, it is not evident how to optimally represent the VM in hardware. It is a challenging problem to make the realisation as energy efficient as possible to maximise the battery lifetime of battery-operated terminals. Fourthly, security will be a primary concern where this system would suffer from weaknesses similar to those encountered in JAVA VMs.

The CEE must be able to cope with malign code so that the system cannot be compromised. Finally, the system needs to be able to handle other functions such as digital rights management. In our initial work we have concentrated on the composition of media decoding functions only, and extending CDL with DRM functions brings new challenges.

This paper aims at initiating a debate around the potential benefits and pitfalls of using the proposed system. We acknowledge that the success of such a system depends on the collective research efforts of a large scientific community rather than single researchers. Through such an effort, we envisage that the system can become a part of our every day lives and help improving the way we use multimedia content.

References

- [1] Mark D. Pesce, "Programming Microsoft DirectShow for Digital Video and Television", Barnes & Noble, ISBN 0-7356-1821-6, 2003.
- [2] <http://www.microsoft.com/whdc/hwdev/archive/devdes/fourcc.msp#XSLTsection124121120120>
- [3] ISO/IEC 9899:1999 – Programming Languages – C Specification
- [4] SIP: Session Initiation Protocol. J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, RFC 3261, June 2002.
- [5] Fox A, Gribble SD, Chawathe Y, Brewer EA. Adapting to network and client variation using active proxies: lessons and perspectives. IEEE Personal Communications, August 1998.
- [6] Amir, McCanne S, Katz R. An active service framework and its application to real-time multimedia transcoding. In: Proceedings of ACM SIGCOMM 98, 1998.
- [7] "MARCH: a distributed content adaptation architecture", S. Ardon, P. Gunningberg, B. Landfeldt, Y. Ismailov, M. Portmann, A. Seneviratne, Wiley International Journal of Communication Systems special issue on Wireless Access to the Global Internet: Mobile Radio Networks and Satellite Systems, vol. 16, issue 1 pp 97-115, 2003.
- [8] <http://www.arm.com/products/solutions/Jazelle.html>
- [9] Yang, B. S., Moon, S. M., Park, S., Lee, J., Lee, S., Park, J., Chung, Y. C., Kim, S., Ebcioğlu, K., and Altman, E. 1999. Latte: A java vm just-in-time compiler with fast and efficient register allocation. In Proceedings of International Conference on Parallel Architectures and Compilation Techniques. 128--138.
- [10] G. Chen et al., "Tuning garbage collection for reducing memory system energy in an embedded java environment", ACM Transactions on Embedded Computing Systems (TECS), pp. 27 – 55, Vol. 1, Issue 1, November 2002