Text to Speech (TTS) in mobile phone application

Milun Jovanović, milunj@yahoo.com
SVOX AG, Baslerstrasse 30, Zurich 8047, Switzerland

ABSTRACT
The main purpose of this paper is to present one possible implementation of TTS in a mobile phone application. Although, requirements for TTS in a mobile phone application are presented and one feasible solution for TTS for embedded, low MIPS and low footprint memory environment on which author of this paper worked in a period of 3 months.

INTRODUCTION IN THE COMPANY
Company SVOX AG is a software technology company that specialized in state-of-art concatenative Text-to-Speech (TTS) software solution. SVOX TTS was developed in over 15 years of research at the Swiss Federal Institute of Technology in Zurich (ETH Zurich).

LINGUISTIC TERMINOLOGY
Morphology: Describes how words can be built from minimal meaningful elements of a language, the so-called morphemes
Lexicon: Collection of words and morphemes which belong to particular language
Phonetics: The investigation of the actual production and perception of spoken utterances
Prosody: The investigation of the manner of articulation of a speech sound sequence in terms of intonation (or melody), rhythm, and loudness, which are phonetically manifested in the acoustic properties pitch (fundamental frequency-F0), speech sound duration and speech signal intensity
Grapheme: The grapheme is the smallest unit of the written language which distinguishes different written words
Phoneme: The phoneme is the smallest unit of the spoken language that distinguishes different meanings of utterance.
Morpheme: The morpheme is the smallest unit of language (in general smaller than a word) carrying a well-defined meaning.
Syllable: Unit of the spoken language which is composed of voiced center (the syllabic nucleus: a vowel, a diphthong, or a sonorant consonant) with an intensity

TEXT TO SPEECH MODEL
A TTS system maps a linear sequence of elements of grapheme medium onto a linear sequence of elements in an acoustic medium. Mapping comprises two parts: the analysis of the grapheme string, during which an underlying representation of the original symbol sequence is constructed, and the subsequent use of this representation for the synthesis of the target sound sequence. The TTS process can therefore be viewed as a text analysis stage followed by a signal synthesis stage, as depicted in Figure 1

![Figure 1. TTS process viewed as an analysis and a subsequent synthesis stage](image1)

This model doesn’t imply that a text or sentence must be fully analyzed before proceeding with the synthesis. The two processes may well be intermingled, as in the case of human being reading aloud a certain text. Input text is processes sentence-wise (sentence by sentence).

The underlying structure, which is the result of the text analysis, usually some more or less detailed representation of the syntactic sentence structure, but it might well contain some further (e.g., semantic) information. It is a quite clear that the input text submitted to TTS system does not contain any characterization of a certain (human or synthetic) speaker. On the other hand, the resulting speech signal will clearly identify a particular speaker speaking in a specific style. Thus, somewhere in the TTS synthesis process, there must be a transition from speaker-independent to speaker dependent (or speaker characterizing) information. Theoretically, this transition could be spread out over the entire TTS process. However, in the linguistic view a speaker independent part is strictly separated from a speaker-dependent part. In this model, which is shown on Figure 2 the terms transcription and phon-acoustical model have been adopted for two basic TTS stages. The transcription maps (or transcribes) the grapheme text onto an abstract intermediate representation of the utterance to be synthesized, and the phon-acoustical model implements the phonetic-acoustic realization of the intermediate realization.

![Figure 2. TTS process viewed as consisting of speaker-independent and a speaker-dependent part.](image2)

STRUCTURE OF TTS SYSTEM
The input text is treated sentence by sentence. Each word in the sentence is looked up in a full-form lexicon or is morphologically decomposed into morphemes (according to a word grammar) which are looked up in a morpheme lexicon, and the syntactic structure of the whole sentence is subsequently analyzed according to a sentence grammar. The result of this analysis is an annotated syntax tree which contains the structure of the whole sentence and the structure of each word. Moreover, for each constituent of the syntax tree, the constituent type and further attributes (like, e.g., case, number, and gender) are marked, and the grapheme and phonemic representation of words and morphemes are contained in the tree.
From the syntax tree, accentuation and prosodic phrasing are determined, which together with the phonetic transcription of each word, constitute the phonological representation of the sentence. This representation is the input to the subsequent generation of duration values for all speech segments and of the fundamental frequency contour.

The speech signal is generated by concatenation of diaphone elements (speech sound transition elements extracted from natural speech). The diaphones are selected according to the phonetic transcription of the utterance, and upon concatenation they are modified such that they match the specified duration and Fo values.

**Figure 3.** Overall structure of TTS system

**GENERATION OF SPEECH SOUND**

In the SVOX TTS is implemented concatenative speech synthesis. That synthesis combine pieces of natural, human speech into new utterance, i.e. the generated speech signal is not entirely synthetic. In order to be able to realize arbitrary utterances, the inventory of basic speech elements must be chosen appropriately. The currently most popular concatenative synthesis is the diphone synthesis, in which the basic elements are diphones, i.e. speech parts ranging from the middle of one speech sound to the middle of the next. The concatenative approach currently produces the highest quality of synthetic speech, and a concatenative synthesis system can be built with a relatively small amount of work. However, the approach is a rather unflexible. Each new voice or each new timbre requires establishing of a new speech element inventory.

One of the major problem of concatenative synthesis is the modification of duration and pitch of the speech elements without changing the spectrum of signal. These modifications are necessary to generate the prosody of the synthetic speech. Three standard solutions are found: changing in Fourier domain, LPC and TD-PSOLA.

**DERIVATION OF PROSODY**

The physical prosodic parameters by which speech segments can be modified are fundamental frequency (Fo) or pitch, segment duration, and signal intensity. These measurable quantities are the physical correlates of the much more abstract linguistic notions of sentence melody, speech rhythm, and loudness. It is generally acknowledged that fundamental frequency and segmental duration are more important for the naturalness of synthetic speech than intensity. Consequently, prosody research has concentrated much more on the first two parameters, and most current TTS system do not include any explicit intensity control. However, if pitch and duration control were nearly perfect in a TTS system, a very good intensity control would become necessary as well, in order for the system to achieve complete naturalness. For the time being, however, much more remains to be done in terms of the quality of melody and rhythm of synthetic speech. One phenomenon so-called declination of Fo is immanent: within each utterance there is a general tendency for Fo to decrease from the beginning towards the end.

**REQUIREMENTS IN A MOBILE PHONE APPLICATION**

Two threads of Layer 3 GSM mobile application are the main users of TTS in a mobile phone. Call control thread (as a part of Connection management) and SMS server thread (Short Message Control layer as sub layer of Connection Management). It is easy to understand that we can replace ringing tone of incoming call with message “John, Martin is calling you” where Martin is a variable from Phone book. Another solution could be SMS reading or e-book reading. Current limitation for SMS is 168 characters but for 3G it can be long e-book.

These two threads can activate TTS while mobile phone is in Idle mode (GSM mobile phone) or in STANDBY mode (GPRS mobile phone).

Nowadays 90% of mobile phones content ARM risc processor as a main processor. ARM7TDMI was target for author’s research and implementation. That processor has system clock on 10Mhz in a mobile phone and risc processors perform usually one instruction in one bus cycle. In a Idle mode of mobile phone 80% MIPS were available to author to implement TTS application. That means that author has available 8.000.000 bus or core cycles in a period of one second and to provide 16 bits PCM stream with sampling rate of 8khz (8000*16 bit/sec).

Another but not to much critical issue was memory limitation. This limitation can be parsed on the two requirements: size of the run-time memory (stack) and size of the overall memory. Author of this work got requirements of 10 Kbytes run-time memory and 250 Kbytes of RAM (flash) memory. In a mobile phone mostly is implemented old solution which provision application from flash to RAM. Nowadays we can find with some mobile phone manufactures that application is executing directly from flash. From the reason of unambiguous of the software, author has to exclude static data from TTS application.

**INTEGRATED ENVIRONMENT AND BENCHMARKING SOLUTIONS**

Author was a part of group of 3 engineers (2 PhD) and my responsibility was to develop and test embedded real time solution from current linguistic and speech generation applications. Integrated development was ADS (Arm Developer Suite) 1.2 and for debugging purpose author has to measure real-time performance (monitoring Statistics and Profiling parameters). Statistics obtain number of bus and core instructions which are executed between two breakpoints (if they are set) or overall statistics from beginning application. This value was most useful in development and in benchmarking software performance.

Arm has developed two different set of cores based on two memory access architectures: Von Neumann and Harvard. Von Neumann cores (e.g. ARM7TDMI) use a single bus for both data and instruction accesses so the cycle types refer both types of memory access. Harvard cores have a separate data and instruction fetches.
Interpreting statistics: linguistic part of TTS consumes 38412 bus cycles for sentence “Milun Jovanović is a good boy” and we can calculate total execution time:

\[
38412 \times 1/10,000,000 = 3.8\text{msec}
\]

Arm Profiler can display a call graph profile that shows approximations of the time spent in each function, the time accounted for by calls to all children of each function, and the time allocated to calls from different parents.

For the stack calculation ADS doesn’t provide good solution and only useful information’s could be provided defining stack size in a process of compiling and if appear warning author will be informed that stack is overflow.

### NEW SOFTWARE REQUIREMENTS AND REAL-TIME ISSUES

As mentioned before TTS application is divided into two parts: language-dependent linguistic part of software (fast part) and language independent speech generation part of software (slow part of application). Current solution implies using OS dependent API functions and generating .wav file. After generating .wav file, next step was payout .wav file. That solution was possible to implement on Simian OS system with perceived delay. For author it was partly acceptable for current SMS of 168 characters. But for longer text, speech quality decline (delay between sentences is increased) and .wav file could become unacceptable huge.

One issue dominated in research: **TTS processing is a very, very, slow application**

New requirements for embedded application appeared:

1. It has to be done embedded solution without .wav file processing – direct writing into the buffer.
2. Interface to sound was responsibility to application whom use TTS (no more knowledge of sound OS functions)
3. Limited size of output buffer in a RAM memory (user defined)
4. Real time performance that linguistic and speech part can execute sequentially (together faster than time)
5. Granularity of software that application can go out of TTS and return later on a same place

Author introduced new part of software: **dispatcher layer.** Dispatcher layer provide API interface, and granularity of application.

Author has more years experience in developing real-time embedded software and first idea was: “Creating one thread (task) which will be message dependant and message will be generated in API functions with clients (Call control and SMS) priority”. Thread (task) will perform TTS per channel bases (every request new channel) That solution is a very common in embedded applications but my colleagues were unsatisfied in the reason that my solution introduces OS dependant’s parts (task, partition, buffers). Author investigated complete developed TTS application and made conclusion: “No difference between TTS and sin or cos function”. Only one difference: sin and cos are fast but TTS is a slow application. That is a reason of introducing granularity of application. Next paragraph explain in detail.

All thread in a mobile phone is message dependant (interruptible). If Call control thread start TTS processing (incoming call) and user reject call using MMI, this message will be in input queue of Call control thread but CC thread will still play TTS. To resolve this issue author has to provide granularity of software.

Dispatcher has to provide multi results output. Dispatcher layer has to keep track of current analyze of text and to keep track of last playout phone. Dispatcher layer has to keep track of expired time and to define which part of application will be executed in a certain time and to provide relevant input and to obtain valid multi results.

**RESEARCH ISSUE, DISCUSSION AND CONCLUSION**

English language consists from 50 phones and theoretically with 50 * 50 = 25000 diphones. (In a modern English language we can find about 1900 diphones.) For TTS application list of phones and diphones represent English language. In the current SVOX application, language is compound from statically Huffman coded Fourier transforms of all phones and diphones. Fourier transform is selected in a reason that allows independent changing pitch and frequency and faster implementation concatenation phones and diphones in a frequency domain. The size image (language) is about 250kbytes and for speech syntheses first step is finding appropriate phone or diphone from Huffman tree, extract, concatenate and finally implement IDFT.

If we write word “Milun”, Name phonetizer (text analyze) part of application will generate list of: phonemes, phoneme duration and Fo(arbitrary, allowed to client to define frequency-colour of voice) e.g.:

\[(m \ 80 \ 135), (a \ 250 \ 133), (l \ 130 \ 131), (v \ 128 \ 129), (n \ 80 \ 127)\]

First parameter is a phone, second is duration and third is frequency.

The main problem real-time in development was IDFT. E.g. for phone air author has to do \((250 \times 8)^2\) multiplication, \((250 \times 8)^2 \times 8\) add instructions and \((250 \times 8)^2\) dividing which together require 8,008,000 processor cycles. ARM7TDMI will do required job in 8000ms that is a much longer of required 250 msec.

To resolve this main problem author had to implement FFT instead of IDFT. Author had to cut every phone and diphone in time domain in array of 64 samples and implement FFT. That solution increased number of elements whom compound phones and that solution excluded concatenation algorithms in frequency domain. It was very fast returning from frequency domain to time domain but new application required TD-PSOLA to concatenate phones and diphones in a time domain. New application was more than 10 times faster than old application.

Last dilemma was elect right FFT algorithm. After detail reading papers from prestige world universities [2] fastest algorithm for FFT for 64 samples is RADIX -4.

**REFERENCE**

[1] The implementation of a Text-to-Speech System for German Dissertation for the degree of Doctor of Technical Science CHRISTOF TRABER

[2] Comparative analysis of FFT algorithms in sequential
and parallel form Michael Balducci, Mississippi State University