TECHNICAL UNIVERSITY OF CRETE
DEPARTMENT OF ELECTRONIC AND COMPUTER ENGINEERING

PROPOSAL & EVALUATION OF A MULTIMODAL SERVICES PLATFORM OVER MOBILE DATA NETWORKS

Dimitrios Pratsolis & Nikolaos Tsourakis

Submitted in partial fulfillment of the
requirements for the degree of Master of Sciences
at the department of Electronic and Computer
Engineering

Committee

Prof. Vassilis Digalakis (supervisor)
Prof. Michael Paterakis
Assoc. Prof. Alexandros Potamianos

July 2006

TABLE OF CONTENTS

TABLE OF FIGURES

TABLES

AKNOWLEDGEMENTS

While working on my MSc I had relied on the support, encouragement, friendship and guidance of many people. So, I sincerely would like to thank several persons for their help during the accomplishment of this research.

First of all, I'm grateful to my supervisor, Professor Vassilios Digalakis, for his help, his valuable guidance and for our excellent collaboration, during these years.

I would also like to thank my friend and colleague Costas Harizakis for his important help regarding the subject of the interconnection between the final application and the speech recognition system.

Moreover, many thanks to my collaborator and friend, above all, Nikos Tsourakis for our excellent collaboration during this research and, mainly, for his patience and support to a certain health problem I faced during the implementation of the proposed architecture.

Last, but not least, I would like to thank my family, my father Stylianos, my mother Despina and my little sister Sofia for their support to everything I have ever decided to do during my life, and this Master Thesis can not be an exception. So, this work is dedicated to those people, who have given me the right to make dreams about my life…

Dimitris Pratsolis
Chania
July 2006

AKNOWLEDGEMENTS

As this work has finally reached to an end, I would like to express my deep appreciation and regard to a number of persons that contributed, one or the other way, in the completion of my thesis.

First and foremost, I should thank Professor Vassilis Digalakis for many things but especially for offering me the ability of aiming to higher grounds.

I also want to thank Professor Michael Paterakis and Associate Professor Alexandros Potamianos for the evaluation of the current work and their participation in the supervisory committee.

I should particularly thank my friend and colleague, Costas Harizakis for his remarks and suggestions during the completion of the thesis and for his irreplaceable contribution in overcoming specific implementation issues.

Last, but not least, I should thank Dimitris for sharing all the ups and downs during the process of the work and the joy of reaching to the end.

As always, any further acknowledgements to my family are redundant, to whom the current work is simply dedicated.

Nikos Tsourakis
Chania
July 2006

Chapter 1

GENERAL FRAMEWORK

1.1 Introduction

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 of cellular telephones (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 and conclude by conducting specific performance simulations.

1.2 Why Speech

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 was previously announced 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.

1.3 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 of 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 1-1).

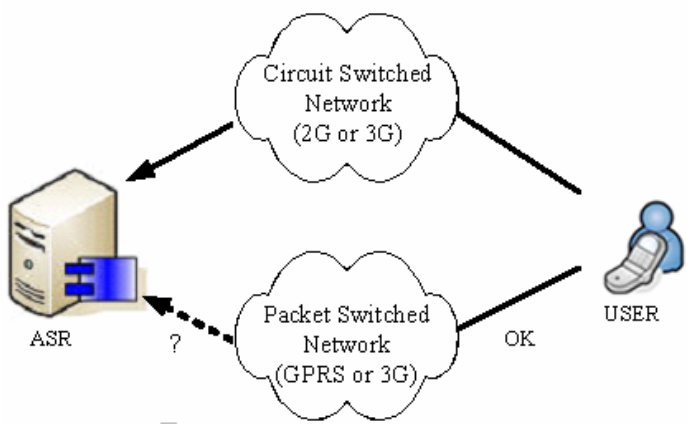

Figure 1-1: Recognition over data channel

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 use the data channel in order to accomplish speech recognition and subsequently speech application development on a mobile device.

1.4 Multimodal Interfaces

Multimodal interfaces process two or more combined user input modes - such as speech, pen, touch, manual gestures, gaze, and head and body movements - in a coordinated manner with multimedia system output. This class of systems represents a new direction for computing, and a paradigm shift away from conventional WIMP (Windows, Icons, Menus and Pointer) interfaces. This new class of interfaces aims to recognize naturally occurring forms of human language and behavior, which incorporate at least one recognition-based technology (e.g., speech, pen, vision). The development of novel multimodal systems has been enabled by the myriad input and output technologies

currently becoming available, including new devices and improvements in recognition-based technologies.

The growing interest in multimodal interface design is inspired largely by the goal of supporting more transparent, flexible, efficient, and powerfully expressive means of human-computer interaction. Multimodal interfaces are expected to be easier to learn and use, and are preferred by users for many applications. They have the potential to expand computing to more challenging applications, to be used by a broader spectrum of everyday people, and to accommodate more adverse usage conditions than in the past. Such systems also have the potential to function in a more robust and stable manner than unimodal recognition systems involving a single recognition-based technology, such as speech, pen, or vision.

The advent of multimodal interfaces based on recognition of human speech, gaze, gesture, and other natural behavior represents only the beginning of a progression toward computational interfaces capable of relatively human-like sensory perception. Such interfaces will eventually interpret continuous input from a large number of different visual, auditory, and tactile input modes, which will be recognized as users engage in everyday activities.

The same system will track and incorporate information from multiple sensors on the user's interface and surrounding physical environment in order to support intelligent adaptation to the user, task and usage environment.

1.5 Mobile Data Networks

The proliferation and development of cellular voice systems over the past several years has exposed the capabilities and the effectiveness of wireless communications and has paved the way for wide-area wireless data applications as well. The demand for such applications is currently increasing and, therefore, there is a strong need for advanced and efficient mobile data technologies.

Historically, wireless data communications was principally the domain of large companies with specialized needs; for example, large organizations that needed to stay in touch with their mobile sales force, or delivery services that needed to keep track of their vehicles

and packages. However, this situation is steadily changing and wireless data communications is becoming as commonplace as its wired counterpart.
The need for wireless data communications arises partially because of the need for mobile computing and partially because of the need for specialized applications, such as computerized dispatch services and mobile fleet management.
Mobile computing, which aims to migrate the computing world onto a mobile environment, is affected primarily by two components: portability and connectivity. Portability, i.e., the ability to untether computers from the conventional desktop environment, is getting increasingly feasible because with the continuous improvement in integration, miniaturization, and battery technology, the differences in performance and cost between desktop and portable computers is shrinking.
Regarding the connectivity, i.e., the ability to connect to external resources and have access to external data, wireless data technology plays a significant part because it can offer ubiquitous connectivity, that is, connectivity at any place, any time. For this reason, wireless data technology can be of real value to the business world since computer users become more productive when they exploit the benefits of connectivity.

1.6 Distributed Speech Recognition

The primary objective of speech recognition is to enable all of us to have easy access to the full range of computer services and communication systems, without the need for all of us to be able to type, or to be near a keyboard. By using a client/server approach in combination with the latest recognition systems, distributed speech recognition (DSR) will deliver the price/performance levels and access flexibility that will begin to make this practicable and affordable. As just one example of a spectrum of possible new applications, you will be able to dictate your meeting notes directly into your enhanced cellular handset immediately after a meeting, and the draft text will already be in your personal computer, ready for editing, by the time you return to your office (or hotel room, or home).
The performance of speech recognition systems receiving speech that has been transmitted over mobile channels can be significantly degraded when compared to

recognition in an office environment. The degradations are a result of both the low bit rate speech coding and the channel transmission errors. A Distributed Speech Recognition (DSR) system overcomes these problems by replacing the circuit-switched channel with an error-protected data channel to send a parameterised representation of speech, which is suitable for recognition. The processing is distributed between the terminal and the network. The terminal performs the feature-parameter extraction, i.e. the front-end of the speech recognition system. These features are transmitted over a data channel to a remote "back-end" recogniser. The end result is that the transmission channel does not affect the recognition system performance and channel invariability is achieved.

1.7 Scope of this Work

In this work we propose and implement a service that incorporates some of 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.
In particular, we investigate specific architectures that utilize current mobile network infrastructure, communication protocols and integration possibilities. We therefore propose architectures, which best fulfill our requests.
Furthermore, we implement of one of the architectures as a proof of concept and examine the interaction with a multimodal system. The specific implementation also provides a generic framework for creating applications of this kind.
In addition, we also study the effects of lossy data networks on speech recognition. We conduct several simulations with specific error models, considering the performance of the system and present the corresponding results.
Finally, within the context of our work we depict possible weaknesses and limitations and propose future enhancements and extensions.

1.8 Outline

The current thesis consists of eight chapters and one appendix, which are organized as follows:

- Chapter 2 serves as an overview of multimodal applications, first introducing the reader to the basic ideas and focusing on some of the issues associated with the specific applications. A brief reference on the corresponding standards is provided and related academic and commercial work is depicted.
- Chapter 3 provides some of the key elements needed in order to comprehend the nature of the evolving mobile data networks. As our work relies on the specific networks, it is essential to discuss the benefits of the IP convergence that already takes place in the mobile data networks and that will allow us to create effective and reliable multimodal applications.
- Chapter 4 presents the protocols incorporated during the development of our system and the protocols generally associated with our work. As the description of such protocols is beyond the scope of the current document, the specific chapter offers the basic ideas behind each one of them.
- Chapter 5 introduces the architectures of the proposed system by presenting their fundamental component, as well as the interconnection among them. Specifically, we present the functionality of each component and its role as part of the whole system.
- Chapter 6 demonstrates the Graphical User Interface (GUI) and its usage. In the context of our work, we implemented and present a stock information application.
- Chapter 7 provides the experimental evaluation of our work. Specific error models were implemented and applied in a specific test set. Evaluation metrics were incorporated and graphs with the result of the process are also provided.
- Chapter 8 serves as an epilogue of our work by presenting the conclusions of the thesis and proposing future extensions.
- The single appendix of the current document provides the detailed specification of all available messages exchanged between specific components of our system and can be used as a reference from the reader.

1.9 Contribution

As the current work is a joint effort of the two authors, Dimitris Pratsolis and Nikos Tsourakis, we should make clear the contribution of each one of them during the progress of the thesis. Specifically, as we can see in the following table, each contributor participated and implemented a number of tasks. The specific tasks concerned the planning and the roadmap of the thesis, the implementation of the proposed system, the experimental evaluation and the writing of the current document. In particular:

D.Pratsolis	**N. Tsourakis**
Planning	**Planning**
Joint effort	
Implementation	**Implementation**
▪ Communication Channel ▪ Audio Manager ▪ HTTP Interconnection ▪ Database Creation & Configuration ▪ ASR Configuration – Recognition Grammars	▪ Dialog Manager ▪ GUI ▪ Mediator Server – ASR Integration ▪ Stock Information Application
Experimental Evaluation	**Experimental Evaluation**
▪ Three-State Markov Model	▪ Gilbert-Elliot Model
Writing	**Writing**
▪ Chapter 3 ▪ Chapter 4 ▪ Chapter 5 (5.1, 5.2, 5.4.1, 5.4.4) ▪ Chapter 6 (6.2) ▪ Chapter 7 (7.3, 7.5, 7.6.2, 7.7) ▪ Chapter 8	▪ Chapter 1 ▪ Chapter 2 ▪ Chapter 5 (5.3, 5.4.1, 5.4.2, 5.4.3) ▪ Chapter 6 (6.1, 6.3) ▪ Chapter 7 (7.1, 7.2, 7.4, 7.6.1) ▪ Appendix

Table 1-1: Contributions

Chapter 2

MULTIMODAL INTERFACES

2.1 Overview

A critical factor of bringing the benefits of networked computers to mass markets is to provide a more natural way of communicating with machines or with other people via machines. Multimodal interfaces capitalizing on the efficient combination of visual displays and speech input, will overcome the limitations of current voice interfaces and mobile devices. These interactions will enable users to speak, write, and type, as well as hear and see using a more natural user interface than today's single-mode interfaces.

With the growing popularity of mobile phones and Personal Data Assistants (PDAs), mobile information access and remote transactions are fast becoming commonplace. However, mobile devices impose their limitations on the end-user experience. For example, mobile phones have relatively small visual displays and a cumbersome keypad input. PDAs have better visual displays, but have the same input limitations. Voice browser applications for mobile devices offer ease of input, but their inherent ephemeral quality limits their use as an output medium. The user interface of the computer, however, must exploit to the senses of sight, sound, and touch that people use to navigate in their physical surroundings.

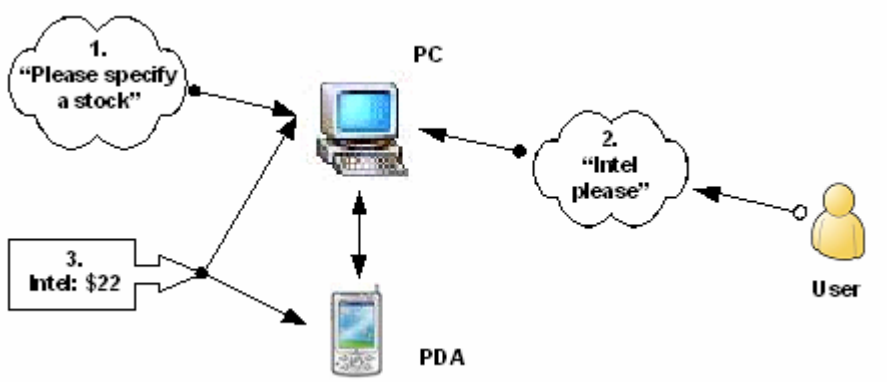

Figure 2-1: Multimodal application example

An example of a multimodal application for mobile stock trading is shown in Figure 2-1. This example demonstrates how your experience is more natural and personalized, as you are able to capitalize on the ease of input of voice browsers and also view detailed information on a visual display in the same session. For example, you could be talking to your PC and continue the session on your PDA after leaving the office. You can exploit the advantages of the different input/output mechanisms working in tandem instead of being constrained by their limitations. Additional features such as speaker verification, interactive displays of visual information (e.g. charts), audio/video notification, etc. can make the interaction more natural. The users are not restricted to a particular user interface but are able to exploit the advantages of multiple synchronized interfaces.

2.2 Multimodal Applications

Multimodal applications represent the convergence of content, video, audio, text and images, with various modes of user interface interaction. This enables us to interact with an application in a variety of ways: input with speech, a keyboard, keypad, mouse and/or stylus, and output such as synthesized speech, audio, plain text, motion video and/or graphics.

The term "mode" denotes a mechanism for input and/or output to a user interface. You can employ each of these modes independently or concurrently. Multimodal applications incorporate any number of modes simultaneously so you can vocalize your name, type in an address, send a phone number from a wireless handset, all within the same session,

form, and application context. The browser will let you select the most appropriate mode of interaction based on your situation, activity, or environment.

The different modes may be supported on a single device or on separate devices working in tandem. Voice may also be offered as an adjunct to browsers with high resolution graphical displays, providing an accessible alternative to using the keyboard or screen.

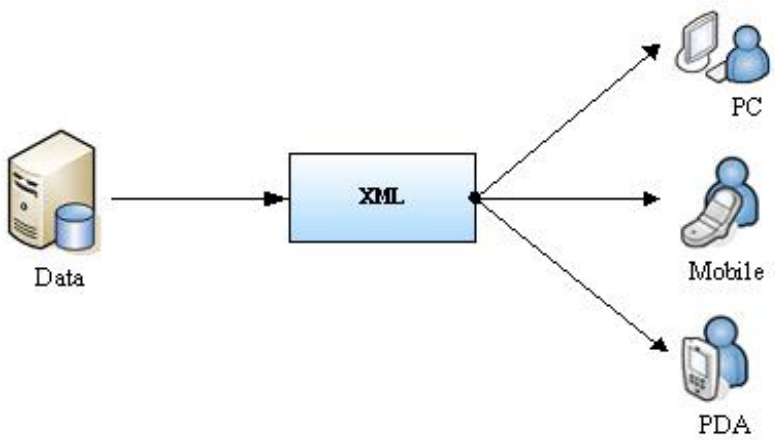

Figure 2-2: Multimodal Applications – Multiple modes, Unified Content and Representation

Multimodal applications are an improvement over multichannel applications. The term "channel" refers to the different browsing platforms or user agents that access, browse, and interact with online applications. Multichannel applications are designed for universal access across different channels, one channel at a time, with no particular attention paid to synchronization or co-ordination among the different channels. You have an array of channels in order to access Internet content, which appears separate but functional and consistent.

However, multichannel applications have their own drawbacks. Each channel must be developed from the ground up using different languages and processes as appropriate. Only certain components, such as the back-end database, can be shared among the different channels. As the range of devices grows and the number of available channels increases, the situation demands more resources and time.

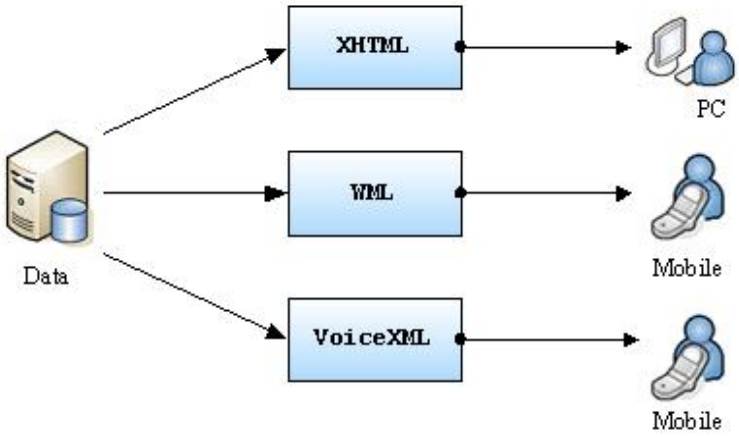

Figure 2-3: MultiChannel Applications – Multiple Channels, Multiple Representations, Unified Content

To obviate this problem, one can have a common language to control user interaction independent of the channel. However, to ensure that the implementations for each channel are interlinked, the highest level of commonality must be implemented, which may lead to a sacrifice of desirable features. A better answer is multimodal applications, where multiple channels are available and synchronized.

2.3 Advantages of Multimodal Applications

Some of the advantages of multimodal applications are depicted below:

- Multimodal interfaces improve the usability of data services such as weather, driving directions, stock quotes, personal information management, and unified messaging. Application Service Providers can provide a wider range of personalized and differentiated offerings using multimodal interfaces.
- Many call center applications and enterprise data services such as account management, brokerage accounts, customer service, and sales force automation offer voice-only interfaces. Multimodal interfaces added to these applications should enhance the users experience.

- With multimodal interfaces, you can easily access and enter information, especially when using small devices by combining multiple input and output devices.
- Multimodal applications may improve your experience with mobile devices and encourage the growth and acceptance of m-Commerce.
- The user is not constrained by the limitations of a particular interaction mode at any given moment. For example, while listening to instructions on a Voice browser, he/she is constrained by the ephemeral nature of the interface and may wish to listen to the instructions again. Multimodal interfaces offer the flexibility to choose the most convenient interaction mode that suits the task and purpose, and can also exploit the resources of multiple interfaces in order to have an enhanced computing experience.

2.4 Issues associated with Multimodal Applications

Some of the issues associated with multimodal applications are:

- Standards for multimodal applications are just being developed.
- Organizations such as W3C and the SALT forum have proposed specifications to be adopted as a standard.
- The current bandwidth and capacity constraints of the current 2G wireless networks makes the development of bandwidth-hungry multimodal applications challenging. The current adoption of 3G wireless networks should ensure seamless convergence of wireless technology and the Internet, and a more widespread adoption of multimodal applications.
- When you are using multimodal interfaces you could face ergonomic issues as you switch from one mode to another, such as alternating between listening and watching. The developers of multimodal applications should be careful to avoid feature bloats with the various interfaces. Introducing contextual cues during mode switching results to a better user experience.

- Synchronization is one of the biggest problems confronting multimodal application development. Synchronization is necessary to prevent collisions between modes, to capture/handle events, and to unify server-client interaction. The standards being developed introduce the need for a limited, unified event model and synchronization between different markup languages.

2.5 Standards with Multimodal Applications

The W3C Multimodal Interaction working group is developing specifications to extend the Web user interface to allow multiple modes of interaction, offering the choice of speaking or employing a key pad, keyboard, mouse, stylus, or any other input device. For output, the user will be able to listen to spoken prompts and audio, and view information on graphical displays. The working group is developing markup specifications for synchronization across multiple modes and devices. The specifications build on top of existing W3C specifications such as XHTML [68], VoiceXML [23], SMIL [69], etc. They propose to use W3C DOM [68] to create interoperable content for various end-user devices and DOM2 [72] events model for synchronization among the modes.

The SALT forum is developing the Speech Application Language Tag (SALT) [22] specification. SALT is a lightweight set of extensions to HTML [73], WML, and XHTML that enable multimodal and telephony access to information, applications, and Web services from PCs, telephones, tablet PCs, and wireless personal digital assistants (PDAs). SALT adds a spoken dialog interface to Web applications for both voice-only browsers (e.g., over the telephone) and multimodal browsers. You can add SALT to a visual page to support speech input and/or output. For applications without a visual display, SALT manages the flow of the dialog and the extent of user initiative by employing the HTML event and scripting model. In this way, the full programmatic control of client side (or server-side) code is available to application developers.

2.6 Related Work

Basic-research projects in the area of multimodal systems have often focused on elaborate system architectures for advanced forms of multi-modality. For example some of the related work includes the following systems:

- Galaxy Communicator [59] is an open-source architecture for constructing dialogue systems. Its plug-and-play approach enables developers to combine architecture-compliant commercial software and cutting-edge research components. Galaxy Communicator was funded by the Defense Advanced Research Projects Agency (DARPA) of the United States Government. The DARPA Communicator program was designed to support the creation of speech-enabled interfaces that scale gracefully across modalities, from speech-only to interfaces that include graphics, maps, pointing and gesture. Although the DARPA Communicator program has concluded, the open source software and documentation are still available on http://communicator.sourceforge.net/.
- The WITAS [60] dialogue system for multi-modal conversations with autonomous mobile robots. Dialogues supported by the specific interface are mixed-initiative, open-ended, and multi-modal, over a dynamic operating environment. A general point of distinction between the system and many others is that it is not restricted to plan-based dialogues. In other words, paths through dialogues need not be specified in advance, as is necessary in many other systems. The WITAS multi-modal dialogue interface interprets spoken language and map-gesture inputs as commands, queries, responses, and declarations to the UAV, and generates synthesized speech and graphical output to express the robot's responses, questions, and reports. Current dialogue capabilities include ambiguity resolution, presupposition checking, processing of anaphoric and deictic expressions, command revision, report generation, and a confirmation backchannel (http://www-csli.stanford.edu/semlab/witas/demo1/).
- QuickSet [61] is a collaborative, handheld, or wearable multimodal system for interacting with distributed applications. In virtue of its modular, agent-based design, QuickSet has been applied to a number of applications in a relatively short

amount of time. Begun as SIMNET in the 1980's [62], distributed, interactive simulation-based training environments attempt to provide a high degree of fidelity in simulating combat equipment, movement, atmospheric effects, etc. One of the U.S. Government's goals, which has partially motivated the specific research, is to develop technologies that can aid in substantially reducing the time and effort needed to create large-scale scenarios. QuickSet addresses two phases of user interaction with these simulations: creating and positioning the entities, and supplying their initial behavior. In the first phase, a user "lays down" or places forces on the terrain, which need to be positioned in realistic ways, given the terrain, mission, available equipment, etc. In addition to force laydown the user needs to supply them with behavior, which may involve complex maneuvering, communication, etc. A major design goal for QuickSet is to provide the same user input capabilities for handheld, desktop, and wall-sized terminal hardware.

- SmartKom [63] is a multimodal dialog system that combines speech, gesture, and mimics input and output. Spontaneous speech understanding is combined with the video-based recognition of natural gestures. One of the major scientific goals of SmartKom is to design new computational methods for the seamless integration and mutual disambiguation of multimodal input and output on a semantic and pragmatic level. SmartKom is based on the situated delegation-oriented dialog paradigm, in which the user delegates a task to a virtual communication assistant, visualized as a life-like character on a graphical display. SmartKom is based on a multi-blackboard architecture with parallel processing threads that support the media fusion and media design processes.

These large and often multi-year projects were important because they provided architectural foundations for sophisticated types of multi-modal integration. On the other hand, for resource-limited mobile devices downward scalability from the often generous resource assumptions in the demo applications of these basic-research projects is sometimes problematic.

There have been several research projects for focusing on telecom applications like:

- The "MUST" project [64] is an acronym for Multimodal and multilingual Services for small mobile Terminals. The single most important goal of MUST is to clarify some of the fundamental usability issues that are involved in multimodal and multilingual services when there is no keyboard for input. Another objective is to obtain a better understanding of the role that language and speech technology will play in future multimodal and multilingual services in the mobile networks Moreover the project aimed to get hands-on experience by integrating existing speech and language technologies into an experimental multimodal interface, in order to conduct human factor experiments with "real" users to assess the value of the language and speech technologies for fast, simple and user-friendly interfaces on small mobile devices (http://www.eurescom.de/public/projects/P1100-series/p1104).
- The SMADA project [65], which stands for Speech-driven Multimodal Automatic Directory Assistance. SMADA aims to improve the efficiency and effectiveness of human language technologies embedded in automated telephone and web-based directory inquiry services. While current technologies offer significant potential cost benefit advantages, their impact on customer satisfaction and customer perceptions of service quality are not as well understood. Project results will be field tested in demonstrators for automated voice and multimodal access directory services via fixed or mobile "phone" networks and the web. Research will be focused on improving the robustness and accuracy of automatic speech recognition (ASR) technology, and on human factor issues in caller-system and operator-system interaction.
- At the core of the MONA project [66] is the concept of multimodality. The MONA server supports the modalities speech, text, graphics and non-speech audio. The MONA presentation enables a new class of mobile applications providing powerful and flexible user interfaces on a wide range of networked devices in a telecom network. A carrier or third-party service provider who deploys the MONA presentation server can deliver end-user services that combine speech and visual interface techniques for single-user or multi-user

applications. MONA applications offer a unique and recognizable user experience that combines several familiar user interface paradigms (http://mona.ftw.at/ monawork.html).

A number of companies are already active in this area that offer solutions to enable multimodal interfaces on mobile devices and in the automotive environment. For example:

- Nuance's X|mode Multimodal System enables a new breed of applications that deliver simple, powerful user interactions on personal wireless devices. Leveraging server-based ASR and TTS technologies, X|mode Multimodal System enables rapid development and deployment of true multimodal applications, blending visual, keypad, and speech capabilities to deliver fluid, efficient access to information systems through voice commands, touch controls, or a combination of both. (http://www.nuance.com/xmode/).
- The Multimodal Toolkit of IBM lets you use standards-compliant combinations of existing languages to create a multimodal application by adding snippets of speech markup to the XHTM visual markup. The voice portion of a multimodal application uses VoiceXML, providing an easy way to manage a voice dialog between a user and a speech recognition engine. XHTML+Voice, informally known as "X+V", is the markup language used to create multimodal applications. Mobile devices are targeted for particular uses. They support the features they need for the functions they are designed to fulfill. A multimodal authoring system lets you speech-enable the visible elements of a visual interface, which has particular importance for mobile devices (www.ibm.com/pvc/ multimodal).
- As a wireless Personal Digital Assistant (PDA), MiPad of Microsoft [67] fully integrates continuous speech recognition (CSR) and spoken language understanding (SLU) to enable users to accomplish many common tasks using a multimodal interface and wireless technologies. It tries to solve the problem of pecking with tiny styluses or typing on minuscule keyboards in today's PDAs or smart phones. It also avoids the problem of being a cellular telephone that

depends on speech-only interaction. MiPad incorporates a built-in microphone that activates whenever a field is selected. As a user taps the screen or uses a built-in roller to navigate, the tapping action narrows the number of possible instructions for spoken language processing. MiPad currently runs on a Windows CE Pocket PC with a Windows 2000 Server where speech recognition is performed. The Dr Who CSR engine has a 64k word vocabulary with a unified context-free grammar and ngram language model. The Dr Who SLU engine is based on a robust chart parser and a plan-based dialog manager. (http://research.microsoft.com/srg/mipad.aspx).

- Kirusa's multimodal solution for IP Voice is a simultaneous multimodal solution that allows any application to be accessed using a combination of voice and visual modes at the same time. The simultaneous multimodal solution consists of a multimodal platform (KMP-IPV) and multimodal clients (KMC-IPV) for mobile handsets. The multimodal platform is the "heart" of the multimodal solution. Functioning like a "multimodal gateway", the platform ties together the activities of the client, application, and the voice browser to realize a multimodal session based on the multimodal markup language. The voice and data connections run simultaneously over an IP connection between the platform and the client. This mechanism enables KMP to utilize a variety of wireless networks including GPRS, Edge, UMTS, CDMA 1X/3X, EV-DO, Flash-OFDM, and 802.11x. The platform is optimized to provide fast response time over low bandwidth wireless networks, such as GPRS. Multimodal clients are available or under development for several smart devices (http://www.kirusa.com/products_kms_ipv.php).

Our work focus on the proposal of distributed architectures for mobile multimodal application development, which utilize existent infrastructure and available standards. The aim is to offer a transparent way of integration with different types of commercially available ASRs and not be restricted by using a specific ASR system. We therefore propose an integration layer that is simple to implement and that targets to resource limited mobile devices, with low memory, CPU and network requirements. The

implemented prototype can also be used for studying the human-computer interaction on a mobile device.

2.7 The Future

Multimodal applications are set to grow in importance in the coming years, bringing benefits to businesses, developers, and end-users. With upcoming technology, devices from the desktop computer to the handheld PDA, from the automobile to the cellular phone, will be able to support multiple modes of access and communication allowing them true anyplace, anywhere, anytime access. Multimodal applications will be the key component in making anyplace, anywhere access more convenient and real, and they will allow you to choose the most suitable form of input and output no matter what the situation.

A distributed multimodal framework involves multiple devices and servers. The Session Initiation Protocol (SIP) [53] is emerging as the preferred choice for initiating and controlling sessions among the multiple devices and servers. The SIP protocol can be used to synchronize several devices, for instance, to update the display on a PDA or desktop in concert with a smaller display on a cell phone. Along with server-side scripts, SIP seems to provide an effective solution for session management, event handling and synchronization among the various modes of a Multimodal application.

To overcome the processing limitations of mobile devices, a solution is to process speech recognition and synthesis remotely on more powerful platforms using Distributed Speech Recognition (DSR). You could use DSR for complex voice dialogs requiring natural language understanding. Simple dialogs could be handled locally, but for richer interaction it will be necessary to couple the device with a remote DSR engine.

The best use for multimodal applications will be in the next generation wireless networks. The 2G and 3G wireless networks promise greater bandwidth, always-on connections, and simultaneous voice and data channels with reduced latency. Also, newer mobile devices have embedded Automatic Speech Recognition interfaces to facilitate speech input. The PC already has very good input and visual display facilities; so multimodal applications will be, at best, a secondary convenience for the desktop.

The ultimate goal for multimodal applications will be to create flawless and completely natural interactions for end-users. That goal is still far off, but technology is evolving rapidly. The future of human-computer interaction will probably involve direct dialog in a world where natural interfaces are used as commonly as the mouse and the Visual Display Unit (VDU).

Chapter 3

MOBILE DATA COMMUNICATIONS

3.1 Introduction

IP convergence is the fusion of mobile and Internet domains, using packet-based networks as a common infrastructure, to enrich the way we communicate. With IP protocols it is possible to enrich the end-user experience by combining voice, video, text, and content seamlessly in person-to-person communication. This experience is further enhanced by bridging the existing gap between mobile and Internet domains and enabling access to services independent of the type of access (e.g., General Packet Radio Service [GPRS], LAN, WLAN).

Connectivity is becoming more widespread and diverse. Wireless access technologies such as Wideband CDMA (WCDMA), CDMA, GSM Enhanced Data Rates for Global Evolution (EDGE), and WLAN are emerging in a broad range of products, and the bandwidth provided by these new access types is continuously increasing. At the same time, the use of broadband connections in homes is growing rapidly. All of these access technologies will have one thing in common — IP. With IP as a common platform, the various access technologies can be seamlessly combined to create a completely new user experience.

A key element in IP convergence is the end-user experience: the value and potential of combining availability with rich conversation and sharing of content to provide enriched services. Even in the world of IP convergence, end users will still perceive person-to-person communications as the most important service. From the end user's perspective, the primary benefit of IP in communications is integration and interaction of services along with network-independent addressing. Voice, video, text, content sharing, and presence are all components of personal interactive communications, which can interact with each other to create services that are both beneficial and easy to use. Therefore, the ability to use one common protocol for all services will offer significant benefits to the end users.

Brand new applications are already beginning to emerge now that the convergence of IP networks is starting to happen. For example:

- Push-to-talk over Cellular (PoC) is a new voice application enabled by SIP [53] and 2G GPRS networks, i.e. two-way, one-to-one private calls and one-to-many group call services similar to a walkie-talkie service.
- Real-time video sharing is an emerging application for sharing live video or video clips in real time during a normal voice call, also enabled by SIP, but in 3G networks.

Other interesting applications will emerge as developers around the world are provided with the tools to develop SIP-based applications to run in converged networks.

To maximize the true potential of IP convergence, the highly innovative developer community must be actively involved. Its indisputable creativity must be harnessed by being given access to the tools, in the form of open APIs and SDKs, to create SIP-based applications. Applications can be quickly developed due to the reuse of common IP protocols, and quickly deployed by utilizing the common connectivity features of SIP to establish connections.

In the following sections, key issues in IP convergence will be highlighted. The concept of IP convergence and the reasons driving it will be discussed. Also, the concepts of real-time video sharing, PoC, and other services, as different components of personal communications and the technology behind these services will be covered.

3.2 Person-to-Person Communications

In the world of IP convergence, services and applications will be redefined. Applications will no longer be isolated entities exchanging information only with the user interface. The next generation of applications will involve peer-to-peer connectivity between IP-capable terminals. The ability to establish a peer-to-peer connection between new IP-enabled mobile devices is the key ingredient for enabling richer person-to-person communication. Once an IP-level connection is established between users, there are a whole range of IP protocols that can be used to exchange information and experiences between end users.

In order to communicate, IP-based applications must have a mechanism to reach the correspondent. Today, fixed and mobile telephone networks perform this critical task of establishing a connection. By dialling the other user's telephone number, the network can establish an ad hoc connection between any two terminals. This critical connectivity capability still does not exist widely in the Internet.

SIP-based session management complemented by critical mobile network capabilities (i.e., authentication, roaming, and network interconnectivity provided by the 3GPP IP Multimedia Subsystem [IMS]) is the required service infrastructure. With such a system in place, it is possible to establish an IP connection between two terminals. Once the connection is established, the IP connection can be used to exchange all types of communication media (voice, video, content, etc.).

3.3 Fast, Easy Service Creation

The phenomenal growth of the Internet and the Web has put the spotlight on IP-based networks and the inherent advantages of IP technology.

In IP convergence networks, the creation platforms are also based on IP protocols. All services, including communications services, are created with Internet-based protocols and languages. The benefit is that when all services are created with open and standard IP protocols, the creation and integration of these services are much faster and easier than those based on telecom protocols.

One of the key benefits of IP convergence is the possibility of giving developers the opportunity to develop new applications. SIP-enabled person-to-person applications can be built on open SIP APIs. This will allow new person-to-person applications to be developed and deployed with speeds not previously possible.

Operators are in the best position to offer added value to their subscribers. First, they have the subscriptions and profile data of the users. Second, they own the provisioning and billing machinery. Third, they control the end-to-end QoS for services. And fourth, they have access to the mobile-network-specific enablers such as location and presence information with the user's permission.

Service creation will occur in three identified domains:

- Network operators. Network operators can offer services that fully exploit the capabilities of their IP network and service capability servers integrated to the solution. The innovation of third-party developers can be used to quickly develop and deploy new, exciting services.
- Third-party service providers. Third-party service providers or corporations can use SIP capabilities by interfacing with the operator network server using an external application server. It is also possible for a third-party service provider to operate a SIP application server itself.
- End users. End users can create services by using the capabilities offered by an operator or a third-party service provider. For example, a Web interface capable of building up services using a simple Call Processing Language (CPL) could be offered to end users.

Since the underlying elements of SIP are so similar to HTTP [37], creating network-based services such as time-dependent call forwarding is quick and straightforward. Millions of software developers can design and implement new SIP-based voice services just as quickly and easily as they develop Web pages.

By not requiring major hardware upgrades to application servers, but rather enabling new software-based services, service providers can reduce the time associated with deploying new features from months to days. This means ever-improving communications services for subscribers.

3.4 Connectivity and Service Mobility Increase User Benefits

The number of access technologies is continuously growing. Differences such as wide and local area mobility vs. fixed-access methods will remain. Although from a broad perspective all networks converge into a single IP network, different networks inherently have some characteristics that differ. Securing the best quality interaction between networks is needed for seamless end-to-end services.

Service mobility means the accessibility of a person's personalized services and his/her personal reachability through any access network or terminal. User interfaces and services are independent of time and place, but are at the same time customized according to the terminal used and are aware of place and time. For example, a session (e.g., a videoconference) could be started on a PC, continued on a mobile terminal, and concluded on a television screen at home.

When all services (both browsing-type mobile Internet services and person-to-person communication services) are IP based, they can be offered through any access network – the operator can create services in one platform and offer them via multiple-access networks, thus greatly increasing the addressable customer base and serving the existing base better. Services that are accessible wherever, whenever, and through whichever device will increase customer loyalty and reduce churn. Service mobility and personal reachability is achieved with SIP and IPv6 Mobility. IPv6 gives a device a universal address, enabling it to be reached from any network.

End users benefit from SIP by having the same address while choosing the most suitable interface —mobile terminal, PC, PDA, etc. While SIP is the generic protocol over all access networks, end users may use the most capable access network available, for example in terms of price, bandwidth, usability, etc. Services are also personalized, which makes the user experience appealing.

3.5 New Business Opportunities

Applications in IP convergence can no longer be viewed as stand-alone. End users will undergo rich experiences due to several components interacting with each other. From

the end user's perspective, the primary benefit of IP in communications is integration and interaction of services. Voice, video, instant messaging, and presence are all different aspects of personal interactive communications, each of which can also interact with the others to create services that are both beneficial and easy to use.

Historically, service integration has been difficult in mobile networks, since they all required their own protocols, standards, and service domains. In the IP convergence era, service integration is easy, since services are created within one domain. It is easy to integrate voice-based applications with content sharing applications, mobile terminal applications with PC applications, and so on. The possibilities are endless. In IP convergence networks, the operator needs only one network to take care of, while SIP remains the enabling protocol.

IP convergence provides opportunities for new revenue streams by enabling new services that were not previously possible or were too complex and expensive to implement. The IP convergence infrastructure enables new flexible billing models, that weren't possible before. The same charging model — price per minute and message — can be maintained for voice telephony and messaging, i.e., for non-real-time applications. Initially, there may be more package-based billing or service combinations. Package-based billing is the most suitable for real-time applications such as SIP-based instant messaging.

The management and operational costs of running a communications service are vastly reduced by using SIP, since new functionalities, such as real-time video sharing, presence, and instant messaging can be added easily to an existing SIP network, avoiding the need to run and build a completely separate, parallel network to support each feature. The equipment costs themselves are reduced, since only one set of proxy and location servers are needed, rather than two. System capacity is improved, because resources can be shared across many services. Infrastructure investments in firewalls, mobility services, etc., which have been put in place for supporting SIP, can also be reused.

3.6 New Paradigm Based on Previous Generation's Legacy

IP convergence builds on well-established service paradigms, for example Short Message Service (SMS) and Multimedia Messaging Service (MMS), by adding new functionality

and new content and service types in user-understandable steps. Because end users can relate to the new services as enhanced service experiences from existing networks, the barriers for adopting them will be significantly reduced, leading to rapid adoption and high penetration.

The market will develop towards IP Convergence services as an evolution. SMS has been the main data service so far, but today MMS, browsing, downloading and other services are taking off rapidly. The service offering will further evolve in the IP convergence era with real-time services such as real-time video sharing and Push to talk over Cellular.

In the development of third-generation mobile networks and beyond, the focus has shifted towards the IP paradigm. In particular, the work so far has focused on the adoption of IP transport, IP mobility, and VoIP protocols (e.g. SIP) in mobile networks.

In the future, IP will allow the provisioning of multimedia services and of other services, such as instant messaging, Web browsing, etc., in an integrated fashion. In particular, users will no longer be forced to see the cellular network as capable of providing only voice services or Internet access. Current attempts to evolve the service infrastructure have simply tried to apply the current concepts and services to basic services, e.g., voice services, without considering the importance that service integration will assume.

An IP-based service architecture, capable of supporting multimedia and service integration, will be a key element of future cellular networks. Also, current service architecture models, as they are evolving at present, do not meet the requirements for service integration, from a technology point of view. Therefore, a distributed IP-based technology architecture approach is needed. In practice, this means the reuse of existing Internet Engineering Task Force (IETF) protocols, when possible, which will allow for an easier deployment and acceptance of the service architecture, both in the cellular and IETF environments.

3.7 Open Standards

Open standards are essential to enable mass deployment of mobile multimedia services and to prevent market fragmentation, by maximizing the number of potential customers using compatible solutions. This applies to all players in the mobile service business —

the content industry, application developers, operators, and manufacturers alike. Open standards cooperation is done in standardization bodies such as the 3rd Generation Partnership Project (3GPP), 3rd Generation Partnership Project 2 (3GPP2), Internet Engineering Task Force (IETF), and Open Mobile Alliance (OMA).

When multimedia applications are implemented on embedded and constrained devices such as mobile devices, it is important to optimize the number of codecs and formats that need to be supported. This has been a leading design constraint for the 3GPP team, which made sure that the selection of codecs and file format for MMS and PSS (Packet-switched Streaming Service) is aligned. In this way, interoperability between MMS and PSS content and multimedia architecture, as well as future multimedia applications, will be ensured.

3.8 IPv6

The widely deployed version of the Internet Protocol, IPv4, was designed for a much smaller number of Internet hosts than the Internet is experiencing today. The IPv4 address space problem will become even greater when hundreds of millions (or billions) of cellular handsets are connected to the Internet. Insufficient IPv4 address space has been one of the main drivers to specify IP version 6 (IPv6). IPv6 is a mature technology that will be adopted in cellular networks and terminals, and is implemented by various manufacturers today. The most important benefit of IPv6 is the very large address space.

The IPv4 address space has been extended by the use of private IPv4 addresses together with Network Address Translation (NAT). Use of private IPv4 addresses and NATs is very common in GPRS networks today. However, NAT breaks the end-to-end nature of the Internet, and restricts the services used in the Internet. A public IP address is especially important for SIP-based peer-to-peer services, such as real-time video sharing and gaming. The general IPv4 NAT cannot be used for SIP-based services. This is due to the fact that in SIP, the signaling and the media stream follow a different route. The SIP signaling carries IP addresses for the inbound media streams within the Session Description Protocol (SDP) [50]. In the case of end points behind IPv4 NATs, the IP

addresses that are exchanged within the SDP are the private IP addresses that are only valid within the address space of the NAT.

In summary, IPv6 provides a robust, future-proof platform for SIP-based services. SIP and IPv6 enable users to originate, receive, and access services from any end node and any location. From the end-user's point of view, IPv6 and SIP enable ubiquitous communication in person-to-person, as well as client/server environments. For services this means increased availability and usage, while more devices are capable of being online.

3.9 Quality of Service (QoS)

In short, Quality of Service (QoS) means providing a consistent, predictable data delivery service. In other words, satisfying customer application requirements. QoS refers to the ability of a communication system to provide an appropriate transport service for delivering relevant traffic stream types of different users. It can sometimes be difficult to define the exact technical parameters required to ensure such delivery, because the end user's perceptions of a "service quality" may differ from one user to another. QoS only manages bandwidth according to application demands and network management settings, and in that regard it cannot provide certainty if it involves sharing. Hence, QoS with a guaranteed service level requires resource allocation to individual data streams. Clearly, QoS is an end-to-end issue, as it affects the terminal as well the network solutions. Any QoS assurances are only as good as the weakest link in the "chain" between sender and receiver.

More often, applications are based on quality, reliability, and timeliness assurances. In particular, services that use voice and video streaming, e.g., VoIP and real-time video sharing, are quite demanding in this sense. Operators can differentiate themselves from their competitors by offering different kinds of service packages that have different kinds of QoS levels. The requirements for transport service vary according to application stream types and end users.

In the future, it will be possible to broaden the scope of these QoS packages into a more application-driven one. QoS may be a parameter of the application, with the actual

service created around that application. Then, if an end user decides to buy that particular service, he/she can do so whether or not he/she belongs to a certain service-level package (provided his/her terminal and access mode support the required QoS).
In summary, QoS does not create additional bandwidth. Instead, it manages the existing bandwidth in such a way as to allow for efficient transportation of traffic through the network, allowing for predictable delays, delay variation (jitter), and packet loss rates.

3.10 Applications in IP Convergence

IP will greatly enhance person-to-person communications by combining multiple media types and communication methods with presence and community information, not only person-to-person but also instant multipoint-to-multipoint, i.e. conferencing. With IP, adding "richer media" (video streams, live video, or application sharing), presence (availability, preferred media type), or community information is straightforward, as all the services use the same communication protocol (IP) and signaling (SIP).
Whereas voice, video and messaging services will remain to be the basis for communication they will be used in a more flexible manner, complemented by presence, location or other context-related information. The ultimate target is described in the rich call concept in which the building blocks for on-line conversations will be integrated to make a seamless end-user experience.

Chapter 4

INTEGRATED PROTOCOLS

4.1 Overview

Network communications are defined by network protocols. A network protocol is a formal set of rules, conventions and data structures that governs how computers exchange information over a network. In other words, a network protocol is a standard procedure and format that two data communication devices must understand, accept and use to be able to talk to each other.

Many standard organizations worldwide and technology vendors define network protocols over years of technology evolution and developments. Protocols can be grouped into different categories like:

- Signaling Protocols, to establish presence, locate users, set up, modify and tear down sessions (e.g. SIP/SDP, H.323).
- Media Transport Protocols, for transmission of packetized audio or video (e.g. UDP, RTP).
- Supporting Protocols, for gateway location, QoS, interdomain Authentication, Authorization, Accounting, address translation, IP, etc (e.g. DNS, RSVP).

The enumeration and the grouping of protocols can sometimes be cumbersome, as we can see in Figure 4-1.

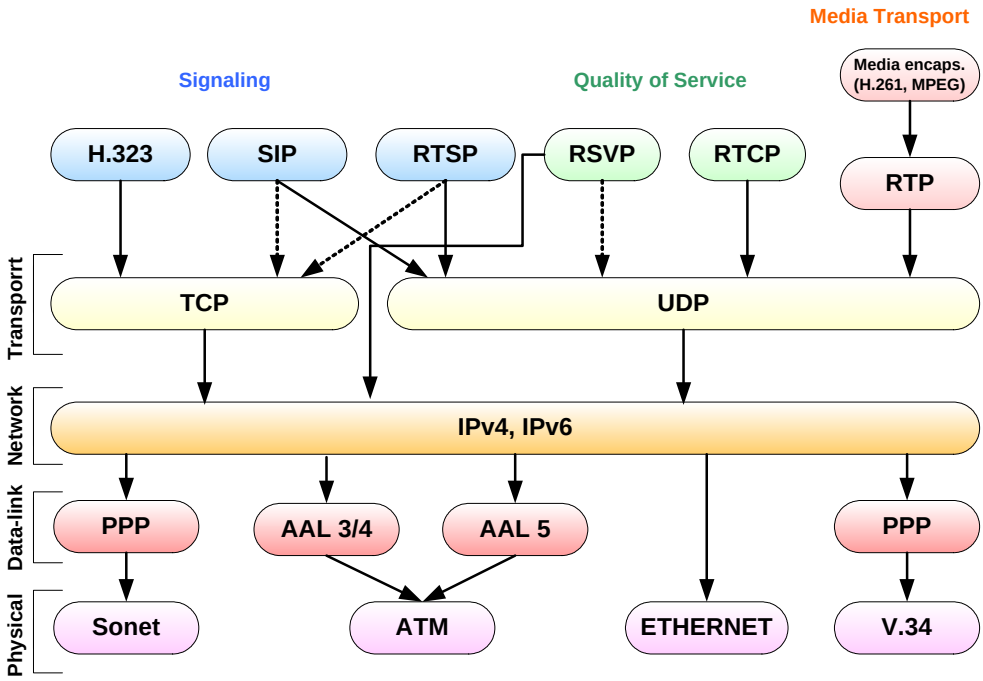

Figure 4-1: Protocol Zoo (Source: http://www.cs.columbia.edu/~hgs/internet/)

In the sections that follow, we will provide a brief overview of the protocols that we integrated in our implementation or were included in our proposed architectures.

4.2 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) [28] is the transport layer protocol in the TCP/IP protocol suite, which provides a reliable stream delivery and virtual connection service to applications through the use of sequenced acknowledgment with retransmission of packets when necessary. Along with the Internet Protocol (IP) [71], TCP represents the heart of the Internet protocols.

Among the services TCP provides are stream data transfer, reliability, efficient flow control, full-duplex operation, and multiplexing.

TCP offers reliability by providing connection-oriented, end-to-end reliable packet delivery. It does this by sequencing bytes with a forwarding acknowledgment number that indicates to the destination the next byte the source expects to receive. Bytes not acknowledged within a specified time period are retransmitted. The reliability mechanism of TCP allows devices to deal with lost, delayed, duplicate, or misread packets. A time-out mechanism allows devices to detect lost packets and request retransmission.

4.3 User Datagram Protocol (UDP)

UDP [29] is a connectionless transport layer protocol, which provides a simple and unreliable message service for transaction-oriented services. UDP is basically an interface between IP and upper-layer processes. UDP protocol ports distinguish multiple applications running on a single device from one another.

Unlike the TCP, UDP adds no reliability, flow-control, or error-recovery functions to IP. Because of UDP's simplicity, UDP headers contain fewer bytes and consume less network overhead than TCP.

UDP is useful in situations where the reliability mechanisms of TCP are not necessary, such as in cases where a higher-layer protocol might provide error and flow control, or real time data transportation is required.

UDP is the transport protocol for several well-known application-layer protocols, including Network File System (NFS) [30], Simple Network Management Protocol (SNMP) [31], Domain Name System (DNS) [32], and Trivial File Transfer Protocol (TFTP) [33].

4.4 Real Time Transport Protocol (RTP)

In this section we will present the real-time transport protocol (RTP) [34], which provides end-to-end delivery services for data with real-time characteristics, such as interactive audio and video. Those services include payload type identification, sequence numbering, timestamping and delivery monitoring. Applications typically run RTP on top of UDP to make use of its multiplexing and checksum services; both protocols contribute parts of the transport protocol functionality.

Note that RTP itself does not provide any mechanism to ensure timely delivery or provide other quality-of-service guarantees, but relies on lower-layer services to do so. It does not guarantee delivery or prevent out-of-order delivery, nor does it assume that the underlying network is reliable and delivers packets in sequence. The sequence numbers included in RTP allow the receiver to reconstruct the sender's packet sequence, but sequence numbers might also be used to determine the proper location of a packet, for example in video decoding, without necessarily decoding packets in sequence.

RTP, consists of two closely-linked parts:

- The real-time transport protocol (RTP), to carry data that has real-time properties.
- The RTP control protocol (RTCP) [34], to monitor the quality of service and to convey information about the participants in an on-going session.

4.5 HyperText Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) [36] is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990.

HTTP allows an open-ended set of methods and headers that indicate the purpose of a request [38]. It builds on the discipline of reference provided by the Uniform Resource Identifier (URI) [39], as a location (URL) [40] or name (URN) [41], for indicating the resource to which method is to be applied. Messages are passed in a format similar to that used by Internet mail [42] as defined by the Multipurpose Internet Mail Extensions (MIME) [43].

HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet systems. In this way, HTTP allows basic hypermedia access to resources available from diverse applications.

The HTTP protocol is a request/response protocol and the communication usually takes place over TCP/IP connections. The default port is TCP 80 [49], but other ports can be used. This does not preclude HTTP from being implemented on top of any other

protocol on the Internet, or on other networks. HTTP only presumes a reliable transport; any protocol that provides such guarantees can be used.

4.6 Session Description Protocol (SDP)

4.6.1 Overview

Session directories assist the advertisement of conference sessions and communicate the relevant conference setup information to prospective participants. SDP is designed to convey such information to recipients. SDP is purely a format for session description – it does not incorporate a transport protocol, and is intended to use different transport protocols as appropriate including the Session Announcement Protocol [51], Session Initiation Protocol [53], Real-Time Streaming Protocol [52], electronic mail using the MIME extensions, and the Hypertext Transport Protocol.

SDP is intended to be general purpose so that it can be used for a wider range of network environments and applications than just multicast session directories. However, it is not intended to support negotiation of session content or media encoding - this is viewed as outside the scope of session description.

4.6.2 SDP Functionality

The purpose of SDP is to convey information about media streams in multimedia sessions to allow the recipients of a session description to participate in the session. SDP is primarily intended for use in an internetwork, although it is sufficiently general that it can describe conferences in other network environments.

Thus far, multicast based sessions on the Internet have differed from many other forms of conferencing in that anyone receiving the traffic can join the session (unless the session traffic is encrypted). In such an environment, SDP serves two primary purposes. It is a means to communicate the existence of a session, and is a means to convey sufficient information to enable joining and participating in the session. In a unicast environment, only the latter purpose is likely to be relevant.

4.6.3 SDP Messages

SDP session descriptions are entirely textual using the ISO 10646 character set in UTF-8 encoding. SDP field names and attributes names use only the US-ASCII subset of UTF-8, but textual fields and attribute values may use the full ISO 10646 character set. SDP is rather a data format than a protocol with an example shown in the table below:

```
v=0
o=nikos 28908044538 289080890 IN IP4 193.175.132.118
s=SIP Tutorial
e=nikos@speech.tuc.gr
c=IN IP4 126.16.69.4
t=28908044900 28908045000
m=audio 49170 RTP/AVP 0 98
a=rtpmap:98 L16/11025/2
```

Table 4-1: SDP sample message

4.7 Session Initiation Protocol (SIP)

4.7.1 Introduction

There are many applications of the Internet that require the creation and management of a session, where a session is considered an exchange of data between an association of participants. The implementation of these applications is complicated by the practices of participants: users may move between endpoints, they may be addressable by multiple names, and they may communicate in several different media – sometimes simultaneously. Numerous protocols have been authored that carry various forms of real-time multimedia session data such as voice, video, or text messages. The Session Initiation Protocol (SIP) [53] works in concert with these protocols by enabling Internet endpoints (called user agents) to discover one another and to agree on a characterization of a session they would like to share. For locating prospective session participants, and for other functions, SIP enables the creation of an infrastructure of network hosts (called proxy servers) to which user agents can send registrations, invitations to sessions, and other requests. SIP is an agile, general-purpose tool for creating, modifying, and

terminating sessions that works independently of underlying transport protocols and without dependency on the type of session that is being established.

4.7.2 Overview of SIP Functionality

SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls. SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports personal mobility [54] - users can maintain a single externally visible identifier regardless of their network location.

SIP supports five facets of establishing and terminating multimedia communications:

- User location: determination of the end system to be used for communication.
- User availability: determination of the willingness of the called party to engage in communications.
- User capabilities: determination of the media and media parameters to be used;
- Session setup: "ringing", establishment of session parameters at both called and calling party.
- Session management: including transfer and termination of sessions, modifying session parameters, and invoking services.

SIP is not a vertically integrated communications system. SIP is rather a component that can be used with other IETF protocols to build a complete multimedia architecture. Typically, these architectures will include protocols such as the Real-time Transport Protocol (RTP) [34] for transporting real-time data and providing QoS feedback, the Real-Time streaming protocol (RTSP) [20] for controlling delivery of streaming media, the Media Gateway Control Protocol (MEGACO) [55] for controlling gateways to the Public Switched Telephone Network (PSTN), and the Session Description Protocol (SDP) [50] for describing multimedia sessions. Therefore, SIP should be used in conjunction with other protocols in order to provide complete services to the users. However, the basic functionality and operation of SIP does not depend on any of these protocols.

SIP does not provide services. Rather, SIP provides primitives that can be used to implement different services. For example, SIP can locate a user and deliver an opaque object to his current location. If this primitive is used to deliver a session description written in SDP, for instance, the endpoints can agree on the parameters of a session. If the same primitive is used to deliver a photo of the caller as well as the session description, a "caller ID" service can be easily implemented.

SIP does not offer conference control services such as floor control or voting and does not prescribe how a conference is to be managed. SIP can be used to initiate a session that uses some other conference control protocol. Since SIP messages and the sessions they establish can pass through entirely different networks, SIP cannot, and does not, provide any kind of network resource reservation capabilities.

The nature of the services provided make security particularly important. To that end, SIP provides a suite of security services, which include denial-of-service prevention, authentication (both user to user and proxy to user), integrity protection, and encryption and privacy services. Finally, SIP works with both IPv4 and IPv6.

4.7.3 SIP Architecture

SIP's basic architecture is client / server in nature. The main entities in SIP are the User Agent, the SIP Proxy Server, the SIP Redirect Server and the Registrar. The User Agents, or SIP endpoints, function as clients (UAC's) when initiating requests and as servers (UAS's) when responding to requests. User Agents communicate with other User Agents directly or via an intermediate server. The User Agent also stores and manages call states.

SIP intermediate servers have the capability to behave as proxy or redirect servers. SIP Proxy Servers forward requests from the User Agent to the next SIP server, User Agent within the network and also retain information for billing/accounting purposes. SIP Redirect Servers respond to client requests and inform them of the requested server's address. Numerous hops can take place until reaching the final destination. SIP's tremendous flexibility allows the servers to contact external location servers in order to determine user or routing policies and therefore, does not bind the user into only one scheme to locate users. In addition, in order to maintain scalability, the SIP servers can either maintain state information or forward requests in a stateless fashion.

4.7.4 SIP Messages

SIP is a text-based protocol and uses the UTF-8 charset (RFC 2279 [56]). A SIP message is either a request from a client to a server, or a response from a server to a client.
Both Request and Response messages use the basic format of RFC 2822 [57], even though the syntax differs in character set and syntax specifics. Both types of messages consist of a start-line, one or more header fields, an empty line indicating the end of the header fields, and an optional message-body. In the table below a sample SIP message (request) is demonstrated:

```
INVITE sip:ntsourak@speech.tuc.gr SIP/2.0
Via: SIP/2.0/UDP host.speech.tuc.gr:5060
From: Dimitris Pratsolis <sip:pratsdim@speech.tuc.gr>
To: Nikos Tsourakis <sip:ntsourak@speech.tuc.gr>
Call-ID: 314159@host.speech.tuc.gr
CSeq: 1 INVITE
```

Table 4-2: SIP sample message

4.8 Real Time Streaming Protocol (RTSP)

4.8.1 Overview

The Real-Time Streaming Protocol (RTSP) [52] establishes and controls either a single or several time-synchronized streams of continuous media such as audio and video. It does not typically deliver the continuous streams itself, although interleaving of the continuous media stream with the control stream is possible. In other words, RTSP acts as a "network remote control" for multimedia servers.
There is no notion of an RTSP connection; instead, a server maintains a session labelled by an identifier. An RTSP session is in no way tied to a transport-level connection such as a TCP connection. During an RTSP session, an RTSP client may open and close many reliable transport connections to the server to issue RTSP requests. Alternatively, it may use a connectionless transport protocol such as UDP.

The streams controlled by RTSP may use RTP [34], but the operation of RTSP does not depend on the transport mechanism used to carry continuous media. The protocol is intentionally similar in syntax and operation to HTTP/1.1 [36] so that extension mechanisms to HTTP can in most cases also be added to RTSP. However, RTSP differs in a number of important aspects from HTTP. The protocol supports the following operations:

- Retrieval of media from media server.
- Invitation of a media server to a conference.
- Addition of media to an existing presentation.

4.8.2 Relationship with Other Protocols

RTSP has some overlap in functionality with HTTP. It also may interact with HTTP in that the initial contact with streaming content is often to be made through a web page. The current protocol specification aims to allow different hand-off points between a web server and the media server implementing RTSP. For example, the presentation description can be retrieved using HTTP or RTSP, which reduces roundtrips in web-browser-based scenarios, yet also allows for standalone RTSP servers and clients, which do not rely on HTTP at all.

However, RTSP differs fundamentally from HTTP in that data delivery takes place out-of-band in a different protocol. HTTP is an asymmetric protocol where the client issues requests and the server responds. In RTSP, both the media client and media server can issue requests. RTSP requests are also not stateless; they may set parameters and continue to control a media stream long after the request has been acknowledged.

Re-using HTTP functionality has advantages in at least two areas, namely security and proxies. The requirements are very similar, so having the ability to adopt HTTP work on caches, proxies and authentication is valuable.

While most real-time media will use RTP as a transport protocol, RTSP is not tied to RTP. RTSP assumes the existence of a presentation description format that can express both static and temporal properties of a presentation containing several media streams.

4.8.3 RTSP Message

RTSP is a text-based protocol and uses the ISO 10646 character set in UTF-8 encoding (RFC 2279 [56]). Lines are terminated by CRLF, but receivers should be prepared to also interpret CR and LF by themselves as line terminators.

Text-based protocols make it easier to add optional parameters in a self-describing manner. Since the number of parameters and the frequency of commands are low, processing efficiency is not a concern. Text-based protocols, if done carefully, also allow easy implementation of research prototypes in scripting languages such as Tcl, Visual Basic and Perl.

RTSP messages can be carried over any lower-layer transport protocol that is 8-bit clean. Requests contain methods, the object the method is operating upon and parameters to further describe the method. Methods are idempotent, unless otherwise noted. Methods are also designed to require little or no state maintenance at the media server.

In the table below, the client requests the description of a presentation or media object identified by the request URL from a server.

```
DESCRIBE rtsp://server.example.com/fizzle/foo RTSP/1.0
CSeq: 312
Accept: application/sdp, application/rtsl, application/mheg
```

Table 4-3: RTSP sample message

4.9 Media Resource Control Protocol (MRCP)

4.9.1 Overview

The Media Resource Control Protocol (MRCP) [58] is designed to provide a mechanism for a client device requiring audio/video stream processing to control processing resources on the network. These media processing resources may be speech recognizers, speech synthesizers, fax, signal detectors, etc. MRCP allows implementation of distributed Interactive Voice Response platforms, for example VoiceXML [23] interpreters.

The MRCP protocol defines the requests, responses and events needed to control the media processing resources. The MRCP protocol defines the state machine for each resource and the required state transitions for each request and server-generated event.
The MRCP protocol does not address how the control session is established with the server and relies on the Real Time Streaming Protocol (RTSP) [20] to establish and maintain the session. The session control protocol is also responsible for establishing the media connection from the client to the network server. The MRCP protocol and its messaging is designed to be carried over RTSP or another protocol as a MIME-type similar to the Session Description Protocol (SDP).

4.9.2 Architecture

The system consists of a client that requires media streams generated or needs media streams processed and a server that has the resources or devices to process or generate the streams. The client establishes a control session with the server for media processing using a protocol such as RTSP. This will also set up and establish the RTP stream between the client and the server or another RTP endpoint. Each resource needed in processing or generating the stream is addressed or referred to by a URL. The client can now use MRCP messages to control the media resources and affect how they process or generate the media stream.

4.9.3 Establishing Control Session and Media Streams

The control session between the client and the server is established using a protocol like RTSP. This protocol will also set up the appropriate RTP streams between the server and the client, allocating ports and setting up transport parameters as needed. Each control session is identified by a unique session-id. The format, usage and life cycle of the session-id is in accordance with the RTSP protocol. The resources within the session are addressed by the individual resource URLs.
The MRCP protocol is designed to work with and tunnel through another protocol like RTSP, and augment its capabilities. MRCP relies on RTSP headers for sequencing, reliability and addressing to make sure that messages get delivered reliably and in the correct order and to the right resource. The MRCP messages are carried in the RTSP message body. The media server delivers the MRCP message to the appropriate

resource or device by looking at the session level message headers and URL information. Another protocol, such as SIP, could be used for tunneling MRCP messages [57].

4.9.4 MRCP over RTSP

RTSP supports both TCP and UDP mechanisms for the client to talk to the server and is differentiated by the RTSP URL. All MRCP based media servers must support TCP for transport and may support UDP. Currently all RTSP messages are request/responses and there is no support for asynchronous events in RTSP. This is because RTSP was designed to work over TCP or UDP and hence could not assume reliability in the underlying protocol.

An example of a MRCP message, which is a response generated from the server and contains the interpretation of the recognition, is shown in the table below:

```
RECOGNITION-COMPLETE 543257 COMPLETE MRCP/1.0
Completion-Cause: 000 success
Waveform-URL: http://web.media.com/session123/audio.wav
Content-Type: application/x-nlsml
Content-Length: 276

<?xml version="1.0"?>
<result x-model="http://IdentityModel"
        xmlns:xf="http://www.w3.org/2000/xforms"
        grammar="session:request1@form-level.store>
        <interpretation>
              <xf:instance name="Person">
                   <Person>
                      <Name> Andre Roy </Name>
                   </Person>
             </xf:instance>
             <input> may I speak to Andre Roy </input>
        </interpretation>
</result>
```

Table 4-4: MRCP sample message

Chapter 5

SYSTEM ARCHITECTURE

5.1 Overview

In the fifth chapter of the current document we will introduce the architectures associated with the proposed system. For each architecture we will present the fundamental components of the system, with emphasis on the supported functionalities and we will describe how the interconnection among them is accomplished.
Moreover the implementation of one of the proposed architectures worked as a proof o concept and offered a generic framework for creating multimodal applications. This specific framework was used in order to create a sample application, as we will see in the next chapter.

5.2 First Architecture

We start with the presentation of the first proposed architecture. As we can see in Figure 5-1, the specific architecture is based on four parts/nodes, namely:

1. The Mobile Application
2. The Automatic Speech Recognition System (ASR)

3. The SIP Servers (Proxy, Redirect, Registrar)
4. The Packet Switched Network

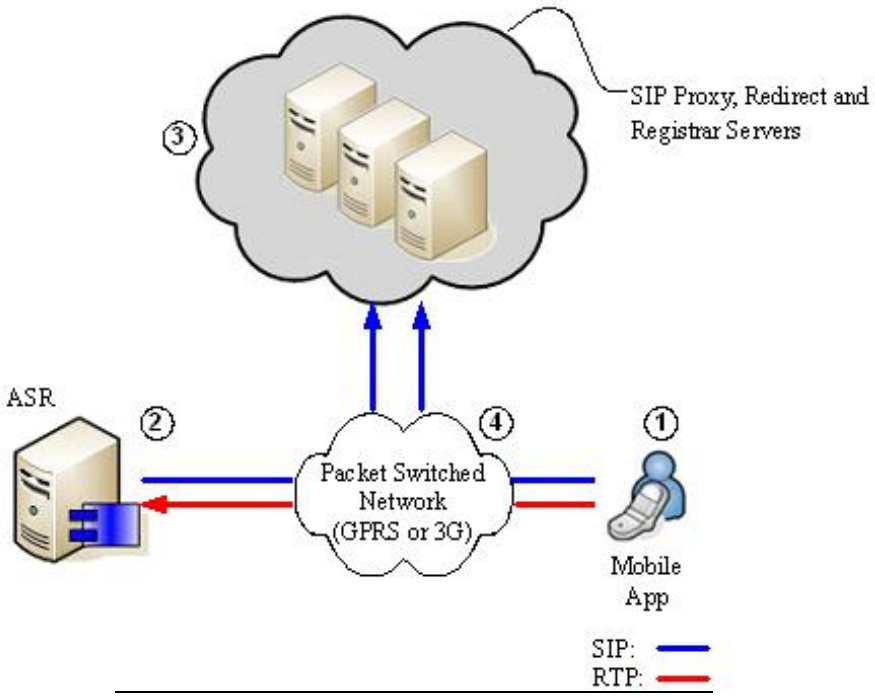

Figure 5-1: First system architecture

The mobile application, the ASR and the SIP related servers comprise autonomous peers using the same data network, either GPRS or 3G. In order to accomplish the interconnection among them, two protocols are employed by the peers, which are the Session Initiation Protocol (SIP) and the Real Time Transport Protocol (RTP).
Specifically, the SIP protocol is used for:

1. Establishing presence, either by the mobile application or the ASR, to the Registrar server.
2. User location, so that the mobile application can locate the address of the ASR, which can change in a period of time.
3. Session setup, in order to establish the connection (session) between the ASR and the mobile application.

The user has a custom application installed on his mobile device, which implements a multimodal information retrieval service. After the session is established, the user can send audio data to the ASR for recognition through the RTP protocol. This session can either expire after a certain time interval that is defined during the initial SIP messages exchange between the application and the ASR or shutdown on demand using a suitable SIP message. On the other hand, the RTP link between the application and the ASR engine can be established on demand, when audio data need to be transmitted for recognition over the network.

After the speech recognition is performed, the ASR can inform the multimodal application running on the mobile device about the corresponding result. This was the problem that we encountered during the implementation of the specific architecture. Although all communication links were successfully established and the recognition took place, due to the limitations of the ASR we couldn't come up with an elegant solution of informing the application with the recognition result. Specifically, the ASR does not offer an explicit or implicit mechanism for accessing SIP methods and subsequently use them in order to send the recognition result. The SIP stack in encapsulated within a module called the "Audio Provider", which transparently handles the signalling and the audio data and must be used as offered. Therefore, only pre-recorded prompts can be rendered by the ASR and announced by the mobile application. Ideally, with the use of a SIP message the ASR should perform the specific information task.

An advantage of the specific architecture is that besides the logic of the application running on the mobile device, there is a server side logic of the application running on the ASR. The latter can perform a number of tasks in order to lighten the load on the mobile application and the network traffic. Such tasks are the data retrieval, the processing of the recognition results or the execution of complicated calculations.

We should note that, in the specific implementation, the ASR and the SIP related Servers can be hosted on the same server, but in a more generalized version they can reside on different servers sharing the same Ethernet network.

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

5.3 Second Architecture

In the presentation of the second proposed architecture we will focus on the use of the Media Resource Control Protocol (MRCP), a state-of-the-art protocol adopted by major speech vendors. MRCP represents a paradigm shift for telephony application developers using speech technologies. Instead of using vendor-specific commands to generate text-to-speech or perform automatic speech recognition, MRCP commands are sent from the client to the server application. Additionally, instead of loading pre-recorded audio files locally, commands are sent to an MRCP server containing a URL of the file to be played to the caller. These commands generate an audio stream sent to the client application, which should then direct it to the caller.

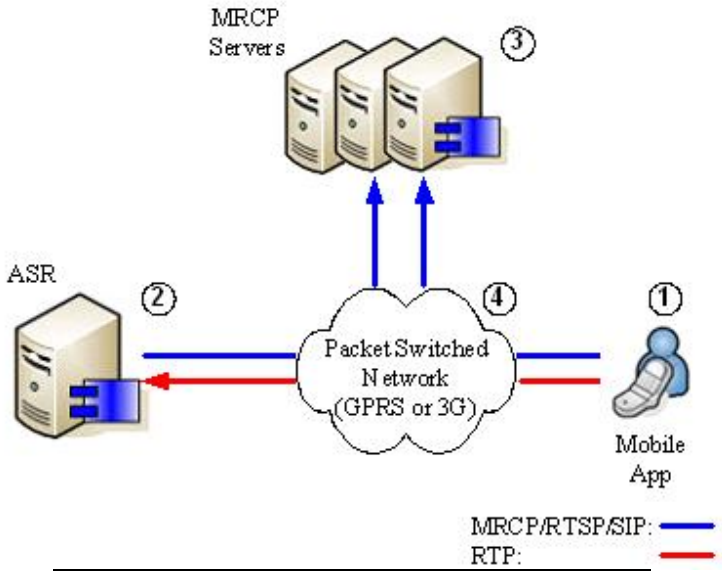

Figure 5-2: Second system architecture

As we can observe in Figure 5-2 the corresponding components of the architecture are:

1. The Mobile Application
2. The Automatic Speech Recognition System (ASR)
3. The MRCP Server (or servers)
4. The Packet Switched Network

Version 1.0 of MRCP uses the Real Time Streaming Protocol (RTSP) to establish connections from an MRCP client application to an MRCP server. All MRCP commands are then tunnelled via RTSP Announce messages between the MRCP client and server. Version 2.0 of MRCP replaces the use of RTSP for command and control with the session initiation protocol (SIP). MRCP commands between the client and the server are used to establish (bi-directional) audio paths between the client and server using the Real time Transport Protocol (RTP). RTP packets sent from the client application to the MRCP server contain the caller's speech, commonly referred to as utterances, sent for recognition. Conversely, RTP packets sent from the server to the client contain either TTS (speech synthesis) or streamed audio retrieved from a web server. A majority of the speech vendors require this RTP traffic to be encoded using the G.711 codec to ensure quality audio for recognition and delivery to the caller.

During the implementation of a multimodal service, besides the audio that may be delivered to the caller, we may want a textual result, which can be rendered on the screen. The specific result can be enclosed in a MRCP message dispatched by the server and delivered on the mobile application.

It is possible to use one MRCP server for recognition and a separate server for TTS or streaming file play. Implementers may choose MRCP servers from multiple vendors based on a vendor's strength or ability to support a particular language.

The specific architecture was not implemented because in the beginning of our work the MRCP was not yet adopted by any ASR system. However it worked as an excellent paradigm in order to introduce and implement the third architecture, which relies on the distributed nature of the MRCP, by offering a transparent and reliable integration with different types of ASR systems.

5.4 Third Architecture

Initially, we will provide a high level overview of the specific system architecture, which was actually implemented and will be presented in more detail.

As we can see in Figure 5-3, five nodes constitute the proposed architecture, namely:

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

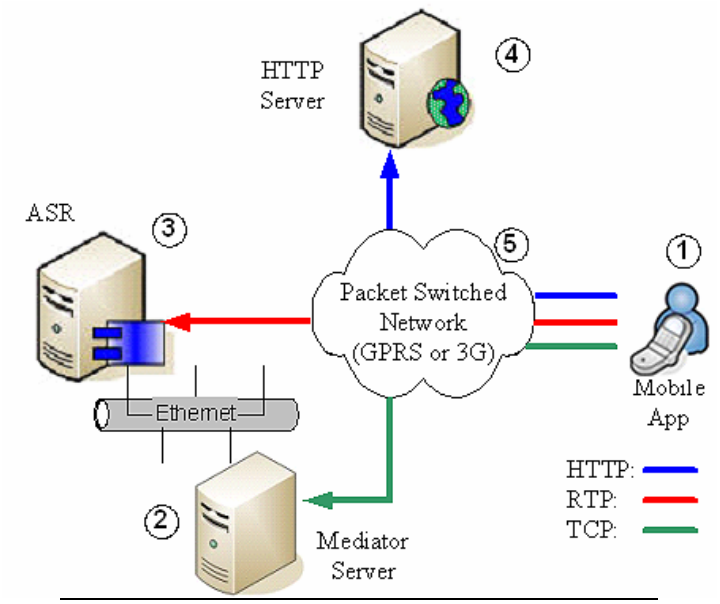

Figure 5-3: Third system architecture

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.

5.4.1 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.

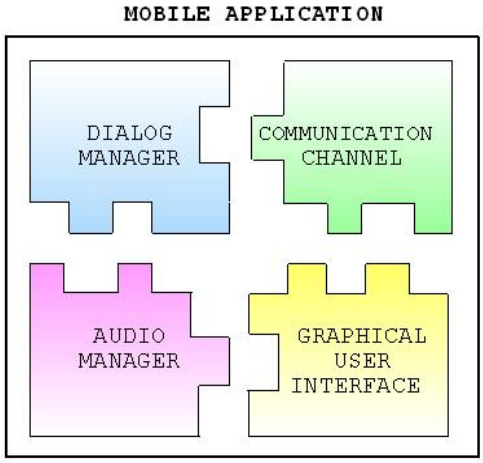

Figure 5-4: Mobile application components

As we can see in Figure 5-4, 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.

5.4.1.1 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.

5.4.1.2 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 modelled 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.

5.4.1.3 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.

5.4.1.4 Graphical User Interface

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. Finally, it offers the ability to insert data using the keypad.
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.

5.4.1.5 UML Diagram

Three of the mentioned components were implemented by three c++ objects with the corresponding UML diagram depicted in Figure 5-5. The class associated with the Communication Channel component is the CCommunicationChannel class, with the Dialog Manager component the CDialogManager class and with the Graphical User Interface component the CSpeechAppUI class.
During the implementation we used extensively the "Observer Design Pattern" [5], a standard way of allowing objects to communicate without tight coupling. Specifically, each interface in the UML diagram is implemented by the corresponding object, e.g.

MDialogManagerObserver interface is implemented by CSpeechAppUI object and used by CDialogManager class. The latter includes a pointer to the specific interface and in this way can use the methods that the CSpeechAppUI class wishes to expose.

Beyond the specific classes, our implementation included several other classes, which cannot be presented in the context of the current document.

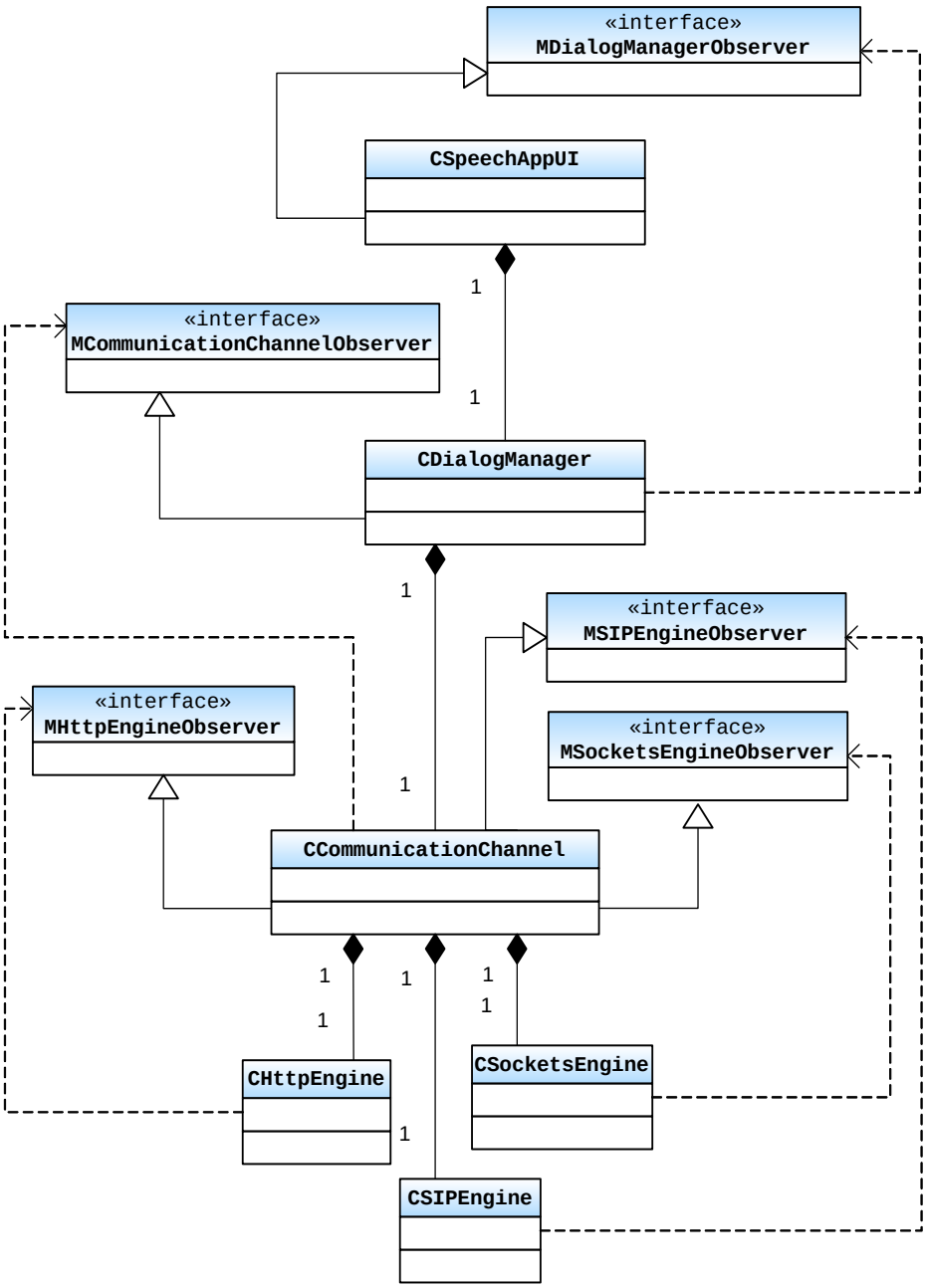

Figure 5-5: Mobile application UML diagram

5.4.2 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 5-1.

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.
INTERPRET	Natural language processing of the input utterance.

Table 5-1: Mediator supported actions

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.

Figure 5-6 depicts the messages that are exchanged between the mobile application and the Mediator Server for the REGOGNIZE action:

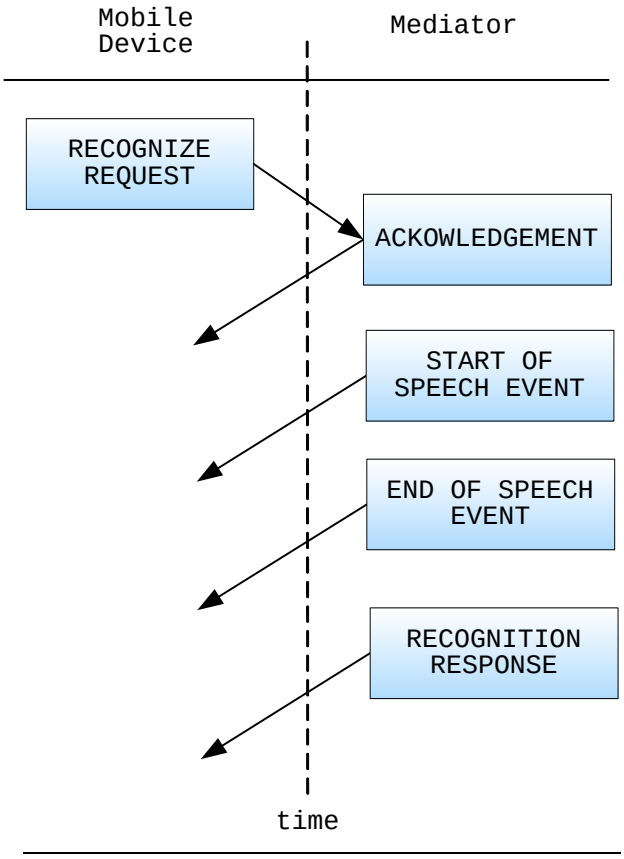

Figure 5-6: The RECOGNIZE action

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.

The complete list and the structure of the available messages, exchanged between the mobile application and the Mediator Server, are presented in the appendix of the current document.

5.4.3 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. In our case we implemented the one that corresponds to the SRI Decipher Speech Recognition System [74][75], which is a large-vocabulary, continuous-speech, speaker-independent system.

5.4.4 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.

Chapter 6

GRAPHICAL USER INTERFACE (GUI)

6.1 Overview

In this section, we present a sample stock-information application that utilizes 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 [5].
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 (CStockAppDM) derived from the Dialog Manager.
2. The StockApp GUI (CStockAppUI) 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.

In Figure 6-1 we demonstrate the UML diagram of the stock application. The derivation of each component (class) is depicted as the interconnection between them. The specific interconnection is again performed with the use of the "Observer Design Pattern" [5].

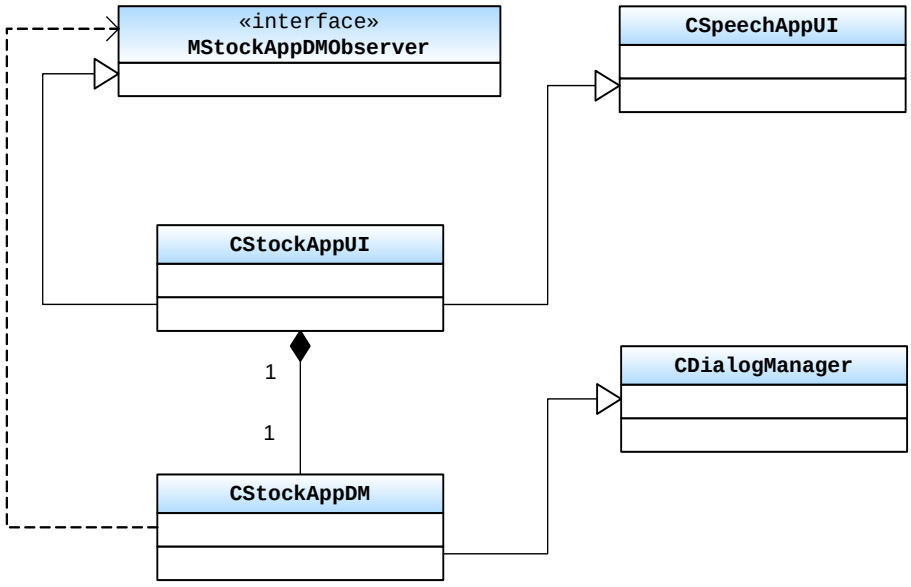

Figure 6-1: Stock application UML diagram

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

6.2 Series 60 Developer Platform

Our implementation was based on the Series 60 Developer Platform provided by Nokia. Series 60 devices are based on Symbian OS, an open, robust, 32-bit multitasking operating system designed for data-enabled mobile phones. Symbian OS is the global industry standard operating system for smartphones, and is licensed to the

world's leading handset manufacturers, who account for over 85 per cent of annual worldwide mobile phone sales.

The Series 60 Developer Platform is a complete smartphone software package that provides a mandatory base of technology implementations. The Series 60 Developer Platform enables application and service developers to produce smartphone products for the Series 60 Platform. While the term "Series 60 Platform" encompasses the technologies of the relevant version of the Series 60 Developer Platform on which it is based, it also provides an optional range of lead software that licensees may wish to support. Devices based on the Platform therefore provide the developer with a guaranteed set of technologies, while allowing the manufacturer to differentiate between devices by choosing the lead software options best tailored to the target consumer segment.

6.3 Stock Application Simulation

Initially, we will provide a brief demonstration of our implementation, executing a simulation on the emulator. As we can see in Figure 6-2, our deployed stock application appears along with the other available applications, which are installed on the emulator.

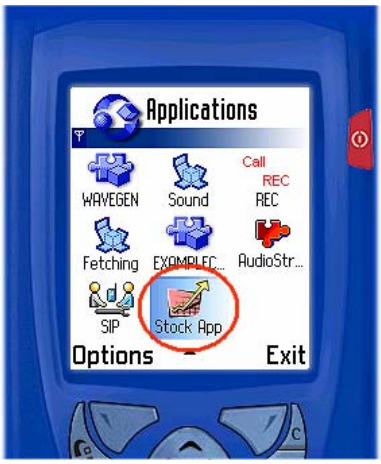

Figure 6-2: Stock application

When it is executed, the screen shot shown in Figure 6-3 appears and the user can choose between the Options menu (1), or exiting (2).

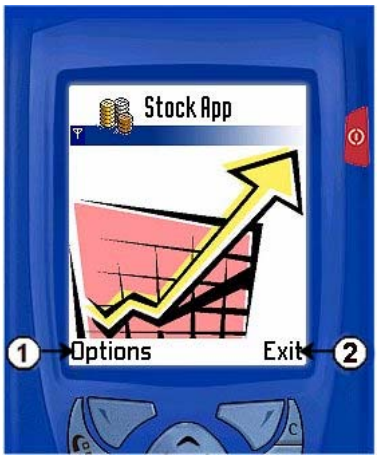

Figure 6-3*:* Initial screen

The main menu offers seven choices. The seventh menu item is used for exiting from the application and is not shown in Figure 6-3.

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 shown in Figure 6-4, 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.

Figure 6-4: Application menu

The second menu item initiates the recording of input speech. As we can see in Figure 6-5, 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 6-5). Recording can be aborted by pressing the CANCEL button and consequently the "ABORT RECOGNITION" action is executed.

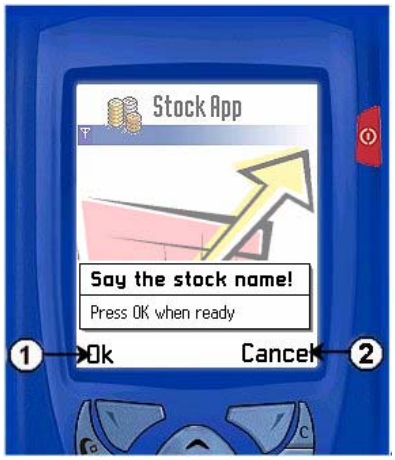

Figure 6-5:User input

After uttering the name of a stock, the user can chose between playing back the previously recorded utterance (menu item 3 – Figure 6-4) and retransmitting it to the ASR engine (menu item 5 – Figure 6-4).

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 6-6.

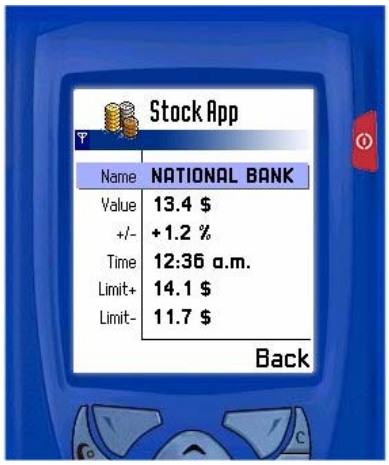

Figure 6-6: Application output

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 6-7: Key input

As we have mentioned earlier, the application offers an alternative mode of input that is the key input (Figure 6-7). 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 6-6.

Chapter 7

EXPERIMENTAL EVALUATION

7.1 Overview

Techniques for modeling and simulating channel conditions play an essential role in understanding network protocol and application behavior. As networks evolve, the design of communication protocols increases in complexity. Evaluating the performance of existing networks provides insights into techniques for optimizing future protocols. The most common techniques include simulation, analysis of empirical data, and analytical models (e.g., channel models).

For example, a detailed understanding of the packet-loss process and burstiness of errors is necessary for the proper design and parameter tuning of error-control protocols.

Streaming audio and video multimedia applications can also benefit from a better understanding of the underlying network behavior. For example, video and audio codecs can perform real-time predictive rate control by using a model of network traffic characteristics to estimate traffic conditions in real-time.

Much of the early work on loss or error modeling occurred in the 1960's in relation to the distribution of bit errors on telephone channels. One approach used was a Markov or multi-state model. Gilbert [7] appears to be the first to describe a burst error model of

this type, later extended by Elliott [8] and Cain and Simpson [9]. Blank and Trafton [10] produced higher state Markov models to represent error distributions. Another approach was to identify the statistical distribution of gaps. Mertz [11] used hyperbolic distributions and Berger and Mandelbrot [12] used Pareto distributions to model inter-error gaps. Lewis and Cox [13] found that in measured error distributions there was strong positive correlation between adjacent gaps. Packet loss modeling in IP networks seems to have followed a similar course, although the root cause of loss (typically congestion) may be different to that of bit errors (typically circuit noise or jitter).

7.2 Gilbert-Elliot Model

The traditional network modeling approach to error modeling is to create a Gilbert model based upon collected network traffic traces. Using this model, one can then dynamically generate artificial network traces for the network under study and use the traces to simulate, and thus, better understand the performance of existing and new network protocols and applications. These traces provide network protocol and application developers with ease of use and repeatability, two critical characteristics for developing and improving network and application performance. More importantly, for new networks under development (or for which there are only limited prototypes), it is often difficult to collect a reasonable amount of traces or to run experiments. By generating synthetic traces that simulate the network being tested, multiple users can simultaneously gain network access and perform experiments.

Errors that occur in wireless channels are a function of specific propagation artifacts such as multi-path and fading. For mobile radio channels, especially those used for typical transmissions from a base station to mobile users in an urban environment, Rayleigh fading is the widely accepted model. The Gilbert-Elliot packet erasure channel model has been proved to model this kind of fading channel with sufficient accuracy [17] [18].

In our work, we also use the Gilbert-Elliot model to simulate the packet error behavior of the wireless channels.

The Gilbert-Elliot (GE) channel model is a two-state Markov model [7] [8], which is widely used to simulate the bursty packet loss behavior. This channel model has been

shown to be able to effectively capture the bursty packet loss behavior of the Internet and wireless channels. Furthermore, the special structure of the Markov model makes it analytically tractable.

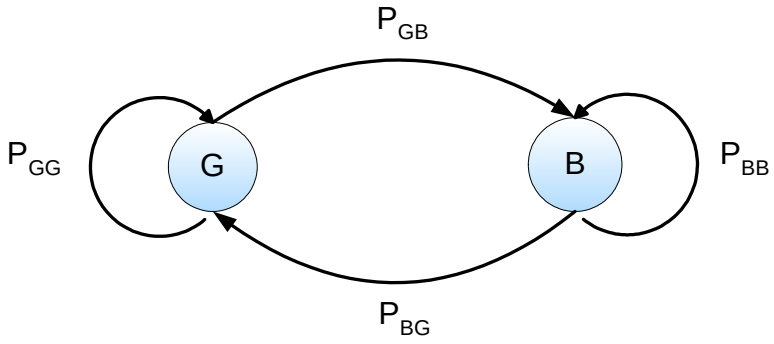

Figure 7-1: Gilbert-Elliot model state transition diagram

The two states of the GE model are denoted as G (good) and B (bad), as illustrated in Figure 7-1. In state G, packets are assumed to be received correctly and timely, whereas in state B, packets are assumed to be lost. This model can be described by the transition probabilities P_{GB} from state G to B and P_{BG} from state B to G. Let P denote the transition probability matrix for the packet error process:

$$P = \begin{pmatrix} P_{GG} & P_{GB} \\ P_{BG} & P_{BB} \end{pmatrix} = \begin{pmatrix} 1-P_{GB} & P_{GB} \\ P_{BG} & 1-P_{BG} \end{pmatrix}$$

The steady-state probability of being in the good state and bad state, respectively, is given by:

$$P_G = \frac{P_{BG}}{P_{GB} + P_{BG}}, \quad P_B = \frac{P_{GB}}{P_{GB} + P_{BG}}$$

Typically, the average packet loss probability (P_e) and the average burst length (L_B) are defined to describe the bursty packet loss behavior of the GE channel model, which are given by:

$$P_e = P_B = \frac{P_{GB}}{P_{GB} + P_{BG}}, \quad L_B = \frac{1}{P_{BG}}$$

7.3 Three-State Markov Chain Model

A Markov model is a general multi-state model in which a system switches between states i and j with some transition probability p(i, j). A 2-state Markov model has some merit in that it is able to capture very short-term dependencies between lost packets, i.e. consecutive losses. These are generally very short duration events (say 1-3 packets in length) but occasional link failures can result in very long loss sequences extending to tens of seconds [14]. By engaging higher order Markov models, it is possible to capture both very short duration consecutive loss events and longer lower density events.

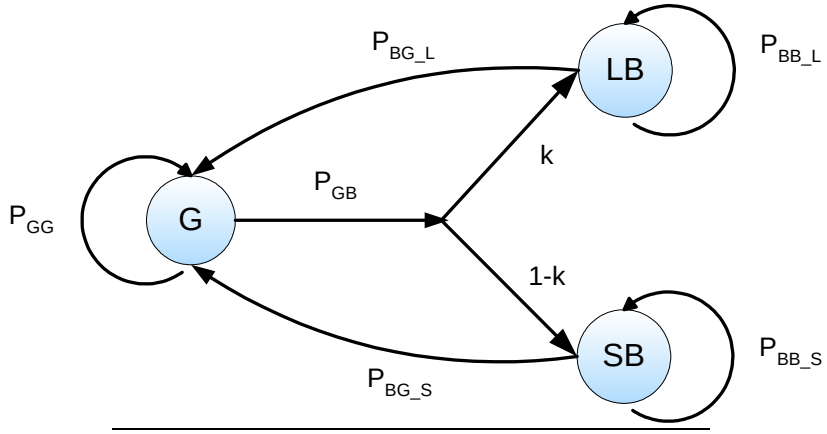

Figure 7-2: Three-state Markov model transition diagram

A corresponding error model is represented by a three-state discrete-time Markov chain [15], as the one shown in Figure 7-2. Errors over the channel occur in the states long bad (LB) and short bad (SB), while the good (G) state is error-free. The difference between the long bad LB and short bad SB states is the time correlation of errors: LB corresponds to long bursts of errors and SB to short ones.

The parameter k is the probability that the Markov chain moves to state LB given that it leaves state G; k also represents the probability that an error burst is long, or, in other terms, the fraction of long bursts over the total number of error bursts.

Let *P* denote again the transition probability matrix for the packet error process, which is given from the equation below:

$$P = \begin{pmatrix} P_{GG} & P_{GB\_S} & P_{GB\_L} \\ P_{BG\_S} & P_{BB\_S} & 0 \\ P_{BG\_L} & 0 & P_{BB\_L} \end{pmatrix} = \begin{pmatrix} 1-P_{GB} & k \cdot P_{GB} & (1-k)P_{GB} \\ P_{BG\_S} & 1-P_{BG\_S} & 0 \\ P_{BG\_L} & 0 & 1-P_{BG\_L} \end{pmatrix}$$

Note that no transitions are allowed between states LB and SB and the model can be fully incorporated when parameters P_{GB}, P_{BG_S}, P_{BG_L} and k are defined.

7.4 Metrics

The word error rate (WER) is a commonly used metric to evaluate speech recognizers. It is a measure of the average number of word errors taking into account three error types: *substitution* (the reference word is replaced by another word), *insertion* (a word is hypothesized that was not in the reference) and *deletion* (a word in the reference transcription is missed). The word error rate is defined as the sum of these errors divided by the number of reference words. Given this definition, the word error can be more than 100%.

$$\%WER = \frac{(insertions + deletions + substitutions)}{word\ count} \cdot 100\%$$

A different error metric and in some cases more appropriate, is the natural language error rate (NLER). Generally, in speech recognition applications, we usually are interested in the interpretation of a spoken utterance rather than its accurate transcription. If for example, the ASR system recognize the utterance "I want *the* chocolate ice cream" as "I want chocolate ice cream" we will come up with a non-zero WER. On the other hand, the specific recognition result will have zero NLER, as we interpreted correctly the user's request that asked for a chocolate ice cream. Therefore, every recognition result is initially associated with one or more unique keys, which are later used during the interpretation process. The NLER is simply defined as the number of NL errors occurred during the examination of the utterances, divided by the number of the reference utterances.

$$\%NLER = \frac{NL\,errors}{utterances} \cdot 100\%$$

We should note that even one error in the aforementioned unique keys within the recognition result, will lead to a NL error for the specific result, in cases that there are more than one interpretation slots per utterance

7.5 Evaluation Process

During the evaluation process we calculated the NL error rate after processing a test set derived from a stock information application, which consisted of 1000 utterances. We applied the error models discussed earlier, using different configuration parameters and simulated different strategies of the ASR system for handling transmission errors. The configuration parameters for each model are the transition probabilities presented earlier. The recognizer can incorporate three different strategies for a missing packet:

1. Replacing the missing packet with silence.
2. Ignoring the missing packet.
3. Replacing the missing packet with the previous one.

Each state in the error models corresponds to a time slot of 20 ms, which is associated with the transmission of a packet of 160 bytes. Thus, a packet transmission is successful only if the error model is in state G for all slots needed for the packet to be transmitted, while it fails otherwise. Each utterance is therefore split into packets of 160 bytes, and the error models determine which one of them will be received by the ASR.

In the Gilbert-Elliot model two parameters must be defined, namely the packet loss probability P_B and the average burst length L_B. For each parameter, the range of possible values is provided and each pair constitutes the current configuration of the model. The parameter values used in our simulations are summarized in the following table:

Gilbert-Elliot Model			
Parameter	Min	Max	Step
$\mathbf{P_B}$	1%	15%	1%

$\mathbf{L_B}$	1	4	1

Table 7-1: Gilbert-Elliot model configuration parameters

The number of different experiments derived from the specific model simulations is:

$$(values\,of\ P_B)\cdot(values\,of\ L_B)\cdot(ASR\,strategies)=15\cdot4\cdot3=180$$

We used a similar process for the Three-State Markov model, where P_{GB}, P_{BG_S}, P_{BG_L} and k are the configuration parameters with a range presented in the table below:

Three-State Markov Model			
Parameter	Min	Max	Step
$\mathbf{P_{GB}}$	1%	10%	1%
$\mathbf{P_{BG\_S}}$	35%	35%	0
$\mathbf{P_{BG\_L}}$	15%	15%	0
k	5%	20%	5%

Table 7-2: Three-State Markov model configuration parameters

Similarly, the number of experiments obtained from the specific simulation is:

$$(values\,of\ P_{GB})\cdot(values\,of\ P_{BG\_S})\cdot(values\,of\ P_{BG\_L})\cdot(values\,of\ k)\cdot(ASR\,strategies)=$$
$$=10\cdot1\cdot1\cdot4\cdot3=120$$

For each experiment, we simulated packet loss for all 1000 sentences of the test set and fed the resulting sentences to the ASR system in order to calculate the NL error rate. We should note that the system could recognize among 500 unique stock names uttered in different contexts.

7.6 Evaluation Results

In this last section we will present the graphs obtained from the simulations with the two models. The specific results are compared with the NL error rate obtained from the original clean test set, which yields a NLER of 4.52%.

7.6.1 Gilbert-Elliot Model Evaluation Results

Initially, four graphs that correspond to the Gilbert-Elliot model will be presented. The NL error rate associated with each one of the three ASR strategies discussed earlier is depicted in Figures 7-3, 7-4 and 7-5. The specific error rate is calculated with respect of the packet loss probability (P_b) in the x-axis and the average burst error length (L_b). Each graph contains four plots that correspond to the different values of L_b.

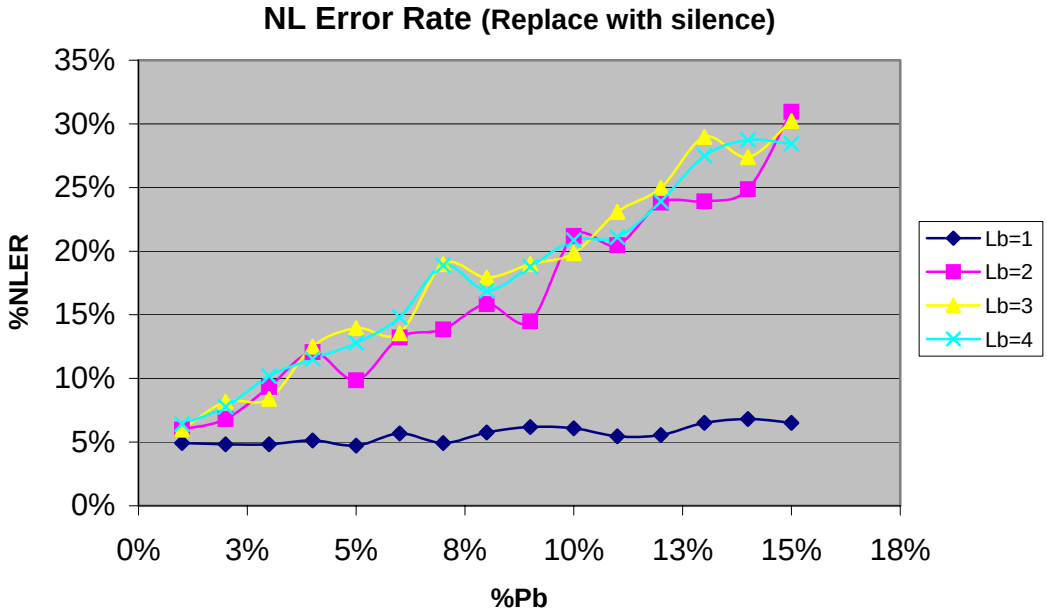

Figure 7-3: NL Error Rate for the first ASR strategy (Gilbert-Elliot model)

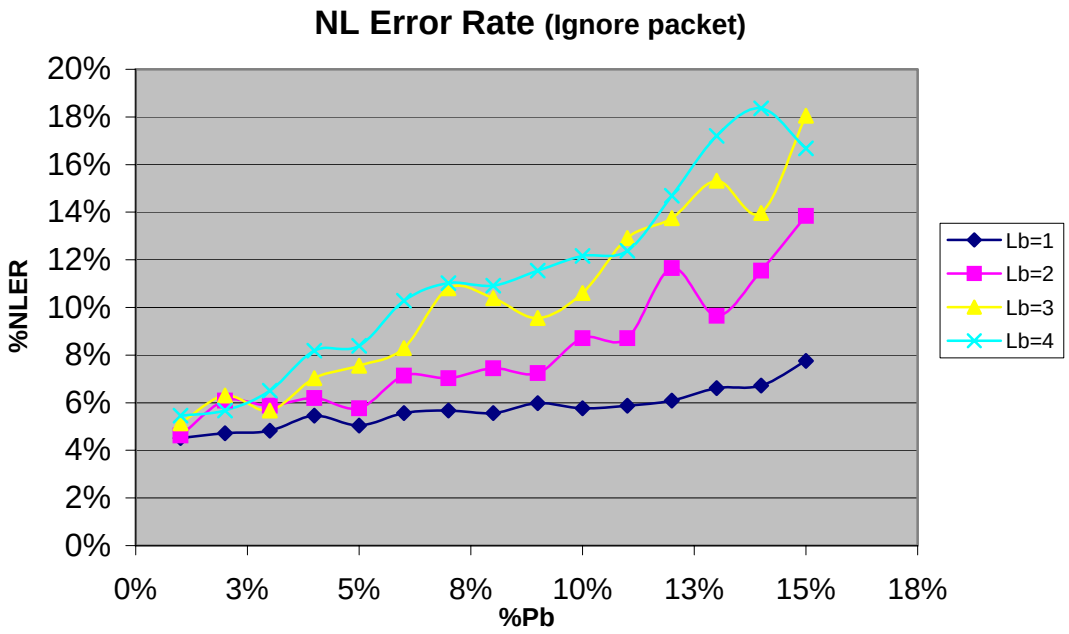

Figure 7-4: NL Error Rate for the second ASR strategy (Gilbert-Elliot model)

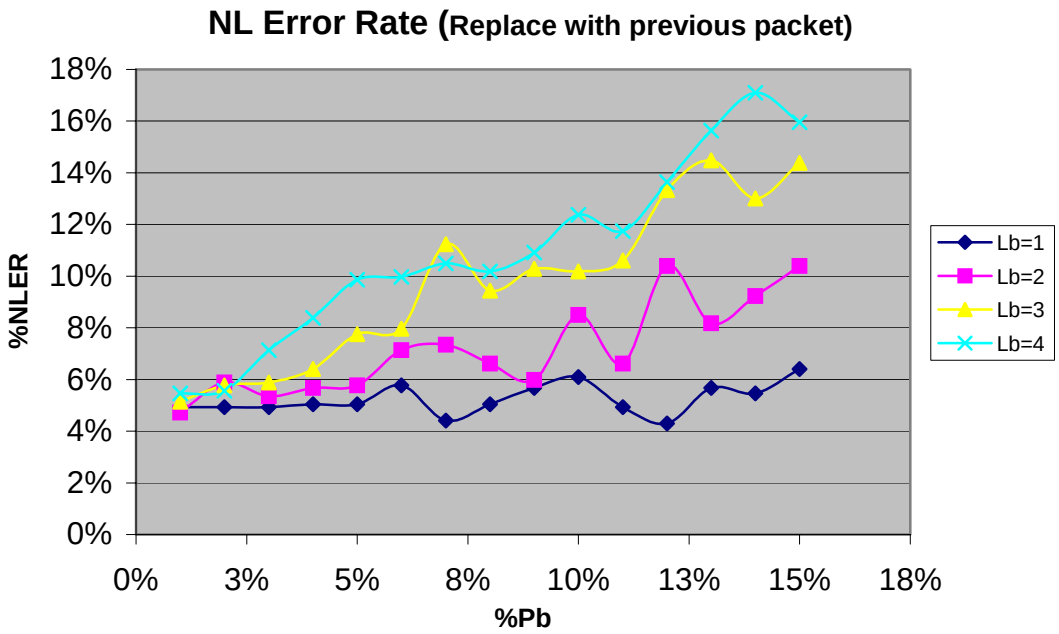

Figure 7-5: NL Error Rate for the third ASR strategy (Gilbert-Elliot model)

We should note that when L_b equals to 1 it is guarantied that no consecutive error packets will be encountered in the packet error burst as $P_{BB}=0$.

As we can observe from the figures above, the strategy of replacing the lost packets with the preceding one yields to the best results. The specific outcome was expected as in most cases there is a strong correlation of adjacent speech samples and consequently of adjacent speech packets. We should also note that the worst results were obtained using the strategy of replacing the missing packets with silence.

Concluding the simulation results for the Gilbert-Elliot model we will provide a series of sample graphs that demonstrate the distribution of the lost packets for a specific configuration. The histograms in Figures 7-7, 7-8 and 7-9 correspond to $P_b=5\%$ and $L_b=2,3,4$ respectively. They present the real effects of applying the model on the waveforms and provide a better understanding on how the lost packets are grouped into error bursts. In the x-axis the number of consecutive lost packets is depicted while the y-axis represents the total number of the corresponding error bursts that occurred after the application of the model.

For example, for an average burst length equal to 2 (Figure 7-7) we encountered 219 lost triplets and 128 lost quadruplets. The total number of packets in the test set is 79190.

We should note that the volume of the different configurations prevented any further demonstrations and this is why we provided only some sample graphs.

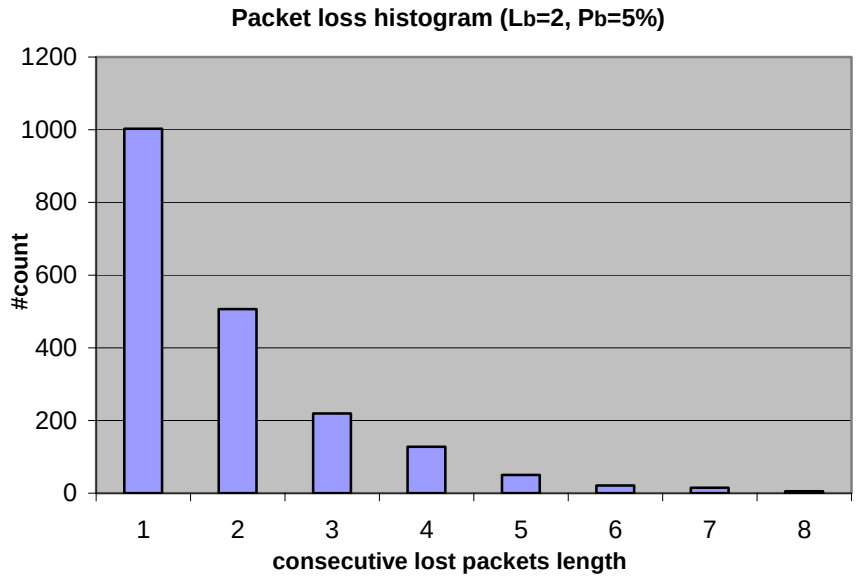

Figure 7-6: Packet loss histogram (L_b=2, P_b=5%)

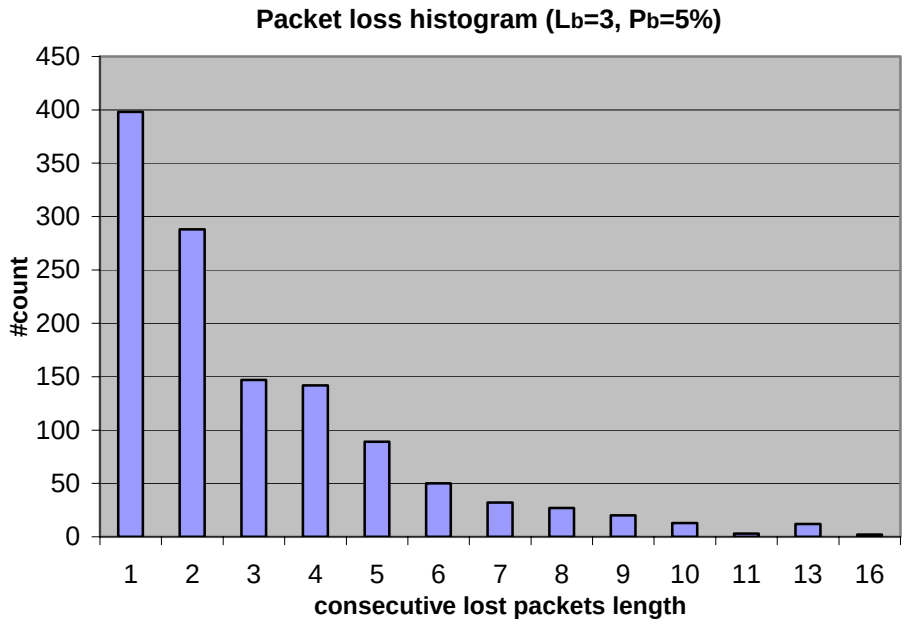

Figure 7-7: Packet loss histogram (L_b=3, P_b=5%)

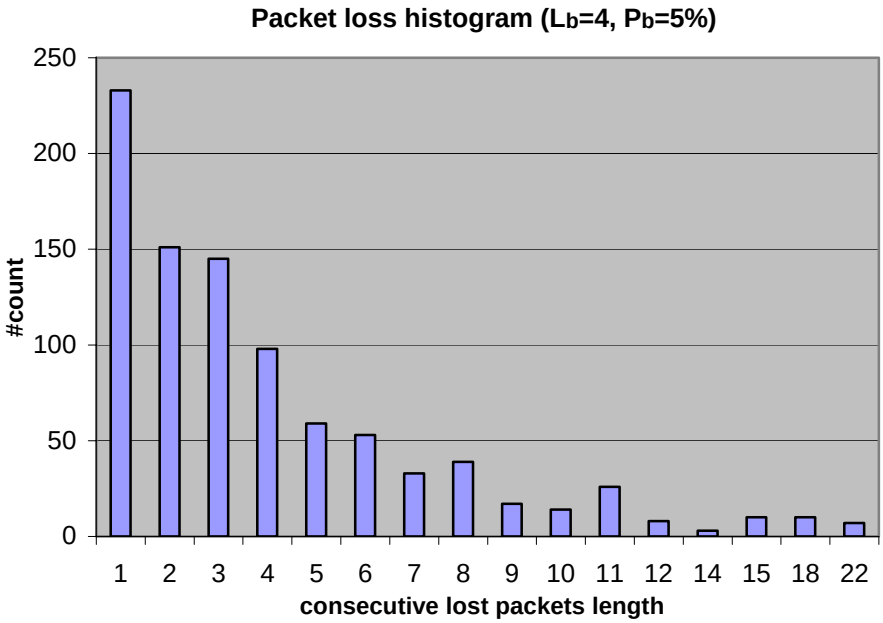

Figure 7-8: Packet loss histogram (L_b=4, P_b=5%)

Using the results from the preceding figures, we calculated the associated average burst length using the formula:

$$L_b = \frac{\sum_{i=1}^{n} count_i \cdot length_i}{\sum_{i=1}^{n} count_i}$$, where *i* is the number of unique pairs (7-1).

The results are presented in the following table, which as expected are almost identical with the average burst length used in the simulations.

Gilbert-Elliot Model	
Used L_b	Calculated L_b
2	1.9
3	3.02
4	4.1

Table 7-3: Calculated average burst length for the Gilbert-Elliot model

7.6.2 Three-State Markov Model Evaluation Results

We will continue with the presentation of the experimental results for the three-state Markov model. Again, the NL error rate associated with each one of the three ASR strategies is depicted in Figures 7-10, 7-11 and 7-12. The specific error rate is calculated with respect to the probability P_{GB} in the x-axis and the parameter k. Each graph contains four plots that correspond to the different values of k.

We should note that when k equals 0, the specific three-state Markov model becomes a Gilbert-Elliot model and no transitions to state LB take place. As k increases, the transition probability to state LB also increases and long error bursts appear in the packet stream.

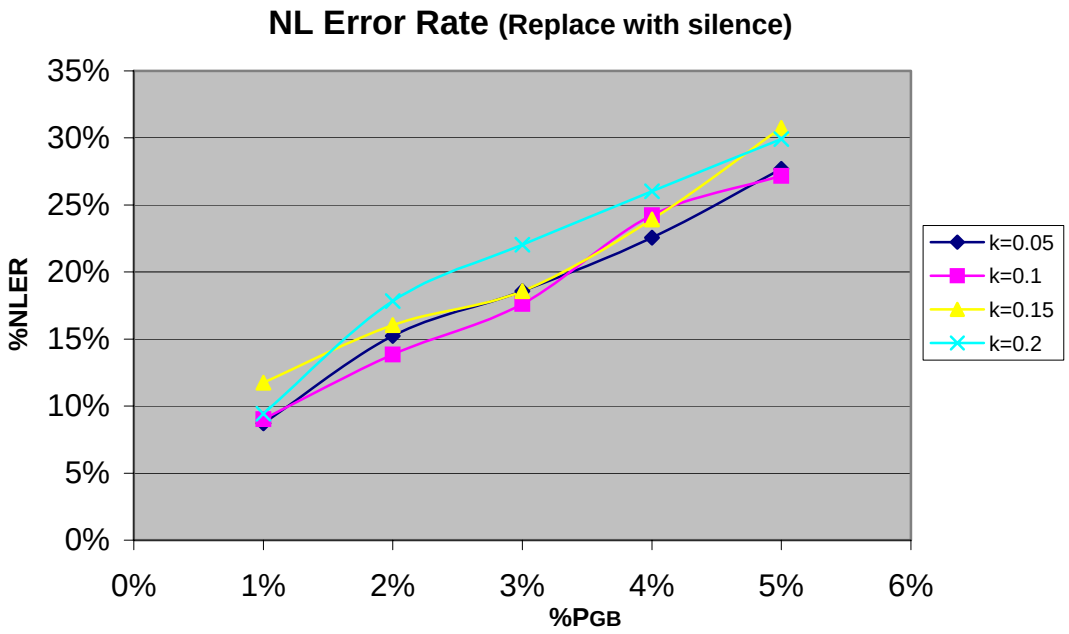

Figure 7-9: NL Error Rate for the first ASR strategy (Three-State Markov model)

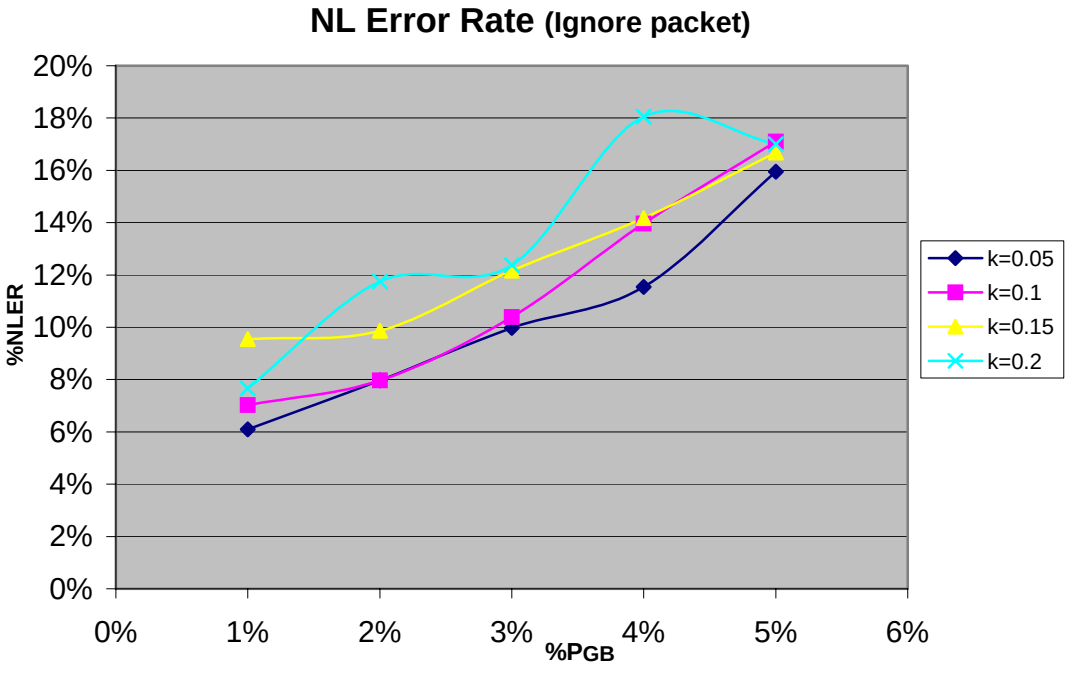

Figure 7-10: NL Error Rate for the second ASR strategy (Three-State Markov model)

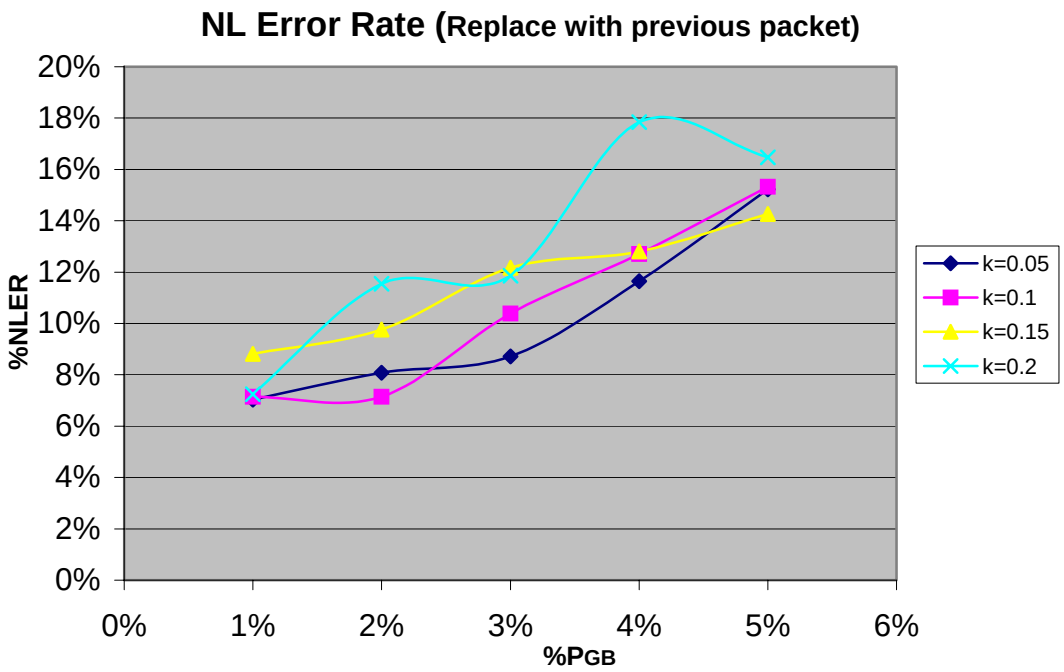

Figure 7-11: NL Error Rate for the third ASR strategy (Three-State Markov model)

Once again, four sample histograms are provided for the specific model (Figures 7-14,15,16,17). The graphs correspond to P_{GB}=2% and k=0.05, 0.1, 0.15, 0.2 respectively. The calculated average burst length, expressed by equation 7-1, is shown in the table below, where we can see a slow, as P_{GB} is rather small, but steady increase.

Three-State Markov model	
k	Calculated L_b
0.05	2.99
0.1	3.04
0.15	3.32
0.2	3.65

Table 7-4: Calculated average burst length for the Three-State Markov model

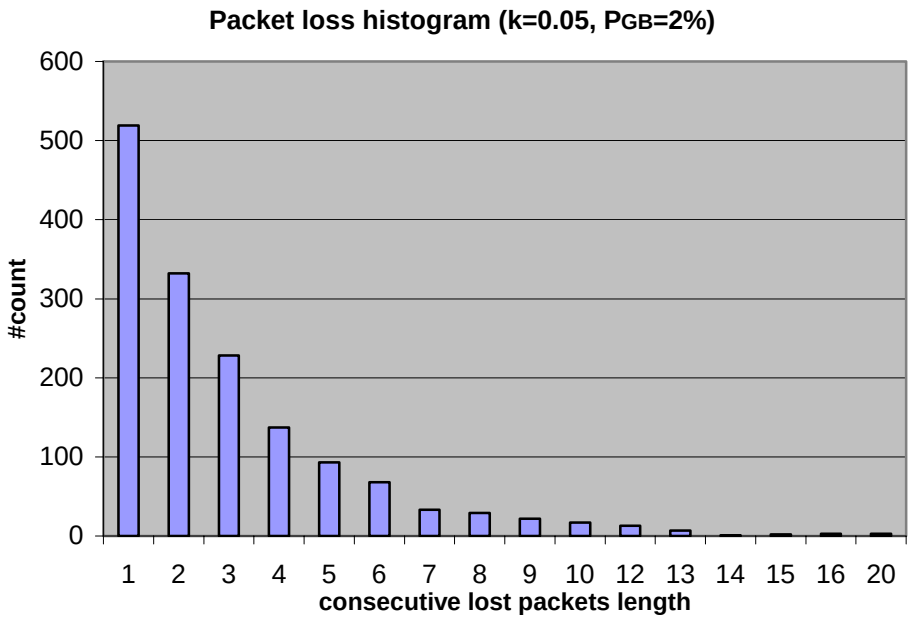

Figure 7-12: Packet loss histogram (k=0.05, P_{GB}=2%)

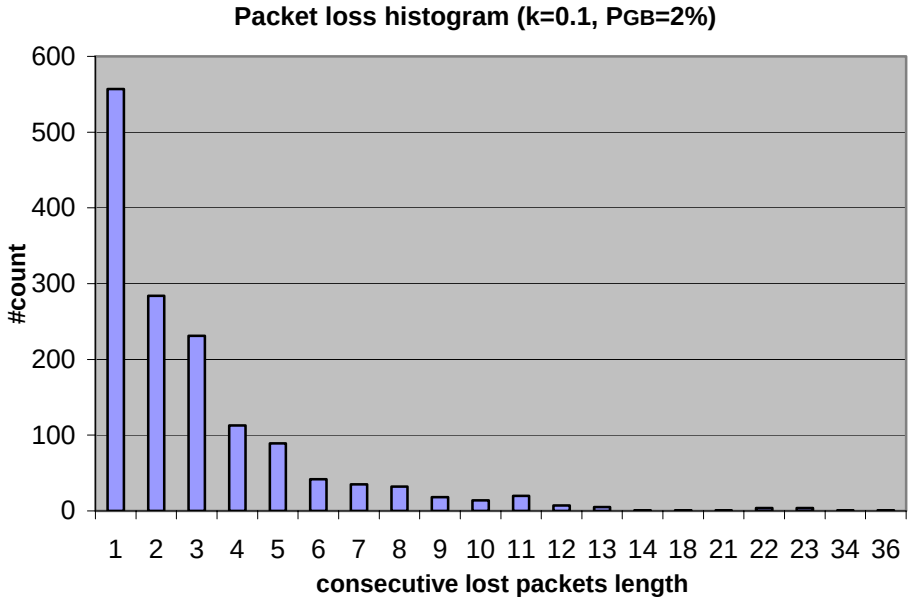

Figure 7-13: Packet loss histogram (k=0.1, P_{GB}=2%)

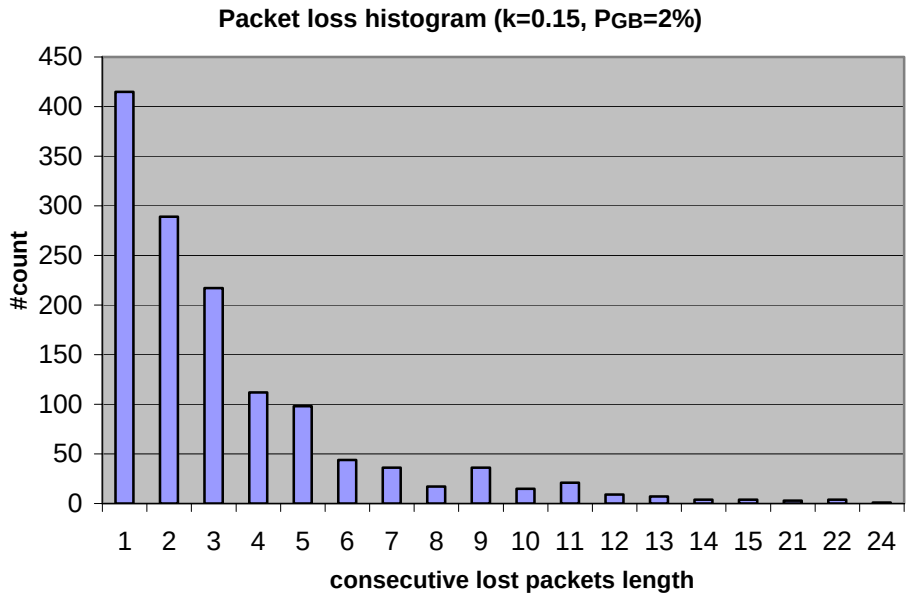

Figure 7-14: Packet loss histogram (k=0.15, P_{GB}=2%)

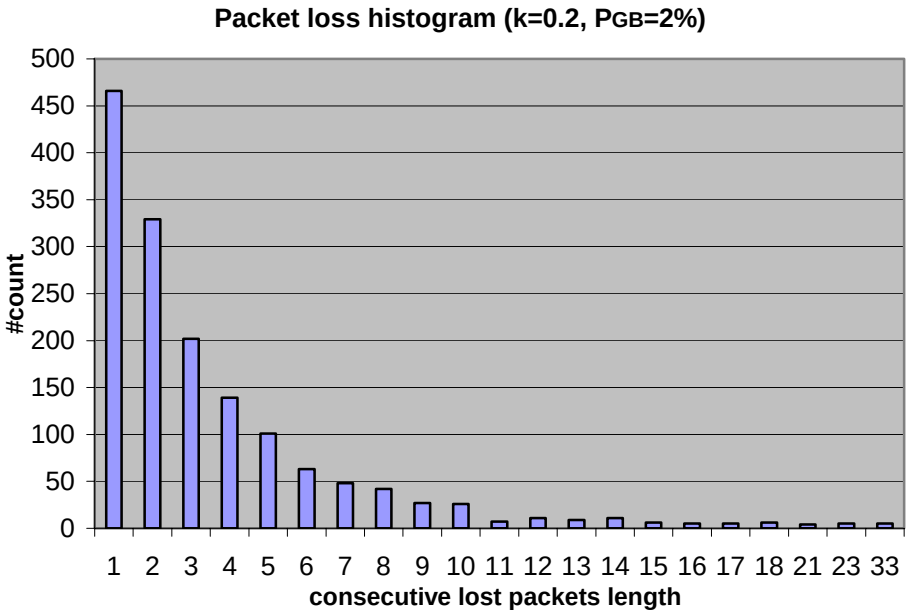

Figure 7-15: Packet loss histogram (k=0.2, P_{GB}=2%)

7.7 Conclusions

In the graphs of the NL error rate presented earlier, particularly in the three-state Markov model, we can observe that there is some measurement noise in the plots of each graph (i.e. Figure 7-16). We would expect, for example, that increased packet-loss probability and longer error bursts should always correspond to worst results. This is an indication of the statistical error appeared during the simulations.

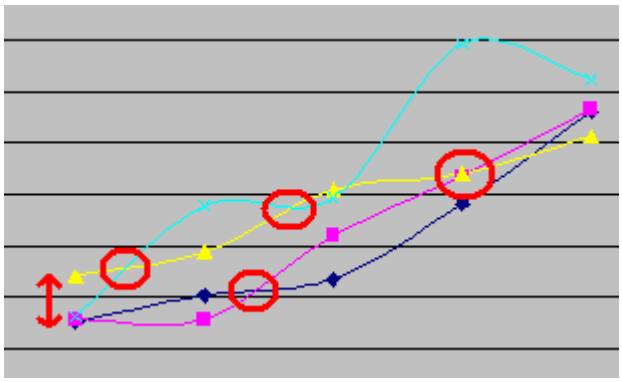

Figure 7-16: Statistical error

The calculation of the confidence intervals that correspond to our simulations could provide a quantification of the uncertainty in the measurements.

To decrease the statistical error, and since it is not feasible to have a significantly larger test set, we could create multiple simulations of the packet-loss on the same test set for each configuration. We can then average the results obtained for each configuration and present them in the corresponding graphs.

Comparing the results from the two models, we can conclude that the Gilbert-Elliot model definitely offers smaller NL error rates for the same packet loss probability. For example for $P_B=P_{GB}=5\%$ and using the strategy of replacing the lost packet with the previous one we yield to NLER between 5%-10% for the Gilbert-Elliot model and 14%-16% for the three-state Markov model. However the latter takes into account the bursty nature of noise in the data networks, that is when an error occurs it is more probable for a consequent error to happen. We, therefore, believe that the three-state Markov model better simulates the underlying data network.

If we now examine the distribution of error bursts in each model, we can observe, as expected, longer error bursts in the three-state Markov model that reach to a length of 36 consecutive packets for P_{GB}=2%, compared to 22 consecutive packets using P_B=5% in the Gilbert-Elliot model.

The configurations used during the experiments may not simulate exactly the conditions encountered in real mobile data networks. The aim of our effort was to obtain a rough estimation of the speech recognition performance, in order to identify the expectations of our system. The specific results can also be used as a reference during the tuning of the parameters of the network.

Chapter 8

CONCLUSIONS – FUTURE WORK

8.1 Conclusions

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 [19] over RTSP [20]. 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 [21], which provides more 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 [22] and the VoiceXML [23] specifications.

On the other hand we conducted several simulations in order to study the results of noisy data networks on the speech recognition performance of our system. As our implementation is closely related with mobile data networks, we wanted an estimation of how the variation of specific parameters can affect the aforementioned performance. We therefore incorporated two widely acceptable error models and demonstrated a series of graphs that depict the resulted performance.

8.2 Future work

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. Moreover the evaluation of our system can also be extended. In particular:

- 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 [24], [25] and [26]. 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. Error protection could also be added to the reduced transmission rates in order to improve recognition performance.
- The interaction experienced by the user can be improved with the personalization of the offered applications. In the stock information application for example, we could provide the ability for the creation of the user's portfolio. During the interaction with the system, the users can concentrate with the stocks of interest and obtain overall results about their variation.

- In a multimodal environment, the creation of the user's profile can be easier. Specific input mechanisms, like text forms on the mobile display, can be incorporated in order to collect user's personal data (gender, age etc.) or the user's experience level (native, intermediate or expert) of using applications of this kind. The specific information can be used in order to group the users into different categories, providing each group with a different way of interaction with the system
- 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. Moreover dynamic announcements can be rendered using a TTS.
- Additional work can also be performed on the analysis of the performance of the system in noisy data networks. Besides the two incorporated error models, one can use a number of available models that better simulate existent data networks (GPRS, 3G etc). Ideally, the evaluation should be performed on real voiceprints collected in the ASR server, which could be obtained after the recording on the mobile device and the transmission over the data network. The specific analysis would provide an objective measurement of the performance.
- Performance tests can also be conducted in terms of the usability of the system. For example, when and where each mode of input is preferable and why. The specific tests can help us identify possible problems during the interaction with the system or introduce desirable enhancements.

Appendix

Overview

In the current section we present the set of actions that we have already mentioned in the preceding chapters. These actions are incorporated during the communication between the mobile application and the Mediator Server, as described in Chapter 5. In particular, each action consists of a number of messages that are exchanged between the two peers. For each one of them, the detailed structure is offered followed by a corresponding example. The data type and a description of each field in the structure are also demonstrated. The flow of messages exchanged between the application and the server, with regard of time is also presented. We should repeat that four types of messages are available, specifically requests, acknowledgements, events and responses.

NEW ENGINE Action

- NEW ENGINE REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	String (Arguments)
DECIPHER_SRI	METHOD_NEW_ENGINE	REQ	10	-package C:\foo

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_NEW_ENGINE	ACK	10	0	STATUS_OK

- NEW ENGINE RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_NEW_ENGINE	RES	10	0	STATUS_OK

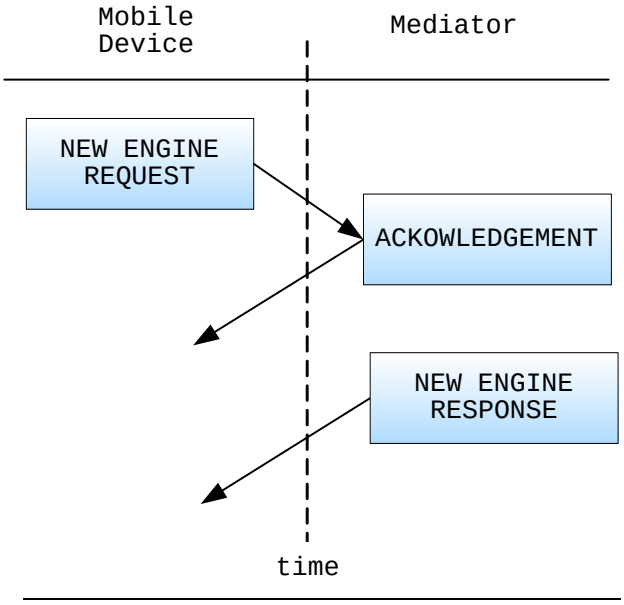

Figure 0-1: NEW ENGINE action

DELETE ENGINE Action

- DELETE ENGINE REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)
DECIPHER_SRI	METHOD_DELETE_ENGINE	REQ	10

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_DELETE_ENGINE	ACK	10	0	STATUS_OK

- DELETE ENGINE RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_DELETE_ENGINE	RES	10	0	STATUS_OK

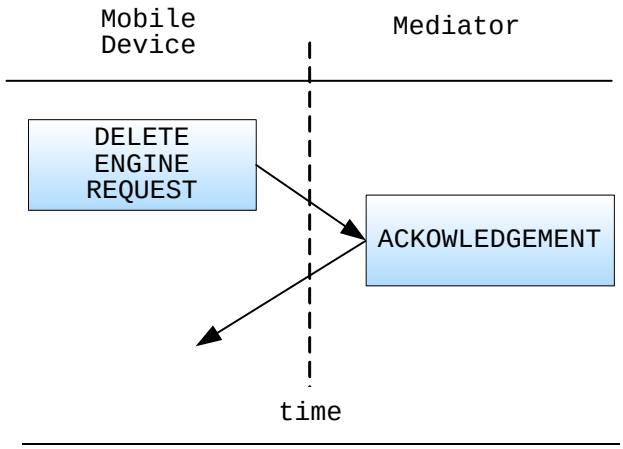

Figure 0-2: DELETE ENGINE action

SET INT PARAM Action

- SET INT PARAM REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	String (Parameter name)	Integer (Parameter Value)
DECIPHER_SRI	METHOD_SET_INT_PARAM	REQ	10	foo	22

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_SET_INT_PARAM	ACK	10	0	STATUS_OK

- SET INT PARAM RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_SET_INT_PARAM	RES	10	0	STATUS_OK

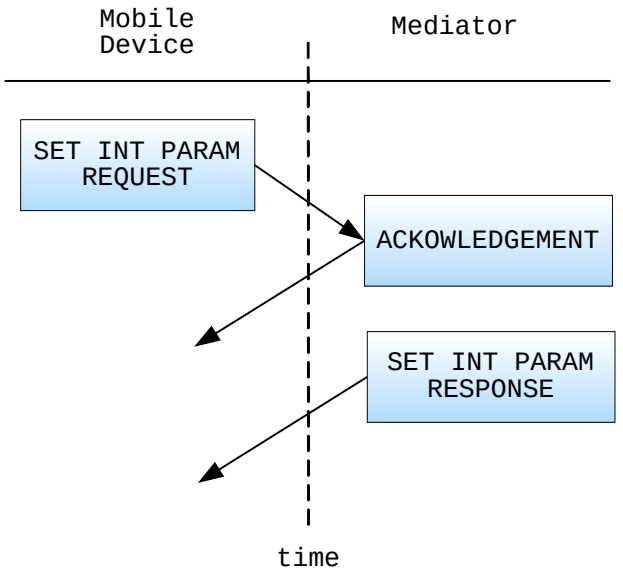

Figure 0-3: SET INT PARAM action

SET FLOAT PARAM Action

- SET FLOAT PARAM REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	String (Parameter name)	Float (Parameter Value)
DECIPHER_SRI	METHOD_SET_FLOAT_PARAM	REQ	10	foo	2.2

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_SET_FLOAT_PARAM	ACK	10	0	STATUS_OK

- SET INT PARAM RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_SET_FLOAT_PARAM	RES	10	0	STATUS_OK

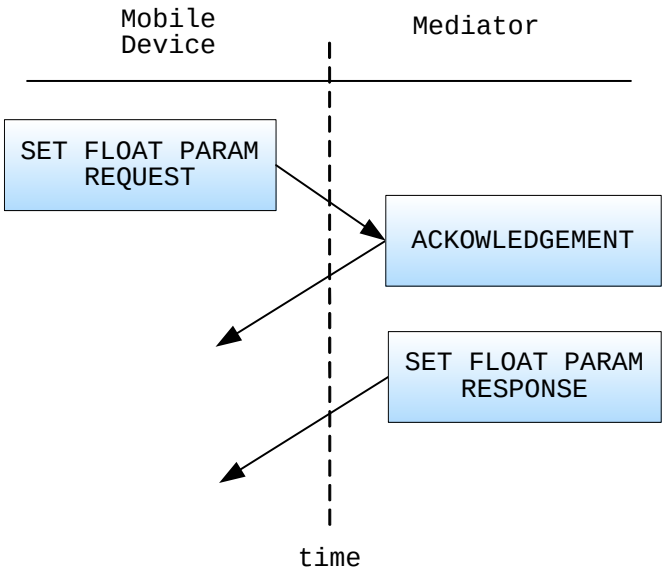

Figure 0-4: SET FLOAT PARAM action

SET STRING PARAM Action

- SET STRING PARAM REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	String (Parameter name)	String (Parameter Value)
DECIPHER_SRI	METHOD_SET_STRING_PARAM	REQ	10	foo	Value

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_SET_STRING_PARAM	ACK	10	0	STATUS_OK

- SET INT PARAM RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_SET_STRING_PARAM	RES	10	0	STATUS_OK

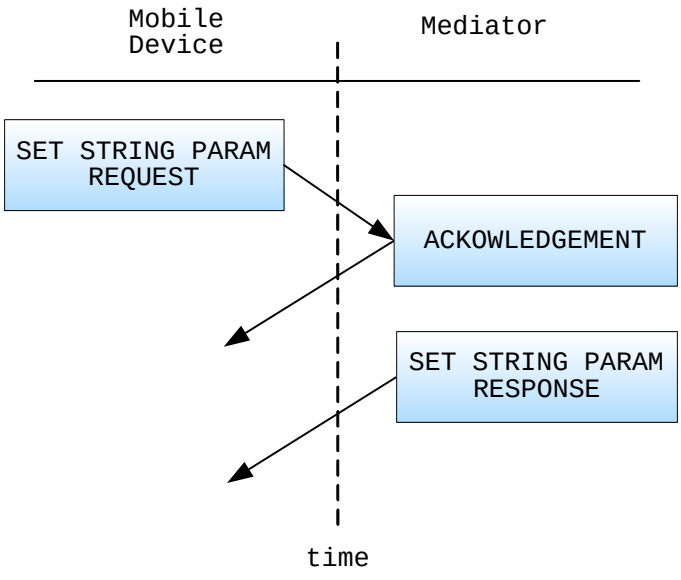

Figure 0-5: SET STRING PARAM action

GET INT PARAM Action

- GET INT PARAM REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	String (Parameter name)
DECIPHER_SRI	METHOD_GET_INT_PARAM	REQ	10	foo

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_GET_INT_PARAM	ACK	10	0	STATUS_OK

- GET INT PARAM RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)	Integer (Parameter Value)
DECIPHER_SRI	METHOD_GET_INT_ PARAM	RES	10	0	STATUS_OK	22

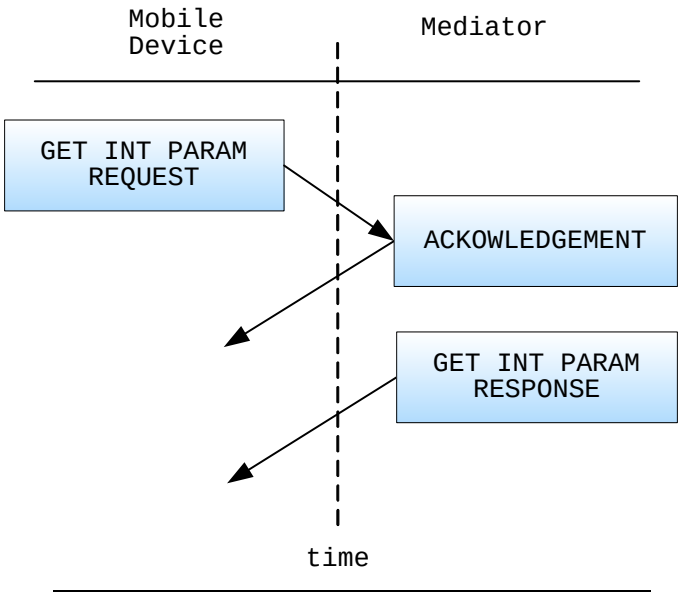

Figure 0-6: GET INT PARAM action

GET FLOAT PARAM Action

- GET FLOAT PARAM REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	String (Parameter name)
DECIPHER_SRI	METHOD_GET_FLOAT_PARAM	REQ	10	foo

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_GET_FLOAT_PARAM	ACK	10	0	STATUS_OK

- GET INT PARAM RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)	Float (Parameter Value)
DECIPHER_SRI	METHOD_GET_ FLOAT_PARAM	RES	10	0	STATUS_OK	2.2

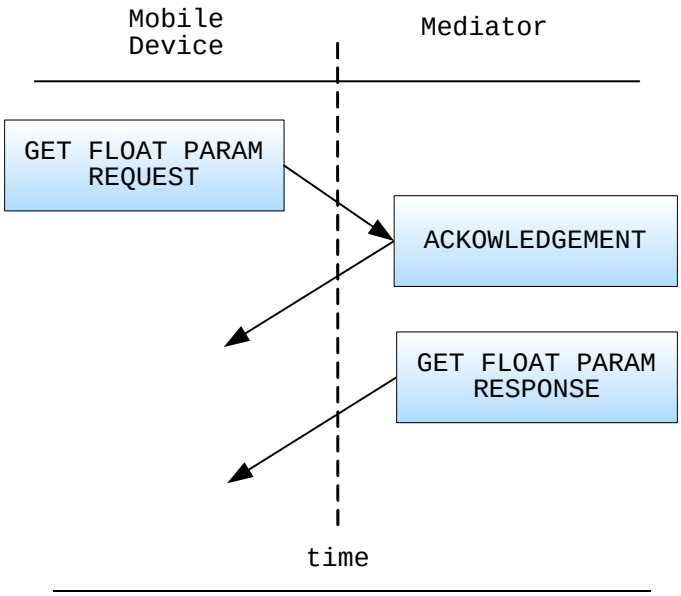

Figure 0-7: GET FLOAT PARAM action

GET STRING PARAM Action

- GET STRING PARAM REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	String (Parameter name)
DECIPHER_SRI	METHOD_GET_STRING_PARAM	REQ	10	foo

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_GET_STRING_PARAM	ACK	10	0	STATUS_OK

- GET STRING PARAM RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)	String (Parameter Value)
DECIPHER_SRI 5	METHOD_GET_STRING_PARAM	RES	10	0	STATUS_OK	Value

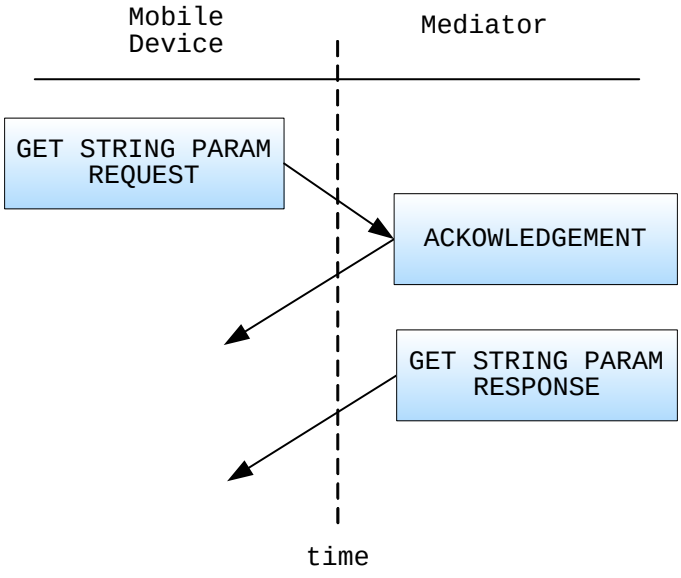

Figure 0-8: GET STRING PARAM action

RECOGNIZE Action

- RECOGNIZE REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	String (Context)	Integer (Hot word)
DECIPHER_SRI	METHOD_RECOGNIZE	REQ	10	.FOO	0

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_RECOGNIZE	ACK	10	0	STATUS_OK

- START OF SPEECH EVENT

String (Platform)	String (Event)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	EVENT_START_OF_SPEECH	EVT	10	0	STATUS_OK

- END OF SPEECH EVENT

String (Platform)	String (Event)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	EVENT_END_OF_SPEECH	EVT	10	0	STATUS_OK

- RECOGNITION RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)	String (Result)
DECIPHER_SRI	METHOD_RECOGNIZE	RES	10	0	STATUS_OK	New York

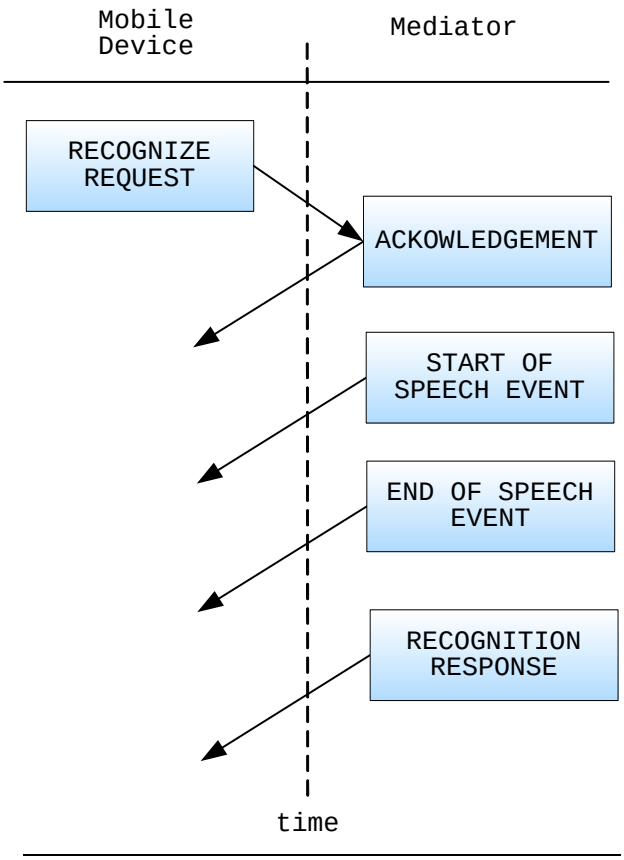

Figure 0-9: RECOGNIZE action

INTERPRET Action

- INTERPRET REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	String (Context)	String (Text)
DECIPHER_SRI	METHOD_INTERPRET	REQ	10	.Foo	Text

- ACKNOWLEDGEMENT

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)
DECIPHER_SRI	METHOD_INTERPRET	ACK	10	0	STATUS_OK

- INTERPRET RESPONSE

String (Platform)	String (Method)	String (Message type)	Integer (Action id)	Integer (Status)	String (Status description)	String (NL result)
DECIPHER_SRI	METHOD_INTERPRET	RES	10	0	STATUS_OK	New York

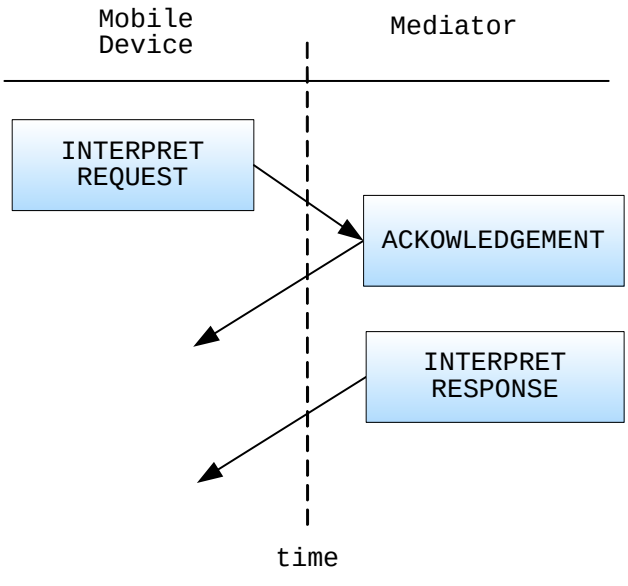

Figure 0-10: INTERPRET action

ABORT Action

- ABORT REQUEST

String (Platform)	String (Method)	String (Message type)	Integer (Action id)
DECIPHER_SRI	METHOD_ABORT	REQ	10

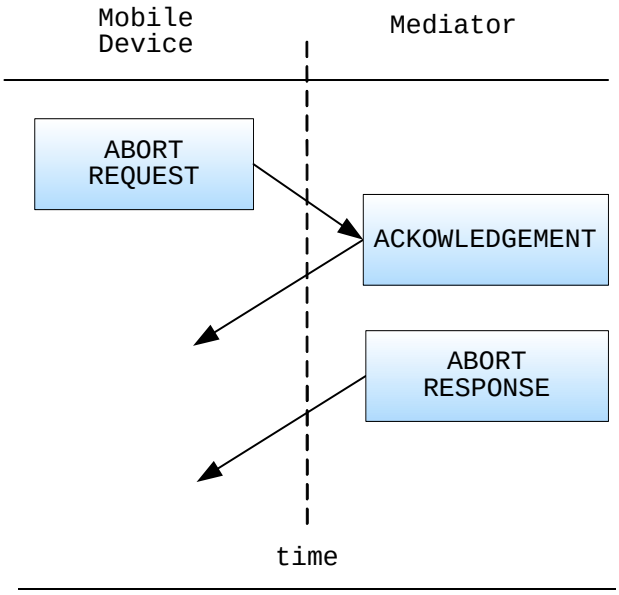

Figure 0-11: ABORT action

ABBREVIATIONS

3GPP	3rd Generation Partnership Project
API	Application Programming Interface
CDMA	Code Division Multiple Access
CGI	Common Gateway Interface
CPL	Call Processing Language
CSR	Continuous Speech Recognition
DNS	Domain Name System
DOM	Document Object Model
DSR	Distributed Speech Recognition
EDGE	Enhanced Data Rates for Global Evolution
GPRS	General Packet Radio Service
GSM	Global System for Mobile communication
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IETF	Internet Engineering Task Force
IMS	IP Multimedia Subsystem
IP	Internet Protocol
ISDN	Integrated Services Digital Network
JSP	Java Server Pages
LAN	Local Area Network
MMS	Multimedia Messaging Service
NAT	Network Address Translation
NFS	Network File System
NLER	Natural Language Error Rate
ODBC	Open Database Connectivity
OMA	Open Mobile Alliance
PDA	Personal Digital Assistant
PoC	Push to Talk
PSS	Packet-switched Streaming Service

QoS	Quality of Service
RTP	Real Time Transport Protocol
RTVS	Real-Time Video Sharing
SALT	Speech Application Language Tags
SDK	Software Development Kit
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SLU	Spoken Language Understanding
SMIL	Synchronized Media Integration Language
SMS	Short Message Service
SNMP	Simple Network Management Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
VDU	Visual Display Unit
VoIP	Voice over IP
W3C	World Wide Web Consortium
WCDMA	Wideband Code Division Multiple Access
WER	Word Error Rate
WIMP	Windows, Icons, Menus and Pointer
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

BIBLIOGRAPHY

[1] Luan Dang, Cullen Jennings and David Kelly. Practical VoIP Using Vocal. O' Reilly & Associates, Inc., 2002.

[2] Dan Harkey, Shan Appajodu and Mike Larkin. Wireless Java Programming for Enterprise Applications. Wiley Publishing, Inc., 2002.

[3] SIP Working Group at http://www.ietf.org/html.charters/sip-charter.html.

[4] Audio-Video Transport Working Group, H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications (RFC: 1889)," tech. rep., Network Working Group, http://src.doc.ic.ac.uk/-packages/rfc/rfc1889.txt, January 1996.

[5] Developing Series 60 Applications. A Guide for Symbian OS C++ Developers. Leigh Edwards, Richard Barker, and the Staff of EMCC Software Ltd.

[6] User's Guide for Series 60 SIP Plug-in, Nokia, May 17 2004.

[7] E. N. Gilbert, "Capacity of a burst-noise channel," Bell Syst. Tech. J., Vol. 39, pp. 1253-1265, September 1960.

[8] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels", Bell Syst. Tech. J., Vol. 42, pp. 1977-1997, September 1963.

[9] Cain J. B., Simpson R. S., The Distribution of Burst Lengths on a Gilbert Channel, IEEE Trans IT-15 Sept 1969.

[10] Blank H. A, Trafton P. J., A Markov Error Channel Model, Proc Nat Telecomm Conference 1973.

[11] Mertz P., Statistics of Hyperbolic Error Distributions in Data Transmission, IRE Trans CS-9, Dec 1961.

[12] Berger J. M., Mandelbrot B. A New Model for Error Clustering in Telephone Circuits. IBM J R&D July 1963.

[13] Lewis P, Cox D., A Statistical Analysis of Telephone Circuit Error Data. IEEE Trans COM-14 1966.

[14] Boutremans C., Iannaccone G., Diot C., Impact of Link Failures on VoIP Performance, Sprint Labs technical report IC/2002/015.

[15] "Short-term Fairness for TCP Flows in 802.11b WLANs", M. Bottigliengo, C. Casetti, C.-F. Chiasserini, M. Meo. CERCOM – Dipartimento di Elettronica, Politecnico di Torino, Italy.

[16] "A Markov-based Channel Model Algorithm for Wireless Networks", Almudena Konrad, Ben Y. Zhao, Anthony D. Joseph, Reiner Ludwig

[17] H. S. Wang, P. Chang, "On Verifying the First-Order Markovian assumption for a Rayleigh Fading Channel Model," IEEE Transactions on Vehicular Technology, Vol. 45, No. 2, pp. 353- 357, May 1996.

[18] M. Zorzi, R. Rao, and L. Milstein, "On the accuracy of a first-order Markov Model for data transmission on fading channels," In Proceedings of IEEE ICUPC, pp. 211–215, November 1995.

[19] 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.

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

[21] "Symbian OS, the mobile operating system", http://www.symbian.com/.

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

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

[24] 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.

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

[26] 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.

[27] Bjarne Stroustrup, **"**An Overview of the C++ Programming Language", AT&T Laboratories.

[28] "Transmission Control Protol", Darpa Internet Prorgam, Protocol Specification, September 1981, http://www.ietf.org/rfc/rfc0793.txt.

[29] "User Datagram Protocol", J. Postel, ISI, August 1980, http://www.ietf.org/rfc/rfc0793.txt.

[30] "Network File System (NFS) version 4 Protocol", S. Shepler et al, April 2003, http://www.ietf.org/rfc/rfc3530.txt.

[31] "A Simple Network Management Protocol (SNMP)", J. Case et al, May 1990, http://www.ietf.org/rfc/rfc1157.txt.

[32] "Domain Names - Concepts and Facilities", P. Mockapetris, ISI, November 1987, http://www.ietf.org/rfc/rfc1034.txt.

[33]" The TFTP Protocol (Revision 2)", K. R. Sollins, MIT, June 1981, http://rfc.net/rfc783.html.

[34] "RTP: A Transport Protocol for Real-Time Applications", H. Schulzrinne et al, July 2003, http://www.ietf.org/rfc/rfc3550.txt.

[35] Clark, D. and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols", in SIGCOMM Symposium on Communications Architectures and Protocols, (Philadelphia, Pennsylvania), pp. 200-208, IEEE Computer Communications Review, Vol. 20(4), September 1990.

[36] "Hypetext Transfer Protocol – HTTP/1.1", R. Fielding et al, June 1999, http://www.ietf.org/rfc/rfc2616.txt.

[37] Berners-Lee, T., Fielding, R. and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.

[38] Masinter, L., "Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)", RFC 2324, 1 April 1998.

[39] Berners-Lee, T., "Universal Resource Identifiers in WWW", RFC 1630, June 1994.

[40] Berners-Lee, T., Masinter, L. and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[41] Sollins, K. and L. Masinter, "Functional Requirements for Uniform Resource Names", RFC 1737, December 1994.

[42] Crocker, D., "Standard for The Format of ARPA Internet Text Messages", STD 11, RFC 822, August 1982.

[43] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[44] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982.

[45] Kantor, B. and P. Lapsley, "Network News Transfer Protocol", RFC 977, February 1986.

[46] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, October 1985.

[47] Anklesaria, F., McCahill, M., Lindner, P., Johnson, D., Torrey, D. and B. Alberti, "The Internet Gopher Protocol (a distributed document search and retrieval protocol)", RFC 1436, March 1993.

[48] Davis, F., Kahle, B., Morris, H., Salem, J., Shen, T., Wang, R., Sui, J., and M. Grinbaum, "WAIS Interface Protocol Prototype Functional Specification", (v1.5), Thinking Machines Corporation, April 1990.

[49] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

[50] "SDP: Session Description Protocol", M. Handley & V. Jacobson, ISI/LBNL, April 1998, http://www.ietf.org/rfc/rfc2327.txt.

[51] Handley, M., "SAP - Session Announcement Protocol", Work in Progress.

[52] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998, http://www.ietf.org/rfc/rfc2326.txt.

[53] "SIP: Session Initiation Protocol", J. Rosenberg et al, June 2002, http://www.ietf.org/rfc/rfc3261.txt.

[54] R. Pandya, "Emerging mobile and personal communication systems", IEEE Communications Magazine, Vol. 33, pp. 44--52, June 1995.

[55] Cuervo, F., Greene, N., Rayhan, A., Huitema, C., Rosen, B. and J. Segers, "Megaco Protocol Version 1.0", RFC 3015, November 2000.

[56] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[57] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[58] "A Media Resource Control Protocol (MRCP), Developed by Cisco, Nuance, and Speechworks", S. Shanmugham et al, April 2006, http://www.rfc-editor.org/rfc/rfc4463.txt.

[59] MITRE, DARPA communicator. http://fofoca.mitre.org.

[60] Lemon, O., Bracy, A., Gruenstein, A., and Peters, S. The "WITAS multi-modal dialogue system I". In Proc. of Eurospeech01, Aalborg, DK, 2001.

[61] Cohen, P., Johnston, M., McGe e, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. "QuickSet: Multimodal interaction for distributed applications". In Proc. of the Fifth ACM International Multimedia Conference, pages 31–40, New York. ACM Press, 1997.

[62] Thorpe, J. A. The new technology of large scale simulator networking: Implications for mastering the art of warfighting 9th InterserviceTraining Systems Conference, 1987, 492-501.

[63] Wahlster, W., Reithinger, N., and Blocher, A. "Smartkom: Multimodal communication with a life-like character". In Proc. of Eurospeech01, Aalborg, DK, 2001.

[64] MUST project webpage: http://www.eurescom.de/public/projects/P1100-series/p1104.

[65] Roessler, H., et al. "Multimodal interaction for mobile environments". In Proc. of International Workshop on Information Presentation and Natural Multimodal Dialogue, Verona, IT, 2001.

[66] "Device independent mobile multimodal user interfaces with the MONA Multimodal Presentation Server", Georg Niklfeld, Hermann Anegg, Michael Pucher, Raimund Schatz, Rainer Simon, Florian Wegscheider, Alexander Gassner, Michael Jank, Günther Pospischil, Proceedings of Eurescom Summit 2005, Heidelberg, Germany, April 27-29, 2005.

[67] Huang, X. et al. "MIPAD: A next generation PDA prototype". In Proc. of the Int. Conf. on Spoken Language Processing (ICSLP), Beijing, 2000.

[68] W3C Recommendation, 2000, "XHTML™ 1.0 The Extensible HyperText Markup Language", http://www.w3.org/TR/xhtml1/.

[69] W3C Recommendation, 1998, "Synchronized Multimedia Integration Language (SMIL) 1.0", http://www.w3.org/TR/REC-smil/.

[70] W3C, Document Object Model, http://www.w3.org/DOM/.

[71] DARPA Internet Program, "INTERNET PROTOCOL – Protocol Specification", September 1981, http://www.ietf.org/rfc/rfc0791.txt.

[72] W3C Recommendation, 2000, "Document Object Model (DOM) Level 2 Core Specification", http://www.w3.org/TR/DOM-Level-2-Core/.

[73] W3C, "HyperText Mark Language", http://www.w3.org/MarkUp/.

[74] H. Murveit, J. Butzberger, V. Digalakis and M. Weintraub. "Large Vocabulary Dictation Using SRI's DECIPHER Speech Recognition System: Progressive Search Techniques", 1993 IEEE ICASSP pp. II-319-II-322.

[75] V. Digalakis, P. Monaco and H. Murveit. "Genones: Generalized Mixture Tying in Continuous Hidden Markov Model-Based Speech Recognizers". IEEE Trans. on Speech and Audio Proc., pp. 281-289, July 1996.