Czech Technical University in Prague
Faculty of Electrical Engineering
Department of telecommunication engineering

Diploma thesis

Resource allocation for vehicular cloud computing

Bc. Nikolay Volkov

Supervisors: doc. Ing. Zdeněk Bečvář, Ph.D.
prof. Ray-Guang Cheng

Study Programme: Communications, Multimedia, Electronics

Field of Study: Networks of Electronic Communication

January 8, 2018

CTU
CZECH TECHNICAL UNIVERSITY IN PRAGUE

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Volkov Nikolay** Personal ID number: **406112**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Telecommunications Engineering**

Study program: **Communications, Multimedia, Electronics**

Branch of study: **Networks of Electronic Communication**

II. Master’s thesis details

Master’s thesis title in English:

Resource allocation for vehicular cloud computing

Master’s thesis title in Czech:

Přidělování prostředků v cloudu automobilů

Guidelines:

Study concept of computing resource sharing in vehicular cloud computing. Assess usability of different wireless technologies for transmission of sensors? and users? data among vehicles and between vehicles and infrastructure. Select suitable approach for joint allocation of radio resources and computing resources for data processing. Assess the selected approach and propose its extension to improve its performance.

Bibliography / sources:

[1] M. Vondra, Z. Becvar, P. Mach, 'Vehicular network-aware route selection considering communication requirements of users for ITS,' IEEE Systems Journal, vol. PP, no. 99, November 2016
[2] T. Adhikary, A. K. Das, M.A. Razzaque, A. Almogren, M. Alrubaian, M. M. Hassan, 'Quality of Service Aware Reliable Task Scheduling in Vehicular Cloud Computing,' Mobile Networks and Applications, vol. 21, no. 3, June 2016.
[3] R. Yu, Y. Zhang, S. Gjessing, W. Xia, K. Yang, ?Toward cloud-based vehicular networks with efficient resource management,? IEEE Network, vol. 27, no. 5, pp. 48-55, October 2013.
[4] M. Eltoweissy, S. Olariu, M. Younis, ?Towards Autonomous Vehicular Clouds,? Ad Hoc Networks, 2010.

Name and workplace of master’s thesis supervisor:

doc. Ing. Zdeněk Bečvář, Ph.D.,

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **24.08.2017** Deadline for master's thesis submission: **09.01.2018**

Assignment valid until: **18.02.2019**

doc. Ing. Zdeněk Bečvář, Ph.D. Supervisor’s signature	Head of department’s signature	prof. Ing. Pavel Ripka, CSc. Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

V Praze dne 08. 01. 2018 .....

Abstrakt

V blízké budoucnosti bude každé vozidlo pravidelně informovat svá sousední vozidla o jejich poloze, rychlosti, výpočetních schopnostech a informacích o chování. Zprávy nesoucí takové informace sníží pravděpodobnost nehod, zlepší bezpečnost řidičů a cestujících, a umožní další multimedialní služby. Hlavními kandydáty jsou dnes IEEE 802.11p a LTE-Advanced s komunikací mezi zařízeními (LTE-D2D). V této práci navržen algoritmus plánování úloh v LTE-Advanced, který bude poskytovat sdílení zdrojů ve cloudu automobilů z hlediska pravděpodobnosti a propustnosti systému. Algoritmus provádí hledáni automobilů z hlediska výpočetních parametrů a času zpracováni tasků s predikci jejich pochybu. Navrženy algoritmus je spolehlivější a výkonnější, průměrně dosažené zlepšení je o 31% větší pravděpodobnost úspěchu tasku, nebo zvýšení propustnosti systému o 31 task za minutu.

Abstract

In a near future, each vehicle will be able to periodically broadcast information to their neighbors vehicles about their position, speed, computational capability and their behavioral information. The messages carrying such information will reduce the probability of accidents, improve the safety of drivers and passengers and allow additional multimedia services. The main candidate technologies for this scope today are IEEE 802.11p and LTE-Advanced with device-to-device communications (LTE-D2D). The aim of this study is to develop the task scheduling algorithm in LTE-Advanced, that will provide resource sharing

Keywords *— Vehicular communication, Task processing, Scheduling, Algorithm, Mobile networks*

in vehicular cloud computing in terms of probability and system throughput. The algorithm searches appropriate vehicles in terms of the performance characteristics and computation time with the movement and connection prediction. The proposed algorithm is more reliable with better performance, the average improvement is +31% of success execution task or +31 tasks per minute of system throughput.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing evolution of computer science, in particular the rapid technological innovation in telecommunications, has allowed humankind to move from the telegraph to satellites and cellphones in only one century. Simultaneous development in different types of wireless communications have brought us from Morse code to Navigation systems, the main purpose of which is to connect people around the world. Among the discoveries that have been made recently is VANET (vehicular ad-hoc network) technology. There are great expectations that VANET will be able to improve road safety, managing transport efficiently and provide a wide variety of services for passengers and drivers.

In general, VANET is a wireless technology enabling a vehicle to communicate with other vehicles and the surrounding environment. VANET is capable of supporting a wide range of services by using different communication media such as microwave transmission and communication satellites. Currently vehicles on the road carry many of on-board computing devices having small-scale processing capabilities. A good number of research efforts have recently been undertaken that are looking for ways to make driving experience safer and smarter through the use VANET applications [1]. As vehicles increasingly become equipped with advanced sensors, a number of applications, ranging from inter vehicular driver healthcare and safety related applications to highway dynamic congestion management applications, are being developed using the huge deployment of data that is being obtained from these sensors. To be of any use, the huge amount of data generated by vehicular sensors needs to be processed immediately. For that reason has appeared (see fig.1.1) Vehicular Cloud Computing (VCC) [1].

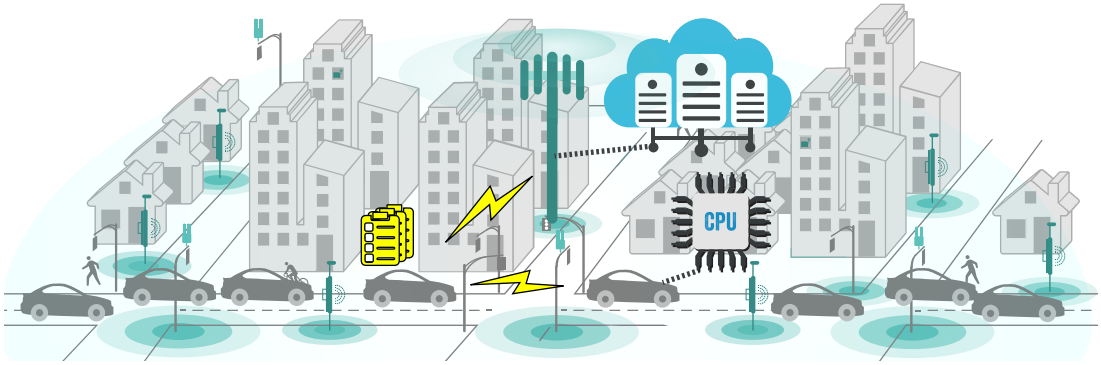


Figure 1.1: Concept of vehicular cloud computing

The difference between VCC and traditional cloud computing is that the VCC infrastructure and the resources are not fixed to a geographic location. The topology changes dynamically therefore connectivity among the vehicles is neither reliable nor persistent. Vehicles have low computing capabilities and a VCC allows to capitalize by accessing spare and idle resources available on neighbour vehicles. In such an environment, possibility to select the neighborhood computing resources in such a manner that the application can be executed reliably within a delay deadline is very essential. The main challenge in VCC is the development of real-time services which can quickly analyze a huge volume of sensory data produced from a variety of on-board sensors such as speedometers, engines, gas tanks, cameras, outside temperature sensors, and so on. In the literature, most of the research on VCC focuses on short time connection or real time applications tasks [2]. Most of them are safety applications, that generate alarm or warning tasks [3]. It has successfully been implemented and tested in several countries like USA, Japan and European Union [4]. A new idea to proceed the big amount of data and reducing connection loss due to dynamical changing topology has been developed by researchers of Bangladesh and Saudi Arabia. They designed an algorithm for an infrastructure less VCC system, named the QoS Aware Reliable Task Scheduling (QARTS) system [1]. QARTS algorithm has been developed and all data are computing into the clusters. It can be implemented when there are heavy traffic or vehicles have a low velocity and high vehicle density on a road, otherwise communication delay in cluster is critical.

In this study, we have developed new task scheduling algorithm with optimal task sorting, that compare and identify the suitable vehicle for different tasks from all available neighbour vehicles. The main objective was to improve VCC services reducing connection loss and increasing the number of successfully executed tasks per time. The major contributions of this study can be summarized as follows:

1. Communication with infrastructure is a particular case in dynamically changing network, where one node has fixed position. For more common case has considered an infrastructureless network using only V2V (vehicle to vehicle) communication. 2. To increase the success probability of execution a task and a throughput of executed tasks in time unit have been formulated as a main objective of the study. 3. Task-scheduling algorithm has been developed to reach all mentioned above goals. 4. The results of a simulations, carried

out on MATLAB, depict that significant performance improvements have been achieved by the proposed algorithm mainly compared to current state-of-the-art model QARTS model [1] and another solutions, that will be described further.

The remainder of the study is organized as follows. The theoretical review and state of-the-art work on task scheduling in VCC is described in Section 2. In Section 3, the assumptions and the system model are demonstrated. The proposed solution of scheduling tasks with all functions presented in Section 4. The performance evaluation of our model is demonstrated in Section 5. Section 6 is Conclusion and future work.

Chapter 2

Literature Review

2.1 The main concept of vehicular cloud computing

The vast number of vehicles on streets, roadways and parking lots will be treated plentiful and underutilized computational resources,which can used for providing public services. Every day, many vehicles, spend hours in a parking garage, driveway or parking lot. The idle vehicles are a vast unexploited resource, which is currently simply wasted. These features make vehicles the perfect candidates for nodes in a cloud computing network. Some vehicle owners may agree to rent out excess on board resources, similar to the holders of huge computing and storage facilities who rent out their excess capacity and benet economically. The travelers normally park their cars in airport parking spaces while they are traveling. The airport authority will power the vehicles' computing resources and allow for on demand access to this parking garage data center. Similarly, the drivers stuck in trafc congestion will agree donate their on board computing resources to help city trafc authorities run complex simulations designed to remove congestion by rescheduling the trafc lights of the city [5].

The Vehicular Cloud Computing can be dened as a group of largely autonomous vehicles whose corporate computing, sensing, communication and physical resources can be coordinated and dynamically allocated to authorized users [6]. The benefits of using the Vehicular Cloud instead of the Internet Cloud are reduced communication delay, reduced spectrum costs and amply expanded range of applications. In the new scenario, the mobiles upload to the Internet Cloud only the content of global, long lasting value and delegate to it only those tasks that are too complex or too energy consuming to process in the Vehicular Cloud [7]. In the Vehicle Cloud, the leading applications are safe driving, urban sensing, content distribution, mobile advertising and intelligent transportation. For example, vehicles pick up information via sensors (congestion, pavement conditions, surrounding cars, environment video clips, advertisements and so on) [7].

2.2 Wireless technologies

The vehicles provide communication with each other directly V2V (vehicle to vehicle) or indirectly through infrastructure (. Today the main candidate technologies for these com-

munications are IEEE 802.11p or WAVE (Wireless Access in the Vehicular Environments) and LTE-Advanced [8].

The IEEE 802.11p has been developed in 2010 [9]. Given the large experimentation records and the large number of devices already available on the market, the main advantage of this technology is that it appears mature for a large scale deployment and still remains the main standard for V2V communications [10]. The Main concerns of WAVE:

- high level of errors when heavy traffic conditions happen
- the lack of perspective further improvements for the standard
- for deployment necessary new expensive devices as a RSU.

The IEEE 802.11p using carrier sensing multiple access (CSMA/CA) with collision avoidance at the medium access control (MAC) layer [10]. It supports 5.850 - 5.925 GHz spectrum in North America [11] and operates in the same band in Europe [12]. On the physical (PHY) layer data rate varies between 3 and 27 Mb/s depending on the adopted MCS ((Modulation Coding Scheme) [9].

While IEEE 802.11p had been the most popular standard for VCC and had been used in appropriate applications until 2016, at the end of that year essential update of 3GPP (3rd Generation Partnership Project) had been announced. LTE-A starts support direct communication between vehicles in Release 14 [13]. Driven by the already available and almost ubiquitous coverage of cellular systems and by the advances in the direct communications among devices, LTE is becoming a new option for connected vehicles in the recent years [8]. It enables to exploiting the same hardware as for cellular networks, which makes VCC easy to implement and not expensive technology. The signicant advantages of LTE-A (release 14):

- cost, vehicles are already equipped with a cellular interfaces
- all specifications are continuously updated and improve each release
- cellular base stations are already deployed, new equipment is not necessary to setup

At the MAC and PHY layers LTE-A is based on SC-FDMA (single carrier frequency division multiple access). Advanced coding techniques and an almost continuous variation of MCS combinations contribute to a higher reliability and range with respect to IEEE 802.11p [8]. The data rate depends on a distance between vehicles, it is 5.5 Mbit/s for distance until 10 m and for 50 m distance is 1.9 Mbit/s [14]. Operation bands for communication between vehicles considered in 5.9 GHz and between vehicle and base station 2 GHz [15].

2.3 Existing algorithms and their methodology

There are many research teams working and developing algorithms for a young but rapidly developing VANET. Some of them explore connection between vehicles and RSU (Road side unite). RSUs support cooperative and distributed applications in which vehicles and RSUs

work together to coordinate actions and to share and process several types of information. Have been exploited RSUs to route packets between any source and destination in the VANET. It has been the first attempt to use the infrastructure backbone to efficiently route packets to very far locations in VANETs by using geographic forwarding [16]. Another ones had been exploited the presence of RSUs to reduce the load on vehicles and to hide the complexity of getting the required data on roads [17]. Had been proposed a new smart parking algorithm for large parking lots through vehicular communication. The proposed scheme can provide the drivers with real-time parking navigation service, intelligent anti theft protection, and friendly parking information dissemination [18]. Another have been proposed a cost efficient RSUs deployment scheme to guarantee that vehicles any place could communicate with RSUs in certain driving time with short-time update certification needed for signing to a RSU [19]. Even some studying drivers behaviour in intersections, that is not only influenced by the rules of priority in the intersection but also by the design of the intersection as well as the behaviour of other road users [20]. Alternative teams develop algorithms for applications and routing protocols. They have been presented a combined transmission range and packet generation rate control algorithm which takes into account the safety of the vehicles and maximizes the control channel utilization [21]. But with the development of technology there is an increasing problem about optimal resource allocation. The main and significant study have been provided researchers from Bangladesh and Saudi Arabia. They developed a reliable task-scheduling model in VCC environment with aiming to minimize execution time so as to satisfy task deadlines. They have achieved better performances in time and reliability domains compared to the state-of-the-art approaches [1].

Chapter 3

System model

This chapter presents several methods for describing designed algorithm. It shows, through the use of examples, pseudo-codes, flowchart and figures how the algorithm works and describes solutions to encountered problems. There will be many definitions, special symbols and keywords that have to simplify algorithm understanding.

The main aim of the algorithm is to allocate the generated tasks by the own vehicle to neighbor vehicles in dynamically changing topology. Note that the own vehicle can also execute the tasks in case of better performance properties in comparison with other vehicles and available performance resources.

Proposed algorithm does not use any CH (cluster head) for re-transmitting messages unlike many others. As advantage we don't need to rebuild a cluster structure in case of network structure changes, is not necessary to collect and exchange information for cluster formation. Moreover, for common cluster-based algorithms delayed response time is critical. This is due to the fact that in order to get the executed task back, we must wait for several transmissions. The minimum response time will be at least the time it takes for the transmission to travel from one vehicle to the cluster, and from the cluster to a nearby vehicle and back.

3.1 Vehicles

In this section we are presenting characteristics and information about all vehicles $\mathbf{A}_a^{1\times K}$ (see Fig. 3.1) in some area, their types and properties.

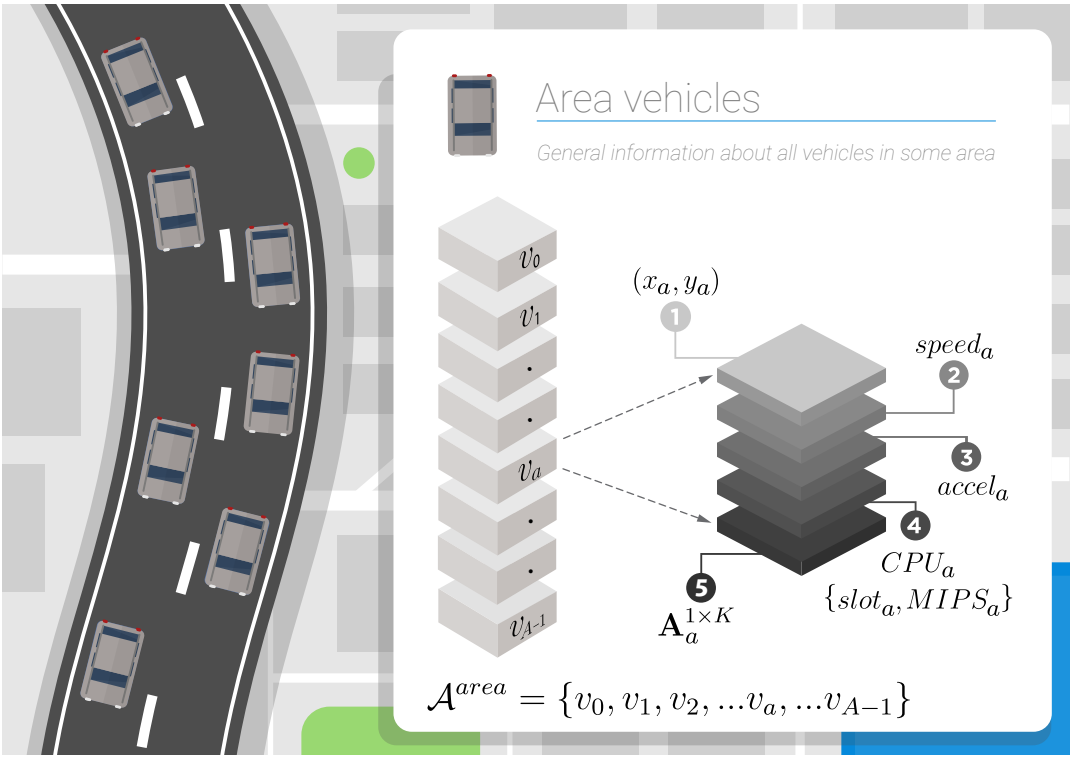


Figure 3.1: Vehicles' parameters in some area

We consider that all vehicles are able to determine their own positions. These parameters can be fulfilled by equipping vehicles with GPS receivers, which is a plausible assumption given the rapid diffusion of this technology in the automotive industry [22].

Definition 3.1.1. *We use* (x_a, y_a) *positions for* a^{th} *vehicle, which means, that the Cartesian coordinate system in two dimensions was chosen (also called an orthogonal coordinate system) for all further calculations. The value of* x *is called the* x*-coordinate or abscissa and the value of* y *is called the* y*-coordinate or ordinate.*

If necessary we can transform it to Geographic coordinate systems [23], because many researchers use one of the best known geographic routing protocols Greedy Perimeter Stateless Routing (GPSR), that uses the Global Positioning System (GPS) coordinates [24] [25].

Definition 3.1.2. *Let* v_a *be* a^{th} *vehicle from all vehicles* $\mathcal{A}^{area}$ *in a certain area, then such set we can describe as:*

$$\mathcal{A}^{area} = \{v_0, v_1, v_2, ...v_a, ...v_{A-1}\}, where\ a \in \langle 0, A-1\rangle,\ a \in \mathbb{N},\ A \in \mathbb{N}$$

3.1.1 Speed and velocity

Each individual vehicle v_a has various characteristics of *speed*, *acceleration/deceleration*, *CPU* (Central processing unit), $\mathbf{A}_a^{1\times K}$ (drivers' behaviour matrix) and driving style. Let us dwell a little dipper to understand correctly further functions of proposed algorithm.

Definition 3.1.3. *$speed_a$ is a speed of a^{th} vehicle shows instantaneous speed* [26], other words the speed of a vehicle at a particular moment (instant) in time.

$$speed_a - \text{an instantaneous speed of } a^{th} \text{ vehicle in meters per second } [\,m/s\,]$$

3.1.2 Acceleration

Definition 3.1.4. *$accel_a$ is an acceleration of a^{th} vehicle, shows how quickly the velocity of an object changes. So, the acceleration is the change in the velocity, divided by the time. Deceleration is the opposite of acceleration. It is the rate at which an object slows down. Deceleration is the final velocity minus the initial velocity divided by that time period, with a negative sign in the result because the velocity is dropping. So for simplicity we use the term acceleration with negative sign for deceleration in further calculations.*

$$accel_a - \text{an instantaneous acceleration of } a^{th} \text{ vehicle}$$
$$\text{in meters per second squared } [\,m/s^2\,]$$

3.1.3 Performance resources

Definition 3.1.5. *CPU_a is an information about performance resources of a^{th} vehicle. The CPU performance include size of core frequency, number of cores and threads per core and others parameters* [27]. *Cores of CPU allow parallel processing tasks* [28], *for our model only one task can be executed in one time and instead of core is used simple abstraction slot. MIPS (million instructions per second) are used as the processing power of the slot.*

$$CPU_a = \{slot_a, MIPS_a\}, \tag{3.1}$$
$$slot\ [-],\ MIPS \text{ in million instructions per second } [\,\mathrm{MI}/s\,]$$

3.1.4 Drivers' behaviour matrix

Intersections are one of the major locations where safety is a big concern to drivers, it is also one of the main criteria to predict the movement of vehicle as accurate as possible. Let us consider another important property of a^{th} vehicle - drivers' behaviour matrix $\mathbf{A}_a^{1\times K}$. That information we collect from RSU[17]. That matrix is collecting all information about behaviours of vehicle and then can be used as a part of prediction of vehicle.

3.1.5 Own vehicle

First of all we need to define a considering vehicle, that will generate packets to be executed. That special vehicle will be the Own vehicle (see Fig. 3.2). The own vehicle has index 0 and $v_0 \in \mathbf{A}_a^{1\times K}$. The wide range of possible applications of the vehicular technology generates tasks x, which the own vehicle needs to be executed.

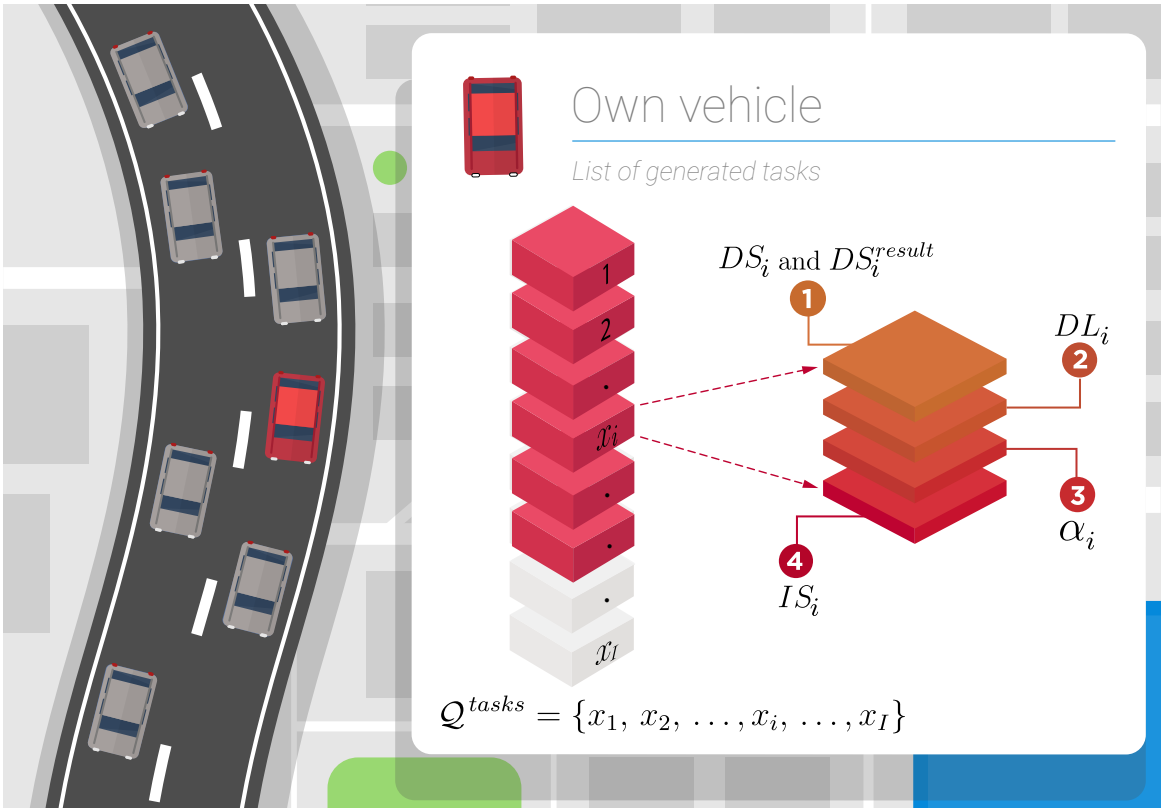


Figure 3.2: Common characteristics of vehicles

Definition 3.1.6. *Let x_i be i^{th} generated task in a list of tasks $\mathcal{Q}^{tasks}$, then such list we can define as:*

$$\mathcal{Q}^{tasks} = \{x_1, x_2, \ldots, x_i, \ldots, x_I\}, where\ i \in \langle 1, I \rangle,\ i \in \mathbb{N},\ I \in \mathbb{N}$$

Each generated task has unique properties, such as DS (data size), DL (deadline), IS (instruction set) and α (see Fig. 3.2). Let us consider each parameter in more detail.

3.1.5.1 Data Size

Definition 3.1.7. *DS_i is a size of i^{th} task. This parameter required to calculate transmission time for sending task for further calculation. Simultaneously, DS_i^{result} is a resulting size of i^{th} task.This parameter required to calculate the time result transmission time for sent i^{th} task.*

$$DS_i,\ DS_i^{result} - \text{size of } i^{th} \text{ task in bytes [B]}$$

3.1.5.2 Deadline

Definition 3.1.8. *DL_i is a deadline of i^{th} task, other words the final time that i^{th} task must be completed by some car from a certain area.*

$$DLi - \text{deadline of } i^{th} \text{ task in seconds } [\,s\,]$$

3.1.5.3 Instruction set

Definition 3.1.9. *The IS provides commands to the CPU, to tell it what it needs to do. For algorithm purpose was considered to use IS as an amount instructions that needed to execute some task, other words that parameter demonstrates complexity of a task. Let IS_i a number of instructions needed to execute i^{th} task, then*

$$IS_i - \text{amount instructions for } i^{th} \text{ task in million instructions } [\,\text{MI}\,]$$

3.1.5.4 Priority

Definition 3.1.10. *The parameter α is a relative priority between response time and success probability. Some applications produce tasks, that are delay-tolerant but require higher probability to being processed successfully. Some other real-time applications might emphasize the total time rather than success probability.*

$$\alpha_i - \text{relative priority between response time and success probability for } i^{th} \text{ task } [\,\text{-}\,]$$

3.1.6 Neighbour vehicles

We have already mentioned an own vehicle and all vehicles in a certain area. The neighbour vehicle is a such vehicle that within a transfer range of own vehicle and vehicles can communicate with each other. A hop means number of different nodes (vehicle/RSU) a task (packet) has to go through in order to reach its final destination address. More over network coverage area is often much larger than radio range of a vehicle, so in order to reach some destination node can use other vehicles as relays. This type of communication is known as multi-hop routing in wireless mesh networks [29].

Multi-hop allows to use available resources of VANET network more efficiently. But on other side we get much more energy efficient than multi-hop routing [30], end-to end delay, lower packet loss [29] and so on. In this thesis, we considered algorithm in single-hop network.

Definition 3.1.11. *All vehicles that appear after scanning a network by the own vehicle* v_0 *will be neighbour vehicles* $\mathcal{N}^{neighb}$*. Let* v_j *is a* j^{th} *neighbour vehicle (see Fig.* 3.3*)* from all neighbour vehicles $\mathcal{N}^{neighb}$ in a certain area $\mathcal{A}^{area}$ after scanning by the own vehicle v_0, so a set of all neighbour vehicle can be defined as

$$\mathcal{N}^{neighb} = \{v_1, v_2, \ldots, v_j, \ldots, v_N\} \tag{3.2}$$
$$\text{where } j \in \langle 1, N \rangle;\ j \in \mathbb{N},\ N \in \mathbb{N};\ \mathcal{N}^{neighb} \subseteq \mathcal{A}^{area}$$

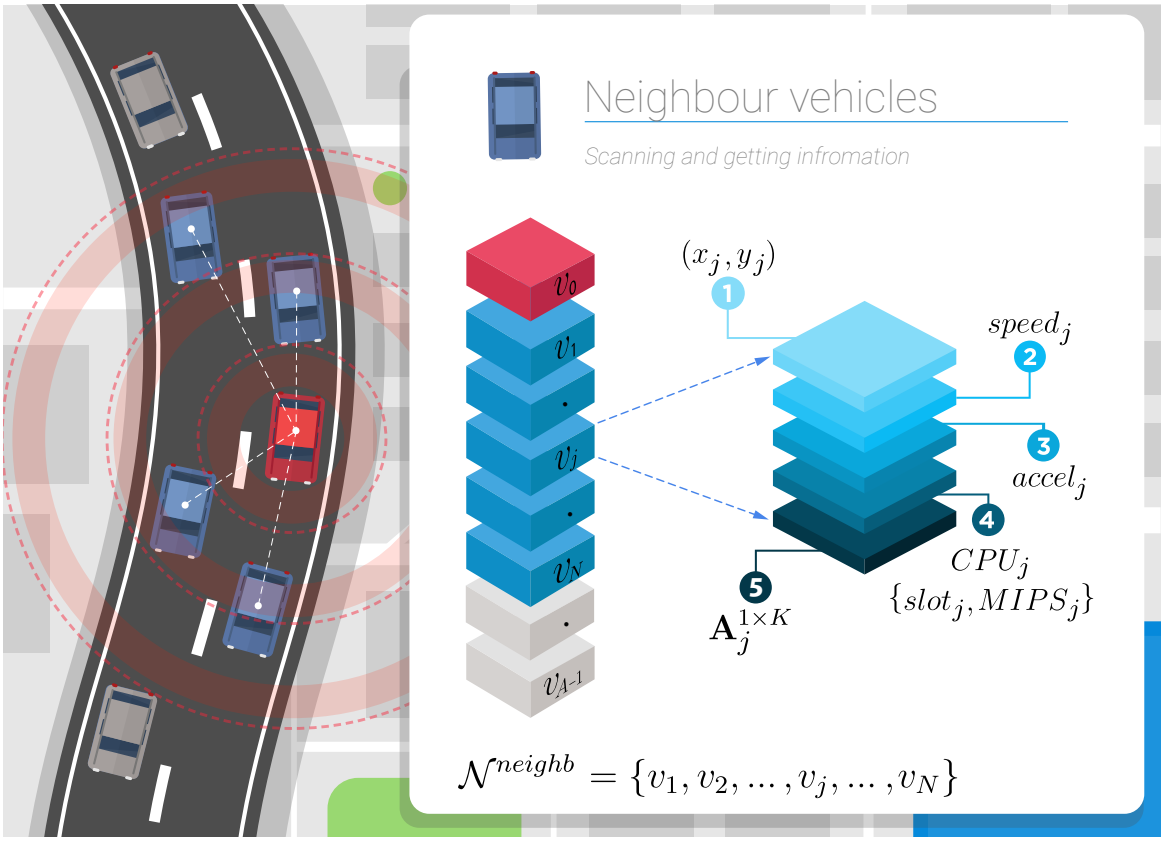


Figure 3.3: Characteristics of Neighbour vehicles

3.2 Assumptions

All scenarios have been provided on infrastructure less simple grid road map, which is a type of city plan where roads run at right angles to each other, forming a grid. Only one way roads are considered for fair comparison with another existing algorithms. Vehicles use only V2V communication type. The vehicular cloud computing environment has been considered, where sensing and generating tasks are providing only by the own vehicle, but computing services can be implemented by each idle vehicle. For more realistic results according to the

general speed limitations in EU (European Union) maximal vehicle's speed for urban roads is 50 km/h [31]. All assumptions can be briefly written in the following way

- grid road map with one way roads
- only V2V communication (infrastructure less network)
- only the own vehicle generates tasks, but each vehicle including the own vehicle can execute them
- maximal speed is 50 km/h [31]

Chapter 4

Proposed algorithm

4.1 Flow chart

In this subsection we considered the functional of algorithm without math functions, just to understand high level concept of proposed method. We already know all parameters required to find optimal vehicle to execute some tasks with different priorities.

Before starting the description a flowchart (see Fig. 3.1) it is very important to reader clearly understand the blocks which are used in the flowchart. Oval or rounded rectangle is signaling the start of a process. Set of operations that change value, form, or location of data represented as a rectangle [32]. Conditional operation determining which of two paths the program will take represented as a rhombus [32]. Finally, there is an operation in orange color, which means, that a process runs in parallel with running main algorithm.

So let us imagine, in some area an own vehicle starts to generate tasks. It can be any task of firmware updating or may be alerting, that something happened on a road, online surfing or even online gaming and so on. Main thing for us, that each task x_i has different priority and dynamically adding to our algorithm (see Fig. 3.1).

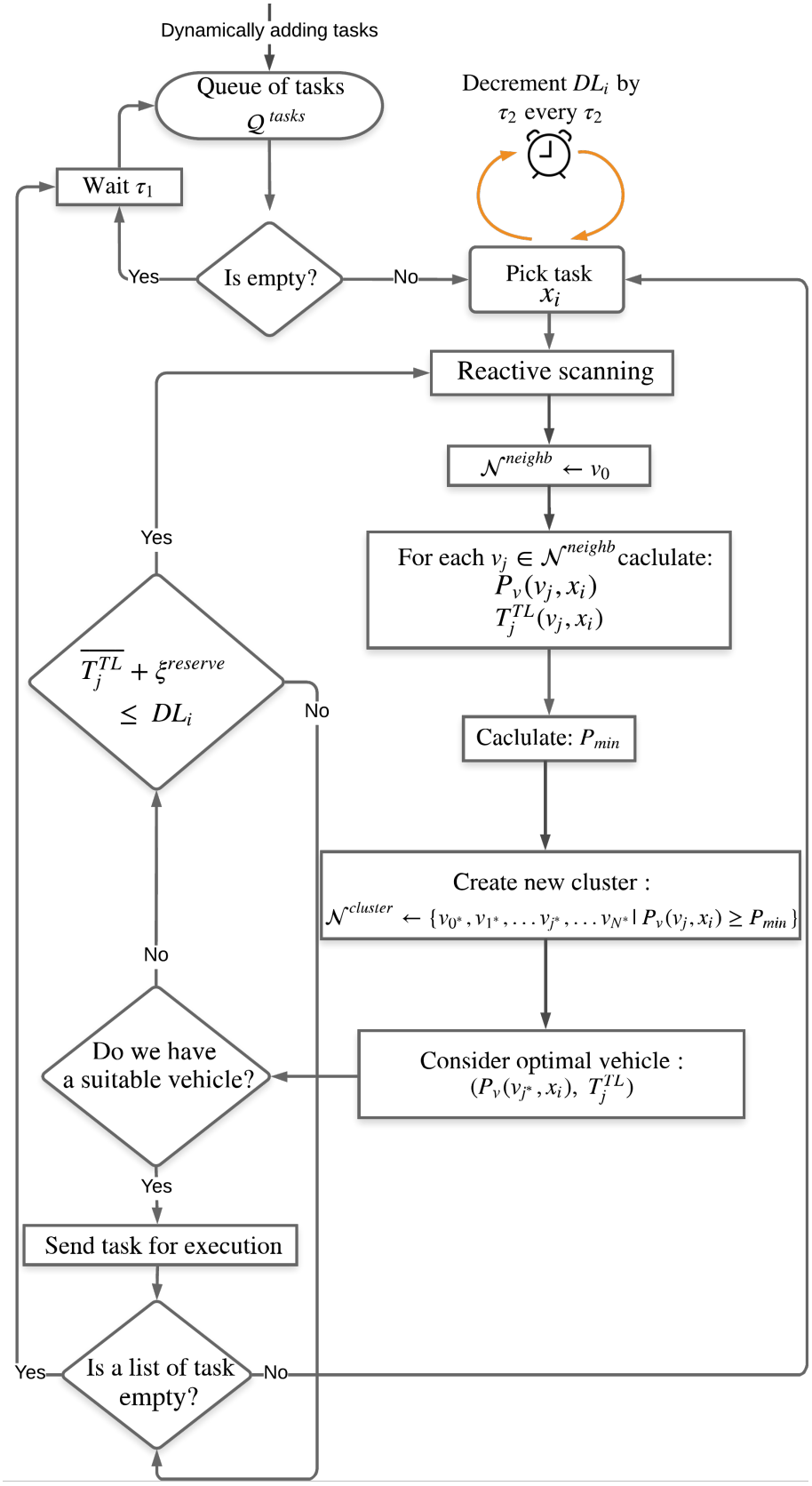


Figure 4.1: Flowchart of the proposed algorithm

If we have no task in queue the algorithm will check every τ_1 milliseconds until some will come. If tasks will arrive faster than a vehicle can process them, the own vehicle puts them into the queue (also called the buffer) until it can get around to transmitting them.

Important to note that each task has DL_i and it decrements by τ_2 milliseconds every τ_2 milliseconds. Using pick function $f^{pick}(x_i)$, the own vehicle picks a task with the lowest DL_i, if there are more than one task has the same DL_i, function also compare it by priority α_i, then task with the highest priority will be chosen. In case if DL_i and α_i will be the same for two or more tasks, those tasks will be considered consistently.

Depending on a type of a priority task, we provide reactive or proactive scanning. Reactive scanning approach provides scanning after getting a task, which leads to some delay, but at the same time it is more accurate, that is essential for high priority task. Hence for a low priority task nothing is needed, since reactive scanning was already provided.

After scanning we need to add the own vehicle v_0 to a set of neighbour vehicles $\mathcal{N}^{neighb}$ for further calculations caused the task can also be executed by own vehicle it self.

$$\mathcal{N}^{neighb} = \{v_0, v_1, v_2, ...v_j, ...v_N\}, where\ j \in \langle 0, N \rangle$$

Subsequently for each v_j determine Success probability $P_s(v_j, x_i)$ for chosen task x_i with the highest DL_i and relative priority α_i. Also calculate the Total time for execution x_i task. Total time consist of three times: transmission time to neighbour, task execution time and reception time.

$$T_j^{TL}(v_j, x_i) = t_j^{tx}(v_j, x_i) + t_j^{exe}(v_j, x_i) + t_j^{rec}(v_j, x_i)$$

Note that for v_0 vehicle success probability P_s is one, or 100%, and it is logical, because we do not need to send and get back the task, so probability, that we lose a task tends to zero. Certainly there is also exists probability, that own vehicle can fail the task, because of full buffer or other technical problem, but we can neglect it. Consequently t_0^{tx} and t_0^{rec} are also zero, because we calculate task by ourselves.

$$P_s(v_0, x_i) = 0 \tag{4.1}$$

$$T_0^{TL}(v_0, x_i) = t_0^{exec}(v_0, x_i) \tag{4.2}$$

Knowing all success probabilities for $J + 1$ vehicles (remember all neighbour vehicles $\mathcal{N}^{neighb}$ + own vehicle v_0), we define the minimal probability. That value will be constant for all vehicles, and updates each new scanning. After compare, we will get a new cluster with vehicles, which values more or equal to minimal probability. After that operation at the best case scenario we all neighbour vehicles will stay with us, but usually somebody has a lower probability, so in normal cases that comparison like a filter, that does not let pass vehicle with a lower value than minimal value.

$$\mathcal{N}^{cluster} \leftarrow (P_s(v_j, x_i) \geq P_{min})$$

$$\mathcal{N}^{cluster} = \{v_0, v_1, ...v_{j^*}, ...v_{N^*}\}, where\ j^* \subseteq j,\ j^* \in \langle 0, N^* \rangle, j^* \in \mathbb{N}$$

We almost have got all information needed to initialize process to execute task x_i. The last thing is to decide which one vehicle is an optimal for specific task with different success probabilities of neighbour vehicles and some priority. Because one vehicle has better

probability to execute, but lower performance resources. In that case considering function $f_j^{cons}(v_{j^*}, x_i)$ helps us, in math it is some kind of weight function.

It might happened that considering function will not have any optimal vehicle, it may be due to the all vehicles are busy, or probabilities so low or our own vehicle rides alone on a road and already execute another task. For that case next block checks this state.

• if there is no suitable vehicle, we need to check if we still have a time to execute this task providing scanning again and applying algorithm once more. For that case, Arithmetic mean value of total time $\overline{T_j^{TL}}$ with some reserve constant $\xi^{reserve}$ in seconds must be lower or equal to DL_i. Otherwise we will not execute task on time and DL_i will be expired. Then we check queue of tasks, if it is still have some tasks, we provide pick function and repeat all steps above, if it is empty we need to wait τ_1 milliseconds and check the queue again.

• if there is a suitable vehicle immediately task will be sent. Then as we mentioned above we check queue of tasks, if it is still have some tasks, we provide pick function and repeat all steps above, for empty queue we need to wait τ_1 milliseconds and check the queue again.

4.2 Algorithm's functions

In this section we describe how each function works and discuss in more detail. There are pick function, success probability of executing task, minimal probability, considering function and other ones, which make up the whole algorithm. At the end of this chapter we finally define algorithm with all functions.

4.2.1 Pick function

This part presents algorithms that solve the following sorting problem:

Input: A sequence of I deadlines $\{DL_1, DL_2, ..., DL_i, ..., DL_I\}$

Output: A reordering $\{DL'_1, DL'_2, ..., DL'_i, ..., DL'_I\}$ of the input sequence such that $DL'_1 \leqslant DL'_2 \leqslant ... \leqslant DL'_i \leqslant ... \leqslant DL'_I$

The input sequence is an I-element queue of tasks x_I, that must be sorted from the lowest value of DL_i. The pick function f^{pick} is based on minimal binary heap algorithm [33].

A binary heap is a heap data structure created by using a binary tree (see Fig. 4.6). The main advantage of that algorithm is low-latency low-complexity [34]. Binary tree has two properties:

- Shape property: all levels of the heap must be fully filled, except the bottom level, which may be partially filled from left to right
- Order property: due to highest priority is the minimum parents are less or equal than children

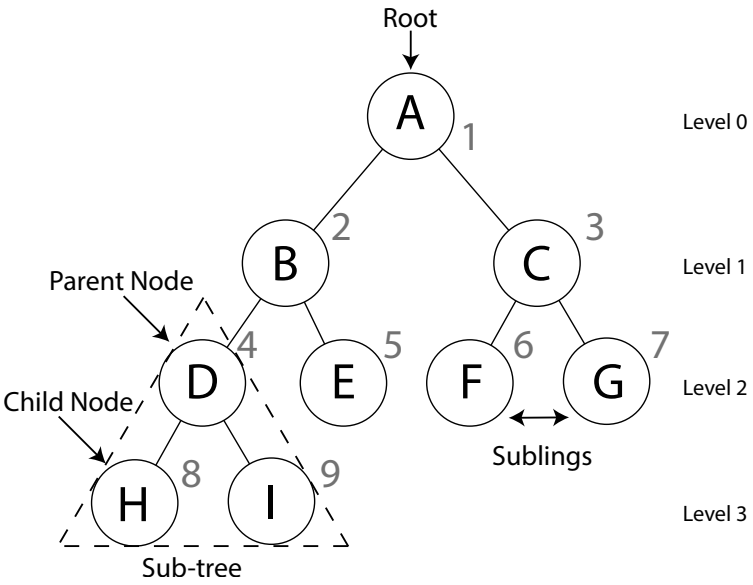


Figure 4.2: Structure of binary tree

The node at the top of the tree is called root. There is only one root per tree. Any node except the root node has one edge upward to a node called parent. The node below connected by its edge downward is called its child node.

4.2.1.1 Inserting new task

All new tasks put at the bottom of the heap. If an inserted task's DL is smaller than its parent node swap the element with its parent. Keep repeating the above operation and if task reaches its correct position algorithm stops.

4.2.1.2 Taking a task from the root

Taking out the task from the root. It is the task with the minimum value of DL. Then taking out the last task from the bottom level and putting to the empty root. If replaced task's DL is greater than any of its child task's DL swap the element with its smallest DL. Keep repeating the above step, if node reaches its correct position algorithm stops.

Example 4.2.1. *The own vehicle in a very short time generates queue of tasks* $\mathcal{Q}^{tasks} = \{x_1, x_2, x_3, x_4, x_5\}$*, which have* $DL_1 = 27\,[\text{s}],\ DL_1 = 25\,[\text{s}],\ DL_3 = 10\ [\text{s}],\ DL_4 = 13\,[\text{s}],\ DL_5 = 17\,[\text{s}]$*. Applying shape property all tasks will be insert consistently (see Fig. 4.6).* x_1 *inserted to the root,* x_2 *to the second place,* x_3 *to third and so on.*

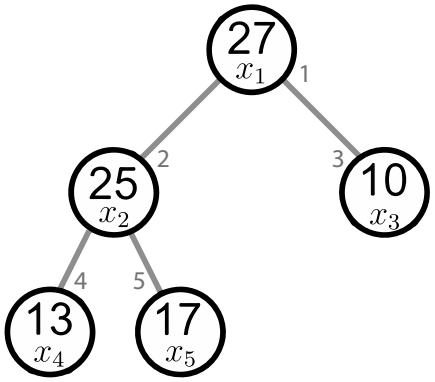
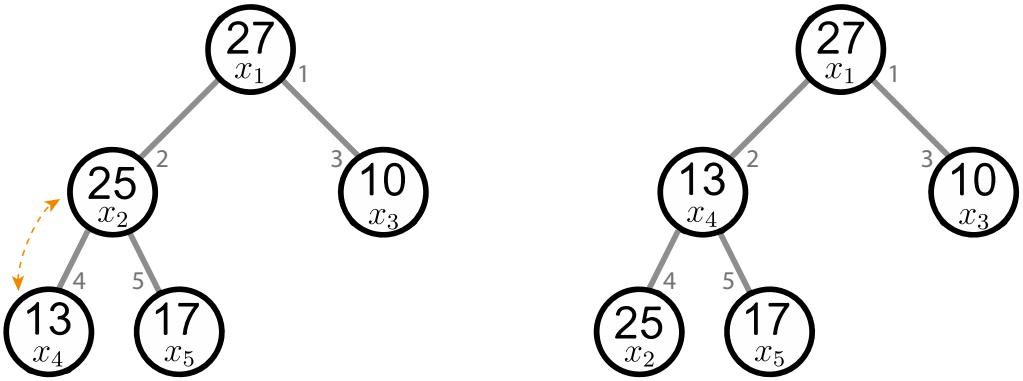


Figure 4.3: Structure of binary tree

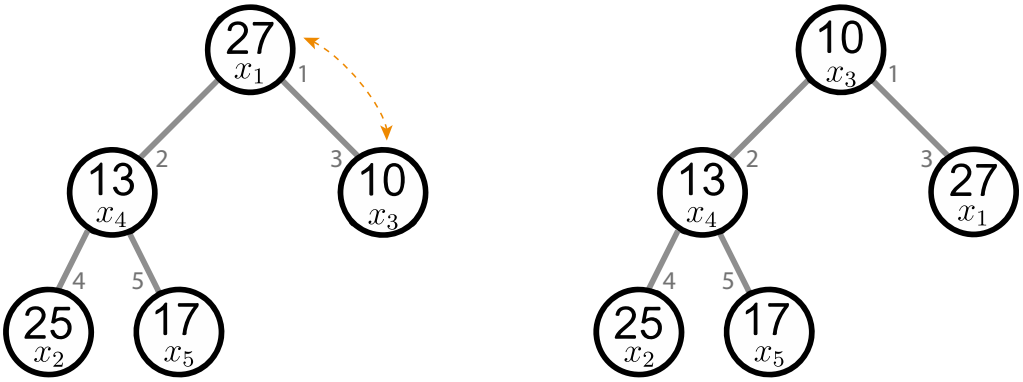
Then start from the bottom of tree to compare all tasks. Take a look at sub-tree that contains nodes #2, #4, #5. Node #2 is a parent of nodes #4 and #5. We see that DL of forth node is smaller than DL of second node, so provide swap (see Fig. 4.4 a). After that operation, we finish with that sub-tree (see Fig. 4.4 b)



a) swap nodes #2 and #4 b) after swap

Figure 4.4: Swap in the sub-tree, first step

Then go for a one level up, take a look at the root node #1 and its children are node #2 and #3. Notice that node #3 has the smallest DL, so swap with the root with node #3 (see Fig. 4.5 a).



a) swap nodes #3 and #1 b) after swap

Figure 4.5: Swap in the sub-tree, second step

Finally example's heap are sorted (see Fig. 4.5 b). According to that figure queue of tasks looks like $\mathcal{Q}^{tasks} = \{x_3, x_4, x_1, x_2, x_5\}$

Example 4.2.2. *Let us extend our example, so right after sorting came the task* x_6 *with the* $DL_6 = 20$ [s]. Again according to shape property, task is inserted consistently and will be a child of node #3 in the node #6. Note the attentive reader will notice, that when another task will come all DL will be lower, but algorithm will be the same and all DL change at the same time, so topology does not change as well. That is why for simplicity we use above example data.

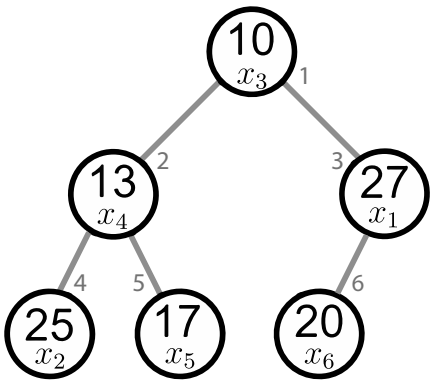
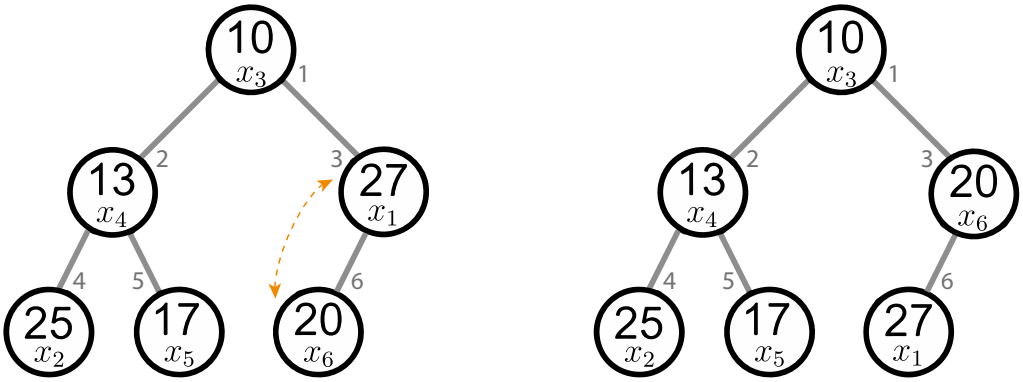


Figure 4.6: Structure of binary tree

Take a look at a new sub-tree that contains nodes #3 and #6. DL of sixth node is smaller than DL of the third node, so provide swap (see Fig. 4.7 a). After that operation, heap is sorted (see Fig. 4.7 b). Queue is $\mathcal{Q}^{tasks} = \{x_3, x_4, x_6, x_2, x_5, x_1\}$



a) swap nodes #3 and #1 b) after swap

Figure 4.7: Swap in the sub-tree, second step

The last and the important feature of pick function f^{pick}, taking the task with the lowest DL from the heap and rebuilding the heap properly. Next and last example will show how effectively it provides it.

Example 4.2.3. *Again, we are using data from the last example for simplicity. According to (see Fig. 4.7b) in the queue* $\mathcal{Q}^{tasks} = \{x_3, x_4, x_6, x_2, x_5, x_1\}$ x_3 *has the lowest* $DL_3 = 10\,[\text{s}]$, *so function picks it for further calculation (see Fig. 4.8) and the root place is empty.*

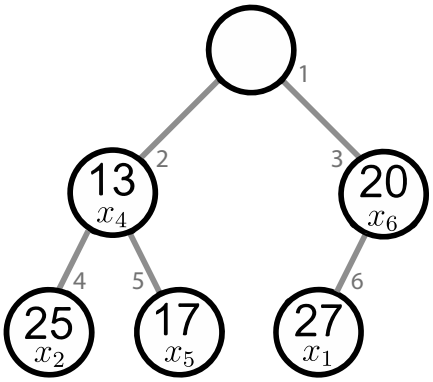
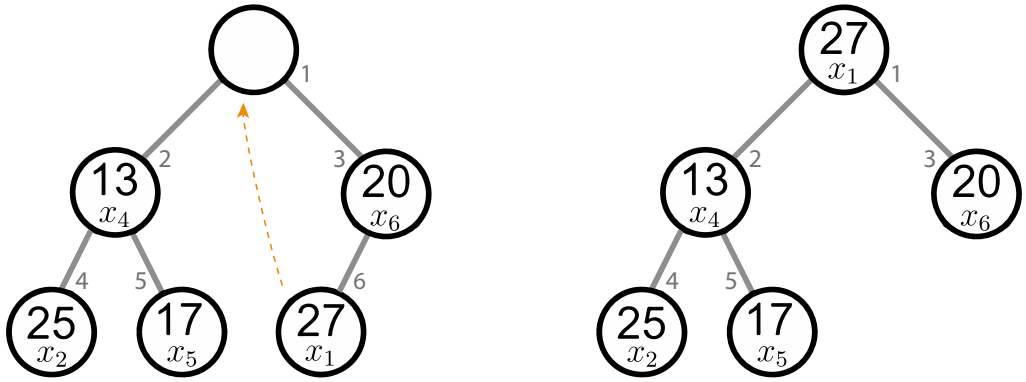


Figure 4.8: Pick the root value

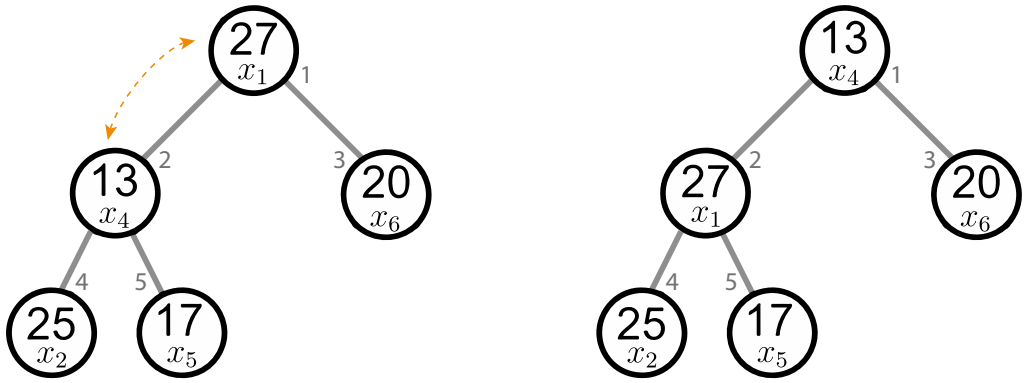
After find the last node #6 and insert it to the root (see Fig. 4.9).



a) swap nodes #3 and #1 b) after swap

Figure 4.9: Replacing the empty space of root by node #6

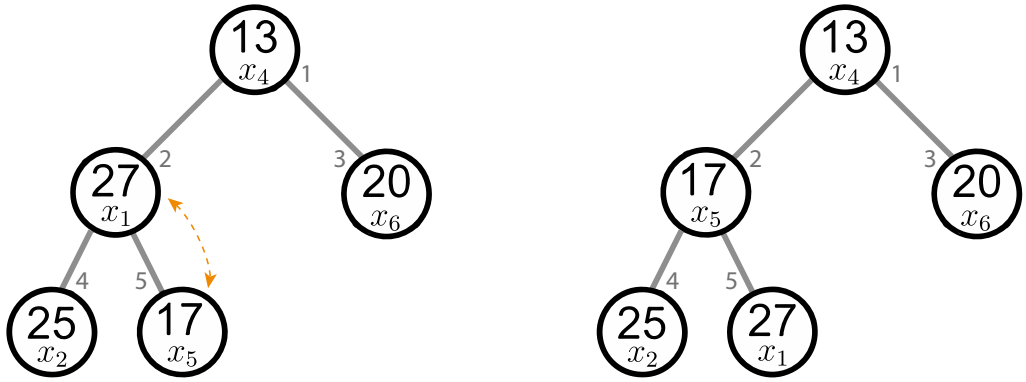
Again we need to provide comparison root node #1 with children #2 #3. The second node has lower value of DL, swap (see Fig. 4.10 a) and structure after swap (see Fig. 4.10 b).



a) swap nodes #3 and #1 b) after swap

Figure 4.10: Swap in the sub-tree, second step

Sub-tree with parent node #2, children #4, #5. Node #5 has the lowest DL (see Fig. 4.10 a) - swap.



a) swap nodes #3 and #1 b) after swap

Figure 4.11: Swap in the sub-tree, third step

Finally after removing root from the heap and rebuilding we get another queue $\mathcal{Q}^{tasks} = \{x_4, x_5, x_6, x_2, x_5\}$

4.2.2 Success probability function

In this subsection we talk about success probability $P_s(v_j, x_i)$ function, that vehicle v_j will execute i^{th} task. It consists of three significant parameters, which mutually complement each other and they are like a three level filter, that helps to choose the optimal vehicle. There are: deadline parameter ϕ_j^{DL}, Driver's behaviour parameter β_j^{beh} and Prediction parameter ψ_j^{con}. Each parameter will be considered in separate paragraphs.

$$P_s(v_j, x_i) = \phi_j^{DL}(v_j, x_i) \ \cdot \ \beta_j^{beh}((x_0, y_0), \mathbf{A}_j^{1\times K}) \ \cdot \ \psi_j^{con}(v_j, x_i) \tag{4.3}$$

Note that according to flowchart in the set of neighbours was already added an own vehicle. $P_s(v_0, x_i)$ *will be always 1, with the exception when the own vehicle is busy.*

4.2.2.1 Deadline filter parameter

According to the title of this paragraph ϕ_j^{DL} is working with a DL of tasks. It shows if some vehicle can execute a task within its DL, otherwise task will be expired.

Initially let us define $T_j^{TL}(v_j, x_i)$, with which ϕ_j^{DL} works very tightly.

Total time of task execution

$T_j^{TL}(v_j, x_i)$ is a total time that j^{th} vehicle needs to execute i^{th} task. It is a sum of three time periods: transmission time to neighbour vehicle $t_{0j}^{tx}(v_j, x_i)$, execution time of task $t_j^{exe}(v_j, x_i)$ and reception time $t_{j0}^{rec}(v_j, x_i)$, in other words time needed to send back executed task.

Transmission time

Let DS_i is a size of i^{th} task and $C_{0j}(d_{0j})$ by Shannon–Hartley theorem is the maximum rate at which information can be transmitted over a communications channel of a specified bandwidth in the presence of noise [35] that depends on a distance between own vehicle v_0 and neighbour vehicle v_j, so transmission time can be expressed as following

$$t_{0j}^{tx}(v_j, x_i) = \frac{DS_i}{C(d_{0j})} \tag{4.4}$$

where $C(d)$ is a channel capacity and according to 3GPP TR 36.785 (Release 14) [15] we got parameters of channel bandwidth $B = 10\,\text{MHz}$ and received power $P_{rx} =, -22\,\text{dBm}$

$$C(d) = B \cdot \log_2\left(1 + \frac{P_{rx}}{PL_d}\right)\Big|_{P_{rx}=-22\ \text{dBm}}^{B=10\,\text{MHz}} \tag{4.5}$$

corresponding to 3GPP TR 36.885 (Release 14) [36] pathloss model for V2V connection is LOS (Line of sight) in WINNER+ B1 [37] (note that the antenna height should be set to 1.5 m.) d is a distance between own vehicle and neighbour vehicle must be less that r, which is a maximal signal range of the own vehicle, other words is breakpoint distance. Note that for distance up to 10 m, the value of PL will be the same as value for 10m

$$PL = 22.7 \cdot \log_{10}(d) + 27 + 20 \cdot log_{10}(f_c)\Big|_{f_c=2\,\text{GHz}}^{10\,\text{m} < d < (r=50\,m)} \tag{4.6}$$

Distance d between two points (between the own vehicle and a neighbour vehicle) in $\mathbb{R}^2$ is denoted $d(v_0, v_j)$ and is defined by

$$d_{0j} = \sqrt{(x_0 - x_j)^2 + (y_0 - y_j)^2} \tag{4.7}$$

Execution time

Let IS_i is a number of instruction needed to execute i^{th} task and $MIPS_j$ is a performance of j^{th} vehicle, the execution time of i^{th} task for j^{th} vehicle can be calculated following equation

$$t_j^{exe}(v_j, x_i) = \frac{IS_i}{MIPS_j} \tag{4.8}$$

Reception time

Reception time is a time needed to send back the executed i^{th} task from j^{th} vehicle to own vehicle v_0. We have already known how to calculate data rate between two vehicles (4.5), but we do not know what the distance will be after transmitting time to neighbour and execution task time between two vehicles. In that case we can just predict it. For prediction the distance between these vehicles we must consider the parameters not yet used. There are actual instantaneous speed and acceleration, knowing that we can calculate the distance how far a vehicle can travel

$$d = speed \cdot t + \frac{accel \cdot t^2}{2} \tag{4.9}$$

Unfortunately this approach (4.9) consider that we have constant values of speed and accel, of course that can leads for not accuracy. It is logical that the greater the time value, the lower the accuracy of the calculations. But for relative short time intervals, which exactly VANET approach, it gives good results.

As we said before for time value we need to consider transmission time and execution time

$$t = t_{0j}^{tx}(v_j, x_i) + t_j^{exe}(v_j, x_i) \tag{4.10}$$

Hence, distances how far each vehicle can travelled are

$$d_0 = speed_0 \cdot (t_{0j}^{tx}(v_j, x_i) + t_j^{exe}(v_j, x_i)) + \frac{accel_0 \ \cdot \ (t_{0j}^{tx}(v_j, x_i) \ + \ t_j^{exe}(v_j, x_i))^2}{2} \tag{4.11}$$

$$d_j = speed_j \cdot (t_{0j}^{tx}(v_j, x_i) + t_j^{exe}(v_j, x_i)) + \frac{accel_j \ \cdot \ (t_{0j}^{tx}(v_j, x_i) \ + \ t_j^{exe}(v_j, x_i))^2}{2} \tag{4.12}$$

Then if we take away these values from each other, we get the distance between two vehicles.

$$d_0 - d_j = d_{0j}$$
$$d_j - d_0 = d_{j0}$$

Let us generalize it, so it does not matter, which vehicle travelled more distance, necessary to know a value

$$\sqrt{(d_0 - d_j)^2} = |d_0 - d_j| = d_{0j} = d_{j0} \tag{4.13}$$

Note that very important to remember, that vehicles were not on the same place, it was some distance when communication initialized. So complement (4.13) adding (4.7) and we get

$$d_{j0}^{rec} = |d_0 - d_j| + d_{j0} \tag{4.14}$$

Finally substitute the values of (4.14) using (4.11), (4.12) and (4.7) we get predicted value of reception distance d^{rec}. In such a way we know all values that need to calculate reception time $t_{j0}^{rec}(v_j, x_i)$:

$$t_{j0}^{rec}(v_j, x_i) = \frac{DS_i^{result}}{C(d^{rec})} \tag{4.15}$$

And knowing transmission time (4.4), execution time (4.8) and reception time (4.15), we get the total time

$$T_j^{TL}(v_j, x_i) = t_j^{tx}(v_j, x_i) + t_j^{exe}(v_j, x_i) + t_j^{rec}(v_j, x_i) \tag{4.16}$$

Definition 4.2.1. *Let $\phi_j^{DL}(v_j, x_i)$ is a deadline filter parameter of j^{th} vehicle for i^{th} task, if the total time T_j^{TL} will be less than the deadline of i^{th} task DL_i, the parameter return 1, otherwise 0. That method helps us to not consider vehicles, that needed more time that i^{th} task has before expired.*

$$\bullet\ \phi_j^{DL}\ (v_j, x_i) = \begin{cases} 1, & if\ \ T_j^{TL}(v_j, x_i) < DL_i \\ 0, & otherwise. \end{cases} \tag{4.17}$$

4.2.2.2 Driver's behaviour parameter

We have to familiarize with the components of driver‘s behaviour parameter before defining it.

Definition 4.2.2. *Let l is a number of roads at an intersection $\mathbf{B}_k^{1\times l}$ and $\mathbf{B}_k^{1\times l}$ is a k^{th} matrix of all intersections K, so matrix of all roads and intersections $\mathbf{S}_a^{1\times K}$ for a^{th} vehicle can be defined by*

$$\mathbf{S}_a^{1\times K} = \begin{pmatrix} \mathbf{B}_1^{1\times l} & \mathbf{B}_2^{1\times l} & \cdots & \mathbf{B}_k^{1\times l} & \cdots & \mathbf{B}_K^{1\times l} \end{pmatrix}, \tag{4.18}$$

where $l \in \langle 1, L\rangle, l, \in \mathbb{N}, L \in \mathbb{N}$ and $k \in \langle 1, K\rangle, k, \in \mathbb{N}, K \in \mathbb{N}$

$$\mathbf{B}_k^{1\times l} = \begin{pmatrix} road_1 & road_2 & \cdots & road_l & \cdots road_L \end{pmatrix}, \tag{4.19}$$

Note, has been considered, that each intersection includes up to 4 roads ($max\{l\} = 4$).

According to the text above number l shows the number of roads at matrix $\mathbf{B}_k^{1\times l}$, but we still do not know how many roads considering area has. So we need to define the new parameter $road_h$.

Definition 4.2.3. *Let $road_h$ is a h^{th} road of all existing roads H, and H is a proper superset of all values of l, then $road_h$ include two parameters. There are ID of a road - h*

and $counter_h$*, that shows how many times some vehicle rode through that road. By default it is 0.*

$$road_h = \{h, counter_h\}. \tag{4.20}$$
$$\textit{units of } h \text{ [-] and of a } \textit{counter} \text{ [-]}$$

where $h \in H$; $H \supseteq l$ and $H \supseteq L$

Knowing intersection matrix (4.19) and definition 4.2.3 we can get another interpretation of (4.18) as

$$\mathbf{S}_a^{1\times K} = \begin{pmatrix} road_1 & road_2 & \cdots & road_h & \cdots & road_H \end{pmatrix} \tag{4.21}$$

Thus, we defined $\mathbf{S}_a^{1\times K}$ as list of all roads on a certain area.

Every driver has personal preferences how to drive from one place to another, sometimes we choose the shortest route, sometimes we want to avoid heavy traffic or bring somebody home and so on. It is a stochastic process, but in general if we compare it with daily routine things it can be neglected, because most of our time we ride the same ways even noticed that. Let us define next component of drivers' behaviour matrix $\mathbf{A}_a^{1\times K}$.

Definition 4.2.4. *Let* $road_f$ *is* f^{th} *road from all roads F, and* $\mathcal{F}_a^{route}$ *be a list of all roads, that vehicles has traveled for a some time, therefore we can explain it as following*

$$\mathcal{F}_a^{route} = \begin{pmatrix} road_1 & road_2 & \cdots & road_f & \cdots & road_F \end{pmatrix}, \tag{4.22}$$

where $f \in F; (H \supseteq F) \vee (F \supseteq H) \vee (H \cap F); f \in \mathbb{N}$, $F \in \mathbb{N}$.

Important to say, that why we define $\mathcal{F}_a^{route}$ as another set, different from $\mathbf{S}_a^{1\times K}$. First for the reason that a^{th} vehicle could travelled outside of considering area, it can be different village, city or even country. Another case, that an a^{th} vehicle can ride just on the part of a certain area, definitely, it depends just on individual vehicle. That is why $\mathcal{F}_a^{route}$ can be subset or proper superset of $\mathbf{S}_a^{1\times K}$, or they can have just a common intersection.

Definition 4.2.5. $road_f$ *as* $road_h$ *also consists of two parameters (see 4.2.3): ID - f and* $counter_f$*. But unlike of* $road_h$*,* $counter_f$ *has a fix value equal 1. It means that* a^{th} *vehicle rode through* f^{th} *road and counter put number 1. Sequence of roads* $\mathcal{F}_a^{route}$ *also shows us from which road we got to actual road. For example sequence* ($road_{f-1}$, $road_f$, $road_{f+1}$) *said that we turn from a road* $(f-1)^{th}$ *to the* $(f)^{th}$ *and then we turn to a* $(f+1)^{th}$*. So* $road_f$ *we can defined as*

$$road_f = \{f, counter_f\}, \tag{4.23}$$
$$\textit{units of } f \text{ [-] and the } counter_f = 1$$

Note, in that sequence same roads can appear multiple times, because $\mathcal{F}_a^{route}$ is a route of a^{th} vehicle and the vehicle can travelled through some roads several times for some time period.

Definition 4.2.6. *Let* $\mathbf{A}_a^{1\times K}$ *be a matrix after adding the route* $\mathcal{F}_a^{route}$ *(4.22) of* a^{th} *vehicle to a matrix of all roads and intersections* $\mathbf{S}_a^{1\times K}$ *(4.21), so finally we can defined drivers' behaviour matrix as*

$$\mathbf{A}_a^{1\times K} = (\mathbf{S}_a^{1\times K} \leftarrow \mathcal{F}_a^{route}) \tag{4.24}$$

After that operation, the $counter_h$ of h^{th} road in $\mathbf{S}_a^{1\times K}$ (4.21) will change on such a value according to a number of same roads in (4.22). After adding the route of a^{th} vehicle we know how many times that vehicle was on each road. The last step will be calculation of probability, that next time a^{th} vehicle will turn from some road to another road according to actual direction of an own vehicle.

Definition 4.2.7. *Let* $P_{a,l}^k$ *be a probability of* a^{th} *vehicle, that turns to* l^{th} *road at* k^{th} *intersection, then for calculation we need to divide counter of* l^{th} *road to sum of all counters of all roads of* k^{th} *intersection*

$$P_{a,l}^k = \frac{counter_l}{\sum\limits_{l \,\in\, \mathbf{B}_k^{1\times l}} counter_l} \cdot 100, \tag{4.25}$$

units are percents [%]

Let us consider simple example to make it less confused and more comprehensible (see Fig. 4.12)

Example 4.2.4. *Knowing route* $\mathcal{F}_a^{route}$ *and matrix of roads and intersection* $\mathbf{S}_a^{1\times l}$*, we need to calculate probability that* a^{th} *vehicle will turn to the* 5^{th} *road from* 2^{nd} *road.*

We have a small area with 5 roads and considered just intersection among roads #2, #4 and #5. The parameters of a^{th} vehicle are:

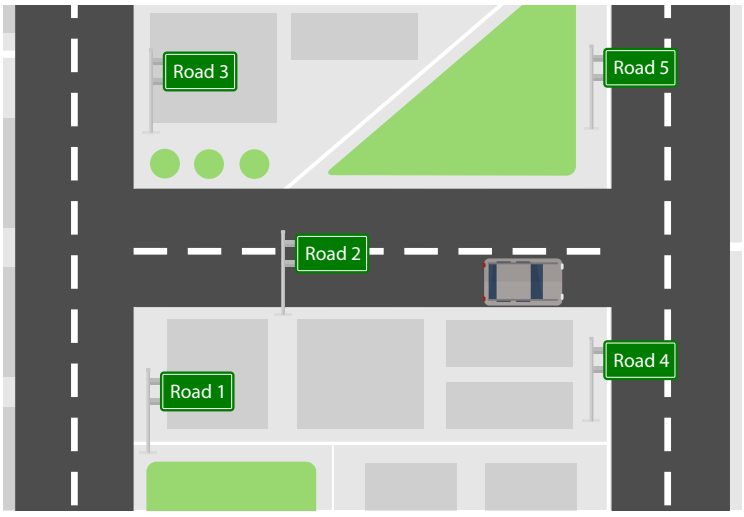


Figure 4.12: Example: An area with 5 roads and 2 intersections

$$\mathcal{F}_a^{route} = \Big(road_1 \quad road_2 \quad road_4 \quad \cdots \quad road_2 \quad road_5 \quad \cdots \quad road_2 \quad road_4 \quad \cdots\Big),$$

$$\mathbf{S}_a^{1\times 2} = \begin{pmatrix} \mathbf{B}_1^{1\times 3} & \mathbf{B}_2^{1\times 3} \end{pmatrix} = \begin{pmatrix} road_1 & road_2 & road_3 & road_4 & road_5 \end{pmatrix},$$

$$\mathbf{B}_2^{1\times 3} = \begin{pmatrix} road_2 & road_4 & road_5 \end{pmatrix},$$

Under the conditions of example it were three turns from the $road_2$:

$$\begin{aligned} \times 2: \ road_2 \rightarrow \ road_4 \\ \times 1: \ road_2 \rightarrow \ road_5 \end{aligned}$$

Then provide calculation (4.24), afterwards the counters of $road_4$ and $road_5$ changes from 0 to 1 and 2 accordingly. Then we can easily calculate (4.25) probability that a^{th} vehicle will turn to the 5^{th} road.

$$P_{a,\ l}^{k} = \frac{count_l}{\sum\limits_{l\ \in\ \mathbf{B}_k^{1\times l}} count_l} \cdot 100$$

$$P_{a,\ l=3}^{k=2} = \frac{1}{1+2} \cdot 100 = 33.33\ [\,\%\,]$$

Note that a probability will be more accurate, when it will be more counts on each intersection. But every vehicle can provide update (4.24) every certain time period and after some sufficient time it can be very useful information for further calculations.

If we come back to driver's behaviour parameter all necessary parameters was defined. For clarification, in the proposed algorithm position values of the own vehicle will help to identify where we are riding to get appropriate information from a driver's behaviour matrix $\mathbf{A}_a^{1\times K}$ about next intersection and roads, that it includes. Coordinates also can show us the direction of the vehicle (the difference of coordination values while the vehicle on a road $(x_{0\,0} - x_{0\,-1})$ and $(y_{0\,0} - y_{0\,-1})$.

Definition 4.2.8. *$\mathbf{A}_a^{1\times K}$ is a driver's behaviour matrix and (x_0, y_0) are coordinates of own vehicle (x_0, y_0), then driver's behaviour parameter will be*

$$\bullet\ \beta_j^{beh}((x_0, y_0), \mathbf{A}_j^{1\times K}) = \frac{count_h}{\sum\limits_{h\ \in\ \mathbf{B}_k} count_h}, \tag{4.26}$$

$$\text{unitless } [\,-\,].$$

As we can see $\beta_j^{beh}((x_0, y_0), \mathbf{A}_j^{1\times K})$ is also can removes is not suitable vehicles. The last parameter of vehicle probability function left, let consider it in the next paragraph.

4.2.2.3 Prediction parameter

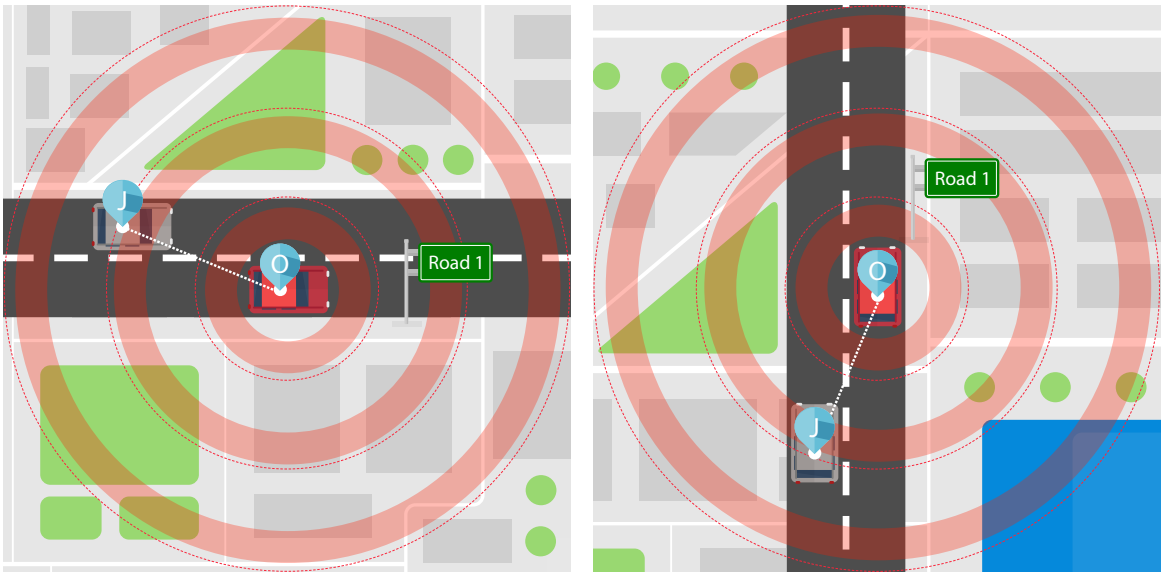
The last component of vehicle probability function $P_s(v_j, x_i)$ is prediction parameter ψ_j^{con}. That parameter checks how long j^{th} vehicle will travelled within our range of signal. If any vehicle needs more time to get task, execute it and send back than connection time, such the vehicle will be removed automatically.

$t_j^{con}(v_j, x_i)$ is a connection time, which means time that j^{th} vehicle will be in our range without losing signal. There are several options and we need consider all of them to be as accurate as possible.

Vertical and Horizontal roads

It is the most common case, when two vehicles v_0 and v_j riding on a same road. So the question is, can we use that neighbour vehicle, for example with very good parameters for task execution? What if this car leaves our signal range before execution task?

Let us consider that vehicles ride on a same road (see Fig. 4.13)



a) Horizontal road b) Vertical road

Figure 4.13: Common cases when vehicles are on a same road

We had a similar situation when we defined reception time, but in that case we knew the connection time, it was transmission time and execution time. But now we do not know even that.

First we need to derive the formula of connection time. We know that an own vehicle and a neighbour vehicle travelled some distance and after connection time j^{th} vehicle will be out of range.

$$|d_j - d_0| = r \tag{4.27}$$

Again we need to remember, that vehicles did not start from a same place, there was some distance (4.7) between them

$$|d_j - d_0| = r - d \tag{4.28}$$

Let substitute values of d_j and d_0 using (4.9), we get

$$\left| speed_j \cdot t_j^{con} + \frac{accel_j \cdot (t_j^{con})^2}{2} - speed_0 \cdot t_j^{con} - \frac{accel_0 \cdot (t_j^{con})^2}{2} \right| = r - d$$

Remove the absolute value from the equation

$$speed_j \cdot t_j^{con} + \frac{accel_j \cdot (t_j^{con})^2}{2} - speed_0 \cdot t_j^{con} - \frac{accel_0 \cdot (t_j^{con})^2}{2} = \pm\ (r - d)$$

Move to the left side from right side of the equation and divide out the greatest common factor from each term

$$(t_j^{con})^2 \cdot \frac{accel_j - accel_0}{2} + t_j^{con} \cdot (speed_j - speed_0) \mp\ (r - d) = 0$$

The discriminant of the quadratic equation is

$$D = (speed_j - speed_0)^2 - 4 \cdot \frac{accel_j - accel_0}{2} \cdot (\mp\ (r - d))$$

$$D = (speed_j - speed_0)^2 \pm 4 \cdot \frac{accel_j - accel_0}{2} \cdot (r - d)$$

The roots of the quadratic equation are

$$t_j^{con} = \frac{-(speed_j - speed_0) \pm \sqrt{(speed_j - speed_0)^2 \pm 2 \cdot (accel_j - accel_0) \cdot (r - d)}}{accel_j - accel_0} \quad (4.29)$$

We know, that a time must have only positive values. But in that case it is quite complicated to identify, cause j^{th} neighbour vehicle can have different values in comparison with an own vehicle. There are four variations for each sign of equation:

- $speed_j > speed_0$ and $accel_j < accel_0$
- $speed_j < speed_0$ and $accel_j > accel_0$
- $speed_j > speed_0$ and $accel_j > accel_0$
- $speed_j < speed_0$ and $accel_j < accel_0$

And we have 4 equations with 4 different variations each, so totally we have 16 equations with 32 roots. Certainly we have not to consider negative and complex roots, but in this case, the algebraic method of solving is more complicated, so we will use the analytical method of solving. We build movement models for each case, it will help to choose appropriate equation for each case.

These two conditions we can consider simultaneously, because it does not matter, which vehicle will overtake. We need to know, after what time will be distance equals r. Next table was get using (4.9) for each vehicle.

To know what kind of value to expect, reference movement model was used. For example consider following values:

Table 4.1: First and second conditions for connection time

a) $speed_j > speed_0$ and $accel_j < accel_0$

	speed	accel
v_j	10	2
v_0	12	1.8

b) $speed_j < speed_0$ and $accel_j > accel_0$

	speed	accel
v_j	12	1.8
v_0	10	2

For those parameters we got according table. First column is a distance between vehicles (4.27), again remember that it does matter for us which vehicle is onward or backward, that is why we can use two conditions together. d_0 is a distance that travelled own vehicle per time t, d_j is a distance that travelled neighbour vehicle per time t. For second condition, will be same values of distance between vehicles, only need to rechange second and third columns, which means values in second column will be for d_0 and in the third for d_j

Table 4.2: Data from the movement model showing distance between vehicles

a) Horizontal road

$\|d\|\ [m]$	$d_j\ [m]$	$d_0\ [m]$	$t\ [s]$
1.9	11	12.9	1
3.6	24	27.6	2
5.1	39	44.1	3
6.4	56	62.4	4
7.5	75	82.5	5
8.4	96	104.4	6
9.1	119	128.1	7
9.6	144	153.6	8
9.9	171	180.9	9
10	200	210	10
9.9	231	240.9	11
9.6	264	273.6	12
9.1	299	308.1	13
8.4	336	344.4	14
7.5	375	382.5	15
6.4	416	422.4	16
5.1	459	464.1	17
3.6	504	507.6	18

b) Vertical road

$\|d\|\ [m]$	$d_j\ [m]$	$d_0\ [m]$	$t\ [s]$
1.9	551	552.9	19
0	600	600	20
2.1	651	648.9	21
4.4	704	699.6	22
6.9	759	752.1	23
9.6	816	806.4	24
12.5	875	862.5	25
15.6	936	920.4	26
18.9	999	980.1	27
22.4	1064	1041.6	28
26.1	1131	1104.9	29
30	1200	1170	30
34.1	1271	1236.9	31
38.4	1344	1305.6	32
42.9	1419	1376.1	33
47.6	1496	1448.4	34
52.5	1575	1522.5	35
57.6	1656	1598.4	36

Then we can get a figure 4.8 showing distance between vehicles depending on time.

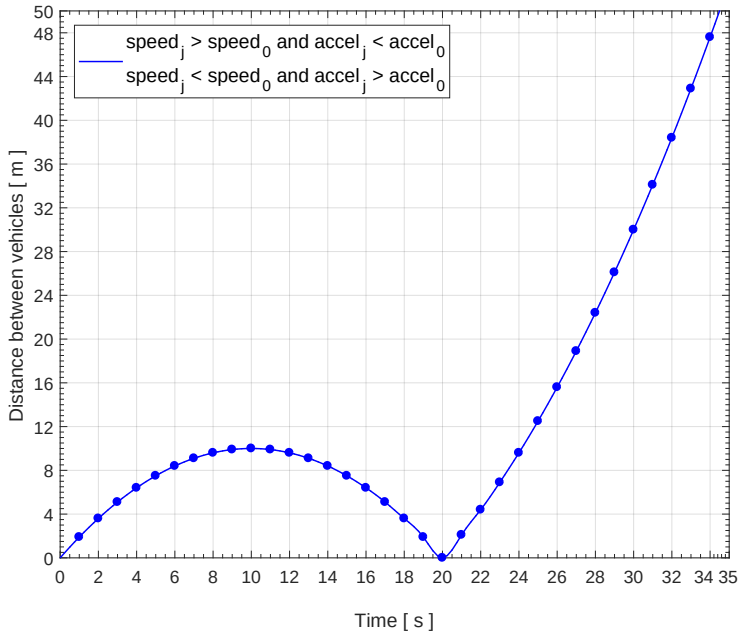


Figure 4.14: Distance between vehicles in time

According to the figure 4.14, first 10 seconds the distance between vehicles increases, because the own vehicle (for second condition a neighbour vehicle) has more speed than neighbour vehicle (for second condition the own vehicle), then in time interval between 10 seconds and 20 the distance decreases, because the own vehicle (neighbour vehicle) has more acceleration and is catching up the neighbour vehicle (the own vehicle). Most important part for us starts after 20 seconds (see Fig. 4.15) distance start increasing, because the own vehicle (neighbour vehicle) has more speed and acceleration, after 35 seconds distance is around 50 meters and we lose signal.

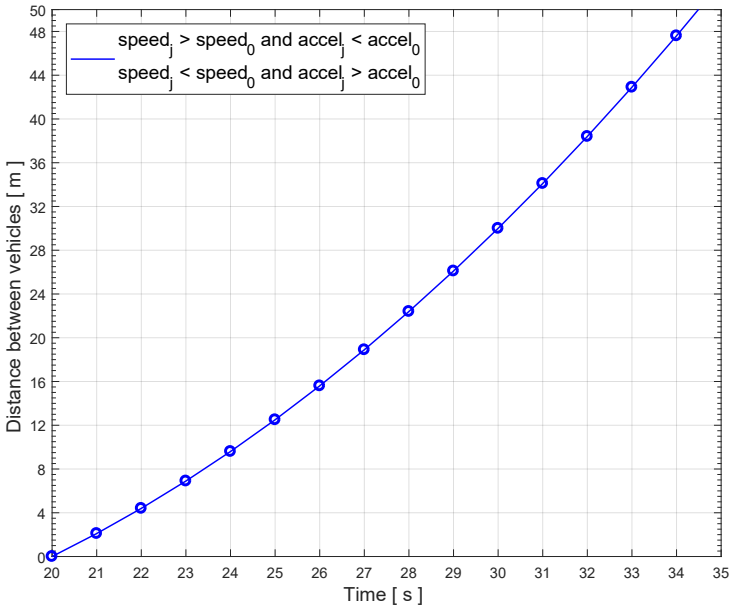


Figure 4.15: Distance between vehicles after 20 [s]

Applying (4.29) we need to find exactly equation, that gives same results as movement model. So in that way

1. For $speed_j < speed_0$ and $accel_j > accel_0$

$$t_j^{con} = \frac{-(speed_j - speed_0) + \sqrt{(speed_j - speed_0)^2 + 2 \cdot (accel_j - accel_0) \cdot (r - d)}}{accel_j - accel_0} \tag{4.30}$$

2. For $speed_j > speed_0$ and $accel_j < accel_0$

$$t_j^{con} = \frac{-(speed_j - speed_0) - \sqrt{(speed_j - speed_0)^2 - 2 \cdot (accel_j - accel_0) \cdot (r - d)}}{accel_j - accel_0} \tag{4.31}$$

Let us consider the third condition and forth conditions.

Table 4.3: Third and forth conditions for connection time

a) $speed_j > speed_0$ and $accel_j > accel_0$

	speed	accel
v_j	12	2
v_0	10	1.8

b) $speed_j < speed_0$ and $accel_j < accel_0$

	speed	accel	height	v_j
10	1.8			
v_0	12	2		

Table 4.4: Data from the movement model showing distance between vehicles

a) Third condition

$\|d\|\ [m]$	$d_j\ [m]$	$d_0\ [m]$	$t\ [s]$
2.1	13	10.9	1
4.4	28	23.6	2
6.9	45	38.1	3
9.6	64	54.4	4
12.5	85	72.5	5
15.6	108	92.4	6
18.9	133	114.1	7
22.4	160	137.6	8
26.1	189	162.9	9
30	220	190	10
34.1	253	218.9	11
38.4	288	249.6	12
42.9	325	282.1	13
47.6	364	316.4	14
52.5	405	352.5	15

b) Forth condition

$\|d\|\ [m]$	$d_j\ [m]$	$d_0\ [m]$	$t\ [s]$
2.1	10.9	13	1
4.4	23.6	28	2
6.9	38.1	45	3
9.6	54.4	64	4
12.5	72.5	85	5
15.6	92.4	108	6
18.9	114.1	133	7
22.4	137.6	160	8
26.1	162.9	189	9
30	190	220	10
34.1	218.9	253	11
38.4	249.6	288	12
42.9	282.1	325	13
47.6	316.4	364	14
52.5	352.5	405	15

Graph will be one for two cases same as in the previous case.

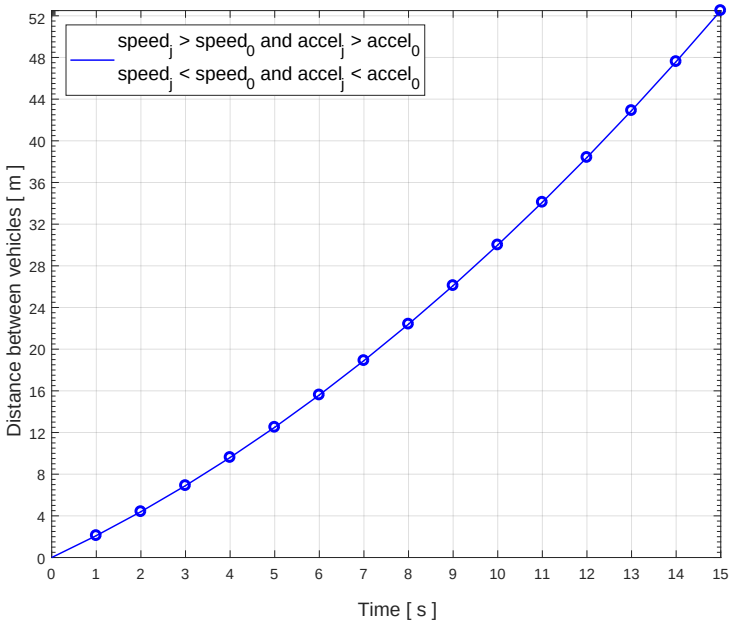


Figure 4.16: Distance between vehicles in time

According to the graph (see Fig. 4.17), the distance between vehicles in the whole domain increasing, because the neighbour vehicle (own vehicle for forth condition) has greater values

of speed and acceleration than the own vehicle (the neighbour vehicle). In the same way as before, we can identify appropriate equation for each condition.

3. For $speed_j > speed_0$ and $accel_j > accel_0$

$$t_j^{con} = \frac{-(speed_j - speed_0) + \sqrt{(speed_j - speed_0)^2 + 2 \cdot (accel_j - accel_0) \cdot (r - d)}}{accel_j - accel_0} \tag{4.32}$$

4. For $speed_j < speed_0$ and $accel_j < accel_0$

$$t_j^{con} = \frac{-(speed_j - speed_0) - \sqrt{(speed_j - speed_0)^2 - 2 \cdot (accel_j - accel_0) \cdot (r - d)}}{accel_j - accel_0} \tag{4.33}$$

Let us compare all four conditions. An attentive reader will notice that for the first condition (4.30) and third equations (4.32) are the same. For second (4.31) and forth condition (4.33) equation is also the same.

Take a look at first and third conditions:

$$speed_j < speed_0 \text{ and } accel_j > accel_0$$
$$speed_j > speed_0 \text{ and } accel_j > accel_0$$

We came to an important conclusion, that a value of *speed* do not impact on the equation. In depends only on acceleration values. To be sure, check the remaining third and forth conditions:

$$speed_j > speed_0 \text{ and } accel_j < accel_0$$
$$speed_j < speed_0 \text{ and } accel_j < accel_0$$

Again, all conditions depend only on acceleration values. The main aim of this paragraph was to get prediction time, when vehicles on vertical or horizontal roads. So finally, we got

1. If $accel_j > accel_0$

$$t_j^{con} = \frac{-(speed_j - speed_0) + \sqrt{(speed_j - speed_0)^2 + 2 \cdot (accel_j - accel_0) \cdot (r - d)}}{accel_j - accel_0} \tag{4.34}$$

2. If $accel_j < accel_0$

$$t_j^{con} = \frac{-(speed_j - speed_0) - \sqrt{(speed_j - speed_0)^2 - 2 \cdot (accel_j - accel_0) \cdot (r - d)}}{accel_j - accel_0} \tag{4.35}$$

We have already considered common cases when cars are on the same road. Now there will be more specific cases.

Vertical road v_j → horizontal road v_0

For better understanding, it is necessary to notify the reader how to understand notations correctly. Each point will be written capital letter and its coordinates are also capital letters. For example point **D** has coordinates (x_D, y_D), exception is for a neighbour vehicle, that always will start from point **J**, but has coordinates (x_j, v_j) and for an own vehicle (x_0, y_0), that always will start from point **O**.

On this situation the neighbour vehicle rides on vertical road and then will turn to the same road, where rides the own vehicle (see Fig. 4.13).

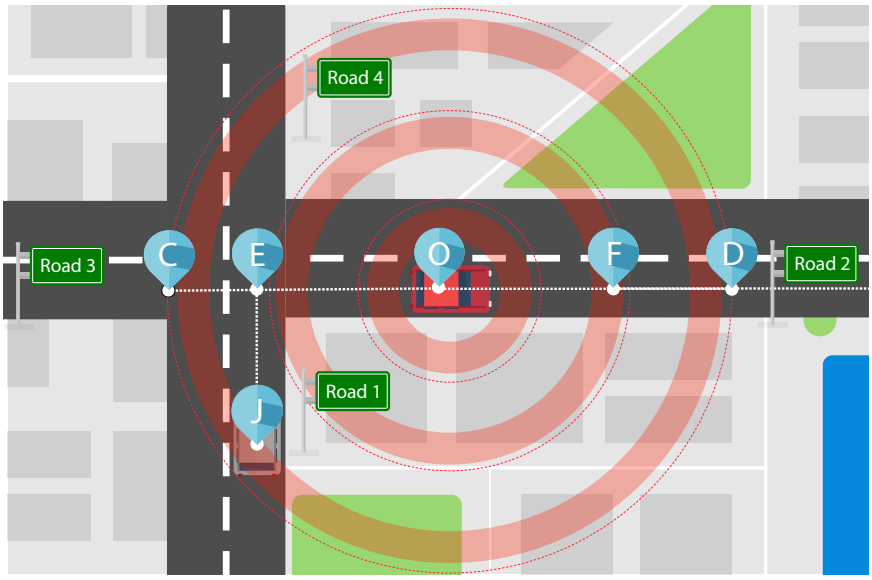


Figure 4.17: Vertical road v_j → horizontal road v_0

Note that we always consider, that a neighbour vehicle will turn to the same road, other case will be removed by our second filter - driver's behaviour parameter β_j^{beh} **??**.

Again we need to calculate connection time, how long the neighbour vehicle will be in our signal radius. Let us derive it, for the beginning calculate the distance between intersection and the neighbour vehicle.

$$\begin{aligned} d_{JE} &= \sqrt{(x_j - x_E)^2 + (y_j - y_E)^2} = \\ &= \sqrt{(y_j - y_E)^2} = \\ &= \sqrt{(y_j - y_0)^2} \end{aligned} \tag{4.36}$$

Note that $\sqrt{(...)^2}$ *notation also said as, that point* J *can be on the other side of road, it does not matter if neighbour vehicle rides from the south or north and turns to the own vehicle's road.*

According to that v_j rides on the vertical road x position will not change or the slightest change can be neglected. Y coordinate of own vehicle has the same value of intersection coordinate (see Fig. 4.13).

Calculate the time $t_{j,JE}$ when v_j will reach an intersection

$$d_{JE} = speed_j \cdot t_{j,JE} + \frac{accel_j \cdot t_{j,JE}^2}{2} \tag{4.37}$$

We know all these parameters (4.79), just need to find the roots of that equation.

$$\frac{accel_j \cdot t_{j,JE}^2}{2} + speed_j \cdot t_{j,JE} - d_{JE} = 0$$

We do not need to consider a negative root, so

$$\begin{aligned} t_{j,JE} = & \frac{\pm\sqrt{2 \cdot accel_j \cdot d_{JE} + speed_j^2} - speed_j}{accel_j} = \\ = & \frac{\sqrt{2 \cdot accel_j \cdot d_{JE} + speed_j^2} - speed_j}{accel_j} = \\ = & \frac{\sqrt{2 \cdot accel_j \cdot (\sqrt{(y_j - y_0)^2}) + speed_j^2} - speed_j}{accel_j} \end{aligned} \tag{4.38}$$

So, we have already know the time, that needed j^{th} neighbour to reach an intersection. Then calculate a distance d_{CE} and time $t_{0,CE}$ when the own vehicle signal will not reach the intersection. In that case the, values of y coordinate will be the same, changes only x coordinate

$$\begin{aligned} d_{CE} &= r - d_{EO} = \\ &= r - \sqrt{(x_E - x_0)^2 + (y_E - y_0)^2} = \\ &= r - \sqrt{(x_E - x_0)^2} \end{aligned} \tag{4.39}$$

$$d_{CE} = speed_0 \cdot t_{0,CE} + \frac{accel_0 \cdot t_{0,CE}^2}{2}$$

$$\frac{accel_0 \cdot t_{0,CE}^2}{2} + speed_0 \cdot t_{0,CE} - d_{CE} = 0$$

$$\begin{aligned} t_{0,CE} = & \frac{\pm\sqrt{2 \cdot accel_0 \cdot d_{CE} + speed_0^2} - speed_0}{accel_0} = \\ = & \frac{\sqrt{2 \cdot accel_0 \cdot d_{CE} + speed_0^2} - speed_0}{accel_0} = \\ = & \frac{\sqrt{2 \cdot accel_0 \cdot (r - \sqrt{(x_E - x_0)^2} + speed_0^2} - speed_0}{accel_0} \end{aligned} \tag{4.40}$$

So we know $t_{j,JE}$ and $t_{0,CE}$, it is logically if the neighbour vehicle needs more time to get to intersection, than the own vehicle to be out of reach, then we lose connection with v_j and our connection time with v_j will be $t_{j,JE}$.

$$\text{If } t_{j,JE} > t_{0,CE} \implies t_j^{con} = t_{0,CE} \tag{4.41}$$

If $t_{j,JE} \leq t_{0,CE}$, which means the neighbour vehicle will be faster than own vehicle. It requires further calculations to get connection time. In that case vehicles will be on the same road, but we do not know their exactly positions.

We can calculate the distance $d_0^{\text{while waiting j}}$, which the own vehicle rode, while waited for the neighbour vehicle for time reaching the intersection by neighbour vehicle $t_{j,JE}$

$$d_0^{\text{while waiting j}} = speed_0 \cdot t_{j,JE} + \frac{accel_0 \cdot t_{j,JE}^2}{2}$$

But we still does not know, the exactly coordinate where the own vehicle will be. Get it from the definition of distance

$$d_0^{\text{while waiting j}} = \sqrt{(x_0 - x_0^{\text{while waiting j}})^2}$$

$$x_0^{\text{while waiting j}} = x_0 \pm d_0^{\text{while waiting j}} \tag{4.42}$$

Note that $\pm$ notation said us, that own vehicle v_0 can have two directions on the horizontal road, if v_0 rides to the east we use "+" sign, for the west direction use "-" sign.

Let us consider, when the vehicles on the same road (see Fig. 4.18)

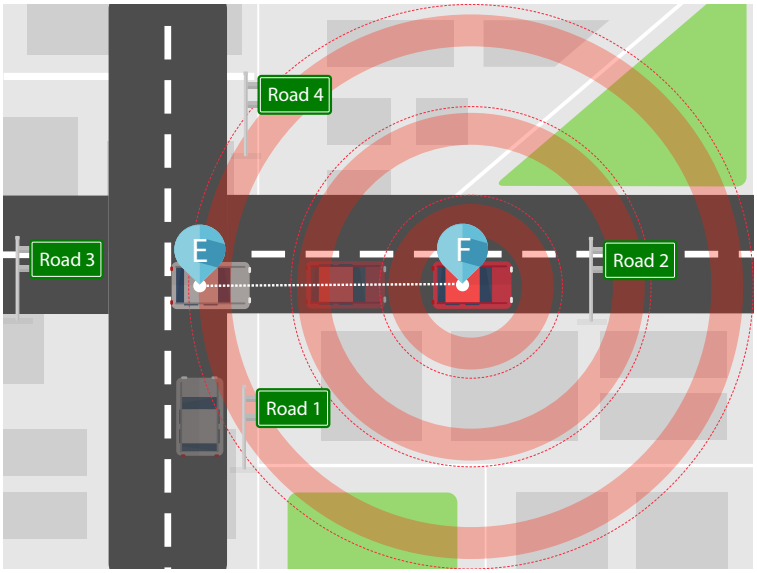


Figure 4.18: Vehicles are on the same road

Coordinates of v_0 after $t_{j,JE}$ are in the node **F**:

$$\mathbf{F} = (x_0 \pm d_0^{\text{while waiting j}}, y_0) \tag{4.43}$$

Coordinates of v_j after $t_{j,JE}$ are in the node **E**:

$$\mathbf{E} = (x_j, y_0) \tag{4.44}$$

The distance between two vehicles on the same road will be

$$\begin{aligned} d^{\text{between 0 and j}} &= \sqrt{\Big(\underbrace{x_0}_{x_0 \pm d_0^{\text{while waiting j}}} - x_j\Big)^2 + \Big(y_0 - \underbrace{y_j}_{y_0}\Big)^2} = \\ &= \sqrt{\left(x_0 \pm d_0^{\text{while waiting j}} - x_j\right)^2} \end{aligned} \tag{4.45}$$

Now according to the values of $accel_j$ and $accel_0$ choose an appropriate formula (4.34) or (4.35), then calculate connection time t_j^{con}. So the total connection time of the neighbour vehicle in case of vertical road and then turning to the same rode as own vehicle will be:

$$t_j^{TL,con} = t_{j,JE} + t_j^{\text{con}} \tag{4.46}$$

Vertical road v_0 → horizontal road v_j

According to this situation own vehicle rides on vertical road (both cases included from north and from south) and wants to turn to a horizontal road (see Fig. 4.19), where rides potential neighbour vehicle. We need to know its time connection.

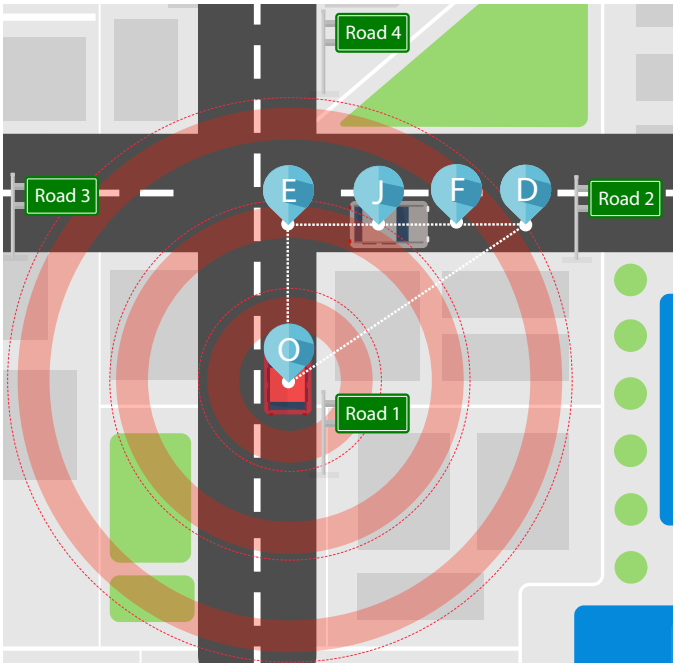


Figure 4.19: Vertical road v_0 → horizontal road v_j

Calculate distance d_{OE} between the own vehicle and intersection

$$\begin{aligned} d_{OE} &= \sqrt{(x_0 - x_E)^2 + (y_0 - y_E)^2)} = \\ &= \sqrt{(y_0 - y_E)^2} = \\ &= \sqrt{(y_0 - y_j)^2} \end{aligned} \tag{4.47}$$

Derive the time $t_{0,OE}$, that the own vehicle will spend to get to an intersection

$$\begin{aligned} d_{OE} &= speed_0 \cdot t_{0,OE} + \frac{accel_0 \cdot t_{0,OE}^2}{2} \\ \frac{accel_0 \cdot t_{0,OE}^2}{2} &+ speed_0 \cdot t_{0,OE} - d_{OE} = 0 \end{aligned} \tag{4.48}$$

We need a positive root of the quadratic equation

$$\begin{aligned} t_{0,OE} &= \frac{\pm\sqrt{2 \cdot accel_0 \cdot d_{OE} + speed_0^2} - speed_0}{accel_0} = \\ &= \frac{\sqrt{2 \cdot accel_0 \cdot d_{OE} + speed_0^2} - speed_0}{accel_0} = \\ &= \frac{\sqrt{2 \cdot accel_0 \cdot \sqrt{(y_0 - y_j)^2} + speed_0^2} - speed_0}{accel_0} \end{aligned} \tag{4.49}$$

Then, necessary to know the distance d_{JD} and time $t_{j,JD}$ when the neighbour vehicle lose a connection with the own vehicle. First of all, we need to find out coordination of point **D**.

$$\begin{aligned} d_{OD} = r &= \sqrt{(x_0 - x_D)^2 + (y_0 - y_D)^2} = \\ &= \sqrt{(x_0 - x_D)^2 + (y_0 - y_j)^2} \end{aligned} \tag{4.50}$$

Finally x_D is

$$x_D = x_0 \pm \sqrt{-y_j^2 + 2y_0y_j + r^2 - y_0^2} \tag{4.51}$$

Note, that sign "+" means, that the own vehicle turns to the east, "-" we use when the own vehicle rides to the west direction

Then we can get the distance d_{JD}

$$\begin{aligned} d_{JD} &= \sqrt{(x_j - x_D)^2 + (y_j - y_D)^2)} = \\ &= \sqrt{(x_j - x_D)^2} = \\ &= \sqrt{\left(x_j - (x_0 \pm \sqrt{-y_j^2 + 2y_0y_j + r^2 - y_0^2})\right)^2} \end{aligned} \tag{4.52}$$

Knowing the distance calculate time $t_{j,JD}$

$$d_{JD} = speed_j \cdot t_{j,JD} + \frac{accel_j \cdot t_{j,JD}^2}{2}$$

$$\frac{accel_j \cdot t_{j,JD}^2}{2} + speed_j \cdot t_{j,JD} - d_{JD} = 0$$

$$\begin{aligned} t_{j,\mathrm{JD}} &= \\ &= \frac{\pm\sqrt{2 \cdot \mathrm{accel}_j \cdot d_{j,JD} + \mathrm{speed}_j^2} - \mathrm{speed}_j}{\mathrm{accel}_j} = \\ &= \frac{\sqrt{2 \cdot \mathrm{accel}_j \cdot d_{j,JD} + \mathrm{speed}_j^2} - \mathrm{speed}_j}{\mathrm{accel}_j} = \\ &= \frac{\sqrt{2 \cdot accel_j \cdot \sqrt{\left(x_j - (x_0 \pm \sqrt{-y_j^2 + 2y_0y_j + r^2 - y_0^2})\right)^2} + speed_j^2} - speed_j}{accel_j} \end{aligned} \tag{4.53}$$

So we know $t_{0,OE}$ and $t_{j,JD}$, it is logically if the own vehicle needs more time to get to intersection, than the neighbour vehicle to be out of reach, then we lose connection with v_j and our connection time with v_j will be $t_{j,JD}$.

$$\text{If } t_{0,OE} > t_{j,JD} \implies t_j^{con} = t_{j,JD} \tag{4.54}$$

If $t_{0,OE} \leq t_{j,JD}$, which means the own vehicle will be faster than neighbour vehicle will lose its connection. It requires further calculations to get connection time t_j^{con}. In that case vehicles will be on the same road, but we do not know their exactly positions. Calculate the distance $d_j^{\text{while waiting 0}}$, which the neighbour vehicle rode while waited for the own vehicle turn for the time $t_{0,OE}$

$$d_j^{\text{while waiting 0}} = speed_j \cdot t_{0,OE} + \frac{accel_j \cdot t_{0,OE}^2}{2}$$

But we still does know, the exactly coordinate where the neighbour vehicle will be. Get it from the definition of distance

$$d_j^{\text{while waiting 0}} = \sqrt{(x_j - x_j^{\text{while waiting 0}})^2}$$

Coordinate while the j^{th} neighbour waiting for the own vehicle ("plus" for the east direction, "minus" for the west) calculating by the following equation:

$$x_j^{\text{while waiting 0}} = x_j \pm d_j^{\text{while waiting 0}} \tag{4.55}$$

Coordinates of v_j after $t_{0,OE}$ are in the node **F**:

$$\mathbf{F} = (x_j \pm d_j^{\text{while waiting 0}}, y_j) \tag{4.56}$$

Coordinates of v_0 after $t_{0,OE}$ are in the node **E**:

$$\mathbf{E} = (x_0, y_j) \tag{4.57}$$

Finally, vehicles are on the same road, let us calculate the distance between two vehicles on the same road

$$\begin{aligned} d^{\text{between 0 and j}} &= \sqrt{\Big(\underbrace{x_j}_{x_j \pm d_j^{\text{while waiting 0}}} - x_0\Big)^2 + \Big(y_j - \underbrace{y_0}_{y_j}\Big)^2} = \\ &= \sqrt{\Big(x_j \pm d_j^{\text{while waiting 0}} - x_0\Big)^2} \end{aligned} \tag{4.58}$$

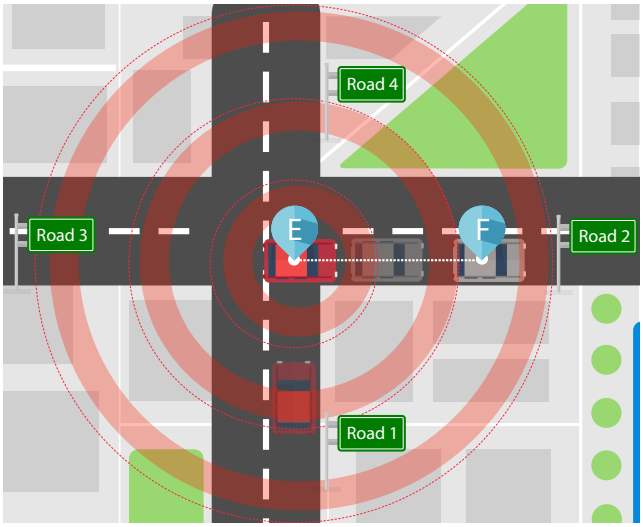


Figure 4.20: Vehicles are on the same road

Now according to the values of $accel_j$ and $accel_0$ choose an appropriate formula (4.34) or (4.35), then calculate connection time t_j^{con}. So the total connection time of the neighbour vehicle in case of horizontal rode and then turning own vehicle to the same rode as the neighbour vehicle will be:

$$t_j^{TL,con} = t_{j,JE} + t_j^{\text{con}} \tag{4.59}$$

Horizontal road v_0 → Vertical road v_j
According to this situation an own vehicle rides on a horizontal road and wants to turn to a vertical road (both cases included to north and to south) and there is rides a potential neighbour vehicle (see Fig. 4.13). We need to know its time connection.

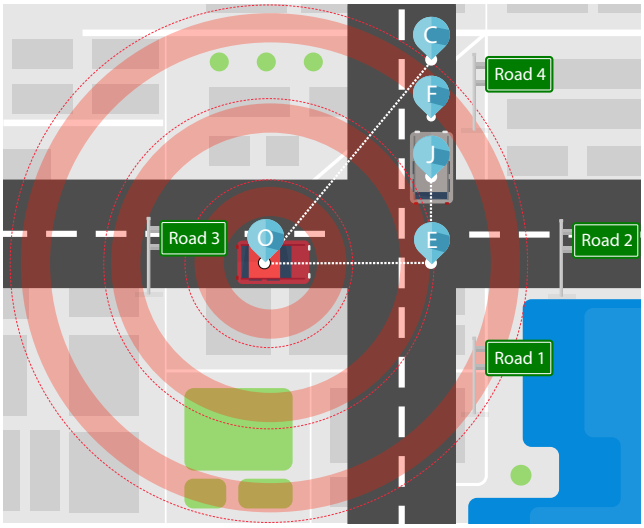


Figure 4.21: Horizontal road $v_0 \rightarrow$ Vertical road v_j

Find out the threshold node, where the neighbour vehicle leave the signal radius (see Fig. 4.13, node **C**). Derive it from the definition of radius

$$\begin{aligned} d_{OC} = r = \sqrt{(x_0 - x_C)^2 + (y_0 - y_C)^2} = \\ = \sqrt{(x_0 - x_j)^2 + (y_0 - y_C)^2} \end{aligned} \tag{4.60}$$

$$y_C = y_0 \pm \sqrt{-x_j^2 + 2x_0x_j + r^2 - x_0^2}, \tag{4.61}$$

For north direction of the own vehicle "+" sign, for south direction "-" sign)

Distance d_{JC} needed to leave our signal range for the neighbour vehicle will be

$$\begin{aligned} d_{JC} = \sqrt{(x_j - x_C)^2 + (y_j - y_C)^2} = \\ = \sqrt{\left(y_j - \left(y_0 \pm \sqrt{-x_j^2 + 2x_0x_j + r^2 - x_0^2}\right)\right)^2} \end{aligned} \tag{4.62}$$

If we know distance, we can calculate the time $t_{j,JC}$

$$d_{JC} = speed_j \cdot t_{j,JC} + \frac{accel_j \cdot t_{j,JC}^2}{2}$$

$$\frac{accel_j \cdot t_{j,JC}^2}{2} + speed_j \cdot t_{j,JC} - d_{JC} = 0$$

$$
\begin{aligned}
t_{j,JC} = & \frac{\pm\sqrt{2 \cdot accel_j \cdot d_{JC} + speed_j^2} - speed_j}{accel_j} = \\
= & \frac{\sqrt{2 \cdot accel_j \cdot d_{JC} + speed_j^2} - speed_j}{accel_j} = \\
= & \frac{\sqrt{2 \cdot accel_j \cdot \sqrt{\left(y_j - (y_0 \pm \sqrt{-x_j^2 + 2x_0x_j + r^2 - x_0^2})\right)^2} + speed_j^2} - speed_j}{accel_j}
\end{aligned}
\tag{4.63}
$$

Consider the own vehicle and calculate distance d_{OE} and time $t_{0,OE}$ to get an intersection.

$$
\begin{aligned}
d_{OE} = r = & \sqrt{(x_0 - x_E)^2 + (y_0 - y_E)^2} = \\
= & \sqrt{(x_0 - x_E)^2} = \\
= & \sqrt{(x_0 - x_j)^2}
\end{aligned}
\tag{4.64}
$$

$$
d_{OE} = speed_0 \cdot t_{0,OE} + \frac{accel_0 \cdot t_{0j,OE}^2}{2}
$$

$$
\frac{accel_0 \cdot t_{0,OE}^2}{2} + speed_0 \cdot t_{0,OE} - d_{OE} = 0
$$

$$
\begin{aligned}
t_{0,OE} = & \frac{\pm\sqrt{2 \cdot accel_0 \cdot d_{OE} + speed_0^2} - speed_0}{accel_0} = \\
= & \frac{\sqrt{2 \cdot accel_0 \cdot d_{OE} + speed_0^2} - speed_0}{accel_0} = \\
= & \frac{\sqrt{2 \cdot accel_0 \cdot \sqrt{(x_0 - x_j)^2} + speed_0^2} - speed_0}{accel_0}
\end{aligned}
\tag{4.65}
$$

So we know $t_{j,JC}$ and $t_{0,OE}$, it is logically if the own vehicle needs more time to get to intersection, than the neighbour vehicle leave signal range, then we lose connection with v_j and our connection time with v_j will be $t_{j,JC}$.

$$
\text{If } t_{0,OE} > t_{j,JC} \implies t_j^{con} = t_{j,JC} \tag{4.66}
$$

If $t_{0,OE} \leq t_{j,JC}$, which means the own vehicle will be faster than the neighbour vehicle. It requires further calculations to get the total connection time. In that case vehicles will be on the same road, but we do not know their exactly positions.

We can calculate the distance $d_j^{\text{while waiting 0}}$, which the neighbour vehicle rode while was waiting for the own vehicle for time reaching the intersection by own vehicle $t_{0,OE}$

$$
d_j^{\text{while waiting 0}} = speed_j \cdot t_{0,OE} + \frac{accel_j \cdot t_{0,OE}^2}{2}
$$

But we still does not know, the exactly coordinate where the own vehicle will be. Get it from the definition of distance

$$d_j^{\text{while waiting 0}} = \sqrt{(y_j - y_j^{\text{while waiting 0}})^2}$$

$$y_j^{\text{while waiting 0}} = y_j \pm d_j^{\text{while waiting 0}} \tag{4.67}$$

where "+" sign if the vehicles will going to north and "-" for south direction

Vehicles are on the same road (see Fig. 4.18) and their coordinates will be
Coordinates of v_0 after $t_{0,OE}$ are in the node F:

$$F = (x_j, y_0)$$

Coordinates of v_j after $t_{j,OE}$ are in the node E:

$$E = (x_j, y_j \pm d_j^{\text{while waiting 0}})$$

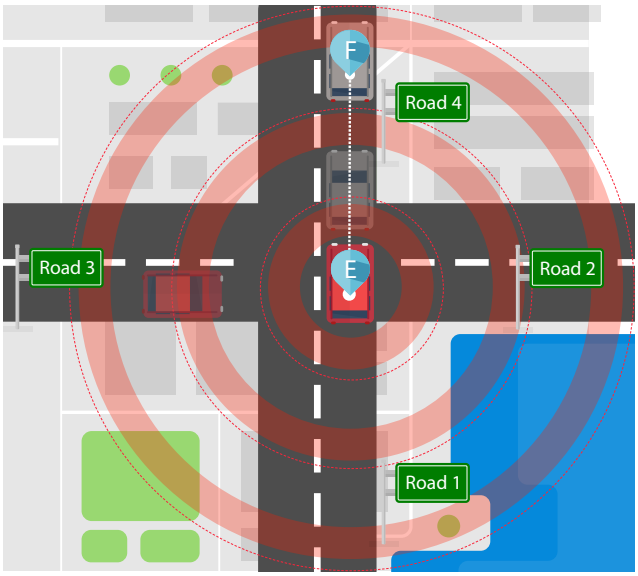


Figure 4.22: Vehicles are on the same road

The distance between two vehicles on the same road will be

$$d^{\text{between 0 and j}} = \sqrt{\left(x_j - \underbrace{x_0}_{x_j}\right)^2 + \left(\underbrace{y_j}_{y_j \pm d_j^{\text{while waiting 0}}} - y_0\right)^2} = \tag{4.68}$$

$$= \sqrt{\left(y_j \pm d_j^{\text{while waiting 0}} - y_0\right)^2}$$

Now according to the values of $accel_j$ and $accel_0$ choose an appropriate formula (4.34) or (4.35), then calculate connection time t_j^{con}. So the total connection time of the neighbour vehicle in case of vertical road and then turning to the same rode as own vehicle will be:

$$t_j^{TL,con} = t_{j,JE} + t_j^{\text{con}} \tag{4.69}$$

Horizontal road v_j → Vertical road v_0

The last case of prediction parameter ψ_j^{con}, when the own vehicle rides on vertical road and get connected to the neighbour vehicle, that rides on the horizontal rode and after turns to the same road as own vehicle (see Fig. 4.18).

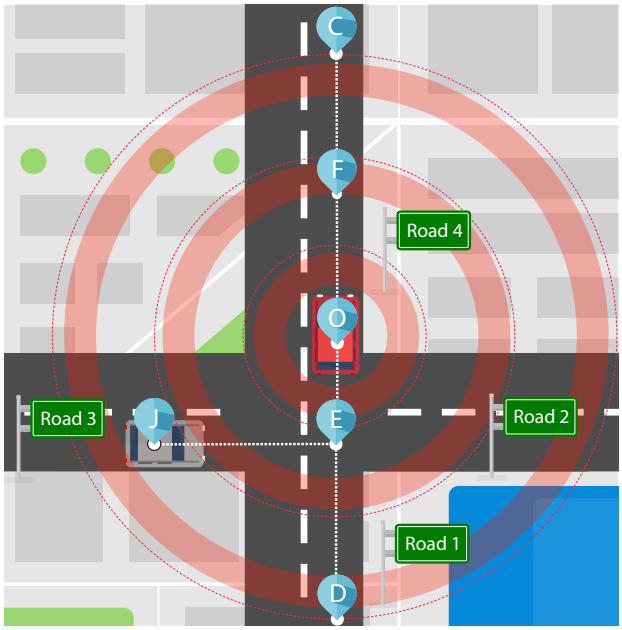


Figure 4.23: Horizontal road v_j → Vertical road v_0

Calculate d_{JE} a distance between j^{th} vehicle and an intersection

$$\begin{aligned} d_{JE} &= \sqrt{(x_j - x_E)^2 + (y_j - y_E)^2)} = \\ &= \sqrt{(x_j - x_E)^2} = \\ &= \sqrt{(x_j - x_0)^2} \end{aligned} \tag{4.70}$$

Then calculate time $t_{j,JE}$ from definition of the distance d_{JE}

$$d_{JE} = speed_j \cdot t_{j,JE} + \frac{accel_j \cdot t_{j,JE}^2}{2} \tag{4.71}$$

$$\frac{accel_j \cdot t_{j,JE}^2}{2} + speed_j \cdot t_{j,JE} - d_{JE} = 0$$

$$
\begin{aligned}
t_{j,JE} = & \frac{\pm\sqrt{2 \cdot accel_j \cdot d_{JE} + speed_j^2} - speed_j}{accel_j} = \\
= & \frac{\sqrt{2 \cdot accel_j \cdot d_{JE} + speed_j^2} - speed_j}{accel_j} = \\
= & \frac{\sqrt{2 \cdot accel_j \cdot \sqrt{(x_j - x_0)^2} + speed_j^2} - speed_j}{accel_j}
\end{aligned}
\tag{4.72}
$$

Then calculate the distance d_{DE}, after that the own vehicle will not reach the intersection.

$$
\begin{aligned}
d_{DE} &= r - d_{EO} = \\
&= r - \sqrt{(x_E - x_O)^2 + (y_E - y_O)^2)} = \\
&= r - \sqrt{(y_E - y_O)^2} = \\
&= r - \sqrt{(y_j - y_0)^2}
\end{aligned}
\tag{4.73}
$$

The time $t_{0,DE}$ when the own vehicle will not reach the intersection

$$d_{DE} = speed_0 \cdot t_{0,DE} + \frac{accel_0 \cdot t_{0,DE}^2}{2}$$

$$\frac{accel_0 \cdot t_{0,DE}^2}{2} + speed_0 \cdot t_{0,DE} - d_{DE} = 0$$

$$
\begin{aligned}
t_{0,DE} = & \frac{\pm\sqrt{2 \cdot accel_0 \cdot d_{DE} + speed_0^2} - speed_0}{accel_0} = \\
= & \frac{\sqrt{2 \cdot accel_0 \cdot d_{DE} + speed_0^2} - speed_0}{accel_0} = \\
= & \frac{\sqrt{2 \cdot accel_0 \cdot (r - \sqrt{(y_j - y_0)^2}) + speed_0^2} - speed_0}{accel_0}
\end{aligned}
\tag{4.74}
$$

So we know $t_{j,JE}$ and $t_{0,DE}$ if the own vehicle needs more time to get to intersection than the neighbour vehicle to be out of reach, then we lose connection with v_j and our connection time with v_j will be $t_{0,DE}$.

$$\text{If } t_{j,JE} > t_{0,DE} \implies t_j^{con} = t_{0,DE} \tag{4.75}$$

If $t_{j,JE} \leq t_{0,DE}$, which means the neighbour vehicle will be faster than the own vehicle will lose its connection. It requires further calculations to get connection time t_j^{con}. In that case vehicles will be on the same road (see Fig. 4.24), but we do not know their exactly positions.

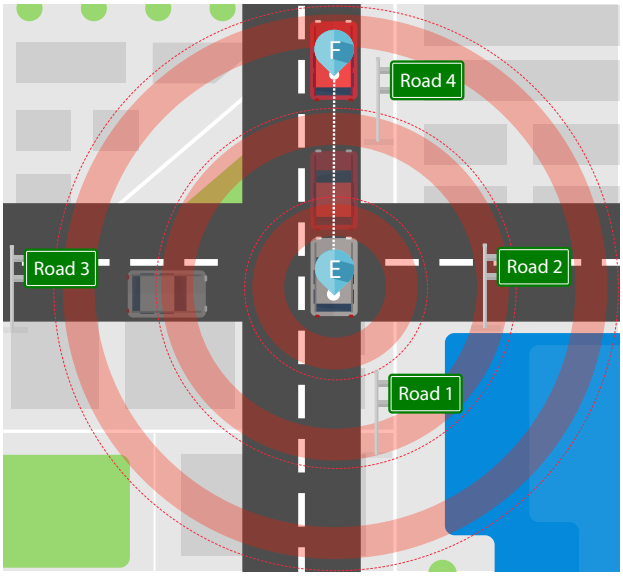


Figure 4.24: Vertical road $v_0 \rightarrow$ horizontal road v_j

Calculate the distance $d_0^{\text{while waiting j}}$, which the own vehicle rode while waited for a turn of the neighbour vehicle for a time $t_{j,JE}$

$$d_0^{\text{while waiting j}} = speed_0 \cdot t_{j,JE} + \frac{accel_0 \cdot t_{j,JE}^2}{2}$$

But we still does know, the exactly coordinate where the neighbour vehicle will be. Get it from the definition of distance

$$d_0^{\text{while waiting j}} = \sqrt{(y_0 - y_0^{\text{while waiting j}})^2}$$

Coordinate y while the own vehicle waiting for the neighbour vehicle ("plus" for the north direction, "minus" for the south) calculating by the following equation:

$$y_0^{\text{while waiting j}} = y_0 \pm d_0^{\text{while waiting j}} \tag{4.76}$$

Coordinates of v_0 after $t_{j,JE}$ are in the node **F**:

$$\mathbf{F} = (x_0, y_0 \pm d_0^{\text{while waiting j}}) \tag{4.77}$$

Coordinates of v_j after $t_{j,JE}$ are in the node **E**:

$$\mathbf{E} = (x_0, y_j) \tag{4.78}$$

The distance between two vehicles on the same road will be

$$\begin{aligned} d^{\text{between 0 and j}} &= \sqrt{\left(\underbrace{x_j}_{x_0} - x_0\right)^2 + \left(y_j - \underbrace{y_0}_{y_0 \pm d_0^{\text{while waiting j}}}\right)^2} = \\ &= \sqrt{\left(y_j - (y_0 \pm d_0^{\text{while waiting j}})\right)^2} \end{aligned} \tag{4.79}$$

Now according to the values of $accel_j$ and $accel_0$ choose an appropriate formula (4.34) or (4.35), then calculate connection time t_j^{con}. So the total connection time of the neighbour vehicle in case of horizontal rode and then turning to the vertical road, where rides own vehicle will be:

$$t_j^{TL,con} = t_{j,JE} + t_j^{\text{con}} \tag{4.80}$$

Finally we know all cases for the last component of probability function $P_s(v_j, x_i)$ - prediction parameter ψ_j^{con}. Remember, that parameter predicts how long j^{th} vehicle will travelled within our range of signal. If any vehicle needs more time to get task, execute it and send back than connection time, that vehicle will be removed automatically.

Definition 4.2.9. *Let $t_j^{TL,con}$ is a total connection time of j^{th} vehicle, $T_j^{total}(v_j, x_i)$ is a total time. If the total time is less than total connection time the prediction parameter ψ_j^{con} returns 1, other wise 0. That means that execute i^{th} task makes no sense, if we know that the neighbour vehicle will not be in our radius the whole total time. So prediction parameter ψ_j^{con} can be defined*

$$\bullet\ \psi_j^{con}\ (v_j, x_i) = \begin{cases} 1, & \textit{if } T_j^{total}(v_j, x_i) < t_j^{con}(v_j, x_i) \\ 0, & \textit{otherwise.} \end{cases} \tag{4.81}$$

4.2.3 Minimal probability

Definition 4.2.10. *Let $P_s(v_j, x_i)$ be a success probability of j^{th} vehicle to execute i^{th} task, α_i is a relative priority between response time and success probability for i^{th} task, so the minimal probability is a multiplication of priority and the mean value of all neighbour vehicles' probabilities*

$$P_{min} = \alpha_i \cdot \frac{\sum_{j=1}^{N} P_s(v_j, x_i)}{N} \tag{4.82}$$

4.2.4 Creating a cluster

A minimal probability needs to filter vehicles by priority of the tasks. If a task has a high priority α_i, which means that probability of completing i^{th} task must be as great as possible, vehicles with low probability will not be considered and will be created a new cluster.

$$\mathcal{N}^{cluster} \leftarrow (P_s(v_j, x_i) \geq P_{min}) \tag{4.83}$$

$$\mathcal{N}^{cluster} = \{v_{0^*}, v_{1^*}, ...v_{j^*}, ...v_{N^*}\}, \tag{4.84}$$

where $\mathrm{N}^* \subseteq N,\ j^* \in \langle 0, N^* \rangle, j^* \in \mathbb{N}$

4.2.5 Considering function

After getting all necessary information from vehicles, we need to consider which one for certain α_i is the most optimal. Some tasks can be delay-tolerant but require higher probability for being processed successfully. Some other real-time applications might emphasize on the total time rather than success probability.

Was considered, that we divide tasks on two groups. There are important tasks and normal tasks. For important tasks success probability has more weight than the total time, for normal tasks and for normal tasks respectively on the contrary. But can happened, that two vehicles have similar parameters, for example $P_s(v_j, x_i) = 0.9$ and $T_j^{TL} = 4$ [s] and $P_s(v_{j+1}, x_i) = 0.8$ and $T_{j+1}^{TL} = 3$ [s] in such case considering function f_j^{cons} was built.

Definition 4.2.11. *Let $T_{j^*}^{TL}$ is a total time for j^{*th} vehicle, $P_s(v_{j^*}, x_i)$ is a success probability of j^{*th} vehicle, so considering function for two different priorities can be defined as*

$$f_j^{cons}(v_{j^*}, x_i) = \\ = \begin{cases} \min\limits_{\forall j^* \in \mathcal{N}^{cluster},\ i \in Q} \left\{ \left(P_s(v_{j^*}, x_i)^{-1} \right)^{1.5} \cdot \dfrac{T_{j^*}^{TL}(v_{j^*}, x_i)}{\max\limits_{\forall j^* \in \mathcal{N}^{cluster},\ i \in Q} \{T_{j^*}^{TL}(v_{j^*}, x_i)\}} \right\}, & \text{for } \alpha_1 \\ \\ \min\limits_{\forall j^* \in \mathcal{N}^{cluster},\ i \in Q} \left\{ \left(P_s(v_{j^*}, x_i)^{-1} \right)^{0.5} \cdot \dfrac{T_{j^*}^{TL}(v_{j^*}, x_i)}{\max\limits_{\forall j^* \in \mathcal{N}^{cluster},\ i \in Q} \{T_{j^*}^{TL}(v_{j^*}, x_i)\}} \right\}, & \text{for } \alpha_2 \end{cases} \tag{4.85}$$

Note that without the power on success probability each factor of multiplication has the same weight and take values from 0 to 1. For high priority or for important tasks first equation is considered. Increasing the power of vehicle success probability increase its weight, other words is more important to execute the task with higher probability than to complete it as soon as possible. Another case is to decrease the power, accordingly the weight will be decreased, which means that to completing time is more important that the success probability.

4.3 Task scheduling algorithm

Finally we have defined all functions of the proposed algorithm. Task scheduling algorithm summarize all functions above and can be expressed as

The complexity of Algorithm 1 is quite straightforward to follow. The statements no.$1 - 3$ are enclosed in a loop and checks whether queue is empty or not. When we get some task use $f^{pick}(x_i)$ function to get a task with the lowest deadline (line no.4). Then we provide reactive scanning, which means, that we do not provide any scanning in advance, an own vehicle scan the network only after getting some task (line no.5). After scanning we will get a list on neighbour vehicles and add to that list an own vehicle (line no.6), because it also can execute task x_i. For statements no.$7 - 14$ we calculate for every vehicle in the list of

Algorithm 1 Task scheduling algorithm

Input : $\mathcal{Q}^{tasks} = \{x_1, x_2, \ldots, x_i, \ldots, x_I\}$
Output: Sending task x_i to v_j for execution

while $\mathcal{Q}^{tasks} \in \varnothing$ **do**
  check every τ_1
end
Calculate $f^{pick}(x_i)$ from $\mathcal{Q}^{tasks}$ (section 4.2.1)
Reactive scanning, getting list $\mathcal{N}^{neighb}$
$\mathcal{N}^{neighb} \leftarrow v_0$
for *each* $v_j \in \mathcal{N}^{neighb}$ **do**
  Calculate $T_j^{TL}(v_j, x_i)$ using (4.16)
  Calculate $\phi_j^{DL}(v_j, x_i)$ using (4.17)
  Calculate $\beta_j^{beh}(v_j, x_i)$ using (4.26)
  Calculate $\psi_j^{con}(v_j, x_i)$ using (4.81)
  Calculate $P_s(v_j, x_i)$ using (4.3)
end
Calculate P_{min} using (4.82)
$\mathcal{N}^{cluster} \leftarrow \{v_0, v_1, ...v_{j^*}, ...v_{N^*} |\ P_s(v_j, x_i) \geq P_{min},\ N^* \subseteq N\}$
Calculate $f_j^{cons}(v_{j^*}, x_i)$ using (4.2.5)
if $f_j^{cons}(v_{j^*}, x_i) \in \varnothing$ **then**
  if $\overline{T_j^{TL}} + \xi^{reserve} \leq DL_i$ **then**
    go to step #5
  else
    drop x_i task
    go to step #1
  end
else
  sending task x_i to v_j for execution
  go to step #1
end

neighbour vehicles all necessary functions. In the statement no.15 the cluster of vehicles will be provided, vehicles with higher value that the minimal probability will be chosen. The last loop on statements no.18 − 27 if the considering function does not return vehicle, we have have two options: 1) try to provide the scanning again in case, that we have enough time (lines no.18 − 19) 2) drop a task 2) if we do not have enough time (lines no.21 − 23), the own vehicle drops task and goes to statement no.1. If the considering functions returns appropriate vehicle for the picked task we send it to that vehicle and go to statement no.1.

Chapter 5

Simulations

In this section, simulations are presented to estimate proposed algorithm and compare it with existing one. Microscopic traffic simulator SUMO (simulation of urban mobility) are used with further integration in MATLAB. Sumo emphasize local behavior of individual vehicles by representing the velocity and position of each vehicle at a given moment. This type of simulation is especially helpful for studying localized traffic interactions, but it comes with the price of reduced scalability [38].

Simulation consists of two interacting applications: SUMO and MATLAB. SUMO is a server and MATLAB is a client. The simulation was written in the style of OOP (Object-Oriented Programming).

The name of starting simulation script is *general.m*. All the parameters of the simulation can be changed in it. There are the number of neighbour vehicles, the number of tasks per second, the length of simulation and the other things.

At the start we execute SUMO, which runs in a server mode, then using the API interface $Traci4Matlab$ we can get the necessary values of each vehicle, there are (*id vehicle*, x, y, *road*, *speed*). But to implement our algorithm, we also need to know the acceleration of the vehicles in each step. Only the maximum possible acceleration is realized in SUMO, but not an actual value. Therefore, acceleration was calculated for each step by using following formula

$$accel_{\,step_x} = \left. \frac{speed_{step_x} - speed_{\,step_{x-1}}}{step\ length} \right|_{step\ \ length = 0.2\,[s]} \tag{5.1}$$

These values in the process of interaction of SUMO and MATLAB are saved in the MATLAB's object Data. That means, we do not read the .xml that SUMO also generates, but use the interface API $Traci4Matlab$, because it requires further calculations.

Data is a classic map, where the key is the simulation step (time) of the simulation and the value is all four vehicles' parameters in this step. As we have all the values of all vehicles in each step of the simulation, we run our written system.

Let us consider the basic concept of the simulation. There is an object $Timer$ that starts first. Then the Task and $Vehicle$ objects are connecting using event-listener interface. The aim of the $Timer$ is to notify connected objects that a certain time has passed (the simulation step). The aim of the $Task$ to change its deadline on the simulation step (in our case 0.2

[s]), when it notified by the timer. The aim of the $Vehicle$ is to change its four parameters ($id\ vehicle, x, y, road, speed$) each step of the simulation (when it is notified by the Timer). The $Vehicle$ object also reads the Data object.

Was implemented so-called life cycle for Task object. Life cycle consist of 7 states. There are: "$Born$", "$InProcess$", "$Sending$", "$InExecution$", "$Receiving$", "$Success$", "$Dead$".

- $Born$: i^{th} task gets state "$Born$", when it was born by the $Factory\ of\ Tasks$ and was added to the $Heap$ of the own vehicle v_0.
- $InProcess$: i^{th} task gets state "$InProcess$", when it was picked by $f^{pick}(x_i)$ (section 4.2.1) from the *binaryheap*.
- $Sending$: i^{th} task gets state "$Sending$", when it was sending to the neighbour vehicle v_j.
- $InExecution$: i^{th} task gets state "$InExecution$", when it was successfully got by neighbour vehicle v_j and started to executed the certain task x_i.
- $Receiving$: i^{th} task gets state "$Receiving$", when it was executed and is receiving by the own vehicle v_0.
- $Success$: i^{th} task gets state "$Success$", when it was successfully received and complete.
- $Dead$: i^{th} task gets state "$Dead$", when it was expired or executed.

Each task x_i can have at least two of these states ("$Born$" and "$Dead$") in case no vehicle were to execute it, and in the best case all states of life cycle.

All states mention above were implemented to simulate proposed algorithm and help to verify the effectiveness of our solution. The whole algorithm implemented in the separate class "Flowchart".

5.1 Simulation assumptions and scenario

In order to conduct simulation for proposed algorithm, one needs to build simulation scenario that imitates the reality. The simulation is divided into two parts: the first part is the mobility and traffic generation which we generate using SUMO, and the second part is the network simulation which is performed using MATLAB. SUMO is an open source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks. It is mainly developed by employees of the Institute of Transportation Systems at the German Aerospace Center [39].

Improved Krauss car-following model has been modified by SUMO to make it more suitable for the real situation. This is the main reason for choosing such a model, it allows us to get results close to real implementation. The estimated speed of vehicles related to the speed and distance between the front cars, the rear cars [40] and their safe speed.

Safe speed (5.2) is a speed of a vehicle in relation to the vehicle ahead of it. It illustrates that the following vehicle is always trying to keep a safe distance with leading vehicle. The

following vehicle always adapt to the deceleration behavior of the leading vehicle [41]. Safe speed is computed as follows:

$$v_{\text{safe}} = v_l(t) + \frac{g(t) - v_l(t)t_r}{\frac{v_l(t) + v_f(t)}{2b} + t_r}, \tag{5.2}$$

Where $v_l(t)$ represents speed of the leading vehicle in time t, $v_f(t)$ represents speed of the following vehicle in time t, g is gap to the leading vehicle in time t, t_r is the driver's reaction time (about $1s$) and b is the maximum deceleration of the vehicle $[m/s^2]$.

Simulation is run for 1800 seconds in SUMO and the results of 5 individual simulations are averaged to get one point on figure. The main reason is to reduce the influence of random variability in the inputs. All parameters are shown in the table 5.1

Table 5.1: Simulation parameters

Parameters	Values
Car-Following Model	Krauss model
Size of map	166 x 166 [m] [1]
Road Length	55 and 84 [m] [1]
Lane Width	3.7 [m]
Number of Lanes	2 and 3 [-]
Number of Intersections	3 [-]
Simulation Duration	1800 [s] [1]
Number of Vehc1es	0 - 40 [-]
Max vehicle's Speed	50 [km/h] or 13.8 [m/s] [31]
Max vehicle's Acceleration	2.5 $[m/s^2]$
Max vehicle's Deceleration	4.5 $[m/s^2]$
Transmission Rate	1.9 - 5.5 [Mbit/s]
Transmission Radius r	50 [m]
Arrival Rate	60 - 420 [tasks/min] or 1-7 [tasks/s]
DS (Data Size)	0.5 - 5 [MB]
DS^{result}(Data Size Result)	0.5 - 5 [MB]
DL (Deadline)	1 - 3 [s]
Relative Priority α	0.8 [-]
IS (Instruction set)	2000 [MI] [1]
Number of Cores per CPU	1 [-]
Number Threads per Core	1 [-]
$MIPS_j$	1500 - 2500 [MI/s] [1]
$MIPS_0$	2000 [MI/s]

An urban environment of square size with 166 m length on each side is created into SUMO (see Fig. 5.1). Simulation scenario consist of one-way roads and 3 intersections (dashed circles on the Fig. 5.1).

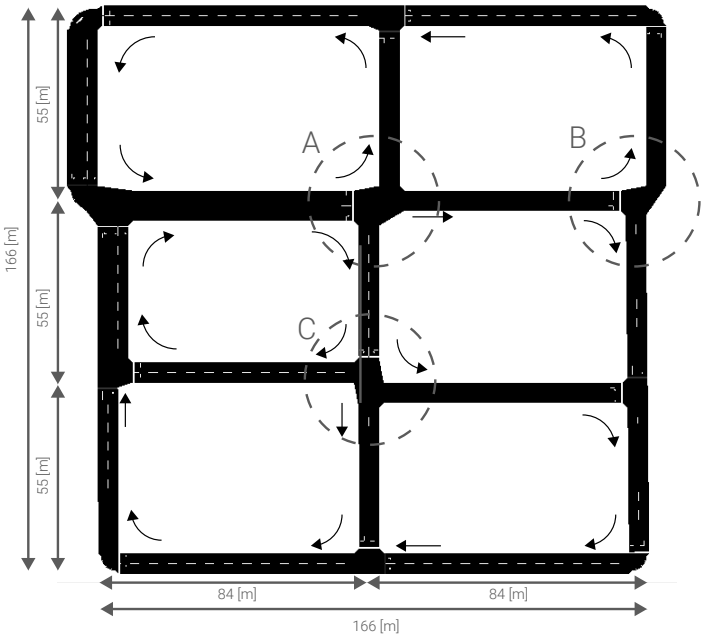
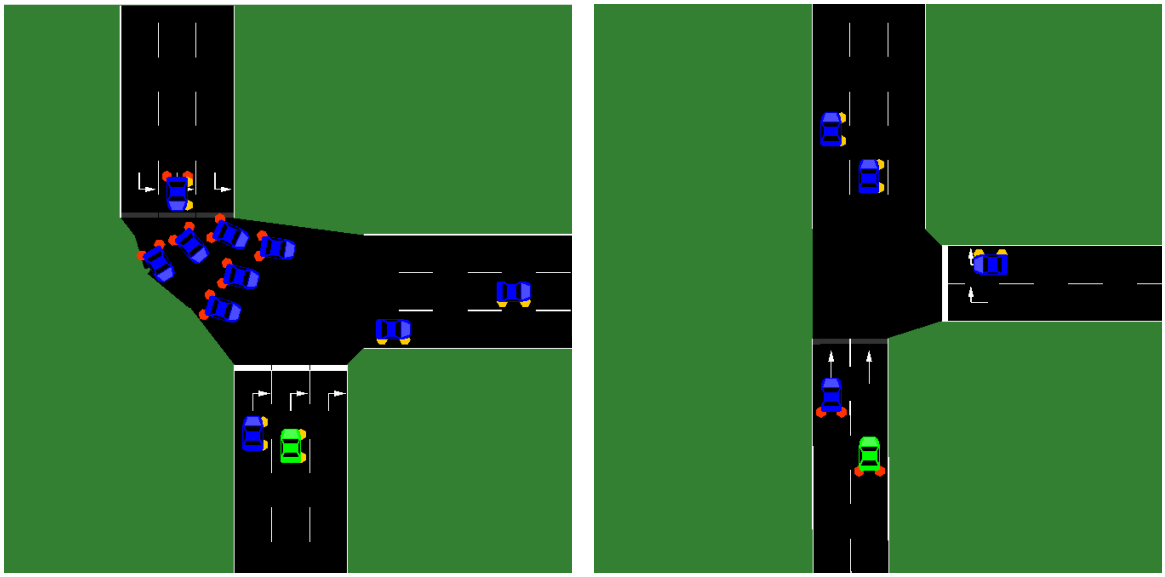


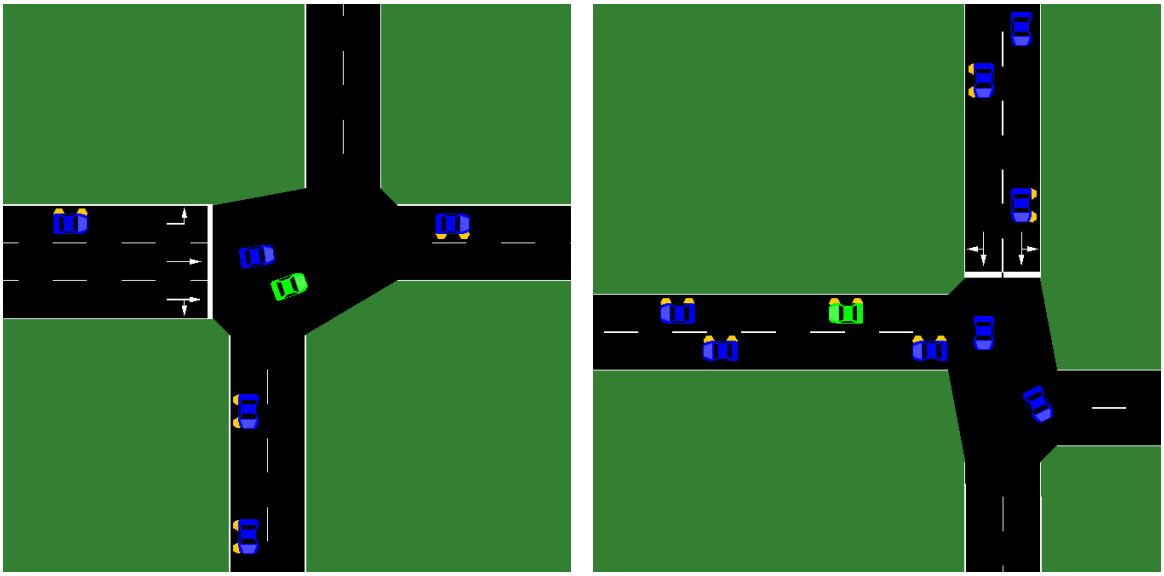
Figure 5.1: Simulation scenario

SUMO can show the real streets or the streets you design with lanes, traffic lights and other things the real world has. It allows to simulate a given traffic demand or dynamic assignments by users. SUMO allows addressing a large set of traffic management topics and gives a lot of help to VANET simulation [42]. Fig. 5.2 and Fig. 5.3 show snapshots of SUMO output at a specific point of time.



a) b)

Figure 5.2: Snapshots of Mobility Scenario Generated by SUMO



a) b)

Figure 5.3: Snapshots of Mobility Scenario Generated by SUMO

5.2 Algorithms

Was considered five algorithms. There are our *task scheduling algorithm*, $QARTS-1$, $QARTS-2$, *random* and *computation by itself*.

QARTS-1 Heuristic task scheduling algorithm was built. Was exploited MapReduce computation model to address the problem of resource heterogeneity and to support computation parallelization. They do not consider any kind of prediction and looking a vehicle with the minimal value of total time T^{TL} [1]. Note that value of time connection, other words a time, how long travelled the neighbour vehicle withing signal range was taken for a constant $(5-10\ s)$, because there is no information about it. They also did not define the value of reception time, so it was chosen as a mean value of all reception times. Picking the tasks carried out by FIFO (First-In-First-Out).

QARTS-2 QARTS-2 is an improved version of QARTS-1 algorithm. Was used the same core, but several features was taken from our algorithm. The f^{pick} function considered for queue of tasks, also was improved the reception time, it is no more constant and predicted by us (4.15)

Our task scheduling algorithm In our approach tasks automatically sorted using binary tree algorithm. And always pick with the lowest value of deadline, if there are more than one such tasks, then we sort them by priority. Success probability function P_s (4.3) provides three types of filtering, that increase choosing accuracy and reliability. Then considering

function on the basis of two parameters looks an optimal vehicle. Even more if we do not have any suitable vehicle, we provide scanning again in case if we have enough time.

Random In that case when the own vehicle is busy it choose randomly a neighbour vehicle for task execution. If there is no vehicle and the own vehicle is busy task will be dropped.

Computation by itself In that case all computations provides by the own vehicle itself. If we are already executing some task, another one in queue will be dropped.

5.3 Performance metrics

Successful task execution (5.3) and the system throughput (5.4) are selected as the performance metrics to evaluate the performance of algorithms in this thesis.

The successful task execution can be obtained from the total number of successful tasks (tasks, that arrived back to the own vehicle) divided by the total number of sent tasks and dropped tasks by own vehicle. System throughput of computation models is measured by the number of task executed in it per unit of time.

The performance is better when number of received tasks is high. Mathematically they can be shown as following equations:

$$\text{Successful Task execution} = \frac{\text{completed tasks}}{\text{sent tasks + dropped tasks}} = \frac{\text{completed tasks}}{\text{all tasks}}\,[-] \tag{5.3}$$

$$\text{System Throughput} = \frac{\text{number of completed tasks}}{\text{time unit}}\,[\,\#\,\text{of tasks/time unit}] \tag{5.4}$$

They depend on various parameters chosen for the simulation. The major parameters are: vehicle density, task size and task arrival rate.

5.4 Simulation results

The remainder of this section is organized as follows. Three scenarios are considered, they are impacts impact of vehicle density, impact of task size and impact of task arrival rate on success task execution and on the system throughput.

Let us take a look on results of the main map on next sections.

5.4.1 Impacts of vehicle density

In Scenario I, the success probability of task execution and the system throughput depending on the vehicle density are presented in Fig. 5.4 and Fig. 5.5, respectively. Task arrival rate is 2 tasks per second, $DS_i = 1\,\text{MB}$, $DS_i^{result} = 1\,\text{MB}$, IS_i is 2000 MI, DL_i of each task is a random value between 1 s and 3 s. Performance of each vehicle is also random, so values of $MIPS_j$ are 1500-2500 MI/s, $MIPS_0 = 2000\ MI/s$.

Increasing the number of vehicles, increases available data processing resources and reduces job execution time, because we have more vehicles to choose. Also increases the transmitting data among the vehicles and thus decreases task transmission time and reception time, therefore the probability of execution task increases as well.

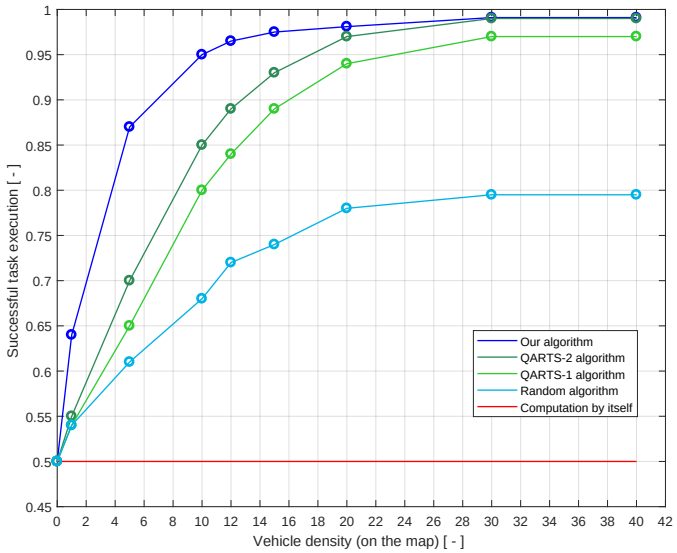


Figure 5.4: Impact of vehicle density on success probability for $\alpha = 0.8$

It is depicted that the percentage of successful job execution increases exponentially with the vehicle density. After 30 vehicles on the map the probability almost is equal 1 and constant (see Fig. 5.4). This is due to the fact, that the relative priority is 0.8, which means that the own vehicle sends high priority tasks and for us is more important the high probability of execution task than the total computation time. However, the probability is high enough for QARTS algorithms as well on high vehicle density (after 20 vehicles). Noticeably, that the prediction of our algorithm gives a significant advantage approximately until 16 vehicles on the map and the maximal improvement around 20% Fig. 5.4) or 21 tasks/min can be executed more Fig. 5.5) on 5 vehicles on the map. Successful task execution of the own vehicle is always 0.5, because we have a time just for each second task (task arrival rate is 2 and the own vehicle can execute 1 task per second) and another half of tasks are dropped, because we are busy.

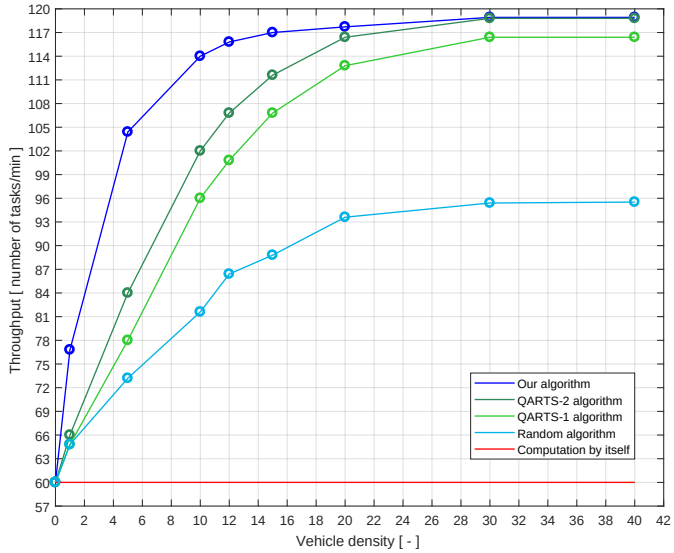


Figure 5.5: Impact of vehicle density on the system throughput for $\alpha = 0.8$

The random algorithm is slightly better than computation by itself and after 22 vehicles can be used for video streams, where the probability is not critical, but is not effective enough to use it all time.

5.4.2 Impacts of task size

In Scenario II, the success probability of task execution and the system throughput depending on the task size for $\alpha = 0.8$ are presented in Figure 5.6 and Figure 5.7, respectively. Vehicle density setup on 20 per map, other simulation parameters are same as the scenario I.

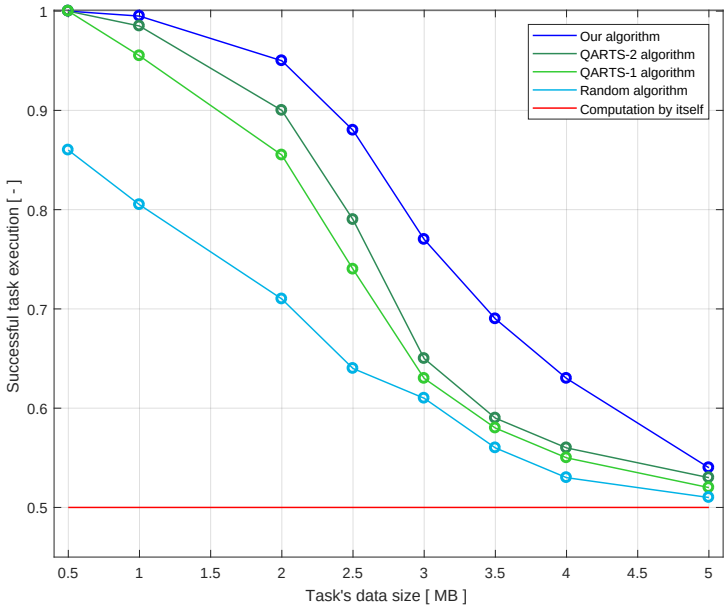


Figure 5.6: Impact of task size on success probability for $\alpha = 0.8$

Increasing size of task decreasing the probability exponentially till 3 MB, then it looks like linear decreasing. Increasing the task size the transmission time and reception time is increasing, so as more heavy the task as more accurate prediction needs. QARTS algorithm was built for *MapReduce* technology and slightly worse until the task less than 1 MB, then decreasing very fast. Our algorithm is more stable and reliable, but after 4 MB it is also not so good and needed more accurate probability to be used.

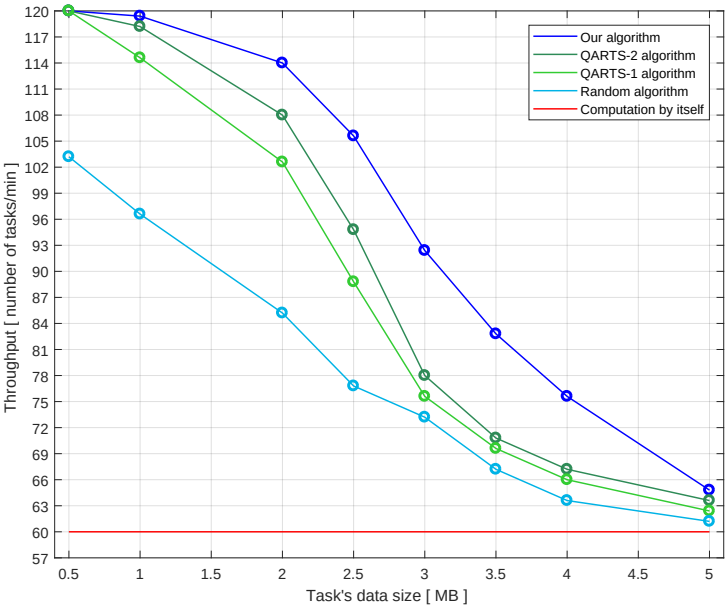


Figure 5.7: Impact of task size on the system throughput for $\alpha = 0.8$

5.4.3 Impacts of task arrival rate

In Scenario III, the success probability of task execution and the system throughput depending on the task arrival rate for $\alpha = 0.8$ are presented in Figure 5.8 and Figure 5.9, respectively. $DS_i = 1\,\text{MB}$, $DS_i^{result} = 1\,\text{MB}$ were considered as optimal for fair comparison, other simulation parameters are same as the scenario II.

The figures depict the fact that the percentage of job execution decreases exponentially for increasing arrival rates. This is because additional computation load forces the system to choose relatively poor nodes for task execution. We see that our prediction approximately reliable until we get 240 tasks per minute, then the percentage is lower than 65%.

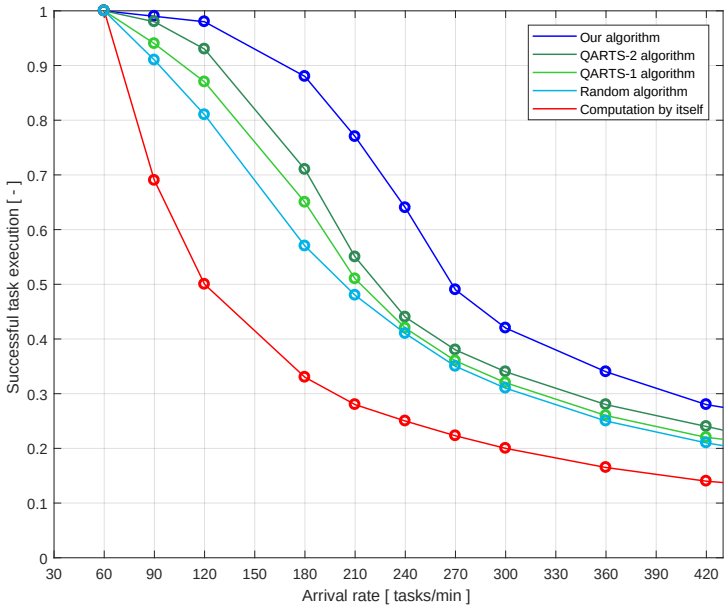


Figure 5.8: Impact of task arrival rate on success probability for $\alpha = 0.8$

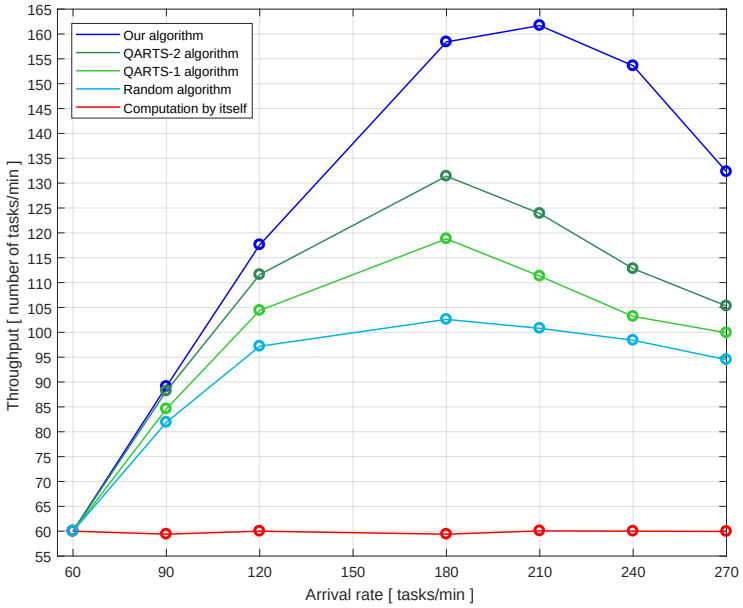


Figure 5.9: Impact of task arrival rate on the system throughput $\alpha = 0.8$

System throughput increases exponentially with the vehicle density until 210 tasks/min for our algorithm and until 180 taks/min for others, and and starts to decrease after reaching a saturation point. After these points the system becomes congested with a large number of tasks in the queue, that as a result leads to a decrease in productivity and the system throughput starts to decrease.

Chapter 6

Conclusion and future work

This thesis has proposed task scheduling algorithm for communication between vehicles in vehicular cloud computing. The new algorithm has a big potential. Important, that there are many opportunities to improve it, because many new functions have developed. We have got more stable and reliable results. However, if we have heavy traffic jam (high vehicle density) or small tasks QARTS algorithm [1] works good as well. We see, that for further deployment the VANET network cannot exist with advance prediction functions.

Comparison QARTS-1 with QARTS-2 (+4.3% or $+8\,tasks/min$), which we have got using optimal sorting of queue of tasks and prediction of reception time. Comparison Task scheduling algorithm with random algorithm (+28% or $+40\,tasks/min$) Comparison Task scheduling algorithm with QARTS-1 algorithm (+19% or $+31\,tasks/min$)

In the future, the algorithm proposed in this thesis can be widely adopted to study more advanced level algorithms. Multi-hop network with vehicle-to-infrastructure connection can be designed. Necessary to consider two-way roads scenario with traffic lights for more realistic results. Then finally after several improvements of functions it can be realized it in real life.

References

[1] T. Adhikary, A. K. Das, M. A. Razzaque, A. Almogren, M. Alrubaian, and M. M. Hassan, "Quality of service aware reliable task scheduling in vehicular cloud computing," *Mobile Networks and Applications*, vol. 21, pp. 482–493, June 2016.

[2] Z. Jiang, S. Zhou, X. Guo, and Z. Niu, "Task replication for deadline-constrained vehicular cloud computing: Optimal policy, performance analysis and implications on road traffic," *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.

[3] H. A. Najada and I. Mahgoub, "Anticipation and alert system of congestion and accidents in vanet using big data analysis for intelligent transportation systems," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, December 2016.

[4] C. Vyas, P. Wararkar, and S. S. Dorle, "Systematic analysis, design and implementation of prioritized vanet in real time application," in *nternational Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pp. 369–374, December 2016.

[5] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *J. Netw. Comput. Appl.*, vol. 40, pp. 325–344, April 2014.

[6] S. Olariu, T. Hristov, and G. Yan, *The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds*, pp. 645–700. John Wiley Sons, Inc., 2013.

[7] M. Gerla, "Vehicular cloud computing," in *The 11th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pp. 152–155, June 2012.

[8] A. Bazzi, B. M. Masini, A. Zanella, and I. Thibault, "Beaconing from connected vehicles: IEEE 802.11p vs. LTE-V2V," in *IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, September 2016.

[9] F. den Hartog, A. Raschella, F. Bouhafs, P. Kempker, B. Boltjes, and M. Seyedebrahimi, "A pathway to solving the Wi-Fi tragedy of the commons in apartment blocks," in *27th International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 1–6, November 2017.

[10] A. Bazzi, B. M. Masini, A. Zanella, and I. Thibault, "On the performance of IEEE 802.11p and LTE-V2V for the cooperative awareness of connected vehicles," *IEEE Transactions on Vehicular Technology*, vol. 66, pp. 10419–10432, November 2017.

[11] V. Prakaulya, N. Pareek, and U. Singh, "Network performance in ieee 802.11 and ieee 802.11p cluster based on vanet," in *International conference of Electronics, Communication and Aerospace Technology (ICECA)*, vol. 2, pp. 495–499, April 2017.

[12] R. Molina-Masegosa and J. Gozalvez, "LTE-V for Sidelink 5G V2X Vehicular Communications: A New 5G Technology for Short-Range Vehicle-to-Everything Communications," *IEEE Vehicular Technology Magazine*, vol. 12, pp. 30–39, December 2017.

[13] S. h. Sun, J. l. Hu, Y. Peng, X. m. Pan, L. Zhao, and J. y. Fang, "Support for vehicle-to-everything services based on lte," *IEEE Wireless Communications*, vol. 23, pp. 4–8, June 2016.

[14] M. Hasan and E. Hossain, "Resource allocation for network-integrated device-to-device communications using smart relays," in *IEEE Globecom Workshops (GC Wkshps)*, pp. 591–596, December 2013.

[15] 3GPP TR 36.785 (Release 14), *Technical Specification Group Radio Access Network; Vehicle to Vehicle (V2V) services based on LTE sidelink; User Equipment (UE) radio transmission and reception.*, October 2016.

[16] K. Mershad, H. Artail, and M. Gerla, "Roamer: Roadside units as message routers in vanets," *Ad Hoc Netw.*, vol. 10, pp. 479–496, May 2012.

[17] K. Mershad, H. Artail, and M. Gerla, "We can deliver messages to far vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, pp. 1099–1115, September 2012.

[18] R. Lu, X. Lin, H. Zhu, and X. Shen, "Spark: A new vanet-based smart parking scheme for large parking lots," in *IEEE INFOCOM*, pp. 1413–1421, April 2010.

[19] Y. Sun, X. Lin, R. Lu, X. Shen, and J. Su, "Roadside units deployment for efficient short-time certificate updating in vanets," in *IEEE International Conference on Communications*, pp. 1–5, May 2010.

[20] G. M. Björklund and L. Åberg, "Driver behaviour in intersections: Formal and informal traffic rules," vol. 8, pp. 239–253, 05 2005.

[21] M. A. Javed and J. Y. Khan, "Performance analysis of an adaptive rate-range control algorithm for vanet safety applications," in *International Conference on Computing, Networking and Communications (ICNC)*, pp. 418–423, February 2014.

[22] M. Fogue, F. J. Martinez, P. Garrido, M. Fiore, C. F. Chiasserini, C. Casetti, J. C. Cano, C. T. Calafate, and P. Manzoni, "Securing warning message dissemination in vanets using cooperative neighbor position verification," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 6, pp. 2538–2550, 2015.

[23] T. Soler and L. D. Hothem, "Coordinate systems used in geodesy: Basic definitions and concepts," *Journal of Surveying Engineering*, vol. 114, no. 2, pp. 84–97, 1988.

[24] C. M. Huang, T.-H. Lin, and K.-C. Tseng, "Bandwidth aggregation over vanet using the geographic member-centric routing protocol (gmr)," in *International Conference on ITS Telecommunications*, pp. 737–742, November 2012.

[25] Z. Liu, Y. Xiang, and W. Sun, "Geosvr: A geographic stateless vanet routing," in *IEEE Conference Anthology*, pp. 1–7, Jan 2013.

[26] K. Fujita and K. Sado, "Instantaneous speed detection with parameter identification for ac servo systems," in *Conference Record of the IEEE Industry Applications Society Annual Meeting*, pp. 632–638 vol.1, October.

[27] Y. Khaliq, A. Qureshi, G. Abbas, and F. Zeeshan, "Calculation of cpu performance, power and cost using hadoop," in *Sixth International Conference on Innovative Computing Technology (INTECH)*, pp. 122–127, August 2016.

[28] R. Low, "More mips per slot (atca or not) [power management]," *Communications Engineer*, vol. 3, no. 1, pp. 40–43.

[29] L. Gui-sen, R. Chen, D. bin, and Z. Shun-zhi, "A remediable broadcasting protocol for vehicular ad hoc network," in *IEEE International Conference on Computer and Communications (ICCC)*, pp. 2183–2187, October 2016.

[30] I. Rubin, Y.-Y. Lin, A. Baiocchi, F. Cuomo, and P. Salvo, "Rapid dissemination of public safety message flows in vehicular networks," *Journal of communications*, vol. 9, no. 8, pp. 616–626, 2014.

[31] B. Wilmots, E. Hermans, T. Brijs, and G. Wets, "Speed control with and without advanced warning sign on the field: An analysis of the effect on driving speed," *Safety Science*, vol. 85, pp. 23 – 32, 2016.

[32] H. Myler, *Fundamentals of engineering programming with C and Fortran.* Cambridge New York: Cambridge University Press, 1998.

[33] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms.* McGraw-Hill Higher Education, 2nd ed., 2001.

[34] Y. Hwang, K. Yang, and K. Cheun, "Low-latency low-complexity heap-based extended min-sum algorithms for non-binary low-density parity-check codes," *IET Communications*, vol. 9, no. 9, pp. 1191–1198, 2015.

[35] L. Surhone, M. Timpledon, and S. Marseken, *Shannon-Hartley Theorem.* VDM Publishing, 2010.

[36] 3GPP TR 36.885 (Release 14), *Technical Specification Group Radio Access Network; Study on LTE-based V2X Services,*, June 2016.

[37] P. Heino, J. Meinilä, P. Kyösti, L. Hentila, T. Jämsä, E. Suikkanen, E. Kunnari, and M. Narandzic, "CP5-026 WINNER+ D5.3 v1.0 WINNER+ Final Channel Models," January 2010.

[38] V. Cristea, V. Gradinescu, C. Gorgorin, R. Diaconescu, and L. Iftode, "Simulation of vanet applications," *Automotive Informatics and Communicative Systems*, pp. 258–276, 2009.

[39] G. Sallam and A. Mahmoud, "Performance Evaluation of OLSR and AODV in VANET Cloud Computing Using Fading Model with SUMO and NS3," in *International Conference on Cloud Computing (ICCC)*, pp. 1–5, April 2015.

[40] S. Ibrahim, K. Choo, Z. Yan, and W. Pedrycz, "Algorithms and Architectures for Parallel Processing: 17th International Conference, ICA3PP, Helsinki, Finland, August 21-23, 2017, Proceedings," 2017.

[41] J. Song, Y. Wu, Z. Xu, and X. Lin, "Research on car-following model based on sumo," in *The 7th IEEE/International Conference on Advanced Infocomm Technology*, pp. 47–55, November 2014.

[42] Y. Su, H. Cai, and J. Shi, "An improved realistic mobility model and mechanism for vanet based on sumo and ns3 collaborative simulations," in *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 900–905, December 2014.