©2010 International Journal of Computer Applications (0975 - 8887)
Volume 1 – No. 23

Mobile Agents for Audio Search & Retrieval

Mrs. Nirmala C R
Asst.Professor, CS & E, BIET
#325, Shamanur Road
Davangere-4

Dr. V Ramaswamy
Professor & Head , IS & E, BIET
#325, Shamanur Road
Davangere-4

Mrs. Jyothi N M
Lecturer, MCA, BIET
#325, Shamanur Road
Davangere-4

ABSTRACT
At present de-facto standard for providing contents in the Internet is the World Wide Web, which implements the client server technique. A technology, which is now emerging on the web, is searching for the audio, video and images and retrieval of the same. In this paper, we describe an architecture that helps the user in performing the above tasks using mobile agents. Here the system is designed and implemented for the search and retrieval of the images and audio files over the Internet. We can observe that several disadvantages of the World Wide Web's client server technology have been overcome by using this architecture. The current commercial applet – based methodologies for accessing audio files from web databases offers limited flexibility, scalability and robustness. Our system is based on aglets which are nothing but Java based mobile agents. The implementation of the architecture shows that its performance is comparable to and in some cases outperforms the current approach. As an application on this architecture we have also created the speech synthesizer Agent for the special purpose of playing an audio from a particular word. To achieve this, we are making use of open source software Free Text to Speech converter1.0.1 as a supporting tool. This architecture makes use of pull model of E-commerce. This work aims at saving the download time, reduces network traffic for the users who compare and purchase audio online.

Categories and Subject Descriptors
H. Information Systems, H.3 INFORMATION STORAGE AND RETRIEVAL, H.3.5 On-Line information Services

General Terms
Algorithms, Performance, Design, Experimentation, Verification.

Keywords
CBSR, Audio Aglet, User, Supplier, Speech Synthesizer Agent,

INTRODUCTION
Last one and a half decade has witnessed tremendous increase in the size of audio and video collections on the web. With the increase in the computational power of both hardware and software, the ability to store, search, and retrieve complex data types in databases such as video, audio and images has also improved to a large extent. These new media types offer other challenges thus demanding a different approach when compared to pure text.

The World Wide Web is rapidly being accepted as universal access mechanism for network information. The popularity of web suggests that web browsers may offer an end-user interface for a large class of applications including search and retrieval of audio files. Content based Audio Search and Retrieval, CBSR is a technique which is fast emerging on the web today.. A content based query matches examples or prototypes to known instances of a certain media type based on the measure of similarity.

The present approaches which support varieties of audio files suffer from a number of disadvantages. All audio files have to be transferred across network to some indexing process. Audio files gatherers seek files using HTTP or FTP requests. The searcher will have to wait between requests, which increases the time required for requesting, searching and receiving the files. This time is identified as a performance bottleneck. Hence an alternative architecture has been proposed for distributed indexation and searching of audio files which combine Content Based Audio File search and retrieval technology, Mobile agent technology.

The proposed architecture, called Audio-Aglet architecture utilizes the latest technology of mobile agents and content based audio search and retrieval and demonstrates its effectiveness over a specific application context (i.e. audio search and retrieval). The main advantage of the architecture is that it frees the remote client which can perform other more important tasks. The implementation of the framework shows that the performance of the system is comparable to current approaches.

Pull Model of Marketing
At present most of the e-commerce sites are designed to support business to customers (B2C) applications. Almost every B2C applications make use of pull model of marketing. This model borrows the interaction pattern between the buyers and sellers from the trading model in everyday model. For example, buyers visit sellers to purchase items that they wish to acquire. The items that are sold by the sellers are characterized by an established value in the market. This ensures that customers are likely to visit the sellers, or in other words the customers are pulled to the sellers. This is also called as Buyer Driven E-Commerce. In this implementation, the users are customers who wish to acquire the audio files on the internet. The sellers are nothing but, the sites which maintain the database of millions of audio tracks.

MOBILE AGENTS

2.1 Introduction
The term "software agents" refer to programs which perform certain tasks on behalf of the user. Software agents can be classified as stationary agents and mobile agents. Stationary

agents achieve the goal by executing on a single machine. On the other hand, mobile agents migrate from one computer to another computer and executes on several machines. Mobility increases the functionality of the mobile agents.

A mobile agent consists of the program code and the program execution state. Initially, a mobile agent resides on a computer called home machine or dispatching server. The agent is then dispatched to execute on a remote computer called mobile agent host. When a mobile agent is dispatched, its entire code and the execution state are transferred to the mobile agent host. The host provides a suitable execution environment for the mobile agent. Another feature of mobile agent is that it can be cloned to execute on several hosts. Upon completion, the mobile agent delivers the results to the sending client.

Aglet Technology developed by IBM Research Labs, Japan is a framework for programming mobile network agents in Java. IBM's mobile agent called 'Aglet', is a light weight java object. One of the main differences between aglet and Java Applets which is a simple mobile code with the itinerary that is being carried along with an aglet. By having the itinerary, aglets are capable of roaming the Internet collecting the information from many servers.

An aglet can be dispatched to any remote host that supports the Java Virtual Machine. The prerequisite for the remote host is to, pre-install Tahiti, a tiny aglet server program implemented in java and provided by Aglet framework.

2.2 Sound in Java

Java has incorporated sound functionality since release 1.3. Recording and playback as well as streaming of multiple audio formats are supported either through the Java Sound API [15] or the higher level Java Media Framework (JMF) [14]. Java Sound is a framework for recording, playing and processing sound within Java. It supports a wide range of formats, both 8 and 16 bits and sample rates from 8 to 48 kHz. Java Sound is relatively low level and is thus useful for sound processing applications. For instance, it is straightforward to record a sound and present it as a byte array, suitable for FFT or other forms of analysis. JMF is also designed for use with time-based media content such as audio and video, and it too has methods for capturing and playback. However, the emphasis is on real-time streaming and presentation of media content rather than analysis.

2.3 Audio Analysis

There are two main categories of audio analysis that could be used in a system such as this, i.e., sound matching and speech recognition.

2.3.1 Sound Matching

Sound matching is a concept where a sound is compared to known sounds to determine how similar they are. Sound matching can use both physical properties of the sound, such as amplitude and frequency, and psycho-acoustical ones, such as onset and offset. Applications of sound matching range from automatic violence detection for films [17] to automatic detection of exciting parts in sports shows [22] to surveillance systems [8]. Sound matching could be used in this system to detect sounds such as word or a phrase spoken in a speech. Much research has been made on the subject and there are some techniques that could without doubt be useful in this project.

- Spevak and Polfreman's sound spotting approach described in [21], which uses mel-frequency cepstral coefficients for feature extraction, commonly employed in speech recognition. Subsequently, self- organizing maps, a particular typeof neural networks, is used for classification and, finally, k-difference inexact matching, a string matching algorithm, is used for pattern matching.

- Pfeiffer et al's audio content analysis [17], primarily designed for automatic violence detection in films, classifies sounds using both physical properties, such as amplitude and frequency, and psycho-acoustical properties, for instance onset and offset.

- Comparisonics' commercially available sound-matching technology [20]. Since the system is commercial, its design is secret. However, its functionality is essentially equivalent to Spevak and Polfreman's method, creating signatures from audio content and comparing them to determine similarity.

2.3.2 Speech Recognition

Speech recognition can be divided into two types; dictation recognition and command recognition. Dictation recognition is designed to recognize a vast range of words and is used to quickly produce large quantities of text in a more natural way than typing. Dictation recognizes many different words. As a result, it is not very accurate. The other type of speech recognition, command recognition, is used to parse commands and can be used in a speech user interface, replacing or enhancing a traditional user interface. Command recognition merely recognizes certain pre-determined words, but with a much higher accuracy than dictation recognition.

Speech recognition is available in Java through the Java Speech API (JSAPI) [19].JSAPI defines interfaces for both speech recognition and synthesis. As with all other Java technologies, it is very high level and aims at complete platform independence. There is no standard JSAPI speech engine. Instead, it relies on third-party software to implement the API. Currently, there are a few free open source speech synthesis engines written entirely in Java. Speech recognition engines, in contrast, are normally commercial, written in native code and hence platform dependent. Examples of available JSAPI implementations are Cloud Garden's JSAPI [7], which can run on top of several different speech engines, and IBM's "Speech for Java" [8], using IBM's own ViaVoice [9].

MOBILE AGENTS FOR AUDIO SEARCH AND RETRIEVAL (MAASR)

System Description

The concept of the entire system is a distributed system that , the audio-aglet architecture is used for the purpose of search and retrieval of the audio files over the internet and web databases[4] for the purpose of purchasing audio online. The framework consists of two main systems or modules.

- User Module
- Supplier Module

The user module[1] consists of a static agent called user agent and the mobile agent called Audio Agent. The audio agent is held responsible for the search and retrieval of the audio file requested. The user module also consists of Speech Synthesizer Agent , which is a stationary agent.

The supplier module[1] consists of a static agent called supplier agent. This agent helps the audio agent to get the required information. The audio agent is coded in such a way that, it can visit any number of hosts and get the required audio files from various sites. Once the audio agent gets the audio files to host system, it stores it in a folder specified. The user is given an option to call the player and play the audio by using the player interfaced with the system.

At the client side, one another important module is developed. This is the audio synthesizer module. This module helps the user to play the audio file from the specific point. For this purpose, user agent creates a Speech Synthesizer Agent , in turn this SSA Makes use of Open Source Speech Synthesizer . Freetts 1.0.1. This application accepts the word or phrases in the form of text and coverts it into speech. Then the SSA is called for the purpose of speech matching. SSA take the speech in its primary representation of byte array and tries to match with the whole audio file for the word, if it is found, then the file is played from that point otherwise returns unsuccessful search or some error message.

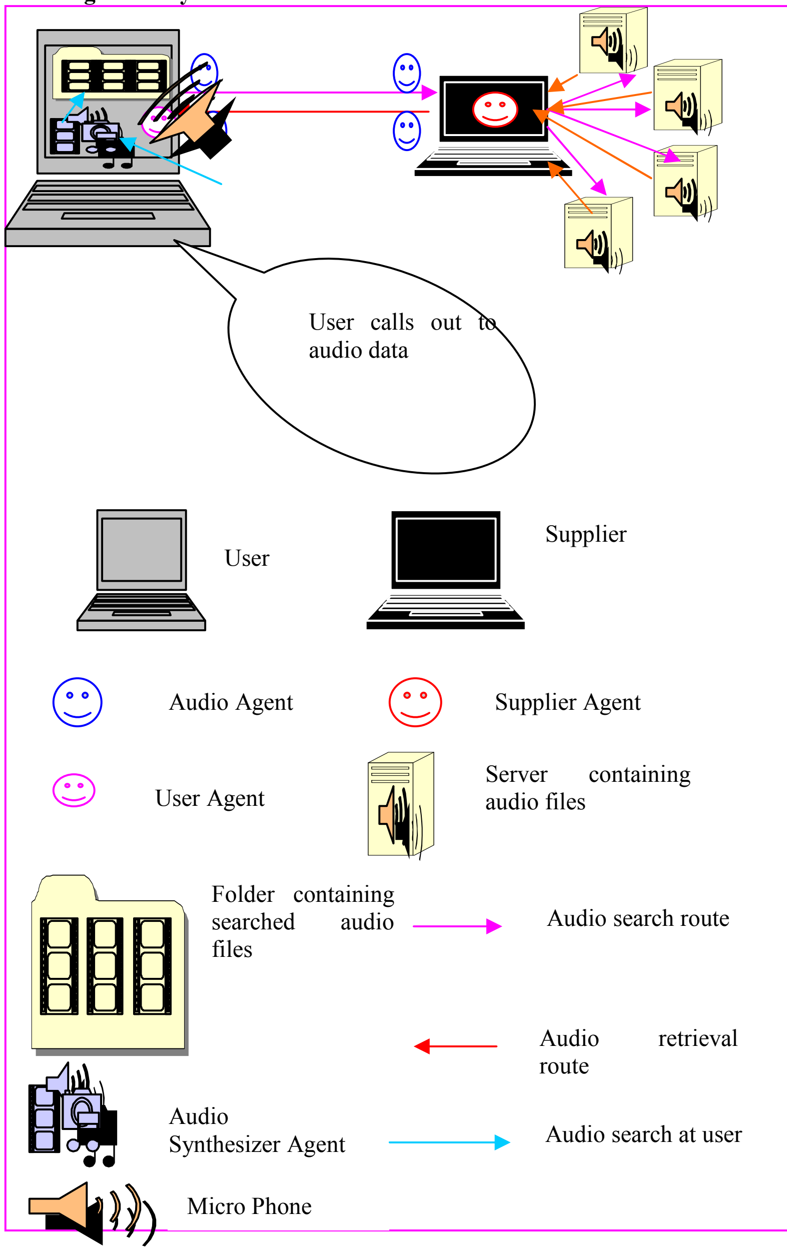
This module finds an application in scenarios such as, there is a very huge audio file which is a recorded voice of Gandhi- ji or Swami Vivekananda or any important lecture delivered by famous scientists like Dr. A P J Abdul Kalam. If the listener is interested in only part of the speech then he can start that from the specific point by specifying the name of the audio file and from which the file has to be played.

This module also finds application where in a audio files are embedded in the power point slides. Some times it is necessary to fast forward the slide to start from different point. If this module is interfaced, just typing the word from where it has to be started, sound matching will be done and the slides will be displayed from that point. These scenarios exists in the case of webinars(Seminars on the web)

The figure 3.1 Shows the different modules of the system with their respective agents. Here multiple agents are utilized to get the work done efficiently without under or over utilizing the capability of the agent.

First, the user has to log on to the system by using name and password provided during the registration. If he is not registered, he is given an option to register with the system. After he registers, he gets a login name and password. Using these login credentials, he/she can log on to the system. Once logs into the system, he invokes the user agent at the dispatching host. This user agent in turn creates the Audio Agent, which is mobile agent and dispatches to the different hosts on the web. The audio agent once reaches the predetermined hosts, it passes the request for the audio files to the static agent at the supplier module. Then the static agent in turn processes the requests on its local database as well as on the directory tree structure and retrieves the results. This result is updated to the audio-agent. The process repeats for every predetermined host and the audio-agent returns to the dispatching server with the results and dumps in the related tables at the dispatching server.

Figure3.1.System Architecture



Developmental Model

3.2.1 Implementation

Here the prototypes are created first with very basic functionalities and then extended into the more complicated system. The main advantage of this approach to the well-known waterfall model, for instance, is that, as initially knew either the full potential or the limitations of the technologies used, the design could be revised and remade as the tools became more familiar.

The first design of the prototype was very rough. Using IBM's aglet framework the basic agents are created, which just search and retrieves the text files. Later it is extended to search and retrieve the images from predetermined hosts. Next, the actual functionality of searching and getting the audio files is implemented. For the purpose of audio matching, first, the sounds like handclaps and whistle were tried.

Later the actual work of matching a specific word in an audio file is done and preferred solutions were,

a. Recording with Java Sound. Java Sound can provide sounds as a byte array that is practical for further processing.
b. Identification with external sound matching. The idea here was to send the recorded sounds as an array of bytes to a sound matching method of Speech Synthesizer Agent, which would return the similarity of the sounds.
c. The volume is used as a parameter to match the sound. And it is very easy to extract the volume of a sound.

3.2.2 Pseudo Code

The system has got three main agents namely, User Agent, Audio Agent and Supplier Agent. Following are the pseudo codes for the agents mentioned above.

User Agent:

```
//class for user agent
Import *.Aglet;
Class UserAgent  extends  Aglets {
        Public void run() {
AudioAglet( object init)
{
 create mobile Audio Agent
}
fill_itinerary( URL) {
Add destination URL's(1……n)
};
get_proxy(Aglet Id)
{
onDispatch()
{
Wait for message agent;
}
}
 Public void AudioAglet (Object init)
{
Public void OnCreation(object args)
{
add MobilityListner;
}
Public void fill_itinerary(URL)
Add predetermined destination URLs;
Public void getProxy (AgletContext.getAgletProxy(Aglet Id.Identity)
{ gets  proxy for an aglet in the current context;
}
{
```

Audio Agent:

```
        //class for Audio Mobile Agent

        Import *.Aglet;
        Class AudioAgent  extends  Aglets {
                Public void run() {
        Get the audio file list from the user to be searched atethe supplier site .
        Get the itinerary filled by the user
        Move to supplier site
        Give the file list to be searched to the Supplier agnet
        And wait for response
        If search is successful it returns with searched audio files
        Else
        Returns with "Requested audio file not available" message.
        Stop
        }
```

Supplier Agent:

```
//class for stationary supplier agent at the supplier site
//import *.aglet;
class Supplier extends Aglet{
pulic void run(){
listen for arrival of mobile audio agent from user sites onArrival of a mobile Audio aglet
for each component on the audio search list
processRequest(component)
}
public void processRequest(reqCompnentType component){
search the local audio files using XML parser
if component is found
send response to AudioAgent  about availability
else
request audio aglet for time to quote
if audio aglet agrees to wait then
invoke the search For _audio file }
```

Speech Synthesizer Agent (SSA):

```
//class for stationary Speech Synthesizer Agent at the user Module
Import  *.aglet;
Class SSA extends Aglet {
        Public void run () {
Wait for FreeTTS to give the sound of the word typed
For a given file
Public void searchWord(word) {
Search for the word in the specified audio file, by using the byte array   representation of the audio recorded. if word found,
}
Public void callPlayer(Component) {
Play the audio starting from matched word
}
Else {
Public void Unsuccessful() {
Print the error message "word not found"
}
```

Experimental Setup

To deploy and check the functionality of the prototype developed, the following LAN set up is made in the CS & E Department. The five hosts namely A, B, C, D, E which are connected via TCP/IP are considered. Each host is configured with JDK1.5 above, Aglet Software Development Kit 2.0.2 and MySql for the purpose to store the standard audio files.

Every host on the network is identified by its IP Address and default port
Host A : (User or dispatching server) IP Address is 8a828d3c:4434
Host B: (Predetermined Host) IP Address is 8a829d3c:4434
Host C: (Predetermined Host) IP Address is 8a830e1c:4434
Host D: (Predetermined Host) IP Address is 8a831c0d:4434
Host E: (Predetermined Host) IP Address is 8a832b8d:4434

Step 1: User Creates the user agent, in turn creates the Audio Agent and dispatches to supplier sites to search for specified audio files.
Step 2: After reaching the predetermined host, Audio Agent hands over the request to Supplier Agent.
Step 3: Supplier Agent searches in its local database for the specific audio file, if found updates the Audio Agent with success or failure. If Success, the bit vector present in Audio Agent is stuffed with the resulting audio files. If, Failure, the bit vector is stuffed with the error message.
Step 4: This procedure repeated for all the predefined hosts in the itinerary.
Step 5: Audio Agent comes back with the requested audio files and gives it to User Agent. The User agent stores the path in the specified table and files in the specified directory for further use .
Step 6: User can run the so collected audio files at any point of time, without getting connected to the internet. He/she can also make use of the Speech Synthesizer Agent to play the file from somewhere in the middle or so.
Step 7: Exit or logout from the System

Observations.

The following observations are made from the above developed prototype.

Two Cases are considered here.

Case I:

1. A single mobile agent is sent to all the five hosts to get the audio files and the total time is calculated by noting down the dispatch time and the arrival time (in milliseconds) to get the images. The Dispatch time is Td and Arrival Time is Ta. Round Trip time is calculated using the following expression.

   $$Rtt(ms)MA = Ta-Td.$$

2. Irrespective the type and size of the of the data that it brings, the time taken was approximately same.
3. It resulted in empty result set , when the size of the file increased beyond the size specified in the mobile agent to bring the information.

Case II:

1. Simple timer program is written and is executed every time the HTTP request is made to different host.
2. The timings are recorded for different type of data like audio, image and text. Here the request time and the receive time (Rt and Rc Respectively) are noted down.
3. The total time taken is calculated by using the following expression

   $$Rtt(ms)HP = Rc-Rt$$

For example, in order to bring two images or two audio files residing at two distinct locations, the user has to make two separate requests and the RttHP will vary depending upon the speed, bandwidth of network connection. where as a mobile agent is capable of bringing both the files for a single request by going to all the predefined hosts in its itinerary. This saves the time to a greater extent.

4. CONCLUSION

Automation of global production and distribution through e-commerce technology offers the opportunity for greatly increased efficiency and responsiveness to changing the buyers preferences and advances in technology. The challenge is to make it work. Using mobile agents for search and retrieval, buyer or user can check out for prices of audio tracks at different vendor sites and can decide on the best price to buy. Irrespective of the type and size of the file, the mean retrieval time is approximately same. Speech synthesizer Agent gives ease of use in playing the audio file from a particular point significantly applied to webinars. The user can enhance search speed by involving multiple mobile agents. This is proved in our previous work. The limitations of this work are, the agents can search for the information only on Tahiti enabled hosts and security threats from the hosts and the network. Final conclusion that could be drawn from this work is, performance is comparable to and in some cases outperforms the current approach.

5. REFERENCES

1) P.Dasgupta, N Narasimhan, L.Moser, and P.M. Mellier Smith," MAgNET: Mobile Agents for Networked Electronic Trading", IEEE Transactions on Knowledge and Data Engineering, Special Issue on Web Technologies Vol. 24, No.6, July/August 1999, pp 509 - 525

2) IBM Agents, Mobile Java™ Agents with Aglets™, by Christian Weigel, Department of Computer Science,University of Applied Sciences Kaiserslautern,Amerikastr. 1, 66482 Zweibrücken, Germany,cweigel@gmx.net - http://www.christianweigel.com/

3) Mobile Agents: What about Them? Did They Deliver what They Promised? Are They Here to Stay?, by George Samaras,Department of Computer Science, University of ,Cyprus,CY-1678 Nicosia, Cyprus,cssamara@cs.ucy.ac.cy

4) MOBILE AGENTS FOR CONTENT-BASED WWW DISTRIBUTED IMAGE RETRIEVAL , by Sabu .M Thampi1, Dr. K. Chandra Sekaran2

5) Digital Watermark Mobile Agents* , by Jian Zhao and Chenghui Luo

6) Chess, D., Harrison, C., and Kershenbaum, A. (1997). *Mobile agents: Are they a good idea? ,*In: Mobile Object Systems: Towards the Programmable Internet, pp. 25-45. Springer-Verlag,

7) Cloud Garden's Java Speech API, December 3 2002. URL:

8) http://www.cloudgarden.com/JSAPI/. ,321 South Main Street,Providence, RI 02903

9) M. Cowling and R. Sitte. Sound identification and direction detection in matlab

10) For surveillance applications. In Proceedings of Australasian MATLAB Users Conference, November 2000.

11) R.H. Glitho, E. Olougouna, and S. Pierre. Mobile agents and their use for information retrieval: A brief overview and an elaborate case study. IEEE Network, 16(1):34–41, 2002.

12) D. Hagimont and L. Ismail. A Performance Evaluation of the Mobile Agent Paradigm. In Proceedings of the 1999 ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications, pages 306–313.

13) ACM Press, 1999. ISBN 1-58113-238-7. Speech for Java, December 3 2002. URL: http://www.alphaworks.ibm.com/tech/speech.

14) IBM ViaVoice, December 3 2002. URL: http://www-3.ibm.com/software/speech/.

15) Java Media Framework API, December 5 2002. URL: http://java.sun.com/products/java-media/jmf/.

16) Java Sound API, December 5 2002. URL: http://java.sun.com/products/javamedia/sound/.

17) Mitsuru Oshima, Guenter Karjoth, and Kouichi Ono. Aglets Specification 1.1 Draft. IBM Corp., September 1998.

18) S. Pfeiffer, S. Fischer, and W. Effelsberg. Automatic audio content analysis. In Proceedings of the fourth ACM international conference on Multimedia, pages 21–30. ACM Press, 1996. ISBN 0-89791-871-1.

19) M. Scarpa, M. Villari, A. Zaia, and A. Puliafito. From client/server to mobile agents: an in-depth analysis of the related performance aspects. In Seventh International Symposium on Computers and Communications, pages 768–773. IEEE Press, July 2002.

20) Sun Microsystems, Inc. Java Speech API Programmer's Guide, October 1998.

21) Comparisonics' homepage, September 27 2002. URL: http://www.comparisonics.com.

22) R. Polfreman and C. Spevak. Sound Spotting – a Frame-Based Approach. Technical report, University of Hertfordshire, 2001.

23) Surya Nepal, Uma Srinivasan, and Graham Reynolds. Automatic detection of 'goal' segments in basketball videos. In Proceedings of the ninth ACM international conference on Multimedia, pages 261–269. ACM Press, 2001. ISBN 1-58113-394-4.

24) Performance Enhancement of E-Commerce Applications using Multiple Mobile Agents