TEXT-TO-SPEECH ENGINES AS TELECOM SERVICE ENABLERS

Mihai SURMEI[*], Dragoş BURILEANU[**], Cristian NEGRESCU[**],
Răzvan PÎRVU[*], Cătălin UNGUREAN[**], Aurelian DERVIŞ[**]

[*] ERICSSON Telecommunications Romania S.R.L.
[**] University "Politehnica" of Bucharest, Faculty of Electronics, Telecommunications and Information Technology

Corresponding author: Dragoş BURILEANU E-mail: bdragos@mESsnet.pub.ro

As customer demand for more convenient access to a variety of services in network-based applications increases, speech synthesis technology is becoming more important for service providers. Several companies already use text-to-speech (TTS) synthesis in offering diverse information to users over the telephone line. In most of these cases the synthesis engine uses a client-server architecture, based on either proprietary hardware/software interfaces or open standards. The paper investigates the possible solutions in applying the TTS technology as a telecommunication service enabler and presents a case study of using a TTS system in Romanian language in an e-mail reader application. An important implementation issue, namely automatic diacritic restoration, is also addressed. The paper finally proposes a TTS server architecture suitable to a production environment.

Key words: Text-to-speech synthesis; Client-server architecture; Open protocols; SIP/RTP; Media resource server; Inter-process communication; E-mail/SMS reading.

1. INTRODUCTION

Although there are several other modalities by which a human can have interactions with a machine, as for example keyboard, mouse, touch screen, etc., it is obvious that speech is one of the most intuitive and natural modality of communication. The argument for spoken language interfaces is further strengthened by their ability to allow for hands-free and eyes-free operation where it would be impractical, unsafe, or even impossible to exchange information in a different manner.

Speech technology could be easily adopted and leveraged in various areas where ease of communication is a key performance indicator. However, a voiced interaction with a non-human peer is a complex task, involving all relevant domains of speech technology such as synthesis, recognition and speaker identification/verification.

The areas of interest could be completely independent, e.g., database information retrieval, car navigation aid, health or public administration, but the speech services to be integrated has more or less the same requirements to fulfill. One approach to unify this is to abstract speech technology as a telecommunication service enabler.

While telephone speech recognition, for example, is already the largest market for speech technology, it must be noticed that there is a broad range of services and products that already integrate synthesized speech into their communication facilities. The subsequent discussion will be mainly focused on network-based speech synthesis solutions.

In many network-based applications one cannot predict the message that needs to be spoken, and the system must generate sentences from arbitrary text (database records, e-mail messages, etc.). This task can be accomplished only by text-to-speech synthesis systems, which must provide at least a very good intelligibility for the resulting speech to be helpful and accepted by the user.

On the other hand, because the development of highly natural synthesis systems, starting from unrestricted text input (also known as "open-domain synthesis" systems) still represents an extremely difficult task, industry speech application developers are making notable efforts to design and offer good quality TTS solutions [3].

One could imagine different service types based on the TTS synthesis technology.

- *Notify/warning services*: news reading or location related traffic information are suggestive examples. This type of service means pushing the information to the common (voice only) user terminal.
- *Legacy terminal adaptation*: means reading e-mails or SMS's from old POTS (Plain Old Telephony System) terminals. In this case the user is popping the information from the e-mail server or from the Service Center; this service could be applied for any text source.
- *Accessibility*: developing voiced services for people with disabilities, by pushing or popping the information from the relevant servers.

Another dimension to take into consideration is the manner to implement such services.

- *Embedded systems*: the user terminal performs the entire speech synthesis task.

It is clear that for complete in-device solutions, the bandwidth requirement is very low as only text will be carried out between peers; in addition, some of the services (e.g., notify/warning) are very well suited for this approach. However, there are also important drawbacks: computing and memory resource constraints, robustness in adverse environments, or cost and power consumption limitations are serious difficulties to deal with [3]. Besides, it is not possible to implement some services (e.g., legacy terminal adaptation) and finally, it is not easy to ensure intellectual property for the software implementation.

- *Network-based systems*: the speech processing is centralized and maintained by a trusted entity, such as a fixed/mobile telephony operator.

The main benefits of this approach can be easily revealed: all of the previously discussed services could be implemented using this approach, the intellectual property is granted, and a new service could be easily deployed using the operator's channels. The obvious limitation is that a voice channel between user and speech server must be established, leading to higher communication costs.

It is important to notice that none of the approaches gives the final answer and perhaps the best solution is a mixed architecture to cope with all the requirements. Further on will discuss only the network-based approach.

2. NETWORK CENTRIC TEXT-TO-SPEECH

One approach to a network centric TTS engine architecture is to use a *proprietary client-server interface*, with proprietary API's [6], [12]. Figure 1 shows this concept.

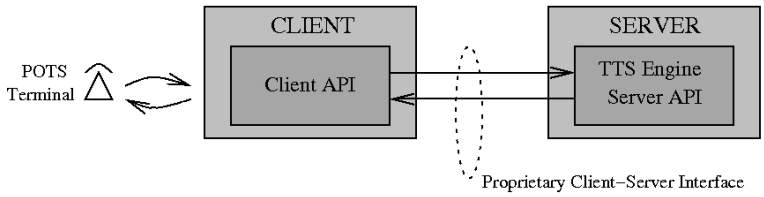

Figure 1. A proprietary client-server TTS-based interface.

This solution could be arbitrarily optimized, leading to high performance systems. The main drawback is the proprietary aspect, leading to difficulties in inter-working with other vendors. For instance, the client side must include and compile a specific library set developed by the server side to ensure high performance.

Another approach is based on *open protocols* (and possible future standards) that guarantee the interoperability of the particular product with any other application platform that implements the same industry standards [11], [12]. This is the actual trend followed by the majority of the players in the telecom industry. To address this need, IETF (Internet Engineering Task Force) organization proposed in the SPEECH Services Control Work Group the *Media Resource Control Protocol version 2* (MRCP v2) [10], currently in a draft form but increasingly adopted by TTS vendors.

This service architecture is presented in Figure 2, which is an excerpt from the draft.

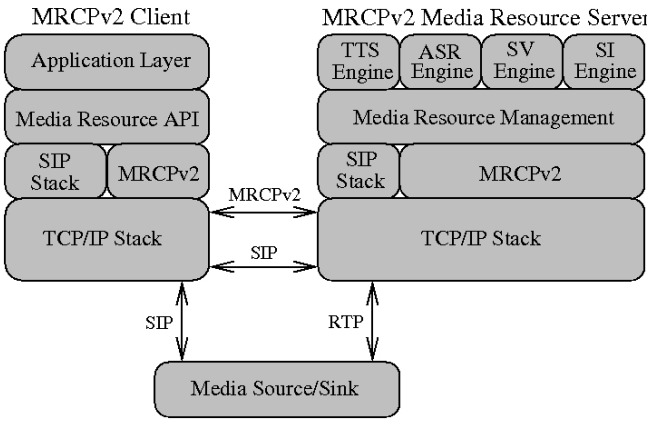

Figure 2. A client-server architecture based on open protocols.

Basically, a system using MRCPv2 consists of a *client* that requires the generation and/or consumption of media streams and a *media resource server* that has the resources or "engines" to process these streams as input or generate them as output. Some of these media processing resources could be speech synthesis engines, automatic speech recognition (ASR) engines, speaker verification (SV) and speaker identification (SI) engines.

The protocol requirements dictate that the client should be capable of reaching a media processing server and setting up communication channels to the media resources, and of sending and receiving control messages and media streams to/from the server. The *Session Initiating Protocol* (SIP) [8] is the signaling protocol that meets these requirements; in fact, we must emphasize that SIP is fast becoming a global standard for any kind of media signaling. MRCPv2 leverages these capabilities by building upon SIP and the *Session Description Protocol* (SDP) [4]. MRCPv2 uses SIP to setup and tear down media and control sessions with the server, and SDP to describe the parameters of the media sessions associated with the dialog between the client and server. In conclusion, MRCPv2 uses SIP and SDP to create the client/server dialog and set up the media channels to the server.

Choosing a protocol depends to a great extent on the existing networks and their evolution in the near future. Figure 3 shows the main network types currently in use. It is a starting point in thinking how text-to-speech technology will fit the reality.

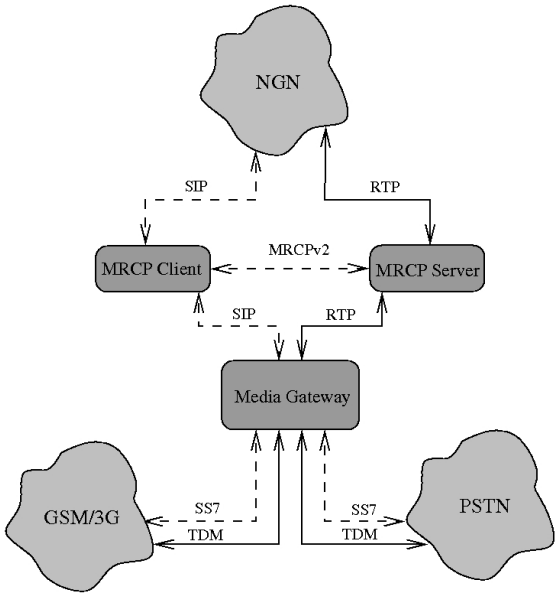

Figure 3. The main network types currently in use.

The networks are evolving towards an all-IP (Internet Protocol) paradigm but this is a slow and costly process [5]. In Figure 3, both PSTN (Public Switching Telephony Network) and GSM (Global System for Mobile Communications)/3G (Third Generation) networks are migrating gradually towards basically a VoIP (Voice over Internet Protocol) network, described as *Next Generation Networks*, NGN's. The *media gateway* is playing a fundamental role in a hybrid network, converting circuit switch control and user plane to packet switch domain. The figure depicts also the signaling – SIP and SS7 (Signaling System no. 7), and the payload – TDM (Time Division Multiplex) and RTP (Real Time Protocol) [9] flows. The trend is to migrate TDM/SS7 based networks towards SIP/RTP architectures.

Using SIP/RTP based protocols together with a media gateway will make a TTS implementation future proof. Moreover, this approach ensures that various components such as media streaming server, call-routing engines and user location server can be combined in various ways to create new service architectures.

3. AN E-MAIL READER APPLICATION USING TTS SYNTHESIS IN ROMANIAN

3.1. The demo platform

Today, e-mail is maybe the most common form of electronic communication. However, the convenience of this particular form of sharing information is limited by the necessity of an Internet connected computer. A service that allows for accessing e-mail by phone enhances the user experience, since information can be retrieved practically anytime and from everywhere [11]. As stated in Chapter 1, TTS technology can play an important role in this network-based service.

Our collective has a long experience in concatenative TTS synthesis. A desktop (PC-based) TTS system in Romanian language is available for several years [1], [2]. The newest version of this system was used to develop an e-mail reader application.

We have implemented a single threaded demo platform. The application architecture is illustrated in Figure 4.

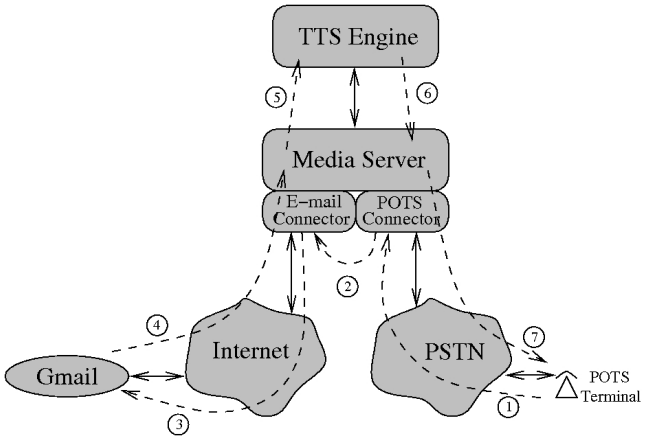

Figure 4. A demo platform based on TTS synthesis in Romanian.

The platform permits to listen the latest message from a Google Gmail account by accessing the platform from a legacy POTS terminal (voice only capable). This is a simple and straightforward example of a convergent service involving PSTN and Internet domains.

The TTS engine could be functionally decomposed into the TTS algorithms, written in C language, and the HTTP (Hypertext Transfer Protocol) server. The internal logical connection is realized using Perl (Practical Extraction and Reporting Language) scripts. The TTS engine is running on a Windows operating system.

Media server is a Linux box running a collection of Perl scripts, where the main script keeps track of the platform FSM (Finite State Machine). As one can notice, the figure is showing a *northbound connection* towards the TTS engine and a *southbound connection* towards Internet and PSTN domains.

The southbound connectors permit:

- E-mail fetching: the e-mail connector will open a POP3 (Post Office Protocol) session to the specified account.
- Calling the platform using an access number: the POTS connector includes a logic to implement a simple IVR (Interactive Voice Response) application. At the beginning, the IVR will prompt a pre-recorded message. After the newest e-mail is fetched and the header is parsed, the sender, the subject and the body of the message are identified, synthesized and presented to the user. The e-mail can be repeated for a predefined number of times. If there aren't any new messages, then the last message can be played.

The northbound interface is using Perl modules implementing the HTTP stack.

The main phases of the call (as showed in Figure 4) are the following:

① The demo platform is triggered by dialing a POTS access number. The media server handles the session and treats all the events that might occur on the line.

② The media server control logic spawns the e-mail connector.

③ The e-mail connector opens the session towards a predefined Gmail account using a POP3 protocol.

④ The last message is fetched from Gmail.

⑤ The media server main script terminates the e-mail connector, opens an HTTP connection to the TTS engine and sends a message containing the text to be translated.

⑥ The HTTP response will contain the resulted audio file encoded as a binary octet stream; the payload will be stored on the media server.

⑦ The media server will fetch the audio file from the disk, re-encode and send it back over the POTS line; the POTS connector will be initialized to the waiting state.

Even if this platform is a prototype, it is a good starting point for making the list of main features for a real TTS media server; this subject will be discussed in Chapter 4.

3.2. TTS algorithm implementation issues

Besides the requirement for an extremely good intelligibility, necessary to deal with the reduced telephone bandwidth and other channel degradation factors, an e-mail or SMS reader application based on TTS technology needs to meet another important constraint, namely handling the missing diacritics problem.

Usually, the majority of users still disregard the *diacritics* (marks above, through, or below letters), even if the terminal (or the operating system) permits this operation, or the diacritics are simply stripped by a computational process that use 7-bit forms [13]. This is the case for Romanian, but also for many languages that use the Latin alphabet (such as French, Spanish, German, Hungarian, Polish, Swedish, etc.) and supplement the basic set by making use of letters with diacritics (or accent marks) to indicate additional sounds (or stress). On the other hand, synthesizing a text generated without diacritics generally leads to a poor intelligibility; sometimes, the syntactic or semantic ambiguities make certain sentences entirely incomprehensible.

The automatic restoration of diacritics in textual mediums where they are missing is a very difficult problem, as there are not evident linguistic rules to accomplish this task. Several studies performed for Spanish, French and Hungarian are presented in [13], [14] and [7].

Romanian language makes use of three diacritic marks, i.e. a breve, a circumflex accent, and a cedilla, leading to five letters with diacritics: **ă**, **â** / **î** (used as the same sound, but in different circumstances), **ş** and **ţ**. Many diacritics indicate only a different noun form (e.g., *casă* – house, and its pair *casa* – the house); in other situations their presence leads to a completely distinct meaning (e.g., *fata* – the girl, but *faţa* – the face). We must note that the percentage of words written with diacritics in a Romanian text is substantial: between 25% and 40% of the total number of words.

After studying large corpora containing texts from various domains, we established that the Romanian words could be classified into the following five categories (we will denote them by Type **1**, **2**, etc.):

1. – *Words without diacritics*

– Words with diacritics:

2. – *Words always written with diacritics*
(*câteva* – some; *ştiinţific* – scientific)

– Words with different meanings when written with or without diacritics:

3. – *Words with a single (possible) diacritic*
(*dou**ă*** – two, and its pair *(a) doua* – the second)

– Words with two or more diacritics:

4. – *Words where all diacritics but one are always present*
(*c**ă**maş**ă*** / *c**ă**maşa* – shirt / the shirt)

5. – *Words where any diacritic can be present or not* (multiple diacritic patterns)
(*pan**ă*** / *pana* / *p**â**n**ă*** – feather / the feather / until)

Particularly, the previous 4th category is interesting, because if one can accurately restore at least some of the diacritics that are present in all word patterns, the synthesized word will be, if not completely natural, anyway more intelligible than the pronunciation of the same word written without diacritics at all; in these situations, after partial diacritic restoration, the word can be included in the 3rd category. In addition, a number of function words of Type **3** can be included in the 2nd category, because their pairs written without diacritics have an extremely low probability of occurrence in Romanian common texts (e.g., ***şi*** / *si* – and / ti, the sevenths tone on the musical scale; ***în*** / *in* – in / flax, etc.).

As an example, one of the studied texts including a total number of 2,955 words contains 1,081 words (36.6%) written with diacritics. From this group, 543 words are of Type **2** and, taking into account the last remark, 196 function words can be also classified as being of Type **2**. The remainder of 342 words (which means 31.6% from the 1,081 words with diacritics) is of Type **3**, **4** or **5** (in fact, the most difficult situations to deal with).

The automatic diacritic restoration algorithm developed for our e-mail reader is based on a hybrid (dictionary and rule set) approach; it will be succinctly described in the following.

We started by building a large electronic dictionary of the most used Romanian words, D1. It contains in present more than 120,000 words and was derived by processing large text corpora, but also by (manually) adding all the inflectional forms of the content words (nouns, verbs, adjectives, and most adverbs) that can include letters with diacritics. Starting from D1, three dictionaries were iteratively inferred to cope with multiple patterns of the words written with diacritics and to facilitate the correct (subsequent) automatic processing. The last dictionary (denoted by D4) contains all the possible forms for each word, indexed twice according to the basic position in the dictionary of the word written without diacritics and respectively to the number of the possible diacritic patterns.

At run time, based on the D4 dictionary, the restoration algorithm (which is in fact a part of the preprocessing module in the TTS system) processes the incoming word string by searching first in the D4 dictionary (on the first positions). If the word is not found, it remains unchanged (it is a better solution than inserting wrong diacritics). If it is found in D4, it is further processed in accordance with the maximum value of its second index and classified as being of Type **1**, ..., **5**. For values 0 and 1, the word is either of Type **1** or **2** and it is simply copied from D4. If that value is greater than 1 and after processing the words from the 4th category as in the preceding discussion, the algorithm makes use of a number of rules based on the properties of the word context. A few of these rules are relatively simple (as for example the one that tracks the presence of the adjacent feminine article "o", which leads to replacing of the final "a" – if it is present in the current word – with "**ă**"), while the others assign the diacritics based on a statistical procedure. In these situations, the diacritics are restored by choosing the most likely of the word's patterns in D4, given the context (e.g., inserting a final "**ă**" for feminine nouns which are not preceded by the article "o", for some adjectives accorded with a feminine noun, or for several verbs at 3rd person singular form, etc.).

The flowchart of the proposed algorithm is presented in Figure 5. In this diagram, C[*n*][*i*] represents one of the particular forms of a word (written without or with diacritics): the first index *n* indicates its basic position in D4, and the second index *i* is related to the number of diacritic patterns.

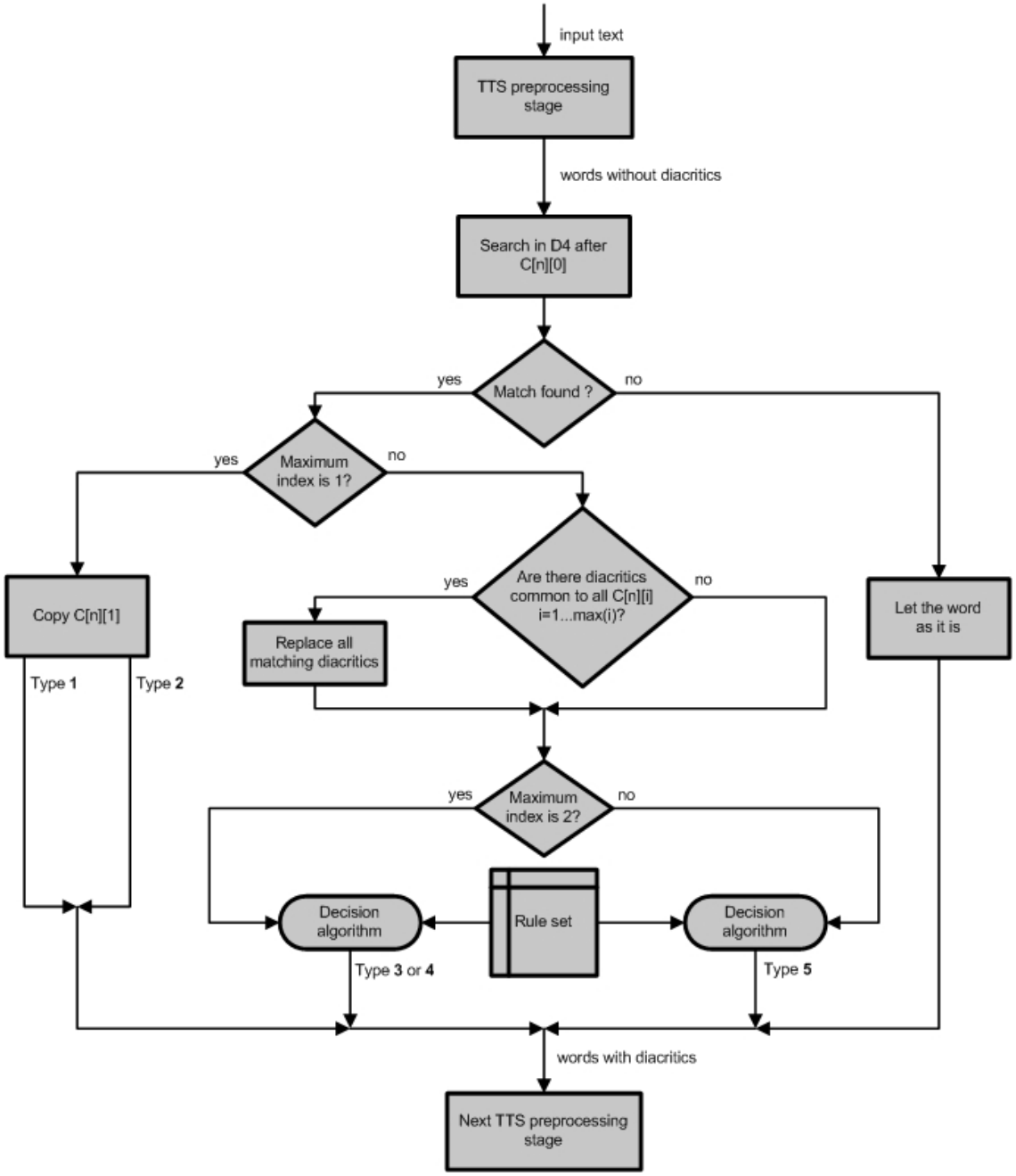

Figure 5. Flowchart of the proposed diacritic restoration algorithm.

We performed several tests in order to estimate the performance of the automatic diacritic restoration algorithm. We first removed all diacritics from three different texts containing a total number of 11,596 words. Then, the algorithm was used for diacritic restoration and the result was compared (letter by letter) to the original texts. The overall accuracy was about 94% (every diacritic missed or incorrectly assigned was considered as an error). We anticipate even better results by increasing the dictionary size (work that is currently in progress).

4. A TTS MEDIA SERVER ARCHITECTURE PROPOSAL

The main purpose of the demo platform was to verify the concepts in an end-to-end scenario with a minimum of resources. For a real production environment, we propose a media server architecture able to cope with carrier-grade requirements. Its most relevant features are discussed in the following.

- The most important characteristic of the TTS media server must be the multithreaded and multi-process design of the software. As a consequence, the hardware should allow for extensive parallel processing.
- TTS engine and media server should run on the same kind of operating system.
- The only supported interfaces should be IP based (as showed on Figure 3), because the PSTN interface or TDM (Time Division Multiplex) technology are not in the scope of a TTS media server.

The proposed architecture is depicted in Figure 6 and addresses the previous list of features.

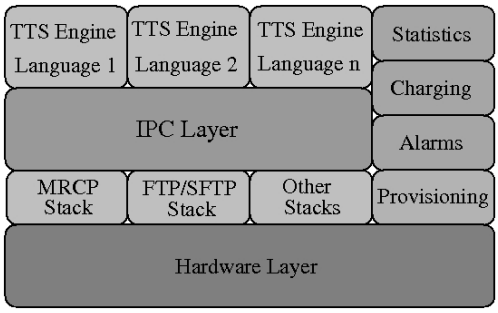

Figure 6. A media server architecture proposal.

The architecture is multi-tier in two respects. There is a layered view on the traffic side, where we have the TTS engine itself that communicates with the stack layers through an IPC (Inter-Process Communication) layer, and we have as well the "helping" layers consisting of Statistics, Charging, Alarms and Provisioning functions, essential for monitoring system performance and for remotely starting or stopping various services.

The TTS engine is by itself a proprietary piece of software, subject to a strict intellectual property management and is the reason to bring together the rest of the technologies necessary to implement such architecture. The system must support multiple languages, with several voices configurable for each language.

The IPC layer function is to decouple the TTS engine from the underlying communication stack details and to provide enough resources for the incoming traffic. Furthermore, IPC layer will make possible to add other communication stacks in a transparent way.

The communication stack layers will offer the right interfaces to the peer nodes, already existing in the network. It is important to have an easy way to implement another stack or to change an existing one in order to minimize the integration effort.

The hardware layer must be scalable while the number of calls is growing, and furthermore it should comply with the telecom requirements regarding redundancy and reliability.

Another aspect related to communication stacks is the requirement for RTP streams. Here we can split the use of the TTS media server in two areas: *intra-operator deployments* and *hosting installations*.

In the first case, the media server will run inside the operator's premises, making possible to fulfill strict RTP QoS (Quality of Service) requirements in the IP backbone. Therefore a stream-oriented approach is quite suitable from the media server perspective.

In the second case, the media server will run outside the scope of the operator's internal network. This makes QoS requirements rather difficult to be accomplished, not because of technical issues but because the RTP stream could cross several domains with different administrations without end-to-end agreement possibilities. In this case, FTP/SFTP (File Transfer Protocol/Secure File Transfer Protocol) could be used, but in this case the service provided to the end-user is not guaranteed to be real-time.

5. CONCLUSIONS

It is already a general consensus that TTS technology, as part of voice-based multimedia user experience, can add naturalness and efficiency to human-computer interaction. From this perspective, we have discussed the characteristics, the benefits and also the evolution of network-based speech synthesis solutions.

The paper tried to reveal, as orthogonal as possible, the needs for TTS from a value added perspective and three main directions where referred: notification/warning, legacy adaptations and accessibility. Another dimension, common to all the previous directions, mentioned two complementary ways to implement TTS-based services: embedded and network centric systems. The discussion included as well the possibility of a hybrid implementation, in order to overcame the drawbacks.

Focusing on the network centric approach, the paper compared the proprietary interface solution with the open protocol way of developing TTS engines. The benefits and disadvantages of each solution where briefly discussed.

Choosing a suite of open protocols to build the engine around is not an easy task and it should take into consideration the existing trends in networking evolution. The migration towards NGN makes more clear the possibility of considering TTS as a service enabler and therefore imagine new convergent services for the end user, leveraging all the enablers such as messaging, call control, presence, location, user status, and so on.

As one of the most spread communication services is e-mail, we developed an end-to-end e-mail reader application using a proprietary TTS system in Romanian, involving legacy telephony interface, POP3 standard communication and off-the-shelf hardware. The demo platform was used as well to test end-user subjective reaction to a specific test-to-speech conversion for Romanian language. Even if its main targets were to verify the network-based concepts and also to test our TTS system in a completely new environment, the platform is in present fully functional.

A particular difficulty in implementing a real TTS-based e-mail reader is the requirement for restoring missing diacritics to text, as a common problem for most languages that use the Latin alphabet. The paper proposed as well a diacritic restoration algorithm based both on a dictionary and on a set of rules. Its (current) overall accuracy of automatically inserting the proper diacritics is about 94%, which is a very good performance compared to the difficulty of the task in Romanian.

We need to mention that we faced (inherently) a lot of challenges building the e-mail reader demo platform, but these difficulties actually helped us to propose a more focused and telecom oriented architecture to implement the TTS engine in the form of a media server, suitable to a real production environment. Last but not least, we have split the deployment of TTS engines in two categories from QoS perspective: intra-operator and hosting installations.

REFERENCES

1. BURILEANU D., *Basic Research and Implementation Decisions for a Text-to-Speech Synthesis System in Romanian*. International Journal of Speech Technology, Kluwer Academic Publishers, Dordrecht, **Vol. 5**, No. 3, pp. 211-225, 2002.
2. BURILEANU D., FECIORU A., ION D., *On Automatic Speech Synthesis for Spoken Language Interfaces*. Speech Technology and Human-Computer Dialogue, Publishing House of the Romanian Academy, Bucharest, pp. 127-138, 2003.
3. BURILEANU D., *Spoken Language Interfaces for Embedded Applications*. Human Factors and Voice Interactive Systems (D. Gardner-Bonneau and H. Blanchard – Eds.), 2nd Edition, Springer Publishing House, Norwell, 2007 (to appear).
4. HANDLEY M., JACOBSON V., *SDP: Session Description Protocol*. RFC 2327, April 1998.
5. JIANG W., LENNOX J., NARAYANAN S., SCHULZRINNE H., SINGH K., WU X., *Integrating Internet Telephony Services*. IEEE Internet Computing, **Vol. 6,** No. 3, pp. 64-72, 2002.
6. MONAGHAN A., KASSAEI M., LUCKIN M., AMADOR-HERNANDEZ M., LOWRY A., FAULKNER D., SANNIER F., *Multilingual TTS for Computer Telephony: The Aculab Approach*. Proceedings of EUROSPEECH'2001, Aalborg, **Vol. 1**, pp. 513-516, 2001.
7. NÉMETH G., ZAINKÓ Cs., FEKETE L., OLASZY G., ENDRÉDI G., OLASZI P., KISS G., KISS P., *The Design, Implementation and Operation of a Hungarian E-mail Reader*. International Journal of Speech Technology, Kluwer Academic Publishers, Dordrecht, **Vol. 3**, No. 3/4, pp. 217-236, 2000.

8. ROSENBERG J., SCHULZRINNE H., CAMARILLO G., JOHNSTON A., PETERSON J., SPARKS R., HANDLEY M., SCHOOLER E., *SIP: Session Initiation Protocol*. RFC 3261, June 2002.
9. SCHULZRINNE H., CASNER S., FREDERICK R., JACOBSON V., *RTP: A Transport Protocol for Real-Time Applications*. STD 64, RFC 3550, July 2003.
10. SHANMUGHAM S., BURNETT D., *Media Resource Control Protocol Version 2 (MRCPv2)*. Draft-ietf-speechsc-mrcpv2-12, March 2007.
11. SINGH K., WU X., LENNOX J., SCHULZRINNE H., *Comprehensive Multi-platform Collaboration*. Columbia University Technical Report CUCS-027-03, New York, Nov. 2003.
12. TALUSANI R.K., *Universal Voice Access Platform*. Proceedings of the Applied Voice Input/Output Society (AVIOS) 2003, San Jose, California, 2003.
13. YAROWSKY D., *Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French*. Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, Las Cruces, NM, pp. 88-95, 1994.
14. YAROWSKY D., *Corpus-based Techniques for Restoring Accents in Spanish and French Text*. Natural Language Processing Using Very Large Corpora, Kluwer Academic Publishers, Dordrecht, pp. 99-120, 1999.