UNIVERSITY OF TARTU

Institute of Computer Science

Software Engineering Curriculum

Vello Vaherpuu

Emergency Vehicle Warning System for Automotives' Navigational Systems

Master's Thesis (30 ECTS)

Supervisor(s): Amnir Hadachi, Ph.D.

Tartu 2019

Emergency Vehicle Warning System for Automotives' Navigational Systems

Abstract:

The purpose of this thesis was to examine different ways how to inform drivers about nearby emergency vehicles and build a prototype of such a system. The system consists of a server and an iOS mobile application. The clients of the mobile application can use it for navigation. In addition to the directions, the routes of nearby emergency vehicles, intersections and overlapping parts are also displayed.

Keywords:

Emergency vehicles, safety, mobile application

CERCS: P170 - Computer science, numerical analysis, systems, control

Lühikokkuvõte:

Käesoleva magistritöö eesmärk oli uurida erinevaid võimalusi, kuidas teavitada juhte lähenavast alarmsõidukist ja luua sellise süsteemi prototüüp. Rakendus koosneb kahest osast: serverist ja iOS mobiilirakendusest. Mobiilirakenduse kasutajatel on võimalik rakendust kasutada navigeerimiseks, kus lisaks juhistele kuvatakse ka lähedal olevate alarmsõidukite teekonnad ja nendega ristuvad punktid või kattuvad osad.

Võtmesõnad:

Alarmsõidukid, turvalisus, mobiilirakendus

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Table of Contents

Table of figures

1 Introduction

By now vehicles have been in mass production for over 100 years and the number of vehicles grows year by year. Due to soundproofing, loud music, confusion, distracting children, hearing impairment, air conditioning or drivers just not paying attention it is apparent that sirens and flashing lights are not enough for emergency vehicles to safely reach their destination. With the help of today's technology, there are additional ways to alert nearby drivers about the presence of emergency vehicles.

This research is done in order to decrease the number of accidents involving emergency vehicles and is highly motivated by the personal experience of the author himself where he was almost T-boned by a fire truck due to not hearing the sirens.

1.1 Objectives

The main goals of the current master's thesis are to examine different methods how emergency vehicles could make themselves more visible and to create a prototype of a system that shows the location and routes of nearby emergency vehicles in real-time.

The main research questions that this thesis addresses:

- What methods are already in use to clear the path of emergency vehicles?
- Examine different ways how could the drivers be warned about the close presence of emergency vehicles and choose the most suitable one for the prototype.

1.2 Limitations

The prototype developed during this thesis relies on data that should be provided by authorities. The data includes the location and state of different emergency vehicles. As this kind of data is hard to obtain and the author does not know exactly how this data looks like, it is mocked by the author. The prototype is designed and developed in a way that if the system should go live in the future it would be easy to adapt.

In Estonia, there are 13 different types of emergency vehicles [1] §1, but the prototype of the system will only focus on the most common ones: ambulances, fire trucks and police vehicles.

1.3 Thesis structure

The work is divided into six chapters. The first chapter describes the technologies that are currently in use in order to clear the path of emergency vehicles. The chapter also examines different ways how the nearby drivers of an emergency vehicle could be directly contacted. The chapter ends with an author's decision on how the prototype should be built. The second chapter focuses on describing the requirements of the system, what the whole system consists of and how should different components of the system communicate with each other. The third chapter explains how the prototype is built and what are the features of the built system. It also gives an overview of system requirements in order to run the prototype. The fourth chapter talks about how different parts of the system were tested. The fifth chapter wraps up the topic. In the final chapter, the author gives information about the shortcomings and improvements which could be done in the future.

2 State-of-the-art

The following chapter focuses on the research questions and gives an overview of different ways of how emergency vehicles could reach their destinations more safely. First, the overview of accidents that involved emergency vehicles in Estonia is given. Next, different methods of how the path of emergency vehicles is cleared are described. Furthermore, different ways of how could the drivers be warned about the close presence of emergency vehicles are discussed. The chapter ends with the author's decision on how the prototype should be built

2.1 Statistics in Estonia

In order to get some insights about how many accidents involving the emergency vehicles happened in Estonia, the author sent a query about accidents during the years 2011-2018 to Estonian Police and Border Guard Board, see Appendix I. In total there were 64 accidents, see Figure 1, 45 accidents involved another vehicle, 10 accidents involved no other parties, 4 accidents involved a pedestrian, 3 accidents involved a bicycle and 2 accidents involved a motorcycle. In total 31 accidents happened in Harju county and 26 of those in Tallinn, the capital and the largest city of Estonia. Unfortunately, there is no data on whether the emergency vehicles were responding to an emergency or the accidents happened during normal traffic. One can only assume that at least half of them happened during an emergency response since the risk of getting into an accident is much higher during emergency response. The fact that 48% of the accidents happened in a single city can be an indicator that emergency vehicles are hard to notice when there is a lot of noise, intersections and other vehicles around.

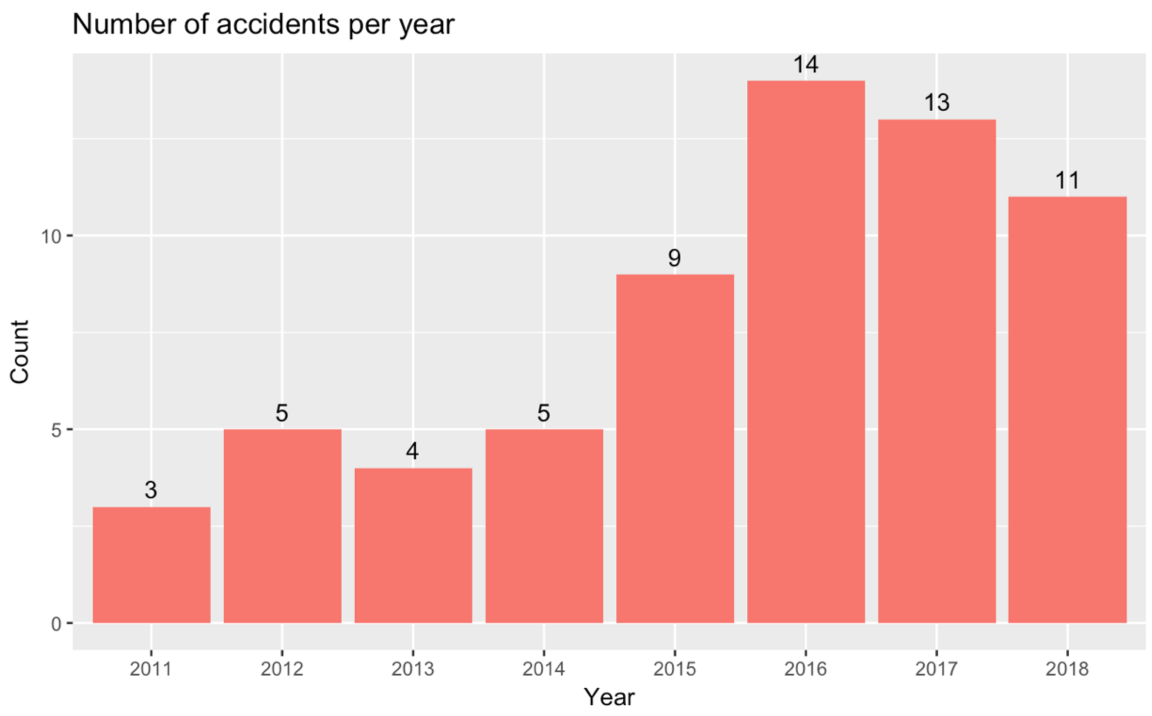


Figure 1. Number of accidents involving emergency vehicle per year in Estonia

2.2 Clearing the path of emergency vehicles

In the following subchapter different methods of how the path of an emergency vehicle could be cleared without directly contacting the drivers are discussed.

Emergency response time

Before the ways how the emergency vehicle path is cleared is described, it is important to discuss different factors that can affect emergency response time. The emergency response time is the time between the moment the emergency call was received at the dispatch and the time the team reached the scene [2]. Before the team leaves the station the response time may already be affected by many factors: the total length of the call, the queue times at dispatch and the time for the emergency team to be ready [2]. During travel between the starting point and destination, the emergency response time can be affected by the speed of an emergency vehicle, the number of congestions, the number of other vehicles in the path of an emergency vehicle and the number of red traffic lights on the path [3]. Delays in any process could lead to terrible consequences from extra damage to properties to the loss of human lives. The current work focuses on the traveling part and everything, considering the calls and what happens before the emergency team leaves the station, is out of scope.

Awareness training

The first step in keeping the path of emergency vehicles clear is training the drivers to notice their surroundings, pay attention to the road and rear-view mirrors. When an emergency vehicle is spotted the driver should react quickly and safely, not putting themselves or others in danger. How exactly the driver should react is written in traffic rules and may differ from country to country, the training is conducted by driving schools.

Sirens and lighting devices

In Estonia according to [1] §5 the emergency vehicles have two types of devices: lighting and sound devices. The lighting devices are flashing light, where the light turns on and off, and beacons, where the lamp rotates. At least one of the lighting devices have to be always visible from every horizontal direction. The sound devices are loudspeaker and intermittent horn which are compulsory for every emergency vehicle that is coloured according to [1] §10 and §11.

Lighting and sound devices serve the same purpose – capture the attention of nearby drivers in order to warn them that the emergency vehicle is in a hurry to reach the destination or warn about hazards in case they are stationary. Over the years a lot of research has been done in order to find out the best ways where to put the sirens, which way they should be faced and how loud the sound should be. As both technologies have reached the point where only minor improvements could be done, further research in this direction is out of the scope of this thesis.

Pre-emption systems

As emergency response time is affected by congestions and red traffic lights, cities around the world have installed different pre-emption systems since the 1960s [4]. Traffic signal pre-emption is the process of allowing emergency vehicles to manipulate traffic signals in their path, using wireless communication devices installed on emergency vehicles and intersections. There are different kinds of pre-emption systems in use: optical, sound-based, radio controlled and global positioning system (GPS) based systems [3, 5].

The most commonly used systems are optical systems that use a strobe-lamp on the vehicle and an optical sensor in each direction of the intersection, in order it to work, a clear line-of-sight is required between the emergency vehicle and the intersection [4, 5], see Figure 2.

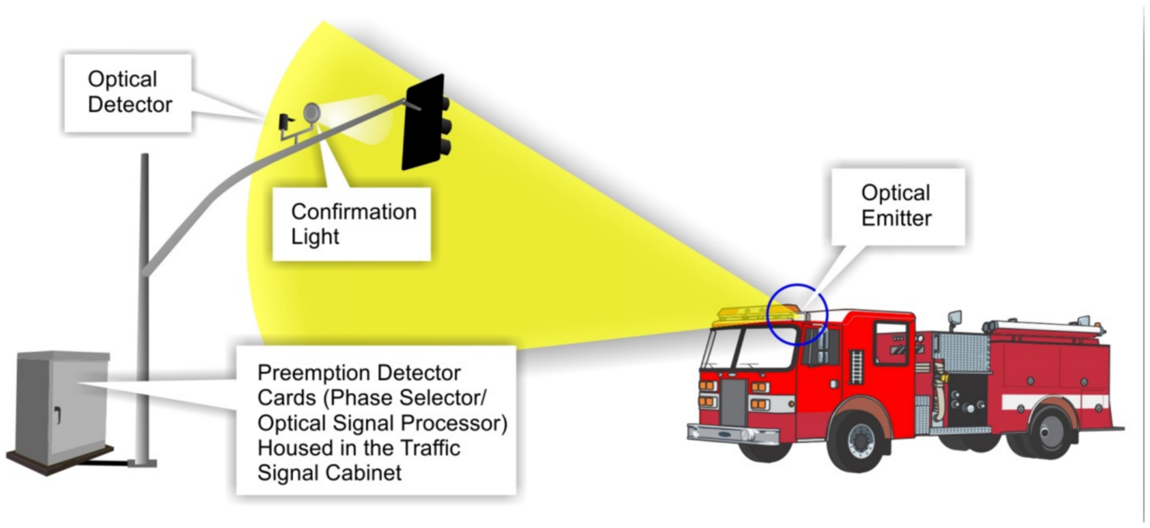

Optical Detector
Confirmation Light
Optical Emitter
Preemption Detector Cards (Phase Selector/ Optical Signal Processor) Housed in the Traffic Signal Cabinet

Figure 2. Components of the optical pre-emption system [6]

Sound-based systems use the sound emitted by the alarming vehicle and directional microphones that can detect the correct decibel levels installed at an intersection. Once a signal is detected and verified, a pre-emption request is sent to the signal controller, see Figure 3. [4, 5]

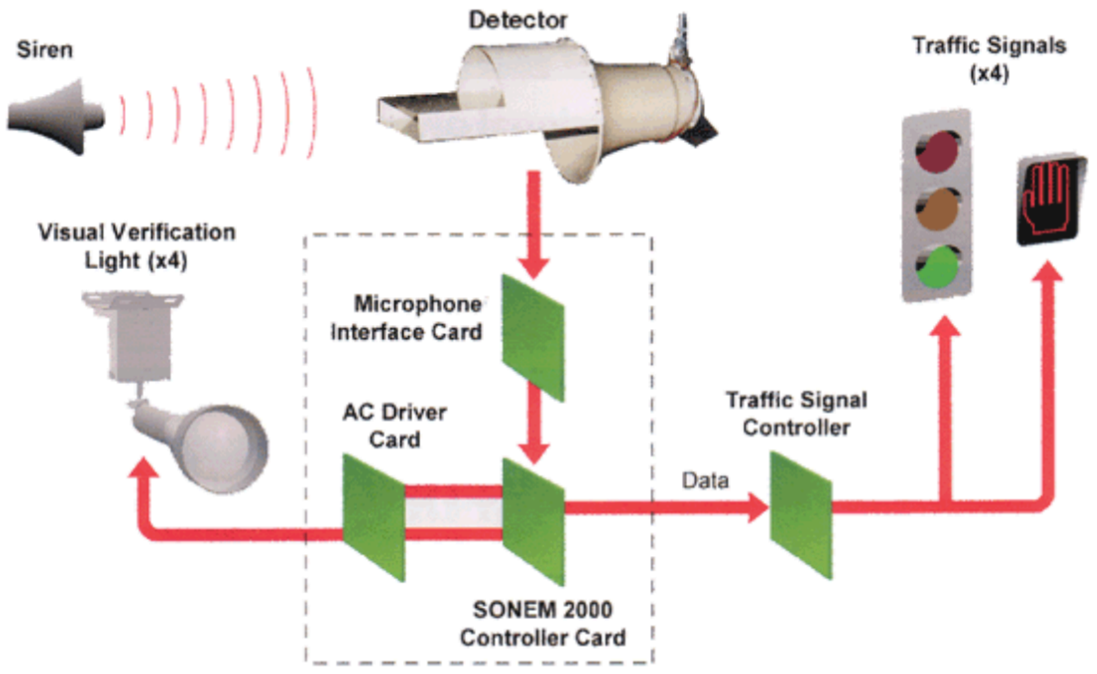

Siren
Detector
Traffic Signals (x4)
Visual Verification Light (x4)
Microphone Interface Card
AC Driver Card
Traffic Signal Controller
Data
SONEM 2000 Controller Card

Figure 3. Example of sound-based pre-emption system SONEM 2000 [7]

Radio-based systems use a directional signal for pre-emption that is transmitted from an emergency vehicle to an intersection via a one-way radio [5], see Figure 4.

The GPS-based system uses the speed, position and travel direction of the emergency vehicle in order to determine the signal pre-emption time. Both the emergency vehicle and intersection have to be equipped with a GPS receiver and a radio transceiver for two-way communication, see Figure 4. [4, 5]

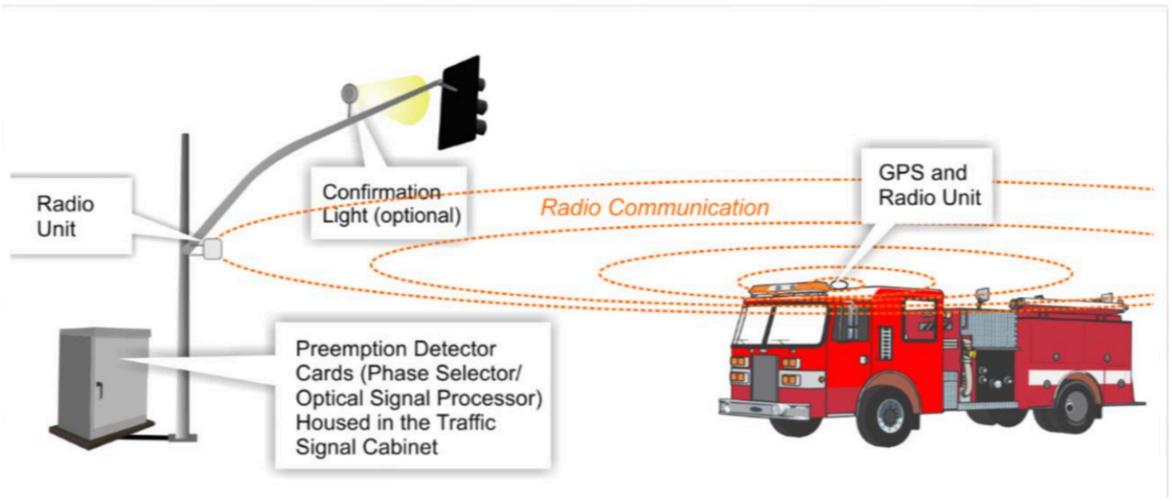

Radio Unit
Confirmation Light (optional)
Radio Communication
GPS and Radio Unit
Preemption Detector Cards (Phase Selector/ Optical Signal Processor) Housed in the Traffic Signal Cabinet

Figure 4. Basic components of radio/GPS-based pre-emption system [6]

2.3 Contacting drivers directly

In the following chapter, the ways how emergency vehicles could inform nearby drivers about their presence is discussed. At the end of the chapter, a decision is made by the author which of those methods is used for building the prototype.

Radio Data System

Radio Data System (RDS) or Radio Broadcast Data System (RBDS), both referred to as RDS from now on, is a standard communication protocol for embedding digital information in FM radio broadcasts. It was developed by the public broadcasters collaborating within the European Broadcasting Union (EBU). The first specification of RDS was initially published by the EBU in 1984 [8].

The features provided by RDS [9] are:

- AF – Alternative Frequencies list
- CT – Clock Time and date
- PTY – Programme Type (news, rock music, etc.)
- ECC – Extended Country Code

- EON – Enhanced Other Networks information
- ODA – Open Data Applications
- PI – Programme Identification
- (L)PS – (Long) Programme Service name
- PTYN – Programme Type name
- (e)RT – (enhanced) RadioText
- TA – Traffic Announcement identification
- TMC – Traffic Message Channel
- TP – Traffic Programme identification

The most important features regarding emergency vehicles are TA, TP and ODA, which could be used to alert drivers about the presence or the path of an emergency vehicle. The advantage of using RDS technology is that almost every modern car has a radio system already installed in the vehicle, which means that drivers do not have to install additional systems and governments can introduce new systems using RDS relatively cheap. One drawback of this technology is that drives know an emergency vehicle is nearby but do not know where exactly.

In fact, researchers from KTH Royal Institute of Technology in Stockholm already have developed a product that allows emergency vehicles, road maintenance providers and others to communicate important real-time information to vehicles in the direct vicinity. The product is called EVAM and uses RDS technology to directly send information to other vehicles. The information is received through the sound system. [10, 11]

V2V communications.

Vehicle-to-vehicle communication refers to a wireless network where vehicles exchange information such as vehicle size, position, speed, direction, acceleration, brake status, turn signal status, loss of stability and more, see Figure 5. V2V uses a dedicated short-range communications (DSRC) standard. [12]

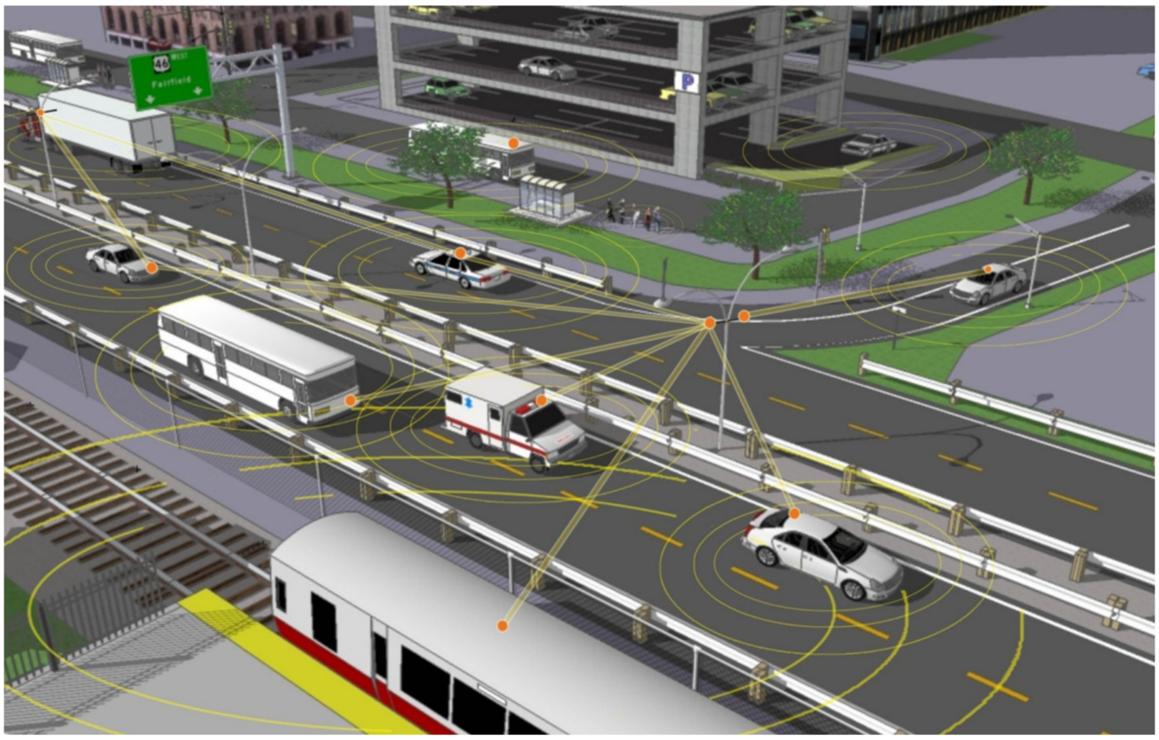


Figure 5. Visual representation of V2V communication [12]

As the V2V network does not rely on pre-existing infrastructure and is not centralized it could be approached as a vehicular ad hoc network (VANET), a subclass of mobile ad hoc network (MANET) [13]. Every vehicle acts as a node that can send, capture and retransmit information. This means that with 5-10 hops on the network the emergency vehicle could gather traffic information kilometers ahead to select the best route. Furthermore, in order to clear the path, the emergency vehicle could broadcast its own information and instructions to nearby vehicles. The drawback of V2V is that it requires newer vehicles that support V2V communications.

ECall and eCall Plus

It is mandatory for all the vehicles of type M1 and N1 sold within the European Union (EU) from April 2018 to have an eCall emergency system installed. The system automatically calls the emergency service when the vehicle is involved in a collision anywhere in the EU. [14]

Ford and Vodafone have gone one step further and are developing a system called eCall Plus which uses vehicle to infrastructure (V2I) and V2V technologies. The system is designed to provide an early warning to the drivers that an emergency vehicle is approaching and where exactly the emergency vehicle is located. Furthermore, the system also provides information on how the drivers should act whether the drivers should pull to the side or form an emergency corridor. [15]

Mobile applications

Another way how emergency vehicles could alert nearby drivers is through mobile devices that are connected to the internet. In 2018 almost 50% of the world population had access to mobile internet and it is estimated that by the year 2025 the number grows to 70% [16]. Furthermore, developing mobile applications is cheaper and faster than developing software for specific vehicle brands, see Table 1.

Over 110 million users benefit from the mobile navigation application Waze a month [17]. Waze is a crowd-sourced application that allows drivers and passengers to report events on the road and display it as an overlay on the map along with routing information [18]. In addition to incidents on the road, Waze already supports sharing information of two types of moving vehicles: garbage trucks and snowplows.

From safety perspective having a mobile application and a newer vehicle that supports Apple CarPlay [19] or Android Auto [20] could be a deciding factor whether the driver gets into an accident or reacts in advance. Currently, there are more than 500 different car models that support Apple CarPlay [21] and over 400 models that support Android Auto [20]. Both systems allow users to connect their mobile devices to a vehicle and use their mobile device through the vehicle's multimedia system. Users can pick-up calls, send messages and use third-party applications.

Characteristics	Mobile applications	Vehicle software
Number of platforms	Basically two (99.98% of total market share): Android (77.22%) and iOS (22.76%) [21].	Manufacturer specific, newer models support connecting mobile devices through Apple CarPlay and Android Auto.
Distribution channels	Apple App Store [23], Google Play [24].	Car dealerships [25].

Cost of updates for users	Usually free, depends on the application.	It usually involves visiting the dealership and is not free.
Updating process	It can be done automatically on the background. Done with a few taps.	The update file is downloaded using a PC onto a USB flash drive or SD card which is then plugged into the terminal in vehicle [25].
Device / vehicle replacement cycle (in U.S)	2.7 years in 2018 [26].	11.6 years in 2016 (light vehicles) [27].

Table 1. Differences between the mobile application and vehicle software development and distribution

2.4 The chosen method for prototype

The author believes that a system that could notify nearby drivers about emergency vehicles should be done using a mobile application similar to Waze. The opinion is based on the number of monthly Waze users and the advantages of developing a mobile application over developing vehicle software, see Table 1. Furthermore, the fact that newer car models support Apple CarPlay and Android Auto means that a mobile application could be used seamlessly as a part of the vehicle's multimedia system.

2.5 Conclusion

In conclusion, there are already numerous ways implemented how the path of emergency vehicles could be cleared: driver training, sirens, lighting and different pre-emption systems. However, most of the accidents involving emergency vehicles still happen in cities where emergency vehicles are harder to notice. In order to give drivers more time and instructions on how to react, the author examined different ways how could emergency vehicles inform nearby drivers directly. One way is to use RDS, which can be used to alert nearby drivers through their sound system but lacks the information about the exact location of the emergency vehicle. Another way is to use V2V and V2I communication, which means in short that vehicles can communicate with each other and to infrastructure. Ford and Vodafone are already developing eCall Plus, which is based on V2V and V2I technologies. ECall Plus is designed to provide an early warning and instructions for nearby drivers of an emergency vehicle. It turned out that nearby drivers of an emergency vehicle could also be warned through mobile applications, which should be faster and cheaper to develop and have the potential to reach almost 50% of the world population. The author made a decision that the prototype of the system should alert nearby drivers of an emergency vehicle through a mobile application.

3 Design and architecture

3.1 Introduction

The main goal of this master's thesis is to develop a prototype of a system that would be used for navigation and alerting drivers about nearby emergency vehicles. In the following chapter system requirements and architecture will be described.

3.2 Terminology

In the following subchapter, the terminology used throughout the system is explained in alphabetical order.

- Authority - an entity who has access to emergency vehicle location data and shares it with the system.
- Client - an entity who has registered in the system in order to use its features through a mobile application.
- Mobile application - application through which clients can navigate and see data of nearby emergency vehicles in real time.
- Navigation mode – a client has set a destination where he/she would like to go and the route how to get to the destination is shown on the map.
- State of emergency vehicle - active, inactive.
- Server - a program that receives data from the authorities, processes it and broadcasts to active clients.
- Types of emergency vehicles - ambulance, fire truck, police.

3.3 System requirements

The following subchapter describes the requirements that were set before the development of the application. The clients receive information through a mobile application, as this was the method chosen during the research process. Necessities consist of functional and non-functional requirements.

Functional requirements:

- Clients can use the application for navigation
- Clients can see the nearby emergency vehicle(s) in real time
- Clients can see where their route intersects with the route of an emergency vehicle(s)

- Clients can see where their route overlaps with the route of an emergency vehicle(s)
- Clients can see emergency vehicles when navigation mode is not active
- Emergency vehicles are distinguishable by their type on the map
- Authorities can update the location of the emergency vehicle(s)
- Authorities can update the route of the emergency vehicle(s)
- Authorities can update the state of the emergency vehicle(s)

Non-functional requirements:

- Usable on iOS version 12.1+
- Usable for 100 concurrent users

3.4 Architecture

The following subchapter describes the general idea and architectural decisions made by the author before the development of the application.

Idea

There are three different components in the system: authorities, clients and the server. The general idea is that authorities provide the information of emergency vehicles under their control, send the data to the server and server broadcasts all the relevant information to the active clients. See Figure 6.

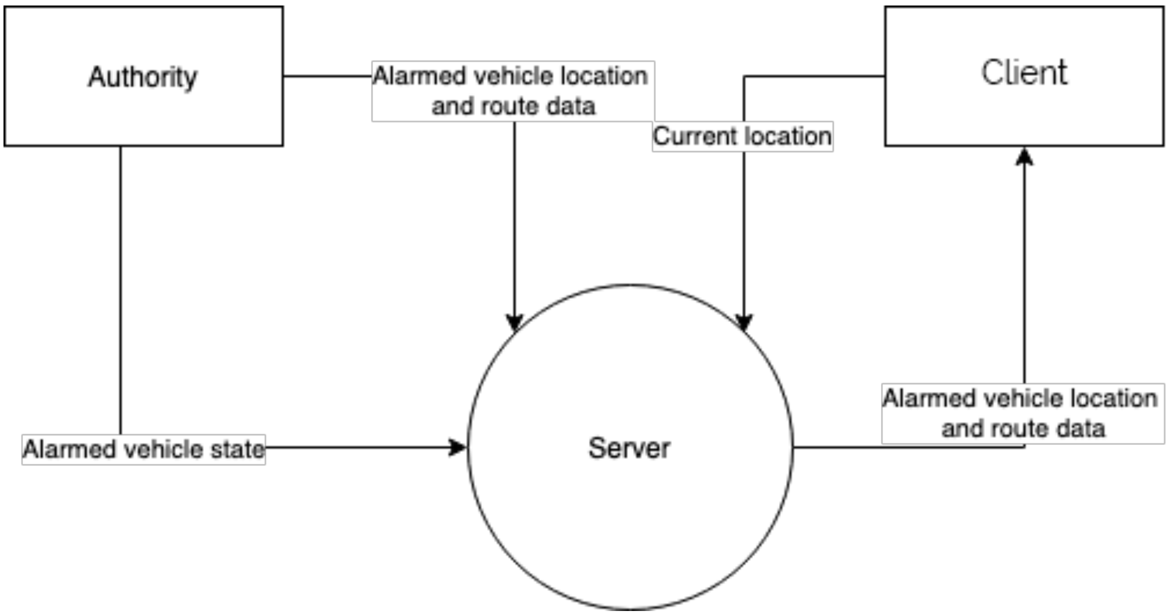

Authority
Alarmed vehicle location and route data
Client
Current location
Server
Alarmed vehicle state
Alarmed vehicle location and route data

Figure 6. Data flow diagram

Communication

One of the main goals is to provide real-time data to clients. In order to achieve that there are three different methods to choose from. The first one is called HTTP Polling where clients ask the server if there is any new information available at a certain regular interval and the server responds with new information or an empty message [28]. HTTP Polling is a good choice if the interval is known and not very short, such as hourly temperature readings [28]. Since the data in our application has to be in real-time this method can become a real burden for the server. The interval time has to be small otherwise the real-time feature will be lost and if the interval is small it may lead to hundreds of requests per second to the server even if there is no information about emergency vehicles available.

Another variant of HTTP Polling is Long Polling [28] which is similar to HTTP Polling. When the server receives a request and no new information is available no response is sent. Instead, the server holds the request and responds to the request when new information appears or a timeout expires. When the client receives a response another request is sent to the server immediately. It is more efficient than regular HTTP Polling but adds another layer of complexity on the server side.

The second way how it could be achieved is to use a WebSocket protocol [29] to enable bi-directional communication between clients and the server. Using this method, the server does not have to deal with unnecessary requests from the clients and pushes the information to clients when it becomes available. The author chose to use this method.

System structure

The system is not large class wise, see Figure 7. There is a class for users that is used for sign-in purposes and it has two subclasses: client and authority. In addition, there is a class for emergency vehicles and for locations. The same structure is used in the database.

The client class is meant for mobile application clients, they have additional fields for first name and last name. To make it possible for the server to query certain clients in the range of emergency vehicles, each active client actively shares his/her location.

The authority class if meant for as the name says authorities, it has an additional field for the name of the authority. The author assumes that each authority has a certain number of emergency vehicles that it is responsible for and at the same time has access to its state and location. The authority is responsible for updating the state and the location of the emergency vehicles the authority has control over.

Every emergency vehicle belongs to one authority, has a name, vehicle type and state. The current system supports three main types of emergency vehicles: police, fire truck and ambulance.

Both the client and the emergency vehicle may have location data that has fields for the current location and the route. The data of the current location and the route is stored and used in GeoJSON [30] format.

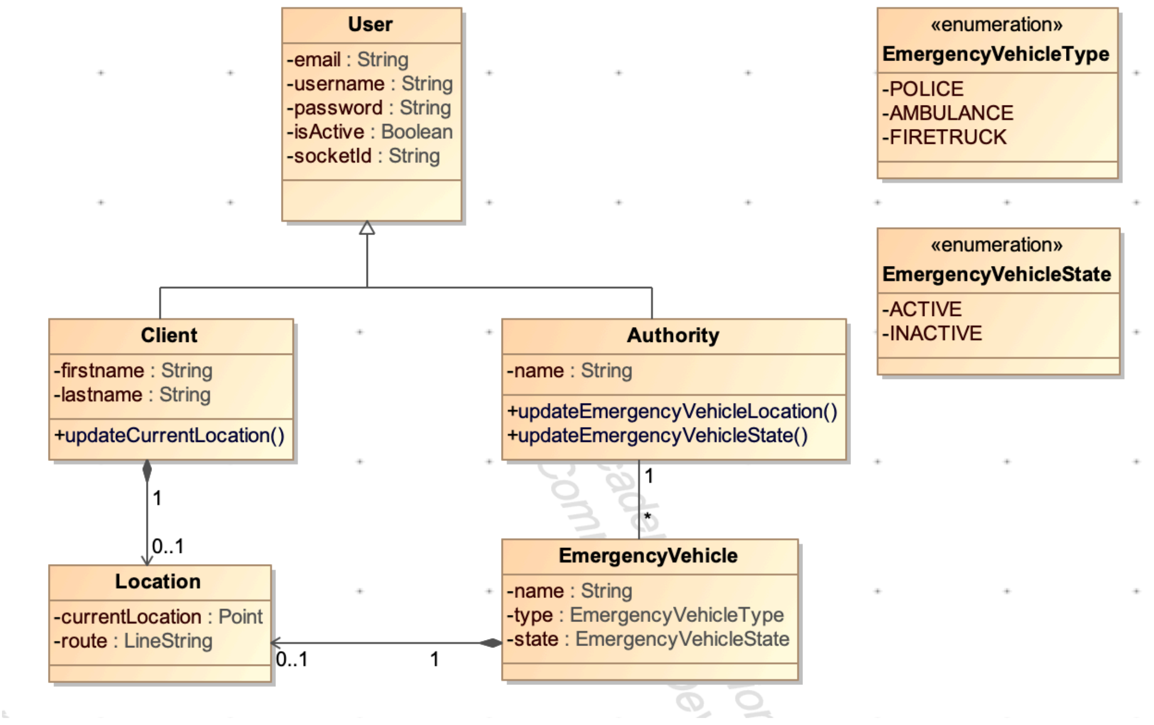


Figure 7. Class diagram describing system structure

3.5 Broadcasting logic

One of the hardest parts was how the broadcasting part should work. The first problem was which party should the data follow - clients or authorities. If the client data would be in focus, then the queries about emergency vehicles would center around client location. And vice versa if the emergency vehicle data is in focus then the queries would be about emergency vehicles location. Usually, there are way more common vehicles around than active emergency vehicles. Taking it into consideration the author decided that emergency vehicles should be focused since it can be handled with fewer queries. For example, there are 4 clients and 1 emergency vehicle. It is much easier to query 4 clients around 1 emergency vehicle than query 1 emergency vehicle 4 times by each client.

The next problem was how often should the data be broadcasted. The first idea was that there should be a worker that checks active emergency vehicles at a certain time interval and broadcasts the data to active clients. For example, the data could be broadcasted every second. The other idea was to broadcast data only when the emergency vehicle location or state is updated by the authority. The second idea should be more efficient and it was chosen by the author.

Having figured out how often and what kind of data is broadcasted all it was left to figure out was how to select clients who should receive the data. The first idea was that only those clients should be selected whose route intersects or overlaps with an emergency vehicle. Another idea was that clients in a certain radius should be selected. The author decided to go with the second idea because from the client point of view it looks great when emergency vehicles appear on the map and it may be the feature that attracts users if the system goes live in the future.

The broadcasting logic can be seen in Figure 8. The authority sets the state of an emergency vehicle to active and the broadcasting begins. First, the emergency vehicle is queried from the database, then the clients in radius will be queried and added to a unique socket room meant for the emergency vehicle. After that, a message containing the emergency vehicle data is broadcasted to all clients in the room and the state of the emergency vehicle is updated in the database. After the emergency vehicle state is updated the authority will start updating emergency vehicle location. The sequence of events of updating the emergency vehicle's location is similar to updating the emergency vehicle state just leaving out the emergency vehicle state updating steps. If an emergency vehicle has done its job and authority will set the state of emergency vehicle inactive, the clients receive the final message that the emergency vehicle state was changed to inactive and the socket room gets destroyed.

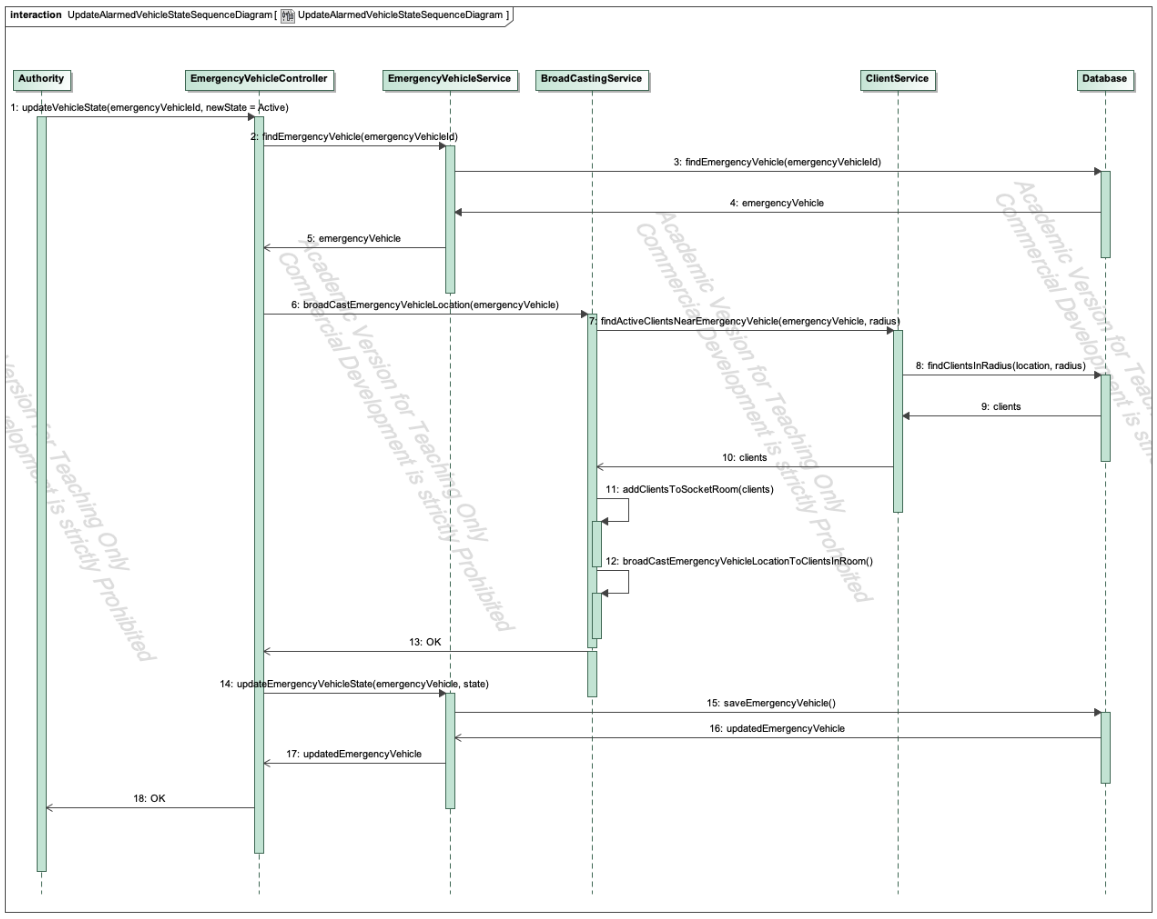

interaction UpdateAlarmedVehicleStateSequenceDiagram [UpdateAlarmedVehicleStateSequenceDiagram]
Authority
EmergencyVehicleController
EmergencyVehicleService
BroadCastingService
ClientService
Database
1: updateVehicleState(emergencyVehicleId, newState = Active)
2: findEmergencyVehicle(emergencyVehicleId)
3: findEmergencyVehicle(emergencyVehicleId)
4: emergencyVehicle
5: emergencyVehicle
6: broadCastEmergencyVehicleLocation(emergencyVehicle)
7: findActiveClientsNearEmergencyVehicle(emergencyVehicle, radius)
8: findClientsInRadius(location, radius)
9: clients
10: clients
11: addClientsToSocketRoom(clients)
12: broadCastEmergencyVehicleLocationToClientsInRoom()
13: OK
14: updateEmergencyVehicleState(emergencyVehicle, state)
15: saveEmergencyVehicle()
16: updatedEmergencyVehicle
17: updatedEmergencyVehicle
18: OK

Figure 8. Broadcasting sequence diagram

3.6 Conclusion

In conclusion, the system consists of three components: authorities, server and a mobile application. The data is provided by the authorities. Then, the server processes it and broadcasts to active clients. The communication is done over WebSockets and there are five different classes in the system structure. The system focuses on the data of emergency vehicles and the data is only broadcasted when the location or state of emergency vehicles changes. In order to receive the data of nearby emergency vehicles, the clients should be in a certain radius.

4 Development and implementation

4.1 Introduction

In the following chapter, the author describes the development process, libraries, software development kits (SDKs) and frameworks that were used and the features that were implemented.

4.2 Development process

The development process of the system started with specifying the requirements and choosing the technologies. After the main requirements were set, different navigation SDKs for iOS were examined and tested to find the one that suits the best. The author decided to use Mapbox Navigation SDK [31]. After that, the development of the server and mobile application started iteratively. When a requirement was implemented it was manually tested and validated.

4.3 Server

The author chose to use Node.js [32] server with Express [33] framework on top of it. To keep it scalable and easier to manage in the future it's all written in Typescript [34]. In order to save time on project configuration, the author decided to use a boilerplate code made by W3tecch as a starting point [35]. It used all the technologies that the author intended to use and some additional ones that were useful. Technologies that were not useful were removed.

In order to store data PostgreSQL [36] relational database was used. As the data that is mostly used is geographical an extension called PostGIS [37] was added to PostgreSQL to support storing geographical objects and location queries. In order to manage database schema changes, migrations and mapping data from the database to object a library called TypeORM [38] is used.

There are two types of geographical data in the system: location also known as a point and a route also known as a polyline. A point is defined by three values: longitude, latitude and elevation. As this system is a prototype the author has not considered elevation to make the development process easier and faster. A polyline is defined as a curve specified by the sequence of points. In order to represent geographical data, the author uses the GeoJSON format. The reason why the GeoJSON format was chosen is that it is easy to read, all the other data is also in JSON (JavaScript Object Notation) [39] format and the Mapbox SDK used in mobile application works with this format out of the box.

In order to calculate intersections and overlapping parts of a client route and emergency vehicle route a library called Turf.js [40] is used. Ideally, the calculation should be done on the client side but since the Turf.js iOS library does not have the required methods available it's done on the server side. The server provides a utility endpoint where the client can query for intersections and overlapping parts of two routes.

As the author chose the WebSockets as the communication method a library Socket.IO [41] is used. The sockets are used to send and receive live data, but the system also has some REST (Representational State Transfer) [42] endpoints for authorization and account creation. The data is always exchanged in JSON format. The data flow in the server can be seen in Figure 9.

In order to manage all of the dependencies and the corresponding versions, a package manager Yarn [43] is used.

The server code is open-source and is available on BitBucket[1]. Furthermore, the repository also contains a friendly user manual[2].

[1] https://bitbucket.org/magistergetouttamyway/backend/src/master/
[2] https://bitbucket.org/magistergetouttamyway/backend/src/master/README.md

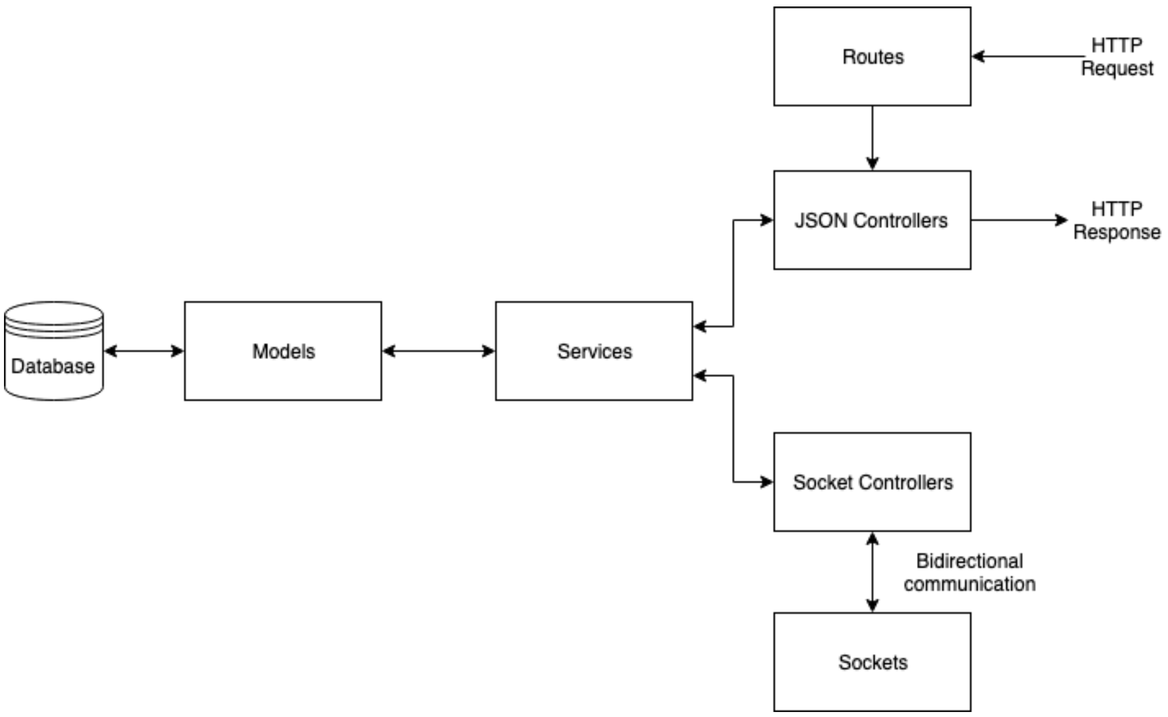


Figure 9. Diagram of data flow in server

4.4 iOS application

The main purpose of the mobile application is to show clients where they are, help them navigate and display the data of nearby active emergency vehicles.

The iOS application is written in Swift 4. In order to provide navigation features to clients, an SDK called Mapbox is used. It was chosen by the author because of the simple setup, it is well documented, it is based on OpenStreetMap [44], it has all the methods to display additional information, it takes care of the navigation part and looks good. For communicating with the server through sockets a Socket.IO client library is used. In order to parse JSON a library called SwiftyJSON [45] is used.

The iOS application code is open-source and is available on BitBucket[3]. In addition, the repository contains a user manual[4].

[3] https://bitbucket.org/magistergetouttamyway/ios-app/src
[4] https://bitbucket.org/magistergetouttamyway/ios-app/src/master/Readme.md

4.5 Functionalities

In the following chapter, the functionalities of the mobile application will be described. The necessary steps to use the application are described first: account registering and authentication. After that, the core features that require user permissions are described.

Authentication

In order to use the application, clients must authorize first with username and password. After a successful sign in a JWT (JSON Web Token) [46] is sent back to them which is later used for requests and socket connections. Furthermore, when a client is connected via socket the socket id is stored in the database so that the server could push data to specific clients. As this is only a prototype the token does not expire. Sign in screen is shown in Figure 10.

In the absence of a user account, a new one can be created tapping on the link “Don’t have an account? Sign up!”. After tapping that button, a new screen will be shown with a sign-up form where all of the fields are required. When a sign up was successful user will be taken back to sign in view and he/she will be able to sign in with the new account. Sign up screen is shown show in Figure 11.

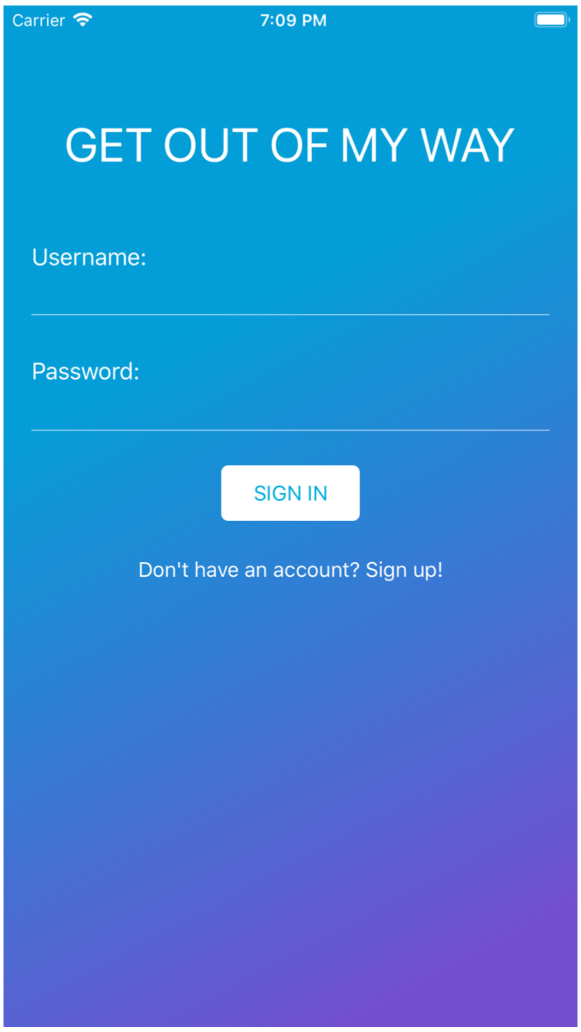


Figure 10. Screenshot of sign in view

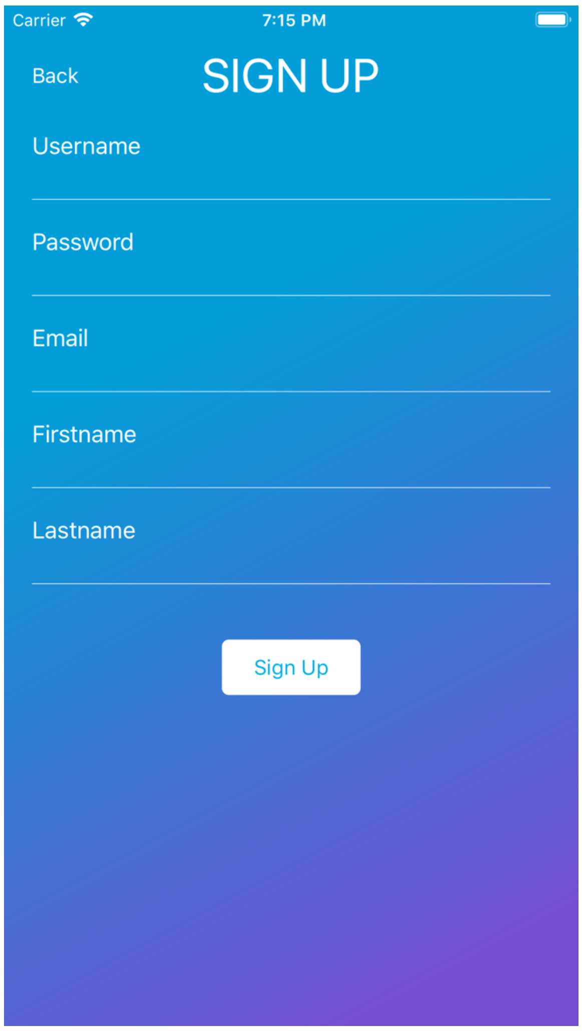


Figure 11. Screenshot of sign up view

Map view

After a successful sign in the map view will be shown to the client. If the client sees the view for the first time it is also required to ask for his/her permission to use his/her current location. If the client decides to not allow it the mobile application loses its purpose and has no real use. If the client granted access to location services he/she will be shown a map focused on the client location, see Figure 12. Furthermore, if there are active emergency vehicles nearby they will also be displayed on the map along with their location and route, see Figure 13.

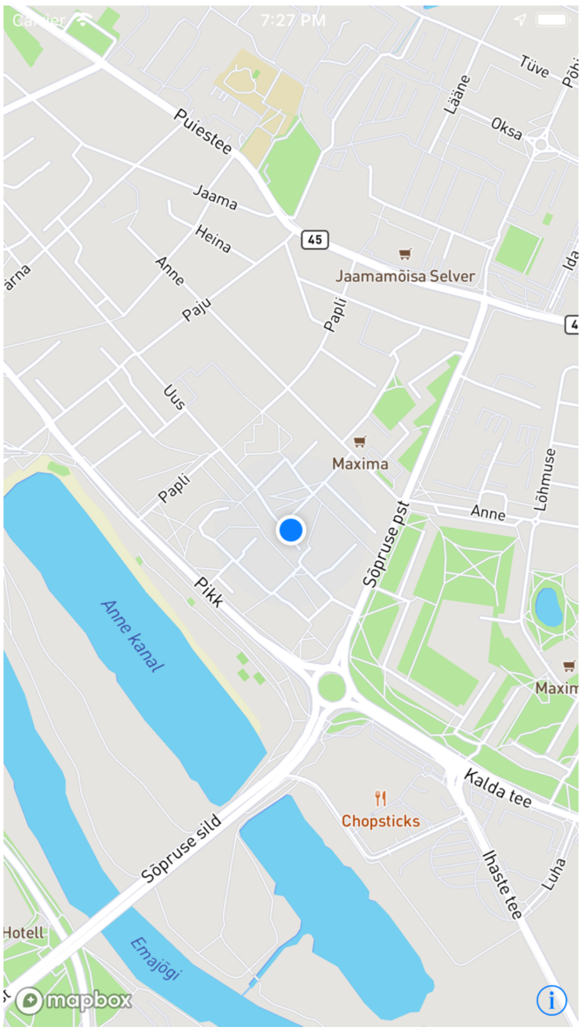


Figure 12. Screenshot of map view after sign in

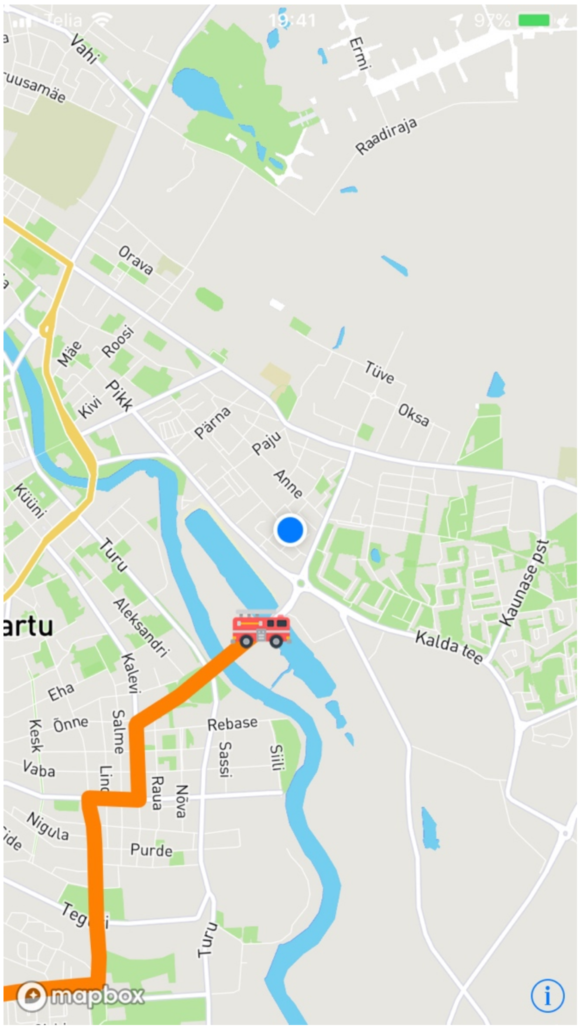


Figure 13. Screenshot of map view with a nearby fire truck

Navigation

In order to start navigating the client has to tap and hold it for a second, also known as a long press, at the destination. After that, at least one route will appear on the map. There may also be alternative routes displayed on the map if possible and the client can select between them by just tapping on the most suitable route. The selected route will be colored blue and alternative routes are colored grey. Along with the routes a button “Start navigation” will be shown on the bottom of the screen, see Figure 14. When client taps on “Start navigation” a navigation view will appear. On the navigation view, the client is shown the next step on the route, option to view the full route, turn sound on and off and if there are problems on the route the client is able to report them. There is also information about the estimated time of arrival and the distance left to the destination, see Figure 15. These features are all provided by the Mapbox SDK.

In order to provide directions for the clients Mapbox underneath uses OSRM (Open Source Routing Machine) [47]. OSRM is a server-based routing engine that combines Open-StreetMap road network data with routing algorithms to find the shortest and fastest route while considering different costs: turn restrictions, traffic lights, braking, accelerating, steep hills for bicycle riders. In order to provide routing data without a large delay, OSRM supports two different pre-processing techniques: Contraction Hierarchies (CH) and Multi-Level Dijkstra (MLD). The main point of pre-processing is to find shortcuts in a graph, in our case on the world map. The pre-processing takes hours, depending on the processing power, but after it is done the queries to find the shortest path take less than a second. By default, OSRM uses the MLD technique and it is recommended for short distances. For very large distances CH should be used instead.

In the mobile application, it works as follows. The user long presses on the destination, user location point, destination point and transportation type are sent to Mapbox Directions API and it responds with the following: one or more available routes as a GeoJSON line, text for turn-by-turn instructions and the distance and estimated travel times. In this application clients cannot choose between different transportation types, it is set to vehicle.

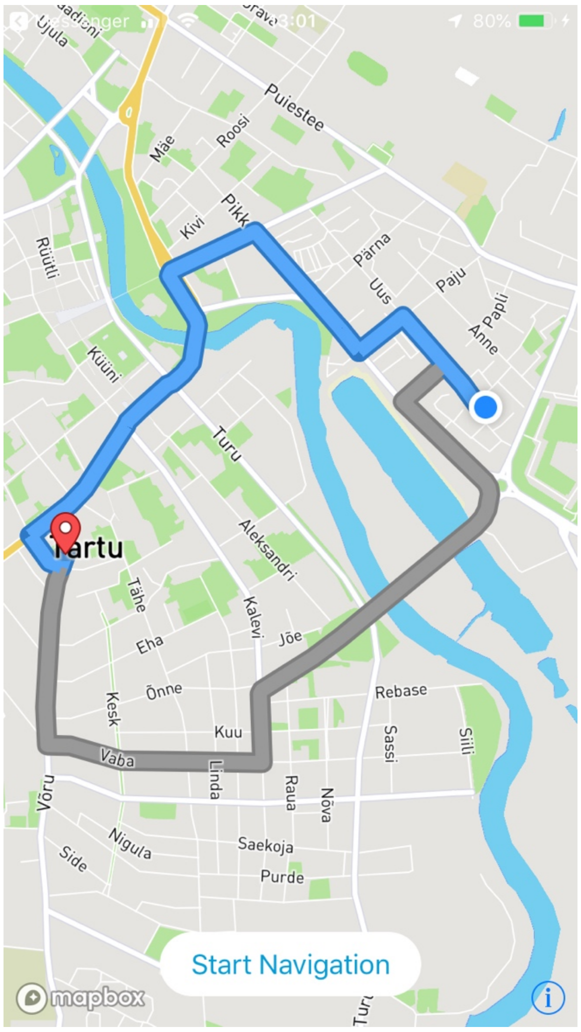


Figure 14. Screenshot of selected destination with an alternative route

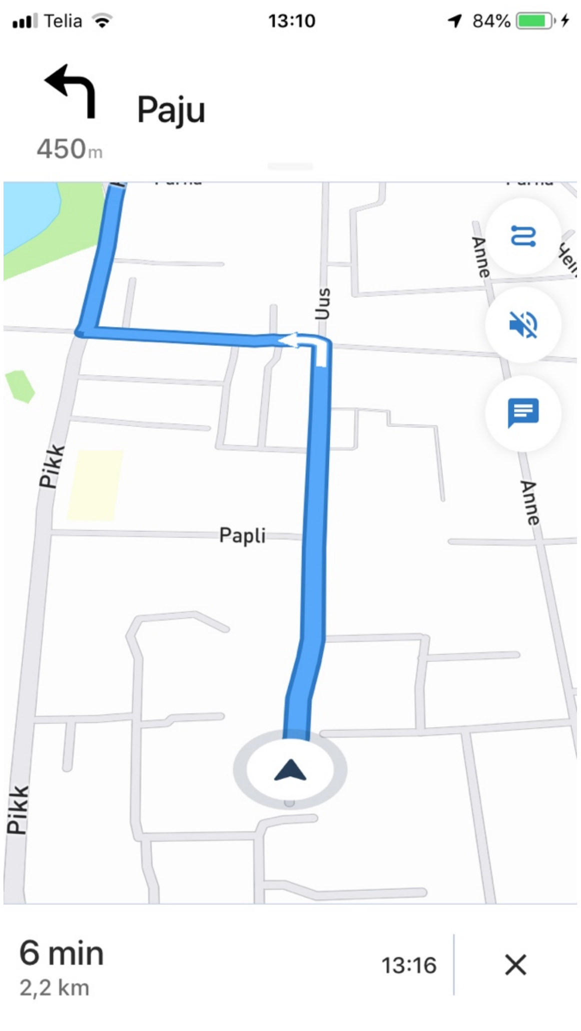


Figure 15 Screenshot of navigation mode with a selected route

When navigation mode is active and emergency vehicles are nearby the client also will be shown the route of the emergency vehicles and emergency vehicles current location. If the route of emergency vehicle intersects with the client route it will be displayed on the map with a warning badge, see Figure 16. The overlapping parts of the client route and the route of an emergency vehicle will be colored red, see Figure 16. The client will see emergency vehicles if the client is in a 4 km range of the emergency vehicle.

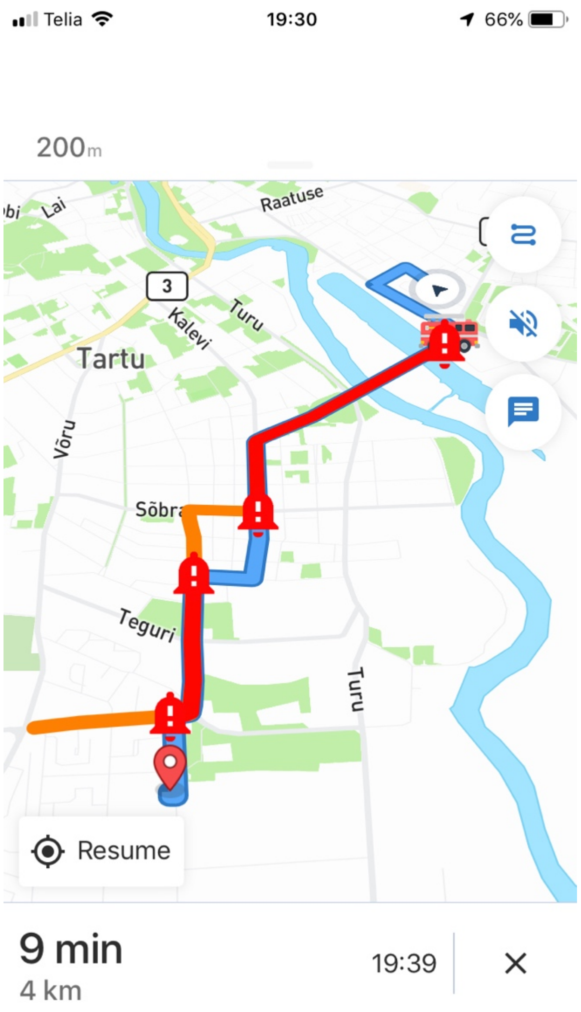


Figure 16. Screenshot of intersecting and overlapping routes of client and fire truck

System requirements for mobile application

The mobile application works on any iPhone that supports iOS version 12.1+ as specified in the non-functional requirements.

Minimal system requirements:

- iPhone 5S or newer model
- Internet access

Hardware requirements for server

In order to measure the performance of the server an Apache HTTP server benchmarking tool [48] was used. The performance was tested on the endpoint which is used for calculating the intersections and overlapping parts of two routes since it contains a function with the complexity of $O(n^2)$. The test was done with the following parameters:

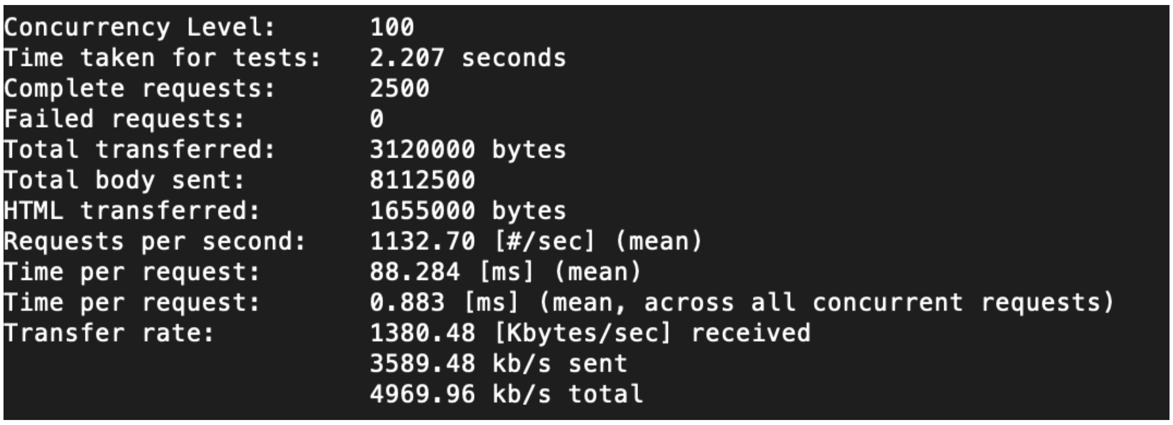
- 100 concurrent requests
- 2500 requests in total

Each request consisted of two routes where one route had 16 points and the other 18 points, in total there were 4 intersection points and 3 overlapping polylines. The test was run on the author's Apple MacBook Pro laptop with the following specifications:

- Processor (CPU): 2,5 GHz Intel Core i7
- Memory (RAM): 16 GB 1600 MHz DDR3
- Operation System: macOS 10.14.3

The test took 2.207s, there were no failed requests and in average 1132.70 requests were sent per second. Each request took 88.284ms on average, see Figure 17. After running similar tests in virtual environments with different specifications, the minimum hardware requirements to run the server without errors are the following:

- Processor (CPU): 1.6 GHz or better
- Memory (RAM): 2GB or more
- Operation System:
    - macOS 10.14.3 and higher
    - Linux



```
Concurrency Level:      100
Time taken for tests:   2.207 seconds
Complete requests:      2500
Failed requests:        0
Total transferred:      3120000 bytes
Total body sent:        8112500
HTML transferred:       1655000 bytes
Requests per second:    1132.70 [#/sec] (mean)
Time per request:       88.284 [ms] (mean)
Time per request:       0.883 [ms] (mean, across all concurrent requests)
Transfer rate:          1380.48 [Kbytes/sec] received
                        3589.48 kb/s sent
                        4969.96 kb/s total
```

Figure 17. Server performance test on the author's laptop

The author estimates that the server could be run without problems on devices with an even lower specification.

4.6 Conclusion

In conclusion, the author developed a server and iOS application. The server is just receiving and broadcasting data without any graphical user interface. The mobile application is only usable for authenticated users who have granted access to his/her location and have access to the internet. It is also possible to create a new account through the application. The main part of the mobile application is built on top of Mapbox SDK to provide navigation features for the client. In addition, the data of nearby emergency vehicles along with intersection points and overlapping parts are displayed on the map. The data of emergency vehicles should be provided by authorities.

5 Validation and testing

In the following chapter, the testing and validation of the system will be described.

5.1 Server

The server is partly covered with unit[5] and end-to-end[6] (e2e) tests. Unit tests validate that the required fields on different objects in the system are indeed required. Furthermore, the authentication part is fully covered with unit tests. While unit tests test single objects or services, e2e test the system as a whole. E2e tests cover all the REST API endpoints of the current system, including user registration, authentication, finding intersections and overlapping's of different routes and querying user information. There are in total of 9 e2e tests and 21-unit tests. The socket controllers' part was manually tested by the author and is covered in the client section.

5.2 Client

The mobile application was tested only manually. In order to do that the author created a small client application that could act like an authority to update the state and location of different emergency vehicles at different times. While the changes were done through the authority client, the mobile application was also open at the same time, making sure that after the status or location of the emergency vehicle was updated, it was also reflected on the mobile application.

5.3 Validation

In order to validate that the software meets the user requirements, the author conducted a test in the real world. It consisted of three drivers, one acting as an emergency vehicle and two acting as drivers. All of the drivers and the author were also connected via Skype so that the author could ask for their locations and send instructions. The driver acting as an emergency vehicle drove in the city of Tartu from one location to another while turning imaginary sirens on and off. For the other two drivers, the test started with account creation and authentication. Having finished the authentication process, the author gave different destination points to the drivers and they started driving. The author monitored the locations of all three drivers and made sure that the locations of clients and the emergency vehicle are

[5] https://bitbucket.org/magistergetouttamyway/backend/src/master/test/unit/
[6] https://bitbucket.org/magistergetouttamyway/backend/src/master/test/e2e/

updated properly. The author also made sure over the Skype that if the clients were close enough to the emergency vehicle it would show up on their map and the intersections and overlapping paths would be displayed in the correct positions.

6 Conclusion

This thesis gave an overview of how the path of emergency vehicles is currently cleared and three different ways were examined how the drivers could be directly informed that an emergency vehicle is nearby. The author decided that the drivers should receive information about nearby emergency vehicles through a mobile application.

The practical work resulted in a prototype of a system that could be used to inform drivers about nearby emergency vehicles. In total two different parts of the system were built: the server and the mobile iOS application. The clients can use the application for navigation while also seeing the nearby emergency vehicles and the places where their routes intersect or overlap.

7 Future perspectives

During this thesis, the author had no real data from authorities. In order to proceed with the project, the first step should be contacting the authorities to make sure that this kind of data exists and see if they are interested in this type of system in the first place. The next step would be adjusting the data model and endpoints to match the data on the authorities side.

Due to the lack of real data, during the thesis, no real user testing was conducted. Future studies should also take a closer look t how the drivers react with and without application.

Finally, this thesis focused on a separate mobile application but actually, it could be used with any client. Since Waze already has over 100 million monthly users and supports two types of moving vehicles, there might be a way how the server could be integrated into Waze. In that way, the clients don't have to choose between different applications.

8 References

[1] Government of the Republic of Estonia, “Riigiteataja,” 28 06 2017. [Online]. Available: https://www.riigiteataja.ee/akt/121062011005. [Accessed 10 05 2019].

[2] Palos Verdes Estates Police Department, “Response Time Explained,” [Online]. Available: http://www.pvestates.org/services/police-department/response-time-explained. [Accessed 10 05 2019].

[3] A. S. Eltayeb, H. O. Almubarak and T. A. Attia, “A GPS Based Traffic Light Pre-emption Control System for Emergency Vehicles,” in *2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)*, 2013.

[4] U.S Department of Transportation, Traffic Signal Preemption for Emergency Vehicles: A Cross-Cutting Study, United States. Federal Highway Administration, 2006.

[5] E. Kwon, K. Sangho and R. Betts, “Route-based Dynamic Preemption of Traffic Signals for Emergency Vehicle Operations,” in *Proceedings of the TRB Annual Meeting*, 2003.

[6] Maricopa Association of Governments, Emergency Vehicle Preemption Study, 2018.

[7] Traffic Systems LLC, “Sonem 2000,” [Online]. Available: http://www.trafficsystemsllc.com/about.htm. [Accessed 10 05 2019].

[8] CENELEC, “Specification of the radio data system (RDS) for VHF/FM sound broadcasting in the frequency range from 87,5 to 108,0 MHz,” 1998. [Online]. Available: http://www.interactive-radio-system.com/docs/EN50067_RDS_Standard.pdf. [Accessed 10 05 2019].

[9] RDS Forum, “RDS features,” [Online]. Available: http://www.rds.org.uk/2010/Glossary-Of-Terms.htm. [Accessed 10 05 2019].

[10] P. Ardell and D. Callahan, “KTH Royal Institute of Technology,” [Online]. Available: https://www.kth.se/en/aktuellt/nyheter/now-drivers-can-hear-ambulances-no-matter-how-loud-their-music-is-playing-1.699714. [Accessed 10 05 2019].

[11] H&E Solutions AB, “EVAM,” [Online]. Available: https://evam.life/products/. [Accessed 10 05 2019].

[12] NHTSA, “Vehicle-to-Vehicle Communications: Readiness of V2V Technology for Application,” 08 2014. [Online]. Available: http://www.nhtsa.gov/staticfiles/rulemaking/pdf/V2V/Readiness-of-V2V-Technology-for-Application-812014.pdf. [Accessed 10 05 2019].

[13] S. Yousefi, M. S. Mousavi and M. Fathy, “Vehicular Ad Hoc Networks (VANETs): Challenges and Perspectives,” in *2006 6th International Conference on ITS Telecommunications*, 2006 6th International Conference on ITS Telecommunications.

[14] European Commission, “The interoperable EU-wide eCall,” 12 05 2019. [Online]. Available: https://ec.europa.eu/transport/themes/its/road/action_plan/ecall_en. [Accessed 10 05 2019].

[15] Ford Media Center, “Prototype Tech Quickly Warns Drivers Of Accidents Ahead, Could Help Clear Way For Emergency Vehicles To Save Lives,” 24 09 2018. [Online]. Available: https://media.ford.com/content/fordmedia/feu/en/news/2018/09/24/prototype-tech-quickly-warns-drivers-of-accidents-ahead--could-h.html. [Accessed 10 05 2019].

[16] GSMA, “The Mobile Market in Numbers,” *The Mobile Economy 2019,* pp. 10-18, 2019.

[17] C. Smith, “15 Interesting Waze Statistics and Facts (2019) | By the Numbers,” 11 05 2019. [Online]. Available: https://expandedramblings.com/index.php/waze-statistics-facts/. [Accessed 11 05 2019].

[18] Waze, [Online]. Available: https://www.waze.com/en. [Accessed 10 05 2019].

[19] Apple, “Apple CarPlay,” [Online]. Available: https://www.apple.com/ios/carplay/. [Accessed 10 05 2019].

[20] Google, “Android Auto,” [Online]. Available: https://www.android.com/intl/en_ca/auto/. [Accessed 10 05 2019].

[21] Apple, “More than 500 models to choose from.,” [Online]. Available: https://www.apple.com/ios/carplay/available-models/. [Accessed 10 05 2019].

[22] StatCounter GlobalStats, “Mobile Operating System Market Share Worldwide,” 04 2019. [Online]. Available: http://gs.statcounter.com/os-market-share/mobile/worldwide. [Accessed 10 05 2019].

[23] Apple, “App Store,” 2019. [Online]. Available: https://www.apple.com/ios/app-store/. [Accessed 10 05 2019].

[24] Google, “Google Play,” [Online]. Available: https://play.google.com/store. [Accessed 10 05 2019].

[25] J. H. Jung, “Method of updating software for vehicle”. U.S Patent 9,274,785, 01 03 2016.

[26] NPD, “The Average Upgrade Cycle of a Smartphone in the U.S. is 32 Months, According to NPD Connected Intelligence,” 2018. [Online]. Available: https://www.npd.com/wps/portal/npd/us/news/press-releases/2018/the-average-upgrade-cycle-of-a-smartphone-in-the-u-s--is-32-months---according-to-npd-connected-intelligence/. [Accessed 10 05 2019].

[27] Bureau of Transportation Statistics, “Average Age of Automobiles and Trucks in Operation in the United States,” [Online]. Available: https://www.bts.gov/content/average-age-automobiles-and-trucks-operation-united-states. [Accessed 10 05 2019].

[28] V. Pimentel and B. G. Nickerson, “Communicating and Displaying Real-Time Data with WebSocket,” *IEEE Internet Computing,* vol. 16, no. 4, pp. 45-53, 07 2012.

[29] I. Fette and A. Melnikov, “The WebSocket Protocol,” 12 2011. [Online]. Available: http://www.rfc-editor.org/info/rfc6455. [Accessed 10 05 2019].

[30] H. Butler, M. Daly, A. Doyle, S. Gilles, S. Hagen and T. Schaub, “The GeoJSON Format,” 08 2016. [Online]. Available: https://www.rfc-editor.org/info/rfc7946. [Accessed 10 05 2019].

[31] Mapbox, “Navigation SDKs for iOS and Android,” [Online]. Available: https://www.mapbox.com/navigation-sdk/. [Accessed 10 05 2019].

[32] Node.js Foundation, “Node.js,” [Online]. Available: https://nodejs.org/en/. [Accessed 10 05 2019].

[33] Node.js Foundation, “Express,” [Online]. Available: https://expressjs.com/. [Accessed 10 05 2019].

[34] Microsoft, “TypeScript,” [Online]. Available: https://www.typescriptlang.org/. [Accessed 10 05 2019].

[35] W3tec, “Express Typescript Boilerplate,” [Online]. Available: https://github.com/w3tecch/express-typescript-boilerplate. [Accessed 10 05 2019].

[36] The PostgreSQL Global Development Group, “PostgreSQL: The World's Most Advanced Open Source Relational Database,” [Online]. Available: https://www.postgresql.org/. [Accessed 10 05 2019].

[37] OSGeo Foundation, “PostGIS,” [Online]. Available: https://postgis.net/. [Accessed 10 05 2019].

[38] “TypeORM,” [Online]. Available: https://typeorm.io/#/. [Accessed 10 05 2019].

[39] “JSON,” [Online]. Available: https://www.json.org/. [Accessed 10 05 2019].

[40] “Turf.js,” [Online]. Available: https://turfjs.org/. [Accessed 10 05 2019].

[41] “Socket.IO,” [Online]. Available: https://socket.io/. [Accessed 10 05 2019].

[42] “REST,” [Online]. Available: https://restfulapi.net/. [Accessed 10 05 2019].

[43] “Yarn,” [Online]. Available: https://yarnpkg.com/en/. [Accessed 10 05 2019].

[44] “OpenStreetMap,” [Online]. Available: https://www.openstreetmap.org/.

[45] “SwiftyJSON,” [Online]. Available: https://github.com/SwiftyJSON/SwiftyJSON. [Accessed 10 05 2019].

[46] “JSON Web Token,” [Online]. Available: https://jwt.io/. [Accessed 10 05 2019].

[47] “OSRM,” [Online]. Available: http://project-osrm.org/. [Accessed 10 05 2019].

[48] “Apache HTTP Server Benchmarking Tool,” [Online]. Available: https://httpd.apache.org/docs/2.4/programs/ab.html.

Appendix

I. Adapted table of accidents involving an emergency vehicle in Estonia during the years 2011-2018

The data was provided by the Estonian Police and Border Guard Board.

Year	Month	Vehicle type	Truck involved	Motorcycle involved	Bicycle involved	Type of accident	County	Municipality	District
2011	jaanuar	Politsei- ja Piirivalveameti sõiduk	0	0	0	Jalakäijaõnnetus	Märkimata	Märkimata	Märkimata
2011	jaanuar	Politsei- ja Piirivalveameti sõiduk	0	0	0	Ühesõidukiõnnetus	Märkimata	Märkimata	Märkimata
2011	veebruar	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Märkimata	Märkimata	Märkimata
2012	jaanuar	Kiirabisõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Lasnamäe linnaosa
2012	jaanuar	Kiirabisõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Toila vald	Märkimata
2012	jaanuar	Politsei- ja Piirivalveameti sõiduk	0	0	1	Kokkupõrge	Lääne-Viru maakond	Vinni vald	Märkimata
2012	september	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Saue vald	Märkimata
2012	november	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Sillamäe linn	Märkimata
2013	aprill	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Tartu maakond	Tartu linn	Tartu linn
2013	mai	Kiirabisõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Haabersti linnaosa
2013	august	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Kristiine linnaosa
2013	august	Kiirabisõiduk	0	0	1	Kokkupõrge	Harju maakond	Tallinn	Nõmme linnaosa
2014	märts	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Tartu maakond	Tartu vald	Tartu linn
2014	aprill	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Toila vald	Märkimata

2014	august	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Mustamäe linnaosa
2014	oktoober	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Kesklinna linnaosa
2014	oktoober	Kiirabisõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Mustamäe linnaosa
2015	veebruar	Päästeameti sõiduk	1	0	0	Ühesõidukiõnnetus	Rapla maakond	Rapla vald	Märkimata
2015	mai	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Narva linn	Märkimata
2015	mai	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Narva-Jõesuu linn	Märkimata
2015	mai	Politsei- ja Piirivalveameti sõiduk	0	0	0	Ühesõidukiõnnetus	Tartu maakond	Luunja vald	Märkimata
2015	juuli	Politsei- ja Piirivalveameti sõiduk	0	0	0	Ühesõidukiõnnetus	Harju maakond	Saue vald	Märkimata
2015	juuli	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Lasnamäe linnaosa
2015	juuli	Politsei- ja Piirivalveameti sõiduk	0	1	0	Muu liiklusõnnetus	Saare maakond	Lääne-Saare vald	Märkimata
2015	august	Päästeameti sõiduk	0	0	0	Jalakäijaõnnetus	Harju maakond	Tallinn	Põhja-Tallinna linnaosa
2015	oktoober	Päästeameti sõiduk	1	0	0	Kokkupõrge	Rapla maakond	Märjamaa vald	Märjamaa alev
2016	jaanuar	Kiirabisõiduk	0	0	0	Kokkupõrge	Lääne-Viru maakond	Rakvere vald	Märkimata
2016	veebruar	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Põhja-Tallinna linnaosa
2016	aprill	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Mustamäe linnaosa
2016	aprill	Kiirabisõiduk	0	0	0	Jalakäijaõnnetus	Harju maakond	Tallinn	Lasnamäe linnaosa
2016	mai	Politsei- ja Piirivalveameti sõiduk	0	0	0	Ühesõidukiõnnetus	Viljandi maakond	Mulgi vald	Halliste alevik
2016	juuni	Politsei- ja Piirivalveameti sõiduk	0	0	0	Ühesõidukiõnnetus	Harju maakond	Tallinn	Pirita linnaosa

2016	juuli	Päästeameti sõiduk	1	0	0	Kokkupõrge	Harju maakond	Tallinn	Nõmme linnaosa
2016	august	Kiirabisõiduk	0	0	0	Kokkupõrge	Harju maakond	Vasalemma vald	Märkimata
2016	september	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Viljandi maakond	Paistu vald	Märkimata
2016	september	Kiirabisõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Nõmme linnaosa
2016	oktoober	Politsei- ja Piirivalveameti sõiduk	0	0	0	Muu liiklusõnnetus	Pärnu maakond	Häädemeeste vald	Märkimata
2016	oktoober	Kiirabisõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Lasnamäe linnaosa
2016	detsember	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Maardu linn	Märkimata
2016	detsember	Politsei- ja Piirivalveameti sõiduk	1	0	0	Kokkupõrge	Rapla maakond	Märjamaa vald	Märkimata
2017	jaanuar	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Narva linn	Märkimata
2017	jaanuar	Kiirabisõiduk	0	0	0	Ühesõidukiõnnetus	Harju maakond	Tallinn	Lasnamäe linnaosa
2017	veebruar	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Jõhvi vald	Jõhvi linn
2017	märts	Politsei- ja Piirivalveameti sõiduk	0	0	0	Ühesõidukiõnnetus	Jõgeva maakond	Mustvee vald	Märkimata
2017	aprill	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Lasnamäe linnaosa
2017	juuli	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Narva linn	Märkimata
2017	august	Politsei- ja Piirivalveameti sõiduk	0	0	0	Ühesõidukiõnnetus	Harju maakond	Nissi vald	Märkimata
2017	september	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maakond	Tallinn	Kristiine linnaosa
2017	september	Politsei- ja Piirivalveameti sõiduk	0	0	0	Ühesõidukiõnnetus	Põlva maakond	Põlva vald	Põlva linn
2017	oktoober	Politsei- ja Piirivalveameti sõiduk	0	0	0	Jalakäijaõnnetus	Harju maakond	Tallinn	Mustamäe linnaosa

2017	detsem-ber	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maa-kond	Lääne-Harju vald	Vasa-lemma al-evik
2017	detsem-ber	Kiirabisõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Toila vald	Paate küla
2017	detsem-ber	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Toila vald	Paate küla
2018	märts	Päästeameti sõiduk	1	0	0	Kokkupõrge	Harju maa-kond	Tallinn	Lasnamäe linnaosa
2018	aprill	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Alutaguse vald	Lipniku küla
2018	aprill	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Ida-Viru maakond	Narva linn	Märkimata
2018	aprill	Politsei- ja Piirivalveameti sõiduk	0	1	0	Kokkupõrge	Pärnu maa-kond	Tori vald	Tammiste küla
2018	mai	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maa-kond	Tallinn	Kesklinna linnaosa
2018	juuni	Kiirabisõiduk	0	0	0	Kokkupõrge	Harju maa-kond	Tallinn	Kesklinna linnaosa
2018	juuni	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Põlva maa-kond	Põlva vald	Vooreküla
2018	juuni	Kiirabisõiduk	0	0	1	Kokkupõrge	Pärnu maa-kond	Saarde vald	Ristiküla
2018	juuli	Kiirabisõiduk	0	0	0	Kokkupõrge	Harju maa-kond	Harku vald	Vääna-Jõesuu küla
2018	oktoober	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Harju maa-kond	Tallinn	Kristiine linnaosa
2018	detsem-ber	Politsei- ja Piirivalveameti sõiduk	0	0	0	Kokkupõrge	Valga maa-kond	Valga vald	Märkimata

II. License

Non-exclusive licence to reproduce thesis and make thesis public