NTNU – Trondheim
Norwegian University of
Science and Technology

TaleTUC: Text-to-Speech and Other Enhancements to Existing Bus Route Information Systems

Trond Bøe Engell

Master of Science in Computer Science
Submission date: June 2012
Supervisor: Bjørn Gamback, IDI
Co-supervisor: Rune Sætre, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

As smartphone sales increase, the demand for content for these devices also increases. Service providers that want to reach out to as many users as possible need to create smartphone applications that satisfy people that do not fall into the "normal user" category. People that require non-visual feedback, such as visually impaired persons, need output in form of auditory signals. Text-to-speech synthesis provides this functionality, giving the smartphone the ability to convey messages in the form of speech.

This thesis describes *TaleTUC: Text-to-speech*, a proof of concept text-to-speech system for the domain of bus route information. The system uses a client-server architecture where the server converts text to computer-generated speech signals and provides playable audio files either directly to the smartphone, or through a Java Servlet that provides functionality to tailor the output (e.g., audio compression). Descriptions of other enhancements to bus route information systems, that are are not directly related to synthesized speech, have also been given.

Three text-to-speech modules have been evaluated and to establish whether there is a link between intelligibility and naturalness in synthesized speech. Non-functional tests (transfer, response time, etc) have also been conducted to get an impression of whether service providers that use "cloud" technology provide a better service than an in-house system. There are no definitive answers to these questions, but results indicate that there might be a link (however small) between intelligibility and naturalness and that an in-house system is still preferable in the domain of bus route information.

Sammendrag

Samtidig som smarttelefonsalget øker, øker også følgelig behovet for innhold til disse enhetene. Tjenesteleverandører som ønsker å nå ut til så mange brukere som mulig trenger å lage applikasjoner for smarttelefoner som tilfredsstiller personer som ikke faller inn under "normal bruker" kategorien. Folk som krever ikke-visuelle tilbakemeldinger, som for eksempel svaksynte, må få respons i form av lydsignaler. Talesyntese tilbyr denne funksjonaliteten, noe som gir smarttelefonen evnen til å formidle meldinger i form av tale.

Denne avhandlingen beskriver *TaleTUC: Tekst-til-tale*, et prototyp talesyntesesystem for bussrutedomenet. Systemet bruker en klient-server-arkitektur der serveren konverterer teksten til datagenererte språksignaler og gir spillbare lydfiler enten direkte til smarttelefonen, eller gjennom en Java Servlet som gir funksjonalitet for å skreddersy lydfilene ytterligere (f.eks lydkomprimering). Det er også gitt beskrivelser av andre forbedringer til bussrutesystemer, som ikke er direkte relatert til syntetisk tale.

Tre tekst-til-tale moduler har blitt evaluert for å fastslå om det er en sammenheng mellom forståelighet og naturlighet i syntetisk tale. Ikke-funksjonelle tester (dataoverføring, responstid, etc) har også blitt gjennomført for å få et inntrykk av om tjenesteytere som bruker "nettsky"-teknologi gir en bedre tjeneste enn et internt system. Det er ingen definitive svar på disse spørsmålene, men resultatene indikerer at det kan være en kobling (om enn liten) mellom forståelighet og naturlighet, og at en internt system fortsatt er å foretrekke i bussrutedomenet.

Preface

This Master's thesis describes the study and work done by Trond Engell. The work is the last final part of his Master of Science (MSc) degree in Computer Science at the Department of Computer and Information Science at the Norwegian University of Science and Technology (NTNU). The assignment is a contribution to the *Fremtidens ultimate intelligente ruteopplysningssystem (FUIROS)* project, where other assignments have been conducted earlier. This includes the two pre-studies *TABuss: An Intelligent Smartphone Application* and *MultiBRIS: A Multiple-platform approach to the Ultimate Bus Route Information System for Mobile Devices*, but also the thesis *An Intelligent Smartphone Application* [21, 2, 29]. The results of the pre-studies have been rewritten in the form of papers and [22, 3]. They were be presented at the *12th International Conference on Innovative Internet Community Systems*[1] on the 13th of June, 2012.

Parts of the Future Work section of this thesis are identical to the ones found in a thesis conducted in parallel to this one, by Marcussen and Andersstuen [20]. Section 6.2 in Future Work is also found in their FUIROS assignment, but has been slightly adjusted to fit the text of this thesis. Subsections 6.2.1 and 6.2.3 were written by Engell and Anderstuen and modified by Marcussen [2]. Subsections 6.2.2 and 6.2.4 were originally written by Marcussen and Eliassen [21]. Those have also been modified by Marcussen and were approved by Eliassen.

[1] http://www.ntnu.edu/i2cs/

Acknowledgments

I would like to thank my supervisors Rune Sætre and Björn Gambäck, along with Rune M. Andersen for their effort in leading me in the right direction. I would also like to thank my lab partners Runar Andersstuen, Lars Moland Eliassen and Christoffer Jun Marcussen for our collaboration in the FUIROS project, and for technical and moral support . Also, I would like to thank Geir Josten Lien for his advice and all the kind people who volunteered to be my test subjects. Finally, I would like to thank my parents for all their encouragement during this process.

Trond Bøe Engell
Trondheim, June 17, 2012

Contents

List of Figures

List of Tables

Terminology and Abbreviations

This section describes the terminology and abbreviations used in this thesis. Some of the explanations below are partially taken from sources such as product web-sites, standardization organization's web-sites and Wikipedia[2].

- **A-law** - A standard companding algorithm, used in European digital communications systems to optimize, i.e., modify, the dynamic range of an analog signal for digitizing.
- **API** - An Application Programming Interface is a source code based specification intended to be used as an interface by software components to communicate with each other.
- **AtB** - Administration agency for public transport in Sør-Trøndelag. Formerly Team Trafikk.
- **ASR** - Automatic Speech Recognition is the translation of spoken words into text.
- **Black Box** - An object which can be viewed solely in terms of its input, output and transfer characteristics without any knowledge of its internal workings.
- **Business Logic** - A non-technical term generally used to describe the functional algorithms that handle information exchange between a database and a user interface. In this thesis, it is specifically used to describe the part of the system that makes the actual computations and calls external services.
- **BusTUC** - A natural language problem solver capable of answering questions about bus departures in Trondheim stated in common English or Norwegian language.

[2] `http://en.wikipedia.org/wiki/Main_Page`

- **CAPI** - The Common ISDN Application Programming Interface is a standardized software interface (ISDN conform).
- **CPU** - Central Processing Unit is the hardware within a computer system which carries out the instructions of a computer program by performing the basic arithmetical, logical, and input/output operations of the system.
- **CSS** - Cascading Style Sheets is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language.
- **(D)DOS** - Denial-Of-Service attack (DoS attack) or Distributed Denial-Of-Service attack (DDoS attack) is a malicious attempt to make a machine or network resource unavailable to its intended users.
- **Diphone** - A diphone is an adjacent pair of phones. It is usually used to refer to a recording of the transition between two phones.
- **DOM** - The Document Object Model is a cross-platform and language-independent convention for representing and interacting with objects in HTML, XHTML and XML documents.
- **FUIROS** - Fremtidens Ultimate Intelligente RuteOpplysningsSystem. In English: The Ultimate Bus Route Information System For The Future.
- **GUI** - A Graphical User Interface is a type of interface that allows users to interact with electronic devices using images rather than text commands.
- **HSDPA** - High-Speed Downlink Packet Access is an enhanced 3G (third generation) mobile telephony communications protocol.
- **HTML5** - A HyperText Markup Language for structuring and presenting content for the World Wide Web, and a core technology of the Internet.
- **IaaS** - Infrastructure as a Service is basic cloud service model, providing computers as physical or more often as virtual machines.
- **IDE** - An Integrated Development Environment is a software application that provides comprehensive facilities to computer programmers for software development.
- **IDI** - Department of Computer and Information Science (Institutt for Datateknikk og Informasjonsvitenskap), NTNU.

- **ISDN** - Integrated Services Digital Network is a set of communications standards for simultaneous digital transmission of voice, video, data, and other network services over the traditional circuits of the public switched telephone network.
- **ISP** - Internet Service Provider is an organization that provides access to the Internet.
- **JSON** - JavaScript Object Notation is a lightweight data-interchange format.
- **L&H** - Lernout & Hauspie was a leading Belgium-based speech and language processing technology company.
- **MultiBRIS** - Multiple-platform approach to the Bus Route Information System is a system developed in parallel with TABuss.
- **NLP** - Natural Language Processing is a field of computer science, artificial intelligence and linguistics concerned with the interactions between computers and human (natural) languages.
- **NTNU** - Norwegian University of Science and Technology (Norges Teknisk-Naturvitenskapelige Universitet).
- **PaaS** - Platform as a Service is a category of cloud computing services that provide a computing platform and a solution stack as a service.
- **PESQ** - Perceptual Evaluation of Speech Quality, is a family of standards comprising a test methodology for automated assessment of the speech quality as experienced by a user of a telephony system.
- **Phone** - A speech sound or gesture considered a physical event without regard to its place in the phonology of a language.
- **RIFF** - The Resource Interchange File Format is a generic file container format for storing data in tagged chunks. It is primarily used to store multimedia such as sound and video, though it may be used to store any arbitrary data.
- **SaaS** - Software as a Service is a software delivery model in which software and associated data are centrally hosted on the cloud.
- **SAPI** - Speech Application Programming Interface is an API developed by Microsoft to allow the use of speech recognition and speech synthesis within Windows applications.

- **SDK** - A Software Development Kit is typically a set of software development tools that allows for the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar platform.
- **SCRUM** - An iterative and incremental agile method for software development that manage software projects.
- **TABuss** - Tore Amble Buss is an intelligent Android bus route application.
- **TTS** - Text-to-Speech is the artificial production of human speech.
- **TUC** - The Understanding Computer is a reasoning system developed at IDI by Tore Amble.
- **WAV** - A Microsoft and IBM audio file format standard for storing an audio bit stream on PCs.
- **W3C** - The World Wide Web Consortium is the main international standards organization for the World Wide Web.
- **WHATWG** - The Web Hypertext Application Technology Working Group is a community of people interested in evolving HTML and related technologies.
- **WiFi** - Wireless local area network is a technology that allows an electronic device to exchange data wirelessly (using radio waves) over a computer network, including high-speed Internet connections.
- **XHTML** - eXtensible HyperText Markup Language is a family of XML markup languages that mirror or extend versions of the widely-used Hypertext Markup Language (HTML).

Chapter 1

Introduction and Goals

In the following sections, the task description and motivation for this thesis are presented. The research questions and goals are also defined. Finally, a research method definition and an overview of the thesis structure is provided. The focus of the thesis has been on creating a text-to-speech extension to TABuss, giving the application the ability to speak in Norwegian [21]. Tweaks and additions that enhance the system itself, have also been explored.

1.1 Task Description

The (original) assignment was given by supervisors (Tore Amble,) Björn Gambäck and co-supervisor Rune Sætre, as a part of the FUIROS project:

> *FUIROS - Fremtidens ultimate intelligente ruteopplysningssystem.*
>
> *BusTUC is a natural language bus route system for Trondheim. It gives information about scheduled bus route passings, but has no information about the real passing times. This is about to change, because AtB has installed GPS tracking of the buses, giving access to real passing times and delays. Besides, with new smart phones arriving rapidly on the market, there are possibilities for GPS localisation and connections to maps. The project shall take a broad view, and consider all possible advanced concepts, resulting in advanced smart phone applications.*

The concrete task in this thesis is to look at possibilities for text-to-speech (TTS) synthesis in Norwegian for use on smartphones in the domain of bus route information systems. It is desirable to make use of available existing solutions (like BUSTER) and to find out what needs to be implemented in order for the system to work optimally for the domain [15]. The end goal is to implement

a text-to-speech synthesis module in order to add multi-modal functionality to smartphone applications (proof of concept). Two applications created for FUIROS last year (TABuss on the Android platform and MultiBRIS on multiple platforms) are good candidates for implementing such a proof of concept module [21, 2, 22, 3].

Another stakeholder in this project is AtB[1], the main bus transportation company in Trondheim, and the bus route information development community represented by Rune M. Andersen. If a text-to-speech standard for bus route information can be established, future development could contribute to the improvement of this standard, which would make it easier to expand beyond Trondheim's borders and to other cities.

1.2 Motivation

This work is a natural extension of the previous work by Andersstuen and Engell, and Marcussen and Eliassen [2, 21]. These previous FUIROS assignments were carried out as pre-studies prior to this thesis and consisted of exploring the possibilities for running bus route information systems with natural language (like BusTUC) on smartphone devices. The paper written by Andersstuen and Engell describes the increase in sales and popularity of smartphones and the importance of adapting software to mobile devices [2]. The worldwide smartphone market has expanded immensely during the last few years. There were 440 million mobile devices sold by vendors in the 3rd quarter of 2011[2]. Of these, 115 million were smartphones. This equals a market share of 26.1% of mobile phones.

This is a motivation for continuous implementation of new functionality targeting these devices. The Future Work section in the same paper describes the potential of speech functionality. Additional support for input and output in the form of speech (Multimodal Interaction[3]) could be added to smartphone applications. This would expand the target audience further to include the visually impaired, elderly and non-natives. The spoken dialog system "Let's Go" (2003) has implemented functionality to make bus route information available for these user groups [32]. For the non-natives, using such a system can even aid in learning the local language [31].

Marcussen and Eliassen developed an application for the Android[4] platform, called *TABuss*, and performed user tests where user feedback indicated

[1] https://www.atb.no/

[2] Deduced from the numbers given in Gartner's press releases: `http://www.gartner.com`

[3] `http://en.wikipedia.org/wiki/Multimodal_interaction`

[4] `http://www.android.com/`

that it has potential to be a popular choice for bus travelers [21]. Therefore, this application will serve as a potential target application for the prototype modules created in this master thesis.

The starting point for the research is an existing speech recognition and speech synthesis system called BUSTER [14, 11]. BUSTER is a spoken dialog system which interacts with BusTUC[5], and provides route suggestions through a dial-up telephone interface. BUSTER is further described in section 3.1.

The combination of the motivation from the BUSTER system with the promising results from the projects of Marcussen and Eliassen, and Andersstuen and Engell, creates a solid motivational foundation to continue the FUIROS project.

1.3 Research Questions and Goals

This section describes the research questions and goals of this master thesis.

Research question 1 (RQ1) What is the most optimal system architecture for a text-to-speech module designed for a smartphone?
a) Is it preferable to put the entire system directly on the smartphone, or will a client-server system be more practical?
b) Can cloud-based solutions potentially be used?

Research question 2 (RQ2) What enhancements are likely to make a bus route information system more efficient?

Research question 3 (RQ3) Does intelligibility and naturalness in speech coincide? If computer-generated speech is highly natural, does it also mean it is intelligible?

Goal 1 Create a text-to-speech module prototype. (RQ1)
The prototype module should be able to receive text and output synthesized speech in Norwegian. The prototype module should be implemented in such a way that it can be easily integrated into other applications, such as the TABuss application [21].

Goal 2 Create an intermediate Java Servlet prototype. (RQ1)
The prototype module will be developed for testing purposes and should be able to do the same as the prototype described in goal 1. It should also provide functionality that enlightens the advantages of adding an intermediate Java Servlet between the server and client in the text-to-speech solution.

[5] `http://busstuc.idi.ntnu.no/`

Goal 3 Create a new web page and implement JSON output for BusTUC. (RQ2)
A new web page should be implemented for BusTUC to meet today's standards. The web page should present the BusTUC Oracle answers both as text to humans and as JSON output for further computer processing.

Goal 4 Conduct a test to see if intelligibility and naturalness are features that go hand in hand in text-to-speech. (RQ3)
The test should use subjects in different scenarios to measure the intelligibility and naturalness of different text-to-speech modules.

1.4 Research Method

This section provides a rough outline of how the research was conducted for this thesis.

Trello[6], which is similar to a SCRUM[7] board, along with the FUIROS wikipedia page[8] was used for planning and assigning tasks from the FUIROS project to be integrated in the thesis.

The work of this thesis was split into four phases:

1. Problem definition and initial research
2. Prototyping
3. Evaluation
4. Documentation

The *research* phase was conducted by reading and analyzing relevant publications from these electronic libraries: DAIM[9], Google Scholar[10], Cite Seer X[11] and IEEE Xplore Digital Library[12].

The *prototype implementation* phase consisted of:

- Development of a TTS module for use within Android applications

[6] http://www.trello.com
[7] http://en.wikipedia.org/wiki/Scrum_(development)
[8] https://www.ntnu.no/wiki/display/FUIROS/
[9] http://daim.idi.ntnu.no/
[10] http://scholar.google.com
[11] http://citeseerx.ist.psu.edu
[12] http://ieeexplore.ieee.org/Xplore/guesthome.jsp

- Development of a Java Servlet extension to the TTS module
- Development of a test application for use in the evaluation phase
- Development of a new web page for BusTUC
- Implementation of JSON output in BusTUC
- Resolve UTF-8 issues in BusTUC

The *evaluation* phase was performed by the use of the test application and evaluation forms in collaboration with test subjects.

The *documentation* phase consisted of documenting the work done in the thesis after the evaluation phase was complete. This includes discussing the results, writing a conclusion and describing future work.

1.5 Thesis Structure

This section gives an overview over the master thesis structure with a summarized description of each chapter.

1. **Introduction and Goals** introduces the thesis with a description of task definition, motivation, research questions, defined goals, research method and the thesis structure.
2. **Theory and Background** presents the TTS and other background theory for this thesis.
3. **Methods** describes the development of the prototypes and the practical work done in this thesis.
4. **Results** presents the development results, with graphs, descriptions and screenshots.
5. **Discussion and Conclusion** discusses the results and reflects on how the goals and research questions were answered. Finally, some conclusion are drawn.
6. **Future Work** describes future work of this thesis and the FUIROS project.

Chapter 2

Theory and Background

2.1 Text-To-Speech (TTS) Technology

Speech synthesis, or text-to-speech (TTS), is the artificial production of human-like speech. Computer systems used for this purpose are called *speech synthesizers*. TTS systems synthesize text strings and files into spoken audio with synthetic voices by using a complex system of linguistic rules and dictionaries.

Different implementations of TTS systems exist. This section discusses some of the concepts on which these systems are built. This thesis is not concerned with how they work in detail, but the following text will give a concise overview of the current techniques used today.

In general, a TTS system can be broken down into three main parts: a linguistic, a phonetic and an acoustic part. First, an ordinary text is input to the system. A linguistic module converts this text into a phonetic representation. From this representation, the phonetic processing module calculates the speech parameters. Finally, an acoustic module uses these parameters to generate a synthetic speech signal.

The following comparison show that the technologies can be split in two in sense of how they work: The natural-sounding but inflexible "playback" systems (concatenative synthesis) , and the parameterizable systems requiring explicit acoustic models which are difficult to formulate (formant synthesis). New approaches try to overcome this separation by adding more control to synthesis methods in different ways [35].

2.1.1 Formant Synthesis

Formant synthesis does not use human speech samples, but rather a set of rules to predict the acoustic realization of speech. Even though these rules are designed carefully by experts, they tend not to capture the complexity of human speech, and the resulting synthetic speech sounds unnatural and "robotlike". However, Formant synthesis is flexible because it allows for control over a wide range of parameters.

2.1.2 Concatenative Synthesis

Concatenative synthesis "strings" together segments of recorded speech. This method produces the most natural-sounding speech. There are three main types of concatenative synthesis:

Diphone Synthesis

Diphone synthesis works by sequencing small pieces of human speech recordings. A speaker records one example of each *diphone* at a monotone pitch. A diphone is the piece of a speech signal that goes from the middle of one phone to the middle of the next phone. This gives a resulting quality that is usually quite a bit better than formant synthesis. The disadvantage, however, is that the voice quality is fixed, determined by the performance of the speaker during diphone recordings.

Unit Selection Synthesis

Unit selection synthesis works like Diphone synthesis, but instead of just one recording of each diphone at monotone pitch, several versions of the diphone are recorded in natural speech. For any target sentence, the most suitable diphone units are stringed together by using a sophisticated selection method, often a weighted decision tree. If suitable units are available, no or very little signal processing is needed. Accordingly, the resulting synthesized speech can sound very natural, to the point where it is indistinguishable from human speech.

Domain-specific Synthesis

Domain-specific synthesis strings together prerecorded words and phrases to create complete sentences. This method is used in applications where the system text output is limited to a particular domain. Examples of this are broad-

casting messages and weather reports [18]. The level of naturalness of systems using this method can be very high due the low variation of sentence types.

2.2 Multi-Modal Output

Most smartphones have big screens in order to utilize touch functionality[1]. The smartphone applications are inherently visual, giving information and feedback through means of a graphical user interface (GUI) and text. However, for users (such as the visually impaired) set in situations where visual feedback is inadequate or even impossible, audible feedback may be an essential feature. For other users it may just add extra value to a product. Still, implementing feedback in the form of speech to an application is no trivial matter and comes with many challenges. One of the major ones is inflexibility in the sense of digital recordings of human speech. This is where TTS comes in handy, giving applications the ability to give *multi-modal* output without many of the disadvantages of digital recordings. Text-to-speech provides a very valuable and flexible alternative for digital audio recordings in cases where:

- Human speech recordings are too expensive.
- Disk storage is insufficient to store recordings.
- The application does not know ahead of time what it will need to speak.
- The information varies too much to record and store all the alternatives.

Bus route information is never fixed for many reasons (new routes, delays etc) and it is therefore not feasible to store all the alternatives as digital recordings. TTS solves this problem by using computers to generate the utterance through synthesized speech.
Many types of applications can benefit from TTS functionality. They cover a wide range of products in the markets of computers (including multimedia), telephony, medicine, automotive and consumer electronics.

In the *telecommunications business*, the technology can be utilized in such applications as home banking, remote e-mail and fax access, database driven inquiry systems and PC-based phone management systems.

Integrated in new features for *automotive electronics* such as phone, navigation and information systems, TTS also helps in reduction of potential safety hazards. It provides a common hands-free interface, thereby keeping the attention of the driver on the road and increasing the driving comfort and safety.

Consumer electronics can also highly benefit from the use of TTS functionality. Toy and appliances markets present one specific range of needs while pocket

[1] `http://en.wikipedia.org/wiki/Touchscreen`

(or hand held) translators, digital answering machines, portable and cellular phones, organizers and PCs cover the other end of the spectrum.

In the industry, TTS can be an essential feature in production of alarm systems and announcement systems. It can also be used for dictation facilitating hands- and eyes-free operation.

In the *computer* industry, TTS provides a considerable added value for both business and home applications. Language learning, PC-based video games, proof reading and data verification, message notification, answering machines and television viewing have become leading TTS features in the home PC market. In computer multimedia, TTS is for example used in addition to synced video data to provide extra intelligibility for visually impaired [19]. Yang et al. describe an application designed for a translation task, giving two people, who speak different languages the possibility to talk to each other over the Internet with the use of speech synthesis [42].

In the *medical field*, TTS is an excellent contribution to the production of helping tools for people with disabilities. One example is use of a communication aid for vocally impaired persons, to help them "talk" [7]. World renowned physician Stephen Hawking uses such technology [38]. Shi and Maier states that for people with disabilities, such as visually impairment (e.g the elderly), the screen text is hard to identify and it would be helpful to have the option to use the ears to listen to the contents of computer applications [36]. Additional support for input and output, in the form of speech (multimodal interaction), can be added to smartphone applications in order to expand the target audience of such applications further to include the visually impaired and elderly and even non-natives. The spoken dialog system *Let's Go* (2003) has implemented functionality to make bus route information available for these user groups [32]. For the non-natives, using such a system can even aid in learning the local language by helping them acquire the vocabulary, grammar, and phonetic knowledge necessary to fulfill the task the system was designed for [31]. TTS in TaleTUC will, in practice, work like these systems, giving bus route information by speech.

2.3 Cloud vs Local Computing

Zhang et al. point out that a standard definition of *cloud computing* is not easily created [44]. They argue that cloud computing is not a new technology, but rather a new operations model that brings together a set of existing technologies to run business in a different way. In the Search Engine Strategies Conference[2]

[2] `http://www.searchenginestrategies.com/`

in 2006, Google's CEO Eric Schmidt used the word *cloud* to describe the emergent business model of Internet-provided services. The term *cloud computing* has since gained popularity and has been used mainly as a marketing term in many different contexts to represent different ideas. Vaquero et al. analyzed over 20 different definitions from a variety of sources in order to find a complete consensus definition [39]. This thesis adopts the definition of Mell and Grance (created for The National Institute of Standards and Technology, NIST) because it is concise and covers the essential aspects of cloud computing:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [23].

Cloud service providers move the computing away from local computers and devices. They deliver computation as a service to developers through another level of abstraction. The goal is to achieve the same or better service and performance as if the software was installed locally on end-user computers. Services are provided through the network to any device compatible with the services' interface. Popular ways to access the services are through web browsers or mobile applications. The cloud does not require the users to have knowledge of the physical location of the services or details about the infrastructure, so the users can save time by only focusing on the input and output of the cloud service. Hayes highlights the Google Docs service as a good example of a cloud service, where major components of the software reside on unseen computers possibly scattered over continents [12]. This abstraction facilitates the use of light-weight (thin) clients, which are often just web or smartphone applications, since the business logic and data can be stored on servers in some remote locations.

A cloud computing environment can be divided into three layers. The *Infrastructure* layer is responsible for physical resource management (servers, routers, power and cooling systems, etc). It also provides resource virtualization that facilitates dynamic resource assignment. The *Platforms* layer consists of operating systems and software frameworks that facilitate the deployment of applications without the cost of buying and managing the underlying hardware and software. The *Application* layer provides cloud applications that leverage the automatic scaling feature for improved performance, availability and low operating cost. Each of these layers may be implemented as a service to the layer above, and conversely, as a customer to the layer below. Figure 2.1 gives an overview of the architecture.

Speech synthesis data can take up many gigabytes of storage space, which

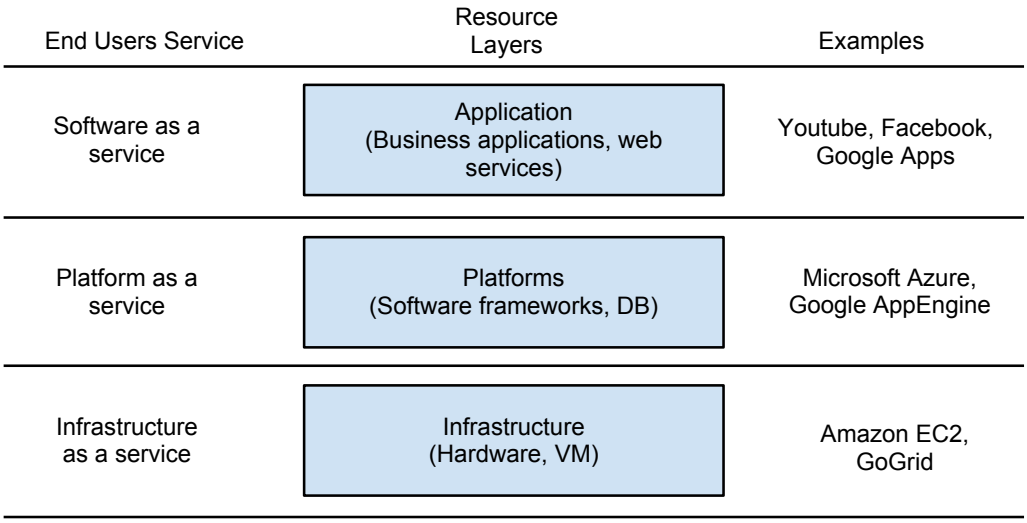

Figure 2.1: The architecture of a cloud computing environment.

makes it infeasible for use directly on most mobile devices today. Also, Andersstuen and Engell show that moving business logic from a smartphone device to a server will conserve battery power since Central Prosessing Unit (CPU)-cycles on the smartphone will use more power than sending/receiving data over a network [2] . It is also easier to update and maintain a server instead of making all the end-users update their applications with lots of data regularly. Keeping the data on servers and deliver synthesized speech on demand is therefore the only viable solution. Using a cloud service for speech synthesis in a smartphone application would bring both advantages and disadvantages. It is a matter of trust whether a company should maintain server infrastructure and services by themselves, or if they should leave it to other companies that specialize in that service domain. Companies specializing in providing voice synthesis through Software as a Service (SaaS) will continuously strive to improve and to keep their TTS services up to date. This may serve to bring more natural, human-like voices in the future. This is a big advantage, since creating, updating and maintaining in-house solutions are time consuming processes.

By outsourcing the server infrastructure to the clouds, the business risks are moved from the service provider (cloud service customers) to the cloud providers. Hardware failures and connectivity issues are then often handled by people that have more expertise and are better equipped for managing and dealing with these risks. The service providers can then cut down the costs of hardware maintenance and staff training [44]. On the other hand, Heiser

and Nicolett state that cloud computing is filled with security risks, saying it has *"unique attributes that require risk assessment in areas such as data integrity, recovery, and privacy, and an evaluation of legal issues in areas such as e-discovery, regulatory compliance, and auditing"*[13]. They also claim that smart customers should perform a thorough security assessment before committing to a cloud vendor, preferably from a neutral third party. Brodkin follows up this article by describing the seven main security risks [5]. He illuminates system vulnerabilities when handling sensitive data. Data access, encryption, location, logging, backup recovery are of high concern. TaleTUC only requires a quality SaaS, meaning that it needs stable uptime, good response times and high quality TTS-conversion. No data storage is needed. No sensitive data is handled since the text queries used for input contain trivial information. Hence, these security risks are of no issue.

The decision of whether to use local, self-maintained servers or to use cloud services comes down to performance. Availability, response time and quality of voice synthesis should be the key properties that point a TTS solution in either direction.

2.4 BusTUC

BusTUC is the main building block of the FUIROS project and the underlying system used by all of its assignments. Amble describes it as a text-based question answering system for questions about bus transportation [1]. The Natural Language Processing (NLP) module BusTUC uses is based on a complex set of rules and is implemented in Prolog[3]. It is versatile in the way that it can answer a variety of alternative formulations requesting the same kind of information. This question-answering system consists of three modules. The BusLOG module includes the bus route database, the list of bus stop of names (including mappings from street descriptions to bus stops), and a route analyzer/planner, which finds the shortest/best route between two given bus stops. Bus transfer is handled if there is no direct route. The second module is a general text understanding module (The Understanding Computer, TUC) which performs rule-based grammatical and semantic parsing. The third and main module integrates TUC and BusLOG, and tailors the system to process a complete inquiry in a single sentence. This is the part which is called BusTUC, as it is the part specifically made for the question-answering mode. In the publicly available online version there is no memory as there is no dialog; i.e., every question is concerning a new, independent inquiry and must contain all the semantic enti-

[3] `http://www.sics.se/isl/sicstuswww/site/index.html`

ties necessary to provide an answer from the bus route database. In the dial-up version, a template-filling dialog approach is used. BusTUC will typically understand and respond to sentences like:

I would like to travel from Studentersamfundet to Lade Kirke in about one hour from now.

When is the next bus from the City Centre to Ila?

The BusTUC system was commercialized, and is publicly available as a service from the bus company in the city of Trondheim, AtB[4]. A web service has been operational since 1998 and a SMS service since 2002.

2.5 Existing TTS Solutions for Norwegian

This section provides a description of existing TTS systems used for speech synthesis in Norwegian. Creating in-house TTS systems from scratch are big projects on their own and out of the scope of this thesis. The focus is therefore on APIs or services that provide interfaces or complete solutions that allow a much shorter implementation phase. The underlying technology used for some of these solutions are company secrets and therefore not described.

2.5.1 Microsoft Speech API (SAPI)

The Speech Application Programming Interface (SAPI)[5] is an API developed by Microsoft to enable speech recognition and speech synthesis in Windows applications. SAPI can be viewed as a piece of middle-ware or as a high-level interface between an application and speech synthesis. SAPI implements the low-level details needed to control and manage the real-time operations of various *speech engines* (voices). Speech engines are runtime packages that include the language model, acoustic model, and other data necessary to provision a speech engine to perform speech synthesis in a particular language. This abstraction facilitates an easier and shorter implementation phase. Shi and Maier state that: *"The development of a standard speech interface like SAPI provides a very positive outcome for those who want to use speech functionality with minimus technical details."*[36]. It is also flexible in the way that SAPI compatible speech engines can be set to use a specific pitch, speed (words/min) and volume. SAPI also

[4] `www.atb.no`

[5] `http://www.microsoft.com/en-us/tellme/`

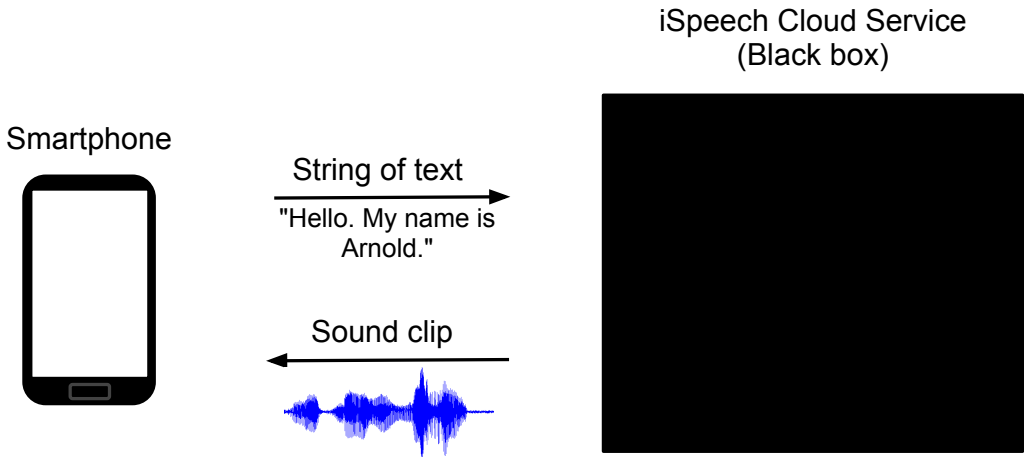

Figure 2.2: The black box structure of the iSpeech cloud.

provides a user lexicon (user dictionary) which allows custom words and pronunciations to be added by a user or application. SAPI was first introduced in Windows 95. The current version of Microsoft Windows, Windows 7, ships with a narrator program[6] that uses SAPI, with the speech engine Microsoft Anna. Also, some versions of Microsoft Office use SAPI. Speech engines are available in 26 languages, Norwegian included.

2.5.2 iSpeech

iSpeech[7] is a provider of cloud-based speech technology. It supports web applications and many mobile platforms, including Android, iPhone and BlackBerry. iSpeech is mostly known for their *DriveSafe.ly* smartphone application that reads your text messages and emails out loud so you can concentrate on the road. Also, they provide free SDKs and APIs for mobile devices that offer over 40 TTS voices with support for more than 25 languages, including Norwegian. Through the API, iSpeech delivers synthesized speech through a black box SaaS. The user provides the text string to be synthesized and the cloud does the speech synthesis. It then returns the audio and automatically plays it (see Figure 2.2).

[6] `http://windows.microsoft.com/en-US/windows-vista/Hear-text-read-aloud-with-Narrator`
[7] `http://www.ispeech.org/`

2.5.3 Nuance

Nuance[8] is a multinational computer software technology corporation that focuses on providing server and embedded speech and imaging applications. Nuance's NDEV Mobile provides access to their speech platform via the Dragon Mobile SDK and API. This platform offers a blackbox TTS service similar to iSpeech's, hiding away details in how the technology works. The SDK provides support for more than 35 languages, including Norwegian. Nuance provides 90 days of free access to the cloud-based speech services and then a running subscription needs to be bought (which is expensive).

2.5.4 eSpeak

eSpeak is a compact open source software speech synthesizer for Windows, Linux and other platforms. The technology used is the formant synthesis method, providing many languages. Still, it only requires a small amount of storage space. Much of the programming for eSpeak's languages was based on information found on Wikipedia, with some subsequent feedback from native speakers. Google Translate has used implementations of eSpeak to provide TTS functionality for many languages, including Norwegian[9].

[8] http://www.nuance.com/

[9] http://googleblog.blogspot.no/2010/05/giving-voice-to-more-languages-on.html

Chapter 3

Methods

This chapter describes the methods used to implement the goals of the thesis. This includes the prototypes for the *TaleTUC: Text-to-speech* system.

3.1 The Nora Client

This section describes the system architecture of the Nora Client and how the complete BUSTER system works.

3.1.1 The BUSTER System

Harborg describes a TTS solution installed on the university network of NTNU [10]. It is a product of the BRAGE project conducted by NTNU, Telenor Research and Innovation[1] and SINTEF Information and Communication Technology[2] to create five demonstrators for information access and retrieval within the domain of timetable information for bus transportation, directory assistance and visitor guidance.

One of these demonstrators, BUSTER, provides information about bus transportation in the city of Trondheim using a dial-up phone dialog interface [15]. It is a spoken dialog system that makes use of BusTUC to answer questions regarding bus routes [1]. This system includes a speech recognizer module to handle natural speech input and a TTS module set up to create the synthesized speech for the response. The speech recognizer has a vocabulary of about 800 words, where around 700 contain names of bus stops and area descriptions of Trondheim. The system is robust in the sense that it degrades the dialog towards a system driven approach. While the underlying text-based system

[1] `http://telenor.com/no/innovasjon/`

[2] `http://www.sintef.no/home/Information-and-Communication-Technology-ICT/`

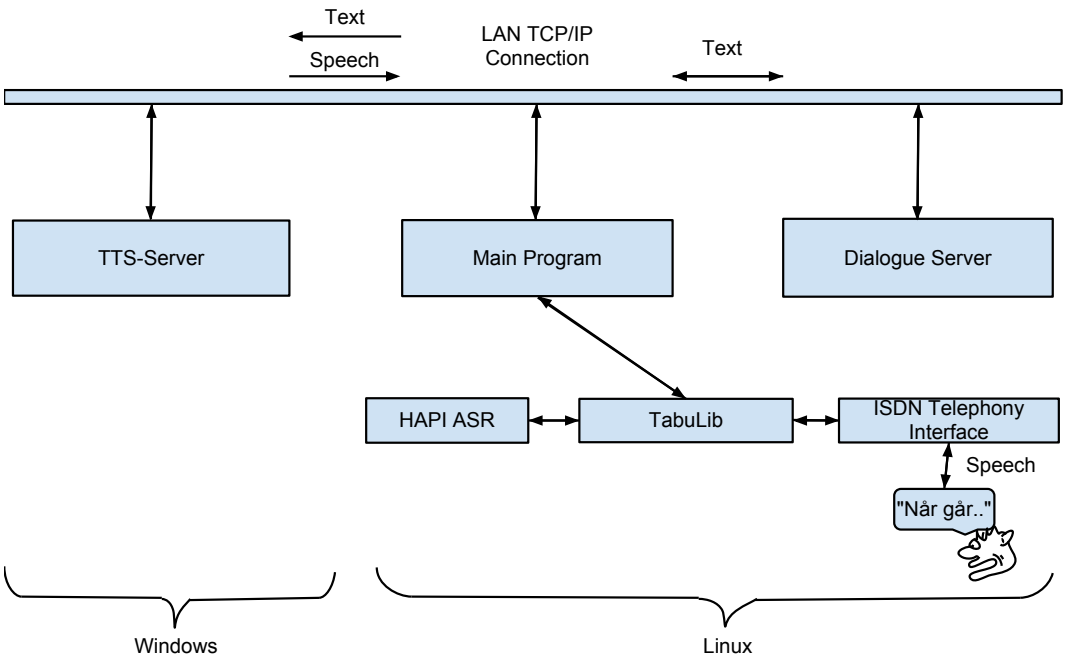

Figure 3.1: The server architecture of BUSTER.

(BusTUC) deals with a wide scope of questions, the speech based version has a grammar that limits the question to only consider how to come from one place to another at a specific day and time. Figure 3.1 shows an overview of the server architecture BUSTER makes use of.

The server architecture is of modular design. The various modules are running on different computers, based on both Linux and Windows. The communication between the three top modules (main program, dialog server and TTS server) is performed over a TCP/IP connection using sockets[3].

The *main program* module is written in the Python[4] programming language. It communicates with the Tabulib program library that handles speech detection and the ISDN[5] telephony interface [17]. It is also integrated with the HAPI automatic speech recognizer (ASR) [26]. The main program runs on a Linux server with a Debian[6] operating system (version 3.1), and with CAPI[7] installed.

[3] http://www.troubleshooters.com/codecorn/sockets/
[4] www.python.org/
[5] http://en.wikipedia.org/wiki/Integrated_Services_Digital_Network
[6] http://www.debian.org/
[7] http://www.capi.org/

An active ISDN telephony card[8] is used. The main program makes a semantic analysis of the text provided by the speech recognizer in order to make sure that only meaningful sentences are passed on to the dialog handling system (on the dialog server). Depending on the type of utterance that is received by the recognizer, it is either sent to the dialog server for further computing or handled locally in the main program to create a textual response to be sent to the TTS module for speech synthesis.

The *dialog server* is a module written in Python. It receives text input from the main program and sends it to the dialog handling system, which is written in Sicstus Prolog[9] (version 3.11). The dialog handling system does some text filtering before the response is sent back to the main program.

The *TTS server* is a module written in the C++ programming language[10] and it supports the TTS engines that are SAPI4 compatible. The speech engine currently used is the commercially available Lernout & Hauspie (L&H) Telecom RealSpeak SAPI4 (version 3.11) Release 6 from Nuance[11], with the Norwegian voice (speech engine) of Nora. This TTS solution will from now on be referred to as *Nora*. The server runs on a Windows XP machine. It has a socket interface for the TTS engine that listens for text input and responds with complete synthesized speech audio data. This audio is then forwarded to the telephony interface for user response.

3.1.2 Client Prototype

The Nora Client is written in the Java programming language on the Android platform for smartphones [4]. The platform is chosen because the main target for this application is the TABuss prototype, which is an application for Android. The client works by receiving the text string to be synthesized as input. It opens up a socket connection to the TTS server and provides it with the text string. The TTS server answers with raw audio data. The client retrieves this data and stores it temporarily in a byte array. It adds a file header for the data and stores the resulting audio file, making it ready for playback on the Android smartphone device. Any Android applications may consequently use the Android media player[12] or similar software to play the audio file. Figure 3.2 depicts how the Nora Client works.

[8] ISDN card AVMB1PCI, see `http://www.avm.de/en/Produkte/Server-Produkte/B1_PCI/index.html`.

[9] `www.sics.se/sicstus/`

[10] `http://en.wikipedia.org/wiki/C%2B%2B`

[11] `http://www.nuance.com/`

[12] `http://developer.android.com/reference/android/media/MediaPlayer.html`

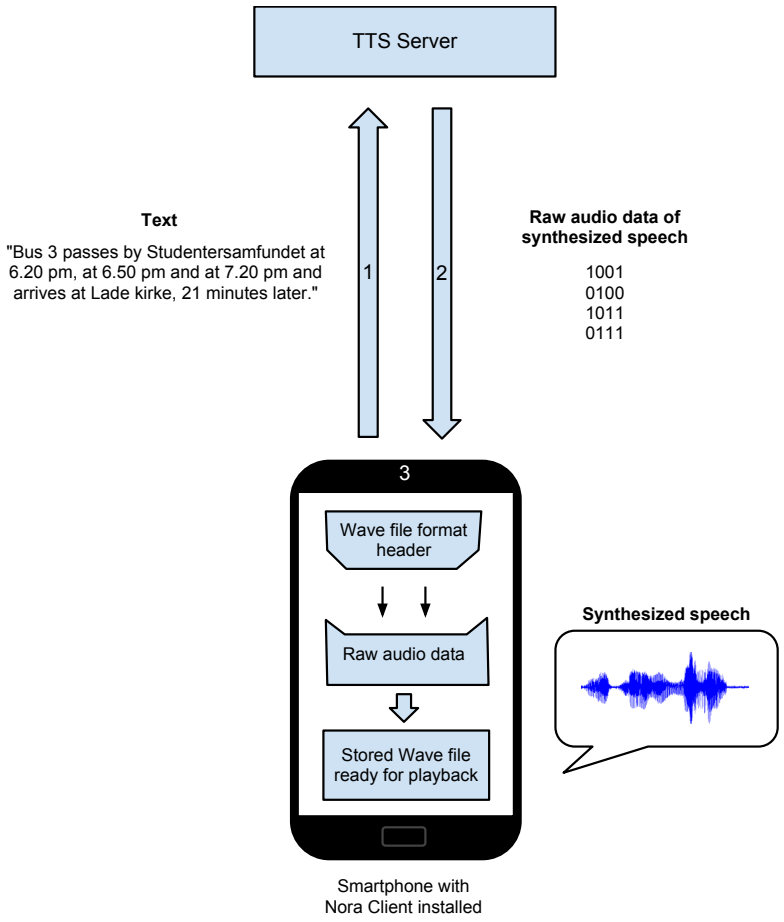

Figure 3.2: The Nora Client prototype.
Step 1: The smartphone sends a text string to the TTS server.
Step 2: The TTS server responds with raw audio data of synthesized speech.
Step 3: The client adds a Wave file header, merges it with the raw audio data and stores the resulting file so it is ready for playback.

The output of the BUSTER TTS module, the voice of Nora, comes in the form of 8-bit ITU-T[13] G.711[14] A-law (.al) speech files. This is raw audio data (bit streams) compressed with an algorithm commonly used in European digital communication systems (telephones mostly) to optimize the dynamic range of an analog signal for digitization[15]. The encoding process breaks the linear audio data into segments. Each progressively higher segment doubles in size. This makes sure that the lowest amplitude signals, where most of the speech information is located, get the highest bit resolution. This is while still allowing enough dynamic range to encode high amplitude signals. The resulting effect is to increases the coding efficiency and give a signal-to-distortion ratio that is superior to that obtained by linear encoding [6]. This method does not provide a very high compression ratio. The A-law compressed files roughly have half the file size. They do not require much processing power to decode though.

Since the output comes in raw data form, it needs to be put in a file format that is directly playable by the Android media player. The Wave[16] file format is compatible[17]. It is a Microsoft and IBM audio file format standard and is very convenient since it is widely supported by most media players and audio tools on many operating system. In order to make the A-law data into a valid Wave file format, one needs to implement a Wave file header. Figure 3.3 shows what the header used for the Nora Client looks like.

3.1.3 Extending Nora with Java Servlet Technology

The Nora Servlet is a server set up as an interface between the smartphone and the TTS server. It is implemented in such a way that it should do all the work that the Nora Client does. Actually, it integrates the Nora Client directly. In addition, it provides compression capabilities in order to conserve the amount of data that has to be sent to the smartphone.

Java Servlet Technology

The server technology used is Java Servlet[18], developed by Sun Microsystems in 1997. The Java Servlet technology is now at version 3.0 and has been around for over a decade. It is a well tested and documented technology. With Java

[13] http://www.itu.int/ITU-T/
[14] http://en.wikipedia.org/wiki/G.711
[15] http://www.digitalpreservation.gov/formats/fdd/fdd000038.shtml
[16] http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html
[17] http://developer.android.com/guide/appendix/media-formats.html
[18] http://www.oracle.com/technetwork/java/javaee/servlet/index.html

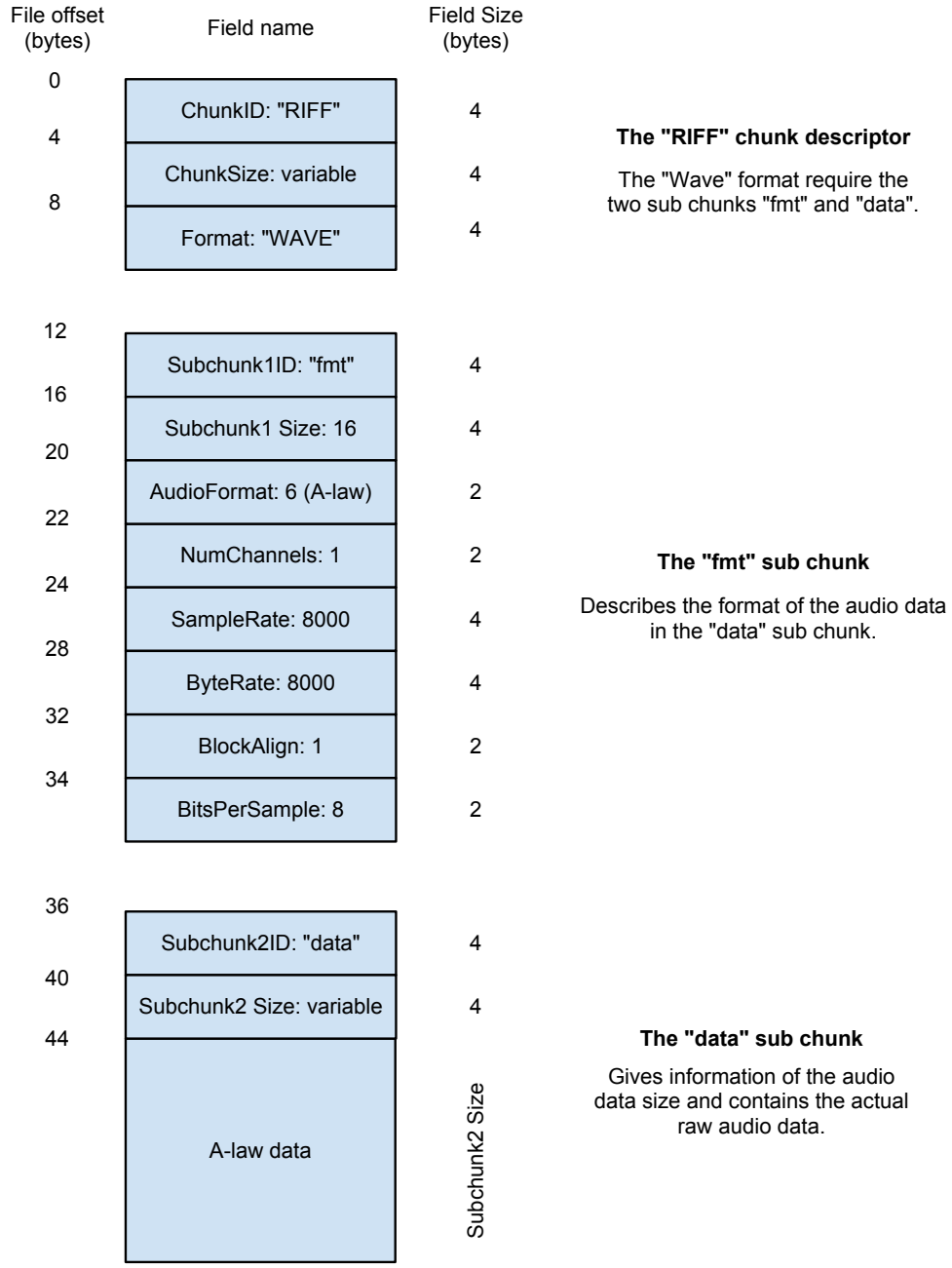

Figure 3.3: The Wave file format header, used for the audio data received by the Nora Client. The Wave file format is a subset of Microsoft's RIFF specification for the storage of multimedia files. A RIFF file starts out with a file header followed by a sequence of data chunks. A Wave file is just a RIFF file with a single "Wave" chunk which consists of two sub chunks: A "fmt " chunk specifying the data format and a "data" chunk containing the actual audio data. The header field values show what values should be set in a Wave file format with A-law audio data.

Servlets, business logic can easily be written in Java and then be made available to consumers through Java Servlets [27]. A Java Servlet can be published through any available Java Servlet container, which makes it very portable. A Java Servlet container, also known as a web container, is a component of a web server that interacts with a Java Servlet and makes the communication between a Java Servlet and a web client possible. There are many Java Servlet containers available. Some are commercial and some are not. The best known commercial containers are IBM's WebSphere[19] and SAP's NetWeaver[20]. Among the non-commercial Java Servlet containers are Apache's Tomcat[21] and Glassfish from Sun Microsystems[22]. As this was an academical project a non-commercial server would be needed. The one chosen is called Jetty, from Eclipse Foundation[23]. It is seen at a rising star in the web community, much because Google chose to use Jetty technology for their cloud service, Google App Engine[24]. What is special about Jetty is that it is made up of pure Java code. As known Java is very portable [4]. This all facilitates a very portable server solution. Using a server one can also provide logging functionality, keep track of user statistics, counteract abuse (such as DDOS[25]).

Nora Servlet Client

Before the string to be synthesized is sent to the Nora Servlet, it is first encoded to UTF-8 to avoid errors of various kind, e.g. whitespace issues. A query is sent to the Nora Servlet as a HTTP GET and parameter data is sent through the URL. Then the client receives the audio data. A typical query to the server looks like this (the UTF-8 encoding will replace the whitespaces with %):

http://vm-6114.idi.ntnu.no:9007/NoraServlet/NoraServlet?text=Bus 8 passes by Torget at 5.58 am and at 6.18 am and arrives at Ila, 11 minutes later.

Nora Servlet

The Nora Servlet has capabilities of tailoring the output from the TTS server in different ways. For instance, it can compress the output (the synthesized speech audio data) further to conserve data transfer from the server to the smartphone

[19] `www.ibm.com/software/websphere/`
[20] `http://www.sap.com/platform/netweaver/index.epx`
[21] `http://tomcat.apache.org/`
[22] `http://glassfish.java.net/`
[23] `http://www.eclipse.org/jetty/`
[24] `http://code.google.com/intl/no/appengine/`
[25] `http://en.wikipedia.org/wiki/Denial-of-service_attack`

device. Keeping down the data transfer size is not only important to shorten the response time of TTS requests, but also because the communication cost will be reduced on network connections that are typically charged on a per byte basis.

Audio codecs that perform compression and decompression, and digital signal processing (DSP), are commonly used in voice communications. They can be configured to conserve bandwidth. However, there is a trade-off between voice quality and bandwidth conservation. The best codecs provide the most bandwidth conservation while producing the least degradation of voice quality. Bandwidth can be measured quantitatively, but voice quality requires human interpretation, although estimates of voice quality can be made by automatic test systems such as Perceptual Evaluation of Speech Quality (PESQ). PESQ is a widely used perceptual measurement method for voice quality in telecommunications [28]. It was developed by OPTICOM GmbH in Germany and forms the basis of ITU-T Recommendation P.862. PESQ is designed for testing voice quality on low bandwidth devices such as telephones, smartphones and hands-free devices. Mean Opinion Score (MOS) results from PESQ achieve a very high correlation with results obtainable using human subjects in a way that is faster, more repeatable, less expensive, and fully automated [33].

With today's compressing codecs, we can compress the output audio data of the TTS server further (than A-law) without significant quality loss. Codecs designed for speech compression, like Speex[26], are optimal. However, Android does not support these[27]. MP3 and Ogg Vorbis (that are supported) are therefore used as compression codecs on the Nora Servlet for testing purposes. Using PESQ to determine which audio codec is optimal is not done in this thesis, but is described in Section 6.1.1.

When a HTTP GET has been sent to the Nora Servlet from the Nora Servlet Client with a text string, the Nora Servlet contacts the TTS server (similarly to the Nora Client) and receives the raw audio data. The raw data is then fit with a Wave header such as the one previously mentioned for the Nora Client. The audio data is then uncompressed (still in Wave format) using an A-law decompression lookup table[28] in order to arrange the data for compression. The uncompressed data is consequently compressed using either MP3 or Ogg Vorbis (or Speex) and sent to the smartphone for playback. The codecs used are LAME MP3 encoder and oggenc2 Ogg Vorbis encoder from rarewares[29]. These are open source-based binaries. These codecs have quality settings (ranging

[26] http://www.speex.org/

[27] `http://developer.android.com/guide/appendix/media-formats.html`

[28] `http://www.threejacks.com/?q=node/176`

[29] `http://www.rarewares.org`

from 0-9) that tell the codec to either focus on preserving the best sound quality or to focus on minimizing file size.

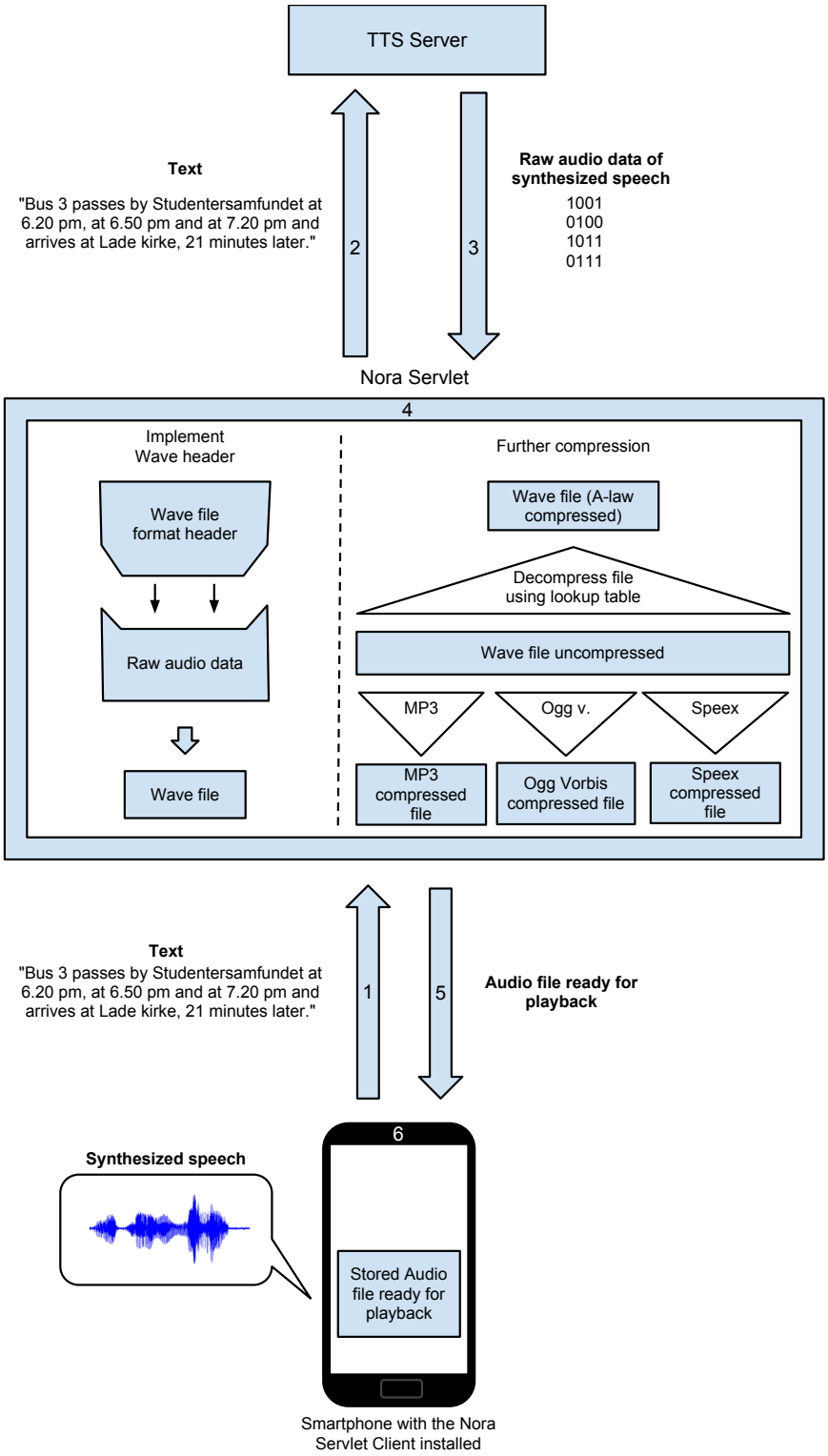

Figure 3.4: The Nora Servlet prototype.
Step 1: The smartphone sends an UTF-8 encoded text string to the Nora Servlet.
Step 2: The text string is simply forwarded to the TTS server.
Step 3: The TTS server responds with raw audio data of synthesized speech.
Step 4: The Nora Servlet implements a Wave file header and merges it with the raw audio data. It is then uncompressed and compressed in another audio file format (MP3, Ogg Vorbis or Speex).
Step 5: The resulting compressed audio file is sent back to the smartphone.
Step 6: The audio file is stored and ready for playback.

3.2 Physical Server Setup

There are two physical servers used in this thesis. They are *busstjener.idi.ntnu.no* and *orwell.idi.ntnu.no*. The server busstjener.idi.ntnu.no has the Nora Servlet installed and orwell.idi.ntnu.no provides the TTS service. All results are produced using these servers. In tables 3.1 and 3.2 the specifications of the servers are given.

Attribute	**Value**
CPU	2x 5.2 GHz, VMware shared pool [30]
Memory	4 GB dedicated
OS	Ubuntu 11.04 (GNU/Linux 2.6.38-8-server x86_64)

Table 3.1: Server information for busstjener.idi.ntnu.no.

Attribute	**Value**
CPU	Pentium 4, 3.0 GHz
Memory	2 GB dedicated
OS	Windows XP Professional Version 2002, Service Pack 3

Table 3.2: Server information for orwell.idi.ntnu.no.

3.3 The Test Application

The Android test application was created for the purpose of testing the three different TTS modules, iSpeech, Nora and Nuance. Similar to the Nora Client it was written in Java. Both the Nora Client and Nora Servlet Client are integrated in this application. iSpeech and Nuance are implemented by the use of their corresponding SDKs (see Section 2.5).

Figure 3.5:a shows the home screen of the test application. From here, the user can navigate either to *Test Environment 1* by tapping the "TTS" button or *Test Environment 2* by tapping the "Test" button".

Figure 3.5:b depicts *Test Environment 1*. It provides tools for testing specific speech synthesis modules. At the top there is a text field where the users can type in any text they want to use as input for any of the modules. The users may also click the "Next" button to scroll through a list of 1400 bus stops in

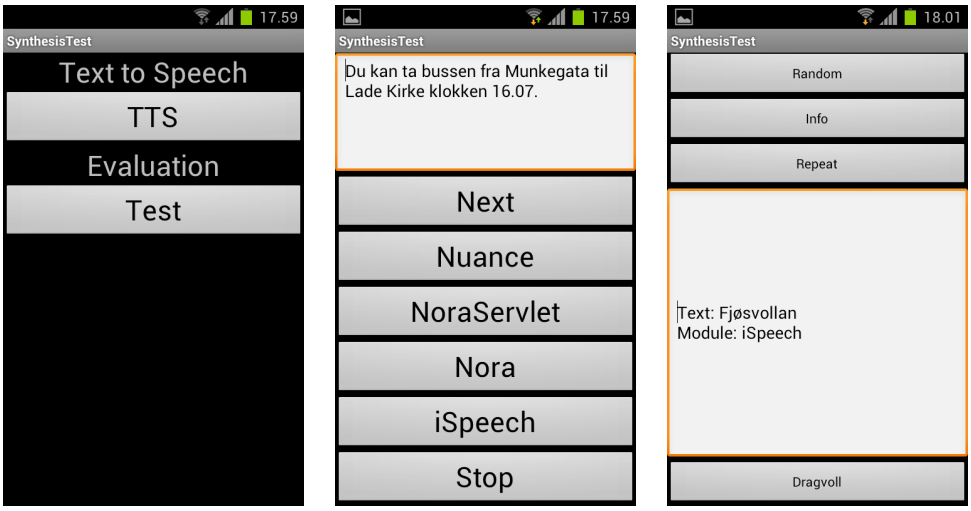

Figure 3.5: The prototype test application used for gathering the evaluation data. Screenshots: (a) Home menu, (b) Test Environment 1 and (c) Test Environment 2.

Trondheim and place one in the text field (replacing any text already present). If the users then click on any of the speech synthesis modules (iSpeech, Nora, Nora through a Java Servlet or Nuance) the module will synthesize the current text string in the text field and the smartphone will play it through the speaker.

Figure 3.5:c depicts *Test Environment 2*. The "Random" button gives synthesized speech by choosing a random speech synthesize module (iSpeech, Nora or Nuance) and providing it with a random bus stop from the bus stop list as text string input. The "Info" button is created for the test examiner in mind. It will present information about what module and text string was used in the previous use of the "Random" button functionality. The "Repeat" button will replay the previous synthesized speech. The "Dragvoll" button at the bottom will play the bus stop name "Dragvoll" synthesized with the Nora module (once with the default pronunciation of the word and once where the pronunciation has been tampered with. See Section 3.6).

3.4 Standardizing Server Communication

The BusTUC system gives answers to bus route information queries in textual format. Both MultiBRIS and TABuss applications already uses for this purpose [2, 21]. As mentioned earlier, BusTUC is written in Prolog. Since BusTUC is

being used as an underlying system for many systems regarding bus route queries, it should give output data that is easy to parse. For developers, this saves a lot of time during the implementation phase. This was achieved by altering the Prolog code, formatting the data into JSON as follows:

```
{
    "transfer": "false",
    "timeset": "false",
    "departures": [
        {
            "busstopname": "Studentersamfundet",
            "busstopnumber": 16011476,
            "busnumber": 92,
            "time": 523,
            "duration": 3,
            "destination": "Sentrumsterminalen"
        },
        {
            "busstopname": "Studentersamfundet",
            "busstopnumber": 16011476,
            "busnumber": 94,
            "time": 527,
            "duration": 3,
            "destination": "Sentrumsterminalen"
        },
        {
            "busstopname": "Studentersamfundet",
            "busstopnumber": 16011476,
            "busnumber": 8,
            "time": 555,
            "duration": 3,
            "destination": "Sentrumsterminalen"
        }
    ]
}
```

Listing 3.1: Bus route answers in the JSON format.

3.5 Character Encoding (UTF-8)

A character encoding system assigns a computer-internal representation (e.g. a number) to every character of an alphabet. Web pages can use a variety of different character encodings, like ASCII, Latin-1, Windows 1252 or Unicode. Most encodings today can only represent a few languages, but Unicode can represent thousands: from Arabic to Chinese to Norwegian.

Unicode Transformation Format (UTF-8) is a character encoding that can represent every character in the Unicode character set, which is a computing industry standard for representation and handling of text expressed in most of the world's writing systems. It is designed for backward compatibility with ASCII and to avoid the complications with UTF-16 and UTF-32. UTF-8 can also be used in programming languages and compilers that are not designed for Unicode. UTF-8 has become the dominant character encoding for the World Wide Web. It is the one of the standards that are included in HTML and XML documents and accounts for more than half of all web pages[31]. Google uses Unicode as the internal format for all the text they search (any other decoding is first converted to Unicode for processing).

FUIROS systems (like BusTUC) and documentation are written in both Norwegian and English. However, bus route information needs to be presented in *either* in Norwegian or English. This causes problems with character encoding systems that do not fully support both languages. Due to the large amount of characters UTF-8 can represent, including the Norwegian-specific characters *æ*, *ø* and *å*, it is a standard that should be used in all of FUIROS systems to avoid such issues.

In the Prolog code of BusTUC, adding an UTF-8 header to all the files solves the problem. The Eclipse[32] Integrated Development Environment (IDE) provides functionality to add a Prolog header, which is UTF-8 specified, to the source code files. It looks like this:

```
/* -*- Mode:Prolog; coding:utf-8; -*- */
```

In order to make every Prolog file UTF-8 compatible, 75 files needed to be manually set with this as header. Unfortunately this caused the *æ*, *ø* and *å* characters in the code to be represented incorrectly, so these also needed to be corrected manually.

[31] `http://googleblog.blogspot.no/2010/01/unicode-nearing-50-of-web.html`
[32] `http://www.eclipse.org/`

3.6 Pronunciation Optimization

This section describes how to alter pronunciation of bus stop names using the L&H User Dictionary Editor (UDE), which is a tool that comes bundled with the L&H Realspeak speech engine mentioned in Section 3.1.

The L&H TTS RealSpeak system, which the Nora voice is based on, supports user dictionaries. User dictionaries make it possible to customize the output of a TTS system and the input of an ASR system. It allows the user to specify special pronunciations for particular words or strings of characters (e.g. abbreviations). When a user dictionary has been loaded, the TTS system will look up every word of the input text in the user dictionary. If a word is found in the dictionary, the TTS system will substitute the pronunciation that has been specified in the dictionary for the pronunciation that would be generated automatically by the TTS system.

The L&H User Dictionary Editor is an application designed to create and edit such user dictionaries. User dictionaries may contain orthographic as well as phonetic information. The L&H UDE includes two phonetic alphabet sets that are used to generate phonetic transcriptions: the L&H+ phonetic alphabet and the International Phonetic Alphabet (IPA)[33]. If the user selects plain orthographic, the destination text of the dictionary entries should be specified in orthographic spelling. Plain orthographic mode is especially useful if the user want to add abbreviations to the user's user dictionary or if you want to define a "sounds like" string for the source text. The User Dictionary Editor allows you to listen to how dictionary entries are pronounced, using an L&H TTS engine. The speech parameters of the TTS system can be set according to your preferences. The User Dictionary Editor allows you to enter several pronunciations for a single word and all of them will be recognized by the ASR.

The UDE-tool is well suited for making alterations to bus stop name pronunciation. In this thesis it was used to alter the pronunciation of the bus stop name "Dragvoll" for testing purposes. This bus stop name was chosen because the intelligibility and naturalness of its pronunciation were deemed poor by the author of this thesis. "Dragvoll", with a capital "D", was given new phonetic values that sounded better:

"dr6AgfOll"

Now it was easy to compare it to the default pronunciation that was given by the word "dragvoll" (without the capital "D"), making it ready for the evaluation test in Section 4.2.

[33] `http://en.wikipedia.org/wiki/International_Phonetic_Alphabet`

3.7 The New BusTUC Web Page

A new web page for BusTUC has been created[34]. The web page was created by the use of HTML5 the and jQuery JavaScript library.
The FUIROS project has a big vision to become the ultimate bus route information system for the future (hence the name). As for the *future* part of the vision: This is a research assignment. The web page written here is a prototype. Therefore it is important for further development of this prototype that it is created within standards that can stand the test of time ("future proof"). This does not only mean functionality, but also that is should be visually pleasing. Renewing this web page consists of removing old technology such as frames, giving it a more streamlined look. Also, replacing old PHP-scripts with JavaScript code is part of this process.

Big software companies like Apple and Adobe both see the potential of the open standards HTML5, JavaScript and CSS [2]. These technologies are standards that have been around for many years and are firmly set in the world wide web. The Web Hypertext Application Technology Working Group (WHATWG) states that preserving backwards compatibility with browsers designed for earlier versions of HTML is one of the key features of HTML5[35].

3.7.1 HTML 5

HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the WHATWG. HTML5 will be the new standard for HTML, XHTML, and the HTML Document Object Model (DOM). The previous version of HTML (Version 4) came in 1999. The web has changed a lot since then and new functionality is needed to give more HTML native support for new functionality. HTML5 is still work in progress. However, some rules for the final HTML5 standards have been established:

1. The new features should be based on HTML, CSS, DOM, and JavaScript
2. Reduce the need for external plugins (like Flash)
3. Provide better error handling
4. Contain more markup, to replace scripting
5. HTML5 should be device independent
6. The development process should be visible to the public

The new main features in HTML5 are:

1. Canvas element for drawing

[34] `http://busstuc.idi.ntnu.no/`
[35] `http://wiki.whatwg.org/wiki/FAQ`

2. Video and audio elements for media playback
3. Better support for local offline storage
4. New content specific elements, like article, footer, header, nav, section
5. New form controls, like calendar, date, time, email, url, search

The most interesting property of HTML5 in the context of this thesis, is that less code is required for development.

3.7.2 CSS 3

Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation semantics of a document written in a markup language. CSS is primarily designed to make it easy to style fonts, color and layout for different parts of an web page.

The new CSS 3 standard differs from the old ones in that it uses modules that handle different types of styling. These modules are manifested text documents, and each module adds new capability or extends features defined in CSS 2 standard. The first CSS 3 draft came already in June 1999, but the the first W3C recommendation for a CSS 3 module was made in in June 2011. JQuery, described later in this section, uses CSS 3 for styling purposes.

3.7.3 JavaScript

JavaScript is a prototype-based scripting language that is dynamic, "weakly typed" and has first-class functions, as explained below. It is a multi-paradigm language, supporting both object-oriented, imperative and functional programming styles. Prototype-based simply means that one does not use classes. Behavior reuse is accomplished through cloning of existing objects which then serves as prototypes. "Weakly typed" means that JavaScript is not strict on how different data types are mixed. JavaScript has first-class functions, which means that it treats functions as first-class objects, and can therefore pass functions as arguments to other functions.

JavaScript was for a long time seen as the black sheep of the web, but since the AJAX web development method became popular JavaScript has redeemed itself. Traditionally JavaScript has only been used on the client sides, but lately better virtual machines have been developed for it to run on. Hence, JavaScript is now also on the server side. As one can see from the HTML 5 section above, JavaScript plays an important role when working with the new HTML standards. As JavaScript is a scripting language, it makes it possible to move most of the business logic to the client side of the web-application, if desired.

The author of this thesis is primarily a Java developer and does not have extensive knowledge of coding web pages in HTML. Using the JavaScript approach, which is very similar to Java in sense of syntax and control, therefore makes this the best choice for programming language.

3.7.4 jQuery

Several JavaScript frameworks out there that help to ease the pain of making a graphical user interface. One of these, *jQuery*[36] is a popular alternative and provides everything needed for the BusTUC web page. jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. Especially the handling of Ajax interactions is something that helps tremendously when creating this particular web page because of the cross domain communication (see Section 3.7.5).

3.7.5 Same Origin Policy

The same origin policy[37] is an important security concept for a number of programming languages browser-side, such as JavaScript, and provides a challenge when developing the web page. The *origin* term describes resources having the same application layer protocol, domain name and, in most browsers, port number. Two resources are considered to be of the same origin if and only if all the values are exactly the same. The policy allows scripts running on web pages originating from the same site to access each other's methods and properties with no restrictions. For web pages on different sites, however, access is denied. This is a real concern for the communication between the server and client developed in this thesis. Data transfers consist of data in both text and JavaScript Object Notation (JSON) format delivered through AJAX requests. Regular AJAX-calls are prohibited by the browser and fail to work due to the same domain policy. Luckily, there are workarounds for this issue. For regular text data requests, an XMLHttpRequest was used. For JSON data requests, the JQuery API has a solution to this problem. The solution is called *getJSON*[38]. Instead of using the standard AJAX request, JQuery injects a <script> tag into the DOM. For example, if the prototype web page in this project wants to load bus-stop data from `http://busDomain.com/stops`, the injected script tag might look something like this:

[36] `http://jquery.com/`

[37] `http://www.w3.org/Security/wiki/Same_Origin_Policy`

[38] `http://api.jquery.com/jQuery.getJSON/`

<script src="http://busDomain.com/stops?callback=someCallback"></script>

The browser on the client side then makes a request to that url and includes the response as if it was any other type of JavaScript include. Because the client passes a callback in the url above, the busDomain server knows that the client wants to be notified when the result comes in and that it should call a certain callback function with the data it sends back as parameters. As long as the remote server is configured to format the response data accordingly, transfers are completed successfully.

Chapter 4

Results

This chapter presents the research results obtained by use of the methods described in chapter 3. The results presented here are then further discussed in Chapter 5. The measurements for all the tests (except *the New BusTUC Web Page*) were done in Trondheim on the Test Application (see Section 3.3) on a Samsung Galaxy SII.[1]

4.1 Speech Synthesis Module Evaluation

This section presents the results of the speech synthesis evaluation test. Three modules were tested: iSpeech, Nora and Nuance. These modules have been set to their default audio quality setting. The goal of this test was to see which module gives the most *intelligible* and *natural* text-to-speech response. The test subjects were told to use their own definition of "natural speech" and to keep in mind Tanner's definition of "intelligible":

"Speech intelligibility is related to the amount of speech items that is recognized correctly."[37]

The text input is limited to be bus stop names in order to refine the testing to be relevant to this thesis' scope.

The test consists of playing back sound clips of bus stop names to the test subjects by using random speech synthesis modules. The test subjects were exposed to the sound clips by holding up the smartphone's speaker close to their ear in a regular phone call fashion. The subjects then repeated what they think they heard before the correct answer was given to them. They were asked to rate how easy it was to understand what the voice in the sound clips had

[1] `http://www.samsung.com/global/microsite/galaxys2/html/`

said. The rating was done by selecting one of four answer alternatives: "Unintelligible", "Very hard to understand" , "A little hard to understand" and "Fully intelligible". (These answers were transformed into points, ranging from zero to three.) 20 subjects were asked to listen to and rate 15 sound clips, 5 from each speech synthesis module. The large amount of tests were performed in order to achieve statistical significance. The tests were conducted in both low-noise and high-noise scenarios. The low-noise tests were performed inside, in a closed group room at the NTNU campus where the background noise could be kept at a minimum. The high-noise tests were performed in various situations outside; on buses, in areas of high traffic, in shopping malls and in city parks. The average score was computed for each module and plotted into the graphs in figures 4.1, 4.2 and 4.3.

The test subjects were also asked after the tests if they had any general comments. This section was only for writing down what the test subjects had on their mind, anything at all, no matter how relevant to this thesis. The test supervisor was not allowed to hint in any direction.

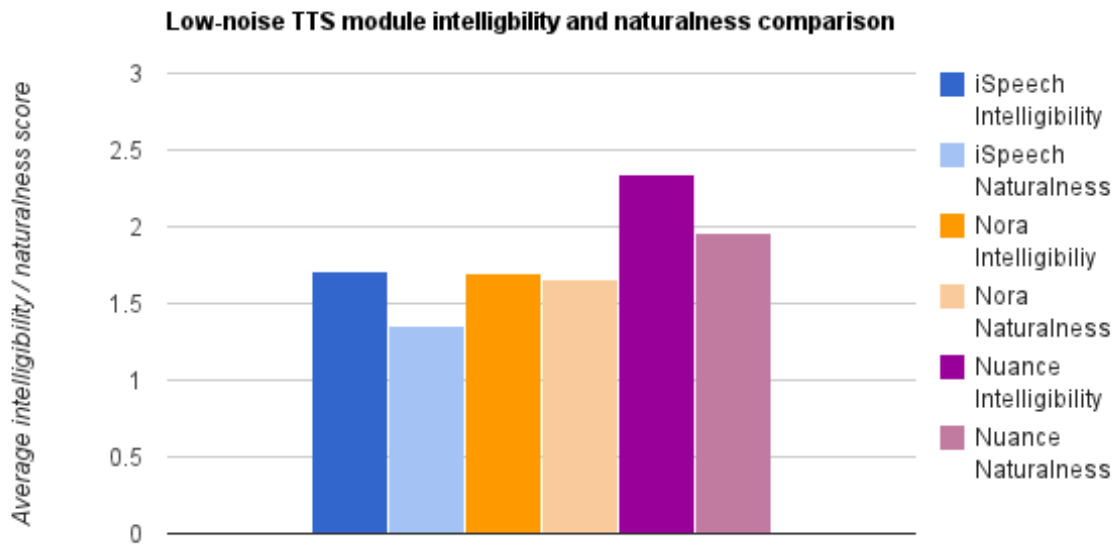

Figure 4.1: The average score for each speech synthesis module in a low-noise environment. The graph shows the average intelligibility and naturalness score for the three speech synthesis modules iSpeech, Nora and Nuance. The y-axis depicts the scores ranging from "Unintelligible/Unnatural" (0) to "Fully intelligible/Natural" (3) (higher is better).

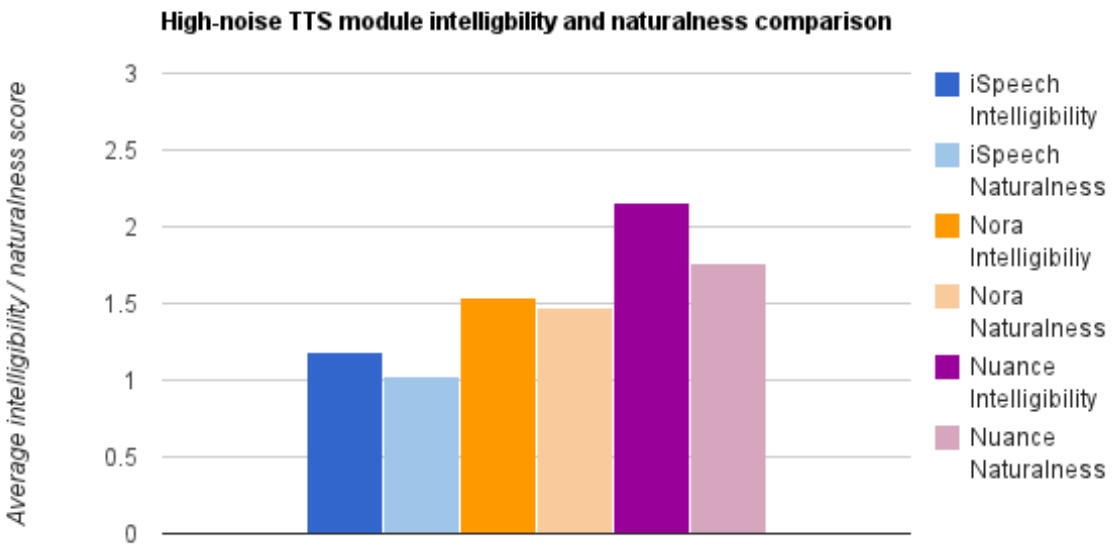

Figure 4.2: The average score for each speech synthesis module in a high-noise environment. The graph shows the average intelligibility and naturalness score for the three speech synthesis modules iSpeech, Nora and Nuance. The y-axis depicts the scores ranging from "Unintelligible/Unnatural" (0) to "Fully intelligible/Natural" (3) (higher is better).

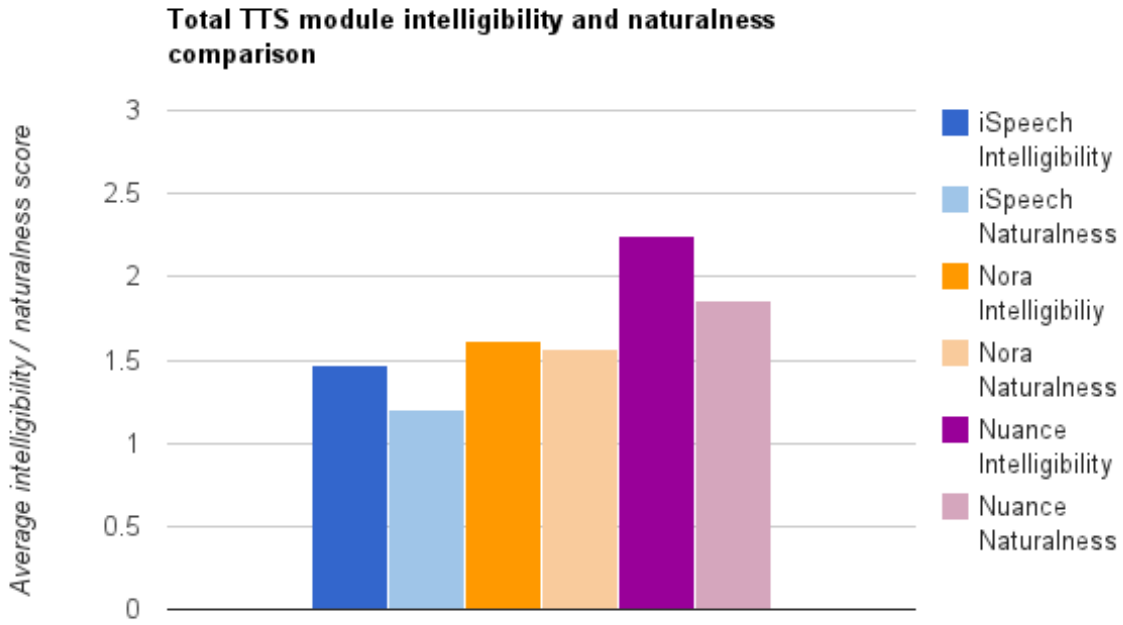

Figure 4.3: The total average score for each speech synthesis module in both low-noise and high-noise environments. The graph shows the average intelligibility and naturalness score for the three speech synthesis modules iSpeech, Nora and Nuance. The y-axis depicts the scores ranging from "Unintelligible/Unnatural" (0) to "Fully intelligible/Natural" (3) (higher is better).

4.2 Pronunciation Optimization

This is a comparison between pronunciations of the bus stop name *Dragvoll* before and after using the user dictionary editor to edit the user lexicon as described in Section 3.6. The Nora Client module was used with the following text string (it was made clear to the test subjects that "best" means "most intelligible and natural" before testing):

Hva høres best ut?.. dragvoll.. Eller.. Dragvoll..

Translated into English:
What sounds the best? dragvoll or Dragvoll?

The bus stop name "dragvoll" has the default pronunciation of the bus stop name. "Dragvoll", with a capital "D", has been altered by editing the user lexicon using the user dictionary editor. Figure 4.4 shows which one of the two pronunciation the test subjects thought was most intelligible and natural.

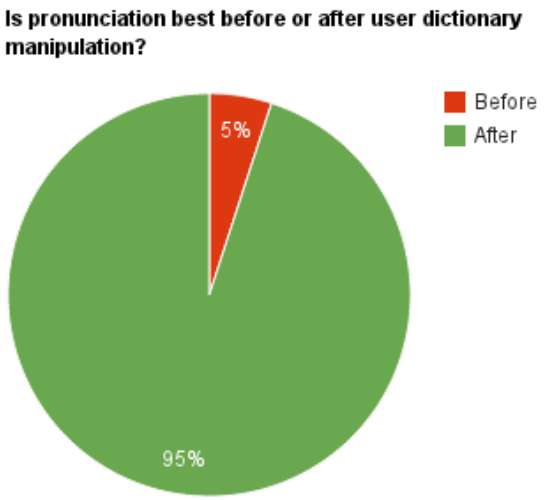

Figure 4.4: The result of making use of the User Dictionary Editor on a bus stop name. The red part is the percentage of people who think the default pronunciation was most intelligible and natural, while the green part is the percentage of people who think the altered pronunciation was most intelligible and natural.

4.3 Response Time

This section displays the results from the response time comparison between the TTS modules. The comparison consisted of measuring how long it took from the text string was sent until playback of the received audio clip was initiated. The *WiFi* test was performed close to a DLINK DIR-825[2] access point, giving Internet access through a 2.4GHz WiFi-network with 70Mbit/10Mbit bandwidth delivered from Canal Digital[3]. The *Mobile network* test was performed outdoors connected to a mobile network (HSDPA[4]) provided by Netcom[5], giving a maximum theoretical bandwidth of 14Mbit. The test alternated between the modules to ensure that the modules operated under the same conditions. The text string for input was an imagined, typical bus route response:

Du kan ta bussen fra Munkegata til Lade Kirke klokken 16.07.

Figure 4.5 shows the average response time for each speech synthesis module.

The next test consists of playing back an arbitrary sound clip from each of the speech synthesis modules with their corresponding average response time as measured in the previous test. The test supervisor held the smartphone up so that the test subject could see when the button that starts the speech synthesis was tapped. The test subjects waited for the sound to be played and were asked if the response time was acceptable. If they answered "Yes", it meant that the response time would not stop them from using the text-to-speech functionality. If the answer was "No", it could make them not use it or at least be unsure. Figure 4.6 shows the response time acceptance for each speech synthesis module.

[2] `http://www.dlink.com/DIR-825`
[3] `http://www.canaldigital.no/`
[4] `http://en.wikipedia.org/wiki/HSDPA`
[5] `https://netcom.no/mobilt-bredband-oversikt-privat`

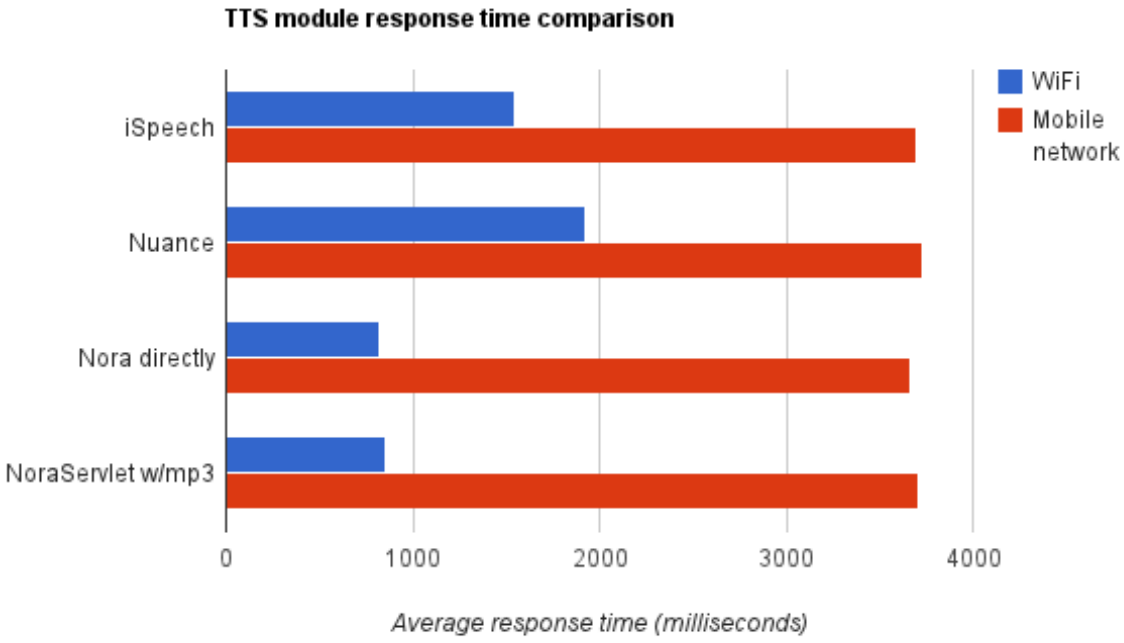

Figure 4.5: The average response time for each speech synthesis module with either a WiFi or mobile network connection. The numbers represent the time it takes from the user requests text-to-speech until the sound clips are ready to be played back on the smartphone (lower is better).

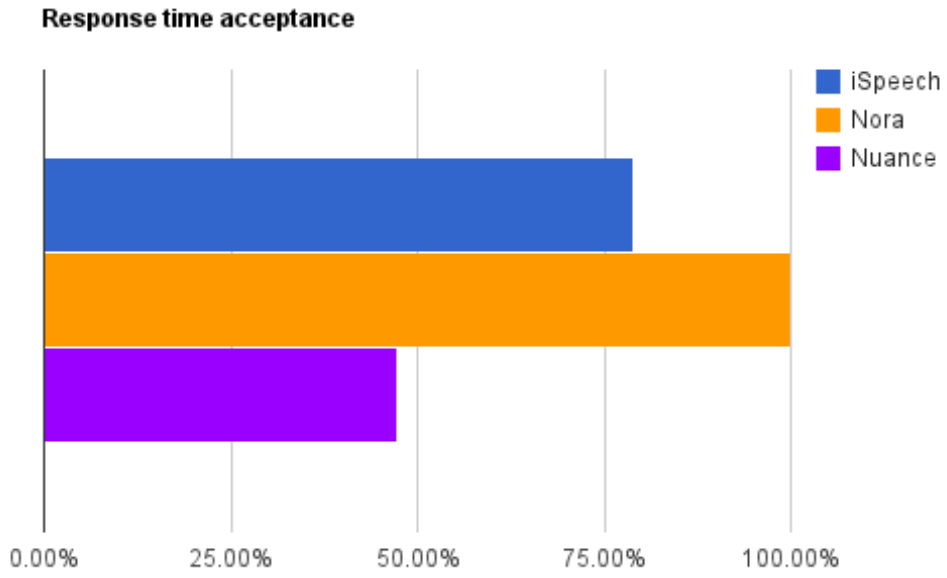

Figure 4.6: The number of test subjects that think the response time for each of the speech synthesis modules (Nuance, iSpeech and the Nora Client) are acceptable (higher is better).

4.4 Uptime

During all the tests in this thesis, both iSpeech and Nuance provided a 100 % uptime. The server providing Nora with synthesis, on the other hand, had to be restarted several times manually.

4.5 Data Transfer

This section describes the amount of data transfered to the smartphone from the different speech synthesis modules. The measurements were done using the Android Application *Traffic Monitor*[6]. The Nora Servlet is set to compress the audio data to MP3 (using the LAME binary), with its quality setting set to produce the smallest file size (see Section 3.1.3). The text used as input for this comparison is a typical answer from BusTUC:

Question:

Når går bussen fra Samfundet til Lade?

In English: *When does the bus leave from Samfundet to Lade?*

Answer:

Buss 94 passerer Studentersamfundet kl. 0527 og kommer til Lade allé 80, 14 minutter senere.

In English: *Bus 94 passes by Studentersamfundet at 5:27 am and arrives at Lade allé 80, 14 minutes later.*

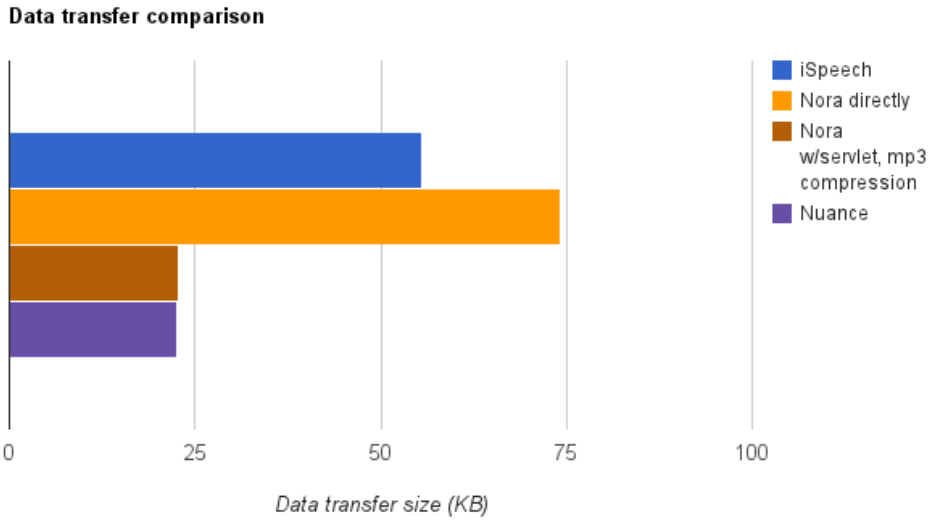

Figure 4.7: The data traffic between each specific speech synthesis module and the smartphone for a typical BusTUC answer (lower is better).

[6] https://play.google.com/store/apps/details?id=com.radioopt.widget

4.6 The New BusTUC Web Page

This section shows images of the new web page created for BusTUC. Figure 4.8 presents the web page design. The frames have been removed and the web page has the possibility to present the BusTUC answers in both textual and JSON format. The web page was tested in Google Chrome (version 19.0.1084.56) and Mozilla Firefox (version 11.0). Figure 4.9 presents the web page giving a textual response to a bus route query. Figure 4.10 presents the web page giving a response in form of JSON to a bus route query.

Figure 4.8: The new web page designed for BusTUC.

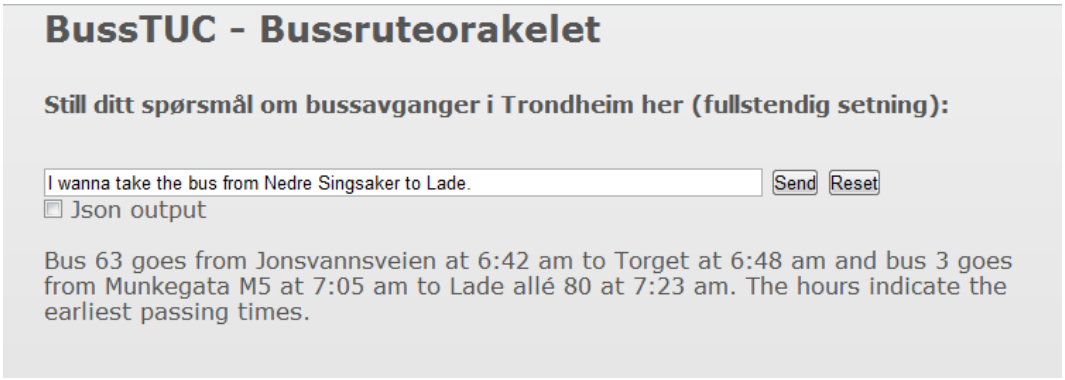

Figure 4.9: The textual response from the new BusTUC web page.

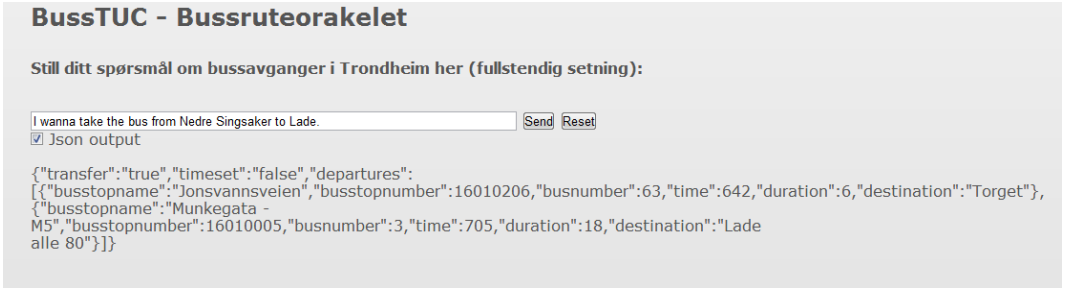

Figure 4.10: The JSON response from the new BusTUC web page.

Chapter 5

Discussion and Conclusion

This chapter will discuss the results. Then a summarized discussion reflects on how the goals and research questions were answered. Finally, conclusions are drawn.

5.1 Tests

This section will discuss the results from Chapter 4. Some of the conclusions in the following texts have been based on the comments the test subjects gave during the tests. These provided a lot of interesting feedback. Maybe even more so than the tests themselves.

5.1.1 Speech Synthesis Module Evaluation

It can be seen in all the graphs that the naturalness scores are consistently lower than the intelligibility scores. This gives reasons to believe that synthesized speech is not required to be natural in order to be understood. But is natural speech then needed? Many of the test subjects said that they thought Nuance was "best", as its speech synthesis sounds more natural and human-like. Others stated that Nora was most natural and consequently best. There was a general strong consensus, though, that iSpeech was the "worst", since it sounded more fragmented, lacked flow and had a "robotlike" quality to it. These responses suggest that a high level of naturalness is required for synthesized speech in order to achieve a good user experience.

Both the intelligibility and naturalness categories in this test have four score alternatives (0,1,2,3) for a bus stop utterance. Given that a test subject has picked a score for intelligibility for an arbitrary bus stop name utterance, there is a 1 in 4 chance that the test subject will pick the same score for naturalness. If

the test subjects were to pick scores randomly, 75 of 300 bus stop name utterances would be given the same score. In this test, 140 of 300 (~47 %) bus stop names utterances made the test subjects pick the same score for intelligibility and naturalness. This is a strong indication that there is at least a link between intelligibility and naturalness in synthesized speech.

One test subject thought all the modules were natural sounding in general, even though the test subject did not understand everything. The test subjected stated that *"Some pronunciations sounded more like strong foreign accent than unnatural."*. This gives reasons to believe that people have very different definitions of the word "natural", which has probably had an unfortunate effect on the test, giving false data.

Looking at the *total* (which summarizes the low-noise and high-noise graphs) results, it is apparent that Nuance is better than iSpeech and Nora both in sense of intelligibility and naturalness. It has high scores for both categories, while Nora and iSpeech end up with "neutral" scores. Several of the test subjects said that they had trouble understanding the dialect of Nora. Some even said the Nora dialect was annoying. It would therefore be reasonable to assume that this had a negative impact for Nora on the tests. It also indicates that TTS should have a more neutral dialect in order to reach out to a larger audience. One reason why iSpeech does not score well could be because it fails to pronounce Norwegian abbreviations correctly. By observation, it always pronounces each letter by itself. This is really bad for bus stop names that often have abbreviations like "St. Olavs Hospital". Another good reason for Nuance's good scores may simply be because the technology is newer. The Nora speech engine was released in 2002 and has been discontinued, while Nuance's TTS solutions today are still being updated[1]. iSpeech also continue to update their TTS software, but is still a bit behind the two other modules in the sense of intelligibility and naturalness.

The test performed in a *high-noise* environment shows that all of the TTS modules suffer from noise, especially iSpeech. iSpeech lost ~0.54 points of intelligibility and ~0.34 points of naturalness, while Nora only lost ~0.14 and ~0.18 compared to the low-noise test. Also, the gap between intelligibility and naturalness for iSpeech seems to dissipate. This probably means that the scores were so low that it was hard for the test subjects to distinguish between the two categories.

It is interesting to see that Nora only performs marginally worse in high-noise environments. The reason for this might be the speed at which words are uttered. A couple of the test subjects commented that Nuance and iSpeech sometimes were a bit harder to understand than Nora because they spoke too

[1] `http://www.nuance.com/for-business/by-product/nuance-vocalizer/index.htm`

quickly. The fact that Nora does not suffer in noisy environments could be linked to the fact that Nora speaks slower and is not that affected by sudden bursts of noise. Even if words are not heard in their entirety, hearing a few fragments of the words might give the brain the opportunity to fill in the blanks and complete the words.

When supervising these tests, it was apparent that some people are more susceptible to noise than others. One of the test subjects did not even notice that there were three different voices (modules) in the test. This could be due to hearing loss, lack of concentration or it could be because the overall quality of the TTS modules is too low to be understood. Human voices are the worst distraction, so if TTS functionality is used closed to large groups of people, headphones or hands-free devices are recommended.

Two test subjects pointed out that knowing the bus stop names prior to the test makes them much easier to understand. This exposes a weakness of this evaluation. All test subjects should be unfamiliar with Trondheim and its bus stop names to give valid results in further studies.

5.1.2 Pronunciation Optimization

Many of the test subjects mentioned that the synthesis modules had very strange pronunciations of bus stop names. One test subject actually failed at recognizing the name of the place where the test subject had lived earlier.

In the pronunciation optimization test, 95 % of test subjects thought that the altered pronunciation sounded more intelligible and natural than the default. It is safe to assume that having functionality to change pronunciation of words (like bus stop names) is very helpful in order to improve speech synthesis for a bus route information system.

5.1.3 Response Time

In the *WiFi* test, the Nora Client module and Nora Servlet module performed nearly equally well. With an average response time of 826 ms and 859 ms, they beat iSpeech (1549 ms) and Nuance (1923 ms) by quite a margin. The 33 ms difference between Nora Client and Nora Servlet indicates that the extra Java Servlet layer (with audio compression) leaves no significant negative impact on performance.

The Nora Client transfers the most data (see Section 5.1.4), but has the fastest response time of all the tested TTS modules. This could have many explanations. It could be because the cloud services of iSpeech and Nuance needs more time to process the request (either because the speech synthesis takes longer

or because the request is queued), but it is also possible that the difference is caused by Internet routing. The Nora TTS server is located in the same city as where the tests were conducted (Trondheim), while the servers for iSpeech and Nuance are situated abroad, in Germany and USA respectively. USA is further away from Trondheim than Germany and has a longer response time. It is therefore reasonable to assume that the time difference correlates with the distance between where the TTS servers are located and the smartphone. The routing theory is strengthened by the fact that iSpeech has a data transfer size that is twice the size of Nuance. It should therefore had had a slower response time due to the requirement of more bandwidth, but it does not.

In the *Mobile network* test, however, all the modules performed similarly worse. All modules had a response time of ~3700 ms. In environments without good WiFi access points nearby it does not matter for the user what module is used in respect to response time. Since the data sent is the same, it gives reasons to believe that the overhead is lost because of the low bandwidth and high latency from the mobile network access provided by the Internet Service Provider (ISP). The performance bottleneck is located in the mobile network connection between the ISP (Netcom) and the smartphone device.

In the *Response acceptance* test, the Nora Client resulted in having a 100 % acceptance rate. While test subject thought that Nora's response time was acceptable, ~79 % of the test subjects thought that iSpeech was acceptable and ~47 % thought that Nuance was acceptable. The reason why Nora comes out best in this test is that it has come under a "sweet spot" where perceived responsiveness does not make the user experience suffer. Over half of the test subjects thought Nuance was too slow, indicating that such a system would probably not gain popularity in the masses if implemented.

During the *response time acceptance* test an unexpected discovery was made. As the smartphone was held up in front of the test subject and speech synthesis through iSpeech was initiated, the test subjects noticed that iSpeech had a loading animation. Many of the test subjected stated that the loading animation makes longer waiting times bearable since you see that something is happening (i.e. being processed). Waloszek states: *"At the perceptual level, immediate feedback is mandatory to maintain the relationship of cause and action: To assure users that a command has been acknowledged"* [41]. Stimulating the users by using a loading animation or progress bar seems to be important for the overall user experience. This is confirmed by Myers, who performed experiments that indicate that progress bars are important and useful user-interface tools, and that they enhance the attractiveness and effectiveness of a programs [24]. The fact that only iSpeech had a loading animation is a weakness in the test and has most likely skewed the test results in iSpeech's favor. Further studies should be

conducted where the users cannot see the screen of the smartphone.

5.1.4 Data Transfer

The results of the data transfer test show that Nuance is far better than iSpeech and Nora when it comes to data transfer size. Nuance (22.59 KB) has about half the data size of iSpeech (55.45 KB) and one third of Nora (74.08 KB). This simply means that Nuance uses a more effective compression algorithm (codec) for its audio data or that it has a codec quality setting that focuses on producing the smallest data size possible rather than preserving audio quality. This could have a major negative impact on audio quality, but since Nuance topped the tests both in the sense of intelligibility and naturalness (and did not get any complaints about noise from the test subjects), audio quality seemed just fine. The Nora Servlet, that used MP3 compression functionality to reduce data size of the audio data, produced almost just as small data size (22.84 KB). This showcases that extra functionality such as audio compression can help reduce the bottleneck presented by the limited bandwidth and the communication cost on network connections that are charged on a per byte basis. It also shows that it is possible for Nora to compete with Nuance in the sense of data transfer size. This does not mean that the quality of audio is the same, though. This must be tested in another study (see Section 6.1.1).

5.1.5 The New BusTUC Web Page

No user-tests or evaluations have been conducted to determine if the new web page[2] is better than the old one[3], but it is not hard to see that the design is tidier and more streamlined. The new look and feel makes the web page more user-friendly and visually pleasing, and it is therefore better equipped to meet the demands of today's Internet users.

5.2 Discussion

Test subject comments like *"Norwegian text-to-speech is exciting!"* and *"Likes that the smartphone talks."* indicate that there is an interest in such functionality present in the public, which is encouraging for future development and use of TTS technology.

[2] `http://busstuc.idi.ntnu.no/`

[3] `http://www.idi.ntnu.no/~tagore/busstuc/`

It is hard to decide whether to use a cloud service or not for a TTS functionality. One must decide if the improved speech synthesis (like Nuance's) is worth the lack of control. By using in-house systems, like BUSTER, one can update and tweak the system. By integrating an intermediate Java Servlet to BUSTER, the system is more flexible since extra functionality (like audio compression algorithms, logging tools, user control and statistics, and security measures) can easily be added. Also, the Nora speech engine pronunciations can be edited to gain more intelligibility and naturalness. This feature is not available in the cloud services (iSpeech and Nuance), leaving the user helpless if, e.g., a bus stop name is pronounced in an especially incomprehensible way. Since the Nora Client is written in Java, it is only fit to be installed on Android devices. By installing the Nora Client on the Nora Servlet (that can handle HTTP GET requests), Nora's TTS service can be offered to multiple platforms (iOS, Symbian and basically all platforms that are equipped with an Internet browser) through web applications on the smartphone [2]. The TaleTUC: Text-to-Speech system is set up in a university which presents learning and research opportunities because you have an in-house system to look into.

Cloud providers (like iSpeech and Nuance) do not provide these opportunities due to their black box approach. In every specific case one must also make an assessment if it is more expensive to maintain the server infrastructure in-house or if it is cheaper to just leave it to the cloud providers. One should also investigate the cloud providers to illuminate any risks related to things such as reliability in sense of their future prognosis (so that one does not invest in companies that go bankrupt) and uptime, preferably by a neutral third party. It should be noted though, that the cloud providers in the tests of this thesis had a 100 % uptime, while the TTS server that Nora uses had to be restarted several times. The reason for the instability is that the server has not been tweaked and tested for commercial use. This is described further in Section 6.1.5.

Nuance's high performance in the *Speech Synthesis Module Evaluation* test would suggest that it is the optimal choice of TTS solution for use in TaleTUC: Text-to-Speech. However, it is very expensive. iSpeech, on the other hand, is free, but does not perform well. Nora, with its SAPI approach, is a good compromise. It does not have the same TTS quality as Nuance (but better than iSpeech), but it is free, comes with total control, superior response time and the possibility to improve bus stop name pronunciation. It is simply very well suited for the domain of bus route information.

5.3 Conclusion

Speech synthesis has recently emerged as a compelling service to aid in everyday tasks. Apple's release of *SIRI*[4] is a good example of a system that has contributed to the gained popularity. The increase in smartphone sales and the importance of digital information has already reached such an enormous level that many companies and even corporations provide information through smartphone applications. Service providers that want to reach out to as many users as possible need to create smartphone applications that satisfy people that do not fall into the "normal user" category. People that require non-visual feedback, such as visually impaired persons, need output in form of auditory signals.

Speech synthesis in Norwegian has matured enough to successfully provide bus route information. Cloud providers do have the potential to deliver good TTS services, but with the users' lack of control it is hard to tailor them to work well in this domain. Norwegian TTS still needs to improve in both intelligibility and naturalness. There are many names that are pronounced incorrectly and it has not reached its full potential. This means that opportunities still exist for researchers to make contributions in the field and bring significant impact to the development of such technology.

The following sections answer the research questions defined in the thesis:

5.3.1 Research Question 1

Is it preferable to put the entire system directly on the smartphone, or will a client-server system be more practical? A client-server system is not only more practical, but unavoidable. Having the entire TTS system on the smartphone is simply not feasible if concatenative synthesis (like Nora) is to be used. The recording data can take up hundreds of megabytes to gigabytes of space. The voice of Nora (SAPI speech engine) takes 700 MB of space and is too big to be stored on the smartphone. The client application would be too big to be published on any "app store" or "app market" (e.g. Google Play). It is therefore a requirement that the speech is synthesized on demand on a server and then sent to the smartphone.

Can cloud-based solutions potentially be used? Using a cloud-based solution can be viable for the domain of bus route information, but if a quality TTS is needed, it is very expensive. Also, since there are no means of control the customer is

[4] `http://www.apple.com/iphone/features/siri.html`

stuck with fixed pronunciations.

5.3.2 Research Question 2

What enhancements are likely to make a bus route information system more efficient? Several enhancements can be done to FUIROS in order for it to become more efficient and fulfill its vision of becoming the ultimate bus route information of the future. The visual and technical update to the BusTUC web page have made it more appealing and will hopefully make it go "viral". Standardizing systems by consequently using the UTF-8 character encoding will help to mitigate issues concerning different languages. This will help to remove incorrect character representation in situations where different systems send text between each other. Standardizing such textual communication with JSON will also help ease the development of current or new potential systems.

5.3.3 Research Question 3

Does intelligibility and naturalness in speech coincide? The test results hinted in the direction that intelligibility and naturalness do coincide, but since no common definition of naturalness was set prior to the tests, a final conclusion cannot be drawn.

Chapter 6

Future Work

6.1 Future Enhancements

The following sections describe the future work related to the TTS systems of the thesis.

6.1.1 Sound Compression

Tests should be conducted that evaluate which codec should be used for compression of speech synthesis audio data.

Both objective and subjective tests can be used to assess sound quality before and after compression. Subjective tests use human test subjects to measure how well the sound is perceived by the end-user, which is optimal. This is expensive and time consuming though, since many subjects are needed to get statistically significant results. PESQ is an alternative, objective way of making these assessments. It uses an algorithm to assess audio degradation that is highly comparable to subjective tests.

Granda et al. evaluate audio quality in a range of speech codecs using PESQ [9]. They show that speech encoded with low quality codecs is either unintelligible or requires great effort by the listener to listen to. On the other hand, speech encoded with medium or high quality codecs was easily understood and hardly distinguishable, so medium quality codecs, like Speex 5 (Speex with quality setting 5) is the best compromise between quality and resource requirements.

Speex is unfortunately not supported by Android to this date. The test could therefore be done in such a way that it e.g. measures the PESQ scores of different codecs (that are supported) at various quality settings and compare them to Speex 5 to determine which codec should be used and at which quality setting.

Figure 6.1 shows an imagined graph depicting the results.

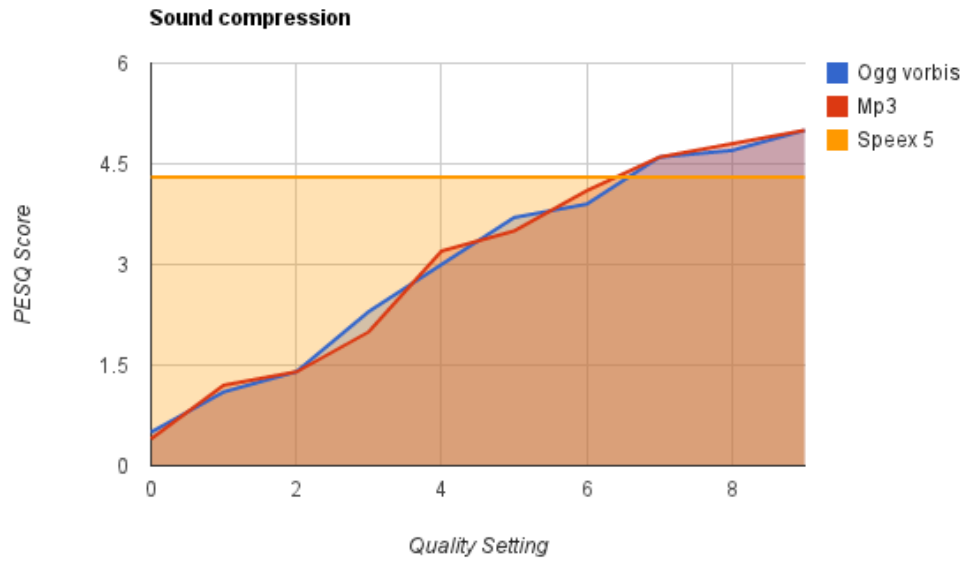

Figure 6.1: The Perceptual Evaluation of Sound Quality (PESQ) score for each quality setting of the audio codecs. Keep in mind that Speex 5 is at constant "5" quality and is consequently not affected by the x-axis.

6.1.2 Integrate with TABuss

The Nora Client should be integrated into the TABuss application and tested extensively by users to see if it is ready for commercial release. Since the Nora Client is a module that has been implemented for just this purpose, only a few lines of code is required for the application to start "talking".

6.1.3 Extending the Nora Servlet with more Functionality

The Nora Servlet should be implement functionality like logging, user control and statistics in order for the the maintainers of FUIROS to have an overview of how the system is being used. This could help them see trends of user behavior and plan future development to meet the demands of these trends accordingly.

6.1.4 Merging Servers

Currently, the BUSTER system is spread over several servers (see Section 3.1). It is desirable to move both the *main program* and *dialog server* to Orwell (the Windows TTS server that hosts the TTS module) so that only one server needs to be maintained.This should not provide many challenges since all the code is written in Python or Prolog, which works on Windows platforms. The only requirements are the availability of Sicstus Prolog and sockets programming for the platform.

6.1.5 Extensive Testing

It is vital that the stability of the Orwell (the server that hosts the TTS module in BUSTER) improves in order for it to be used for commercially available products. Reliability is key for a successful end-user application. Once in a while, during the development of the server, errors occurred that could not be replicated and therefore not fixed. With many systems working together it is not always easy to pinpoint the exact origin of an error. Therefore it would be beneficial to conduct extensive testing of the server to try to get rid of as many bugs as possible. A proper load test should also be conducted to estimate what hardware specifications is needed in relation to concurrent queries for synthesized speech. Extensive testing should also be conducted on the Nora Client prototype in order to reveal weaknesses and fix them.

6.1.6 Improve Bus Stop Pronounciation

The pronunciation of active bus stop's names in Trondheim should be assessed. Those that have low intelligibility and/or naturalness should be edited with the User Dictionary Editor (see Section 3.6) in making them more intelligible and natural.

6.1.7 Update To New SAPI Speech Engine

The Nora speech engine is beginning to get quite old (released 2002) and at one point it should be updated to achieve even higher levels of intelligibility and naturalness. There exists newer[1] speech engines in Norwegian for SAPI. The problem with installing these is that they require newer versions of SAPI to function (the current version of SAPI is 5.4). SAPI5 was a complete redesign from previous versions and neither engines nor applications which used older versions of SAPI could use the new version without considerable modification. Installing SAPI5 will, in turn, require an update to the server program (that provides the synthesized speech) on the TTS server (BUSTER). There should be made an effort in finding a speech engine that speaks in a more neutral dialect than Nora (like the eastern-Norwegian dialect that Nuance uses) in order to satisfy a bigger audience. Also, when installing these new speech engines, the documentation for these should be checked so no copyright claims are violated if TaleTUC: Text-to-Speech should ever be commercialized.

6.1.8 Nora Parameters

The Nora speech engine has settings (pitch, speed (words/min) and volume) that tailor the synthesized speech output of the TTS server. Currently, these must be set prior to server startup. Preferably, these could be set on every speech synthesis request. Also, the Nora Servlet should be able to switch between file formats and compression codecs and quality. All of these things can be achieved by reworking the code for the TTS server and Nora Servlet such that they take input parameters along with the text to be synthesized.

6.1.9 BusTUC Tram

The BusTUC system has fundamental functionality to provide *tram* route information in Trondheim. The Prolog code for this functionality is not up to date and does not to this date function together with the *bus* route information parts

[1] E.g. Hulda: `http://www.microsoft.com/en-us/download/details.aspx?id=3971`

of the system. This issue should be investigated and fixed so tram users may make use of this service.

6.2 FUIROS and FUIROS Related Technologies

The following sections describe the future work for FUIROS in general, TABuss, MultiBRIS and BusTUC.

6.2.1 Geographical Expansion of FUIROS and Standards

An idea for future work for FUIROS is to add support for other cities in Norway, and use a single system to provide public transportation information for the entire country. Then, a single client application could access route information based on the mobile device's location.

A challenge for an effective expansion is the need for standards. It would aid the development if all of the bus agencies in Norway used the same standards for sharing routes and real-time data. Norway's largest bus agency, Trafikanten AS[2], already uses such a standard. This standard, which is called SIRI[3], is used for the distribution of real-time data. It is an XML protocol that allows distributed computers to exchange real-time information about public transport services and vehicles.

Through a JSON-API Trafikanten AS has made the *StopMonitoring* part of SIRI available for public use. The Stop Monitoring section is described by the SIRI standard as follows:

> *The Stop Timetable (ST) and Stop Monitoring services (SM) provide stop-centric information about current and forthcoming vehicle arrivals and departures at a nominated stop or Monitoring Point, typically for departures within the next 20-60 minutes for display to the public. The SM service is suited in particular for providing departure boards on all forms of device*[4].

SIRI is already in use by Trafikanten and therefore represents a good example of what could be a national standard for sharing real-time public transport information. For the notion of a single system, implementing the Siri standard could be the first step towards achieving this.

[2] `www.ruter.no`

[3] `http://www.kizoom.com/standards/siri/`

[4] `http://www.kizoom.com/standards/siri/documentation.htm`

However, the major challenge for such standards is probably not technical, but rather political and financial. An approach that avoids the distributed standardization challenge could be constructed by absorbing the existing transportation agency systems one-by-one. This system would effectually become the mediation layer that creates the standard, seen from an application developer's point-of-view. This would also increase the amount of work needed to expand the system substantially, compared to expanding a system based on standards. The advantage of this approach would be that such a system could establish a position of power in relation to public transport data sharing. It is reasonable to believe that a system that has a standard way to communicate route data for an entire country would become vastly popular in the development community. By providing the "back-end" to "front-end" mediation layer for the majority of available public transportation client-applications, one would be in a position of power. This is an advantage that could be used to encourage the use of standards such as SIRI, among the public transportation agencies.

6.2.2 TABuss

The following sections first identify possibilities with the new smartphone technologies. Future work involving context-awareness is then described. Finally, suggestions for future extensions to TABuss are provided.

Updated Version on Google Play

With the use of TABuss as a TaleTUC client, an updated version of TABuss has to be uploaded to Google Play. This should optimally be put on hold until TaleTUC supports all the bus stops in Trondheim. User testing of the current prototype can be useful, however, and an imminent release could therefore be considered.

New Smartphone Technologies

Based on the experiences of Marcussen and Andersstuen with developing the *TaleTUC: ASR* client widget, a widget could be created for TABuss [20, 21]. This could provide information such as real-time passings of buses for the closest bus stop to the user's location. Touch events could trigger the widget itself to provide som information, or trigger the start-up of TABuss.

Another interesting field is Near Field Communication (NFC) [43] and the use of this technology in mobile applications [34]. A usage in TABuss could be

to detect Radio Frequency Identification (RFID) [25] tags that have been integrated into every bus stop. When the user is close enough to a bus stop, the application could trigger the display of the next passing buses. RFID tags integrated into bus stops could also be used for speech synthesis purposes. Visually impaired people could benefit from a functionality where the system reads out loud the next passing buses, when they approach a bus stop.

A new implementation that involves AtB, is the purchase of bus tickets. It is possible to buy tickets through a service provided by AtB, by sending a text message to 2027 (Norwegian number) and specifying the type of ticket (adult, child, military, etc). This sending of a text message could be triggered by the user approaching the bus stop. It should be integrated into already existing functionalities, to avoid unnecessary sending of text messages. An example is when the user has performed a query to MultiBRIS and has received route suggestions. The user could then select the suggestion he or she wants to use, an action that alerts the RFID reader to start the SMS service when the user approaches the selected departure bus stop.

Context-Awareness

An extension involving context-awareness for TABuss is to use more sensors than only the location sensor, which has been done by Raento et al. [30]. Their system uses four sensors: location, user interaction, communication behaviour and physical environment. This means that besides location information, their system monitors: what actions the user performs, calls, text messages and surrounding devices.

For TABuss, this sensor information could be used to introduce context-awareness to the user interface. The age differences between potential target users is large, and an adaptive user interface could be a solution. The user interface could through sensors track the user's actions, register some trends and then adjust visibility and availability accordingly. An example is to track the usage of the Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) modules. If they are often used features, access to them could be made quicker.

The tracking of user trends could also be used to perfect route suggestions. People of different ages have different levels of mobility and have different walking speeds. This has been addressed by Vieira et al. [40] in their proposed system *UbiBus*. *UbiBus* considers different people's and vehicle's mobility, and other factors that can affect a bus departure. An interesting idea is for AtB to contribute to such functionalities in order to improve route suggestions. Buses have installed cameras could be used to monitor how crowded a bus is. This could prove beneficial for handicapped people, or people with small children, who need seats or at least clear floor area.

Another suggestion is to use context through calendar information, by monitoring scheduled appointments. When an appointment is approaching, the user could be prompted with a query suggestion. Khalil and Connelly [16] stated that it is an inevitable fact that people's actions not always mirror their intentions. Even though an appointment has been scheduled, the user is not guaranteed to attend. TABuss queries should therefore not be run automatically in this case, only a query suggestion should be prompted. Automatic query runs could cause unnecessary data traffic when the user has chosen not to attend a scheduled appointment, or has chosen another form of transportation.

A challenge with introducing context-aware extensions is privacy. If such information is to be stored on a server, a secure login mechanism is necessary. Marcussen and Andersstuen, in their TaleTUC ASR system, uses the device IDs of the smartphones to separate users, which is a sufficient solution when non-sensitive data is stored [20]. To introduce the factors proposed by Raento et al. [30] will either require that these factors are stored on the device, or that a secure storage functionality is created server-side. With server-side storage and a login mechanism, it is important to minimize the user requirements. Users may reject an application that requires too many involvements, when they want to get a quick route suggestion.

Future Extensions of TABuss

A future extension could be to integrate TABuss into a tourist application. The *Trondheim Guide*[5] is an intelligent travel guide which already provides some bus route information. This information is limited, and no information on arrival/departure times was found during testing. Another alternative is *City Explorer*[6], which is a framework for city exploration. In relation to TaleTUC, tourist information could be a domain to extend the ASR functionality to cover. TABuss as a TaleTUC client with integrated *City Explorer* functionalites could then use this.

6.2.3 MultiBRIS

Flinn et al. [8] developed a system that dynamically decides whether to perform server-side or client-side computations. These decisions are based on monitored resource usage both on the server and the client. This functionality could be implemented for MultiBRIS, and prevent delays when the MultiBRIS server is

[5] `www.trondheim.no/app`

[6] `http://www.sintef.no/Projectweb/UbiCompForAll/Results/Software/City-Explorer/`

busy, which can be caused by a high traffic load. In those situations, the clients should do the necessary operations instead of relying on MultiBRIS. Clients such as TABuss must have functionality that calculates route suggestions, and allow for queries to be sent to BusTUC and AtB's real-time system. This puts extra computational pressure on the client, but facilitates a solution that can provide route suggestions with and without the involvement of MultiBRIS.

Flinn et al. [8] also describe the idea to let the client learn what is best practise in the different situations, when taking into account factors such as low battery power. The client could monitor the resource usage for performed operations over time, and learn which tasks to compute client-side and which to compute server-side. Experiences gained after each operation could be stored as cases, and a Case-based reasoning (CBR) functionality could be used for retrieval [20].

6.2.4 BusTUC

Future work on BusTUC includes the research of other intelligent route information solutions. Because BusTUC is the only available candidate in Trondheim, there are no systems to compare it with. One specific task would be to do research on similar systems found outside of Trondheim, and develop comparable prototypes. If research shows that BusTUC is the best solution, a goal could be to establish it as a standard for bus route information in Norway. This standard, together with the SIRI standard, could then be two of the building blocks of a common standard for the exchange of transportation information.

Another option is to expand BusTUC and the concept of a natural language route information system outside of Trondheim to cities of different sizes and number of inhabitants.

Bibliography

[1] T. Amble. BusTUC: a natural language bus route oracle. In *Proceedings of the sixth conference on Applied natural language processing*, pages 1–6. Association for Computational Linguistics, 2000.

[2] Runar Andersstuen and Trond Engell. Multibris: A Multiple-platform approach to the Ultimate Bus Route Information System for Mobile Devices. Technical report, Department of Computer and Information Science, NTNU, December 2011.

[3] Runar Andersstuen, Trond Engell, Rune Sætre, and Björn Gambäck. A Multiple Platform Approach to Building a Bus Route Information System for Mobile Devices. In *12th International Conference on Innovative Internet Community Systems*, June 2012.

[4] Ken Arnold, James Gosling, and David Holmes. *The Java Programming Language*. Pearson, 4rd edition, 2005. ISBN 0321349806. URL `http://www.worldcat.org/isbn/8177587722`.

[5] B.J. Brodkin. Gartner: Seven cloud-computing security risks. *Infoworld*, pages 2–3, 2008.

[6] Charles W. Brokish and Michele Lewis. A-law and mu-law companding implementations using the tms320c54x. *Infoworld*, 1997. URL `http://www.eettaiwan.com/ARTICLES/2001MAY/PDF1/2001MAY02_NTEK_DSP_AN1135.PDF`.

[7] D.J. Calder and D. Phil. The development of synthesized speech systems for the vocally handicapped. In *Singapore ICCS/ISITA'92.'Communications on the Move'*, pages 442–446. IEEE, 1992.

[8] Jason Flinn, Soyoung Park, and M. Satyanarayanan. Balancing performance, energy, and quality in pervasive computing. In *Proceedings of the 22nd International Conference on. Distributed Computing Systems*, pages 217–226, 2002.

[9] J.C. Granda, J. Quiroga, D.F. Garcia, and F.J. Suarez. Quality assessment of speech codecs in synchronous e-learning environments. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[10] Erik Harborg. Demonstrators in the BRAGE project. Technical report, SINTEF, 2007.

[11] O. Hartvigsen, E. Harborg, T. Amble, and M.H. Johnsen. Marvina - A Norwegian Speech Centric, Multimodal Visitors Guide. In *NODALIDA 2007 Proceedings (The 16th Nordic Conference of Computational Linguistics)*, 2007.

[12] Brian Hayes. Cloud computing. *Communications of the ACM*, 51(7), 2008.

[13] Jay Heiser and Mark Nicolett. Assessing the security risks of cloud computing. 2008. URL `http://www.gartner.com/DisplayDocument?id=685308`.

[14] Magne Hallstein Johnsen, Torbjørn Svendsen, Tore Amble, Trym Holter, and Erik Harborg. TABOR - a Norwegian spoken dialogue system for bus travel information. In *INTERSPEECH*, pages 1049–1052. ISCA, 2000. URL `http://dblp.uni-trier.de/db/conf/interspeech/interspeech2000.html#JohnsenSAHH00`.

[15] M.H. Johnsen, T. Amble, and E. Harborg. A Norwegian Spoken Dialogue System for Bus Travel Information. *Telektronikk (2)*, 2003.

[16] Ashraf Khalil and Kay Connelly. Improving cell phone awareness by using calendar information. In Maria Costabile and Fabio Paternò, editors, *Human–Computer Interaction – INTERACT 2005*, volume 3585 of *Lecture Notes in Computer Science*, pages 588–600. Springer Berlin / Heidelberg, 2005. ISBN 978–3–540–28943–2.

[17] J.E. Knudsen, FT Johansen, J. Rugelbak, and ND Warakagoda. Tabulib 1.4 reference manual. *Telenor R&D*, (36):2000, 2000.

[18] LF Lamel, JL Gauvain, B. Prouts, C. Bouhier, and R. Boesch. Generation and synthesis of broadcast messages. In *Proc. ESCA-NATO Workshop on Applications of Speech Technology*, pages 207–210, 1993.

[19] J.C. Lee, M.S. Hahn, H.S. Lee, J.W. Yang, and Y. Lee. Text-to speech conversion system for synchronizing between synthesized speech and a moving picture in a multimedia environment and a method of the same, August 23 2011. US Patent RE42,647.

[20] Christoffer Marcussen and Runar Andersstuen. Speech-to-text for bus route information systems. Master's thesis, Department of Computer and Information Science, NTNU, June 2012.

[21] Christoffer Marcussen and Lars Moland Eliassen. TABuss: An Intelligent Smartphone Application. Technical report, Department of Computer and Information Science, NTNU, December 2011.

[22] Christoffer Marcussen, Lars Eliassen, Rune Sætre, and Björn Gambäck. Context-Awareness and Real-Time Information in an Intelligent Smartphone Application. In *12th International Conference on Innovative Internet Community Systems*, June 2012.

[23] Peter Mell and Timothy Grance. The NIST definition of cloud computing. `http://www.nist.gov/itl/cloud/index.cfm`, September 2011.

[24] Brad A. Myers. The importance of percent-done progress indicators for computer-human interfaces. *SIGCHI Bull.*, 16(4):11–17, April 1985. ISSN 0736-6906. doi: 10.1145/1165385.317459. URL `http://doi.acm.org/10.1145/1165385.317459`.

[25] E.W.T. Ngai, Karen K.L. Moon, Frederick J. Riggins, and Candace Y Yi. RFID research: An academic literature review (1995-2005) and future research directions. *International Journal of Production Economics*, 112(2): 510 – 520, 2008. ISSN 0925-5273. URL `http://www.sciencedirect.com/science/article/pii/S0925527307001934`.

[26] J. Odell, D. Kershaw, D. Ollason, V. Valtchev, and D. Whitehouse. The HAPI Book V1. 4. *Entropic Ltd., Jan*, 1999.

[27] Bruce W. Perry. *Java Servlet & JSP Cookbook*. O'Reilly Media, 1st edition, December 2003. ISBN 0596005725. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0596005725`.

[28] Z. Qiao, L. Sun, and E. Ifeachor. Case study of PESQ performance in live wireless mobile voip environment. In *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*, pages 1–6. IEEE, 2008.

[29] Magnus Raaum. An intelligent smartphone application. Master's thesis, NTNU, 2010.

[30] Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen. Contextphone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, 4:51–59, 2005. ISSN 1536-1268. doi: http://doi.ieeecomputersociety.org/10.1109/MPRV.2005.29.

[31] A. Raux and M. Eskenazi. Non-native users in the Let's go!! spoken dialogue system: Dealing with linguistic mismatch. In *Proceedings of HLT-NAACL*, volume 4, 2004.

[32] A. Raux, B. Langner, D. Bohus, A.W. Black, and M. Eskenazi. Let's go public! taking a spoken dialog system to the real world. In *Ninth European Conference on Speech Communication and Technology*, 2005.

[33] A.W. Rix, J.G. Beerends, M.P. Hollier, and A.P. Hekstra. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 2, pages 749–752. IEEE, 2001.

[34] Miguel Sánchez, Montserrat Mateos, Juan Fraile, and David Pizarro. Touch Me: A New and Easier Way for Accessibility Using Smartphones and NFC. In Javier Bajo Pérez, Miguel A. Sánchez, Philippe Mathieu, Juan M. Corchado Rodríguez, Emmanuel Adam, Alfonso Ortega, María N. Moreno, Elena Navarro, Benjamin Hirsch, Henrique Lopes-Cardoso, and Vicente Julián, editors, *Highlights on Practical Applications of Agents and Multi-Agent Systems*, volume 156 of *Advances in Intelligent and Soft Computing*, pages 307–314. Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-28761-9.

[35] M. Schröder. Expressive speech synthesis: Past, present, and possible futures. *Affective information processing*, 2009.

[36] H. Shi and A. Maier. Speech-enabled windows application using Microsoft SAPI. *International Journal of Computer Science and Network Security*, 6(9):33–37, 2006.

[37] W.P. Tanner. The measurement of speech intelligibility. Technical report, Michigan University, 1970.

[38] P.J. Tritton. Automatic translation of spanish text to phonetics: Using letter-to-sound rules. *Hispania*, 74(2):478–480, 1991.

[39] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, pages 50–55, 2009.

[40] Vaninha Vieira, Luiz Rodrigo Caldas, and Ana Carolina Salgado. Towards an ubiquitous and context sensitive public transportation system. *International Conference on Ubi-Media Computing*, 0:174–179, 2011. doi: http://doi.ieeecomputersociety.org/10.1109/U-MEDIA.2011.19.

[41] G. Waloszek. Waiting at the computer: Busy indicators and system feedback–part 1. *SAP User Experience, SAP AG*, 2008.

[42] J. Yang, J. Xiao, and M. Ritter. Automatic selection of visemes for image-based visual speech synthesis. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 2, pages 1081–1084. IEEE, 2000.

[43] J. Ylinen, M. Koskela, L. Iso-Anttila, and P. Loula. Near field communication network services. In *Digital Society, 2009. ICDS '09. Third International Conference on*, pages 89 –93, feb. 2009. doi: 10.1109/ICDS.2009.43.

[44] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1): 7–18, 2010.