

A Real-Time Demand Management and Route Guidance System for Eliminating Congestion

Christos Makridis, *Student, IEEE**, Charalambos Menelaou, *Member, IEEE**, Stelios Timotheou, *Senior, IEEE**,
and Christos G. Panayiotou, *Senior, IEEE**

* KIOS Research & Innovation Center of Excellence,
Department of Electrical and Computer Engineering,
University of Cyprus, Nicosia, Cyprus

{cmakri07, menelaou.charalampos, timotheou.stelios, christosp}@ucy.ac.cy

Abstract—Traffic congestion is a problem that burdens urban cities daily. Despite the development of numerous research methodologies and technological solutions, the problem still persists. This paper presents a system that combines traffic and demand management mechanisms to avoid the emergence of congestion by sustaining the occupancy of each road in the network up to a threshold, using a reservation scheme and keeping track of the future states of the network. Traffic management is responsible for navigating vehicles through congestion-free paths, while demand management is responsible for identifying the time that each vehicle will enter the network. Considering the size of actual urban networks and the number of vehicles utilizing the infrastructure, this is not an easy task. This work designs and validates a responsive and scalable real-life system for online traffic and demand management, while addressing the challenge of managing and processing large numbers of requests and data. The complexity and feasibility of the system are evaluated through microsimulations of real and artificial road networks. Its traffic efficiency is also evaluated, by running extensive simulations of a real city, emulating realistic conditions in different scenarios.

Index Terms—System Design, Control, Optimization of Intelligent Transportation Systems, Congestion

I. INTRODUCTION

URBAN mobility is vital for the sustainable growth and development of modern societies. Despite their great technological prosperity, societies still face one of the greatest challenges of urban mobility, namely the problem of traffic congestion. Traffic congestion occurs when demand for mobility surpasses the network's capacity [1], resulting in increased commuting times, long queues, and gridlocks. This can give rise to anxious commuters, eventually deteriorating the quality of life, health and safety of citizens [2].

Recent technological advancements have enabled the development of smart navigation services that find routes for drivers, such as Google Maps [3], Waze [4], and Here WeGo [5]. All currently available navigation services aim to improve the travel experience of drivers by reducing the duration of

individual journeys (i.e., user optimum). However, the goal of reducing individual travel times can negatively impact the road network as a whole (i.e., system optimum) [6]. In particular, when there is traffic in some roads, a large number of drivers tend to avoid these roads. Subsequently, the demand for other parts of the network increases, triggering congestion and traffic oscillations [7].

A solution to the problem of finding the system optimum is to effectively avoid the creation of congestion and prevent traffic oscillations. This can be achieved by the proposed infrastructure reservation scheme that combines traffic management for guiding vehicles in uncongested areas and demand management for controlling the time that vehicles enter the network. Assuming free-flow conditions are ensured, the reservation scheme is able to estimate the future number of vehicles traversing any road segment at any time, allowing it to restrict the number of vehicles in any road segment below a designated threshold (i.e., capacity threshold). Note that even though we may not know the precise capacity of every road, we know that a maximum capacity exists and it can be estimated. By routing vehicles through road segments only when their estimated capacity has not been reached, free-flow conditions are ensured.

This work aims to design, develop, analyze, and validate a system that implements reservations for realizing a joint traffic and demand management solution to mitigate congestion in the network. Simultaneously, the system aims to optimize the performance of all users (system-wide optimization) by formulating and solving the Earliest Arrival Time at Destination (EATD) problem on a network that changes over time which proves to be an NP-complete problem [8]. Using the route reservations it is possible to predict the future state of every road, in a sense that ensures all future vehicles will always be routed through non-congested roads of the network. In detail, we propose a complete system design, including a communication protocol of a client-server model, which allows clients (i.e., conventional or autonomous vehicles) to determine their start times and traverse routes while ensuring that the road network remains uncongested at all times. Each client's request contains their Origin-Destination (OD) pair and desired departure time. The server (i.e., the Route Reservation Controller (RRC)) solves the EATD problem and determines the time a vehicle should depart and the route

This work is supported by the European Union (i. ERC, URANUS, No. 101088124, and ii. Horizon 2020 Teaming, KIOS CoE, No. 739551), and the Government of the Republic of Cyprus through the Deputy Ministry of Research, Innovation, and Digital Strategy. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

to be followed. Through the reservations, the RRC keeps track of the number of vehicles that enter a road segment at any time, ensuring that new reservations will maintain each road segment's occupancy under a predetermined threshold. This system design allows for the complete elimination of congestion by ensuring that excess demand is distributed over the network both in space and time.

The contribution of this work is the design and implementation of a complete system architecture that is applicable to any urban traffic network in real life, capable of accepting journey requests by users and deciding their departure time (demand management) and assigned path (traffic management), while ensuring that no congestion will occur, by implementing a reservation scheme to know the future state of the roads in the network. This allows our system to prevent oscillations and ensure that all vehicles will be guided through non-congested roads. Aiming for fast response times [9], the system utilizes various techniques to integrate and enhance the state-of-the-art algorithm for the reservation scheme [8]. Specifically, we:

- Design the entities and their roles for the various operations of the system. These are: 1) the Clients, which can be conventional or autonomous vehicles, 2) the Server (i.e., the RRC), and 3) the Reservation Database (DB), which stores the *reservation data* of the system.
- Develop the designed system by defining a *Communication Protocol* which describes how the entities interact with each other, and how they handle *reservation data*. In this stage, we achieve reduced response times by using appropriate data structures, algorithms, and optimization techniques.
- Assess the complexity and scalability of the developed system by running and comparing exhaustive simulation experiments of real cities and artificial road networks under low, medium, and high demands.
- Analyze the traffic performance of the developed system on how well it copes with the traffic congestion problem through microsimulation scenarios of a real city, emulating realistic conditions. A key finding is that the average total travel time of users is minimized, for threshold values near the critical capacity threshold of the network.

The rest of the paper is organized as follows. A literature review is discussed in Section II. Section III gives the conceptual framework of the system, while Section IV describes its development and optimizations. Section V provides an evaluation of the system regarding computational complexity. Section VI validates the traffic performance of the system, regarding how well it solves the traffic congestion problem, by running realistic microsimulations. Finally, Section VII concludes this work and elaborates future research avenues.

II. LITERATURE REVIEW

Over the years, many mechanisms have been developed and used for coping with traffic congestion. Some main categories include infrastructure expansion, traffic management, and demand management. Although they have some similarities, each category focuses on different strategies for reducing traffic congestion in road networks.

Infrastructure expansion is a mechanism well used by many cities. As the name suggests, it alters the infrastructure to increase the capacity of the network, thus allowing more vehicles in the network. It achieves this by building and widening roads and intersections, while increasing the available transportation modes (e.g., more bus routes). Although such methods can be effective, they are costly to build and maintain and most importantly, they are not resilient to future growth of cities, due to the limited space for infrastructure expansion [10].

Traffic management is another group of strategies, aiming to improve traffic efficiency and safety by dealing with the overall traffic flow, rather than increasing the capacity of the network. It manages and controls the traffic on road networks by utilizing operational strategies, such as perimeter control and route guidance.

Perimeter Control [11] strategies aim to control the amount of traffic flow within an area by allowing or restricting vehicles at the boundaries of the area. More specifically, they split the road network of the area into sub-networks (i.e., regions), trying to maintain the density of vehicles in each region under a desired threshold [12]. To achieve this, they restrict the flow at the borders of the regions by using various control mechanisms (e.g., street closures, traffic light signaling, traffic flow metering). Although perimeter control is a promising approach for coping with congestion, it suffers from the inability to control endogenous flows within each region and forces vehicles to queue outside each region, therefore shifting the problem to the borders [13].

Route guidance manages traffic by providing drivers with real-time routes and travel information [14]. Route guidance methods utilize data from road sensors, traffic surveillance systems, or other real-time and historical data to estimate/predict traffic conditions and dynamically plan the routes of travellers [15]. An example of a route guidance service is Waze [4], [16] which was bought by Google in 2013 [17]. Waze works by engaging drivers to participate in real-time sharing of information through the Waze mobile application. It also encourages drivers to inform the system about accidents, speed traps, or give feedback on the accuracy of the acquired data [18]. This information is put together to construct traffic conditions across the network, allowing Waze to compute optimal routes that avoid congested roads or other unpleasant road events. Although being common in modern transportation systems, route guidance schemes alone, aim to minimize the travel times of drivers (i.e., individual benefit), overlooking their impact on the road network as a whole (i.e., system optimum) [6], [7]. This is important when many vehicles are guided through the same, so far uncongested roads of the network, leading to a demand increase in these roads, eventually causing traffic oscillations [19]. Furthermore, when the demand of a network is exceedingly high, route guidance alone is incapable of managing congestion, as the network becomes overpopulated.

Demand management mechanisms reduce or redistribute the traffic demand of a network during peak-hours, or encourage commuters to use alternative modes of travel, to limit congestion [20]. Examples of such mechanisms found in many cities include access control, park and ride intermodality, and

TABLE I
COMPARISON OF METHODOLOGIES

Methodology	Advantages	Disadvantages	References
Infrastructure Expansion	<ul style="list-style-type: none"> Increases network capacity. 	<ul style="list-style-type: none"> High cost to build and maintain. Limited by available space. Not sustainable. 	[10]
Perimeter Control	<ul style="list-style-type: none"> Controls traffic entering a region to maintain desired density. 	<ul style="list-style-type: none"> Cannot control internal flows. May create queues at region borders. 	[11]–[13]
Route Guidance	<ul style="list-style-type: none"> Provides real-time route information. Uses data to estimate/predict traffic conditions. 	<ul style="list-style-type: none"> Focuses on individual benefit rather than system-wide optimization. Can lead to traffic oscillations. 	[3]–[7], [14]–[19]
Demand Management	<ul style="list-style-type: none"> Reduces/redistributes traffic demand. Encourages use of alternative transport modes to limit congestion. 	<ul style="list-style-type: none"> Effectiveness depends on commuter compliance and cooperation. 	[20]–[27]
Joint Traffic and Demand Management (Route Reservation Systems)	<ul style="list-style-type: none"> Prevents network congestion at all times. All commuters experience minimal travel times. Any wait time that would have happened in the network will be productively reclaimed at the origin, before entering the transportation network. 	<ul style="list-style-type: none"> Requires initial implementation effort. Wait times at the origin. Depends on user compliance with reservations. 	[8], [28]–[31]

pricing (e.g., congestion pricing, area pricing, parking pricing, etc.) [21]. Furthermore, joint demand and traffic management have been addressed in a Macroscopic framework [29], [30].

Pricing strategies manage the demand and congestion on road networks by applying tolls and prices, influencing the commuters' choices during peak-hours. Some methods have a fixed toll, while others have a varied toll, based on the current congestion state, incentivizing commuters to use less congested parts of the network [22]. Other methods have a credit-based pricing scheme, rewarding commuters that choose uncongested roads, while penalizing commuters who choose more convenient roads during peak-hours, potentially contributing to congestion [23]. However, this puts commuters in a dilemma on sacrificing efficiency over some cost, which brings up the issue of social fairness.

Some other examples of demand management include motivating commuters to use alternative modes of transport (e.g., bicycles, buses, ride sharing schemes) [24], or introducing lanes to the road infrastructure which are restricted to High-Occupancy Vehicles (HOV) during peak-hours, (e.g., vehicles with more than one passenger like buses or car pools). However, the effectiveness of such schemes is limited to the willingness of commuters to utilize them when general purpose lanes are congested [25]. Also, applying flexible working hours and work from home policies, allow commuters to choose commuting during off-peak hours or to not commute at all [26], [27], essentially reducing the demand during peak-hours.

Artificial Intelligence (AI) and Machine Learning (ML) approaches have formed a promising advancement in intelligent transportation. There are various studies in the context of AI algorithms and ML for optimizing traffic management, addressing issues like smart traffic signal control, flow prediction, congestion detection, and automatic signal recognition [32]

[33].

Route reservation architecture stands out as a promising technology with the potential to revolutionize traffic management and congestion mitigation, without any training model. It is inspired by a reservation scheme for the ground-holding problem in Air Traffic Management (ATM), for airplanes to avoid congestion when reaching their landing runway. It works by dividing the airspace capacity both in space and time [28], which is effective in increasing the airport utilization, regardless of the runway capacity remaining constant. This is also the main motivation for the reservation scheme of urban areas [31], seeking for a solution to traffic congestion, without the need of expanding the infrastructure.

Some real-world examples of the effectiveness of large-scale reservation management can be viewed in various applications other than traffic, such as doctor appointments, restaurant reservations, or event bookings for stadiums or theaters. Specifically, the capacity of such systems is considered as the number of clients that each service can serve for predefined periods of time (e.g., the number of patients that a doctor can examine in an hour, the number of customers that a restaurant can accommodate for dinner, or the number of seats that a theater has for an entire event). Furthermore some of the stay-at-home policies applied against the pandemic of COVID-19 can be viewed as a form of demand management, requiring citizens to request an allowance for moving in the city for a specific period of time, unintentionally impacting traffic conditions positively [34].

Motivated by the aforementioned reservation strategies, this work proposes a system which jointly combines traffic and demand management to solve the traffic congestion problem through a reservation scheme. This combination allows the system to achieve traffic efficiency without the need of expanding the infrastructure. Also, due to its demand management



Fig. 1. System's concept overview.

nature, it is able to control endogenous flows, and prevent the network from becoming overpopulated. In this way, the formation of queues in the network is avoided through the introduction of delayed departures (i.e., origin delays), contrary to other strategies such as perimeter control. Transferring delays outside the road network implies that users can constructively utilize their time while waiting for departure. Furthermore, the utilization of route reservations ensures a system-wide approach towards route guidance, eliminating traffic oscillations. Subsequently, the proposed system is network-wide efficient, promising its users the earliest arrival at their destination.

Table I outlines the key advantages and disadvantages of the methodologies reviewed in this work.

III. CONCEPTUAL FRAMEWORK

The proposed system aims to allow a scalable joint traffic and demand management mechanism that is applicable to real urban traffic networks. In this context, each client of the network requests a route reservation from the system prior to their departure, by providing an Origin and Destination pair (OD pair), as well as a desired departure time. The system then analyzes the already received reservations to find an appropriate path at a specific departure time to assign to the client. The assigned path and departure time are guaranteed to give the earliest arrival time at the destination of the request, while sustaining the occupancy of all road segments of the network up to a predetermined threshold. A low threshold would lead to an underutilized network, while a high threshold would lead to congestion. Finding the optimal threshold leads to efficiently utilizing the network while not experiencing any congestion.

Fig. 1 illustrates the conceptual framework with a toy example. The client (red vehicle) submits a route reservation request to the RRC to go from the origin O to the destination D , by providing their OD pair and their desired departure time. The prior reservation data of each horizontal road segment

are illustrated as bar-plots, showing the number of already reserved vehicles over time. Their threshold is set to 5 vehicles per time-slot. For simplicity of the example, vertical road segments are assumed to have infinite threshold, while the travel time of both horizontal and vertical road segments is 1 time-slot. For the rest of the example, we will refer to a (horizontal or vertical) road segment i traversed at time-slot k as the tuple $\langle i, k \rangle$. The figure shows that $\langle H1, 1 \rangle$, and $\langle H1, 2 \rangle$ have reached the threshold of 5 vehicles, making the RRC to consider $H1$ as *non-admissible* at time-slots 1, and 2, meaning that a vehicle is not eligible to drive through $H1$ during these time-slots. Similarly, $\langle H2, 3 \rangle$, $\langle H2, 4 \rangle$, $\langle H2, 5 \rangle$, $\langle H3, 2 \rangle$, and $\langle H4, 3 \rangle$ are all considered non-admissible, while any other $\langle i, k \rangle$ is considered admissible. Given the above, the RRC should avoid to guide any vehicle through road segments which will become non-admissible when the vehicle is ready to traverse them.

Assuming the desired departure time is $k = 0$, the RRC considers the first two options for guiding the client to traverse: a) $\langle H1, 1 \rangle$, or b) $\langle V1, 1 \rangle$. However, a) is non-admissible and cannot be chosen, while b) is admissible and thus selected. Nevertheless, following b) leads to the non-admissible $\langle H3, 2 \rangle$ in the next time-slot, rendering option b) also not feasible. Left with no other choice, the client cannot depart at $k = 0$, so the RRC introduces a wait time of 1 time-slot at the origin. By departing at $k = 1$, the RRC has two new options for guiding the client through: a) $\langle H1, 2 \rangle$, or b) $\langle V1, 2 \rangle$. However, a) is non-admissible and cannot be chosen, while b) is admissible and thus selected. Following b) allows the client to be guided through the admissible $\langle H3, 3 \rangle$. From there, the RRC has two new options for guiding the client through: a) $\langle V2, 4 \rangle$, or b) $\langle H4, 4 \rangle$. Despite both being admissible, option a) leads to the non-admissible $\langle H2, 5 \rangle$. Hence, the RRC discards a) and chooses b). From there, the client can reach the destination $\langle D, 6 \rangle$, by traversing $\langle V3, 5 \rangle$. The light blue line indicates the path that the RRC has assigned to the client

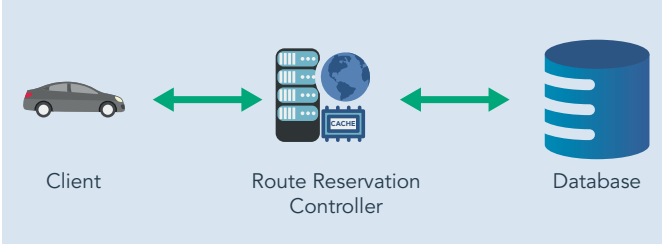


Fig. 2. System overview

$(O, V1, H3, H4, V3, D)$, while the text at the start of the path denotes the assigned departure time, which incorporates the wait time at the origin of 1 time-slot. Once a feasible path is computed, the reservations are also updated, thus $\langle H3, 3 \rangle$ will increase to 4, and $\langle H4, 4 \rangle$ will become 5.

For the operation of the system we define three entities, illustrated in Fig. 2:

- *Client* (i.e., conventional or autonomous vehicles)
- *Route Reservation Controller* (RRC)
- *Reservation Database* (DB)

For deploying the system in a real city, the client entity should be able to send route reservation requests to the RRC and receive responses through a User Interface (UI). This can be implemented in any platform, such as a web-application, onboard units, or smartphone apps, with no significant hardware requirements. Furthermore, both the RRC and the DB should ideally be hosted on a server (or two separate servers), and deployed either locally or as cloud services. Notably, the server requirements are not demanding, with experimental evidence in Section V demonstrating that a reasonably capable computer can proficiently manage numerous requests, even for large networks. The design of the entities is detailed in the subsequent three subsections. The fourth subsection defines the *Communication protocol*, which brings the three entities together.

A. Client

The *Client* entity is the interface from which clients can send new route reservation requests to the RRC. We designed the interface as a web-application that can run over various platforms (e.g., onboard units, mobile phones). For this work, we implemented a proof of concept User Interface (UI) as an Android [35] application, from which clients can form a route reservation request q and send it to the RRC. The request and its corresponding response are encoded in JSON format, which is a lightweight, yet compact way to move data between entities. Table II presents this encoding, where the request q is shown to consist of an origin O_q , a destination D_q , and a desired departure time \tilde{t}_q^d . The origin and destination are further described as pairs of latitude and longitude coordinates. Similarly, the response consists of an assigned departure time and a path, which is described by various points in space. Each point indicates the beginning of each road segment in the path, described as pairs of latitude and longitude coordinates.

For achieving user-friendly visualizations, the application decodes the JSON format into dialogues, map indicators

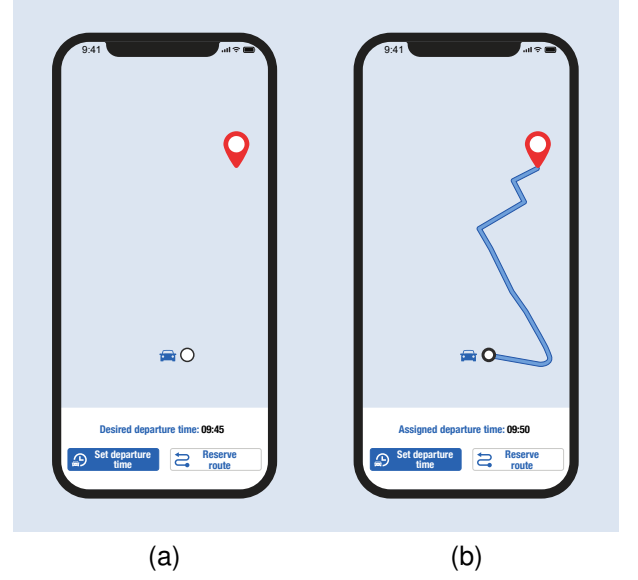


Fig. 3. Client Interface for: (a) Reservation Request, (b) Response.

TABLE II
JSON REPRESENTATIONS OF ROUTE REQUEST AND RESPONSE

Route Request q						
Origin O_q		Destination D_q		Desired departure time \tilde{t}_q^d		
Lat	Lon	Lat	Lon			

Route Response q						
Path \mathcal{P}_q^*						Assigned Departure Time t_q^{d*}
Road 1		Road 2		...	Road N	
Lat	Lon	Lat	Lon		Lat	Lon

and buttons. Fig. 3 illustrates the two screens of a route reservation user interface for clients, which includes the basic functionalities that are necessary for this work. In particular, Fig. 3a depicts the screen that enables clients to choose their origin and destination, by putting a black circle and a red pin-point, respectively. The desired departure time can be set by clicking the “Set departure time” button and then the request can be sent through the “Reserve route” button. At that time, the RRC calculates a solution as described in Section III-B and creates a response containing the assigned path \mathcal{P}_q^* and departure time \tilde{t}_q^{d*} . As seen on Fig. 3b, the information of the response can be interpreted as the route indicated with a blue line and a dialog box that indicates the assigned departure time.

B. Route Reservation Controller

The RRC runs on a web-server, responsible for serving all client route reservation requests for a road network, following a First Come First Served (FCFS) policy. It formulates the road network as a dual graph [36] $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where the vertices set \mathcal{V} is the set of all road segments in the network and the edges set \mathcal{E} is the set of all connections between the road segments. For each route reservation request q , the RRC formulates the problem of the Earliest Arrival Time at Destination (EATD), and it can solve it using any method. In this work we use a version of the state-of-the-art algorithm for solving the EATD problem using reservations, the Iterative Dijkstra Approach (IDA) [8], with input the 3-tuple $\langle O_q, D_q, \tilde{t}_q^d \rangle$ that forms the request q , and the future *reservation data*, which will be further

discussed in Section IV-B. Here, $O_q \in \mathcal{V}$ and $D_q \in \mathcal{V}$ are the origin and destination road segments, and $\tilde{t}_q^d \in \mathbb{N}$ is the desired departure time-slot from the origin of request q . Based on the input, the IDA outputs a path \mathcal{P}_q^* , and an assigned departure time t_q^{d*} , ensuring the earliest arrival time to the destination for which the number of vehicles of all road segments in the network $\forall i \in \mathcal{V}$ at any time-slot are maintained up to their predetermined threshold h_i , which is defined as:

$$h_i = H \cdot \frac{L_i \cdot N_i}{L} \approx H \cdot \rho_i^J, \quad (1)$$

where H is a percentage defining a *system-wide threshold*, L_i is the length of road segment i , N_i is the number of its lanes, L is the average vehicle length (including the minimum gap between vehicles), and ρ_i^J is the jam density of road segment i . Subsequently, the choice of threshold percentage H determines how many vehicles are allowed in any road at any instance. The optimal *system-wide capacity threshold* H^* that maximizes the flow of the network, while maintaining no congestion, can be approximated through many methods. Inspired by the work in [31], we chose to use the Macroscopic Fundamental Diagram (MFD) [37] of the network, which allows us to approximate H^* as:

$$H^* = \frac{\rho^C}{\rho^J}, \quad (2)$$

where ρ^C is the critical density of the network when the flow is maximum, ρ^J is its jam density, and both can be obtained from the MFD, or other approaches (e.g., ML/AI, perturbation analysis, etc.).

The RRC also estimates the position of the vehicle, in order to reserve the appropriate road segments, during the time of their expected traversal. For the estimation, it uses the average free-flow speed v_f of vehicles in the network, which can also be estimated through many methods. Again, we chose to use the MFD to acquire v_f . It is important to note that v_f incorporates any traffic signals delays in the network.

The RRC uses the output of IDA to make the appropriate reservation by updating the reservation data on the DB, and creating a response to inform the client about the result.

C. Reservation Database

The *reservation database* is a storage system responsible for storing the *reservation data* of each road segment for all time-slots. Storing and utilizing large amounts of data in big cities imposes a challenge. For efficient data writing and retrieval, we selected InfluxDB [38], a time-series DB, optimized for handling time-series data, such as *reservation data* which are spatio-temporal.

Although InfluxDB *reads* and *writes* data exceptionally fast, its *update* and *delete* operations are not intended for real-time use, forming a challenge for our system. As new requests are served, updating *reservation data* is necessary, resulting in slow response times and data inconsistencies. To overcome this, we developed a *versioning scheme*, which instead of *updating* the reservation data of each road segment i , it *writes* new data by keeping track of their version e_i . Each time new data are written in the DB, the RRC versions them as

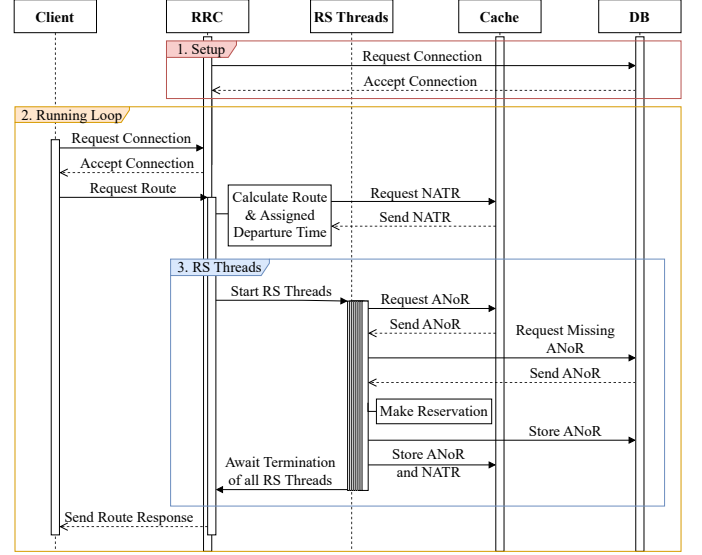


Fig. 4. Communication Protocol sequence diagram. Vertical rectangles describe active processes of the entities. Parallel vertical rectangles describe parallel processes. Solid-line arrows represent request messages between entities, and dashed-line arrows represent reply messages.

$e_i = e_i + 1$. When retrieval is needed, the DB *reads* only the data with the latest version. For simplicity, we will abbreviate versioning in this work's mathematical notation, assuming that we always read and write data in the latest version.

Nonetheless, writing all reservation updates on the DB leads to enormous storage requirements and latency due to frequent communication with the DB. To cope with this, we introduced a *cache mechanism* that creates a *cache window* for each road segment in the network to *write* and *update* reservation data of predefined time periods, locally on the RRC, instead of the DB. After the predefined period passes, the RRC *writes* the updated reservation data to the DB using the aforementioned *versioning scheme*. The *cache window* then clears any unnecessary reservation data, becoming ready for the next time period. This not only reduces the amount of stored data in the DB, but also the communication between the RRC and the DB, allowing faster responses to route reservation requests.

D. Communication Protocol

The *Communication protocol* defines how the three entities of the system interact, and how they handle reservation data. Fig. 4 illustrates the sequence diagram of the protocol, which shows the initialization process of the RRC, requesting a connection to the DB. The RRC is then ready to accept connections from clients through its main phase. Each Client requests a new connection to the RRC, which the RRC accepts. The Client can then form a route reservation request q and send it to the RRC, which runs the IDA to find the optimal path \mathcal{P}_q^* and assigned departure time t_q^{d*} . To do so, the IDA needs to check the current and future state of reservation data for each road segment. These are stored as *Non Admissible Time Ranges (NATR)* in the cache, which the RRC can request to read. The NATR and cache are described in detail in Sections IV-B, and IV-C, respectively. The RRC then initiates

a Road Segment Thread (RS Thread) for each road segment in the path, which is responsible for updating its corresponding *Accumulated Number of Reservations (ANoRs)*. ANoRs are also described in Section IV-B. If some of the ANoRs needed to make the reservation are not stored in the cache, the RS Thread requests them from the DB, receives them as a response, and updates both NATR and ANoRs locally on the cache to make the reservation. When it is time to clear the cache, the RRC writes the ANoR to the DB. Finally, when all RS Threads terminate, the RRC sends a Route Response back to the Client with the resulted path \mathcal{P}_q^* and assigned departure time t_q^{d*} .

IV. SYSTEM DEVELOPMENT AND OPTIMIZATION

The proposed system aims to efficiently respond to route reservation requests of real traffic networks, fast and reliably. In the following subsections we describe the elements involved in the development process to achieve that. First we describe the chosen routing algorithm and its implementation. Then, we define the structure of the *reservation data* which are necessary for processing route reservation requests, and finally how they are utilized by the *Cache Mechanism* and the *RRC*.

A. Routing algorithm

Our proposed system can use any algorithm to solve the EATD problem, which under the admissibility constraints is an NP-complete problem [8]. For that, we chose to use the IDA, which is currently the state-of-the-art heuristic solution for the EATD problem [31]. Given a source, a destination, and a desired departure time, IDA uses the current and future reservation data of the network to output a path and an assigned departure time which ensure the earliest arrival time to the destination, while sustaining the occupancy of all road segments in the network at or below their predefined threshold.

The IDA formulates an *inner* and an *outer* loop [39]. The *inner loop* is a time-dependent [40] extension of the Dijkstra algorithm [41], yielding a path from the source to the destination. Some of the road segments in the path might be non-admissible during the time of their traversal, therefore some wait time is necessary for them to become admissible. The *outer loop* is responsible for discarding such paths by iteratively calling the *inner loop*, each time with a different departure time, until the resulting path consists only from admissible road segments (i.e., the only waiting time is at the origin).

Specifically, the original Dijkstra algorithm finds the shortest route in terms of travel time, from a single source to all destinations. On the other hand, the Uniform-Cost-Search [42] (UCS) algorithm, although similar to the Dijkstra algorithm, is a single source, single destination shortest route algorithm. We propose an adaptation of UCS instead of Dijkstra to the IDA. The benefit in performance of UCS over Dijkstra for our system lies on two things: 1) It stops iterating once reaching the destination road segment, while the Dijkstra algorithm stops when it reaches all road segments. 2) It looks for the road segment with the lowest cost over the set of the road segments that someone can visit during the current iteration (i.e., the

frontier), while the Dijkstra algorithm tries to look for it out of all the road segments of the network for every iteration of its operation. The utilization of the frontier narrows the search for the lowest cost significantly, thus allowing faster convergence to the solution of a shortest route between two points. Finally, the work in [39] proposes the use of a heap, in the Dijkstra adaptation of IDA, thus improving its complexity. The use of heap can also be implemented in the UCS adaptation of the IDA, yielding further computational improvement.

B. Reservation Data

When a client makes route reservation request q , the RRC determines a path \mathcal{P}_q^* for the client to follow and proceeds to reserve it by updating the *reservation data* accordingly. We define two main data structures associated with the *reservation data*:

- *Accumulated Number of Reservations (ANoRs)*: An ANoR $C_i(k) \in \mathbb{N}$ denotes the number of vehicles that are reserved for road segment $i \in \mathcal{V}$ at a time-slot $k \in \mathbb{N}$. ANoRs are stored dynamically, binding memory only when $C_i(k) > 0$, and are updated every time a new reservation is made. Eq. (3) describes that a new reservation for road segment i which starts at time-slot $t_{qi}^d \in \mathbb{N}$ increases $C_i(k)$ by 1 for τ_i consecutive time-slots, where τ_i is the travel time of i .

$$C_i(k) := C_i(k) + 1 : \forall k \in \{t_{qi}^d, \dots, t_{qi}^d + \tau_i - 1\} \quad (3)$$

- *Non-Admissible Time Ranges (NATRs)* [43]: A NATR $\mathcal{R}_{il} \in \mathcal{D}_i \subseteq \mathbb{N}^2$ consists of a 2-tuple $\langle r_{il}^s, r_{il}^e \rangle$, describing that all time-slots $\forall k \in \{r_{il}^s, r_{il}^s + 1, \dots, r_{il}^e\}$ are considered non-admissible for road segment i . Here, \mathcal{D}_i is the set of all NATRs of road segment i , r_{il}^s is the start time-slot, and r_{il}^e is the end time-slot of the l^{th} NATR of road segment i . For efficiency, the set \mathcal{D}_i is stored as a dynamic list of time ranges, and it updates when an ANoR reaches the threshold of its corresponding road segment. Eq. (4) describes that a time-slot $k \in \mathbb{N}$ is considered non-admissible for road segment i if and only if the addition of a new reservation at that time-slot will cause any time-slot of the road's traversal $\tilde{k} \in \{k, \dots, k + \tau_i - 1\}$ to have an ANoR $C_i(\tilde{k})$ that reaches the road's threshold h_i .

$$\begin{aligned} \exists k, l : k \in \mathcal{R}_{il} \in \mathcal{D}_i &\iff \\ \exists \tilde{k} \in \{k, \dots, k + \tau_i - 1\} : C_i(\tilde{k}) &= h_i \end{aligned} \quad (4)$$

Remark 1. Some time-slots $k \in \mathcal{R}_{il}$ can be non-admissible even though road segment i does not explicitly exceed its threshold at time-slot k .

Remark 2. A single request might cause the ANoRs of a road segment to reach its threshold in multiple time-slots, possibly having overlapping 2-tuples of NATRs. If so, the set of NATRs is merged into a sorted set of non-overlapping 2-tuples.

C. Cache Mechanism

The cache mechanism is responsible for storing and updating some *reservation data* locally on the RRC. This allows the

Time-slot k	17	18	19	20	21	22	23	24	25	26	27	28
ANoR $C_i(k)$	3	2	5	5	4	3	5	5	1	3	4	5
Admissible	T	F	F	F	T	F	F	F	T	T	F	F
NATR \mathcal{D}_i				[18, 20], [22, 24], [27, 28]								

Fig. 5. Instance of the reservation data of road segment i with travel time $\tau_i = 2$ and threshold $h_i = 5$. The blue rectangles indicate which reservation data remain in the *Cache* after the *cache window* moves to begin at time-slot $c_i^s = t_{now} = 20$, and end at time-slot $c_i^e = 26$.

RRC to process the local *reservation data* faster, substantially reducing the communication with the DB. Specifically, the cache mechanism creates a cache window for each road segment which holds both the ANoR and NATR for predefined time periods. In contrast, the DB holds only the ANoR of each road segment for all time-slots.

For each road segment $i \in \mathcal{V}$, a sliding *cache window* is assigned to store both the $C_i(k)$ and \mathcal{D}_i for a range of time-slots $k \in \mathbb{N}$. It begins at time-slot c_i^s and ends at time-slot $c_i^e = c_i^s + n$, where n is the *cache size*. Initially, c_i^s is assigned to be the current time-slot t_{now} when the system starts. Periodically, the *cache window* moves to efficiently utilize the memory, increasing c_i^s by m time-slots (i.e., $c_i^s := c_i^s + m$), where m denotes the *cache step*. This happens when the current time-slot t_{now} exceeds $c_i^s + m$. Whenever the *cache window* moves, the *reservation data* for time-slots within the *cache window* (i.e., $\forall k \in \{c_i^s, \dots, c_i^e\}$) remain in the *Cache*, while the *reservation data* outside the *cache window* are considered *out of range* and are handled differently, depending on their type and the value of time-slot they describe, as follows:

- ANoR: $C_i(k)$ for road segment i remains in the cache if $c_i^s \leq k \leq c_i^e$. Otherwise it gets transferred from the cache to the DB.
- NATR: Each $\mathcal{R}_{il} \in \mathcal{D}_i$ for road segment i and some index l remains in the cache if $r_{il}^e \geq c_i^s$, otherwise it is deleted.

As mentioned in Section IV-A, for each iteration of the *inner loop* of the IDA, a path \mathcal{P}_q is calculated, which might include road segments that need some wait time to become admissible. Eq. (5) describes the calculation of the wait time $w_i(k)$ for road segment i to become admissible from time-slot k , depending on whether k belongs to a NATR $\mathcal{R}_{il} \in \mathcal{D}_i$ for some index l . To find the corresponding NATR, the IDA performs a binary search [44] through \mathcal{D}_i .

$$w_i(k) = \begin{cases} r_{il}^e - k + 1, & \exists l : k \in \mathcal{R}_{il} \in \mathcal{D}_i, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

After the RRC chooses the path \mathcal{P}_q^* with no wait time for any road segment $i \in \mathcal{P}_q^*$, the reservation is made by updating $C_i(k), \forall i \in \mathcal{P}_q^*$, as described in Eq. (3). For time-slots not within the *cache window*, the RRC requests the corresponding ANoR for all affected time-slots from the DB, updates and stores them temporarily on the *cache window*, until the next time the *cache window* moves.

Fig. 5 illustrates an example of how the reservation data of a single road segment $i \in \mathcal{V}$ are represented for time-slots $k \in \{18, \dots, 28\}$. The road segment's travel time is $\tau_i = 2$ and its threshold is $h_i = 5$. It is true that $C_i(\bar{k}) = h_i, \forall \bar{k} \in \{19, 20, 23, 24, 28\}$, meaning that the road segment has reached its threshold during these time-slots. As a result, the non-admissible time-slots of that road segment are: $\hat{k} \in \{18, 19, 20, 22, 23, 24, 27, 28\}$, meaning that a new reservation beginning at any of these time-slots will result in i exceeding its threshold at any $\bar{k} \in \{19, 20, 23, 24, 28\}$. The figure also shows that the *cache window* has just moved to begin at $c_i^s = t_{now} = 20$ and end at time $c_i^e = 26$. Therefore, the $C_i(k)$ of time-slots $k \in \{20, \dots, 26\}$ are stored in the *cache window*, and those of $k \in \{17, 18, 19, 27, 28\}$ are stored in the DB. NATR on the other hand are only stored in the *cache window*, and only for time ranges with $r_{il}^e \geq c_i^s$ (i.e., $\mathcal{D}_i = \{[18, 20], [22, 24], [27, 28]\}$).

D. RRC Phases

The RRC is designed to operate in two phases: the *setup phase*, and the *main phase* for allowing clients to connect to it, and for processing their route reservation requests. Fig. 6 shows these operating phases in the form of a flow diagram.

1) *Setup phase*: The setup phase is executed once to initiate the operation of the RRC for a particular city. At this phase, the following parameters need to be defined to load the appropriate network and attributes of the system:

- The *network file*. This describes the intersections, road segments, and the connections between them. It can be acquired from an open source map service (e.g., OpenStreetMap [45]) in XML format, or be generated via corresponding tools.
- The *threshold percentage* H . This will determine how many vehicles can enter each road segment at any instance, as shown in Eq. (1).
- The expected average *free-flow speed* v_f of the vehicles to be traversing the network. This value will be used for estimating the travel time of road segments in the network.
- The *average vehicle length* L (including the minimum gap between vehicles).
- The *time duration of a reservation time-slot* T in seconds.
- The *cache size* n and *cache step* m .

The values of these parameters affect the traffic performance of the system, therefore the optimal values should be used. The value of H can be acquired from the MFD as shown in Eq. (2). The value of v_f can also be acquired from the MFD, as the slope of the flow when the density is slightly above zero. The values of L , T , n , and m , can all be assumed to take the default values of 7.5 m, 1 s, 1 h, and 1 h, respectively.

2) *Main phase*: The main phase utilizes a queue for sequentially processing all client requests, following an FCFS policy. If the request queue is empty, the loop *awaits* for a new reservation request. Otherwise, the RRC proceeds to process the first request by *running* the IDA, to calculate the path \mathcal{P}_q^* and assigned departure time t_q^{d*} using the cached NATRs. Then, the RRC proceeds to make the reservation by *initiating*

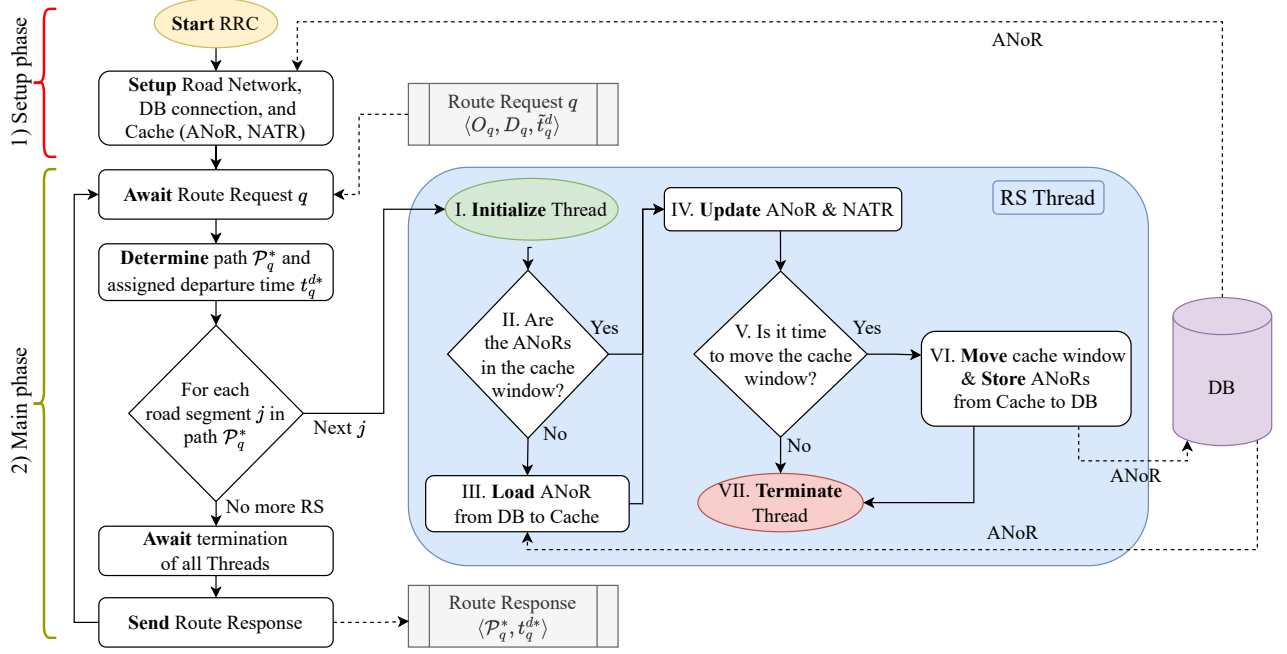


Fig. 6. Flow diagram of the RRC.

a Road Segment Thread (RS Thread) for each Road Segment in the assigned path.

The life-cycle of an RS Thread is outlined in Fig. 6 as the light-shaded rectangle. Each RS Thread is responsible for managing the reservation data in the *cache window* of its corresponding road segment to make a reservation. It *checks* if any time-slot k that describes the traversal of road segment i , has its corresponding ANoR $C_i(k)$ missing from the cache window, and *loads* them from the DB. It then makes the reservation by *updating* the corresponding ANoR and NATR in the cache window, as described in Eq. (3), and Eq. (4), respectively. If it is time to move the cache window, it stores the appropriate ANoR to the DB. Due to reservation data being independent for each road segment, all RS Threads of a single request can run in *parallel* with no concurrency issues, substantially reducing computation times.

Meantime, the RRC *awaits* for the termination of all RS Threads, to *send* a route response to the client that issued the request, containing the resulted path \mathcal{P}_q^* and assigned departure time t_q^{d*} . Finally, it loops back to *awaiting* new route reservation requests in the request queue.

V. COMPUTATIONAL COMPLEXITY EVALUATION

To assess our system in terms of complexity and scalability, we conducted various experiments and compared their average computation time, considering urban areas from both artificially generated, and real urban traffic networks. For the average computation time, we measured the time the RRC needs to process a route reservation request, make the reservation (either on the Cache or the DB), and then prepare the response to send to the user. However, we did not measure the time that the route reservation request needs to reach the RRC from the client, neither the time that the response needs

to reach the client from the RRC. Such communication delays are out of the scope of this work, as they pose a nearly constant overhead to each request and response, whose latency relies heavily on the reliability of the connection between the client and the RRC.

The descriptions of the considered networks are shown in Table III, where they are distinguished into two groups: a) artificial b) real. For each network \mathcal{G} , the table shows the number of junctions, the number of its road segments as vertices \mathcal{V} , the number of the connections between them as edges \mathcal{E} , its total length in km, and its total area in km^2 . The artificial networks were generated through tools provided by the Simulation of Urban MObility (SUMO) library [46], while the realistic networks were taken from parts of real cities through OpenStreetMap [45]. The network *SF* is part of San Francisco, USA. *Nic*, *Nic_2*, *Nic_3*, and *Nic_4* are subsets of the road network of the city of Nicosia, Cyprus. *ITSC*, and *ITSC_2* [47] are parts of the city of Dublin, Ireland, *MoST* [48] is the city of Monaco, Monaco, and *InTAS* [49] is the city of Ingolstadt, Germany.

The proposed implementation methods (i.e., the aggregation of ANoR into NATR, the cache, the DB, the utilization of both the cache and DB using RS threads, and the versioning scheme of the cache) are crucial for the proposed system, as without them, response times would be high enough to render the system impractical for real-life use. Along with the proposed implementation methods, the system runs the IDA for the EATD algorithm, for which two different adaptations are compared (i.e., Dijkstra using Fibonacci Heap (DFH), and Uniform-Cost Search using Fibonacci Heap (UCSFH)). We evaluated both adaptations, running experiments with duration equal to 3 h with different demand flows of OD pairs distributed uniformly in space, ranging between 5000–35 000 veh/h with an increment step of 5000 veh/h. Although such high demand

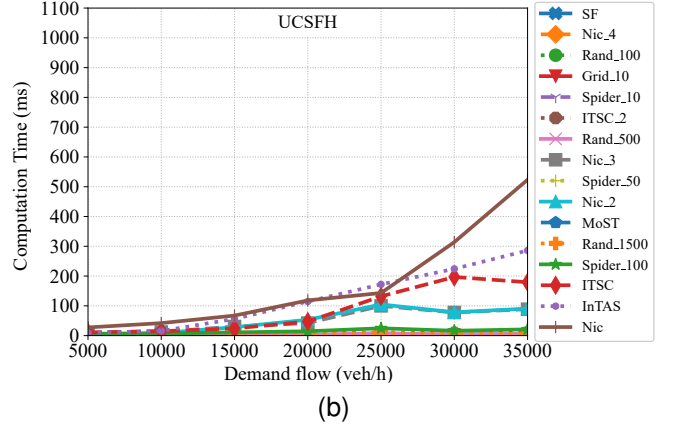
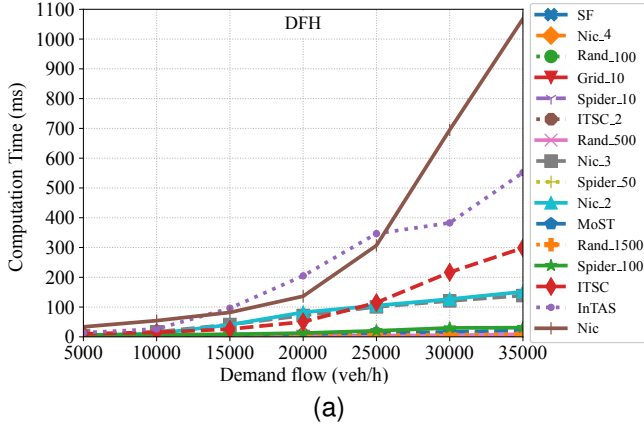


Fig. 7. Average computation time in ms running on different implementations of the IDA: (a) DFH, (b) UCSFH.

TABLE III
SAMPLE ROAD NETWORKS

Type	Network \mathcal{G}	Junctions	\mathcal{V}	\mathcal{E}	Len. (km)	Area (km ²)
Artif.	<i>Rand_100</i>	100	280	882	45	2.5
	<i>Grid_10</i>	100	360	1320	355	81
	<i>Spider_10</i>	130	494	1898	84	6
	<i>Rand_500</i>	500	1476	4866	237	13
	<i>Spider_50</i>	650	2574	10218	1674	30
	<i>Rand_1500</i>	1500	4576	15432	741	38
	<i>Spider_100</i>	1300	5174	20618	6462	61.5
Real	<i>SF</i>	129	203	355	8	1
	<i>Nic_4</i>	148	235	544	70	5.5
	<i>ITSC_2</i>	531	1236	2378	71	4
	<i>Nic_3</i>	1094	1872	3219	120	4.5
	<i>Nic_2</i>	1788	2867	4934	250	10
	<i>MoST</i>	2004	3615	5842	470	23.5
	<i>ITSC</i>	2895	6596	12360	483	30
	<i>InTAS</i>	3332	7726	13636	710	27.5
	<i>Nic</i>	8571	16161	42990	1177	46

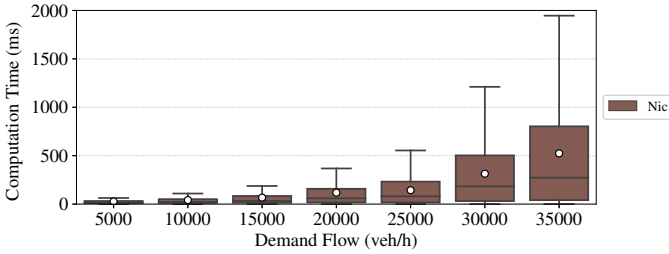


Fig. 8. Computation time box-plots over demand flow for UCSFH. The white circles indicate the mean for each flow. The boxes show the second and third quartiles of the Origin Delays, horizontally split by the median line. The whiskers show the rest of the distribution, from 5% and up to 95%, while outliers further than that are not shown.

flows might be unrealistic for certain networks due to their sizes, they serve the purpose of stressing the performance and capabilities of our system. The time-length of reservation time-slots was set to $T = 1$ s. Vehicles were assumed to travel at free-flow speed $v_f = 32.5$ km/h, while having the same length $L = 7.5$ m (5 m for the vehicle and 2.5 m for the minimum gap between vehicles). The threshold percentage of each road segment was set to $H = 25\%$. The cache size n , and cache step m were both set to 1 h.

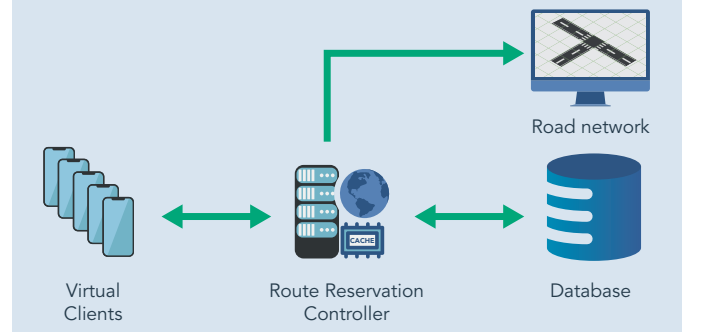


Fig. 9. System diagram with SUMO [46].

Figures 7a and 7b show the average computation time per request over demand flows ranging from 5000 veh/h to 35 000 veh/h for all considered networks, implementing the IDA with the DFH, and UCSFH adaptations. The networks are sorted ascending by their number of vertices. It is observed that for the lowest flows of 5000 veh/h, both implementations have similar computation times. For medium flows of 20 000 veh/h, the computation times are slightly larger, with the UCSFH being 20% – 40% faster than the DFH. Finally, for the highest flows of 35 000 veh/h, the computation times are much higher, with the UCSFH implementation being faster by 33 – 51%.

Figure 8 shows the box-plot quartiles of the computation time of requests over demand flows ranging from 5000 veh/h to 35 000 veh/h for the *Nic* network, which showed the largest computation times of all examined networks. It can be observed that even for the highest demand flow, the response time for more than 75% of requests is under 1 s, for up to 50% of the requests it is under 0.3 s, while there is a 25% of the requests that got their response between 0.8 s and 2 s.

VI. TRAFFIC PERFORMANCE EVALUATION

The traffic performance of our system is important for assessing how well it solves the traffic congestion problem in a real road network. We performed experiments on a real-life calibrated network, where drivers were simulated using *Simulation of Urban MObility (SUMO) Simulator* [46], a microscopic, space-continuous, and time-discrete traffic flow simulator.

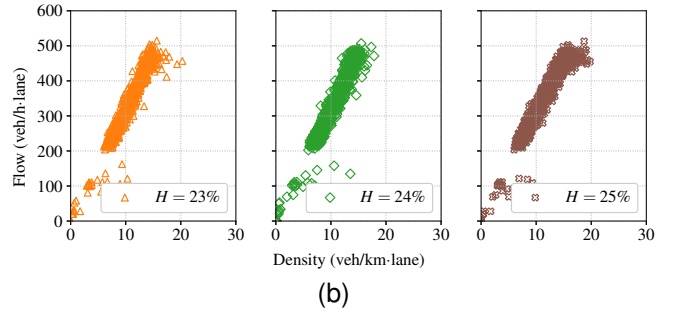
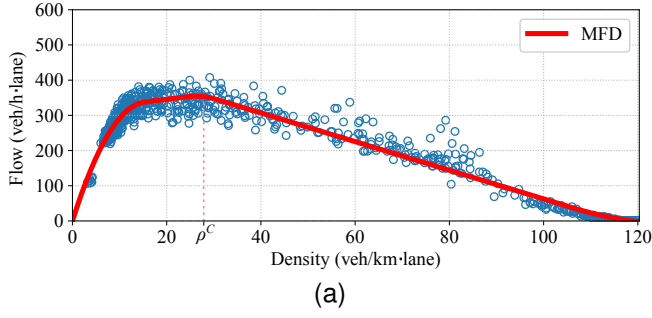


Fig. 10. Average flow over average density of all road segments of the ten experiments with different OD pairs: (a) Uncontrolled, (b) RRC for threshold percentages of 23%, 24%, and 25%.

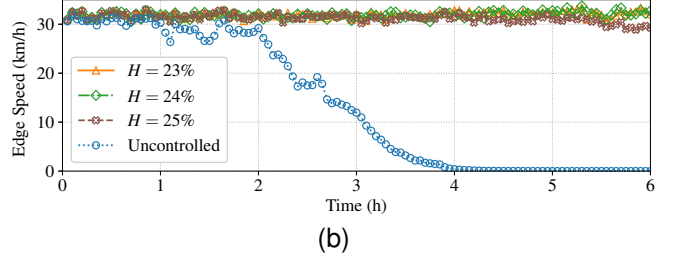
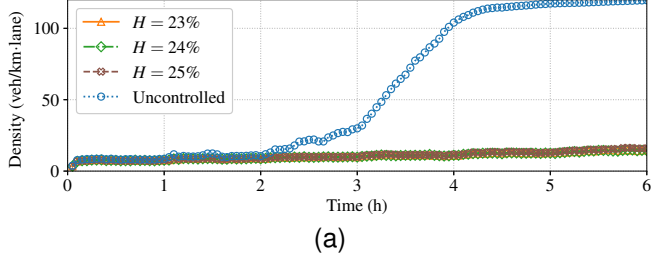


Fig. 11. Average measurements over time for the ten OD pairs, comparing the Uncontrolled experiments versus the RRC for threshold percentages of 23%, 24%, and 25%: (a) Density, (b) Speed.

Fig. 9 depicts how our system is set up for this evaluation. Virtual clients are generated to send real route requests to the RRC, which finds the appropriate route and actual departure time for each request. When sending the response back to the clients, it also informs the simulator, using a bridge connection formed by the *TraCI* library, provided by SUMO. *TraCI* stands for *Traffic Control Interface*, and gives access to properties of a running *SUMO Simulation*, allowing the retrieval of values of simulated objects [50], [51].

For the simulation, we used the network of Aglantzia, an urban area in the city of Nicosia, Cyprus, with 870 intersections, 1930 road segments as vertices, 3762 connections between them as edges, and a total length of 162 km within an area of about 6.5 km². The network includes 11 traffic light controlled intersections with signal schedules automatically generated by SUMO, while drivers were modeled according to the Krauss' car-following model [52]. We assumed the time-length of a time-slot to be $T = 1$ s. For setting the rest of the system's parameters, as described in Section IV-D, we acquired the MFD of the network, which provided the values for the capacity threshold percentage H^* , and the free-flow speed v_f . Then we used these parameters to assess our system under varying threshold percentages near the capacity threshold, and varying user compliance percentages.

A. MFD Extraction

To extract the MFD of the network, we generated ten scenarios with different OD pairs distributed uniformly in space. The same scenarios were used for two types of experiments: 1) The Uncontrolled experiments, where vehicles were following the shortest path to their desired departure time, 2) The RRC experiments with different values of H , where vehicles were following the instructions of the RRC in terms of actual departure time and route to follow. The

duration of each experiment was 6 h. For the first hour, the flow was 5000 veh/h, and was increasing by 1000 veh/h each hour, reaching a flow of 10 000 veh/h at the sixth hour.

Fig. 10a depicts the average internal-flow over density per lane of the network for all ten Uncontrolled experiments, forming an MFD, consistent with the ones acquired in [53] for other cities. Every point in the figure depicts a 3 min interval measurement. The solid red line indicates the MFD, which allows us to obtain the critical density $\rho^C = 28$ veh/km, where the flow is maximum, and the jam density of the network $\rho^J = 120$ veh/km. This yields the capacity threshold percentage of the network as described in Eq. (2), $H^* = 23\%$. The free-flow speed of the network can be acquired as the slope at the beginning of the MFD, which is $v_f = 32.5$ km/h.

Fig. 10b shows that the maximum flows of the RRC experiments yield a 37% improvement from the Uncontrolled scenarios, reaching 480 veh/h compared to 350 veh/h.

Figures 11a and 11b illustrate respectively the density and edge speed over time, averaged over the ten different scenarios for the Uncontrolled and RRC experiments for the aforementioned values of H . These figures show that as the flow increases for the Uncontrolled experiments, the density grows larger, reaching the jam density $\rho^J = 120$ veh/km, while the average edge speed goes to 0 km/h. Contrary, as the flow of the RRC experiments for the different values of H increases, their density remains below 20 veh/km, and their average edge speed fluctuates around 32 km/h, indicating no congestion.

B. Varying threshold percentage

To further assess our system's efficiency, we ran the Uncontrolled experiments and RRC with different values of H , to study its behavior under low, near the critical density, and high threshold percentages. As the capacity threshold

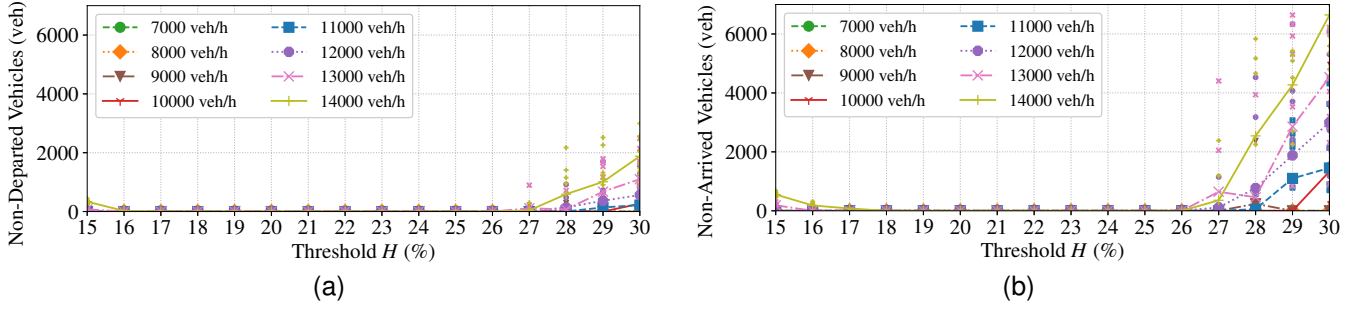


Fig. 12. Average number of vehicles for different flows over threshold percentage that have: (a) Departed in the network, (b) Arrived at their destination. The scattered smaller points depict the corresponding number of vehicles of each simulation separately.

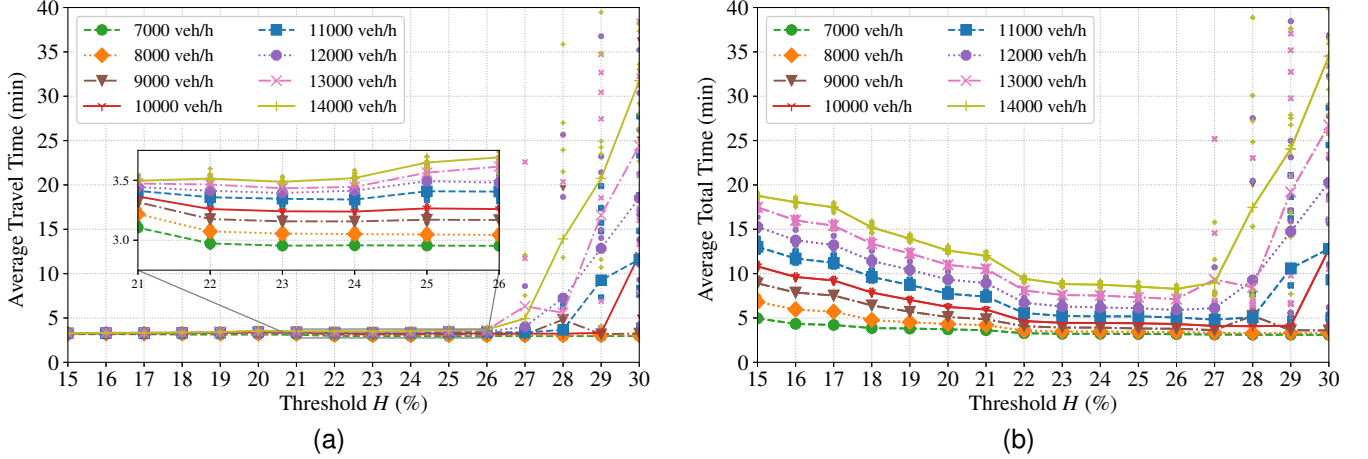


Fig. 13. Average time for different flows over threshold percentage for: (a) Travel Time (TT), (b) Total Time (ToT). The scattered smaller points depict the mean time of each simulation separately.

percentage is $H^* = 23\%$, see Section VI-A, we experimented with H ranging between $15\% - 30\%$ with 1% increments. For each type of experiment (RRC with different H , and Uncontrolled), we ran the same ten scenarios with different OD pairs distributed uniformly in space, spanning across 1 h, for different flows of OD pairs distributed uniformly in space, ranging between $7000 - 14000$ veh/h with 1000 veh/h increment. Note that although the demand spans over 1 h, the simulation duration was set to 2 h to allow the network to clear completely. For comparison purposes, we also ran the Dynamic Rerouting (DR) algorithm provided by SUMO, to compare our results. This technique allows all vehicles to continuously run a rerouting algorithm to find in real-time the path with the lowest estimated travel time, considering the current state of congestion. When a vehicle intends to follow a path which is expected to be congested, a new path is assigned to that vehicle, on-the-go, avoiding any congested roads.

Figures 12a and 12b illustrate the average number of vehicles that have not managed to depart from their origin (non-departed), and the average number of vehicles that have not managed to arrive at their destination (non-arrived) respectively, for each flow, over different H . It can be observed that lower values of H over-restricted the network by giving large initial waiting times, therefore scheduling some vehicles to depart after the end of the simulation. On the other hand, higher values of H under-restricted the network, allowing an excessive number of vehicles to depart, resulting in congestion, which forced the simulator to physically not be able to

allow further vehicles to depart. With the same reasoning, the number of non-arrived vehicles was analogous, reaching up to $200 - 500$ vehicles for low values of H , and around $1000 - 6000$ vehicles for high values of H . However, for values of H around the capacity threshold (i.e., $18\% - 26\%$), all vehicles departed and arrived at their destinations successfully, before the end of the simulation, indicating that the network was congestion-free.

Figures 13a and 13b illustrate the average Travel Time (TT) and the average Total Time (ToT) respectively, for each flow, over different H . Note that ToT incorporates the travel times and all delays that occur between the desired departure and arrival of a vehicle, including any waiting delay at the origin.

As observed in Fig. 13a, TT remains low for all flows, reaching up to 3.8 min for H being low and near the capacity threshold (i.e., $15\% - 26\%$), indicating that there is no congestion in the network. Higher values of H yield TT values up to 35 min, due to the occurrence of congestion.

In Fig. 13b, a plateau can be observed for H near the capacity threshold (i.e., $22\% - 26\%$), where ToT is minimum for all flows. For the highest flow, the minimum ToT value is around $8 - 9$ min, and grows up to 19 min as H reduces to 15% , and up to 35 min as it increases to 30% . A similar trend is followed by all flows, except the three lowest, which keep reducing their ToT values as H increases. It is important to note that the plateau indicates that the selected H may yield high quality results even if the capacity threshold percentage is not accurately identified.

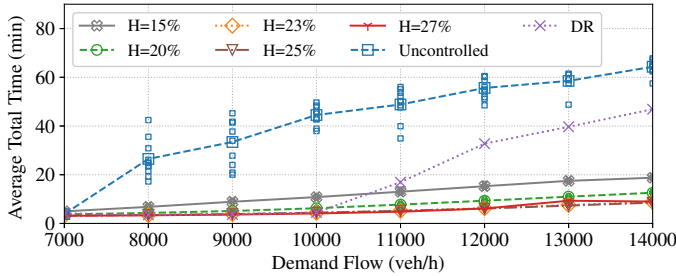


Fig. 14. Average Total Time (ToT) over demand for the Uncontrolled, the DR, and the RRC experiments with threshold percentages of 15%, 20%, 23%, 25%, and 27%. The smaller scattered markers depict the mean ToT of each scenario separately.

Fig. 14 shows the average ToT over demand for the Uncontrolled, the DR, and the RRC experiments with threshold percentages of 15%, 20%, 23%, 25%, and 27%. It shows that as the demand flow increases, the ToT of the Uncontrolled experiments becomes much larger than the corresponding RRC experiments. It also shows that as the threshold percentage approaches values near the capacity threshold, ToT decreases, with up to 80% improvement from the Uncontrolled ToT. Regarding DR, the plot shows that for low demand flows it has similar average ToT with the RRC experiments of $H = 23\%$. However, for high demands, the average ToT is significantly higher, since rerouting alone does not control the capacity of the network. As a result, the network becomes overpopulated, leaving no alternative routes that do not suffer from congestion.

Fig. 15a and Fig. 15b illustrate the Origin Delays, and Extra Travel Distance Percentage of clients, respectively, as box-plot quartiles over flows, for the RRC experiments with the capacity threshold percentage $H = 23\%$.

Fig. 15a indicates that even for high demand flows, the first three quartiles are relatively low, suggesting that the majority of clients are given manageable origin delays, while the fourth quartiles are notably larger. However, clients can possibly use origin delays productively, in contrast to delays in the Uncontrolled experiments, which are not at the origin.

Fig. 15b indicates that even for high demand flows, almost up to 75% of clients have had extra travel distances less than 10% of their corresponding shortest path, while the rest have had 10 – 60%. For comparison, the travel distances for the Uncontrolled experiments range between 400 – 3300 m, averaging at 1750 m. These results show that the extra travel distance for most clients is negligible, while for the rest it is still a small burden, compared to the benefits it gives.

C. Varying compliance

In this subsection we run the same experiments with different percentages of compliance, to test the traffic efficiency of our system when not all clients follow their assigned path and departure time. Specifically, we experimented for values of H between 18% – 25% at a medium flow of 10 000 veh/h, and a high flow of 14 000 veh/h, with 70%, 80%, 90%, and 100% compliance. The percentage of compliance indicates the percentage of clients that followed their assigned path and departure time, while the rest followed the shortest path to the

destination at their desired departure time (i.e., no waiting at the Origin).

Figures 16a and 16b illustrate the average Total Time (ToT) for the medium and high flows, respectively. It can be observed in both figures that when H is lower than the capacity threshold, compliance of 90% can be slightly beneficial to the system's average ToT, compared to full compliance. Compliance of 80% can also be beneficial for the medium flow, or even for the high flow with an even smaller threshold. For H near the capacity threshold, compliance of 90% has similar behavior as full compliance, while compliance of 80% makes the system have traffic oscillations, thus higher ToT. Compliance of 70% has higher ToT for all values of H . Nevertheless, for low enough H , the ToT is lower for all tested compliance rates, compared to the uncontrolled experiments for both medium and high flows. Regarding DR, it is shown that for the medium demand flow, the average ToT is almost identical to the average ToT of compliance percentages 80% – 100%. However, for the high demand scenarios, the average ToT of DR is significantly higher, even for the lowest compliance percentages. This demonstrates the ability of the RRC to perform well even in high demand scenarios.

These results indicate that with medium to high flows, a threshold lower than the capacity threshold can be beneficial when there is a small percentage of non-compliant commuters. However, full compliance will yield better results when the threshold is near the capacity threshold.

VII. CONCLUSIONS AND FUTURE WORK

In this work we designed a system as a scalable tool for joint traffic and demand management in large-scale real urban area networks, using a reservation scheme. Specifically, we introduced the three entities of the system and their roles, which are: 1) the Clients, 2) the Server (i.e., RRC), and 3) The DB. We also defined a *Communication Protocol* for describing the interactions between the entities, and how they handle reservation data. We achieved scalability by using appropriate data structures, algorithms, and optimization techniques to reduce response times, including a *Cache Mechanism* which allows the RRC to temporarily store some reservation data locally for immediate access.

We evaluated the complexity of our system showing that it can handle well reservations for both artificial and real networks, with response times under 100 ms in most cases, even during high demand.

We also evaluated the traffic performance of our system in terms of how well it copes with the traffic congestion problem in realistic networks, showing that our system allows significantly large numbers of clients to utilize the network, without causing congestion at any time. It is important to note that even if the capacity threshold parameter of the network is not accurately identified, our system can still yield high quality results. This indicates that the threshold and speed parameters can be approximated. Although the MFD acts as a mechanism to indicate good values for these parameters, future works may include different approaches for acquiring the required parameters, or even determining different parameters

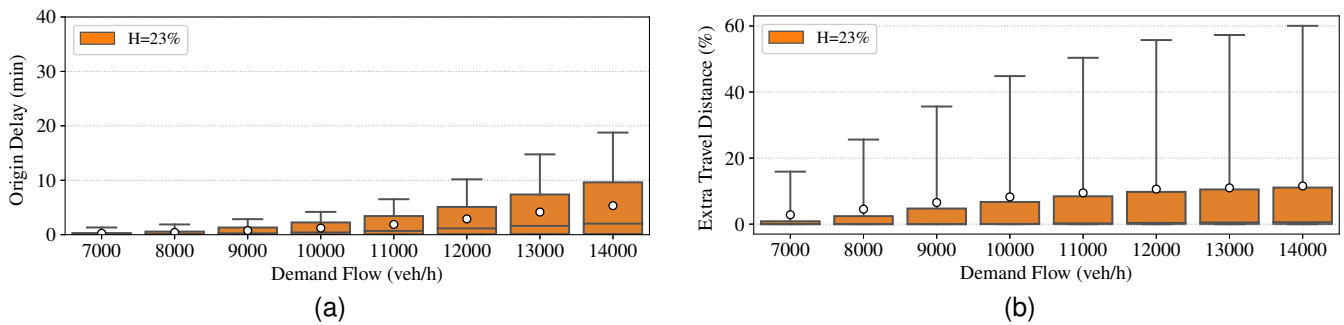


Fig. 15. Box-plot quartiles of: (a) Origin Delays, (b) Extra Travel Distance, over flow for capacity threshold percentage $H = 23\%$. The white circles indicate the mean for each flow. The boxes show the second and third quartiles of the Origin Delays, horizontally split by the median line. The whiskers show the rest of the distribution, from 5% and up to 95%, while outliers further than that are not shown.

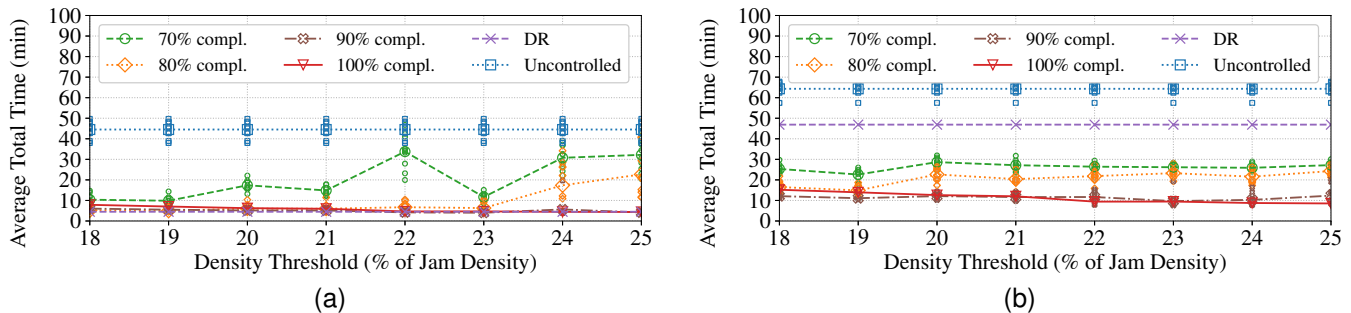


Fig. 16. Average Total Time (ToT) for different compliance over threshold percentages: (a) 10 000 veh/h flow, (b) 14 000 veh/h flow. The scattered smaller points depict the mean time of each simulation separately.

for different road segments. Moreover, we showed that a small percentage of non-compliant commuters can be beneficial to the system if the system's parameters are more restrictive.

Future work may also include obtaining the critical threshold percentage for each road segment individually rather than approximating a system-wide threshold. Moreover, there is room for experimenting with a multi-modal extension of the system, combining various means of transport, including public transport [54]. This could be done by making the default system parameters dynamic. Also, the experimental results show a relation between compliance percentages and threshold percentages, opening way for the development of a method for dynamically adapting the threshold percentages based on the expected compliance rates. Another interesting view is to investigate the different priority policies for serving requests. This could give priority to HOV, or users which are usually belated when requesting a route reservation. Furthermore, the system behavior should be studied when allowing users to choose between late departure or early arrival, or even when introducing a pricing mechanism for handling deviation of drivers over the proposed routes and departure times. Finally, an incident response study would be beneficial for our system, enabling it to cope with accidents and unexpected road closures.

REFERENCES

- [1] H. Rehborn, S. L. Klenov, and J. Palmer, "An empirical study of common traffic congestion features based on traffic data measured in the usa, the uk, and germany," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 23-24, pp. 4466–4485, 2011.
- [2] J. I. Levy, J. J. Buonocore, and K. Von Stackelberg, "Evaluation of the public health impacts of traffic congestion: a health risk assessment," *Environmental health*, vol. 9, no. 1, pp. 1–12, 2010.
- [3] "Google maps." [Online]. Available: <http://www.google.com/maps>
- [4] "Waze." [Online]. Available: <http://www.waze.com/>
- [5] "Here wego." [Online]. Available: <https://wego.here.com/>
- [6] S. Çolak, A. Lima, and M. C. González, "Understanding congested travel in urban areas," *Nature communications*, vol. 7, no. 1, pp. 1–8, 2016.
- [7] T. Roughgarden and É. Tardos, "How bad is selfish routing?" *Journal of the ACM (JACM)*, vol. 49, no. 2, pp. 236–259, 2002.
- [8] C. Menelaou, P. Kolios, S. Timotheou, and C. G. Panayiotou, "On the complexity of congestion free routing in transportation networks," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 2819–2824.
- [9] R. A. Doherty and P. Sorenson, "Keeping users in the flow: mapping system responsiveness with user experience," *Procedia Manufacturing*, vol. 3, pp. 4384–4391, 2015.
- [10] C. Chen, Z. Jia, and P. Varaiya, "Causes and cures of highway congestion," *IEEE Control Systems Magazine*, vol. 21, no. 6, pp. 26–32, 2001.
- [11] K. Aboudolas and N. Geroliminis, "Perimeter and boundary flow control in multi-reservoir heterogeneous networks," *Transportation Research Part B: Methodological*, vol. 55, pp. 265–281, 2013.
- [12] A. Mazloumian, N. Geroliminis, and D. 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*, vol. 368, no. 1928, pp. 4627–4647, 2010.
- [13] C. F. Daganzo, "Urban gridlock: Macroscopic modeling and mitigation approaches," *Transportation Research Part B: Methodological*, vol. 41, no. 1, pp. 49–62, 2007.
- [14] Q. K. Pu and H. H. Li, "Mobile navigation system," Sep. 18 2001, uS Patent 6,292,743.
- [15] M. Khanjary and S. M. Hashemi, "Route guidance systems: Review and classification," in *2012 6th Euro American Conference on Telematics and Information Systems (EATIS)*. IEEE, 2012, pp. 1–7.
- [16] S. Vasserman, M. Feldman, and A. Hassidim, "Implementing the wisdom of waze," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
- [17] E. Demers and G. Yemen, "Introduction to consolidation accounting: Google's acquisition of waze," 2015.
- [18] S. van der Graaf, "In waze we trust: algorithmic governance of the public sphere," *Media and Communication*, vol. 6, no. 4, pp. 153–162, 2018.
- [19] D. Braess, A. Nagurny, and T. Wakolbinger, "On a paradox of traffic planning," *Transportation science*, vol. 39, no. 4, pp. 446–450, 2005.

- [20] M. Mahmood, M. A. Bashir, S. Akhter *et al.*, "Traffic management system and travel demand management (tdm) strategies: suggestions for urban cities in bangladesh," *Asian Journal of Management and Humanity Sciences*, vol. 4, no. 2-3, pp. 161–178, 2009.
- [21] B. Schlag and J. Schade, "Public acceptability of traffic demand management in europe," *Traffic Engineering+ Control*, vol. 41, no. 8, pp. 314–18, 2000.
- [22] P. Kachroo, S. Gupta, S. Agarwal, and K. Ozbay, "Optimal control for congestion pricing: Theory, simulation, and evaluation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1234–1240, 2016.
- [23] D. Wu, Y. Yin, S. Lawphongpanich, and H. Yang, "Design of more equitable congestion pricing and tradable credit schemes for multimodal transportation networks," *Transportation Research Part B: Methodological*, vol. 46, no. 9, pp. 1273–1287, 2012.
- [24] J. Pfrommer, J. Warrington, G. Schildbach, and M. Morari, "Dynamic vehicle redistribution and online price incentives in shared mobility systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1567–1578, 2014.
- [25] J. Dahlgren, "High occupancy vehicle lanes: Not always more effective than general purpose lanes," *Transportation Research Part A: Policy and Practice*, vol. 32, no. 2, pp. 99–114, 1998.
- [26] A. M. Brewer, "Work design, flexible work arrangements and travel behaviour: policy implications," *Transport Policy*, vol. 5, no. 2, pp. 93–101, 1998.
- [27] B. P. Loo and Z. Huang, "Spatio-temporal variations of traffic congestion under work from home (wfh) arrangements: Lessons learned from covid-19," *Cities*, vol. 124, p. 103610, 2022.
- [28] C. G. Panayiotou and C. G. Cassandras, "A sample path approach for solving the ground-holding policy problem in air traffic control," *IEEE Transactions on control systems technology*, vol. 9, no. 3, pp. 510–523, 2001.
- [29] C. Menelaou, S. Timotheou, P. Kolios, and C. G. Panayiotou, "A linear formulation of the on-time arrival problem in multi-regional networks," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 3589–3594.
- [30] —, "Convexification approaches for regional route guidance and demand management with generalized mfd," *Transportation Research Part C: Emerging Technologies*, vol. 154, p. 104245, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0968090X23002346>
- [31] C. Menelaou, P. Kolios, S. Timotheou, C. G. Panayiotou, and M. Polycarpou, "Controlling road congestion via a low-complexity route reservation approach," *Transportation Research part C: Emerging technologies*, vol. 81, pp. 118–136, 2017.
- [32] Y. Modi, R. Teli, A. Mehta, K. Shah, and M. Shah, "A comprehensive review on intelligent traffic management using machine learning algorithms," *Innovative infrastructure solutions*, vol. 7, no. 1, p. 128, 2022.
- [33] Z. Yin, T. Liu, C. Wang, H. Wang, and Z.-P. Jiang, "Reducing urban traffic congestion using deep learning and model predictive control," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2023.
- [34] Z. Liu and R. Stern, "Quantifying the traffic impacts of the covid-19 shutdown," *Journal of Transportation Engineering, Part A: Systems*, vol. 147, no. 5, p. 04021014, 2021. [Online]. Available: <https://ascelibrary.org/doi/abs/10.1061/JTEPBS.0000527>
- [35] Google, "Android," 2008. [Online]. Available: <https://developer.android.com/>
- [36] J. Anez, T. De La Barra, and B. Pérez, "Dual graph representation of transport networks," *Transportation Research Part B: Methodological*, vol. 30, no. 3, pp. 209–216, 1996.
- [37] C. F. Daganzo and N. Geroliminis, "An analytical approximation for the macroscopic fundamental diagram of urban traffic," *Transportation Research Part B: Methodological*, vol. 42, no. 9, pp. 771–781, 2008. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0191261508000799>
- [38] S. N. Z. Naqvi, S. Yfantidou, and E. Zimányi, "Time series databases and influxdb," *Studienarbeit, Université Libre de Bruxelles*, 2017.
- [39] C. Menelaou, P. Kolios, S. Timotheou, and C. G. Panayiotou, "Congestion free vehicle scheduling using a route reservation strategy," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 2103–2108.
- [40] A. Orda and R. Rom, "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length," *Journal of the ACM (JACM)*, vol. 37, no. 3, pp. 607–625, 1990.
- [41] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [42] B. J. H. Verwer, P. W. Verbeek, and S. T. Dekker, "An efficient uniform cost algorithm applied to distance transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 4, pp. 425–429, 1989.
- [43] C. Menelaou, S. Timotheou, P. Kolios, C. G. Panayiotou, and M. M. Polycarpou, "Optimal path selection in a continuous-time route reservation architecture," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.
- [44] R. Nowak, "Generalized binary search," in *2008 46th Annual Allerton Conference on Communication, Control, and Computing*. IEEE, 2008, pp. 568–574.
- [45] OpenStreetMap contributors, "Planet dump retrieved from <https://planet.osm.org>," <https://www.openstreetmap.org>, 2017.
- [46] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: <https://elib.dlr.de/124092/>
- [47] M. Gueriau and I. Dusparic, "Quantifying the impact of connected and autonomous vehicles on traffic efficiency and safety in mixed traffic," in *23rd IEEE International Conference on Intelligent Transportation Systems*, 2020.
- [48] L. Codeca and J. Härri, "Monaco sumo traffic (most) scenario: A 3d mobility scenario for cooperative its," *EPiC Series in Engineering*, vol. 2, pp. 43–55, 2018.
- [49] S. C. Lobo, S. Neumeier, E. M. Fernandez, and C. Facchi, "Intas—the ingolstadt traffic scenario for sumo," *arXiv preprint arXiv:2011.11995*, 2020.
- [50] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "Traci: an interface for coupling road traffic and network simulators," in *Proceedings of the 11th communications and networking simulation symposium*, 2008, pp. 155–163.
- [51] A. F. Acosta, J. E. Espinosa, and J. Espinosa, "Traci4matlab: Enabling the integration of the sumo road traffic simulator and matlab® through a software re-engineering process," in *Modeling Mobility with Open Data*. Springer, 2015, pp. 155–170.
- [52] S. Krauß, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," *Physical Review E*, vol. 55, no. 5, p. 5597, 1997.
- [53] E. J. Gonzales, C. Chavis, Y. Li, and C. F. Daganzo, "Multimodal transport modeling for nairobi, kenya: insights and recommendations with an evidence-based model," 2009.
- [54] H. Liu, J. Han, Y. Fu, J. Zhou, X. Lu, and H. Xiong, "Multi-modal transportation recommendation with unified route representation learning," *Proceedings of the VLDB Endowment*, vol. 14, no. 3, pp. 342–350, 2020.

ABOUT THE AUTHORS



Christos Makridis (cmakri07@ucy.ac.cy) earned his M.Sc. degree in Intelligent Critical Infrastructure Systems at the Department of Electrical and Computer Engineering (ECE) at the University of Cyprus. He is currently pursuing a Ph.D. degree at the ECE Department of the same University and works as a researcher at the KIOS CoE with main focus on Intelligent Transportation Systems.



Charalambos Menelaou (cmenel02@ucy.ac.cy) holds a PhD degree from the ECE Department of University of Cyprus. He is a Researcher Associate at KIOS CoE. His research interests focus on control and optimization of large-scale urban networks. Dr. Menelaou is a member of the IEEE Intelligent Transportation and IEEE Vehicular Technology Societies, since 2014.



Stelios Timotheou (timotheou.stelios@ucy.ac.cy) holds a PhD in Intelligent Systems and Networks (2010), from the EEE Department of Imperial College London. He is an Associate Professor at the ECE Department and a faculty member at the KIOS CoE, of the University of Cyprus. His research focuses on monitoring, control and optimization of critical infrastructure systems, with emphasis on intelligent transportation systems and communication systems.



Christos G. Panayiotou (christosp@ucy.ac.cy) holds a Ph.D. degree in ECE from the University of Massachusetts at Amherst (1999). He is a Professor at the ECE Department at the University of Cyprus. He is also the Deputy Director of the KIOS CoE for which he is also a founding member. His research interests include distributed and intelligent control systems, wireless, ad hoc and sensor networks, computer communication networks, fault diagnosis, optimization and control of discrete-event systems, resource allocation, transportation networks

and intelligent buildings.