Intelligent Distributed Multimedia Collection: Content Aggregation and Integration

Jelle Nelis, Dieter Verslype, Chris Develder
Ghent University – IBBT
Dept. of Information Technology – IBCN
Ghent, Belgium
Email: Jelle.Nelis@INTEC.UGent.be

***Abstract*—People's multimedia content is spread around their home network and content services on the Internet, such as YouTube, Flickr, Facebook. In this paper we present a system that aggregates all the multimedia content of the end user and integrates it into a unified collection for the user's convenience. The system provides location transparency of multimedia content, content filtering on player compatibility and metadata completion to aid in improved usability. This effectively enables the user to rediscover his multimedia collection without any technical knowledge.**

A proof-of-concept implementation known as Intelligent Distributed Multimedia Collection (IDMC) has been made that is able to detect and browse UPnP MediaServer devices as well as collect information from YouTube. This implementation also contains a media player and is able to control UPnP MediaRenderer devices remotely. Furthermore, performance has been measured to assess different ways of iterating through a multimedia collection.

I. Introduction

Today, multimedia is everywhere. Network Attached Storage devices become commonplace, Internet services enabling users to store their multimedia content for sharing or remote viewing purposes are popping up. There is a tendency to distribute a personal multimedia collection, be it due to a lack of organisational skills, limited storage capacity or functionality demand. This means that end users are unable to fully benefit from their own multimedia collection: several usability problems exist. A multimedia collection is difficult to organize and easily resorts in chaos, such that users are unable to locate their desired content because they are unaware of the device they stored it on. In addition, they are unaware of codecs or resolutions; they just want it to work on all devices in their home.

The Digital Living Network Alliance (DLNA) [1], [2] tried to solve common use cases like consuming multimedia in the home by defining device classes. Devices can deliver (Digital Media Server), consume (Digital Media Renderer and Digital Media Player) or control (Digital Media Controller) multimedia. Part of this effort is certifying devices against these device classes. The underlying technology used by DLNA is UPnP [3] of which they use the AV specification [4] to solve the use cases regarding multimedia consumption.

However, the scope of devices considered here cannot be limited to this one technology, several other technologies exist that can deliver content within the home network and from the Internet.

In [5], a system comparable to the one presented in this paper is discussed. However, only UPnP MediaServer devices are considered as possible data providers and the integration of similar content items used in [5] is limited to integration based on local metadata. This paper adds technology independence (beyond UPnP) and more clever metadata integration, such as using Internet services to enhance the gathered metadata.

In [6], an architecture is presented that solves location transparency of multimedia content by implementing a virtual MediaServer that redirects all requests to the MediaServer devices that actually contain the content item. Not only does our current work present a system that performs more integration functionality in terms of content aggregation, such as metadata completion, it also supports the concept brought forward in [6] as a pluggable view of the system (see Section III).

The goal of this paper is to present an architecture for an intelligent system to manage a distributed multimedia collection that gives users a complete, personalized view of the huge pile of available multimedia content. The system is able to easily add new multimedia sources. Examples are online multimedia sources like YouTube, Flickr, Facebook, etc. and locally stored multimedia found using different protocols. The system performs duplicate detection, merges metadata of resources deemed duplicate and completes content metadata using external sources (e.g. Discogs for audio content).

Most importantly, it offers the multimedia collection in such a way that the user can consume their multimedia without any technical knowledge. One example is that when a multimedia item is not supported by a particular device due to a codec incompatibility, the system takes care of this: incompatible content items will not be shown or an appropriate transcoding action will be taken. The paper is structured as follows, Section II discusses the architecture solving the problems put forward in this section. In Section III, a proof-of-concept implementation is presented on which a performance assessment of different ways of aggregating multimedia collections was performed. Results of these tests are reported in Section IV.

II. Architecture

The problems discussed in Section I have been addressed in a student project called Intelligent Distributed Multimedia

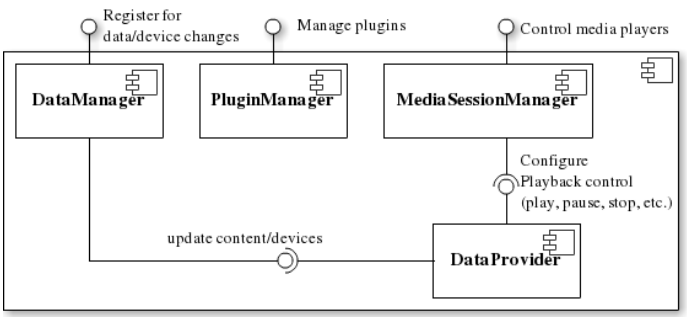

Fig. 1. High level IDMC architecture

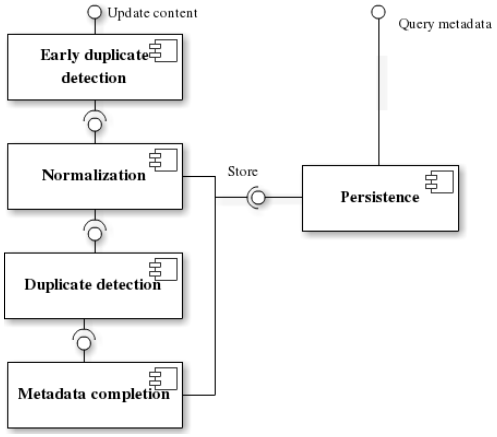

Fig. 2. DataManager component

Collection, in this paper referred to as IDMC.

A component diagram of the most important part of the IDMC architecture can be seen in Fig. 1. The major driver behind the architecture is runtime addition of data providers such as providers of local content available through UPnP and remote content such as YouTube. Therefore we work with pluggable data providers that can detect new and updated content and signal it to the DataManager component. These data providers will be managed by the PluginManager component. The DataManager component is responsible for all actions performed on newly discovered multimedia metadata.

The external interfaces at the top of Fig. 1 can be used by whichever user interface will be built on top of it. In this paper we will however, not focus on the user interfaces, but rather on the DataManager component. In Fig. 2, the further decomposition of the DataManager component is shown. Essentially, it is a pipeline that filters the metadata added to the system. The pipeline is designed in such a way that filters can be added or removed easily so future improvements can be added without a hassle.

Currently, the algorithm performs the following steps: for performance reasons, a first quick check is performed to filter duplicates based on easily comparable parameters like content location. This makes sense since we identified in [5] that current UPnP MediaServer implementations offer their multimedia metadata in a number of different ways, which

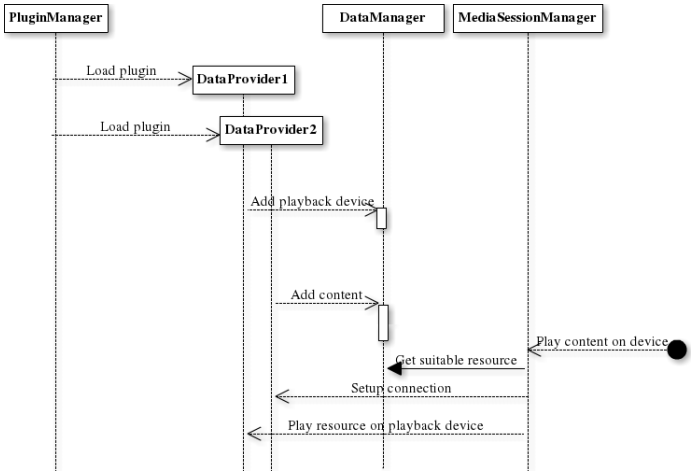

Fig. 3. Simplified sequence diagram

means that even within the same collection, the same content will be listed more than once and as such, there will be duplicates that are easy to filter.

The Normalization component makes sure the metadata gets cleaned so it is ready for further processing. After normalization, a second stage duplicate detection is performed. This component is also responsible for cross referencing metadata, i.e. suppose a resource on the network can be identified to be part of the same content item, then the metadata of the current content item can enrich the metadata of the previously added content item. As the last step in the algorithm, external sources are used to get more information about the content item being processed.

The Persistence component is responsible for storing the listing of the unified library and thus contains the metadata of the aggregated multimedia collection. Every filter in the pipeline can search for and change metadata in the Persistence component while processing. Furthermore, it exposes a search interface for user interfaces to provide a view on the multimedia collection.

Fig. 3 shows the simplified interactions between IDMC components to offer the required functionality. Everything starts with the PluginManager component loading data provider plugins. These plugins come in two flavors, DataProvider1 in Fig. 3 is able to detect and control playback devices while DataProvider2 can detect content on some medium (e.g. UPnP MediaServer, Flickr, etc.). The DataManager components keeps a list of playback devices and upon receival of content from DataProvider2 will perform the integration steps mentioned above.

After these steps, the MediaSessionManager component is able to use this information to guide the user in consuming his/her multimedia content. Content items that are not compatible with the media player the user wants to play content on, will automatically be filtered. This can be done by using the functionality of the DataManager component.

After the user has selected which item he/she wants to play, the MediaSessionManager component will setup a connection

between the storage device and the playback device. This is done using the best possible resource (in terms of resolution, codec, streaming protocol, etc.) as found by the DataManager component. Differences in device control protocols are abstracted by the data provider plugins, which are ultimately responsible for the actual control.

III. Proof-of-Concept

A demonstration has been implemented of the architecture discussed in Section II. As discussed previously, the architecture foresees pluggable data providers. To demonstrate this, two plugins were developed, a UPnP plugin and a YouTube plugin. The UPnP plugin detects UPnP MediaServer devices and browses the collection it exposes, the YouTube plugin imports the favorites of a given user account.

The demonstration environment consisted of a home network with several networked devices. It included a Sony PlayStation 3 (Media Player), a Sony Bravia TV (Media Renderer), a Windows-PC and a Linux-PC (both MediaServer). There is a subtle difference between a Media Renderer and a Media Player: a Media Player is able to play content through its own user interface while a Media Renderer has the same functionality as a Media Player, but can also be remote controlled via the network.

Two user interfaces have been developed, each of which provided a different view on the aggregated multimedia collection and the system. A native graphical user interface was developed to be able to browse the collection on your local computer. This user interface was able to remote control the playback functions of the Sony Bravia TV through its MediaRenderer interface. It also included plugin management functionality to be able to enable support for, in this case, YouTube at runtime. This demonstrates the flexibility of the architecture in supporting different content retrieval technologies.

Since having an aggregated multimedia collection locally does not help interoperability with legacy devices, another user interface was developed. A UPnP MediaServer was written to give Media Players, in this case the Sony PlayStation 3, the possibility to use the functionality provided by the demonstrated system. This way the Media Player was able to browse through the unified collection without changing the software client side.

IV. Performance measurement

Retrieval of multimedia metadata can be done in several ways. A data provider conceptually presents its data as a file system which can be represented as a graph. Fig. 4 presents an example of such a representation. Leaf nodes represent files while non-leaf nodes represent containers. Containers typically contain similar content items. An often used structure is having a distinction on the first level between the different types of multimedia, such as audio, video and pictures. Within each of those containers several different sublevels can be found, e.g. artist followed by album for audio items.

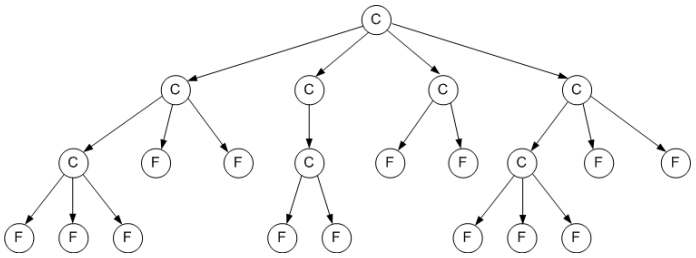

Fig. 4. Multimedia content tree, C = container, F = file

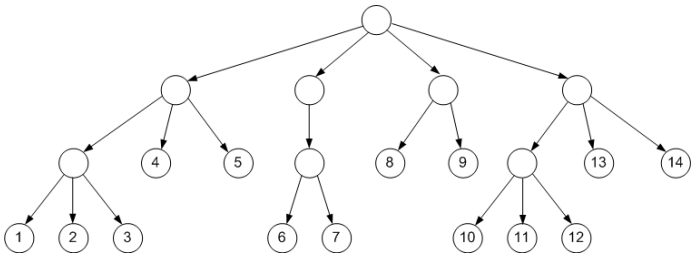

Fig. 5. Per-leaf aggregation

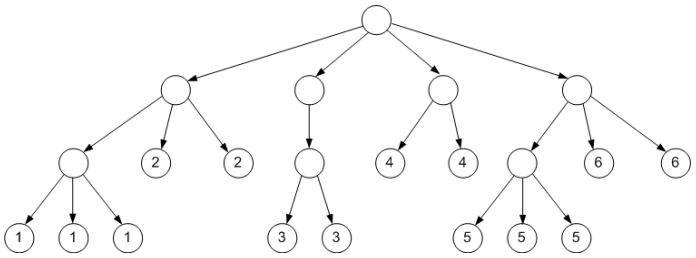

Fig. 6. Per-container aggregation

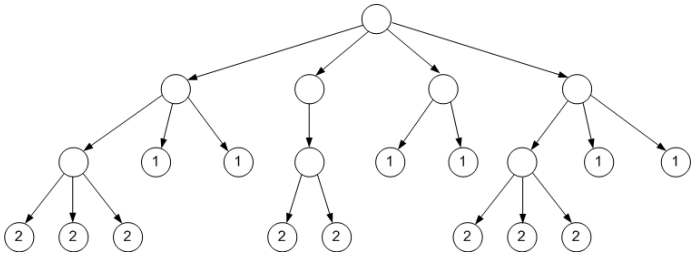

Fig. 7. Per-level aggregation

In our performance assessment, measurements compared aggregation on a per-leaf basis or by enabling batching of aggregation requests. In Fig. 5 it is shown how per-leaf aggregation is performed: every leaf node gets pushed through the pipeline right after discovering it.

A first batching scheme is shown in Fig. 6, leaf nodes get cached until a new container is seen. The next possibility is to cache all content items that reside on the same level in the content item tree, this batching scheme can be seen in Fig. 7. Lastly, it is possible to cache every content item that is present on a device on the network, this just means all leaf nodes will be added to the data manager at once.

Tests were performed using a dummy implementation of

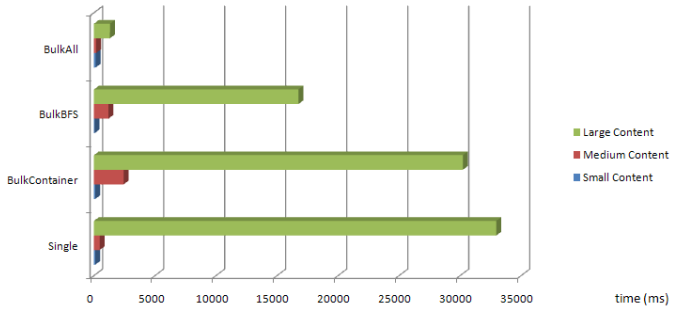

Fig. 8. Persistence performance

a UPnP MediaServer to serve multimedia content metadata. Three different multimedia collections were used:

- small
  - 10 audio items
- medium
  - 100 audio items
  - 10 images
  - 1 video
- large
  - 1000 audio items
  - 200 images
  - 20 videos

For each collection, the four different techniques discussed before were tested. The time to browse the collection (without network delay), as well as the time to parse the returned DIDL-Lite structures [7] were measured. However, these times were negligible when compared to the time needed to persist the aggregated items. In Fig. 8 the time measurements for persisting the different multimedia collections are shown. Several conclusions can be drawn from this graph, first of all, from the perspective of the DataManager component, it is clear that as much items as possible should be added at the same time. With regard to the size of the multimedia collection, the method to add a single item at a time, proves not to be scalable since its execution time explodes for the large collection. It is clear that adding content items in bulk outperforms individual additions.

A disadvantage of batching aggregation requests is the need for more temporary memory. Furthermore, adding content items earlier might improve user perceived performance since preliminary results will be visible more quickly.

V. Conclusions

In this paper we presented a system that gives an aggregated view of all the content a user owns including content stored in the cloud. This is achieved by allowing pluggable data providers to add content to the system. A UPnP MediaServer plugin was developed to show local content discovery and a YouTube plugin was developed to show the user's multimedia content on the Internet can be added to his unified collection.

When multimedia content gets added to the system, it will pass a pipeline in order to get integrated in the unified collection. This pipeline removes duplicates, merges the metadata of the content item being added with existing metadata and performs metadata completion by using external information sources.

Performance measurements were performed on the system as a whole using different strategies to add content items. This shows that some kind of batching technique needs to be used since adding content items one at a time becomes needlessly slow when reaching large collections (over 1000 items).

VI. Future work

The system discussed in this paper acted as a proof-of-concept to show it is possible to provide a non-trivial service to end users using devices in the user's home network and services on the Internet. It succeeded to do so. However, it still has some shortcomings. It is tailored to one specific use case, namely the aggregation and integration of a user's multimedia content. Currently, we are working to make the system more generic so as to be able to interact with a variety of different device types independent of the technology spoken by the device/service. Obviously, the use case presented in this paper should still be possible using the more generic approach, but additional benefits arrive when it is possible to use new device/service types. True integration of the complete home network is possible, state changes of devices can act as triggers for user specified actions and applications can take into account the current context.

Acknowledgment

The authors would like to thank the students Jeroen De Meyst, Jeroen De Ridder, Sam Govaert, Thijs Mergaert, Niels Nuyttens, Daan Raman, Thomas Roelens, Xavier Smet and Pieter Van Lysebetten for their effort in designing, implementing and testing the system discussed in this paper.

References

[1] IEC, *IEC 62481-1 ed1.0: Digital living network alliance (DLNA) home networked device interoperability guidelines - Part 1: Architecture and protocols*, 2007.

[2] ——, *IEC 62481-2 ed1.0: Digital living network alliance (DLNA) home networked device interoperability guidelines - Part 2: DLNA media formats*, 2007.

[3] A. Presser *et al.*, "UPnP Device Architecture," 15 Oct. 2008. [Online]. Available: http://www.upnp.org/resources/documents.asp

[4] J. Ritchie, T. Kühnel, J. Kang, and W. van der Beek, "UPnP AV Architecture:1," 30 Sep. 2008. [Online]. Available: http://www.upnp.org/specs/av/default.asp

[5] K. Mets, J. Nelis, D. Verslype, P. Leroux, W. Haerick, F. De Turck, and C. Develder, "Design of a context aware multimedia management system for home environments," in *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD '09. Computation World:*, 2009, pp. 49 –54.

[6] J. Park and S. Kim, "A transparent contents sharing service with virtual media server," in *Convergence Information Technology, 2007. International Conference on*, 2007, pp. 764 –767.

[7] "XML Schema for ContentDirectory:3 Structure and Metadata (DIDL_Lite)," UPnP Forum, September 2008, latest version: http://www.upnp.org/schemas/av/didl-lite-v2.xsd.