Dissertation

AN IMPLEMENTATION OF A ROUTE RESERVATION ARCHITECTURE

Christos Makridis

UNIVERSITY OF CYPRUS

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

December 2020

ABSTRACT

Route Reservation Architecture seems to be a reliable and durable traffic congestion mitigation mechanism that can significantly reduce drivers' travel-times. In this framework, a Route Reservation controller is responsible for coordinating drivers' departure times and routes to follow so as to arrive at their destination at the earliest time. Despite its great efficiency, its real-life implementation is a significant challenge considering the size and the complexity of urban networks. This work investigates and develops all the modules required to realize the Route Reservation Architecture as a real-life traffic and demand management mechanism. In doing so, this work proposes all the needed communication protocols and algorithmic methods that ensure its fast and reliable application and addresses any architecture's scalability issues. The developed modules are evaluated over a large-scale urban area where a microscopic simulation serves as a realistic transportation network.

Christos Makridis – University of Cyprus, 2020

AN IMPLEMENTATION OF A ROUTE RESERVATION ARCHITECTURE

Christos Makridis

A Thesis
Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science
at the
University of Cyprus

Recommended for Acceptance
by the Department of Electrical and Computer Engineering
December, 2020

APPROVAL PAGE

Master of Science Thesis

AN IMPLEMENTATION OF A ROUTE RESERVATION ARCHITECTURE

Presented by

Christos Makridis

Research Supervisor

C.G. Panayiotou

Committee Member

S. Timotheou

Committee Member

C. Menelaou

University of Cyprus

December, 2020

ACKNOWLEDGEMENTS

For the completion of this thesis, I would like to express my gratitude to both my supervisor Dr. Christos G. Panayiotou and expert Dr. Charalambos Menelaou. I appreciate the time and effort they have put on guiding me through this thesis, as well as for teaching me the necessary knowledge for achieving its fulfillment.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Traffic congestion is a significant problem for modern societies, with transport operators and city dwellers thoroughly looking for solutions. The impact of congestion is noticeably large in economic, psychological and environmental terms. In more detail, idling with the engine running wastes fuel, resulting in not only burdening the drivers' pockets but also increasing the amount of harmful emissions in the environment. What is more, engaging in congestion wastes some serious amount of time that could have been used more productively, hence making drivers anxious.

The main cause for congestion to occur in a road network is to have multiple vehicles demanding simultaneously the same road segment. At the same time, some other road segments of the road network might remain unutilized, although they could have been accommodating part of the traffic and thus disentangling congestion [4]. In this direction, a better distribution of the demand on both the space and time domains could ease the situation, without altering the existing infrastructure of the road network.

This principle is followed by the Route Reservation Architecture (RRA), as described in [22], which is an alternative routing method that can avoid congestion in the first place. In this framework, when drivers want to utilize road infrastructure, they send a reservation request to the Route Reservation Controller (RRC), including their origin-destination pair and their desired departure time. In turn, the RRC is responsible for identifying each driver's appropriate route and departure time in such a way as to ensure that none of the road segments will be utilized at times that their critical capacities are expected to be reached. Afterwards, the RRC responds to the driver with a route to follow and a departure time, while making the appropriate reservations at the time frame that the driver is expected to traverse each road segment. This ensures that none of the next requests will utilize the same part of the infrastructure at the same time.

Among other mechanisms, the RRA seems to be a promising solution to the traffic congestion problem. It guides vehicles through uncongested road segments hence benefiting individual drivers, as well as it denies traversal in road segments that are nearly at capacity, hence ensuring that no congestion occurs and thus benefiting all other drivers in the road network. Nevertheless, the RRA requires a reliable implementation that can serve real-time route requests, as well as a complete driver compliance. The former will be investigated in this thesis, describing the implementation, while the second regards future work.

1.1 Objective

Despite the great efficiency of the RRA, its real-life implementation has not been investigated yet. Therefore, this work aims to examine all the implementation aspects of the RRA and overcome performance buries that may arise. It also aims to ensure a safe, reliable and scalable operation as well as to investigate and develop the necessary communication protocol between the drivers and the RRC. Finally, it aims for an evaluation of the RRA, using a microscopic simulation to represent reality.

1.2 Contribution

The contribution of this thesis is to investigate and develop all the required modules to realize the RRA as a real-life traffic and demand management mechanism. Therefore, we explicitly designed and implemented all the required modules (e.g., data structures, optimization algorithms, and database designs) to transform the RRA into a real-life traffic monitoring and control tool.

In more detail, a user-friendly User Interface was implemented as an Android application that is capable of sending route reservation requests to the RRC and visualizing the results. Moreover, the RRC was implemented in JAVA, being able to serve real-time requests by conducting the Reservation Database, following the RRA principle of calculating a path and a departure time as to avoid congestion.

Nevertheless, the interaction with the Database as well as the algorithm for finding the solution can escalate to high computation times for large urban areas and hence should be optimized in order to be applicable for a real-time implementation. One of the optimizations made for the RRC regarded the technology used for the Database, which was chosen to be appropriate for time-series data, as well as allowing batch storing and retrieving of data, thus resulting in fast interactions between the RRC and the Database. Another optimization regarded the algorithm responsible for calculating the route path and departure time as to avoid congestion. Specifically, by using different than the ordinary data structures, the computational complexity of the algorithm was reduced.

Finally, a simulator was used in order to represent reality and hence allow the conduction of experiments. This allowed the comparison between the RRA and a no control approach, as to get information about the effect of the RRA in the road network's utilization.

1.3 Outline

The rest of this thesis is organized as follows. In Chapter 2 some related work is presented. In Chapter 3 a detail description of the RRA is given while Chapter 4 describes this work's implementation. Chapter 5 provides a performance evaluation considering a realistic as possible environment, where Chapter 6 concludes this work and elaborates future research avenues.

Chapter 2

Related Work

2.1 Traffic mitigation mechanisms

Recent technological developments enable a plethora of traffic mitigation mechanisms such as route guidance schemes and navigation services (e.g., WAZE and HERE WeGo) [29]. The majority of navigation services aim to minimize the travel time of individual drivers by providing them with accurate real-time network-level state information. However, despite their significant advances, routing methods that benefit the travel time of individuals seem to negatively affect the overall network operation, as they worsen the traffic congestion problem [7].

A recent work in route guidance schemes is included in [2] where even though it can offer drivers with their shortest travel time path and network's state information, it can also predict future states of the network (e.g., average travel time and speed) by utilizing either static [17] or stochastic models [30]. Nevertheless, according to [11], such predictive mechanisms might oversight the negative effects of congested road segments and hence have inaccurate predictions of the future states of the network. Moreover, despite the exceptional technological innovation of the current available routing services (e.g., WAZE), they might significantly advert effects to the network [7]. The reason is that by providing real-time traffic state information to users, all rational drivers will prefer to follow the least congested paths. Thus, the non-congested parts of the network will become overloaded and vice-versa. This behavior aggravates congestion and disregards the utilization of the road infrastructure, even with not so high demand.

Another approach is the dynamic system-optimal traffic assignment method [5]. It formulates a control problem using states, aiming to minimize the total travel time, constraining a rigid amount of traffic for a specific pre-defined period of time. A promising extension of this approach is described in [32], settled for regional routing. It uses aggregation and approximation techniques, by taking into consideration each region's macroscopic dynamics. Similarly, the work in [33] reduces congestion in urban areas, by using a route choice strategy, with limited traffic state measurements.

Other efforts aiming to address the traffic congestion problem include the gating and perimeter control methods [1]. These methods assume that an urban area is partitioned into a set of homogeneous regions[21], within which a boundary flow control mechanism is responsible to restrict the input flows within each region in case that the density of the region is going to exceed its critical density [15] [16]. Nevertheless, their successful implementation requires a vast amount of data to be acquired from the network (e.g., using loop detectors) [1].

Finally, the proposed implementation regards the RRA, which first originated from solutions of the ground holding problem for Air Traffic Management and Control Systems (ATM/ATC), [23]. The ATM/ATC systems utilize an airport's runway without altering its capacity [9], by dividing it into both the space and time domains and using a time-slot reservation mechanism to improve its efficiency, [8], [28]. Similarly, the RRA uses a time-slot reservation model in order to avoid congestion, by dividing the road network into segments of both the space and time domains. In more detail, the RRA uses a controller responsible to keep track of the estimated number of vehicles that are expected to be traversing each road segment at each time-slot. In order for the controller to be able to estimate these values, each driver that wants to use the road network needs to first make a reservation by informing the controller about their origin, destination and desired departure time. The controller then responds with a route and a departure time, depending on the current reservation state, in such a way as to ensure that the critical density of all road segments will not be exceeded. Regarding the RRA, the work in [24] describes the Earliest-Destination Arrival Time (EDAT) problem, as well as the Traffic Load Balancing Problem, while proposing solutions for both of them. Although, their complexity is relatively high, [22] proposed a low-complexity solution algorithm for the EDAT problem, which is the one implemented in this thesis.

2.2 Existing technologies

A dynamic route guidance demonstration program took place in Illinois [3], where an in-vehicle navigation and route guidance system was implemented. It used dynamically updated travel time information on a specific road system, monitored and assessed by loop detectors, on-board sensors, video-based license plate recognition units and a radar system to track vehicles, extracting average travel times over the road network.

A more modern solution which also uses dynamic route guidance, is using maximum flow theory to balance the traffic load of the road network and hence avoid congestion [31]. In order to be promising for large areas, it uses a Ford-Fulkerson algorithm for finding routes that maximize the network's flow [12], implementing MapReduce primitives[6] through a Cloud Computing Platform [19]. In more detail it partitions the road network into smaller maps, distributed in a cloud of computers. This way each map's flow is being maximized without exceeding any road segment's capacity. Nevertheless, this approach has great computational need by the cloud.

Another solution uses multi-modal routing [13], which is a combination of routing guidance using vehicles and other means of transport. In this context, a mobile application was set up, able to propose routes according to the preferences of commuters in terms of private and public transport. The application has access to a real time database, which handles information about traffic congestion, as well public transportation means' schedules. Although such a mechanism can reduce the demands of the road network, it does not focus on solving the general problem of congestion, rather it aims on accommodating a user of the road network and hence the individual benefit.

Chapter 3

Route Reservation Architecture

As mentioned in Chapter 1, the RRA is a routing method that can avoid congestion in the first place, by using a reservation scheme. Drivers utilize the road network by requesting a route for a source and destination pair at a specific departure time. The RRA, according to previous reservations, proposes an appropriate route and departure time in order to ensure that congestion is avoided. Fig. 1 depicts the overview of the RRA. Its main component is the RRC, which consists of two units:

- The Road Network Information Storage unit, which is responsible for holding an overview of a road infrastructure (e.g., geographic boundaries, connections between road segments, speed limits), by parsing a Road Network XML File from a map service (i.e., OpenStreetMap [27]).
- The Route Dispatcher unit, which is responsible for receiving route reservation requests from Drivers. By consulting the Reservation Database (DB), it responds back with proposed routes and departure times, coordinated by a communication protocol.

3.1 Communication Protocol

In the RRA, information is interchanged between the RRC, drivers and the reservation DB through a communication protocol. Fig. 2 shows this communication protocol. First, the RRC requests a connection with the DB that responds with the current reservation status. Once the DB connection is established, the RRC is able to accept requests from drivers. Then, a driver can request a connection, for which the RRC responds with the necessary road network information that regard the driver (i.e., the network's geographic boundaries). For the next step, the driver sends

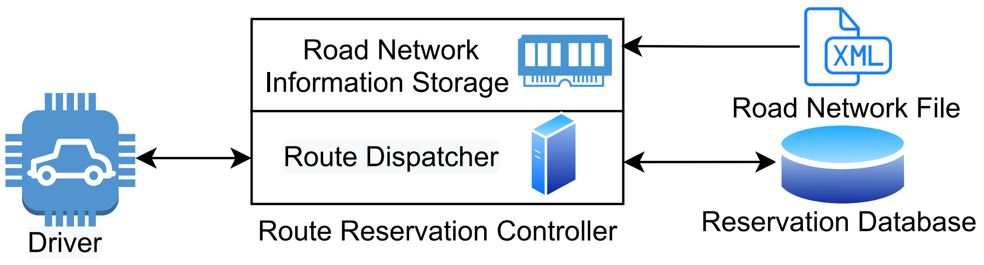


Figure 1: RRA overview

a route reservation request to the RRC, that includes the driver's origin, destination and desired departure time, which comply to the aforementioned network information. Based on the current reservation status, the RRC identifies the appropriate route and departure time of the request. It then updates the DB for the reserved road segments and response to the drivers with the route to follow and the time to depart.

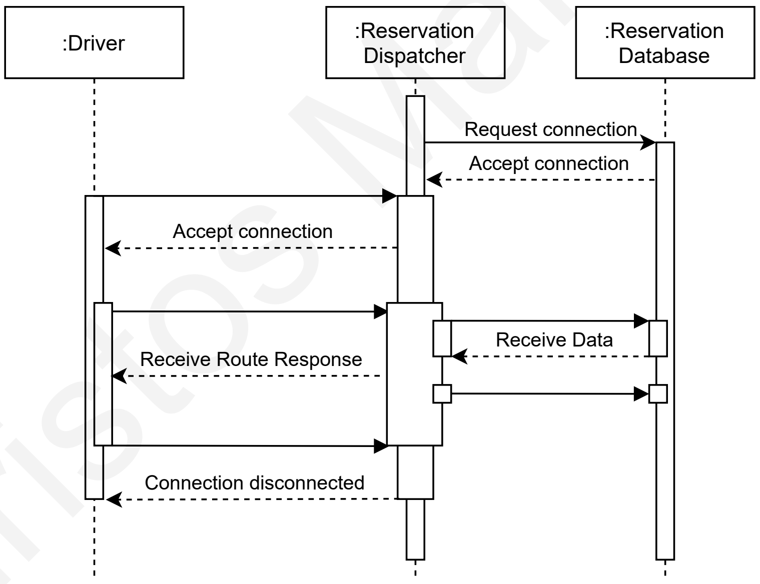


Figure 2: Communication Protocol sequential diagram

The format of the interactions between the RRC and the drivers was chosen to be in JSON, as it is a lightweight format for transporting data between entities. Fig. 3 shows how JSON allows a compact yet complete representation of the data. The route reservation requests are described utilizing tags, consisted of the source, the destination and the desired departure time of the driver. The response is also described in tags, consisting of the departure time and the route. Note that for each tag, JSON allows to include various further details, such as coordinates (i.e., latitude and longitude) for the source and destination, as well as for each road segment of the resulted route.

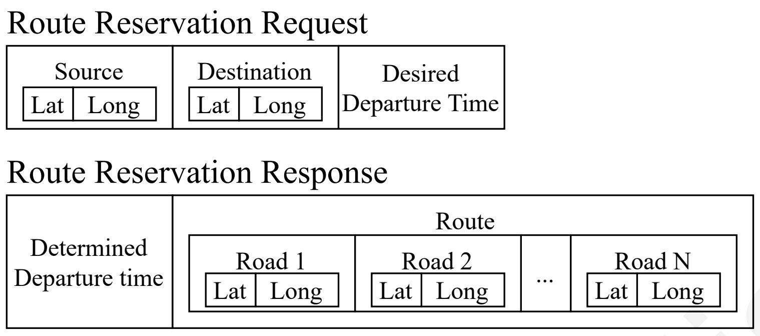
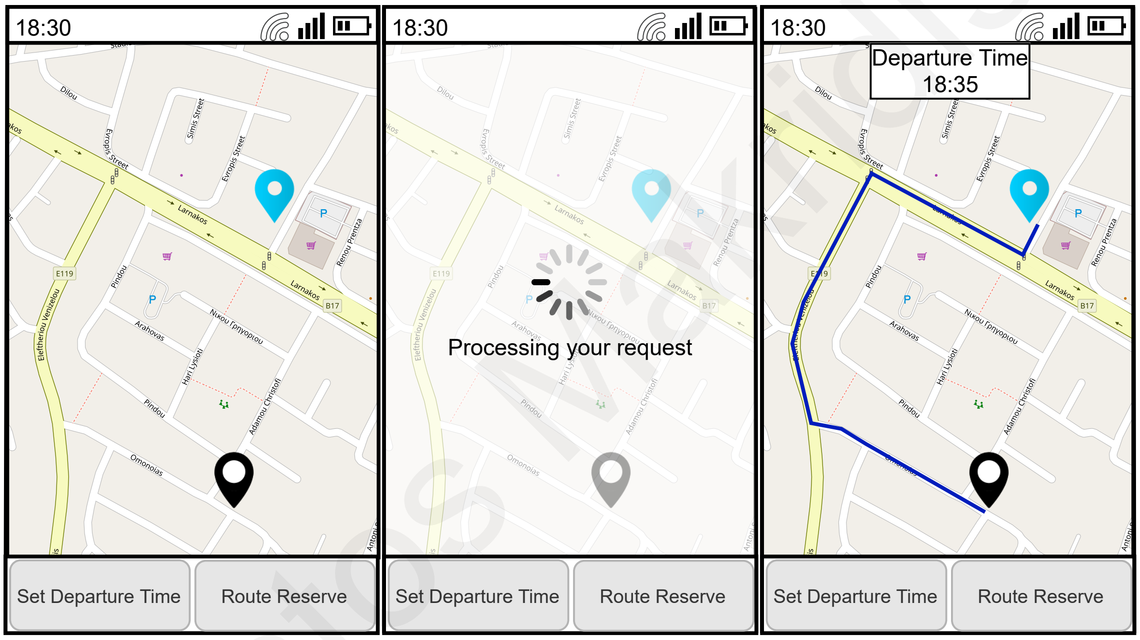


Figure 3: JSON representations

3.2 Driver Interface

A driver can interact with the RRC and vice versa through either smart-phones or onboard vehicle devices. In this work we developed an Android application, which encodes and decodes JSON formatted interactions into user-friendly dialogs, map indicators and buttons. Figs. 4 illustrates the three screens of a route reservation user interface. More specific, Fig. 4a depicts the screen that enables drivers to choose their origin and destination, by utilizing a black and a blue pin-point, respectively. The requested departure time can be set b clicking the “Set Departure time” button and then the request can be sent by clicking the “Route Request” button. While the RRC is processing the request, a corresponding message is indicated to the driver, as viewed on Fig. 4b. When a solution is found, a response is created, containing the departure time and a list of longitudes and latitudes that represent the route’s roads. As seen on Fig. 4c, this information can be interpreted as the route indicated with a blue line and a dialog box that indicates the departure time.



(a) (b) (c)

Figure 4: User Interface Prototype: (a) Reservation Request, (b) Waiting response, (c) Response illustration

Chapter 4

Implementation

The real-life application of the RRA is a challenge due to the sufficiently low response time required for the RRC to respond to a driver, as well as for the vast amount of data needed for making decisions. These requirements raise the need of tolerating that amount of data, hence investigating to utilize a reliable and responsive DB solution. Moreover, optimizing the time complexity of the algorithm is also considered, thus investigating optimization techniques using data structures. Among various programming options, the RRC was implemented in JAVA due to the reliability, portability and maintainability of the language, as well as because it implements an extensive amount of high-level data structures and tools. Such tools include libraries which allow parsing of road network files, exactly as generated from open source maps (i.e., openStreetMap [27]), conversions between geographical representations, the communication between the RRC and the DB and various other necessary fundamental processes for the overall development of the application.

4.1 Database

Databases are a great tool for storing large quantities of data. For the purposes of realizing the RRA, due to the size of urban cities, the amount of information needed is significantly large. Therefore, some solutions were investigated in order to be able to store such quantities of data, while being able to retrieve them fast and reliably in real time. In more detail, for this implementation, the DB stores reservation information for all road segments (i, j), such as their *accumulated number of vehicle reservations* $n_{ij}(t)$ and their *admissibility states* $x_{ij}(t)$ for all time slots t. This information can escalate to high space requirements, especially for big cities.

As the RRC requires access to all this data, a reliable storage system is needed. A candidate open source solution was MySQL [25]. Its framework requires a pre-configuration of the schema (i.e., formal description of relationships between data), which although it allows it to store and retrieve great volumes of data structurally, it draws back in real time usage of these operations. Therefore it was rejected due to the latency caused by the high amount of retrieval requests, as relational DBs are not optimized for.

Another open source solution was investigated, called InfluxDB, a time series DB optimized for fast, high-availability storage and retrieval of time series data, derived from various real-time operations [26]. It works with schema-free, SQL-like queries, requested through an HTTP API, hence allowing multiple independent data retrievals. This makes InfluxDB appropriate for real-time storing and retrieving of information.

In order to take advantage of the capabilities of InfluxDB, relational DB approaches should be avoided. Therefore, tables (i.e., measurements in InfluxDB terminology) should contain information only associated with time (i.e., time series), regarding solely one entity. As a result, for this implementation, every road segment (i, j) implements its own two measurements, one for its *accumulated number of reservations* $n_{ij}(t)$ and one for its *admissibility states* $x_{ij}(t)$ over time t. Moreover, in order to benefit from the multiple independent queries functionality of InfluxDB, a batch technique for the communication between the RRC and the DB was used. It creates multiple independent queries for any operation that requires information for multiple road segments from the DB, grouped together into a non sequential batch request, thus minimizing connection overheads.

The benefit of this approach is illustrated in a toy example on Fig. 5, where three sequential retrieval requests are shown to take up time almost twice as that of three non sequential independent retrieval. This is beneficial, as the connection overhead in the non sequential approach of each request mostly overlaps with other requests' overheads, hence needing less time overall.commented line goes before the last sentence.

4.2 Data structures for optimization

We investigated some alternative data representations to be exploited by the Route Reservation Architecture Algorithm (RRAA), in order to reduce its space and time complexity, ending up with solutions for improving two of its most computationally complex operations.

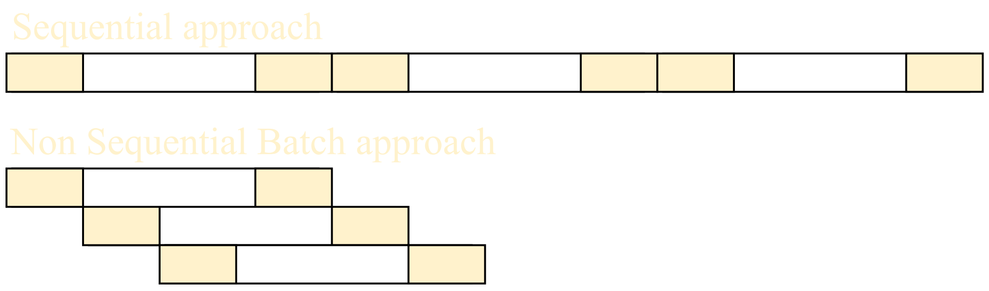


Figure 5: Sequential and Non Sequential Batch for data retrieval requests

4.2.1 First solution

The first solution is the use of min-heap data structure, as it gives an excellent advantage over retrieving the lowest value from a set. It utilizes two operations, referred to as the *extract-min* and *decrease-key* [14]. The former is responsible for finding the minimum value in a set and removing it. The *decrease-key* operation is responsible for changing the value of an element in the set to a lower value. These operations are crucial for applications such as the Dijkstra algorithm [10], which the RRAA is based on. Although both of them could be implemented for a simple set data structure, they would require at least complexity of $\mathcal{O}(m)$, by using a search algorithm, where m is the number of elements in the set. Nevertheless, the retrieval of the minimum value, as well as the decreasing of a key value for a min-heap, both have complexities of $\mathcal{O}(\log m)$. As a result, this implementation has used the JAVA data structure of PriorityQueue, which implements the min-heap.

4.2.2 Second solution

The second solution regards the computation of the *altered cost* $c'_{ij}(t)$ of each road segment (i, j) traversed at time t, expressed mathematically as:

$$c'_{ij}(t) = \begin{cases} \tau_{ij}, \text{ if } x_{ij}(t) = 1 \\ \tau_{ij} + w_{ij}(t), \text{ if } x_{ij}(t) = 0 \end{cases} \tag{1}$$

τ_{ij} is the known travel time of (i, j). $x_{ij}(t)$ is the admissibility state of (i, j) when departing from junction i at time t and is considered admissible when $x_{ij}(t) = 1$, or non admissible when $x_{ij}(t) = 0$. $w_{ij}(t)$ is the respective intermediate delay and is calculated based on how long a vehicle needs to wait for the road segment to become admissible. That is the difference between the time t' of the first admissible state $x_{ij}(t') = 1$ and the vehicle's departure time t from junction

i, with $t <= t'$. The operation of finding t' with a simple search algorithm would have had a time complexity of $\mathcal{O}(N)$, where N is the number of time slots to check for admissibility. In contrast, the following approach allows for a better time complexity, by exploiting the fact that the *admissibility states* are represented by a binary value (i.e., 0 and 1) and thus can be grouped together into intervals of the same values. Note that one value is the compliment of the other, hence it is sufficient to store solely intervals of a certain admissibility (e.g., admissibility states of value 0). Therefore, we chose to only store the non-admissible time intervals into a non overlapping sorted list for every road segment. Mathematically the non-admissibility list is expressed by: $S_{ij} = \{[t^l_{ij1}, t^u_{ij1}], \cdots, [t^l_{ijK_{ij}}, t^u_{ijK_{ij}}]\}$. Variables t^l_{ijk} and t^u_{ijk} denote respectively the lower and upper bounds of the closed range $[t^l_{ijk}, t^u_{ijk}]$ that defines a time interval, in which the road segment (i, j) is non admissible, with $k \in \{1, \cdots, K_{ij}\}$. Variable K_{ij} is the number of non-admissible time intervals of (i, j). Note that $t^l_{ijk} \leq t^u_{ijk} < t^l_{ijk+1}$. In the case of new intervals overlapping with some of the sorted previous ones, a *merging process* is needed, following an algorithm of time complexity $\mathcal{O}(N)$, guaranteeing that the resulted intervals will also be sorted. Moreover, by using this representation, it is possible to use a less computationally complex search algorithm, such as Binary Search. Therefore, finding an interval in S_{ij} needs time complexity $\mathcal{O}(\log N)$. More specifically, for obtaining the first admissible state of road segment (i, j) when traversing at time t, if $x_{ij}(t) = 0$, the Binary Search algorithm will find a matching intervals such that $t' \in [t^l_{ijk}, t^u_{ijk}]$, meaning that (i, j) will become admissible at time $t' = t^u_{ijk} + 1$. In the other case of $x_{ij}(t) = 1$, it will not find such an interval, concluding that (i, j) at time $t' = t$ is admissible.

A simple example can be depicted in Fig. 6, where a set of *admissibility states* $x_{ij}(t)$ is represented by a list of non admissible time ranges $S_{ij}(t)$.

Time t:	0	1	2	3	4	5	6	7	8	9	10	11	12
Admissibility state x_{ij}(t):	1	1	0	1	0	0	0	0	0	1	1	0	0

Non-admissible Time Ranges S_{ij}(t): [2-2], [4-8], [11-12]

Figure 6: Time ranges example

4.3 Route Reservation Architecture Algorithm

The RRAA, as proposed in [22], relies on two loops: the inner and the outer. The inner loop is responsible for finding the least time consuming route from an origin to a destination, departing at a specific time, by taking into consideration the travel time plus the delays of waiting for congested road segments to become uncongested. The outer loop calls the inner loop, first using a departure time equal to the requested departure time $t = t_0$. When the inner loop ends, the outer loop checks whether the solution contains any road segments that include waiting times. If so, the maximum wait time W of the route is added to the accumulated wait at the origin $w_{ij}(t)$, and the inner loop runs again, with departure time equal to $t = t + W$. This process repeats, until the resulting route's road segments do not incorporate any waiting times. In order to run the aforementioned loops, the RRC needs to communicate with the DB, gaining knowledge of the *admissibility states* of the road segments and thus be able to calculate the delay time for congested roads to become uncongested. This procedure breaks down to the steps shown on Fig. 7 and will be explained further in the next paragraphs. Its input is a Route Request which contains the source, the destination and the desired departure time of a driver.

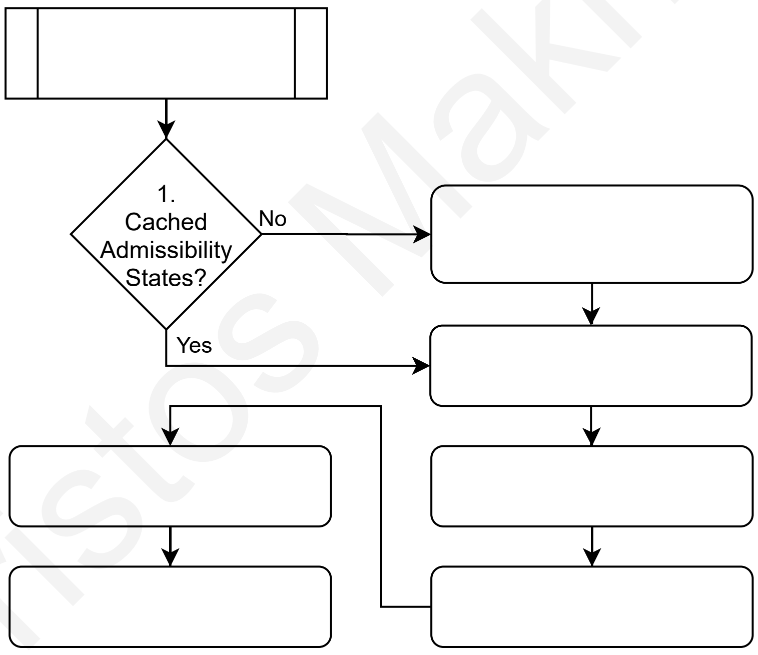


Figure 7: Route Reservation Architecture Algorithm

Step 1: Cached Admissibility States?

The RRAA is caching *admissibility states* $x_{ij}(t)$ for all road segments, $(i, j) \in \mathcal{E}$, for all time slots $t \in \mathcal{R}$, where $\mathcal{E}$ defines the set of road segments of the network and $\mathcal{R}$ a predefined quantized time range for caching. As a result, the RRAA firstly needs to check if $\mathcal{R}$ includes the departure time and a sufficient period ahead of it. If that is not the case, the RRAA moves to step 2. Otherwise it moves on to step 3.

Step 2: Cache Admissibility States from DB

The RRAA requests from the DB the missing *admissibility states* $x_{ij}(t)$, of all road segments (i, j) and for all time slots $t \in \mathcal{R}$, such that R contains the requested departure time and a sufficient period ahead of it.

Step 3: Find Route

The RRAA identifies the appropriate route d_D^* for the request, using the aforementioned outer and inner loops. In more detail, the inner loop uses the min-heap data structure that goes over the two basic operations *extract-min* and *decrease-key*, as described in Section 4.2. At first, it assumes an infinite arrival time a_j for all junctions j, except for the origin junction of the requested route O, which is set to zero. Then, it labels the origin O as i and expands its adjacent road segments (i, j). For each one of them, the *altered cost* $c'_{ij}(t)$ is being calculated and hence the expected arrival time a'_j at junction j. The *decrease-key* operation checks if the newly calculated expected arrival a'_j is lower than the previously known expected arrival a_j. If so, the old value is replaced with the new one $a_j = a'_j$. The inner loop repeats, but this time, instead of the origin junction O, it uses the *extract-min* operation to find the junction i with the lowest calculated expected arrival a_i^* that has not been expanded yet. The inner loop ends when the *extract-min* operation finds the destination junction D and returns the route d_D. The outer loop repeats this process M times, by increasing the initial departure time t_0 in each iteration, until route d_D^* is found, which has no intermediate delays.

Step 4: Request Route's data from DB

Once the RRAA finds the best route d_D^*, it needs to reserve the elaborated segments at the time frame that the driver is expected to traverse them. To do so, it requests from the DB the *accumulated number of reservations* $n_{ij}(t)$ for all the route's road segments $(i, j) \in d_D^*$, at all time slots t that the driver is expected to be traveling. Note that this is not needed for the *admissibility state*, as it is already cached.

Step 5: Reserve Route's Road Segments

This step reserves the road segments of the solution, by increasing by one the value of the *accumulated number of reservations* $n_{ij}(t)$, for each time slot t that the driver is expected to occupy for all the route's road segments $(i, j) \in d_D^*$.

Step 6: Update Route's Admissibility States

This step calculates the new *admissibility states* $x_{ij}(t)$ for the elaborated segments at the time frame that the driver is expected to traverse them. In more detail, a road segment at a given time is considered admissible only if the addition of another vehicle keeps the density of the road below its critical density. Note that when a road segment at a specific time becomes non-admissible, it must also become non-admissible for τ_{ij} previous time-slots as well, regardless of its *accumulated number of reservations* $n_{ij}(t)$. In more detail, with n_{ij}^C denoting the *critical accumulated number of reservations* of road segment (i, j), if $n_{ij}(t) + 1 > n_{ij}^C$ holds, then the range $[1 + t - \tau_{ij}, t]$, with $t - \tau_{ij} \geq -1$ is considered non admissible. This will guarantee that a vehicle that enters a road segment will surely exit it without causing congestion in the meantime. All newly calculated *admissibility states* need to be *merged* with the already stored *admissibility states*.

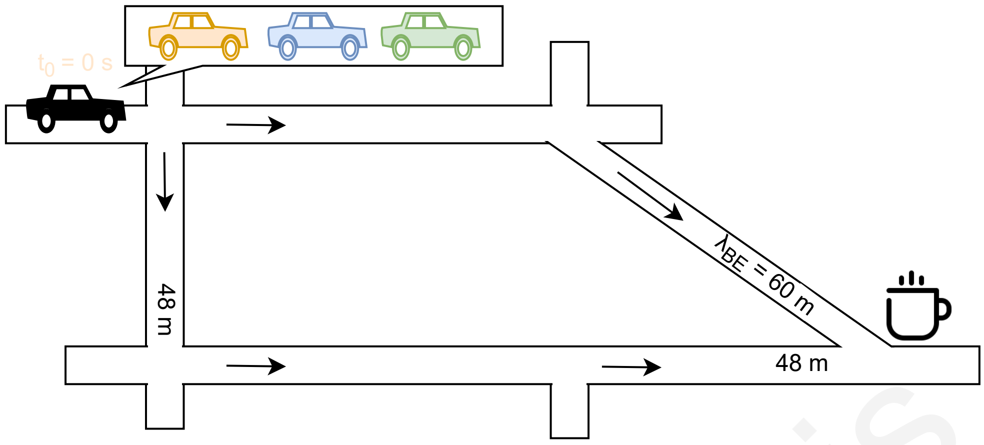
Step 7: Upload Reservations and Admissibility States to DB

This last step uploads the updated *accumulated number of reservations* $n_{ij}(t)$ and *admissibility states* $x_{ij}(t)$ to the DB.

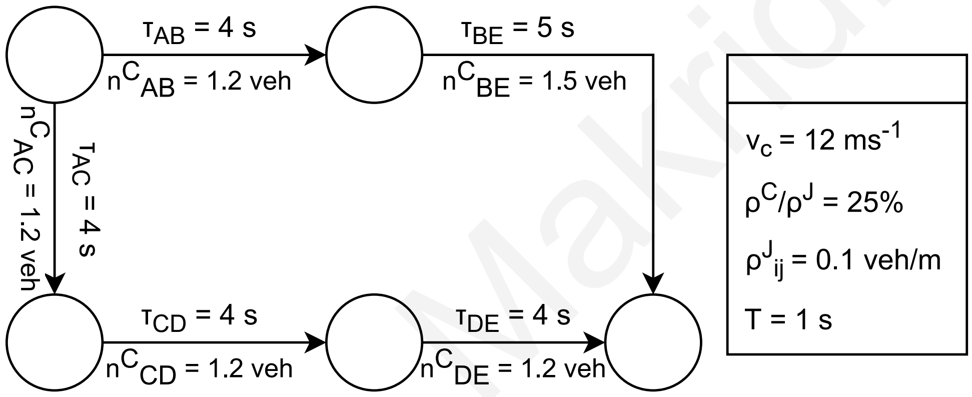
4.4 Toy Example

Fig. 8a illustrates a toy example road network, where five junctions connect five one-way road segments. Three vehicles are shown to each make a reservation request for a route starting from junction A and destining to junction E, departing at time $t_0 = 0\,\mathrm{s}$. The RRC is first configured for this particular road network by processing the network's XML file representation and storing its information in the RRC's storage unit, as depicted in Fig. 8b. In more detail, it includes the formulation of the graph G that represents the road network:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$
$$\{A, B, C, D, E\} \in \mathcal{V}$$
$$\{(A, B), (B, E), (A, C), (C, D), (D, E)\} \in \mathcal{E}$$



(a)



(b)

Figure 8: Example road network: (a) Illustration, (b) Formulation

with $\mathcal{E}$ representing the road segments and $\mathcal{V}$ the junctions of the road network. For each road segment (i, j), the XML file defines their length:

$$\lambda_{AB} = \lambda_{AC} = \lambda_{CD} = \lambda_{DE} = 48\,\mathrm{m}$$
$$\lambda_{BE} = 60\,\mathrm{m}$$

and their number of lanes:

$$N_{AB} = N_{AC} = N_{CD} = N_{DE} = N_{BE} = 1$$

Moreover, it states the time-slots duration $T = 1\,\mathrm{s}$, the speed at capacity $v_c = 12\,\mathrm{m\,s^{-1}}$, the ratio of the networks critical density over its jam density $(\rho^C/\rho^J) = 0.25$ and the jam density $\rho^J_{ij} = 0.1\,\mathrm{veh/m}$ for each road segment (i, j). Finally, the RRC calculates the number of time-slots τ_{ij}

required to traverse each road segment (i, j) and its critical density ρ_{ij}^C, using the equations:

$$\tau_{ij} = \lfloor \lambda_{ij}/v_c/T \rceil \tag{2}$$

$$\rho_{ij}^C = (\rho^C/\rho^J)\rho_{ij}^J \tag{3}$$

Note that $\lfloor z \rceil$ defines the closest integer to z, thus:

$$\begin{aligned} &\tau_{AB} = \tau_{AC} = \tau_{CD} = \tau_{DE} = 4\,\text{slots} \\ &\tau_{BE} = 5\,\text{slots} \\ &\rho_{ij}^C = 0.025\,\text{veh/m}, \forall(i,j) \in \mathcal{E} \end{aligned}$$

For simplicity, another variable is introduced, which denotes the *critical number of accumulated vehicles* $n_{ij}^C = \rho_{ij}^C * \lambda_{ij} * N_{ij}$. By the inequation of the instantaneous density being less or equal than the critical density we get:

$$\rho_{ij}(t) \leq \rho_{ij}^C \tag{4}$$

$$\rho_{ij}(t) * \lambda_{ij} * N_{ij} \leq \rho_{ij}^C * \lambda_{ij} * N_{ij} \tag{5}$$

$$n_{ij}(t) \leq n_{ij}^C \tag{6}$$

Therefore, the critical number of accumulated vehicles for each road segment is:

$$\begin{aligned} &n_{AB}^C = n_{AC}^C = n_{CD}^C = n_{DE}^C = 1.2\,\text{veh} \\ &n_{BE}^C = 1.5\,\text{veh/m} \end{aligned}$$

In order to serve a route reservation request, the *accumulated numbers of reservations* $n_{ij}(t)$ for each road segment $(i, j) \in \mathcal{E}$ and each time-slot t are needed. These values are solely stored in the DB and are only requested by the RRC when a route is found, to be updated. Fig. 9 depicts both the *accumulated number of reservations* $n_{ij}(t)$ and the *non-admissibility lists* S_{ij}, with empty cells representing zero reservations. The RRC checks the non-admissibility list S_{ij} of each road segment, to determine the amount of wait time $w_{ij}(t)$ needed for (i, j) at time t to become admissible. This list is initially empty for each road segment and is filled depending on upcoming reservations. It is stored on the DB and cached in the RRC's storage unit.

All three reservations in this example regard a route departing from junction A at time $t_0 = 0$ and arriving at junction E. For each one of them, the RRC runs the outer loop of the RRAA, which in turn runs its inner loop for departure time $t = t_0 = 0$.

4.4.1 First reservation

Only for the first reservation, the RRC requests to cache the non-admissibility list S_{ij} of each road segment $(i, j) \in \mathcal{E}$ from the DB for the whole day. The inner loop takes the road segments that begin from the origin of the route A and calculates their altered costs $c'_{AB}(t)$ and $c'_{AC}(t)$. It repeats the process for road segments (B, E), (C, D), (D, E), choosing them in an increasing order of their leading road segment's arrival time.

Inner loop

Beginning on junction A, the RRC calculates the altered cost of the two road segments (A, B) and (A, C) with departure time at $t = t_0 = 0$. For road segment (A, B), due to the fact that the non-admissibility list S_{AB} is empty, any time slot is admissible and hence:

$$w_{AB}(0) = 0$$
$$c'_{AB}(0) = \tau_{AB} + w_{AB}(0) = 4$$

Similarly for road segment (A, C), we get:

$$w_{AC}(0) = 0$$
$$c'_{AC}(0) = \tau_{AC} + w_{AC}(0) = 4$$

The algorithm then defines a new variable for the arrival time of each road segment $a_{ij} = c'_{ij}(t)+t$. Therefore:

$$a_{AB} = c'_{AB}(t) + t = c'_{AB}(0) + 0 = 4$$
$$a_{AC} = c'_{AC}(t) + t = c'_{AC}(0) + 0 = 4$$

After that, the algorithm chooses the road segment with the lowest arrival time. Regardless that the values are equal in this case, it chooses the (A, B) road segment. Then, it repeats the previous process, but now beginning on junction B. It checks the only road segment starting from that junction, which is (B, E), departing at time $t = a_{AB} = 4$. Similarly to the previous cases, the non-admissibility list is empty, hence:

$$w_{BE}(4) = 0$$
$$c'_{BE}(4) = \tau_{BE} + w_{BE}(4) = 5$$
$$a_{BE} = c'_{BE}(4) + 4 = 9$$

The process repeats. This time the lowest arrival time is for road segment (A, C). Therefore, departing from junction C at time $t = a_{AC} = 4$, the RRC checks the altered cost of road segment (C, D). The non-admissibility list is empty, hence:

$$\begin{aligned} &w_{CD}(4) = 0 \\ &c'_{CD}(4) = \tau_{CD} + w_{CD}(4) = 4 \\ &a_{CD} = c'_{CD}(4) + 4 = 8 \end{aligned}$$

The process repeats for another time. Now, the lowest arrival time is for road segment (D, E). Therefore, departing from junction D at time $t = a_{CD} = 8$, the RRC checks the altered cost of road segment (D, E). The non-admissibility list is empty, hence:

$$\begin{aligned} &w_{DE}(8) = 0 \\ &c'_{DE}(8) = \tau_{DE} + w_{DE}(8) = 4 \\ &a_{DE} = c'_{DE}(8) + 8 = 12 \end{aligned}$$

The inner loop ends when the chosen road segment leads to the destination E. For this case it is (B, E) with arrival time $a_{BE} = 9$, hence the route: $\{(A, B), (B, E)\}$. Now, the outer loop checks if there are any $w_{ij}(t) > 0$ in the solution. As there are none, the outer loop also ends. The accumulated numbers of reservations $n_{ij}(t)$ are then being requested from the DB, as shown in Fig. 9a. It then increases their initial values by one and updates their non-admissibility lists. As a result, for the road segment (A, B), slots in the range $[0, 3]$ are incremented and set as non admissible, while for road segment (B, E), slots in the range $[4, 8]$ are incremented and are considered non admissible for the range $[0, 8]$, resulting in Fig. 9b.

4.4.2 Second reservation

The inner loop repeats the same process as the first reservation. The fastest route of the previous reservation becomes non-admissible, thus having a later arrival time due to incorporating waiting time.

Inner loop

Beginning on junction A, the RRC calculates the altered cost of the two road segments (A, B) and (A, C) with departure time at $t = t_0 = 0$. Firstly, for road segment (A, B), as the non-admissibility list $S_{AB} = \{[t^l_{AB1}, t^u_{AB1}] = [0, 3]$ contains $t = 0$, it calculates:

$$w_{AB}(0) = t^u_{AB1} + 1 - t = 4$$
$$c'_{AB}(0) = \tau_{AB} + w_{AB}(0) = 8$$
$$a_{AB} = c'_{AB}() + 0 = 8$$

On the other hand, for road segment (A, C), the non-admissibility list is empty, hence:

$$w_{AC}(0) = 0$$
$$c'_{AC}(0) = \tau_{AC} + w_{AC}(0) = 4$$
$$a_{AC} = c'_{AC}(0) + 0 = 4$$

The process repeats by taking the road segment with the lowest arrival time. This is road segment (A, C). Therefore, departing from junction C at time $t = a_{AC} = 4$, the RRC checks the altered cost of road segment (C, D). The non-admissibility list is empty, hence:

$$w_{CD}(4) = 0$$
$$c'_{CD}(4) = \tau_{CD} + w_{CD}(4) = 4$$
$$a_{CD} = c'_{CD}(4) + 4 = 8$$

The process repeats for another time. Now, the lowest arrival time is for road segment (D, E). Therefore, departing from junction D at time $t = a_{CD} = 8$, the RRC checks the altered cost of road segment (D, E). The non-admissibility list is empty, hence:

$$w_{DE}(8) = 0$$
$$c'_{DE}(8) = \tau_{DE} + w_{DE}(8) = 4$$
$$a_{DE} = c'_{DE}(8) + 8 = 12$$

The lowest arrival time is now for road segment (A, B). Therefore, departing from junction B at time $t = a_{AB} = 8$, the RRC checks the altered cost of road segment (B, E). The non-admissibility list $S_{BE} = \{[t^l_{BE1}, t^u_{BE1}] = [0, 8]$ contains $t = 8$, hence:

$$w_{BE}(8) = t^u_{BE1} + 1 - t = 1$$
$$c'_{BE}(8) = \tau_{BE} + w_{BE}(8) = 6$$
$$a_{BE} = c'_{BE}(8) + 8 = 14$$

As a result, a different route is concluded: $\{(A, C), (C, D), (D, E)\}$. As there are no $w_{ij}(t) > 0$ in this solution, the outer loop also ends and the appropriate accumulated numbers of reservations $n_{ij}(t)$ are being requested from the DB as shown in Fig. 9b. It then increases their values by one and updates their non-admissibility lists, resulting in Fig. 9c.

4.4.3 Third reservation

The inner loop repeats the same process as the previous reservations.

Inner loop 1

Beginning on junction A, the RRC calculates the altered cost of the two road segments (A, B) and (A, C) with departure time at $t = t_0 = 0$. For road segment (A, B), as the non-admissibility list $S_{AB} = \{[t^l_{AB1}, t^u_{AB1}] = [0, 3]$ contains $t = 0$, it calculates:

$$\begin{aligned}
w_{AB}(0) &= t^u_{AB1} + 1 - t = 4 \\
c'_{AB}(0) &= \tau_{AB} + w_{AB}(0) = 8 \\
a_{AB} &= c'_{AB}(0) + 0 = 8
\end{aligned}$$

Similarly for road segment (A, C), with the non-admissibility list $S_{AC} = \{[t^l_{AC1}, t^u_{AC1}] = [0, 3]$, which contains $t = 0$, we get:

$$\begin{aligned}
w_{AC}(0) &= t^u_{AC1} + 1 - t = 4 \\
c'_{AC}(0) &= \tau_{AC} + w_{AC}(0) = 8 \\
a_{AC} &= c'_{AC}(0) + 0 = 8
\end{aligned}$$

The process repeats with the road segment with the lowest arrival time. Regardless that the values are equal, it takes (A, B). Therefore, departing from junction B at time $t = a_{AB} = 8$, the RRC checks the altered cost of road segment (B, E). The non-admissibility list $S_{BE} = \{[t^l_{BE1}, t^u_{BE1}] = [0, 8]$ contains $t = 8$, hence:

$$\begin{aligned}
w_{BE}(8) &= t^u_{BE1} + 1 - t = 1 \\
c'_{BE}(8) &= \tau_{BE} + w_{BE}(8) = 6 \\
a_{BE} &= c'_{BE}(8) + 8 = 14
\end{aligned}$$

The road segment with the lowest arrival time now is (A, C). Therefore, departing from junction C at time $t = a_{AC} = 8$, the RRC checks the altered cost of road segment (C, D). The non-admissibility list $S_{CD} = \{[t^l_{CD1}, t^u_{CD1}] = [1, 7]$, which does not contain $t = 8$, hence:

$$\begin{aligned} &w_{CD}(8) = 0 \\ &c'_{CD}(8) = \tau_{CD} + w_{CD}(8) = 4 \\ &a_{CD} = c'_{CD}(8) + 8 = 12 \end{aligned}$$

Next, the lowest arrival time is for road segment (C, D). Therefore, departing from junction D at time $t = a_{CD} = 12$, the RRC checks the altered cost of road segment (D, E). The non-admissibility list $S_{DE} = \{[t^l_{DE1}, t^u_{DE1}] = [5, 11]$, which does not contain $t = 12$, hence:

$$\begin{aligned} &w_{DE}(12) = 0 \\ &c'_{DE}(12) = \tau_{DE} + w_{DE}(12) = 4 \\ &a_{DE} = c'_{DE}(12) + 12 = 16 \end{aligned}$$

The inner loop results in the lowest arrival time to be for road segment (B, E) with $a_{BE} = 14$, hence the route $\{(A, B), (B, E)\}$. Now, the outer loop checks if there are any $w_{ij}(t) > 0$ in the solution, which are: $w_{AB}(0) = 4$ and $w_{BE}(8) = 1$.

The outer loop calls the inner loop again, but this time for departure time $t' = t + max(w_{AB}(0), w_{BE}(8)) = 4$.

Inner loop 2

Beginning on junction A, the RRC calculates the altered cost of the two road segments (A, B) and (A, C) with departure time at $t = t' = 4$. For road segment (A, B), as the non-admissibility list $S_{AB} = \{[t^l_{AB1}, t^u_{AB1}] = [0, 3]$ does not contain $t = 4$, it calculates:

$$\begin{aligned} &w_{AB}(4) = 0 \\ &c'_{AB}(4) = \tau_{AB} + w_{AB}(4) = 4 \\ &a_{AB} = c'_{AB}(4) + 4 = 8 \end{aligned}$$

Similarly for road segment (A, C), with the non-admissibility list $S_{AC} = \{[t^l_{AC1}, t^u_{AC1}] = [0, 3]$, which does not contain $t = 4$, we get:

$$\begin{aligned} &w_{AC}(4) = 0 \\ &c'_{AC}(4) = \tau_{AC} + w_{AC}(4) = 4 \\ &a_{AC} = c'_{AC}(4) + 4 = 8 \end{aligned}$$

The rest of the steps of this inner loop have the same results as the previous run of the inner loop, resulting in having the road segment with the lowest arrival time at the end to be (B, E) with $a_{BE} = 14$, hence the route $\{(A, B), (B, E)\}$. Nevertheless, the outer loop finds a waiting time, which is $w_{BE}(8) = 1$.

The outer loop calls the inner loop again, but this time for departure time $t'' = t' + w_{BE}(8) = 5$.

Inner loop 3

Beginning on junction A, the RRC calculates the altered cost of the two road segments (A, B) and (A, C) with departure time at $t = t_0 = 5$. For road segment (A, B), as the non-admissibility list $S_{AB} = \{[t^l_{AB1}, t^u_{AB1}] = [0, 3]$ does not contain $t = 5$, it calculates:

$$\begin{aligned} &w_{AB}(5) = 0 \\ &c'_{AB}(5) = \tau_{AB} + w_{AB}(5) = 4 \\ &a_{AB} = c'_{AB}(5) + 5 = 9 \end{aligned}$$

Similarly for road segment (A, C), with the non-admissibility list $S_{AC} = \{[t^l_{AC1}, t^u_{AC1}] = [0, 3]$, which does not contain $t = 4$, we get:

$$\begin{aligned} &w_{AC}(5) = 0 \\ &c'_{AC}(5) = \tau_{AC} + w_{AC}(5) = 4 \\ &a_{AC} = c'_{AC}(5) + 5 = 9 \end{aligned}$$

The process repeats with the road segment with the lowest arrival time which is (A, B). Therefore, departing from junction B at time $t = a_{AB} = 9$, the RRC checks the altered cost of road segment (B, E). The non-admissibility list $S_{BE} = \{[t^l_{BE1}, t^u_{BE1}] = [0, 8]$ does not contain $t = 9$, hence:

$$\begin{aligned} &w_{BE}(9) = 0 \\ &c'_{BE}(9) = \tau_{BE} + w_{BE}(9) = 5 \\ &a_{BE} = c'_{BE}(9) + 9 = 14 \end{aligned}$$

The road segment with the lowest arrival time now is (A, C). Therefore, departing from junction C at time $t = a_{AC} = 9$, the RRC checks the altered cost of road segment (C, D). The non-admissibility list $S_{CD} = \{[t^l_{CD1}, t^u_{CD1}] = [1, 7]$ does not contain $t = 9$, hence:

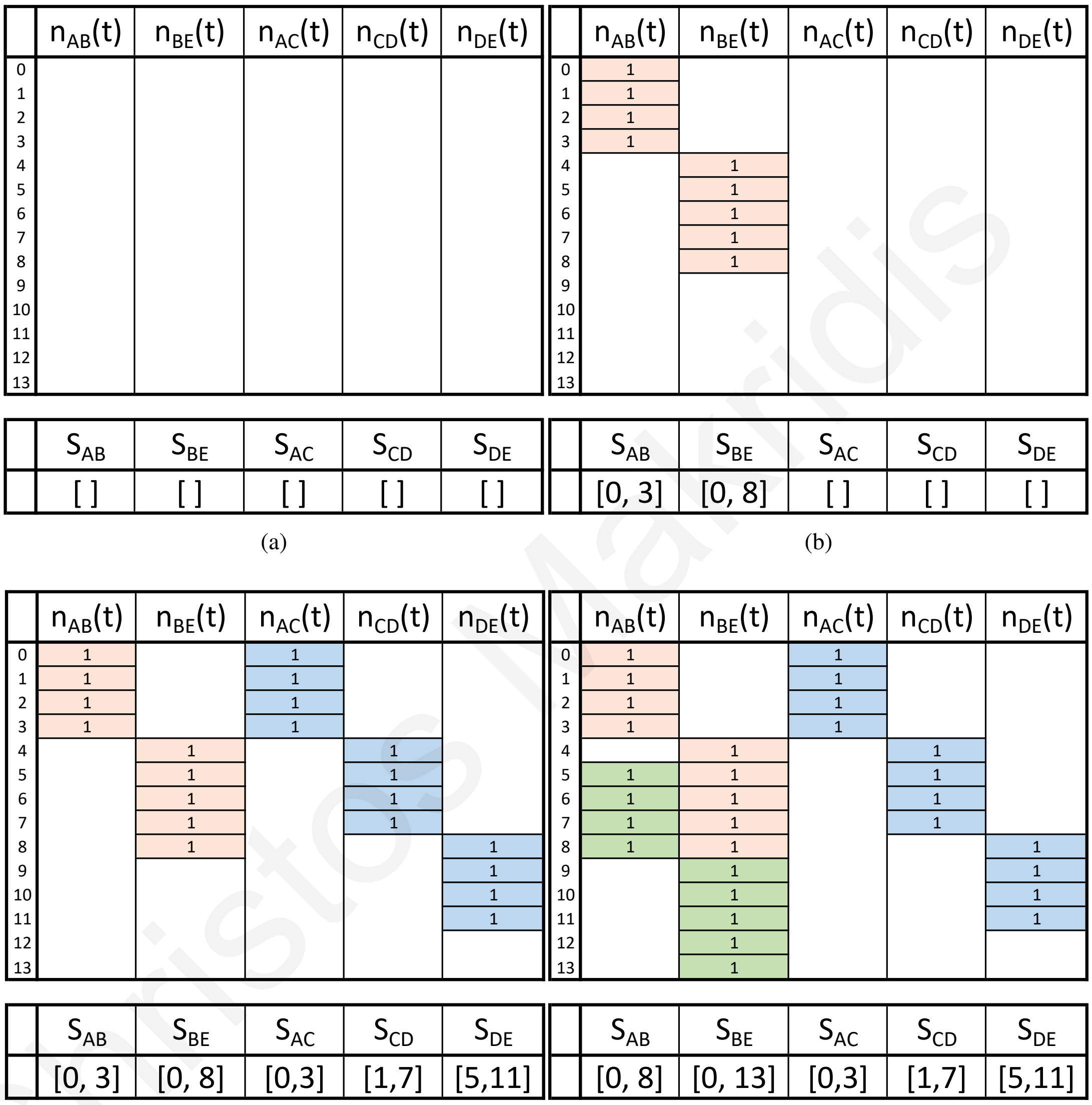
$$w_{CD}(9) = 0$$
$$c'_{CD}(9) = \tau_{CD} + w_{CD}(9) = 4$$
$$a_{CD} = c'_{CD}(9) + 9 = 13$$

Next, the lowest arrival time is for road segment (C, D). Therefore, departing from junction D at time $t = a_{CD} = 13$, the RRC checks the altered cost of road segment (D, E). The non-admissibility list $S_{DE} = \{[t^l_{DE1}, t^u_{DE1}] = [5, 11]$ does not contain $t = 13$, hence:

$$w_{DE}(13) = 0$$
$$c'_{DE}(13) = \tau_{DE} + w_{DE}(13) = 4$$
$$a_{DE} = c'_{DE}(13) + 13 = 17$$

Finally, the inner loop takes the road segment with the lowest arrival time, (B, E) with $a_{BE} = 14$, hence the route $\{(A, B), (B, E)\}$. Now that there are no intermediate delays $w_{ij}(t) > 0$ the outer loop ends and the appropriate accumulated numbers of reservations $n_{ij}(t)$ are being requested from the DB as shown in Fig. 9c. It then increases their values by one and updates their non-admissibility lists, resulting in Fig. 9d.

This concludes the toy example, with the proposed route to be $\{(A, B), (B, E)\}$, departing at time $t'' = 5$.



t	$n_{AB}(t)$	$n_{BE}(t)$	$n_{AC}(t)$	$n_{CD}(t)$	$n_{DE}(t)$
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

	S_{AB}	S_{BE}	S_{AC}	S_{CD}	S_{DE}
	[ ]	[ ]	[ ]	[ ]	[ ]

(a)

t	$n_{AB}(t)$	$n_{BE}(t)$	$n_{AC}(t)$	$n_{CD}(t)$	$n_{DE}(t)$
0	1				
1	1				
2	1				
3	1				
4		1			
5		1			
6		1			
7		1			
8		1			
9					
10					
11					
12					
13					

	S_{AB}	S_{BE}	S_{AC}	S_{CD}	S_{DE}
	[0, 3]	[0, 8]	[ ]	[ ]	[ ]

(b)

t	$n_{AB}(t)$	$n_{BE}(t)$	$n_{AC}(t)$	$n_{CD}(t)$	$n_{DE}(t)$
0	1		1		
1	1		1		
2	1		1		
3	1		1		
4		1		1	
5		1		1	
6		1		1	
7		1		1	
8		1			1
9					1
10					1
11					1
12					
13					

	S_{AB}	S_{BE}	S_{AC}	S_{CD}	S_{DE}
	[0, 3]	[0, 8]	[0,3]	[1,7]	[5,11]

(c)

t	$n_{AB}(t)$	$n_{BE}(t)$	$n_{AC}(t)$	$n_{CD}(t)$	$n_{DE}(t)$
0	1		1		
1	1		1		
2	1		1		
3	1		1		
4		1		1	
5	1	1		1	
6	1	1		1	
7	1	1		1	
8	1	1			1
9		1			1
10		1			1
11		1			1
12		1			
13		1			

	S_{AB}	S_{BE}	S_{AC}	S_{CD}	S_{DE}
	[0, 8]	[0, 13]	[0,3]	[1,7]	[5,11]

(d)

Figure 9: Database tables for the accumulated number of vehicle reservations $n_{ij}(t)$ and non-admissibility list S_{ij} for road segments (i, j): (a) No reservations, (b) First reservation, (c) Second reservation, (d) Third reservation

Chapter 5

Experimental results

The proposed developments are evaluated over a microsimulator (i.e., SUMO, [20]), which is a traffic simulation package designed for handling large networks. It has various configurations for making a scenario appropriate for different investigations. Among them is configuring a road network's parameters to represent reality. Moreover, it allows vehicles to be modeled using car-following models such as the Krauss' model [18], or to be set to travel at speeds that follow the speed limit. Moreover input flows in the network can be set up, while vehicles departure times can be set explicitly. SUMO is able to run the scenarios in a graphical representation, hence also visualizing congestion where it occurs. Finally, some statistics can be obtained from the run simulations, such as average times spent driving or waiting in traffic, average fuel used or emissions produced, as well as numbers of vehicles that have entered or left the road network over time.

For this implementation, SUMO simulator is considering an urban area in the city of Nicosia (e.g., Aglantzia area). Aglantzia is an area that its road network spans over $7\,\mathrm{km}^2$, consisting of approximately 2700 single-lane, 170 double-lane and 20 triple-lane road segments. The experiment is evaluated over ten different demand scenarios starting form $3000\,\mathrm{veh/h}$ to $21\,000\,\mathrm{veh/h}$ by increasing demand by $3000\,\mathrm{veh/h}$ for each subsequent scenario. All of them follow a demand duration of one hour, with simulation time-step $T = 0.5\,\mathrm{s}$. The O-D pairs of the requests are distributed randomly according to binomial distribution, with drivers modeled according to Krauss' car-following model, see [18]. Moreover, vehicles are assumed to travel at speed at capacity, while the critical density of road segments is set to $0.25 * \rho^J_{ij}$. For each scenario, a server was set up, running the implementation of the RRC and through the smartphone application, a client (i.e., a driver) connected to the server and requested routes[1] . The results of the dispatcher, instead of being sent to the client, were forwarded to the SUMO simulator, which in turn visualized the scenario.

5.1 Overall results

Fig. 10 depicts, the average travel times (TT) of all the vehicles that were routed in the road network, without elaborating any departure or intermediate delays. From the figure it is clear that for the lower flow rate scenarios of 3000 veh/h up to 12 000 veh/h, the average TT is similar for both approaches, approximately near 3.5 min, as no congestion is experienced. With a further increase in the flow rate of the NC, congestion starts to emerge, a fact that dramatically affects the TT. On the other hand with the same increase of flow rate, the RRC sustains the network's operation up to its critical capacity, benefiting the system operation by maintaining TT near to the no congestion conditions.

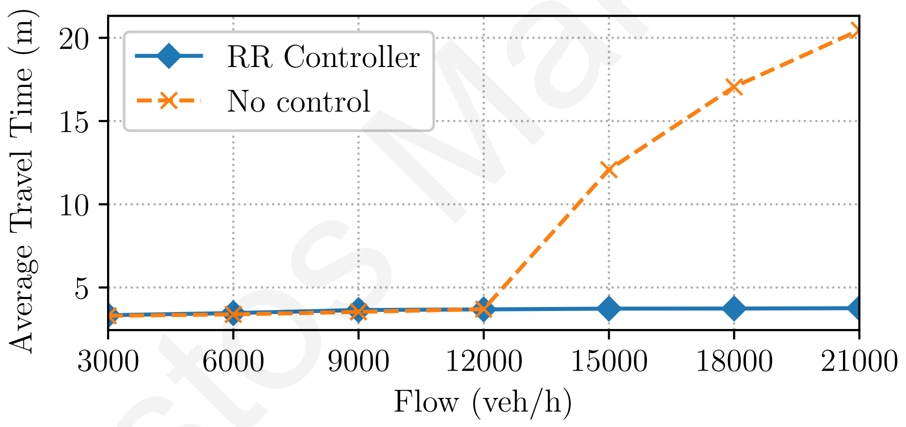


Figure 10: Average Travel Time

Fig. 11 indicates the average total travel times (TTT) of all the vehicles in each scenario. TTT incorporates all delays that occur between the departure and the arrival of a vehicle, but not the origin delays before departure. It can be observed that for the lower flow rate scenarios of 3000 veh/h up to 12 000 veh/h, the average TTT is similar for both approaches, at around 3.5 min. As the flow rates increase, the average TTT becomes higher for the NC, reaching almost 300 min, while for the RRC, the increase is insignificant remaining near 3.5 min.

[1]For the purposes of the experiment, the client is able to request more than one routes with different O-D pairs and departure times that comply with the desired flow rates of the experiment.

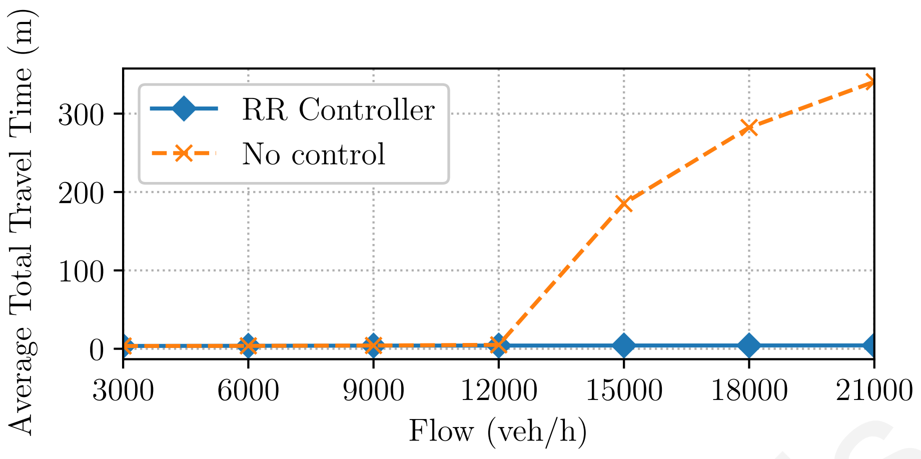


Figure 11: Average Total Travel Time

Fig. 12 shows the average origin delays that occur before departure of all vehicles in each scenario. According to the findings, as the flow rate increases, the average origin delays for the RRC increase, reaching 20 min for the largest flow rate scenario of 21 000 veh/h. Note that although the NC will never give origin delays, the ones given by the RRC are essential for maintaining no congestion, thus the average delayed departure times of the RRC are significantly lower than the average total delays of the NC.

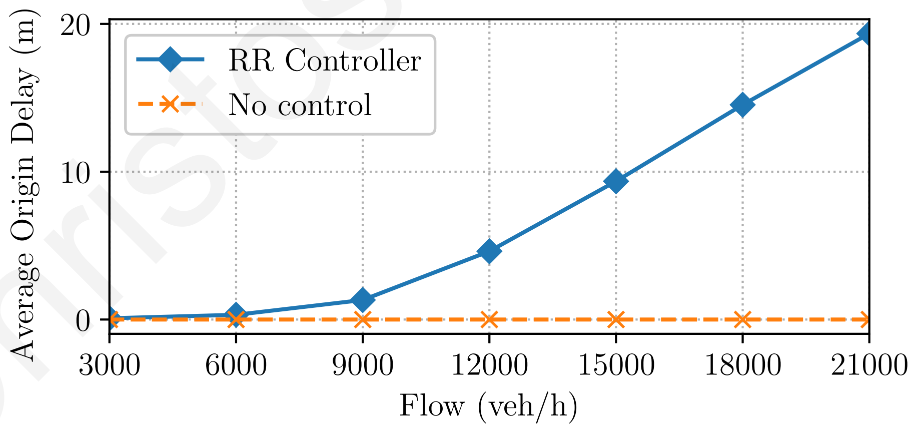


Figure 12: Average Origin Delay

The computational performance of the RRC approach is illustrated in Fig. 13, where the average computation time was measured to be between 30 ms and 41 ms for the lowest and highest demand scenarios, respectively. These results of serving a request are fast enough and thus are applicable for a real-life application.

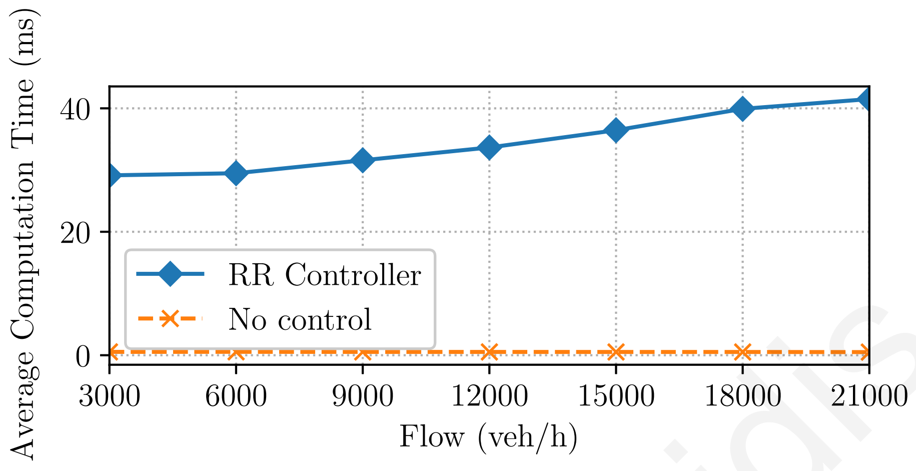


Figure 13: Average Origin Delay

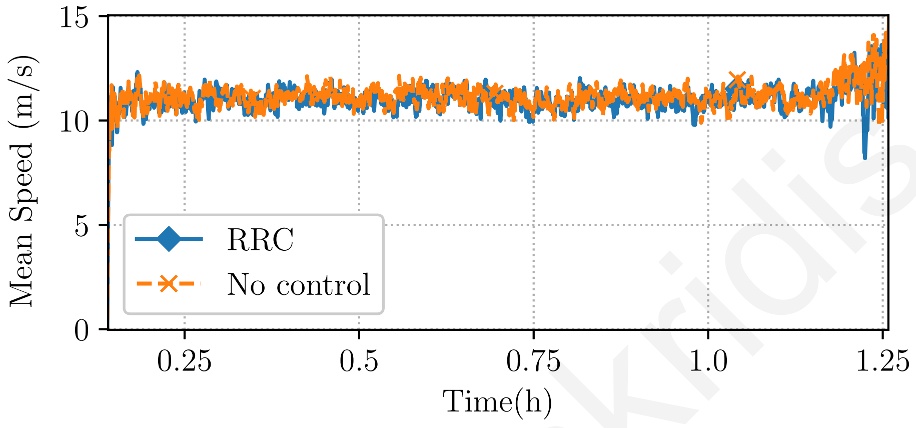
5.2 Lowest and highest demand results

Fig. 14 illustrates the mean speed of the lowest demand of 3000 $\mathrm{veh/h}$ and highest demand of 21 000 $\mathrm{veh/h}$. The results are similar between the two approaches for the low demand scenario, fluctuating around $11\,\mathrm{m\,s^{-1}}$. This is expected due to the fact that on a relatively empty network, the RRC acts like it uses a simple shortest path algorithm. Although the network is not empty, the demand is not high enough to cause significant changes in routes or origin delays. On the other hand, the high demand scenario results for the RRC to be fluctuating around the value of $10\,\mathrm{m\,s^{-1}}$, while for the NC to drop to nearly $0\,\mathrm{m\,s^{-1}}$ through the first $0.75\,\mathrm{h}$ and stay there.

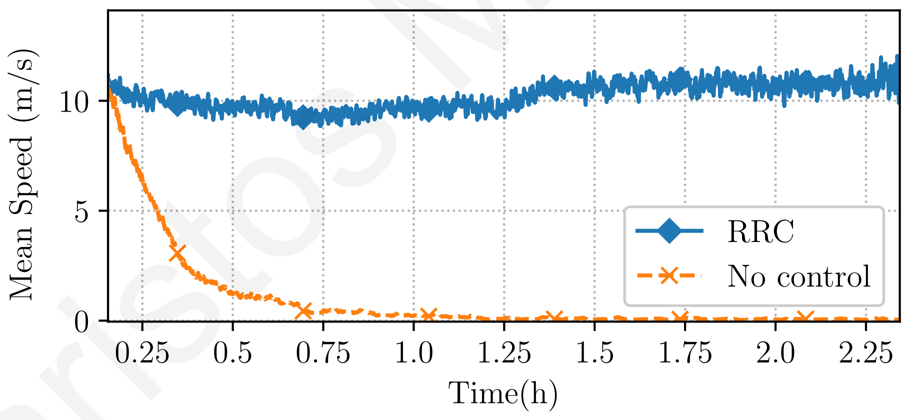
Figs. 15 depict the two scenarios' cumulative curves of vehicles for each approach. The cumulative curves for each scenario and each approach indicate the amount of vehicles that have departed from their origin and those that have arrived at their destination over time. As shown in both the low and high demand figures, the solid lines illustrate the results for the RRC, with the blue line for the departed and the orange line for the arrived at destination vehicles. On the other hand, dashed lines indicate the NC, with the green line for the departed and the red line for the arrived at destination vehicles. For the low demand scenario, the cumulative curves of both the RRC and NC seem to be almost identical for the departures. The same is true for the arrivals of both approaches. However, it is clear for the high demand scenario that these findings are much different. The NC seems to have tried to depart most of the vehicles in a relatively short period of time of about one hour, resulting in congestion, while the RRC spreads the demand over a larger period of around $2.25\,\mathrm{h}$. All of the RRC's vehicles have arrived at their destination shortly after their departure, while the network stays congestion free, as indicated by the horizontal difference of the departed and arrived vehicles. As expected, the NC, due to suffering from congestion, has much lower arrivals from the RRC at any time. Note that the RRC manages to have all of the $21\,000\,\mathrm{vehicles}$ arrived at their destination at around $2.5\,\mathrm{h}$, while the NC at $15.5\,\mathrm{h}$.

A closer look on the experiment, running a scenario with a flow rate of $2000\,\mathrm{veh/h}$ can be visualized in this video[2] .

[2]https://www.dropbox.com/s/f5tximh1mjirs4n/ RRA_vs_NoControl.mpeg?dl=0

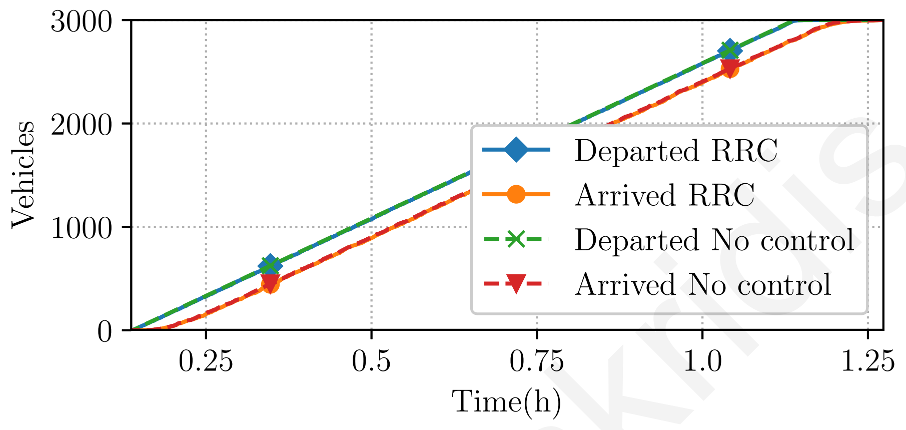


(a)

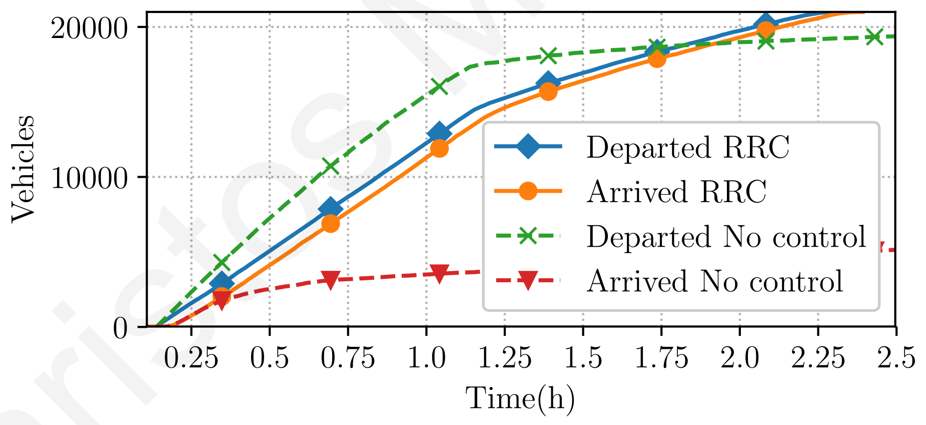


(b)

Figure 14: Mean speed for scenarios running the RRC and the NC: (a) Low demand scenario of 3000 veh/h, (b) High demand scenario of 21 000 veh/h.



(a)



(b)

Figure 15: Cumulative curves for scenarios running the RRC and the NC: (a) Low demand scenario of 3000 veh/h, (b) High demand scenario of 21 000 veh/h.

Chapter 6

Conclusions

This work implements all the required modules to realize the RRA as a real-life traffic and demand management mechanism. It defines a communication protocol to ensure a fast and reliable real-life interaction between the dispatcher and drivers. Moreover, it implements the appropriate algorithms for the RRA, considering the vast amount of data that have to be exchanged between different actors and to be stored in order to keep track of reservation status. This work solves any architecture's scalability issues by investigating for the appropriate database and using specific data structures and optimization techniques. Finally, it evaluates the developed modules over a large-scale urban area, where a microscopic simulation served as a realistic transportation network. The results showed that the RRA is indeed an implementable solution for solving the congestion-free routing problem, which is also applicable for real-life scenarios.

Future work will include the realization of the RRA for multi-region networks. This can be done by accumulating reservations in larger subareas instead of individual links. Moreover, a more ambitious application would be a pilot city where drivers will be using this work's implementation. Finally, an extension of this work could be realized, in which a pricing mechanism is conducted, responsible for handling driver deviation of the proposed routes and departure times.

Bibliography

[1] Konstantinos Aboudolas and Nikolas Geroliminis. Perimeter and boundary flow control in multi-reservoir heterogeneous networks. *Transportation Research Part B: Methodological*, 55:265–281, 2013.

[2] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. In *Algorithm engineering*, pages 19–80. Springer, 2016.

[3] David E Boyce, Allan Kirson, and Joseph L Schofer. Design and implementation of advance: The illinois dynamic navigation and route guidance demonstration program. In *Vehicle Navigation and Information Systems Conference, 1991*, volume 2, pages 415–426. IEEE, 1991.

[4] Chao Chen, Zhanfeng Jia, and Pravin Varaiya. Causes and cures of highway congestion. *IEEE Control Systems Magazine*, 21(6):26–32, 2001.

[5] Andy HF Chow. Properties of system optimal traffic assignment with departure time choice and its solution method. *Transportation Research Part B: Methodological*, 43(3):325–344, 2009.

[6] Jonathan Cohen. Graph twiddling in a mapreduce world. *Computing in Science & Engineering*, 11(4):29–41, 2009.

[7] Serdar Çolak, Antonio Lima, and Marta C González. Understanding congested travel in urban areas. *Nature communications*, 7(1):1–8, 2016.

[8] Daniele Condorelli. Efficient and equitable airport slot allocation. *Rivista di politica economica*, 1(2):81–104, 2007.

[9] Richard De Neufville, Amedeo R Odoni, Peter P Belobaba, and Tom G Reynolds. *Airport systems: Planning, design, and management*. McGraw-Hill Education, 2013.

[10] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[11] Lili Du, Shuwei Chen, and Lanshan Han. Coordinated online in-vehicle navigation guidance based on routing game theory. *Transportation Research Record*, 2497(1):106–116, 2015.

[12] Lester Randolph Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[13] Konstantinos Gkiotsalitis and Antony Stathopoulos. A mobile application for real-time multimodal routing under a set of users' preferences. *Journal of Intelligent Transportation Systems*, 19(2):149–166, 2015.

[14] Andrew V Goldberg and Robert E Tarjan. Expected performance of dijkstra's shortest path algorithm. *NEC Research Institute Report*, 1996.

[15] Jack Haddad and Nikolas Geroliminis. On the stability of traffic perimeter control in two-region urban cities. *Transportation Research Part B: Methodological*, 46(9):1159–1176, 2012.

[16] Yuxuan Ji and Nikolas Geroliminis. On the spatial partitioning of urban transportation networks. *Transportation Research Part B: Methodological*, 46(10):1639–1656, 2012.

[17] David E Kaufman and Robert L Smith. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *Journal of Intelligent Transportation Systems*, 1(1):1–11, 1993.

[18] Stefan Krauß, Peter Wagner, and Christian Gawron. Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55(5):5597, 1997.

[19] ZhenJiang Li, Cheng Chen, and Kai Wang. Cloud computing for agent-based urban transportation systems. *IEEE Intelligent Systems*, 26(1):73–79, 2011.

[20] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[21] Amin Mazloumian, Nikolas Geroliminis, and Dirk Helbing. The spatial variability of vehicle densities as determinant of urban network capacity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4627–4647, 2010.

[22] C Menelaou, P Kolios, S Timotheou, Christos G Panayiotou, and MP Polycarpou. Controlling road congestion via a low-complexity route reservation approach. *Transportation research part C: emerging technologies*, 81:118–136, 2017.

[23] Charalambos Menelaou, Panayiotis Kolios, Stelios Timotheou, and Christos Panayiotou. On the complexity of congestion free routing in transportation networks. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 2819–2824. IEEE, 2015.

[24] Charalambos Menelaou, Stelios Timotheou, Panayiotis Kolios, and Christos G Panayiotou. Improved road usage through congestion-free route reservations. *Transportation Research Record*, 2621(1):71–80, 2017.

[25] AB MySQL. Mysql, 2001.

[26] Syeda Noor Zehra Naqvi, Sofia Yfantidou, and Esteban Zimányi. Time series databases and influxdb. *Studienarbeit, Université Libre de Bruxelles*, 2017.

[27] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . `https://www.openstreetmap.org`, 2017.

[28] Christos G Panayiotou and Christos G Cassandras. A sample path approach for solving the ground-holding policy problem in air traffic control. *IEEE Transactions on control systems technology*, 9(3):510–523, 2001.

[29] Qing Kent Pu and Hui Henry Li. Mobile navigation system, September 18 2001. US Patent 6,292,743.

[30] Lin Xiao and Hong K Lo. Adaptive vehicle navigation with en route stochastic traffic information. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):1900–1912, 2014.

[31] Peijun Ye, Cheng Chen, and Fenghua Zhu. Dynamic route guidance using maximum flow theory and its mapreduce implementation. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 180–185. IEEE, 2011.

[32] Mehmet Yildirimoglu and Nikolas Geroliminis. Approximating dynamic equilibrium conditions with macroscopic fundamental diagrams. *Transportation Research Part B: Methodological*, 70:186–200, 2014.

[33] Mehmet Yildirimoglu, Mohsen Ramezani, and Nikolas Geroliminis. Equilibrium analysis and route guidance in large-scale networks with mfd dynamics. *Transportation Research Procedia*, 9:185–204, 2015.