AN ARCHITECTURE FOR MULTIMODAL APPLICATIONS OVER WIRELESS DATA NETWORKS

Nikos Tsourakis, Dimitris Pratsolis, Costas Harizakis, Vassilis Digalakis

Department of Electronic and Computer Engineering
Technical University of Crete, Greece
{ntsourak, pratsdim, harizak, vas}@telecom.tuc.gr

ABSTRACT

As speech is a natural way of interaction between humans and machines, more and more applications will arise that exploit the specific feature. Multimodality comes as the next step in order to offer even more natural, friendlier and personalized interfaces. In this work, we present the architecture of a distributed system that utilizes the data channel (GPRS or 3G), in order to accommodate multimodal applications on a mobile device. Within the context of our work we also introduce a generic framework for creating applications of this kind. Finally, we integrate the proposed system in order to demonstrate a sample application.

INTRODUCTION

Speech offers an attractive alternative to keypad input for telephone-based interaction, where small displays and keypads make viewing and data-entry difficult. Speech is a natural interface and allows users to occupy their hands and eyes to other tasks. Mobile devices should by nature be as compact as possible; therefore speech input minimizes the need for extra and larger input hardware.

In most cases, speech interfaces should work complementary, rather than as a replacement to other forms of input. For instance, when sensitive information such as PIN numbers and passwords are requested, users may prefer a text-based input instead of their voice. In addition, the use of speech in very noisy environments may be cumbersome.

On the other hand, obtaining information by voice can sometimes be less efficient than retrieving it from a text-based mobile application. The user, for example, might forget after several minutes the information that had previously heard or even be misled by his understanding of the specific information.

An application on a handheld device that combines a speech-based user interface and an information display would be ideal in a number of cases. The presented output can contain a variety of information, which might be confusing when announced through the speaker. The specific information can later be revisited, saving time, effort and money.

In this work we propose and implement a service that incorporates the topics discussed so far. The convergence of different ways of interaction to a common, integrated multimodal interface hosted on a mobile device, is the subject of our research.

RECOGNITION OVER DATA CHANNEL

The trend of the continuously increasing use of data communication is expanding to the mobile wireless world, as it has already taken place in the world of landline communications. The need for data access is growing and so is the demand for new applications that take advantage of the offered medium. Smart, portable devices, using speech input would be a perfect choice for accessing the large volume of information. At present the memory and computational limitations encountered to the mobile devices do not permit autonomous speech recognition deployment, hence distributed, client-server solutions are employed. Centralized servers can accommodate the burden of executing complicated processes, and therefore a mechanism for accessing the specific servers should be proposed. This particular mechanism must offer transparent and reliable integration with different types of commercially available Automatic Speech Recognition (ASR) systems in the form of an easily adopted infrastructure, capable of providing speech recognition services to the user.

Many companies worldwide offer speech recognition applications that utilize the existing circuit-switched network. However IP connections for mobile phones with GPRS or 3G also exist. This fact brings the necessity of a standardized way of using IP services from the mobile devices (Figure 1).

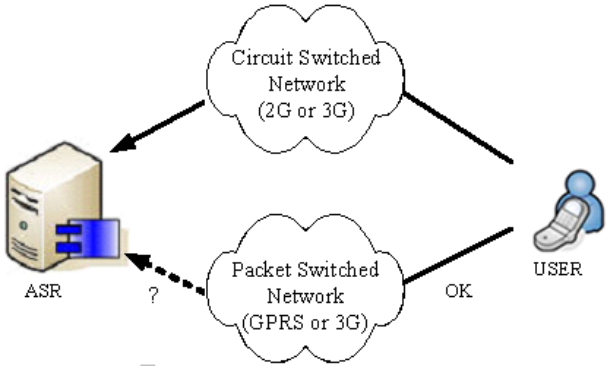

Figure 1:Recognition over data channel

Users connected to the circuit switched-network can simply call the remote ASR system, which is configured to accept and serve the incoming calls. The response of the interaction is basically an announcement. In the packet-switched network, the request and the response is a data flow, thus better customized and manipulated.

In the current work we introduce the utilization of the data channel in order to accomplish speech recognition and subsequently speech application development on a mobile device.

SYSTEM ARCHITECTURE

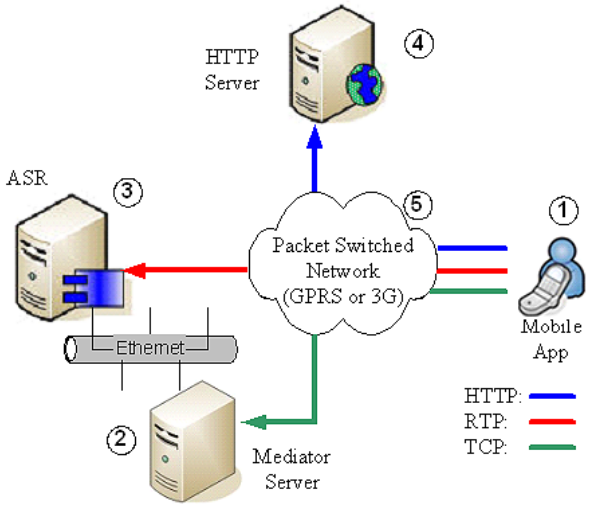

Figure 2: System architecture

Initially, we will provide a high level overview of our proposal. As we can see in Figure 2, five nodes constitute the proposed system architecture, namely:

1. The Mobile Application
2. The Mediator Server
3. The Automatic Recognition System (ASR)
4. The HTTP Server
5. The Packet Switched Network

The first four components work as autonomous peers in the same network and utilize the data network, either GPRS or 3G (component 5). Two protocols are employed in order to establish and use the communication link among components 1, 2 and 3. Specifically, the link is implemented using the Transport Control Protocol (TCP) and the Real-Time Transport Protocol (RTP), while the information retrieval is performed using the HTTP protocol.

The user has a custom application installed on his mobile device, which implements a multimodal information retrieval service (e.g. stock market service, travel service, entertainment service etc). The specific application dispatches speech recognition requests to the ASR through the Mediator Server and utilizes the HTTP Server in order to retrieve and then present enriched information concerning the user's demands. The RTP link between the application and the ASR engine is established on demand, when audio data need to be transmitted for recognition over the network.

In the specific implementation, the Mediator and the ASR can be hosted on the same server, but in a more generalized version they can reside on different servers sharing the same Ethernet network.

The proposed architecture assumes that mobile devices have unobstructed access to all other peers, through proper configuration of the packet switched network.

MOBILE APPLICATION

The application installed on the mobile device contains all the necessary logic in order to initiate and process all transactions involved while the user is interacting with the system. This logic is also responsible of gathering users' speech as well as forwarding it to the ASR, using the RTP protocol. Initially, the application interacts with the Mediator Server in order to prepare the ASR and obtain configuration parameters for the system. It is the Mediator that will eventually supply the recognition result to the application. The specific result is processed by the latter, in order to create and transmit an HTTP request to the HTTP Server, the response of which contains the information that will be presented to the user. The application is also responsible of formulating the response and presenting it on the device.

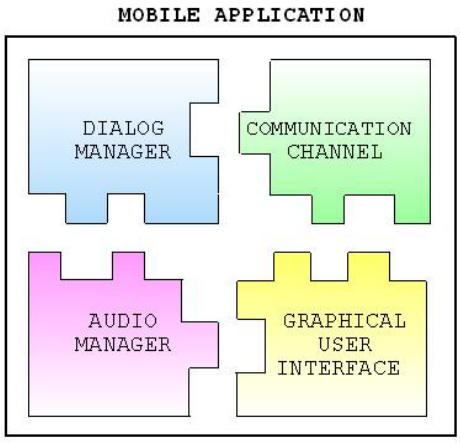

Figure 3: Mobile application components

As we can see in Figure 3, four components constitute the base of our application. Specifically:

1. The Communication Channel
2. The Dialog Manager
3. The Audio Manager
4. The Graphical User Interface

In particular, the aforementioned components offer a generic framework that contains all the necessary features for any mobile application that wishes to employ speech recognition services. One can customize the specific framework and implement a new application that utilizes the presented system. In this way, a bouquet of applications can be introduced, which share a common basis and follow a common approach.

Communication Channel

The specific component is responsible for all communications that take place and concern the mobile application. It serves as an abstraction layer to all communication services and protocols, allowing the transparent message and data exchange between all involved components.

Messages are exchanged in a request-response fashion using a reliable network protocol such as TCP/IP, while audio data is forwarded to the ASR engine through the time sensitive but yet unreliable RTP/UDP protocol.

The Communication Channel transmits requests of the mobile application to the Mediator Server and receives messages or possible errors from the latter using the TCP protocol. It can also transmit audio data to the ASR, after converting the recorded audio stream to RTP packets. Finally, HTTP requests are forwarded to the HTTP server.

Dialog Manager

The Dialog Manager constitutes the application logic. The specific module is composed by one or more finite state machines, each one defined as a set of states and transitions. States may generate events, send messages through the Communication Channel, perform state-specific operations and/or trigger a transition to another state.

There are several actions that need to be performed before using the proposed application development framework (such as initializing the ASR engine), each one modeled and managed/handled by a single finite state machine. The proposed application framework includes reference implementations for the most commonly used state machines, all of which are discussed more thoroughly later in this document.

Moreover, the Dialog Manager determines the dialog flow according to the received messages from the Mediator Server. Although the flow is predefined, it must handle unexpected messages and deal with them gracefully.

Audio Manager

Audio Manager provides the mechanism for collecting the speech input. It provides the necessary methods for recording, playing back and processing audio data. The format of the audio data can be customized and in our reference implementation a sampling rate of 8KHz, 8 bits per sample, mono audio, and PCM audio format has been chosen. The specific configuration offers a bit rate of 64Kbits/sec.

Graphical User Interface (GUI)

The supplied GUI, offers a basic and minimal functionality, which should be used mainly for test purposes or for non-demanding applications. It provides all necessary menus for establishing a connection with the Mediator Server as well as recording and transmitting audio data to the ASR engine.

The introductory splash screen can be customized according to the application needs and built-in audio sounds offer a friendlier interaction with the user.

Finally, the component provides a generic mechanism for presenting messages to the user, which includes error messages, information messages and action messages.

MEDIATOR SERVER

This module works as the mediator between the application on the mobile device and the ASR engine. It is responsible of forwarding text-based messages between the aforementioned entities and of ensuring the proper interconnection between them. When the need of using a different ASR engine arises, one should only implement a new mediator for the specific ASR without reinstalling a new mobile application to the end-users' mobile device. Most ASR engines offer their services through a set of usually rich Application Programming Interfaces (APIs). However, the proposed mediator specification requires the implementation of only a limited subset of this functionality.

Specifically, a mediator should support the actions presented in Table 1. Each action is composed by a set of text-based messages including:

1. A REQUEST from the mobile application to the Mediator, initiating an action.
2. An ACKOWLEDGEMENT from the Mediator to the mobile application, confirming that the REQUEST was received and is currently being processed.

3. One or more EVENT messages from the Mediator to the mobile application, providing additional information regarding the progress of the on-going action.
4. A RESULT message, indicating the completion of the on-going action.

TABLE 1 – Supported Actions

CREATE RECOGNITION ENGINE Initialize and set up the recognition engine.
DELETE RECOGNITION ENGINE Delete the created recognition engine.
SET INTEGER PARAMETER Set the value of an integer type parameter.
GET INTEGER PARAMETER Retrieve the value of an integer type parameter.
SET FLOAT PARAMETER Set the value of a float type parameter.
GET FLOAT PARAMETER Retrieve the value of a float type parameter.
SET STRING PARAMETER Set the value of a string type parameter.
GET STRING PARAMETER Retrieve the value of a string type parameter.
RECOGNIZE Start a new recognition action.
ABORT RECOGNITION Abort the current recognition action.

Figure 4 depicts the messages that are exchanged between the mobile application and the Mediator Server for the REGOGNIZE action:

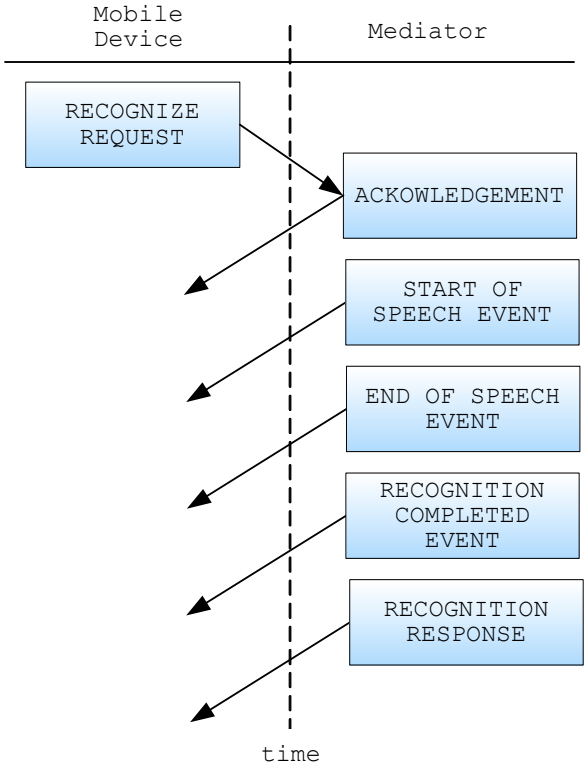

Figure 4: The RECOGNIZE action

The mobile application submits a recognition request. The mediator forwards the request to the ASR engine, the response of which is propagated to the mobile application in the form of an acknowledgement message. Several events are also received and propagated to the mobile application as part of the on-going recognition action. Finally, the recognition result is obtained through an appropriate result message.

The mediator must be able to serve multiple mobile applications at the same time and distinguish the different demands of each one of them.

AUTOMATIC SPEECH RECOGNITION SYSTEM

The ASR system should be able to serve multiple applications that can be installed on the mobile devices, each one of them requesting recognition of a spoken utterance in a different context.

This is typically achieved through the use of recognition grammars, which can be either precompiled and stored in the ASR system, or created and modified dynamically, while the mobile application is running.

As we have already mentioned, the proposed architecture can work regardless of the ASR that we engage. One should only implement the common interface offered by the Mediator for the specific ASR.

HTTP SERVER

We have chosen the HTTP protocol for our systems data-retrieval mechanism, since the volume of the exchanged data is rather limited. Once we know the user's request, we can seek for the suitable answer. Therefore, the following 4-step procedure is incorporated:

1. The spoken request is recognized and interpreted by the ASR engine.
2. An HTTP message containing the user request is constructed and sent to the remote HTTP server.
3. The HTTP server invokes the appropriate servlet (JSP).
4. The servlet retrieves any required information from a database and creates a response, which is sent back to the mobile device.

SIMULATION ENVIRONMENT

The reference implementation platform is the Series 60 Developer Platform provided by Nokia. It is a complete smartphone software package that provides an open, secure and scalable business opportunity to mobile operators and third-party developers. Having been

adopted by several mobile phone manufacturers, the platform offers a variety of features and integrated technologies.

Series 60 devices are based on the Symbian OS, an open, robust, 32-bit multitasking operating system designed for smartphones. All components were implemented using the C++ programming language.

STOCK SAMPLE APPLICATION

In this last section, we present a sample stock-information application that implements the proposed architecture. In particular, the objective was to offer a natural interaction to the end-user as well as a compact and complete representation of the information he or she may be interested in. It has been installed and tested on the mobile device emulator.

Our implementation encapsulates the components that need to be part of any application. Moreover, it introduces two more components, integrally related with the proposed framework:

1. The StockApp Dialog Manager derived from the Dialog Manager.
2. The StockApp GUI derived from the Graphical User Interface.

The StockApp Dialog Manager implementation delegates most of its functionality to the base Dialog Manager component. Its only concern is to inform the latter about the recognition grammars that will be requested from the ASR. Moreover, it is responsible, after receiving the recognition result, to perform a customized HTTP request and parse the corresponding response.

The second component contains custom bitmaps and icons, used by the stock information application. In addition, it defines the layout of all interaction screens, including the voice-input form, as well as the information-presentation charts.

Our implementation has been tested using the English language. However, it can easily be internationalized by modifying a set of configuration files.

Stock Application Simulation

In conclusion, we will provide a brief demonstration of our implementation, executing a simulation on the emulator. As we can see in Figure 5, our deployed stock application appears along with the other available applications, which are installed on the emulator.

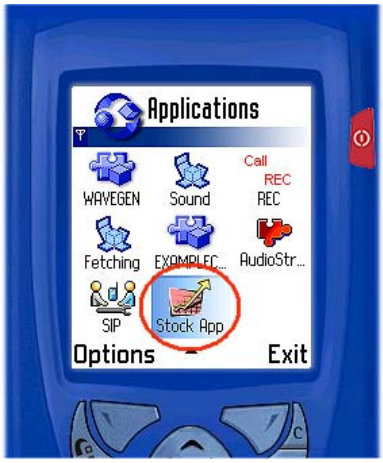

Figure 5: Stock application

When it is executed the screen shot shown in Figure 6 appears and the user can choose between the Options menu (1) or exiting (2).

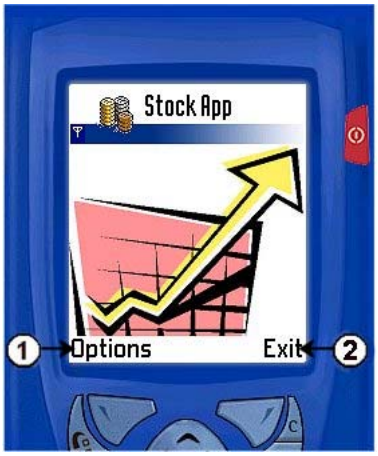

Figure 6: Initial screen

The main menu offers seven choices. The seventh menu item is used for exiting from the application and is not shown in Figure 7.

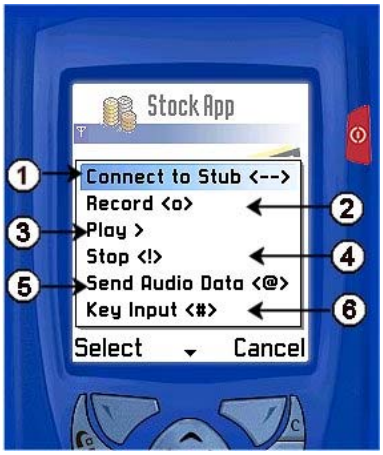

Figure 7: Application menu

Initially, the user should select the first menu item in order to connect to the mediator. At this point the "CREATE RECOGNITION ENGINE" action will be executed and the ASR will be ready to receive audio for recognition. The completion event of this action contains the remote address of the ASR engine, where any recorded audio data will be forwarded for recognition. We should note that the menu appeared in Figure 7, is the minimal menu inherited from the Graphical User Interface component discussed earlier. Its usage is not mandatory and one can offer an application-based menu.

The second menu item initiates the recording of input speech. As we can see in Figure 8, the user is prompted to utter a stock name. The ASR server is waiting to accept audio data from the mobile device and is prepared to recognize a stock name. The recorded audio is converted into audio packets, which are sent to the ASR, when the user presses OK (Figure 8). Recording can be aborted by pressing the CANCEL button and consequently the "ABORT RECOGNITION" action is executed.

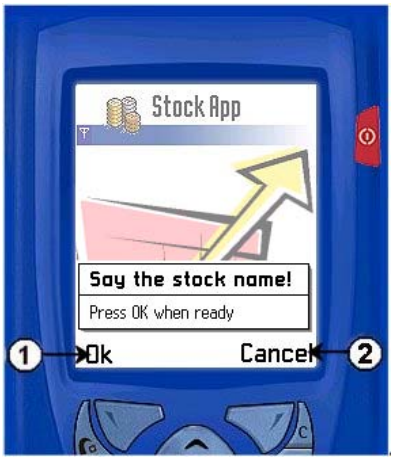

Figure 8:User input

After uttering the name of a stock, the user can chose between playing back the previously recorded utterance (menu item 3 – Figure 7) and retransmitting it to the ASR engine (menu item 5 – Figure 7).

Another mode of input can be incorporated when the user selects the sixth menu item. The specific mode concerns key input, which is demonstrated in the next paragraphs.

When the seventh menu item is chosen, the "DELETE RECOGNITION ENGINE" action is executed and the application exits.

After all the appropriate messages (requests, acknowledgements, and events) are exchanged, the recognition response arrives. The specific response contains all the textual representation of the stock name that the user asked for. In our case, every stock name is associated with a unique key. This key is supplied as a parameter to the HTTP request, which is submitted to the remote HTTP server. The HTTP server processes the received request, invokes the appropriate servlet, which in turn retrieves the requested information from a database. This information is converted into a well-formed textual message, which is transmitted back to the stock application. The application processes the received textual data and presents it to the user using the layout shown in Figure 9.

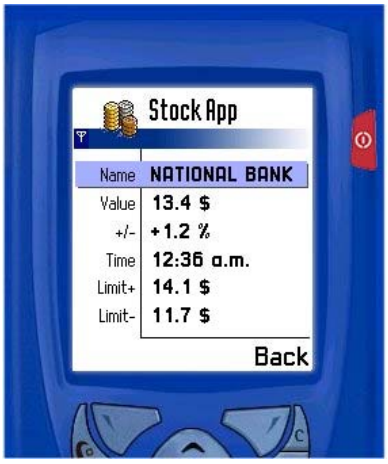

Figure 9: Application output

As we can observe, the user is informed about the name of the stock, its current value, the variation of its value, the time of the last update and the daily maximum and minimum reached value. By pressing the back button, the user can return to the initial screen for a new recognition task.

Figure 10: Key input

As we have mentioned earlier, the application offers an alternative mode of input that is the key input (Figure 10). The users instead of uttering the stock name can simply choose one of the available stocks in the list. Each stock name appeared in the list is associated with the same unique key returned by the recognition result. The drawback of key input is that the user must search

in a number of stock names and the corresponding list must be updated frequently. On the other hand it can be very helpful in noisy environments or when the user wishes private transactions. After the user chooses one of the available stocks, a corresponding HTTP request is performed with the key associated with the stock and the result is again the one depicted in Figure 9.

FUTURE WORK

In this work we described the basic components of a distributed system, aimed at creating multimodal applications on mobile devices. We demonstrated a transparent way of integrating those applications with any kind of automatic recognition system. Our primary objective was to provide an integration layer that is simple to implement and has low memory, CPU and network bandwidth requirements, as the amount of the exchanged messages is rather limited and their structure is minimal. Moreover the proposed integration layer that depends on widely available technologies offered by most mobile devices, was used in order to develop a platform that will allow us to study the human-computer interaction on mobile devices. The W3C has proposed a similar but far more complete and complicated protocol stack for managing ASR and Text-to-Speech (TTS) engines over the network, the MRCP, Shanmugham et al (1), over RTSP, Schulzrinne et al (2). Our development roadmap includes the adoption of the MRCP, as it is becoming an emerging standard, used by most commercial ASR and TTS products.

We also utilized the proposed system in order to create a sample application and therefore demonstrate its effectiveness. The offered framework was implemented using the widely appreciated C++ API of the Symbian OS (3), which provides unparalleled flexibility and extensibility compared to all other available APIs. As mobile devices' capabilities increase, more sophisticated APIs may be adopted, such as those proposed by the SALT (4) and the VoiceXML (5) specifications.

Multimodality presents the benefits that we have discussed so far. Although our work converges to the specific direction, new additions and improvements are welcomed and necessary.

The system could be improved with the use of algorithms that perform the feature-parameter extraction, or the front-end of the speech recognition system on the mobile device, Digalakis et al (6), (7) and Ramabadran et al (8). These features could later be transmitted over the data channel to the remote back-end recognizer. In this way, the burden of transmitting raw audio data could be lightened, thus reducing the response time and the cost.

Finally, besides the text data that is the result of the interaction between the user and the system, we can obtain and represent multimedia, like audio or images, from the HTTP Server. The specific information could be rendered on the mobile device and therefore extend the set of possible applications that can be implemented.

REFERENCES

1. S. Shanmugham, P. Monaco, B. Eberman, 2005, "A Media Resource Control Protocol Developed by Cisco, Nuance, and Speechworks", Internet Engineering Task Force, http://www.ietf.cnri.reston.va.us/internet-drafts/draft-shanmugham-mrcp-07.txt.

2. H. Schulzrinne, A. Rao, R. Lanphier, 1998, "Real Time Streaming Protocol (RTSP)", Internet Engineering Task Force, http://www.ietf.org/rfc/rfc2326.txt.

3. "Symbian OS, the mobile operating system", http://www.symbian.com/.

4. Salt Forum, 2002, "Speech Application Language Tags (SALT) 1.0 Specification", http://www.saltforum.org/saltforum/downloads/SALT1.0.pdf.

5. W3C Recommendation, 2004, "Voice Extensible Markup Language (VoiceXML) Version 2.0", http://www.w3.org/TR/voicexml20/.

6. V. Digalakis, L. Neumeyer, and M. Perakakis, 1999, "Quantization of cepstral parameters for speech recognition over the World Wide Web", IEEE J. Select. Areas Commun., vol. 17, pp. 82–90.

7. V. Digalakis, L. Neumeyer and M. Perakakis, 1998 "Product-code Vector Quantization of Cepstral Parameters for Speech Recognition over the Web", Proceedings ICSLP.

8. T. Ramabadran, A. Sorin, M. McLaughlin, D. Chazan, D. Pearch and R. Hoory, 2004, "The ETSI Extended Distributed Speech Recognition (DSR) Standards: Server-Side Speech Reconstruction", Proceedings ICASSP.