An Architecture for Voice-Enabled Interfaces over Local Wireless Networks

M. Bagein, O. Pietquin, C. Ris and G. Wilfart

Faculté Polytechnique de Mons - TCTS Lab.
Parc Initialis - Av. Copernic, 1
B-7000 Mons - Belgium
{bagein, pietquin, ris, wilfart}@tcts.fpms.ac.be

Abstract

Lots of industrial tasks need contacts between operators and a central Information Management System. Permanent contact creates a more effective and efficient workforce with a direct update in the central Information Management System. In the case of mobile operators (e.g. warehouses or picking operators), this can be realized thanks to electronic remote devices connected to a private wireless network.

Especially in mobile environment, voice often appears as a natural way to interface with computers. Indeed, a voice-enabled interface (understand voice recognition, dialogue management and speech synthesis) can improve user's ergonomics, as hands and eyes can still be focused on the current task therefore improving the productivity by avoiding the necessary amount of time of extra manipulation as form-filling, data encoding, etc.

In this paper, we propose a generic framework for the implementation of efficient mobile and distributed voice-enabled interfaces.

Keywords: wireless network, vocal interface, distributed processing.

1. Introduction

In many domains, such as logistics, stock management or distribution centers, a human operator interacts with machines. This interaction is made via Human-Computer Interfaces (HCI), which allow users to communicate with an application server. Such domains often require operators to be mobile, for instance to pick up and manipulate items of a wide variety of characteristics (e.g. pens, jewels, etc.).

An efficient HCI must then allow communications over a wireless network. In this environment, voice also appears as a natural way to interface with machines, adding extra advantages in terms of ergonomics and productivity, as hands and eyes remain free.

However, such an interface imposes ergonomic, but also economic and technical constraints. In the rest of this section, we investigate the requirements of the system.

In section 2, we will propose a software architecture that meets those requirements. Then, we detail the voice interface in terms of dialogue management (section 3), Automatic Speech Recognition (ASR, section 4), and Text-To-Speech (TTS, section 5) systems.

Ergonomic constraints

Mobile devices should not interfere with the operator's movements. They should be light, robust and low consuming. Those constraints mainly impose the use of embedded devices that have typically low CPU and low memory capacities.

The application itself should be as intuitive as possible, and possibly require no particular user training. In such a way, the design of the dialogue is a crucial point of the voice application, as, if efficient, it can greatly improve the performance of the overall system in terms of operator's satisfaction.

Economic constraints

Due to the possible use at large scale, mobile hardware should be low cost. In this sense, speech interface hardware is usually cheaper than its visual counterpart. This gives the voice interface another particular advantage.

Unfortunately, this could imply low-quality acoustic devices (microphones, ...), certainly not dedicated to speech recognition. The poor quality of the acquisition must be taken into account by the speech recognition software.

Technical constraints

Target applications imply that the environment is often noisy. Thus, the speech recognition software should be robust enough to handle this problem.

Moreover, a natural voice HCI faces the complex problem of spontaneous speech recognition, namely handling hesitations, out of vocabulary words, etc. Once again, an efficient dialogue design can greatly improve the performance of the voice interface.

The use of speech synthesis is preferred over the use of prerecorded prompts, for an increased flexibility of the dialogue and an easier upgrade of the applications.

On another hand, mobile devices are connected to the fixed terminals through a wireless network. Due to the common medium of the radio waves (atmosphere), the unique network resource between mobile and fixed devices is limited, shared and not expandable.

2. Software Architecture

In our approach, we consider that the operator uses speech to interact with the customer's application.

In the next topics, we detail the implementation of speech technologies (TTS, ASR and dialogue management) in a mobile environment and its integration in a classical third-part software architecture: data browser, data presentation server and information management system (figure 1).

Data browser

Embedded audio software: the minimal requirements for mobile audio devices are a microphone, a headset, a keypad and a wireless network adapter. A keypad is also required to leave the operator a part of the process control (start, stop, pause,

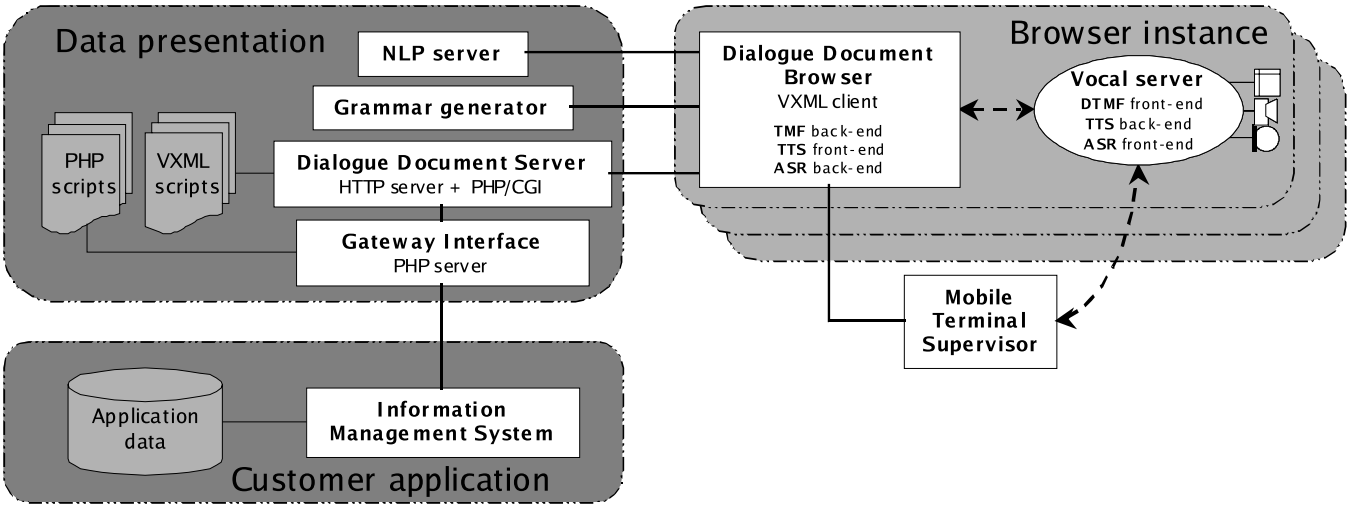

Figure 1: software architecture. Dotted arrows symbolize data stream over wireless network.

resume, and/or repeat keys). At the software level, the operator interacts essentially with the application through ASR and TTS systems but the possibility to recording and playing audio memos must be also available. Due to ergonomic and economic reasons, the embedded CPU and memory resources could be very limited in the mobile device. Then, the embedded software set cannot include all the components required by the browser instance.

One solution is to split the browser algorithms between mobile and fixed devices. In our approach, we consider that only keypad handler, audio player, audio recorder, ASR front-end and TTS back-end are implemented in mobile devices (figure 2). The counterparts of ASR and TTS and the dialogue document browser (VXML interpreter - [1]) are implemented on fixed devices. Each mobile runs a single instance of the embedded TTS/ASR software and this instance is connected, through socket mechanisms over the wireless network, to a single instance of a VXML browser.

The wireless network introduces new constraints. According to the current radio conditions (distance, obstacles, reverberation, etc.), the available bandwidth can severely degrade. As concrete applications can require up to several hundreds of simultaneous communications, the use of efficient coding algorithms (in terms of bandwidth an CPU requirement) is essential.

Dialogue document browser: the dialogue document browser parses VXML documents. According to the dialogue description, the browser can start the audio output dataflow (audio playing or TTS) and the audio input dataflow (audio recording, ASR or keypad event). The result of the input is then analyzed to start a new action: jump to another point of current dialogue, run JavaScript action, load a new dialogue document, etc.

We consider two types of audio output: audio memos and sentences. Audio memos are directly sent to the mobile device but in case of sentences, a TTS system is required. It is possible to process sentences in a full TTS process and transmit the result as audio memos but it is more efficient in terms of bandwidth and CPU to split the TTS algorithms and transmit intermediate data over the wireless network. The TTS front-end acts as a layer using a translator (section 5) between sentence and intermediate data. Once again, an optimization is possible by using a single NLP resource shared by all TTS front-end algorithms, and, with this point of view, the NLP resource can be considered as part of data presentation (NLP server).

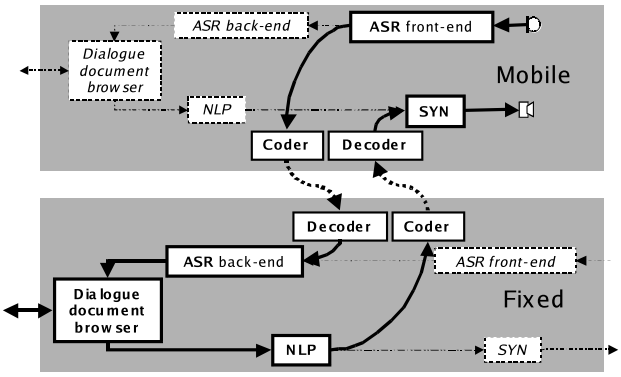

Figure 2: algorithm distribution over wireless network. TTS (front-end = NLP, back-end = SYN) and ASR algorithms are illustrated with their respective codecs.

In the case of user input, before performing speech recognition, the ASR process must be initialized with a recognition grammar (section 4). As the NLP server, a single grammar generator resource can be shared by all ASR back-end algorithms, so the grammar generator can also be considered as part of data presentation.

Mobile Terminal Supervisor: when a mobile becomes operational (switch on) and when it enters in the radio coverage, (reaching wireless network dedicated to the vocal application), the mobile device request the Mobile Terminal Supervisor to be join the application. Then, the new mobile is registered and it can describe its characteristics, availability, and hints as to which dataflow flows can be serviced. Besides useful actions (login/logout and authorization management), the Mobile Terminal Supervisor must periodically check the current status of each mobile and detect connection losses. Those status informations are required to handle suspended dialogue mechanisms (section 3). An implementation of Network Element Control Protocol (NECP) [2] is used to perform this management.

Data presentation

The vocal scenarios of the application are described in a set of dialogue documents, which respect the VXML formalism [1]. The goal of the dialogue document server is to diffuse dialogue documents towards the browsers. A usual HTTP server can be used for this task. Dialogue documents could be stored in a file system or could also be generated from information management system data. In this last case, a gateway interface (PHP server) should be used to generate queries and format the result inside a VXML template.

As said before, the Grammar Generator and the NLP server are considered as presentation servers and are included in the data presentation.

Customer information system

One of the main advantages of this architecture is the possibility to add a vocal interface without re-designing the customer Information Management System (IMS). All communications with the IMS are made through the dialogue document server which can support well-known gateway interfaces (PHP, Perl CGI, batch scripts, ...).

3. Dialogue Management

Spoken Dialogue Systems (SDS) don't only involve speech and Natural Language Processing (NLP) systems but some management of the interaction is also necessary. A SDS can be seen as an automaton that steps from state to state according to the results of the actions it performs in its environment (the human operator and the IMS in this case). Possible actions might be TTS prompts to human operators, databases queries etc. Designing a dialogue management system consist in assigning particular actions to each dialogue state. An optimal dialogue strategy then a mapping between the dialogue state space and the action set that leads to an appropriate and ergonomic sequence of interactions allowing the achievement of the operator's goal. Optimal dialogue strategy design is a complex task and is the object of many recent researches [3].

For this specific application, some points have to be taken into account. First, as said in the introduction, for some reasons such as noisy environment or low cost acquisition systems, the speech recognition process can become less accurate. Substitutions, additions or deletions of words in the recognized command can occur. The use of judiciously designed dialogue management can improve the speech recognition performance. For example, confirmation mechanisms may be used if necessary.

Secondly, as the main mode of interaction of the application is voice, the user may not have any visual information. In usual Graphic User Interfaces (GUI), to ensure the exactness of database queries, the possible choices in a given state can be presented in a combo-box, for example. Those choices can be constrained by partial queries based on previous choices that occurred in previous states. Such a mechanism cannot be used with vocal interfaces, as it should be necessary to enumerate all the possible choices in an audio prompt. This would be very bad for ergonomics if the list is large. The user should than be allowed to build queries that potentially provide no result. If it is the case the dialogue manager has to warn and advise the user in order to build a correct query in a natural way. In another hand, in the ergonomic point of view, users should not have to wait and or should know why. In a GUI, progression bars, dynamic mouse cursors or status bars provide information about the status of the current process. As our application relies on an existing IMS that we should not modify, it is necessary to deal with its possible weakness. For example, database queries might be time consuming. The dialogue strategy should then take timing constraints into account and should not allow long pauses between utterances. So, it is necessary to introduce new utterances in the dialogue that provide information about the system's state.

Finally, mobile devices evolving in a wireless network area may go out of radio coverage. Moreover, human operators may be constrained to suspend the dialogue, because they are interrupted by a colleague for example. In order to recover an incomplete interaction, the system keeps trace of the dialogue's history by filling in a template all along the interaction. It gathers all the information obtained so far by the system and remains active while the dialogue has not come to its end. This represents the dialogue's current state. Nevertheless, being able to continue a suspended dialogue is not enough. The system must also be able to decide whether it has to remind the operator about the dialogue state. Indeed, if the dialogue has been suspended for a long time, the operator may have forgotten the interaction's history. On another hand, for ergonomic reasons, this should not be systematic, as a short interruption does not require reminding. Moreover, the embedded part of the system must be able to warn the operator that it has detected a connection loss and prevent him to continue the interaction.

4. ASR System

Automatic speech recognition is the part of the vocal interface, which aims at transcribing into text what has been pronounced by the user. Even if the technology is rather mature nowadays, integrating speech recognition into concrete applications is still a difficult problem. Noise conditions, speaker characteristics, spontaneous speech are typical difficulties we have to face. Integrating this technology in a wireless network architecture imposes several additional technical constraints that justify the need of distributing the ASR processing on both the mobile and fixed terminal.

Distributed speech recognition (DSR)

The part of the processing which is achieved on each terminal and the actual implementation of these processes result from a trade-off between the computation power available on the mobile end (mobile phones, PDA, laptops, ...), the transmission rate over the wireless connection and the required recognition performance.

1. Indeed, the computation and memory capacity of the mobile devices can be very constrained while some parts of a speech recognition system require a lot of CPU. The part of the processing achieved on the mobile terminal will therefore be limited by its intrinsic performance.
2. In the framework of a wireless connection between the terminals, the available band-with of the network appears as a major criterion in the choice of the implemented technologies. The necessity to reduce as much as possible the bit rate will also impose, in a certain way, the actual distribution of the tasks between the fixed and mobile terminals.
3. The coding algorithms applied to the data in order to limit the bit rate will have some impact on the performance of the speech recognition system depending on the type of compression schemes. We know, for instance, that a coding performed on the sampled speech signal (cf. GSM codec), is much more detrimental for the recognition performance than a coding of the parameters issued from the acoustic analysis of the speech signal (cf. DSR codec),

especially when the transmission conditions are getting worse [4].

4. The maintenance of the applications - database or acoustic models updates, ... - is another aspect of the problem that should be taken into account in the process distribution strategy. Indeed, it is preferable to avoid, as much as possible, the maintenance operations on the mobile terminals, operations that could become very complex at large scale.

Those particular points raised by the problem of distributed speech recognition have been widely discussed in the speech community. The international project AURORA [5] has been created in order to propose solutions and finally establish an international standard for DSR (recently published by the ETSI - European Telecom Standards Institute [6]).

The speech recognition processing can roughly be split into four steps: 1) the Voice Activity Detector (VAD) which isolates the useful speech signal from the signal flow acquired from the microphone, note that VAD should be coupled with a barge-in functionality to allow the user to interrupt audio prompts when necessary, 2) the acoustic analysis which aims at extracting from the speech signal, a relevant and non redundant representation of its acoustical content, 3) the acoustic modeling which aims at classifying the acoustic features into language dependent acoustic units (phonemes, diphones, syllables, ...), and 4) the decoding that integrates the vocabulary and the grammar into a hidden Markov model structure in order to obtain the word sequence hypothesis [7].

The ETSI standard proposes to perform the voice activity detection and the acoustic analysis on the mobile device while the acoustic modeling and the decoding are achieved on the fixed terminal. This solution satisfies all the constraints listed above:

- Acoustic analysis requires very low and fixed memory and CPU load.
- Coding of the acoustic features is efficient and has almost no impact on the recognition performance. A bit rate of 4.8 kbits/s is proposed by the ETSI standard.
- Acoustic feature extraction requires no external data such as databases or models, it is independent of the language and of the application so that the maintenance of the ”mobile” application is reduced to the minimum, practically software upgrade.

Noise robustness

Different external parameters can alter the performance of the speech recognition systems. Some of them such as intra- or inter-speaker variability, co-articulation effects, ... can be efficiently handled by the acoustic models. However, the actual conditions in which the voice interface is used induces particular causes of disturbance that require a specific processing. This is the case of environment noise specific to each application (background conversation, warehouse, engines, ...) or convolutive noise, due to the characteristics of the possibly low quality of the voice acquisition devices such as embedded microphone. Noise robust algorithms must therefore be integrated at different level of the speech recognition process. In particular, noise reduction techniques can be applied to the acoustic analysis, as it is the case in the ETSI standard, given that such processing should not drastically increase the CPU load on the mobile device. Besides, a priori knowledge of the actual conditions of usage should be used as much as possible in order to adapt the speech recognition system to the application.

The grammar generator

An ASR system recognizes not only words, but also complete sentences or commands. It is obvious that for a sentence to be intelligible, not any combination of words is possible. This is especially true in dialogues, where the system expects an answer to a precise question. The set of all the combinations of words that an ASR system can recognize composes the grammar.

Using a grammar improves the recognition, not only by limiting the number of possible sentences, but also by taking decisions on better-recognized parts of the speech, thus enabling error recovery and improving noise robustness.

A dialogue system makes use of several grammars, depending on the dialogue state. For better ergonomics, the operator should also be able to end or suspend a dialogue, or to ask for help at any point. This results in two requirements for the ASR system:

1. it must be able to handle several active grammars at the same time.
2. it must be able to switch between grammars (load and unload, activate and deactivate grammars). An ASR system must meet these requirements to conform to the VXML formalism [1].

The role of the grammar generator is not only to generate grammars from Information Management System (IMS) data, but also:

1. to organize them in some optimal way, in a formalism that is understood by the ASR system. This step is known as grammar compilation.
2. to ensure necessary grammars are available with a minimum latency for the user
3. to ensure that grammars are up-to-date.

To satisfy the last two points, the grammar generator should be able to compile grammars fast enough, and to reuse already compiled grammars (cache mechanism) as long as they do not need to be updated.

Another requirement of the system is that it should handle queries that return no result. To do so, the ASR system must be able to recognize such queries. For instance, if a car retailer has two models A and B, and model A can be blue, while model B cannot, the operator might still ask for a blue B model. Assuming the operator asked for a B model, if the grammar is limited to available colors for a B model, the system cannot recognize the query ”blue B model” and will not be able to warn the operator such a car does not exist. Instead, if the system recognizes ”blue B model”, it can make a query to the IMS, which returns a void result that is interpreted as ”this car does not exist”.

Of course, to keep the grammars up-to-date, it is necessary to check if anything changed in the IMS, thus query its state. This means that the use of the vocal interface generates additional queries. This, together with the use of a cache, justifies that we use a single grammar generator that acts as a server of VXML browsers.

5. TTS System

In many cases, vocal messages (information, questions, ...) can be sent to the user with pre-recorded audio prompts. This technique seems simple but, in fact, is not compliant at all due to the obligation of pre-recording all audio prompts. A Text-To-Speech system aims at transcribing human readable texts in speech sound signal. With such a system, audio prompts can be automatically

generated from prompt texts avoiding audio recording. Moreover, if the TTS system is used inline, new prompts based on the result of information management system queries can be generated on the fly. Then the evolution from day to day of the information management system does not require an vocal interface update.

TTS algorithm distribution

TTS systems need databases [8, 9]. The size of those databases, typically about 30 Mb, is not compatible with mobile device resources. Due to the limited embedded resources and wireless transmission rate, the TTS implementation is also split into two parts. The fixed part of the TTS is the Natural Language Processing (NLP), which aims at translating human readable texts into phonetic transcriptions (including prosodic features). The embedded part is the speech synthesizer, which aims at translating phonetic transcriptions into audio stream. The bit-rate required to transmit phonetic transcriptions from the NLP to the speech synthesizer is lower than 1.5 kbits/s.

Natural Language Processing

The NLP requires a lot of memory and CPU resources. Nerveless, a sentence and its phonetic transcription is very compact. Moreover, a large set of sentences is unchanged in the day-to-day usage of the application. Then, retrieving transcription from previous sentences with a cache mechanism prior submitting a sentence to the NLP process reduces drastically the memory and CPU resources requirements. Eventually, the set of unchanged sentences can be shared between all operators and this last point justifies the presence of a NLP server as a part of the Data Presentation.

Synthesis database distribution

Nevertheless, one of the databases, containing the voice segments, has to be embedded and its size is usually comprised between 5Mb and 2Mb in compressed version [10]. If this size is too big for the mobile device, it is possible to consider a distribution mechanism for voice segment database.

This system implements a database manager associated to a cache memory in the mobile device and a database server in the fixed device. When the embedded synthesizer receives the phonetic stream from the browser, it searches the necessary voice segments in its cache. If a segment is absent from its cache, the embedded synthesizer has to decide which segment(s) must be flushed from the cache and which one(s) must be added. This decision results from cost function based on a priori knowledge about segment probabilities, cache size, next segment occurrence and network transmission rate. Then, the synthesizer can build the query and access the database segment server.

This technique reduces the memory requirement in the mobile device but increases the network bandwidth requirement due to the database segment transmission. So, this technique must be used as far as the required bandwidth is lower than for a direct audio transmission.

6. Communication and Network

Network Protocols

The audio data transmission from mobile devices towards fixed terminals (in ASR) constitutes the biggest network load.

Thus, The choice of the protocol set has been studied with special care. Using WIFI, based on IP network, brings obvious economic advantages but also technical constraints in continuous data flow transmission. At recognition time, the data volume is not predictable (the time that the operator speaks) but the rate flow is constant justifying the implementation of RTP protocol. For example, the AURORA [5] project proposes a bit rate of 144 bytes/240ms.

ASR systems cannot deal with packet losses (each packet covers 240ms of speech). TCP guarantees the packet transmission over IP but increases significantly the network load with its internal validation mechanisms between customer and server. The a posteriori transmission control mechanism on partial packet streams at the level of RTCP/UDP uses less network resources than TCP.

WIFI and collision avoidance

The packet transmission over a radio medium requires a collision avoidance mechanism at the MAC level [11]. This mechanism prevents from a considerable amount of packet losses, making implicitly more reliable the packet transmission over UDP. This advantage will simplify the error recovery mechanism at the RTCP level.

Independency between audio quality and network bandwidth

The Feature Extraction algorithm generates a flow of acoustic vectors. The volume of these acoustic vectors is independent of the audio signal sampling rate. In other words, a microphone audio signal sampled at 16KHz/16bits or 8KHz/8bits generates acoustic vectors different in content but identical in volume. This characteristic makes the ASR data flow independent of the quality of the audio signal. For synthesis, audio quality depends on the voice database quality and not on phonemic input flow. Thus, audio quality in recognition and synthesis is independent of the data transmission rate over the network. Then, mobiles devices with different audio capabilities, improving speech resources in synthesizer (different audio quality, set of embedded voices) and/or in recognition (more robust or specialized front-end), can coexist on the same infrastructure. In the same way, software or even hardware upgrades will not have any influence on the average network load.

7. Conclusion

In this paper we presented a set of software solutions to face the problem of interfacing distant applications over wireless networks using voice enabled interfaces. In this framework, a distributed architecture and particular designs of speech processing tools were exposed.

Acknowledgments

This work is sponsored by the Walloon Region through the ”Initiative” MODIVOC project.

8. References

[1] VXML http://www.w3.org/TR/2003/CR-voicexml20-20030220/.

[2] Cerpa, A., Elson, J., Beheshti, H., Chankhunthod, A., Danzig, P., Jalan, R., Neerdaels, C., Shroeder, T. and G. Tomlinson, ”NECP: The Network Element Control Protocol”, Work in Progress, http://www.circlemud.org/∼jelson/writings/.

[3] E. Levin, R. Pieraccini, W. Eckert, G. Di Fabbrizio, and S. Narayanan., ”Spoken Language Dialogue: From Theory to

Practice" in Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU),1999.

[4] D. Pearce, "Enabling New Speech Driven Services for Mobile Devices: an Overview of the ETSI standards activities for Distributed Speech Recognition Front-Ends", in Proc. AVIOS 2000, San Jose, CA, May 2000.

[5] Aurora 2.0 - http://www.elda.fr/proj/aurora2.html.

[6] ETSI - Distributed Speech Recognition http://www.etsi.org/frameset/home.htm?/technicalactiv/DSR/dsr.htm.

[7] L.R. Rabiner, "An Introduction to Hidden Markov Models", in IEEE ASSP Magazine, pp. 4-16, January 1986.

[8] T. Dutoit, "An Introduction to Text-To-Speech Synthesis", Kluwer Academic Publishers, Dordrecht, 1997.

[9] Paul A Taylor, Alan Black and Richard Caley, "The Architecture of the Festival Speech Synthesis System", in The Third ESCA Workshop in Speech Synthesis, pp. 147-151, 1998.

[10] O. Van Der Vreken, N. Pierret, T. Dutoit, V. Pagel, F. Malfrêre, "New Techniques for the Compression of Synthesizer Databases" in Proc. ISCAS 97, Hong Kong, pp. 2641-2644, 1997

[11] CSMA/CD Access Method and Physical Layer Specifications, ANSI/IEEE Std 802.3 [ISO/IEC 8802-3]