# INDUSTRIAL AUDIO FINGERPRINTING DISTRIBUTED SYSTEM WITH CORBA AND WEB SERVICES

Jose P. G. Mahedero, Vadim Tarasov, Eloi Batlle, Enric Guaus, Jaume Masip
Audiovisual Institute
Universitat Pompeu Fabra
Ocata 1, Barcelona, Spain
{jpgarcia, vtarasov, eloi, eguaus, jmasip }@iua.upf.es

## ABSTRACT

With digital technologies, music content providers face serious challenges to protect their rights. Due to the widespread nature of music sources, it is very difficult to centralize the audio management. Audio fingerprinting allows the identification of audio content regardless of the audio format and without the need of additional metadata. Monitoring the audio being broadcasted by the TV and radio stations of a country requires the design and implementation of a scalable, robust and modular framework. We have chosen CORBA as distributed environment. The whole functionality needs to be decoupled from clients. To do so, Web services have been deployed. The audio identification core uses a Hidden Markov Model-based audio fingerprinting technology. The paper discusses the design and implementation issues of a complete distributing system that automatically monitors audio content, specifically music and commercials.Today, a working prototype of such a system already exists, and is dedicated to monitoring several radio and tv stations in Spain.

## 1. INTRODUCTION

Due to the outgrowing field of digital rights management (DRM), the need to automate the tracking of different audio sources arose. Traditionally this task was accomplished by sampling the sources (or just asking them) and storing what had been broadcasted in a certain period of time. At the time the only sources of audio which companies kept track of were radio stations.

Time went by and some factors showed the need for companies to automate the tracking process. Among these factors we could mention the growth of the music rights management companies and their significance, the increasing number of radio stations, and one of the most important (if not the most) the arrival of the Internet and the resulting exponential growth of music transfer.

As the number of sources and the quantity of audio itself was really huge, it became clear that a distributed system supporting many pluggable sources was an optimal solution.

The system needed to be powerful enough to serve different clients [1] each of them having its own requirements. It also had to provide efficient load balancing and fault tolerance. The experience in distributed systems pointed the use of CORBA (http://www.CORBA.org) as a robust platform to integrate and cooperate many heterogeneous systems [5].

## 2. MAIN DESCRIPTION

We have implemented a system composed by several nodes in charge of the audio recognition. They communicate in a hierarchical structure in wich each node tries to recognize audio. In case a node is unable to identify a pattern, it passes the request to the next node of the hierarchy. The identification procedure continues until it reaches the top level of the hierarchy.

Nodes act as caches of the top level node. Each of them has a list of patterns representing a portion of all the models present in the system, so a node will identify a portion of what the whole the system is able to. If the audio fragment is not identified by the top level node, the audio passes to manual (human) processing. See figure 1

Each identification transaction (either positive or negative) is logged in order to provide feedback information to the clients. They can also monitorize the behavior of the system and adjust global parameters depending on their needs.



[1] We will use the term *client* to refer to a program that will communicate with the system as well as the end user of the system. This non distinction is made because from the point of view of the system these terms (program client and user client) are similar and the system makes no difference.

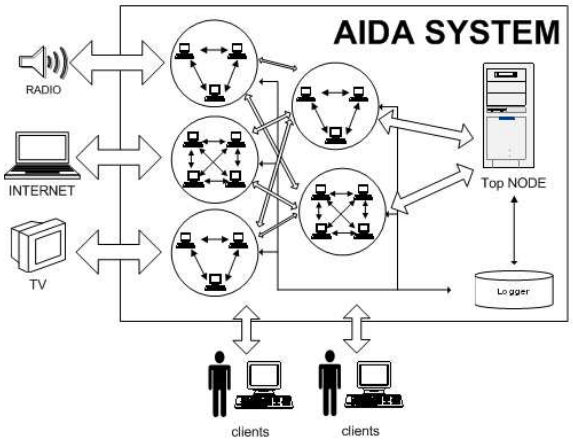


**Figure 1**. main description

From a logical point of view, our system is composed of different layers each of them with its own purpose:

**Identification** : the core of the system. This layer is in charge of the identification of the audio. It takes audio streams as input and tries to identify them using the audio patterns datase. It consists of C++ audio recognition libraries.

**Distribution** : groups physical units into *virtual nodes* that cooperate for the same purpose, for example commercials, music or TV virtual node. It manages the load of each virtual node and is in charge of distributing jobs to the nodes depending on their respective load. This layer sends jobs to the *identification layer* and loggs the results received from it.

This layer also manages each virtual node's pattern list to avoid overloading the top node by providing lower level nodes with feedback information from higher level nodes. This process results in an updated pattern database for each node. Thus, a lower level node will not fail to identify any given pattern more than once. At the same time with load balancing, this layer is responsible for error recovery and handle system failures. CORBA is used to provide seamless integration between remote components.

**Intelligence** : it provides mechanisms for prioritizing jobs either on demand by client request or to satisfy the *distribution layer* needs. It also executes jobs periodically or can even estimate processing times to give clients a better feedback. As well as the *distribution layer*, this one uses CORBA as a platform for process distribution and coordination.

**Integration** : a layer in charge of making the system accessible to many different clients each of them having its own operating system and requirements. This is accomplished by using web services, which provide a functional API of the system.

**User interface** : the use of web services lets up to client the implementation of the end interface, so this part is not covered in this paper.

## 3. IDENTIFICATION LAYER

System overview The system presented here goes beyond the template matching paradigm to a statistical pattern matching paradigm. The goals are to incorporate robustness to the system by statistically modeling the audio evolution while reducing the size of the fingerprint by considering local and global perceptual redundancies in a corpus of music data.

Feature extraction A filter-bank based analysis approximates the human inner ear behavior and the Mel-cepstrum coefficients (MFCC) [2] analise the music timbers [4].

Channel estimation If the distorting channel $\mathcal{H}(\omega)$ is slowly varying we can design a filter that, applied to the time sequence of parameters, is able to minimize the effects of the channel. The filter we designed for our system is $H(z) = 0.99\frac{1-z^{-1}}{1-0.98z^{-1}}$.

Hidden Markov Model Observers

Polyphonic music can be seen as a sequences of simultaneous acoustic events (notes). In [1], we defined a set of abstract events that allow a mathematical description of complex music as sequences of events. These events are captured by Hidden Markov Models(HMM)[6] that model abstract audio generators. (trained with the Expectation-Maximization algorithm [3])

Identification Algorithm

Each HMM fingerprint in the song database uniquely identifies each song among the others. The fingerprint database is generated using the Viterbi algorithm [8]. The identification algorithm matches an input streaming audio against all the fingerprints to determine whenever a song section has been detected. The Viterbi algorithm is used again with the purpose of exploiting the observation capabilities of the HMM models contained in the fingerprint sequences.

## 4. DISTRIBUTION LAYER

This layer has many tasks to accomplish: *group nodes*: virtual nodes represent a hardware abstraction layer that increases system scalability. *Assure a good load balancing*: in a distributed system it is essential to make a good load balancing in order to have an almost constant throughput. This is also a task of this layer. *Fault tolerance*: a node may be disconnected during processing without any consequences. *Feedback clients and nodes*: clients need to receive the results of the recognition processes as well

as to monitorize the global behaviour of the system. As we mentioned in the *Main description section* (2), nodes are fed back from the top node to prevent future identification failures.

This layer is composed of *three different levels* (see figure2):

- Physical level: each node is seen as a single processing unit.

- Logical unit level: hardware resources are grouped together into virtual nodes. This level divides identification tasks depending on which kind of audio it will identify. For example we can have three units for commercial audio identification grouped together into one virtual node. Other example could be several logical units hosted on the same physical machine. The number of physical units that form each virtual node can be configured to fit the system needs.

- Process level: this is composed of virtual nodes and a logger. Each virtual node is built of *three elements*:*Data providers*: provide data from heterogeneous sources in a unified manner. *Recognizer*: identifies the audio supplied by the data providers. *Controller*: handles communication and synchronization issues between data providers and recognizer. It also communicates with other controllers to perform load balancing and error recovery. In case of identification failure it is charge of passing the identification task to the top level node.

Apart from virtual nodes there is a special component named *logger* who is in charge of logging all the events received from the controllers of the virtual nodes. It writes logs to the databases. They can be later consulted by clients by calling the services provided by the *Integration layer*. Logs reflect information about the recognition process such as timing, which node processed the audio, kind of audio processed etc. This information help clients to track the whole identification process.

## 5. INTELLIGENCE LAYER

The Intelligence layer serves for the process optimization.It is separated into several intelligent agents that perform *automatic stream tagging* based on metadata. The system preprocesses the stream description, separates the fragments with voice from those with music. It also performs *source signal cache management*. The source signal is cached and arranged in manageable fragments allowing to minimize the number of queries made to external data storage system, thus minimizing the traffic. This layer also provides identification result optimization, automatic job management, automatic pattern list generation and pattern reindexation.

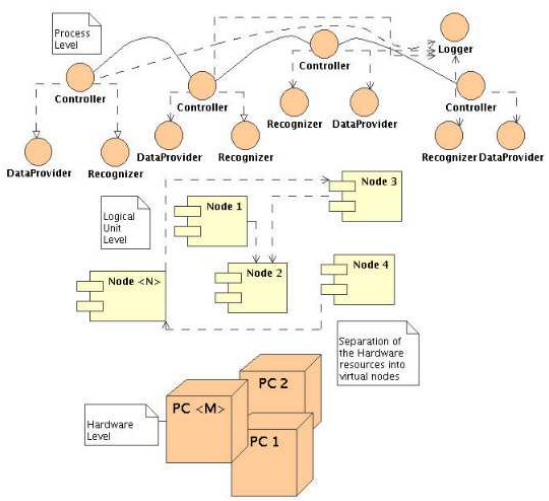


**Figure 2**. Distribution layer

## 6. INTEGRATION LAYER

This layer integrates modules provided by the distribution (see 4 Distribution) and the Intelligence (see 5 Intelligence) layer with the clients. At the same time is lets server-client interaction as much open as possible by providing a functional API.
*SOAP* (http://www.w3.org/TR/soap/) is a lightweight protocol based on XML-RPC calls. It is designed to work in a decentralized environment. It sits on top of the protocol stack and can use many different transport protocols such as SMTP, FTP or other. However its most widely used over HTTP because of its facilities for firewall/proxy filtering.[7]

From a general point of view, working with SOAP consists on making requests to a Web Service (from now on *WS* or simply *service*) in a specific SOAP format, and receive responses from the WS. The Service interface is described in a WSDL (http://www.w3.org/TR/wsdl, a superset of XML) file indicating methods calls, objects, and exceptions that will be sent across the net. A Web Service is a set of related methods exposed to clients via SOAP protocol. As all the exchange of information is made with XML (both data and control messages), SOAP makes possible the interaction between different platforms or languages such as COM, CORBA, Perl, Tcl, the Java-language, C, C++, Python, or PHP programs running anywhere on the Internet.

*Features of Web Services*: Applications can communicate across the Internet, they are language and platform independent, they can run over many protocols, and they are human readable.

Due to their features, WS suit perfectly our needs. The aims of the layer are accomplished by using WS because they act as a bridge between clients and modules in the underlying layers which use a CORBA infrastructure. By

providing a fine grained functional API, WS define the way clients communicate with the system, but don't impose any restriction on their implementation. In a MVC pattern this would mean a complete separation of the *View* from *Model* and *Controller* and force clients only to follow SOAP encoding rules but not to use a specific language or end user interface.

### 6.1. implementation

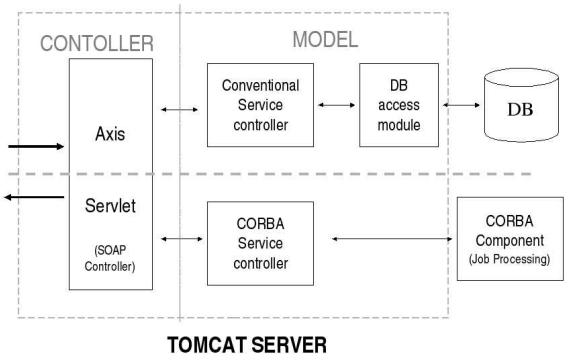


**Figure 3**. Integration layer organization

This layer is built on different modules (see figure 3):

- *Axis*: Axis (http://ws.apache.org/axis/) is the name of the implementation of the SOAP specification we use. Its is a servlet that must be plugged in to a servlet container. Its responsibility is to route SOAP messages between services and clients.
- *SOAP Server*: we use Tomcat Server as a servlet container to route SOAP requests and response to and from AXIS servlet.
- *SQL Server*: mysql is used as a persistence layer (audio model storage, system information, etc.)
- *Controllers*: there are two different controllers. *Service controllers* that manage database access modules and *CORBA controllers* that handle interactions with the underlying layers using CORBA.
- *CORBA Components*: components in charge of distribution, load balancing and audio processing. Apart from distribution tasks, they accept audio recognition jobs from their controllers and invoke *Identification layer* components.

## 7. CONCLUSIONS

The combination of Hidden Markov Models applied to fingerprinting, with CORBA and Web Services results in a robust, scalable and performant distributed framework. Hidden Markov Models provide an accurate fingerprinting technique which is made robust and scalable with the use of CORBA. The integration provided by Web Services lets as much open as possible the interaction between clients and system functionalities.

## 8. REFERENCES


[1] E. Batlle. Automatic Song Identification in Broadcast Audio. *Proc. IASTED Signal and Image Processing Conference*, 2002.

[2] J. A. F. E. Batlle, C. Nadeu. Feature Decorrelation Methods in Speech Recognition. *Int Conf on Spoken Language Processing*, 1998.

[3] A. D. et Altri. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, January 1977.

[4] B. Logan. Mel Frequency Cepstral Coefficients for Music Modeling. *Proc. ISMIR*, 2000.

[5] S. V. Michi Henning. *Advanced CORBA programming with C++*. Addison Wesley, 1999.

[6] L. R. Rabiner. A Tutorial on HMM and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[7] E. D. Simon St Laurent, Joe Johnston. *Programming Web Services with XML-RPC*. O Reilly, 2001.

[8] A. J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Identification. *IEEE Trans. Info. Theory*, 1967.