University POLITEHNICA of Bucharest

Faculty of Automatic Control and Computers,
Computer Science and Engineering Department

BACHELOR THESIS

Human-Robot Interaction

Scientific Advisers:
Prof. dr. ing. Țăpuș Nicolae
Prof. dr. ing. Țăpuș Adriana

Author:
Ciocîran Ștefan-Dan

Bucharest, 2017

acknowledgements

acknowledgements

Abstract

The Abstract

Contents

List of Figures

List of Tables

Notations and Abbreviations

CS – Computer Science
UPB – University Politehnica of Bucharest

Chapter 1

Introduction

This is just a demo file. It should not be used as a sample for a thesis.

TODO:
Remove this line (this is a TODO)

1.1 Project Description

1.1.1 Project Scope

This thesis presents the **MySuperProject**.

This is an example of a footnote [2]. You can see here a reference to Section 1.1.2.

Here we have defined the CS abbreviation. and the UPB abbreviation.

The main scope of this project is to qualify xLuna for use in critical systems.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristiqu dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristiqu dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

1.1.2 Project Objectives

We have now included Figure 1.1.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristiqu dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales

[2] www.google.com

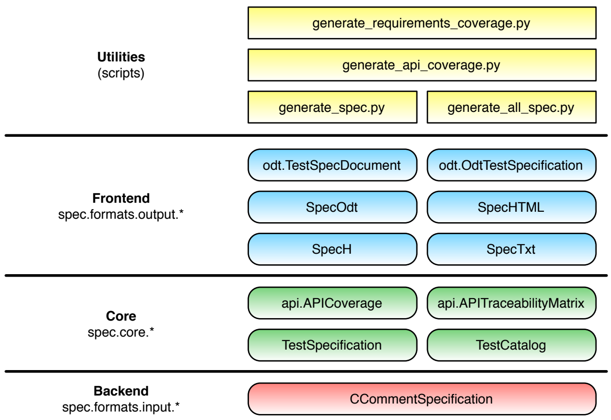

Figure 1.1: Reporting Framework

pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

We can also have citations like [6].

1.1.3 Related Work

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristiqu dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristiqu dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquam lectus vel orci malesuada accumsan. Sed lacinia egestas tortor, eget tristiqu dolor congue sit amet. Curabitur ut nisl a nisi consequat mollis sit amet quis nisl. Vestibulum hendrerit velit at odio sodales pretium. Nam quis tortor sed ante varius sodales. Etiam lacus arcu, placerat sed laoreet a, facilisis sed nunc. Nam gravida fringilla ligula, eu congue lorem feugiat eu.

We are now discussing the **Ultimate answer to all knowledge**. This line is particularly important it also adds an index entry for *Ultimate answer to all knowledge*.

1.1.4 Demo listings

We can also include listings like the following:

```
CSRCS = app.c
SRC_DIR =..
include $(SRC_DIR)/config/application.cfg
```

Listing 1.1: Application Makefile

Listings can also be referenced. References don't have to include chapter/table/figure numbers... so we can have hyperlinks like this.

1.1.5 Tables

We can also have tables... like Table 1.1.

Table 1.1: Generated reports - associated Makefile targets and scripts

Generated report	Makefile target	Script
Full Test Specification	full_spec	generate_all_spec.py
Test Report	test_report	generate_report.py
Requirements Coverage	requirements_coverage	generate_requirements_coverage.py
API Coverage	api_coverage	generate_api_coverage.py

Chapter 2

Basic Concepts

In the previous chapter I presented a short introduction of the ENRICHME project and subjects of work. Before I present a more detailed description of the main goals and the implementation of them I would like to discuss about the basic conecepts of working with robots and of the tehnologies and programing languages used.

2.1 Theoretical Background

To begin with, I will make a description of the theory behind my work. This includes presenting the robot, the protocols used for real time comunication, taking news from the news sites and some security basic browser security.

2.1.1 Heart Rate

2.1.2 WebRTC

WebRTC (Web Real-Time Communication) [1] is a open-source project that does not need plugins to be used on browsers. It has a collection of communications protocols and APIs (application programming interfaces) that enable real-time communication between peers. That are two distinct APIs C++ API (fore browser developers) and Web API (for web developers, javascript API). I used the second one because is more portable between diferent types of systems. Even if WebRTC allow you to do file transfer, chat or desktop sharing in my project is necesarly only the video coferencing feature used. For this to work you need a STUN server [7] and for some networks a TURN server [8]. The tehnology used for creating the connexion peer-to-peer is represented by the ICE [11]. For transmiting the audio and video data webRTC use RTP (Real-Time Transport Portocol).

2.1.3 RSS 2.0 Feed

RSS 2.0 (Really Simple Syndication) [2] is know as a tpe of web feed [4]. A web feed (news feed) is a data format to provide informations which frequently change their contents. It create a plublisher-subscribers system.The publisher create a web feed and the users subscribe to it. Mostly the rss feed is used for news media to send micro-news to their customers. with this tehnology you can create a news agregator where you have multiple rss feeds from different sites and you filter them for your users. They can see a short description to the news, and after that

they can have the option to go to the entire news on the parent site.
There are two types of web feed most used: rss feed and atom feed. I used rss feed because is implemented by most of the news sites and I wanted a big base of news. Because of the browsers same origin policy (Section 2.1.4) I had to use an API that transform rss feed in jsonp.

2.1.4 Same Origin Policy

Same Origin Policy [9] is a concept used in web application security model. In browsers which respect this policy you are not allowed to use data in scripts from pages that do not have the same origin. This policy must be respect so you will not have malicious code running in your browser or your webpage. Origin is defined as concatanation of the protocol used, hostname and port of a website. You can bypass this restriction by using the cross-origin communication with JSONP [10].

2.2 ROS

2.2.1 ROS-Topics

2.2.2 ROS-Publishers

2.2.3 ROS-Subscribers

2.3 JavaScript

"JavaScript is an interpreted programing language with object-oriented (OO) capabilities. Syntactically, the core JavaScript language resembles C, C++, and Java, with programing constructs such as the if statement, the while loop, and the && operator. The similary end with this syntactic resemblance, however. Javascript is a loosely typed language, which means that variables do not need to have a type specified. Objects in JavaScript map property names to arbitrary property values. In this way, they are more like hash tables or associative arrays(in Perl) than they are like structs (in C) or objects (in C++ or Java). The OO inheritance mechanism og JavaScript is prototype-based like of the little-known language Self. This is quite different from inheritance in C++ abd Java. Like Perl, JavaScript is an interpreted language, and it draws inspiration from Perl in a number of areas, such as its regular-expression and array-handling features" [5]
I used JavaScript in my project as the frontend language , but also as backend of the authentication and signal sever. JavaScript is an easy to use programing language and it has more features and developing than php. Another feature is that is more client-side than php. In the next subsections I will present some framework used in my developing.

2.3.1 JQuery

JQuery is a JavaScript library which make thinks like HTML (Hyper Text Makeup Language) document traversal and manipulation, event handling, animation, and Ajax more easy-to-use. It is compatible with most of browsers and also it has behind it a big comunity. I used Jquery because it make the developing of websites fast and easy to understand by outher developers, but some parts of the code was done in pure JavaScript.

2.3.2 SOCKET.IO

SOCKET.IO is a JavaScript library (frontend and backend APIs) used for making realiable real-time web application. You used it for making a connexion beetween the webpage given by the webserver to the backend server, mostly for message from backend server. It can use two methods. The first one used websockets that makes things fast and friendly with network bandwidth. Second it used pooling that is the clasic way to ask the server from time to time if it has something to send to you.

2.3.3 Sweet Alert 2

Sweet Alert 2 is JavaScript library and also a css file, which make the alerts from the browser customizable. We need this library because the application is mostly used by old people we need to have alerts that have bigger text and are more center in the webpage. Also for the call part you cand have multiple option for responding the alert.

2.3.4 Node.js

Node.js is open-source JavaScript run-time environment that allow to write server-side JavaScript code and it is saw as on of the fundamental elements of "JavaScript everywere" paradigm. This make the client-side code and server-side code to be write in the same programing language that makes it more easy-to-understand for both parts. I used it for the server-side of my web application. I used also some extra library with it like SOSCKET.IO, JWTOKEN and some paradimgs like REST. Node.js is a fast as performance and also as time of developing, and because is JavaScript there is easy for other developers to use.

2.3.5 Roslibjs

Roslibjs is the standard JavaScript library for interacting with ROS systems from a browser. It connects with rosbridge trough WebSockets and has an API that cand let you do things like publishing, subscribing, calling services and most of the essential ROS functionality. Because a big part of my project is the web application in a browser which run on a system with ROS I had to use as my only way to comunicate with ROS and other parts of the big project

2.4 Python

" Python is a general-purpose, high-level programming language whose design philosophy emphasizes code readability. Python's syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C, and the language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming styles. It features a fully dynamic type system and automatic memory management, similar to that of Scheme, Ruby, Perl and Tclm and has a large and comprehensive standard library. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts. Using third-party tools, Python code can be packaged into standalone executable programs. Python interpreters are available for many operating systems.

CPython, the reference implementation of Python, is free and open source software and has a community-based development model, as do nearly all of its alternative implementations. CPython is managed by the non-profit Python Software Foundation. " [12]

2.4.1 Rospy

Rospy is a Python client library that offer an API for interacting with ROS systems. It allows you to do interface with topics, services and parameters. The design of rospy is very easy for programers to use and take a less time in developing. This makes rospy a very good aproach for prototyping in ROS. Some of the ROS tools are made with rospy.

2.5 MongoDB

MongoDB [3] is a document-oriented and general-purpose database. I used it because its documents are in fact BSONs (similar to JSONs) which are very easy to use in JavaScript and Python. Also MongoDB is know to be a scalable database, so it makes easy implementation of the chat part, where scalability is a must. On the server-side I used it for keeping the user credentials, informations, chat messages and connections list, on the robot-side for keeping the news preferences in newspapers and news categories of the user. Because I use Node.js, but also Python, I had to use some libraries, that I will present next.

2.5.1 PyMongo

PyMongo is a Python distribution that make a connexion between Pyhton and MongoDB. It is easy to use by developers and easy to understand. Also you can transform data recived from the database in dictionaries in Pyhton.

2.5.2 MongoDB Node.js Driver

For Node.js we have an official MongoDB driver that provides both callback and promise interaction. This driver is easy-to-use and easy-to-understand by all MongoDB users, because it use almost the the same interface.

Chapter 3

Architectural Overview

With a good understanding of basic concepts of the tehnology used we can go foward and talk about the architectural of the entire system. I will talk separetly the two apps.

3.1 Call App

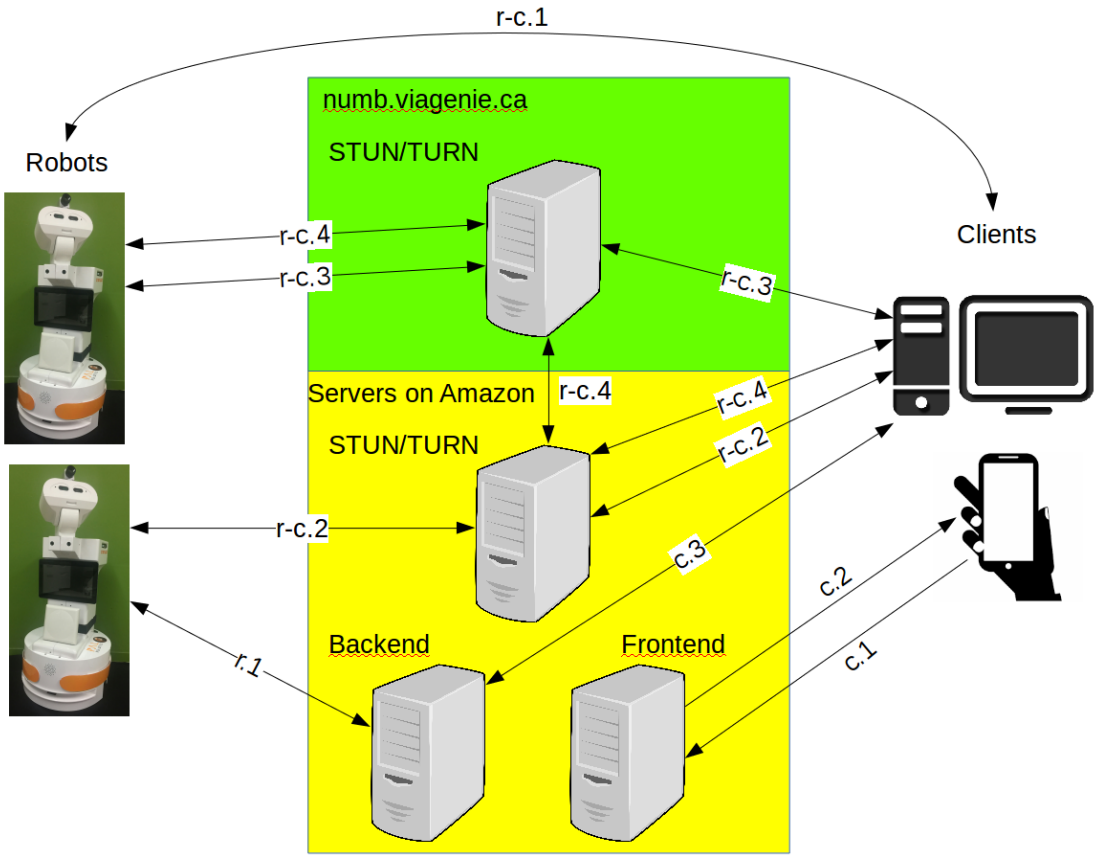

Figure 3.1: Call App

Call App is the part of the system which main ocupation is the social interaction between the

robot users and their relatives seen as clients users, but you can also have this between robots users or clients users. The big system is represented in Figure 3.1. Let's start talking about the entire workflow of calling, but firstly we will use some notation: Alice for the client or robot that is starting the call and Bob for the client or robot reciving the call. If Alice wants to call Bob she cand do that only if he is online. If Bob is online than Alice send a message different from normal messages telling Bob that she wants to start a call. Now Bob can refuse the call and send back to Alice a reject message and then both stops for having a call or he can accept the call and send back to Alice a message telling that he accept the call. When Alice recive the message she create a RTC offer package (that include some basic networking information about Alice and maybe some ICE candidates) and send it to Bob, in the same she set her local description with this offer description and she starts to gather ICE candidates and send them to Bob when she finds them. Bob recive the RTC offer , set his remote description with this offer description, and creates an RTC answer that has almost the same information as the offer, set this answer as his local description and send it to Alice, in the same time he starts to gather ICE candidates and send them to Alice. Alice recive the RTC answer and set it as remote description. After some ICE candidates changes between Alice and Bob if they do not find any matches for a RTC connection they will closed it. If they found they will go to connection established state and they will start to transmit audio/video data. They are tree types of connection posible. First one if Alice and Bob are in the same network or both have public ips they cand make a direct connection (r-c.1). Second one if Alice or Bob are under a different NAT network they can make a direct connection with the help of a STUN Server or they comunicate through a common TURN server (r-c.2 adn r-c.3). Third one if Alice and Bob are under a different NAT network and they do not have a common TURN server accesible but the servers are accesible between them they cand make a connection trought two servers (r-c.4). At one moment if Alice or Bob wants to end the call they send a stop message, After both close the connection. A perfect workflow is represented in Figure 3.2.

3.1.1 Robots

The robots have the webpage application implemented in the hmi_bridge ROS package that is part of the big project ENRICHME, so it does need any connection to frontend server for taking the html or JavaScript scripts. It only makes a connection using SOCKET.IO to the backend server called in the Figure 3.1 r.1. This connection authentificate the robot to the server and announce the others users that is online and it can have a video call. Trough this connection the robots can recive events like when a user go online or offline, recive its contacts list, recive particulary information about some users. Some other two importants thinks that goes trough this connection are the messages and the informaton about starting a call. First the messages, the robot can send them trough r.1, if the other client or robot is online it will recive the message trough its r.1 or c.3 connection, if is offline than the message will be keep in the MongoDB database on the backend server (or in some specialised MongoDB database servers) and the recive users can request them lately trough REST API or SOCKET.IO connection. All the messages are keep in database, so the users cand see them lately. For the chat part the robots see their ROS parametre fro language and makes the right keyboard for the user. Secondly the call part, the robots use their r.1 for transmmiting a call request or to respond a reciving call, and for all the other messages related to the call app, except audio/video data. You cand send audio/video over the r.1 connection but will have a bad performance because it used TCP not UDP and also it always go trough the server. Some difference between robots and normal clients are that robots do not used the camera so the video is taken trough a ROS topic for a RGB camera and you can not add connection on the robot because of the type of users that we have on this aprt of application.

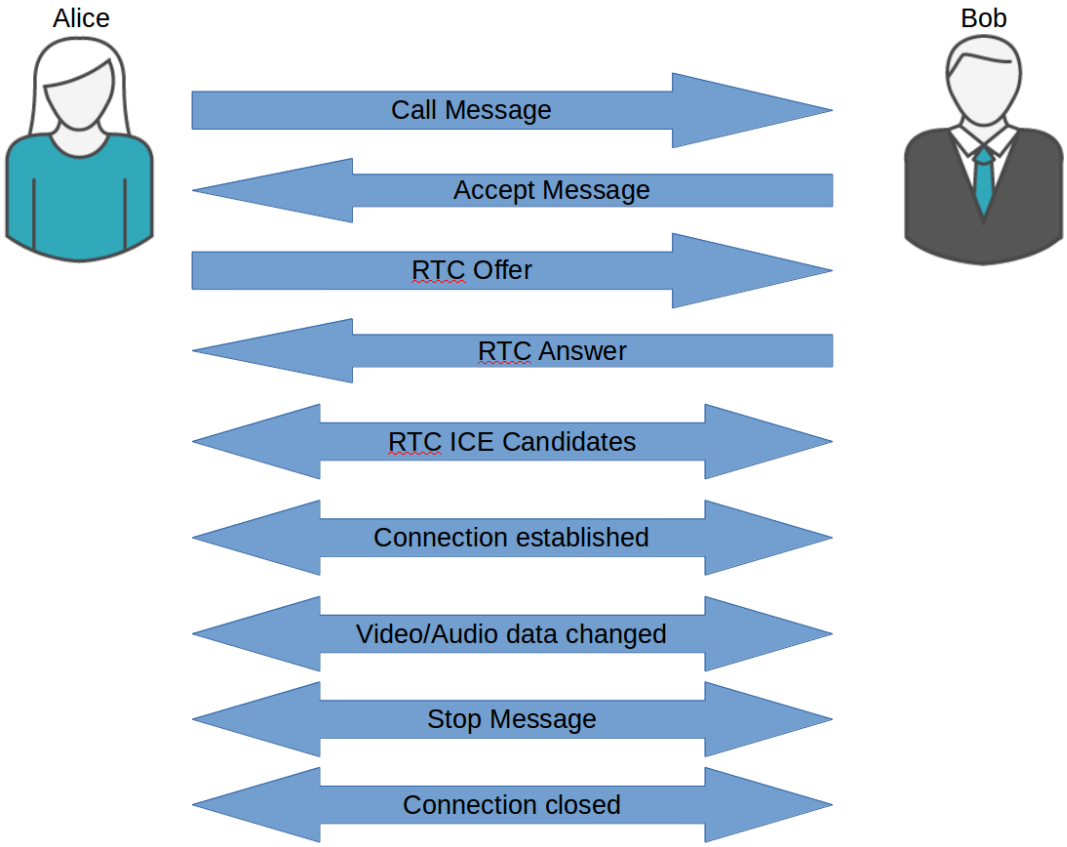

Figure 3.2: Alice-Bob call

3.1.2 Backend Server

The Backend Server offer a REST API and SOCKET.IO connections. The REST API offer services like creating a new user, adding new contacts to the contacts list of an user, basics information about a user, authentification for users, old messages between users, contacts list. The SOCKET.IO connections are used for sending normal messages between users or special messages needed to make RTC connection between users and to keep an online/offline track of the users. In my implementation the MongoDB database is also on the backend server, but if the system can be implemented with the database on another server or on multiple servers agregate for more stability and failure response. Because of the online/offline feature implemented in SOCKET.IO this system is limited at only one backend sever. Changing this feature in MongoDB database but making more a pooling from the clients and robots will incrase the scability of the backend server, but also the will incrase the bandwidth used and decrease performance when the system has less number of users than 40000. Because this number will not be reach in the project ENRICHME I used this type of architectural that makes the implementation more easy to understand and developing. The backend server run on two ports 80 and 443. I will discus the details in the next subsections.

3.1.3 Clients and Frontend Server

Because only clients use the frontend server I chose to talk about them togheter. The clients make a request to the frontend server for the html and JavaScript scripts (c.1) and get a them

as response (c.2) and after that they make the SOCKET.IO connection with the backend server (c.3). I made the design of the page in only one html file and in more JavaScript scripts so the users will have the sensation of using an app not a webpage, and also for the performance, because after the first request the clients will not request anything more from the frontend server. This design and the fact that the project does not have a big number users the performance is high. If there it need to implement the system with less cost we can fusion the frontend server and the backend server without cost to performance too high. Because browsers let webcam and microphone share only over secure connections or locahost inscure connection (more for testing) it is a must to have a ssl certificate for the frontend servers. Because of same origin policy and security issues for browsers also the backend server need a certificate and that is why the backend server is running on two ports 80 for the robots that are using an locahost webpage and 443 for the clients.

3.1.4 STUN/TURN servers

STUN/TURN servers are used for making connections between users that transmit video/audion data. Because of some policy of firewalls or proxy is best to have more of this types of servers. I used on made by me on the amazon coud system and a public one, but free, server from numb.viagenie.ca. The role of STUN is that the clients can discover their network information about them if they are under a NAT network. The TURN servers come in used when users can not make a direct connection and they react as a relay server for their data.

3.2 News App

Figure 3.3. The news app is a news agregator which takes news from different newspapers websites and show them to the user in a nice format, with the option to be read by the robot. Because it is not posible anymore to take an rss feed with the same origin policy implemented in browsers I used an internet API call rss2json.com that takes rss feed you want and transform it in a JSONP format that can bypass this problem. The architecture is simple for this app. You have the newspapers web sites that are publishing rss feed, that also have different feeds for different categories. The rss2json.com API server, where you send your API KEY with the rss feed url and the number of news, is your providers of JSONP formated news. The robots recive the JSONP and show the news to the users with the option to be read loud by them. Users also have the option to change the categories of news they want and even the newspapers.

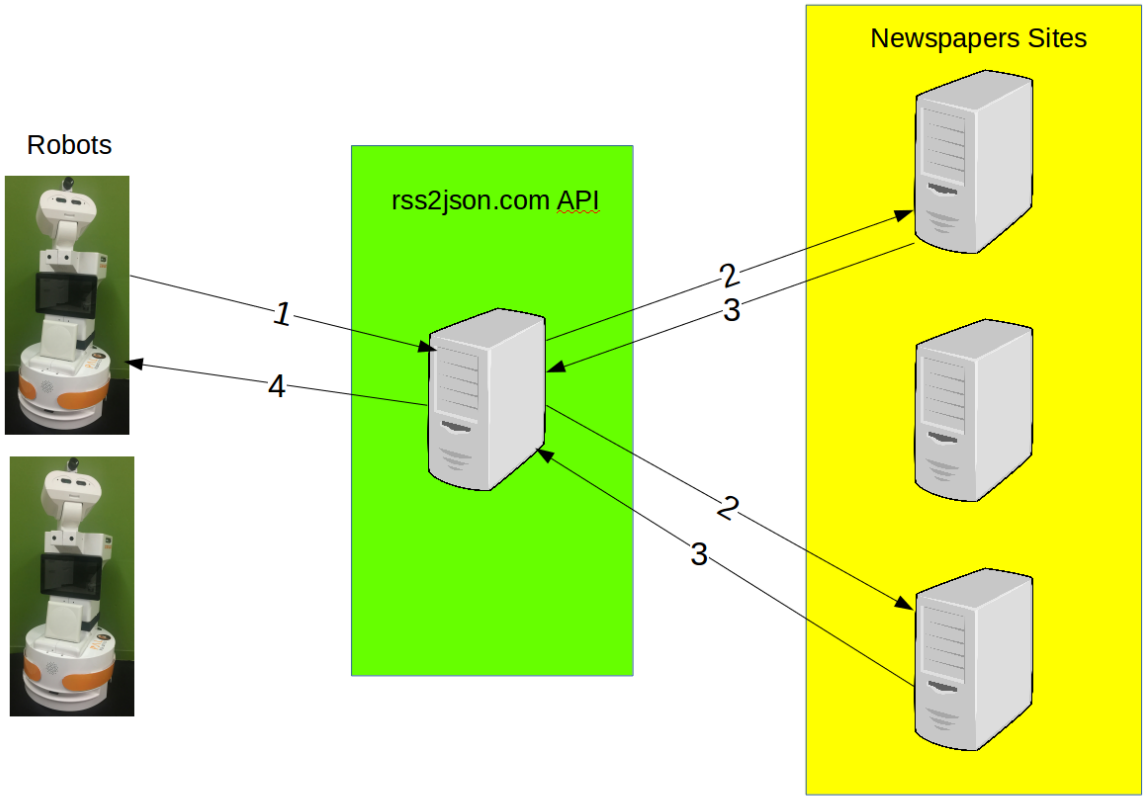

Figure 3.3: News App

Chapter 4

User Interface

With the knwoledge of architectural we can go foward to the user interface. This is also an important part of the project because must adapt for persons with speacials needs. That is the reason why on robots you have buttons and the font-size bigger than client side of the page.

4.1 Call App

The call app is separated in tree parts. First part is the home which contains a list of you connections and on client you have also the option to add new connections. If you press any of the names of the users in the list will pop-up a small window that contains information about the user and options like "message", "call", or "ok" ,that let you chose another user. If you press message you go to the chat part, which on robot has a keyboard in the language specific to the country of the robot that is almost half of the screen, a box that contains the messages between users and some specific buttons. First button give you more information about the remote user. The second one is button to get messages from the past. The third one is button to go back to home page. When the client side you have more space for messages.

4.2 News App

Appendix A

Project Build System Makefiles

A.1 Makefile.test

```
1  # Makefile containing targets specific to testing
2
3  TEST_CASE_SPEC_FILE=full_test_spec.odt
4  API_COVERAGE_FILE=api_coverage.csv
5  REQUIREMENTS_COVERAGE_FILE=requirements_coverage.csv
6  TEST_REPORT_FILE=test_report.odt
7
8
9  # Test Case Specification targets
10
11 .PHONY: full_spec
12 full_spec: $(TEST_CASE_SPEC_FILE)
13         @echo
14         @echo "Generated␣full␣Test␣Case␣Specification␣into␣\"$^\""
15         @echo "Please␣remove␣manually␣the␣generated␣file."
16
17 .PHONY: $(TEST_CASE_SPEC_FILE)
18 $(TEST_CASE_SPEC_FILE):
19         $(TEST_ROOT)/common/tools/generate_all_spec.py --format=odt
               -o $@ $(TEST_ROOT)/functional-tests $(TEST_ROOT)/
               performance-tests $(TEST_ROOT)/robustness-tests
20 #
21
22 # ...
```

Listing A.1: Testing Targets Makefile (Makefile.test)

Bibliography

[1] Adam Bergkvist, Daniel C Burnett, Cullen Jennings, and Anant Narayanan. Webrtc 1.0: Real-time communication between browsers. *Working draft, W3C*, 91, 2012.

[2] RSS Advisory Board. Rss 2.0 specification (2009). *URL: http://www.rssboard.org/rss-specification*, 2014.

[3] Kristina Chodorow. *MongoDB: the definitive guide*. " O'Reilly Media, Inc.", 2013.

[4] Jim Downing. Web feeds and repositories. 2008.

[5] David Flanagan. *JavaScript: the definitive guide*. " O'Reilly Media, Inc.", 2006.

[6] International Organization for Standardization. Iso/iec 26300:2006 open document format. http://std.dkuug.dk/keld/iso26300-odf/is26300/iso_iec_26300:2006_e.pdf, December 2006.

[7] R Mahy. Network working group j. rosenberg request for comments: 3489 j. weinberger category: Standards track dynamicsoft c. huitema microsoft. 2003.

[8] Rohan Mahy, Philip Matthews, and Jonathan Rosenberg. Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun). Technical report, 2010.

[9] Mozilla Developer Network. Same-origin policy, 2014. *URL: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy (b ezocht op 10-05-2015)*.

[10] S Ozses and S Ergul. Cross-domain communications with jsonp, 2009.

[11] Jonathan Rosenberg. Interactive connectivity establishment (ice): A methodology for network address translator (nat) traversal for offer/answer protocols. 2010.

[12] Guido Van Rossum et al. Python programming language. In *USENIX Annual Technical Conference*, volume 41, page 36, 2007.

Index