Optimal Access Management for Cooperative Intersection Control

Alejandro Ivan Morales Medina, Falco Creemers, Erjen Lefeber, and Nathan van de Wouw

***Abstract*— This paper presents an intersection access management methodology that optimizes the crossing sequence of an automated intersection, where the low-level vehicle control is performed by a cooperative intersection control (CIC) strategy. While the CIC regulates the safe and efficient relative motion of vehicles in the intersection, a high-level hybrid queuing model is proposed to describe the dynamics of the vehicle queues associated to each intersection lane. This model, including constraints, is used to design an optimal access management approach based on the model predictive control that minimizes the time that the vehicles spend within the intersection, thereby optimizing the traffic throughput of the intersection. The performance of this methodology is studied by means of two representative examples. The impact of the design parameters of the optimal access management approach is shown for a T-intersection case study. Moreover, using a real-life five lane intersection case study, the proposed approach is compared to a vehicle-actuated traffic light approach, and a first come first served approach. The comparison shows the benefits of the automated optimal serving of vehicles from different lanes.**

***Index Terms*— Optimal intersection management, cooperative intersection control, hybrid dynamical queuing system, model predictive control, mixed-integer linear programming.**

I. Introduction

ONE of the impacts of the continuous increase of population in urban areas, which according to [1] will be a total of 66% of the world's population by 2050, is the saturation of the transportation network of the cities of the world. The congestion experienced in everyday traffic is caused by a variety of factors, one of which is the interruption of the traffic flow at road intersections [2]. Classical systems used to control road intersections (namely traffic lights, roundabouts, and stop signs) have fallen short to account for the irregularities of the congested traffic flow. It is worth noting that these classical systems also introduce other problems such as excessive vehicle idling, unbalanced waiting times, and potential collisions.

Manuscript received June 20, 2018; revised December 11, 2018; accepted April 8, 2019. Date of publication May 2, 2019; date of current version May 1, 2020. This work was supported in part by the European Union Seventh Framework Programme [FP7-ICT-2013-10] under Grant 612035 and in part by the Interoperable GCDC (Grand Cooperative Driving Challenge) AutoMation Experience (i-GAME). The Associate Editor for this paper was J. Haddad. *(Corresponding author: Alejandro Ivan Morales Medina.)*

A. I. Morales Medina, F. Creemers, and E. Lefeber are with the Department of Mechanical Engineering, Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands (e-mail: a.i.morales.medina@tue.nl; f.m.g.creemers@alumnus.tue.nl; a.a.j.lefeber@tue.nl).

N. van de Wouw is with the Department of Mechanical Engineering, Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands, also with the Civil, Environmental and Geo-Engineering Department, University of Minnesota, Minneapolis, MN 55455 USA, and also with the Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: n.v.d.wouw@tue.nl).

Digital Object Identifier 10.1109/TITS.2019.2913589

The field of *Intelligent Transportation Systems* has risen (aided by technological advances on sensing, communication, and automation) as response to the increase in complexity of transportation networks [3]. Under the category of *Cooperative Intersection Management* we find a range of solutions for the road intersection problem [4], which rely on the ability of vehicles to communicate with other vehicles (V2V communication), and with the infrastructure (V2I communication). The solutions fall in two main categories, namely Resource Allocation (RA) and Trajectory Planning (TP). Note that both categories include centralized and distributed instances. The main difference between the RA and TP approaches is the granularity of the representation of the space-time of the intersection. The RA approach defines the intersection as a set of space tiles that can be allocated in time, by a scheduler that avoids conflicts, to a vehicle that requests access to the intersection; examples are found in [5]–[8]. On the other hand, the TP approaches consider predefined trajectories through the intersection that vehicles ought to follow, while maintaining a safe distance, using continuous-time control strategies; examples are found in [9]–[14].

The Cooperative Intersection Control (CIC) strategy, proposed in [15], is a TP approach that achieves a safe crossing of vehicles through an unsignalized intersection by defining and regulating virtual platoons of vehicles driving on different lanes of the intersection. As presented in [15], the CIC approach implements a First-Come-First-Served algorithm to assign the crossing sequence of vehicles, which performs better than a fixed traffic light cycle but is far from optimal in terms of throughput. The purpose of this work is to develop an algorithm that optimizes the assignment of the crossing sequence for the CIC methodology.

Within *Cooperative Intersection Management* approaches, we find solutions that optimize the crossing sequence of vehicles. In general, these solutions use Model Predictive Control (MPC) to achieve optimization; note that MPC uses a prediction model and a cost function to be minimized. The work in [16] presents a distributed approach that decomposes the optimization problem into a time-slot allocation problem and a vehicle control problem, in this work a given crossing sequence is considered. The works in [17]–[19] present an approach which considers a quadratic cost function that penalizes the difference between the dynamical state of a vehicle (velocity, and acceleration) and the dynamical state required to achieve a safe crossing of each vehicle. A distributed solution is presented in [17], [18], whereas a centralized solution is presented in [19]. The work in [20] presents a centralized approach with a similar quadratic cost function, but it considers space sampling instead of time sampling. In contrast, the centralized approach in [21], defines a cost function in terms of the time that a vehicle spends in the intersection, its fuel consumption and a comfort level metric.

The application of MPC in the aforementioned approaches for intersection control focuses on controlling directly the dynamics and interaction of individual vehicles. The objective of this work is to achieve an optimal crossing sequence for CIC, as proposed in [15], where the latter already regulates a safe and efficient relative motion between the vehicles in the intersection. Hence, the MPC-based intersection access management approach, proposed here, does not control the vehicles, but focuses on controlling the sequence in which vehicles are granted access to the intersection to optimize throughput. In support of such an approach, we propose to add a level of abstraction to model and control the intersection itself. For this matter, we consider that road intersections have been modeled successfully at such abstraction level using queuing theory [22]. Such queuing models have aided the development of optimal traffic light cycles, as shown in [23]–[25].

A preliminary version of this approach is presented in [26], which differs from this work in the following ways. Firstly, in the current work, we present an in-depth model description of the intersection dynamics, compared to a concise one in [26]. Secondly, in [26], the discretization of the queuing model allows for the occurrence of controlled events only at sampling instants, whereas in the current paper the proposed approach allows for the description of the occurrence of controlled events during the inter-sampling periods. This extension, in turn, allows for a less conservative solution in terms of intersection throughput. Thirdly, we now propose a computationally more efficient formulation of the optimization problem underlying the intersection control approach, which further supports applicability. Finally, a more extensive simulation case study is presented.

The main contribution of this work is the development of a methodology that optimizes the crossing sequence of vehicles through an intersection such that the time vehicles wait to gain access to the intersection is minimized. In Section II, the problem statement is formalized by introducing the necessary assumptions needed to regard the intersection problem as a queuing problem. Section III presents the hybrid dynamical queuing system, including its constraints, that models the behavior of the intersection. This model is then used, in Section IV, to design a MPC controller that minimizes the time that vehicles spend in the intersection. The performance of the controller is studied, in Section V, by means of two representative case studies. Finally, this work is concluded in Section VI.

Before introducing the problem statement, we introduce some notional conventions below.

A. Notation

Consider the set of all real numbers $\mathbb{R}$, the empty set $\emptyset$, and the sets $\mathbb{N} = \{0, 1, 2, \cdots\}$, $\mathbb{R}_{\geq 0} = \{z | z \geq 0, z \in \mathbb{R}\}$, $\mathbb{D}_a = \{z | 0 \leq z \leq a, a > 0\}$, $\mathbb{B} = \{0, 1\}$, $\mathbb{O} = \{0\}$, $\mathbb{I} = \{1\}$. Moreover, consider the matrices $\gamma_a \in \mathbb{I}^{a \times 1}$, $O_{a,b} \in \mathbb{O}^{a \times b}$, and $I_n \in \mathbb{B}^{n \times n}$, the elements of which are

$$(I_n)_{ab} = \begin{cases} 1, & \text{if } a = b, \\ 0, & \text{if } a \neq b. \end{cases} \tag{1}$$

We denote the Hadamard product (or element-wise product) of two matrices of the same dimension as $(C \circ D)_{ab} = (C)_{ab}(D)_{ab}$. Finally, consider that $\operatorname{dom} z$ represents the domain of z.

II. Problem Statement

This section presents the underlying assumptions to consider the intersection management problem as a queue serving problem. But before we start with such definitions, we offer a short summary of the CIC strategy, which is the low-level cooperative vehicle dynamics control strategy used as a basis for the intersection management problem considered here. Note that the optimal solution to the intersection management problem, presented in this work, is considered as a high-level layer applied to the CIC strategy.

The CIC strategy, in [15], is designed for Cooperative Autonomous Vehicles (CAVs), which are vehicles equipped with actuators to accelerate/decelerate and steer, a GPS to determine its position in space, sensors that measure the dynamical state of the vehicle (namely, longitudinal velocity and acceleration, and yaw rate), and a wireless communication antenna. Each vehicle broadcasts a heartbeat message which contains its position, dynamical state, and directional intention. Three main controllers regulate the dynamical state of each vehicle, namely, the Path-Following Control (PFC), which ensures that the vehicle stays on a given path, the Cruise Control (CC), which is a velocity regulator, and the Cooperative Adaptive Cruise Control (CACC), which is an inter-vehicle distance regulator. Note that, in the context of the CIC strategy, vehicles are able to define and regulate a virtual inter-vehicle distance, which is defined for vehicles driving on different lanes of the intersection. The regulation of such virtual inter-vehicle distance achieves the safe crossing of vehicles with crossing paths. With the aforementioned control strategies in play, the vehicles are able to cross the intersection in a First Come First Served (FCFS) basis. Using CIC strategy for the (cooperative) control of the individual vehicle dynamics, we can make the following assumptions:

- Each vehicle follows a fixed path through the intersection.
- The vehicles travel with a velocity less than or equal to a given maximum velocity value $\check{v}$.
- Collision avoidance is achieved by constantly regulating the relative motion between vehicles with crossing paths.
- The dynamical state of each vehicle is known.

Note that the velocity assumption is a consequence of the interaction between vehicles regulated with CC and CACC.

Now, let us start with the preliminary definitions. First, consider the generalization of a road intersection or just intersection for short. An intersection is the point in which three or more road segments (or arms) meet. Each road segment is either a one- or two-way street divided into lanes. We refer to the lanes that direct traffic towards the intersection as input

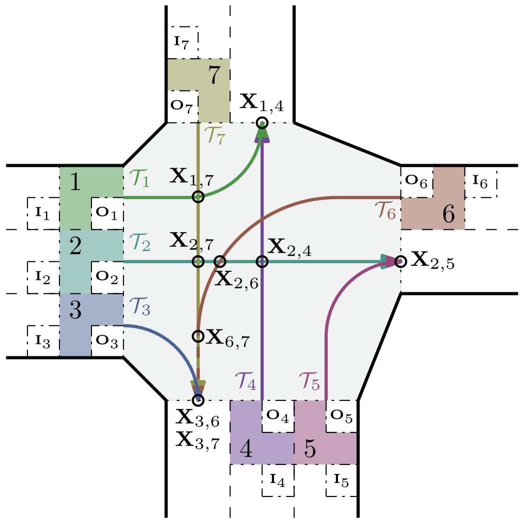

Fig. 1. Representation of the queues, paths, and collision points of an intersection with four arms, seven input lanes (colored), and three output lanes (white).

lanes. On the other hand, we refer to the lanes that direct traffic away from the intersection as output lanes. Figure 1 depicts a particular example.

Consider an intersection with n input lanes such that each input lane $q \in \mathcal{Q} = \{1, \cdots, n\}$ has a distinct path $\mathcal{T}_q \subset \mathbb{R}^2$ associated with it, as depicted in Figure 1. In other words, each particular input lane (e.g. lane 1 in Figure 1) fully determines the path through the intersection (e.g., $\mathcal{T}_1$ in Figure1) and, consequently, also the output lane. Note that $\mathbf{O}_q$ is the starting point of $\mathcal{T}_q$. Two paths can be either crossing or non-crossing. To define this property, consider the set

$$\mathbb{T} = \{(a, b) | a, b \in \mathcal{Q}; a \neq b; \mathcal{T}_{a,b} \neq \emptyset\}, \tag{2}$$

where $\mathcal{T}_{a,b} = \mathcal{T}_{b,a} = \mathcal{T}_a \cap \mathcal{T}_b$. Note that $\mathcal{T}_{a,b} = \emptyset$ means that paths a and b do not intersect. Therefore, the path of lane $a \in \mathcal{Q}$ crosses the path of lane $b \in \mathcal{Q}$ if $(a, b) \in \mathbb{T}$ (for instance, $\mathcal{T}_1$ and $\mathcal{T}_7$ in Figure 1). On the other hand, the path of lane $a \in \mathcal{Q}$ does not cross the path of lane $b \in \mathcal{Q}$ if $(a, b) \notin \mathbb{T}$ (for instance, $\mathcal{T}_1$ and $\mathcal{T}_2$ in Figure 1). Moreover, note that the set $\mathcal{T}_{a,b}$ may contain more than one point (for instance $\mathcal{T}_{6,7}$, see $\mathcal{T}_6$ and $\mathcal{T}_7$ in Figure 1). Therefore, we define a unique collision point as

$$\mathbf{X}_{a,b} = \arg \min_{\mathbf{X} \in \mathcal{T}_{a,b}} \|\mathbf{X} - \mathbf{O}_a\|. \tag{3}$$

It is worth noting that $\mathbf{X}_{a,b} = \mathbf{X}_{b,a}$, and that the calculation of the collision point gives the same result if $\mathbf{O}_b$ is used instead of $\mathbf{O}_a$. Several collision points are depicted in Figure 1.

Consider that every path can be parameterized as $\mathcal{T}_q(\tau)$, $\forall \tau \in \mathbb{R}$, such that the path starting point is defined as $\mathbf{O}_q := \mathcal{T}_q(0)$. Given this parameterization we can define the curvilinear path coordinate, associated to each path $\mathcal{T}_q(\tau)$, as

$$s_q(\tau) = \int_0^\tau |\mathcal{T}'_q(\sigma)| d\sigma, \tag{4}$$

where $\mathcal{T}'_q(\sigma) := d\mathcal{T}_q/d\sigma$. Note that (4) is referred to as the arc length integral. With this definition we can give an insight of the functionality of the CIC strategy. Consider the vehicles $V1$ and $V2$, driving on lane a and b, respectively, such that $V1$ crosses the intersection before $V2$. The position of $V1$ on the path $\mathcal{T}_a$ can be defined as $s_a(\tau_1)$. Similarly, the position of $V2$ on the path $\mathcal{T}_b$ can be defined as $s_b(\tau_2)$. Therefore, we can define the virtual inter-vehicle distance between $V1$ and $V2$ as $\Delta s_{a,b}(\tau_1, \tau_2) = s_a(\tau_1) - s_b(\tau_2) - L_1 - S_{a,b} + S_{b,a}$, where L_1 is the length of vehicle $V1$, $S_{a,b} = s_a(\tau_{a,b})$, and $S_{b,a} = s_b(\tau_{b,a})$, with $\tau_{a,b}$ and $\tau_{b,a}$ defined such that $\mathcal{T}_a(\tau_{a,b}) = \mathbf{X}_{a,b}$, and $\mathcal{T}_b(\tau_{b,a}) = \mathbf{X}_{b,a}$. This (virtual) inter-vehicle distance is then regulated (by the CACC) to the reference value Δs_{ref}, such that $\lim_{t \to \infty} \Delta s_{a,b}(t) - \Delta s_{ref} = 0$.

The first step towards defining the queue serving problem is to consider a space along each lane in which a vehicle is considered to be in the queue (this is depicted in Figure 1 by the colored blocks). Every time a vehicle crosses the point $\mathbf{I}_q$ the queue length $x_q \in \mathbb{N}$ increases by one and the vehicle is considered to be in the queue. Similarly, when a vehicle crosses the point $\mathbf{O}_q$, or when it is granted access, the queue length decreases by one and the vehicle is considered to be served (i.e., it then enters and passes through the intersection). We refer to the zone delimited by $\mathbf{I}_q$ and $\mathbf{O}_q$ as the queue zone. The vehicle at the front of queue is instructed to stop at $\mathbf{O}_q$ if no access has been granted to it, as if waiting on a lane controlled by a traffic light. However, in many cases the queue zone would contain a number of *moving* vehicles, which is referred to as a *moving queue*, awaiting a decision on access to the intersection, given that the queue zone is long enough and that the vehicle server is fast. Although the *moving queue* concept describes accurately the behavior of vehicles approaching the intersection, in the remainder of this work we consider (for the sake of simplicity) a *static queue* of vehicles. In other words, we consider that all vehicles in every queue are waiting at point $\mathbf{O}_q$ to gain access to the intersection. Note that to implement the *moving queue* concept instead of the *static queue* concept we need to consider the time it takes a vehicle to reach $\mathbf{O}_q$, which would need to be estimated, based on the dynamical state of the vehicle.

The second step towards defining the queue serving problem is to define the service time matrix $T^s \in \mathbb{R}^{n \times n}$, the elements of which $T^s_{a,b} := (T^s)_{ab}$ represent the time that the queue server needs to wait to serve queue b after queue a was served. First, consider a pair of queues such that we can define the time to collision $\tilde{t}_{a,b}$, $\forall (a, b) \in \mathbb{T}$, with $\mathbb{T}$ defined as in (2), which is the time that a vehicle in queue a takes to travel from $\mathbf{O}_a$ to $\mathbf{X}_{a,b}$, and, similarly, $\tilde{t}_{b,a}$, $\forall (b, a) \in \mathbb{T}$ is the time that a vehicle in queue b takes to travel from $\mathbf{O}_b$ to $\mathbf{X}_{b,a}$. Finally, consider the time gap $\bar{t}_h > 0$ which is the desired minimum time it takes one vehicle to cross a specific point after other vehicle has crossed it. We can calculate $\bar{t}_h$ as $\bar{t}_h = \Delta s_{ref}/\check{v}$, where Δs_{ref} is the reference virtual inter-vehicle distance and $\check{v}$ is the maximum velocity. Considering all the aforementioned terms, we can define the elements of the service matrix as

$$T^s_{a,b} = \begin{cases} 0, & \forall (a, b) \notin \mathbb{T}, \\ \bar{t}_h + \tilde{t}_{a,b} - \tilde{t}_{b,a}, & \forall (a, b) \in \mathbb{T}, \\ \bar{t}_h, & \text{for } a = b, \end{cases} \tag{5}$$

note that $T^s_{a,b} \neq T^s_{b,a}$, and $T^s_{a,b} \in \mathbb{R}$. The calculation of the elements of (5) in the context of the CIC strategy in [15] would require a time prediction strategy based on the current state (position, velocity, and acceleration) of each vehicle at the front of each queue, and its relation to other vehicles. However, this prediction strategy is out of the scope of this work where we consider static queues of vehicles. For the remainder of this paper, we consider that the constant values $\tilde{t}_{a,b}, \tilde{t}_{b,a} \in \mathbb{R}, \forall a, b \in \mathcal{Q}$, are calculated based on the geometry of the intersection and the nominal velocity of the vehicles within the intersection.

In conclusion we, can state the queue serving problem as follows. Given a set of queues $\mathcal{Q}$ design a queue server, constrained by the service time matrix T^s, that minimizes the time it takes to empty all a and b queues.

III. Hybrid Queuing Dynamical System

Consider a road intersection, as defined in the previous section, such that a queue $q \in \mathcal{Q}$ is associated to each of its n input lanes. Two distinct events modify the state of each queue: an arrival w_q to the queue, and a departure u_q from the queue (leading to access to the intersection). The state of each queue is represented by the queue length x_q, and by two inter-event timers, namely the inter-arrival timer $t_{u,q}$, and the inter-departure timer $t_{w,q}$. When an event occurs, either an arrival or a departure, the queue length is updated and the inter-event timer, either inter-arrival or inter-departure, is reset to zero.

To design a Model Predictive Controller, the development of which is presented in Section IV, we need to define a prediction model, of the process and its disturbances, and a cost function [27]. The prediction model is used to estimate the state of the process t_N seconds into the future. This prediction time is defined as $t_N = N\Delta t$ where N is the prediction horizon and Δt is the sampling time. If we model the dynamics of the queue state as a plain discrete system (such that we have an On-Sampling-Event (OSE) representation) we would need to select a small sampling time to represent accurately the inter-event timers. A small sampling time Δt would require a big prediction horizon N to achieve a reasonable prediction time t_N. This would result in a high computational burden since the optimization problem (of size proportional to N) has to be solved every Δt seconds. To reduce the computational burden, we represent the queuing process as a hybrid dynamical system such that the inter-event timers are represented in continuous time, and the queue lengths jump on event times. Then, this hybrid system is sampled in such a way that we allow for an Inter-Sampling-Event (ISE) representation. This means that we define a prediction model that describes if an event happens and when it happens during the sampling interval. Therefore, we can use a bigger sampling time Δt and a lower prediction horizon N to achieve a reasonable prediction time t_N. Which in turn, will decrease the computational burden. It is worth nothing that the preliminary work in [26] uses an OSE representation.

Consider the hybrid dynamical system

$$\begin{cases} \dot{z} = f(z,e), & \text{if } (z,e) \in \mathcal{C}, \\ z^+ = g(z,e), & \text{if } (z,e) \in \mathcal{D}, \end{cases} \tag{6}$$

where $z \in \mathbb{R}^{\alpha \times 1}$ is the state vector, $\dot{z}$ is the rate of change of the state z, z^+ is the value of the state after an instantaneous change, $e \in \mathbb{R}^{\beta \times 1}$ is the input vector, $f(z,e)$ is the flow map (or vector field), $g(z,e)$ is the jump map, $\mathcal{C}$ is the flow set, and $\mathcal{D}$ is the jump set. The solutions of (6) are defined on the hybrid time domain (t, j), where t represents continuous time, and j represents the event counter. Definitions on the solution concept for the hybrid dynamical system in (6) are presented in the Appendix.

Now we can introduce the definitions that govern the dynamics of the state of the queues. The input and state vectors in (6) are given by

$$e := \begin{bmatrix} w^T & u^T \end{bmatrix}^T, \quad \text{and} \quad z := \begin{bmatrix} x^T & t_w^T & t_u^T \end{bmatrix}^T, \tag{7}$$

where $w \in \mathbb{B}^{n\times 1}$ is the arrivals vector, and $u \in \mathbb{B}^{n\times 1}$ is the departures vector, $x \in \mathbb{N}^{n\times 1}$ is the queue lengths vector, $t_w \in \mathbb{R}_{\geq 0}^{n\times 1}$ is the inter-arrival timers vector, and $t_u \in \mathbb{R}_{\geq 0}^{n\times 1}$ is the inter-departure timers vector. Note that for every queue $q \in \mathcal{Q}$ we can define, considering all the vectors that constitute (7), the individual queue states $w_q := (w)_q$, $u_q := (u)_q$, $x_q := (x)_q$, $t_{u,q} := (t_u)_q$, and $t_{w,q} := (t_w)_q$.

The arrivals and departures of vehicles (for individual queues) are defined in continuous t time as follows:

$$w_q(t) = \begin{cases} 1 & \text{if a vehicle arrives at time } t, \\ 0 & \text{otherwise,} \end{cases}$$
$$u_q(t) = \begin{cases} 1 & \text{if a vehicle departs at time } t, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

which are instantaneous events. These events can be described in the hybrid time domain for all queues as follows:

$$w(t,j) = \begin{cases} w(t), & j = \min\{j^* | (t, j^*) \in \operatorname{dom}\phi\}, \\ 0, & j \neq \min\{j^* | (t, j^*) \in \operatorname{dom}\phi\}, \end{cases}$$
$$u(t,j) = \begin{cases} u(t), & j = \min\{j^* | (t, j^*) \in \operatorname{dom}\phi\}, \\ 0, & j \neq \min\{j^* | (t, j^*) \in \operatorname{dom}\phi\}, \end{cases} \tag{9}$$

with $(t, j) \in \operatorname{dom}\phi$. Note that ϕ is the solution of the hybrid system, and that the conditions on the event counter j in (9) are to ensure that there are no repeated events at time t induced by the continuous-times inputs $w(t)$ and $u(t)$.

The flow map $f(z,e)$ in (6) is defined as

$$f(z,e) := \begin{bmatrix} O_{1,n} & \gamma_n^T & \gamma_n^T \end{bmatrix}^T. \tag{10}$$

The definition of the jump map $g(z,e)$ in (6) is made in parts. Consider

$$g(z,e) = \begin{bmatrix} g_1(z,e)^T & g_2(z,e)^T & g_3(z,e)^T \end{bmatrix}^T. \tag{11}$$

The jump map $g_1(z,e) \in \mathbb{N}^{n\times 1}$ describes a jump of the the queue length state and is defined as

$$g_1(z,e) := x + w - u, \tag{12}$$

The jump map $g_2(z,e) \in \mathbb{R}_{\geq 0}^{n\times 1}$ describes a reset of the inter-arrival timers, for every $q \in \mathcal{Q}$ there is a sub-map $g_{2,q}(z,e) := (g_2(z,e))_q$ defined by

$$g_{2,q}(z,e) := \begin{cases} 0, & \text{if } w_q = 1, \\ t_{w,q}, & \text{otherwise.} \end{cases} \tag{13}$$

Finally, the jump map $g_3(z,e) \in \mathbb{R}^{n\times 1}_{\geq 0}$ describes a reset of the inter-departure timers, for every $q \in \mathcal{Q}$ there is a sub-map $g_{3,q}(z,e) := (g_3(z,e))_q$ defined by

$$g_{3,q}(z,e) := \begin{cases} 0, & \text{if } u_q = 1, \\ t_{u,q}, & \text{otherwise.} \end{cases} \tag{14}$$

The instantaneous events, defined in (9), represent the arrivals and departures of vehicles. The vehicles can not arrive at (or depart from) the same lane at the same continuous time instant t since the vehicles are modeled as physical entities which cannot be at the same place at the same time. Therefore, there exists a minimum inter-arrival and inter-departure time for each queue. For this matter, the following realistic assumption is posed.

Assumption 1: For all $t \geq 0$ such that $u_q(t) = 1$, with $q \in \mathcal{Q}$, there exist an $\epsilon_u > 0$ such that $u_q(t', j') = 0$, $\forall (t', j') \in \operatorname{dom}\phi$, with $t' \in (t, t+\epsilon_u]$. Additionally, for all $t \geq 0$ such that $w_q(t) = 1$, with $q \in \mathcal{Q}$, there exist an $\epsilon_w > 0$ such that $w_q(t', j') = 0$, $\forall (t', j') \in \operatorname{dom}\phi$, with $t' \in (t, t+\epsilon_w]$.

It is worth noting that this assumption prevents Zeno behavior, which is when an unbounded number of events occur in a bounded time interval. Next, we design the flow and jump sets, $\mathcal{C}$ and $\mathcal{D}$ in (6), such that Assumption 1 is satisfied. In other words, the hybrid system in (6) flows as long as there are no events (arrivals, or departures), or during events that violate Assumption 1, and jumps when there is an event which is in correspondence with Assumption 1.

Two pairs of flow sets and jump sets are defined, one pair ($\mathcal{C}_u$,$\mathcal{D}_u$) dependent on the arrivals, and one pair ($\mathcal{C}_w$,$\mathcal{D}_w$) dependent on the departures. The sets dependent on the departures are given by

$$\mathcal{C}_u = \bigcup_{q\in\mathcal{Q}} \left(\mathcal{C}^*_{u,q} \cup \mathcal{C}^\star_{u,q}\right), \quad \text{and} \quad \mathcal{D}_u = \bigcup_{q\in\mathcal{Q}} \mathcal{D}_{u,q}, \tag{15}$$

where

$$\begin{aligned} \mathcal{C}^*_{u,q} &= \left\{(t_{u,q}, u_q) | t_{u,q} \geq 0, u_q = 0\right\}, \\ \mathcal{C}^\star_{u,q} &= \left\{(t_{u,q}, u_q) | t_{u,q} \leq \epsilon_u, u_q = 1\right\}, \\ \mathcal{D}_{u,q} &= \left\{(t_{u,q}, u_q) | t_{u,q} \geq \epsilon_u, u_q = 1\right\}. \end{aligned} \tag{16}$$

The sets dependent on the arrivals are given by

$$\mathcal{C}_w = \bigcup_{q\in\mathcal{Q}} \left(\mathcal{C}^*_{w,q} \cup \mathcal{C}^\star_{w,q}\right), \quad \text{and} \quad \mathcal{D}_w = \bigcup_{q\in\mathcal{Q}} \mathcal{D}_{w,q}, \tag{17}$$

where

$$\begin{aligned} \mathcal{C}^*_{w,q} &= \left\{(t_{w,q}, w_q) | t_{w,q} \geq 0, w_q = 0\right\}, \\ \mathcal{C}^\star_{w,q} &= \left\{(t_{w,q}, w_q) | t_{w,q} \leq \epsilon_w, w_q = 1\right\}, \\ \mathcal{D}_{w,q} &= \left\{(t_{w,q}, w_q) | t_{w,q} \geq \epsilon_u, w_q = 1\right\}. \end{aligned} \tag{18}$$

Combining the sets defined above, we can define the flow set $\mathcal{C}$ and the jump set $\mathcal{D}$ in (6) as

$$\begin{aligned} \mathcal{C} &:= \left\{(z,e) | (t_u, u) \in \mathcal{C}_u \ \wedge \ (t_w, w) \in \mathcal{C}_w\right\}, \\ \mathcal{D} &:= \left\{(z,e) | (t_u, u) \in \mathcal{D}_u \ \vee \ (t_w, w) \in \mathcal{D}_w\right\}. \end{aligned} \tag{19}$$

Finally, we can conclude the definition of the hybrid queuing dynamical system in (6), where the input vector e and the state vector z are defined as in (7), the flow map $f(z,e)$ is defined as in (10), the jump map $g(z,e)$ is defined as in (11)-(14), and the flow set $\mathcal{C}$ and the flow set $\mathcal{D}$ are defined as in (15)-(19). Note that, in the input vector e, we consider w as a disturbance to the system and u as the control input.

A. Constraints on the Control Input

The queue server is constrained by the serving time matrix $T^s \in \mathbb{R}^{n\times n}_{\geq 0}$. In other words, we need to wait $T^s_{a,b}$ seconds to authorize a departure from queue $b \in \mathcal{Q}$ which follows a departure from queue $a \in \mathcal{Q}$. We can write the aforementioned condition as

$$u_b(t,j) \in \begin{cases} \mathbb{O}, & \text{if } \exists a \in \mathcal{Q}, \text{such that } t_{u,a}(t,j) < T^s_{a,b}, \\ \mathbb{B}, & \text{otherwise.} \end{cases} \tag{20}$$

Moreover, if two queues are conflicting (meaning that $(a,b) \in \mathbb{T}$) then a departure from queue $a \in \mathcal{Q}$ and from queue $b \in \mathcal{Q}$ cannot be authorized at the same time t. Such constraint can be written as

$$\begin{aligned} &u_a(t, j_1) + u_b(t, j_2) \leq 1, \\ &\qquad \forall (a,b) \in \mathbb{T}, \ (t, j_1), (t, j_2) \in \operatorname{dom}\phi. \end{aligned} \tag{21}$$

Finally, a departure from an empty queue should not be authorized. This constraint can be written as

$$u_a(t,j) \in \mathbb{O}, \quad \text{if } x_a(t,j) = 0, \tag{22}$$

with $a \in \mathcal{Q}$.

IV. Optimal Intersection Access Management

This section presents the Optimal Intersection Access Management (OIAM) strategy based on model predictive control. Hereto, the discretization of the hybrid queuing system dynamics, described in Section III, is presented in Section IV-A. Based on this discretization we formulate part of the MPC problem by defining a prediction model, defined in Section IV-B, and a cost function, defined in Section IV-C. By considering the constraints on the control input, presented in Section IV-D, we shape the MPC problem as a Mixed-Integer Linear Programming (MILP) optimization problem, which is presented in Section IV-E, the solution of which yields the optimal control law.

A. Discretization of the Hybrid Queuing Dynamical System

Consider a sampling time interval Δt such that the time instant $t_k = k\Delta t$, $\forall k \in \mathbb{N}$. Given this sampling interval we can write the discretized version of (6) as

$$z(k+1) = h\big(z(k), e(k)\big), \tag{23}$$

where $z(k) = z(t_k, j^*)$, $j^* = \max\{j | (t_k, j) \in \operatorname{dom}\phi\}$, with z defined as in (7). Note that the event counter j^* ensures that $z(k)$ represents the value of $z(t_k, j)$ after the latest jump. We know from the definitions in (8) and (9) that the vector $e(t,j) = \left[w(t,j)^T \ u(t,j)^T\right]^T$ consists of instantaneous (in t) signals which cannot be simply sampled. Therefore, to define $e(k) = \left[w(k)^T \ u(k)^T\right]^T$ we need to introduce the following assumption.

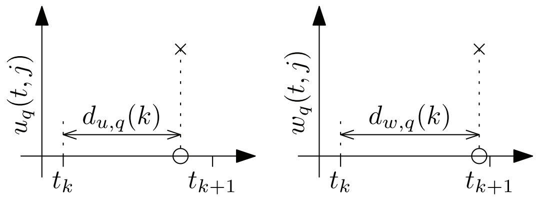

Fig. 2. Illustration of the event delays $d_{u,q}(k)$ and $d_{w,q}(k)$.

Assumption 2: The sampling interval Δt is such that there is at most one arrival and at most one departure on each queue $q \in \mathcal{Q}$ for each sampling interval $[t_k, t_{k+1})$, for $k \in \mathbb{N}$. In other words, $u_q(t,j)$ and $w_q(t,j)$ are non-zero at most once for $t \in [t_k, t_{k+1})$ for each $q \in \mathcal{Q}$ and $(t,j) \in \operatorname{dom}\phi$.

Note that this assumption can be satisfied by taking

$$\Delta t < \min(\epsilon_u, \epsilon_w), \tag{24}$$

with ϵ_u and ϵ_w as in Assumption 1 (so it is taken for granted that Assumption 1 is satisfied). From (5), we know that a departure may be authorized at least every M^s seconds, with

$$M^s = \min\{T^s_{a,b} > 0\}, \quad \forall a, b \in \mathcal{Q}, \tag{25}$$

such that we can define

$$\epsilon_u \leq M^s. \tag{26}$$

Therefore, we can define a definite bound for the sampling time in (24), considering (26) and $\epsilon_u < \epsilon_w$, as $\Delta t < M^s$.

Consider the instantaneous signals $u_q(t,j)$, $w_q(t,j)$, depicted in Figure 2. We can define the so-called event delay for each instantaneous signal, for the time instant t_k, as follows:

$$d_{w,q}(k) = \begin{cases} & \text{if } \exists t^* \in [t_k, t_{k+1}), \text{such that} \\ t^* - t_k, & w_q(t^*, j^*) = 1, \text{for some} \\ & (t^*, j^*) \in \operatorname{dom}\phi, \\ \Delta t, & \text{otherwise,} \end{cases} \tag{27}$$

and

$$d_{u,q}(k) = \begin{cases} & \text{if } \exists t^* \in [t_k, t_{k+1}), \text{such that} \\ t^* - t_k, & u_q(t^*, j^*) = 1, \text{for some} \\ & (t^*, j^*) \in \operatorname{dom}\phi, \\ \Delta t, & \text{otherwise,} \end{cases} \tag{28}$$

where $d_{u,q}(k) := (d_u(k))_q$ and $d_{w,q}(k) := (d_w(k))_q$, for $d_u(k), d_w(k) \in \mathbb{D}^{n\times 1}_{\Delta t}$. Therefore, we can write the discretized version of the arrivals and departures as

$$w_q(k) = \begin{cases} 1, & \text{if } d_{w,q}(k) < \Delta t, \\ 0, & \text{otherwise,} \end{cases}$$
$$u_q(k) = \begin{cases} 1, & \text{if } d_{u,q}(k) < \Delta t, \\ 0, & \text{otherwise,} \end{cases} \tag{29}$$

where $w_q(k) := (w(k))_q$ and $u_q(k) := (u(k))_q$ for $w(k), u(k) \in \mathbb{B}^{n\times 1}$, which represent if an event (arrival or departure) occurs in the time interval $[t_k, t_{k+1})$. Moreover, the definitions in (27), (28), and (29) are necessary to discretize the dynamics of the inter-event timers, including the jumps, in (13) and (14), as follows:

$$t_{w,q}(k+1) = \begin{cases} \Delta t - d_{w,q}(k), & \text{if } w_q(k) = 1, \\ \Delta t + t_{w,q}(k), & \text{otherwise,} \end{cases}$$
$$t_{u,q}(k+1) = \begin{cases} \Delta t - d_{u,q}(k), & \text{if } u_q(k) = 1, \\ \Delta t + t_{u,q}(k), & \text{otherwise,} \end{cases} \tag{30}$$

where $t_{w,q}(k) := (t_w(k))_q$, $t_{u,q}(k) := (t_u(k))_q$, for $t_w(k), t_u(k) \in \mathbb{R}^{n\times 1}_{\geq 0}$; such that if an event occurs in the time interval $[t_k, t_{k+1})$, then the inter-event timers are reset considering the values of the corresponding event delays.

Finally, we can define the discrete map $z(k+1) = h\big(z(k), e(k)\big)$ in (23) as

$$z(k+1) = \begin{bmatrix} x(k) + w(k) - u(k) \\ \gamma_n \Delta t + t_w(k) - w(k) \circ (t_w(k) + d_w(k)) \\ \gamma_n \Delta t + t_u(k) - u(k) \circ (t_u(k) + d_u(k)) \end{bmatrix}. \tag{31}$$

Note that in this definition $d_u(k)$ becomes a second control decision variable besides $u(k)$. Moreover, $d_w(k)$ is determined by the disturbance $w(t,j)$.

Constraints on the Control Input: Given the definition in (31), we can define the discrete version of the constraints in (20), (21), and (22) as

$$T^s_{a,b} u_b(k) \leq t_{u,a}(k) + d_{u,a}(k), \tag{32}$$
$$u_a(k) + u_b(k) \leq 1, \quad \forall (a,b) \in \mathbb{T}, \tag{33}$$
$$u_a(k) \leq x_a(k). \tag{34}$$

Note that in all cases $a, b \in \mathcal{Q}$.

B. Prediction Model

The model presented in the previous section shows a bilinear form, which is a type of nonlinearity. According to [28], this bilinear problem can be solved using a Mixed Integer Linear Program (MILP) formulation. It is worth noting that the reformulation as in [28] arrives at the same solution as the nonlinear program, the approaches differ in the performance of the solver.

We note that the dynamics of the inter-departure timers $t_u(k)$ and the dynamics of the queue lengths $x(k)$ are both independent of the dynamics of the inter-arrival timers $t_w(k)$ (the latter of which is relevant for the definition of the disturbance model). Moreover, we note that the dynamics of the inter-departure timers contain a product of the input $u(k)$ with the departure timing delay $d_u(k)$ and the timers $t_u(k)$ themselves, see (31). To remove the aforementioned products we introduce the variables $\tau(k)$ and $\delta(k)$ which should satisfy:

$$\tau(k) := u(k) \circ t_u(k),$$
$$\delta(k) := u(k) \circ d_u(k), \tag{35}$$

such that $\tau(k) \in \mathbb{R}^{n\times 1}_{\geq 0}$ and $\delta(k) \in \mathbb{D}^{n\times 1}_{\Delta t}$, which can be achieved by introducing additional constraints to the prediction model that are discussed later in this section. Now, we can rewrite (31), considering (35) (while omitting the dynamics of $t_w(k)$ for the reasons described above), as follows:

$$\zeta(k+1) = A\zeta(k) + B_1 \nu(k) + B_2 w(k) + C, \tag{36}$$

with

$$\zeta(k) = \begin{bmatrix} x(k) \\ t_u(k) \end{bmatrix}, \quad \nu(k) = \begin{bmatrix} u(k) \\ d_u(k) \\ \tau(k) \\ \delta(k) \end{bmatrix}, \tag{37}$$

where $\zeta(k)$ is the state vector, $\nu(k)$ is the input vector, and

$$A = \begin{bmatrix} I_n & O_{n,n} \\ O_{n,n} & I_n \end{bmatrix}, \quad B_1 = \begin{bmatrix} -I_n & O_{n,n} & O_{n,n} & O_{n,n} \\ O_{n,n} & O_{n,n} & -I_n & -I_n \end{bmatrix},$$
$$B_2 = \begin{bmatrix} I_n \\ O_{n,n} \end{bmatrix}, \quad C = \begin{bmatrix} O_{n,1} \\ \gamma_n \Delta t \end{bmatrix}, \tag{38}$$

where B_1 and B_2 are the input matrices, and A is the system matrix. Note that $\tau(k)$ is included in the input vector $\nu(k)$ because its value is ultimately determined by $u(k)$, and that the definition in (36) is independent of $d_u(k)$ which is anyway included in the input vector $\nu(k)$ because it is relevant for the definition of the constraints of the model.

To support Model Predictive Control we define a discrete-time prediction model for N time steps into the future. By considering (36), for N steps into the future we obtain

$$\begin{aligned} \hat{\zeta}(k_1)_k &= A\hat{\zeta}(k_0)_k + B_1\hat{\nu}(k_0)_k + B_2\hat{w}(k_0)_k + C, \\ \hat{\zeta}(k_2)_k &= A\hat{\zeta}(k_1)_k + B_1\hat{\nu}(k_1)_k + B_2\hat{w}(k_1)_k + C, \\ &\vdots \\ \hat{\zeta}(k_N)_k &= A\hat{\zeta}(k_{N-1})_k + B_1\hat{\nu}(k_{N-1})_k + B_2\hat{w}(k_{N-1})_k + C, \end{aligned} \tag{39}$$

where the vectors $\hat{\sigma}(k_i)_k$, $\forall i \in \{0, \cdots, N\}$, represent the predicted value of the vector $\sigma \in \{\zeta, \nu, w\}$ at time $k_i = k + i$ evaluated at time k. Note that $\hat{\zeta}(k_0)_k = \zeta(k)$. Finally, the prediction model can be defined, using (39) and recurring substitution, as follows:

$$\underline{\hat{\zeta}}_k = \Lambda\zeta(k) + \Omega_1\underline{\hat{\nu}}_k + \Omega_2\underline{\hat{w}}_k + \Gamma, \tag{40}$$

where

$$\underline{\hat{\zeta}}_k := \begin{bmatrix} \hat{\zeta}(k_1)_k \\ \hat{\zeta}(k_2)_k \\ \vdots \\ \hat{\zeta}(k_N)_k \end{bmatrix}, \underline{\hat{\nu}}_k := \begin{bmatrix} \hat{\nu}(k_0)_k \\ \hat{\nu}(k_1)_k \\ \vdots \\ \hat{\nu}(k_{N-1})_k \end{bmatrix}, \underline{\hat{w}}_k := \begin{bmatrix} \hat{w}(k_0)_k \\ \hat{w}(k_1)_k \\ \vdots \\ \hat{w}(k_{N-1})_k \end{bmatrix}, \tag{41}$$

$$\Lambda = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \Omega_1 = \begin{bmatrix} B_1 & O_{b_1} & \cdots & O_{b_1} \\ AB_1 & B_1 & \cdots & O_{b_1} \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B_1 & A^{N-2}B_1 & \cdots & B_1 \end{bmatrix}, \tag{42}$$

$$\Omega_2 = \begin{bmatrix} B_2 & O_{b_2} & \cdots & O_{b_2} \\ AB_2 & B_2 & \cdots & O_{b_2} \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B_2 & A^{N-2}B_2 & \cdots & B_2 \end{bmatrix}, \quad \Gamma = \begin{bmatrix} C \\ C \\ \vdots \\ C \end{bmatrix}, \tag{43}$$

with $b_1 = \dim B_1$ and $b_2 = \dim B_2$. Note that Λ, Ω_1, and Ω_2 are the system matrices of the prediction model, and Γ is a constant matrix. Moreover, $\underline{\hat{\zeta}}_k$, $\underline{\hat{\nu}}_k$, and $\underline{\hat{w}}_k$ are the predicted state, input, and disturbance vectors at time k, respectively.

Disturbance Model: From (41) we know that it is necessary to determine the future values of the disturbance vector represented by $\underline{\hat{w}}_k$, the elements of which are given by

$$(\underline{\hat{w}}_k)_i = \hat{w}(k_i)_k, \quad \forall i \in \{0, \cdots, N-1\}, \tag{44}$$

which represent the arrivals into a queue. We define the elements of the vector in (44) as $\hat{w}_q(k_i)_k := (\hat{w}(k_i)_k)_q$, $\forall q \in \mathcal{Q}$.

Consider that a vehicle arrives to queue $q \in \mathcal{Q}$ every $W_q > 0$ seconds. The value of W_q at the time instant t_k, represented by $W_q(k)$ can be calculated in two ways. It can either be set as constant which requires prior knowledge of the input flow to the intersection, or be calculated using a moving average window approach. In the latter approach, the number of arrivals in a time window is divided by the length of the time window in seconds which yields an average value of the arrival rate. In this work, we consider deterministic arrivals with known arrival rates. Therefore we set W_q as constant.

If a vehicle arrives every W_q seconds, then, according to the definition in (13), the value of the inter-arrival timer is bounded as $0 \leq t_{w,q}(t, j) < W_q$. According to (27), if an arrival occurs in the time interval $[t_k, t_{k+1})$ the arrival delay is bounded as $0 \leq d_{w,q}(k) < \Delta t$, and is calculated as $d_{w,q}(k) = W_q - t_{w,q}(k)$. Therefore, according to (29), we can define the arrival predictions in (44) as

$$\hat{w}_q(k_i)_k = \begin{cases} 1, & \text{if } W_q - \hat{t}_{w,q}(k_i)_k < \Delta t, \\ 0, & \text{otherwise,} \end{cases} \tag{45}$$

where $i \in \{0, \cdots, N-1\}$, and $\hat{t}_{w,q}(k_0)_k := t_{w,q}(k)$. Moreover, the dynamics of the prediction of the inter-arrival timer, based on (30), become

$$\hat{t}_{w,q}(k_{i+1})_k = \Delta t + \hat{t}_{w,q}(k_i)_k - \hat{w}_q(k_i)_k W_q, \tag{46}$$

where $i \in \{0, \cdots, N-1\}$.

C. Cost Function

The objective of the MPC-based design for OIAM is to minimize the time that vehicles spend within the intersection. The total time that a vehicle spends within the intersection is the sum of the time it waits to gain access plus the time it takes to cross the intersection. Here, a CIC approach, as presented in [15], is used to ensure the desired behavior of vehicles and interaction between them. Therefore, we can assume that the time it takes a vehicle to cross the intersection, once it has gained access, is fixed. Hence, we focus on minimizing the time a vehicle waits to gain access to the intersection. On this basis, we can restate the objective to minimizing the weighted average delay. According to Little's Law [29], this objective is equivalent (in steady state) to minimizing the weighted average queue lengths. Therefore, given the definitions in (37), we can write the cost function to be minimized as

$$J\left(\underline{\hat{\zeta}}_k\right) = \sum_{j=1}^{N} \hat{c}\,\hat{\zeta}(k_j)_k = \begin{bmatrix} \hat{c} & \cdots & \hat{c} \end{bmatrix} \begin{bmatrix} \hat{\zeta}(k_1)_k \\ \vdots \\ \hat{\zeta}(k_N)_k \end{bmatrix} = \underline{\hat{c}}\,\underline{\hat{\zeta}}_k, \tag{47}$$

where $\hat{c} = \begin{bmatrix} c_x & c_t \end{bmatrix}$. The queue lengths gain vector is given by $c_x = \begin{bmatrix} c_{x,1} \cdots c_{x,N} \end{bmatrix}$, and the inter-departure timers gain vector is given by $c_t = O_{1,N}$.

D. Constraints on the Prediction Model

Let us write the service time constraints in (32), the conflicting queues constraints in (33), the empty queue constraints in (34), and the constraints introduced by each of the linearizing variables in (35), using the column vectors defined in (37).

1) Service Time Constraints: The constraints in (32) represent the waiting time to authorize a departure from queue $b \in \mathcal{Q}$ after a departure is authorized from queue $a \in \mathcal{Q}$. According to (28), if $u_a(k) = 0$ then $d_{u,a}(k) = \Delta t$, which in the context of (32) implies that if $T^s_{a,b} u_b(k) \leq t_{u,a}(k) + \Delta t$, which means that we have to wait an extra Δt to authorize a vehicle from queue b when a vehicle from queue a was not authorized, which is actually unnecessary. To deal with this discrepancy we recall the definition of $\delta(k)$ in (35), which for queue a is given by $\delta_a(k) = u_a(k) d_{u,a}(k)$, and we rewrite (32) as

$$T^s_{a,b} u_b(k) \leq t_{u,a}(k) + \delta_{u,a}(k), \quad \forall a, b \in \mathcal{Q}, \tag{48}$$

which reflects the fact that the value of $d_{u,a}(k)$ is just relevant when $u_a(k) = 1$. We can write a general expression for a single queue as

$$(T^s \circ C_q) u(k) - t_u(k) - \delta_u(k) \leq O_{n,1}, \quad \forall q \in \mathcal{Q}, \tag{49}$$

where $C_q \in \mathbb{B}^{n \times n}$ is defined as

$$(C_q)_{ij} = \begin{cases} 1, & \text{if } j = q, \\ 0, & \text{otherwise}, \end{cases} \tag{50}$$

for all $i, j \in \mathcal{Q}$. Considering the column vectors in (37), (49) becomes

$$M_1 \zeta(k) + E_1 \nu(k) \leq b_1, \tag{51}$$

where $b_1 = O_{n^2,1}$,

$$M_1 = \begin{bmatrix} O_{n,n} & -I_n \\ \vdots & \vdots \\ O_{n,n} & -I_n \end{bmatrix}, \quad E_1 = \begin{bmatrix} T^s \circ C_1 & O_{n,2n} & -I_n \\ \vdots & \vdots & \vdots \\ T^s \circ C_n & O_{n,2n} & -I_n \end{bmatrix}.$$

2) Conflicting Queues Constraints: The constraints in (33) can be rewritten as

$$u_a(k) + u_b(k) \leq 1 + B^s_{a,b}, \quad \forall a, b \in \mathcal{Q}, \tag{52}$$

where

$$B^s_{a,b} = \begin{cases} 0, & \text{if } (a, b) \in \mathbb{T}, \\ 1, & \text{otherwise} , \end{cases} \tag{53}$$

with $\mathbb{T}$ as in (2), such that $B^s_{a,b} := (B^s)_{ab}$. Therefore, we can rewrite (52), using (50), as

$$(C_q + I_n) u(k) \leq \gamma_n + (\Upsilon C_q B^s)^T, \quad \forall q \in \mathcal{Q}, \tag{54}$$

where $\Upsilon = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{B}^{1 \times n}$, which, rewritten using the column vectors in (37), becomes

$$M_2 \zeta(k) + E_2 \nu(k) \leq b_2, \tag{55}$$

where $M_2 = O_{n^2,2n}$,

$$E_2 = \begin{bmatrix} C_1 + I_n & O_{n,3n} \\ \vdots & \vdots \\ C_n + I_n & O_{n,3n} \end{bmatrix}, \quad b_2 = \begin{bmatrix} \gamma_n + (\Upsilon C_1 B^s)^T \\ \vdots \\ \gamma_n + (\Upsilon C_n B^s)^T \end{bmatrix}.$$

3) Empty Queue Constraints: The constraints in (34) can be rewritten as

$$u_q(k) - x_q(k) \leq 0, \quad \forall q \in \mathcal{Q}, \tag{56}$$

which, rewritten in terms of the column vectors in (37), becomes

$$M_3 \zeta(k) + E_3 \nu(k) \leq b_3, \tag{57}$$

where

$$M_3 = \begin{bmatrix} -I_n & O_{n,n} \end{bmatrix}, \quad E_3 = \begin{bmatrix} I_n & O_{n,3n} \end{bmatrix}, \quad b_3 = \begin{bmatrix} O_{n,1} \end{bmatrix}.$$

4) Constraints for Linearizing the Product of Variables: In order to rewrite the system in (31) in the form of (36), the definitions in (35) were introduced to linearize the product of two variables. These definitions add the following constraints to the prediction model.

Consider the definition of $\tau_q(k) := (\tau(k))_q$ given by $\tau_q(k) = u_q(k) t_{u,q}(k)$, where $u_q(k) \in \mathbb{B}$ and $0 \leq t_{u,q}(k) \leq \bar{t}_u$, which is the product of a binary variable and a bounded real variable. Note that the upper-bound $\bar{t}_u$ can be defined as $\bar{t}_u = \|t_u(k)\|_\infty + t_N$, since none of the elements of the vector $t_u(k)$ will grow unbounded along the prediction time $t_N = N \Delta t$. This definition adds the following inequalities:

$$\begin{aligned} \tau_q(k) &\geq t_{u,q}(k) - (1 - u_q(k)) \bar{t}_u, \\ \tau_q(k) &\leq u_q(k) \bar{t}_u, \\ \tau_q(k) &\leq t_{u,q}(k), \end{aligned} \tag{58}$$

such that if $u_q(k) = 1$ then $\tau_q(k) = t_{u,q}(k)$, and if $u_q(k) = 0$ then $\tau_q(k) = 0$. The vector form of these inequalities is given by

$$\begin{aligned} -\tau(k) + t_u(k) + \bar{t}_u u(k) &\leq \bar{t}_u \gamma_n, \\ \tau(k) - \bar{t}_u u(k) &\leq O_{n,1}, \\ \tau(k) - t_u(k) &\leq O_{n,1}, \end{aligned} \tag{59}$$

which, rewritten in terms of the column vectors in (37), becomes

$$M_4 \zeta(k) + E_4 \nu(k) \leq b_4, \tag{60}$$

where

$$M_4 = \begin{bmatrix} O_{n,n} & I_n \\ O_{n,n} & O_{n,n} \\ O_{n,n} & -I_n \end{bmatrix}, \quad b_4 = \begin{bmatrix} \bar{t}_u \gamma_n \\ O_{n,1} \\ O_{n,1} \end{bmatrix},$$

$$E_4 = \begin{bmatrix} \bar{t}_u I_n & O_{n,n} & -I_n & O_{n,n} \\ -\bar{t}_u I_n & O_{n,n} & I_n & O_{n,n} \\ O_{n,n} & O_{n,n} & I_n & O_{n,n} \end{bmatrix}.$$

The definition of $\delta_q(k) := (\delta(k))_q$ (given by $\delta_q(k) = u_q(k) d_{u,q}(k)$, where $u_q(k) \in \mathbb{B}$ and $0 \leq d_{u,q}(k) \leq \bar{d}_u$ (in this case we can define $\bar{d}_u = \Delta t$) is analogous to the definition of $\tau_q(k)$. Therefore, we can directly define the constraints, introduced by this definition, as

$$\begin{aligned} -\delta(k) + d_u(k) + \bar{d}_u u(k) &\le \bar{d}_u \gamma_n, \\ \delta(k) - \bar{d}_u u(k) &\le O_{n,1}, \\ \delta(k) - d_u(k) &\le O_{n,1}, \end{aligned} \tag{61}$$

which, rewritten in terms of the column vectors in (37), becomes

$$M_5 \zeta(k) + E_5 \nu(k) \le b_5, \tag{62}$$

where $M_5 = O_{3n,2n}$,

$$E_5 = \begin{bmatrix} \bar{d}_u I_n & I_n & O_{n,n} & -I_n \\ -\bar{d}_u I_n & O_{n,n} & O_{n,n} & I_n \\ O_{n,n} & -I_n & O_{n,n} & I_n \end{bmatrix}, \quad b_5 = \begin{bmatrix} \bar{d}_u \gamma_n \\ O_{n,1} \\ O_{n,1} \end{bmatrix}.$$

5) Consolidation of the Prediction Model Constraints: By combining (51), (55), (57), (60), and (62) we can define

$$M\zeta(k) + E\nu(k) \le b, \tag{63}$$

where

$$M = \begin{bmatrix} M_1 \\ \vdots \\ M_5 \end{bmatrix}, \quad E = \begin{bmatrix} E_1 \\ \vdots \\ E_5 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_5 \end{bmatrix}. \tag{64}$$

The prediction of (63) N steps into the future gives

$$\begin{aligned} M\hat{\zeta}(k_i)_k + E\hat{\nu}(k_i)_k &\le b, \\ M\hat{\zeta}(k_N)_k &\le b, \end{aligned} \tag{65}$$

where $\forall i \in \{0, \cdots, N-1\}$, which can be rewritten as

$$\underline{M}\,\hat{\underline{\zeta}}_k + \underline{E}\,\hat{\underline{\nu}}_k + \underline{M}_0 \zeta(k) \le \underline{b} \tag{66}$$

where

$$\underline{M} = \begin{bmatrix} 0 & \cdots & 0 \\ M & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & M \end{bmatrix}, \quad \underline{E} = \begin{bmatrix} E & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & E \\ 0 & \cdots & 0 \end{bmatrix}, \tag{67}$$

$$\underline{M}_0 = \begin{bmatrix} M \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \underline{b} = \begin{bmatrix} b \\ b \\ \vdots \\ b \end{bmatrix}. \tag{68}$$

E. Model Predictive Control as a Mixed-Integer Linear Programming Optimization Problem

The Model Predictive Control problem presented in the previous sections can be formulated as a Mixed-Integer Linear Programming (MILP) optimization problem. Hereto, consider the control action vector, applied to the system in (31), at time t_k defined as

$$U(k) = \begin{bmatrix} u(k) \\ d_u(k) \end{bmatrix}, \tag{69}$$

where $u(k)$ reflects whether departures are issued from queues and, if so, when after the time t_k they should be issued, as indicated by $d_u(k)$.

Define the optimization vector

$$\lambda_k = \begin{bmatrix} \hat{\underline{\nu}}_k \\ \hat{\underline{\zeta}}_k \end{bmatrix}, \tag{70}$$

with $\hat{\underline{\nu}}_k$ and $\hat{\underline{\zeta}}_k$ defined as in (41), which contains the prediction, N time steps into the future, of the state and input vectors. With this vector define the cost function

$$\gamma(\lambda_k) = \rho\lambda_k, \tag{71}$$

where $\rho = \begin{bmatrix} O_{1,4nN} & \hat{c} \end{bmatrix}$, with $\hat{c}$ defined as in (47). Given this cost function we obtain the optimal α (in which the elements of (69) and their prediction over the entire prediction horizon are embedded) by solving the following MILP problem

$$\lambda_k^* = \arg\min_{\lambda_k} \gamma(\lambda_k), \quad \text{subject to } \begin{cases} \eta(\lambda_k) = 0, \\ \mu(\lambda_k) \le 0, \end{cases} \tag{72}$$

where

$$\begin{aligned} \eta(\lambda_k) &= \begin{bmatrix} -\Omega_1 & I_{2nN} \end{bmatrix} \lambda_k - \Lambda\zeta(k) - \Omega_2 \hat{\underline{\omega}}_k - \Gamma, \\ \mu(\lambda_k) &= \begin{bmatrix} \underline{E} & \underline{M} \end{bmatrix} \lambda_k - \underline{b} + \underline{M}_0 \zeta(k), \end{aligned} \tag{73}$$

with $\zeta(k)$ as in (37); the elements of $\hat{\underline{\omega}}_k$ as in (45); Λ, Ω_1, Ω_2, and Γ as in (42) and (43); and $\underline{M}_0$, $\underline{M}$, $\underline{E}$, and $\underline{b}$ as in (67) and (68). Finally, given this optimal vector, we can define the control action vector in (69) as

$$U(k) = \Phi\lambda_k^*, \tag{74}$$

where $\Phi = \begin{bmatrix} I_{2n} & O_{2n,2n(3N-1)} \end{bmatrix}$.

V. Simulation Results

This section presents the results of two simulation case studies for which deterministic vehicle arrivals are considered. The performance of different access management protocols are compared. The hybrid system, described in Section III, is used to perform the simulations. A discrete-time controller subsystem samples the hybrid system and calculates and executes the control action for that time instant (using a zero-order hold). It is worth noting that the dynamics of the vehicles within the intersection are not taken into account in these simulations, given the assumption of the presence of a CIC strategy realizing constant vehicle velocities. As we mentioned in Section II, the elements of the service time matrix are calculated based on the geometry of the intersection and a given nominal velocity. For instance, in Figure 1, if we consider that the distance $s_4(\tau_4)$, with $\mathcal{T}(\tau_4) = \mathbf{X}_{2,4}$, equals 20 m and we consider a nominal velocity $\check{v} = 10$ m/s, then we conclude that $\tilde{t}_{2,4} = 2$s.

To aid the measurement of the performance, consider the average inter-departure time γ_u for all the queues of the intersection. Moreover, consider the served vehicles counter θ. The average inter-departure time and the served vehicles counter are updated each time a vehicle is served from any queue. On this basis, consider the following hybrid system that describes the dynamics of this performance measure. Define

$$\begin{cases} \dot{\alpha} = \hat{f}(\alpha, \beta), & \text{if } (\alpha, \beta) \in \hat{\mathcal{C}}, \\ \alpha^+ = \hat{g}(\alpha, \beta), & \text{if } (\alpha, \beta) \in \hat{\mathcal{D}}, \end{cases} \tag{75}$$

where the state vector is $\alpha := \begin{bmatrix}\gamma_u & \theta\end{bmatrix}^T$, the input vector is

$$\beta = \begin{bmatrix} O_{n,n} & O_{n,n} & O_{n,n} \\ O_{n,n} & O_{n,n} & I_n \end{bmatrix} z + \begin{bmatrix} O_{n,n} & I_n \\ O_{n,n} & O_{n,n} \end{bmatrix} e, \quad (76)$$

with z and e defined as in (7), such that $\beta := \begin{bmatrix}u^T & t_u^T\end{bmatrix}^T$, where u is the departures vector and t_u is the inter-departure timers vector. Moreover, the flow map is $\hat{f}(\alpha, \beta) := O_{2,1}$, and the jump map is $\hat{g}(\alpha, \beta) = \begin{bmatrix}\hat{g}_1(\alpha, \beta) & \hat{g}_2(\alpha, \beta)\end{bmatrix}^T$, with

$$\hat{g}_1(\alpha, \beta) := \begin{cases} \dfrac{\theta \gamma_u + t_{u,q}}{\theta + 1}, & \text{if } \exists q \in \mathcal{Q}, \text{ such that } u_q = 1, \\ \gamma_u, & \text{otherwise,} \end{cases} \quad (77)$$

and

$$\hat{g}_2(\alpha, \beta) := \begin{cases} \theta + 1, & \text{if } \exists q \in \mathcal{Q}, \text{ such that } u_q = 1, \\ \theta, & \text{otherwise.} \end{cases} \quad (78)$$

Finally, the flow set is $\hat{\mathcal{C}} = \{(\alpha, \beta) | (t_u, u) \in \mathcal{C}_u\}$, and the jump set is $\hat{\mathcal{D}} = \{(\alpha, \beta) | (t_u, u) \in \mathcal{D}_u\}$, with $\mathcal{C}_u$ and $\mathcal{D}_u$ defined as in (15).

Moreover, a First-Come-First-Served (FCFS) access protocol is also considered. As the name indicates, this protocol serves the vehicles in the same order they enter the intersection. Since we consider initial queue lengths different than zero, the serving order of the vehicles in the queues at time zero is defined by determining their arrival time, which is calculated using the arrival time interval matrix W and the initial queue lengths $x(0)$.

In Section V-A, we show the effectiveness of the OIAM methodology applied to a three-lane intersection. Moreover, in Section V-B, we introduce a benchmark example of a real-life five-lane intersection for which we compare the presented methodology with a vehicle actuated traffic light and a FCFS protocol. In both cases, we consider that vehicles arrive to the intersection in a deterministic fashion (for the sake of comparison), and that the queues can be emptied in a finite amount of time.

As a final remark, note that, running on an Intel®Core™i5-3570 processor, we solve the MILP problem, defined in Section IV-E, using the MATLAB `intlinprog()` function which takes the LP solution to the relaxed problem at a node, it rounds the integer components in a way that attempts to maintain feasibility, then, if needed, it searches the neighborhood of the current best integer-feasible solution point (if available) to find a new and better solution.

A. Three-Lane Intersection Scenario

We propose a scenario based on the intersection depicted in Figure 3. The road formed by lanes one and three is hereafter referred to as the main road, whereas the road formed by lane two is hereafter referred to as the secondary road. Note that this scenario is proposed in [30], which in turn inspired the work in [15].

In an intersection controlled by a traffic light the whole flow of the main road has to be stopped to give access to the secondary road. The automation of the intersection, presented

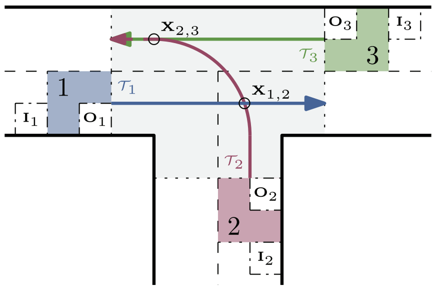

Fig. 3. Three-lane intersection.

in [15], allows for a continuous flow of vehicles by means of virtual platoons, which are formed by an access management protocol on the basis of a First-Come-First-Served principle.

To define the intersection problem as a queuing problem consider the service times matrix, the values of which are defined in (5), given by

$$T^s = \begin{bmatrix} 1.25 & 2.95 & 0 \\ -0.45 & 1.25 & 1.92 \\ 0 & 0.58 & 1.25 \end{bmatrix} \quad (79)$$

which is calculated using $\tilde{t}_{1,2} = 5.15$, $\tilde{t}_{2,1} = 3.46$, $\tilde{t}_{2,3} = 6.92$, $\tilde{t}_{3,2} = 6.25$, and $\bar{t}_h = 1.25$. Note that these values (which are merely illustrative) are chosen to represent an intersection of relatively small physical dimensions, and were calculated by fixing a value for $\tilde{t}_{1,2}$ and using it as the scale of the geometry depicted in Figure 3. Moreover, the value of $\bar{t}_h$ is chosen to represent a close vehicle following within the intersection which can be achieved with virtual platooning, see [15]. To finalize the problem setting, consider that the intersection has initial queue lengths $x(0) = \begin{bmatrix}20 & 10 & 20\end{bmatrix}^T$, and that the arrival time interval matrix is given by $W = \begin{bmatrix}4 & 6 & 3\end{bmatrix}^T$.

To analyze the performance of the presented methodology, consider three access management protocols. Namely, the aforementioned FCFS protocol [15], and the OIAM protocol considering both an On-Sampling-Event (OSE) representation [26], and an Inter-Sampling-Event (ISE) representation. Both the OSE and the ISE representations are described in Section III. Note that we can achieve the OSE representation by setting $\bar{d}_u = 0$ in (61). Moreover, in both cases, we consider a cost vector $c_x = \begin{bmatrix}1 & 1 & 1\end{bmatrix}$, and a prediction time $t_N = 4\,\text{s}$, with sampling interval $\Delta t = 0.5\,\text{s}$ and prediction horizon $N = 8$.

Figure 4 shows the queue lengths' response for three access protocols, all of which reach a stable steady state (meaning that the queue lengths remain in the vicinity of zero). Moreover, Figure 5 shows the average inter-departure time γ_u for each access protocol. Note that all approaches settle at the same value due to the lack of saturation of the queues. The main difference between these protocols is the settling time which is influenced by γ_u. Note that the FCFS approach shows a high value of γ_u which is a result of the instantaneous reactiveness of this approach; the vehicles are granted access in the same order as they requested it. The benefits of the MPC

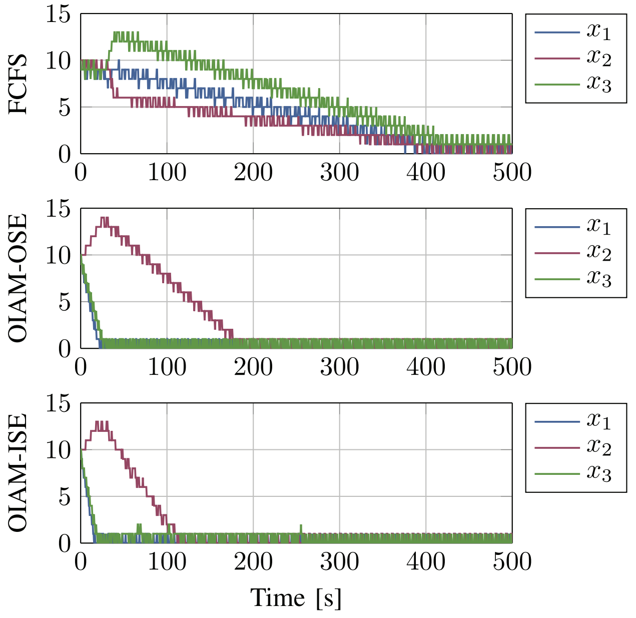

Fig. 4. Queue lengths of a three-lane intersection for different access management protocols. Namely, First-Come-First-Served (FCFS), On-Sampling-Event (OSE), and Inter-Sampling-Event (ISE). For both OSE and ISE the sampling time is $\Delta t = 0.5$ s.

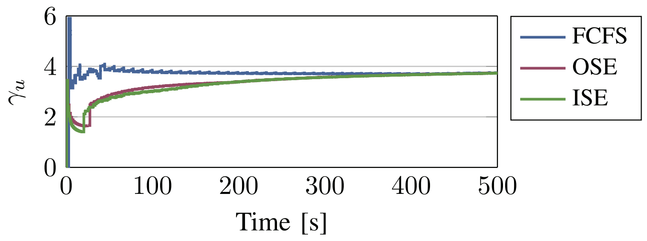

Fig. 5. Average inter-departure time for different access management protocols. Namely, First-Come-First-Served (FCFS), On-Sampling-Event (OSE), and Inter-Sampling-Event (ISE).

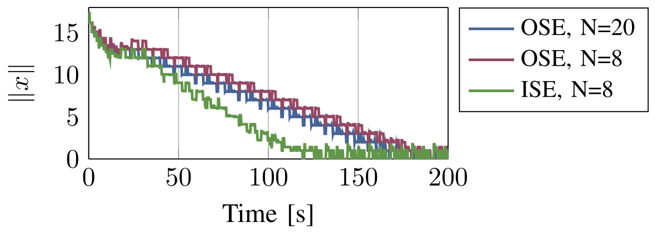

Fig. 6. Norm of the queue lengths vector for different settings of the MPC approach.

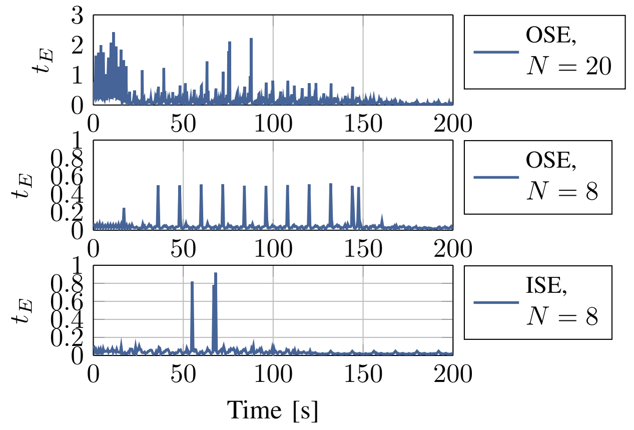

Fig. 7. Execution time of the MILP solver for different event representations.

TABLE I
AVERAGE EXECUTION TIMES FOR DIFFERENT MPC SETTINGS

Type	t_N	N	Δt	$\bar{t}_E$
ISE	4 s	8	500 ms	39.7 ms
OSE	4 s	8	500 ms	49.4 ms
OSE	4 s	20	200 ms	184.3 ms

approach is noticeable in the response of both the OSE and ISE representations. The OSE representation is limited to grant access just at sampling instances which limits its performance. The benefit of the ISE representation is evident, showing a faster settling time and a lower average inter-departure time.

To study the influence of the selection of the sampling interval Δt and the prediction horizon N on the performance of the OIAM approach, consider the following. Keeping the prediction time constant, if the sampling time would be small enough we should achieve the same performance from the OSE and the ISE representations. The trade-off is that the prediction horizon N has to be increased which in turn increases the size of the MILP problem to be solved. Figure 6 shows the norm of the queue lengths vector $\|x\|$ for different settings of the MPC approach. Additionally, the execution time t_E (which we define as the time it takes to solve the MILP problem) is depicted in Figure 7. Note that the settings for each case are presented in Table I.

It can be seen (in Figure 6) that the performance of the OSE representation diverges from the ISE representation even when the sampling time has been reduced to a point in which solving the optimization problem takes, in average, as much as the sampling time itself. Compare the sampling time Δt with the average execution time $\bar{t}_E$ in Table I. On the other hand, we achieve a good performance by solving a small optimization problem with an ISE representation. Figure 7, shows that the ISE representation shows a somewhat consistent execution time (despite having some peaks that exceed the sampling time) which suggests that a real-time implementation of this approach is feasible. Moreover, if a real-time implementation is pursued, then an analysis on the most effective way to solve the MILP problem should be performed, such that the execution time can be minimized.

B. Five-Lane Intersection

To further benchmark the performance of the presented methodology we introduce the intersection depicted in Figure 8. This intersection is a representation of a real-life intersection in the Netherlands, labeled S4 in [31], which is controlled with traffic lights. Using the service rates and set-up

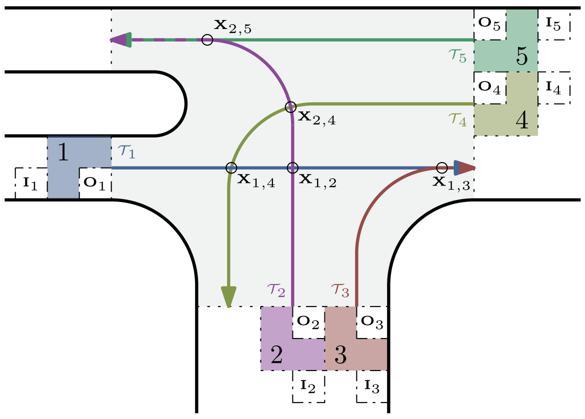

Fig. 8. Real-life five-lane intersection.

TABLE II
OPTIMAL CYCLE FOR THE VEHICLE ACTUATED TRAFFIC LIGHT

Mode	Queues served	Serve until
1	3, 4, 5	Queues 3 and 4 are empty
2	1, 5	Both queues are empty
3	2, 3	Queue 2 is empty

times in [31] we can construct the matrix

$$\hat{T}^s = \begin{bmatrix} 2 & 4 & 4 & 5 & 0 \\ 5 & 2 & 0 & 4 & 6 \\ 5 & 0 & 2.2 & 0 & 0 \\ 5 & 5 & 0 & 2 & 0 \\ 0 & 4 & 0 & 0 & 1.9 \end{bmatrix}, \tag{80}$$

the elements of which, given by $(\hat{T}^s)_{ab}$, represent the time needed to greenlight lane a after lane b has been stopped by a red light. Moreover, the arrival rates, also defined in [31], are given by $W = \begin{bmatrix}5.11 \ 21.56 \ 18.56 \ 21.95 \ 9.73\end{bmatrix}^T$. Note that for this scenario, a FCFS, an OIAM, and an Optimal Traffic Light (OTL) protocols are considered.

The aforementioned OTL protocol is a vehicle actuated traffic light scenario which operates using an optimal cycle, shown in Table II, determined based on [32].

To compare the OTL approach with the FCFS approach, and the proposed OIAM approach we need to define the service time matrix in (5). Since the dimensions of the intersection are not defined we need to adopt some assumptions. First, we assume that $\tilde{t}_{2,5} = \max \hat{T}^s$ which, by comparison, makes $\tilde{t}_{1,2} = 3.41$, $\tilde{t}_{1,3} = 6.20$, $\tilde{t}_{1,4} = 2.20$, $\tilde{t}_{2,1} = 2.64$, $\tilde{t}_{2,4} = 3.84$, $\tilde{t}_{3,1} = 3.52$, $\tilde{t}_{4,1} = 5.38$, $\tilde{t}_{4,2} = 3.59$, and $\tilde{t}_{5,2} = 5.12$. Next, we assume that $\bar{t}_h = 2$ (as in the traffic light case) which could be set lower to consider cooperative autonomous vehicles, defined in [15]. The result of the aforementioned assumptions is the following service time matrix:

$$T^s = \begin{bmatrix} 2 & 2.77 & 4.66 & -1.15 & 0 \\ 1.23 & 2 & 0 & 2.25 & 2.88 \\ -0.66 & 0 & 2 & 0 & 0 \\ 5.15 & 1.75 & 0 & 2 & 0 \\ 0 & 1.12 & 0 & 0 & 2 \end{bmatrix}. \tag{81}$$

Moreover, we set the parameters of proposed MPC-based OIAM approach as follows. Sampling interval $\Delta t = 1$ s, prediction horizon $N = 5$, and cost vector $c_x = \begin{bmatrix}1 \ 1 \ 1 \ 1 \ 1\end{bmatrix}$.

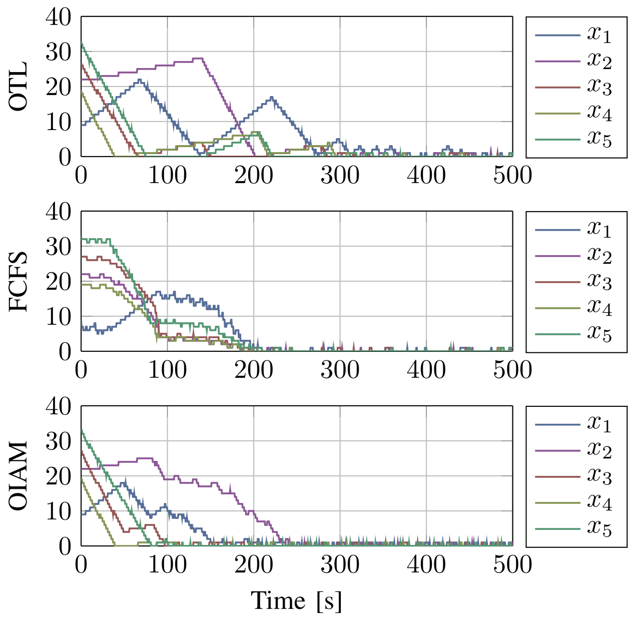

Fig. 9. Queue lengths of a five-lane intersection for different access management protocols. Namely, Optimal Traffic Lights (OTL), First-Come-First-Served (FCFS), and Inter-Sampling-Event (ISE).

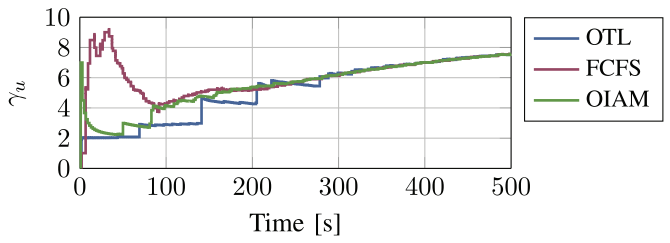

Fig. 10. Average inter-departure time for different access management protocols. Namely, Optimal Traffic Lights (OTL), First-Come-First-Served (FCFS), and Inter-Sampling-Event (ISE).

Finally, the initial queue lengths for the following comparison are set as $x(0) = \begin{bmatrix}33 \ 19 \ 27 \ 22 \ 9\end{bmatrix}^T$.

Figure 9 and Figure 10 show the queue lengths and the average inter-departure times, respectively, for all the proposed protocols. Both the FCFS and the OIAM approaches show a faster settling time in comparison with the OTL approach; see Figure 9. This is because both the FCFS and the OIAM protocols allow for a continuous serving of vehicles, whereas the OTL protocol just allows some queues to be served at the same time. However, from Figure 10, we see that the FCFS protocol shows a high average inter-departure time in the first hundred seconds due to the continuous switching of the server between queues. In this respect the OIAM and the OTL protocols achieve a response that minimizes the average inter-departure. Above all, the proposed OIAM approach achieves both a fast settling time and a minimum average inter-departure time with an average execution time $\bar{t}_E = 26.6$ ms, which suggest that a real-time implementation is feasible. Note that the implementation would require a centralized unit which receives messages from the vehicles approaching the intersection, determines the crossing sequence by means of the OIAM approach, and communicates such sequence back to the vehicles to execute.

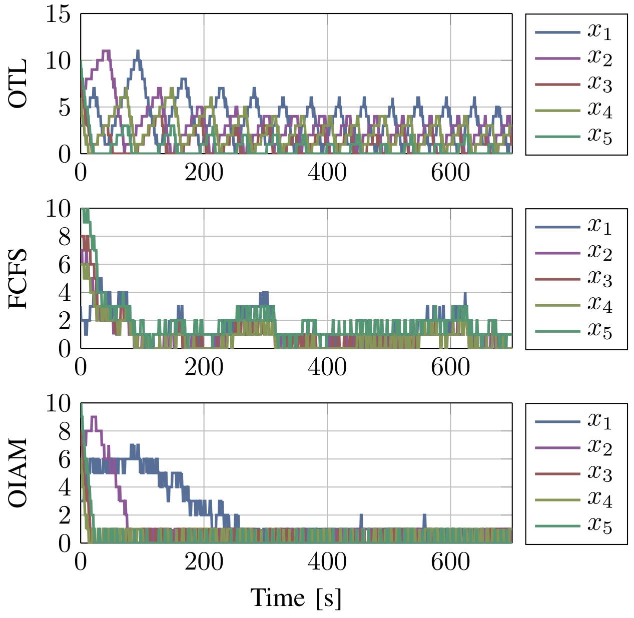

Fig. 11. Queue lengths of a five-lane intersection, with high arrival rate, for different access management protocols. Namely, Optimal Traffic Lights (OTL), First-Come-First-Served (FCFS), and Inter-Sampling-Event (ISE).

Finally, Figure 11 shows the queue lengths for the higher arrival rate $W = \left[5.11\ 8.98\ 9.23\ 9.15\ 9.73\right]^T$, with initial queue lengths $x(0) = \left[3\ 7\ 8\ 6\ 10\right]^T$. We observe that the OIAM approach is able to keep the queues empty while the OTL and FCFS reach a steady-state cycle that resembles a normal traffic light scenario. With this example, we show the proactive nature of the OIAM approach which adapts to the traffic changes.

VI. Conclusions

This paper presents an optimal access management methodology for a cooperative intersection control system, developed in [15]. The road intersection is modeled as a hybrid dynamical queuing system which, in turn, is controlled using Model Predictive Control which allows for inter-sampling control actions. The proposed Optimal Intersection Access Management strategy minimizes the weighted sum of the queue lengths which translates into minimizing the time that vehicles wait to be given access to the intersection. The performance of this methodology is assessed by two simulation studies. The impact of design parameter tuning is shown for a three-lane intersection. For which we also analyze the computational benefits of the inter-sampling event representation. Moreover, a comparison between the performance of the proposed strategy against an optimal vehicle actuated traffic light controller, and a First-Come-First-Served access protocol is shown for a real-life five lane intersection. The presented access management methodology shows a satisfactory performance with a low average execution time, which is a result of the inter-sampling control action definition of the system discretization.

The results presented in this paper assume that the vehicles waiting to gain access to the intersection form static queues. Such assumption neglects the dynamics of the vehicles crossing the intersection. Considering such vehicle dynamics leads to a queuing problem with either a time-varying service matrix, or time-varying constraints. The implication of such consideration and its application are subject of current research.

Appendix

The generic form of a hybrid dynamical system, as defined in [33], [34], is given by

$$\begin{cases} \dot{z} = f(z, e), & \text{if } (z, e) \in \mathcal{C}, \\ z^+ = g(z, e), & \text{if } (z, e) \in \mathcal{D}, \end{cases} \tag{82}$$

where $z \in \mathbb{R}^{\alpha \times 1}$ is the state vector, $\dot{z}$ is the rate of change of the state z, z^+ is the value of the state after an instantaneous change, $e \in \mathbb{R}^{\beta \times 1}$ is the input vector, $f(z, e)$ is the flow map (or vector field), $g(z, e)$ is the jump map, $\mathcal{C}$ is the flow set, and $\mathcal{D}$ is the jump set. The solutions of (82) are defined on the hybrid time domain $(t, j) \in \mathbb{E}$, where t represents continuous time, and j represents the event counter. The subset $\mathbb{E} \subset \mathbb{R}_{\geq 0} \times \mathbb{N}$ is called a *compact hybrid time domain* if $\mathbb{E} = \bigcup_{j=0}^{J}([t_j, t_{j+1}], j)$ for some finite sequence of times $0 = t_0 \leq \cdots \leq t_{J+1}$. We say E is a *hybrid time domain* if, for each $(T, J) \in \mathbb{E}$, the set $\mathbb{E} \cap ([0, T] \times \{0, \cdots, J\})$ is a compact hybrid time domain.

A *hybrid signal* is a function defined on a hybrid time domain. A hybrid signal $e : \operatorname{dom} e \mapsto \mathbb{R}^{\beta \times 1}$ is called a *hybrid input* if $e(\cdot, j)$ is Lebesgue measurable and locally essentially bounded for each j. A hybrid signal $z : \operatorname{dom} z \mapsto \mathbb{R}^{\alpha \times 1}$ is called a *hybrid arc* if $z(\cdot, j)$ is locally absolutely continuous for each j. A hybrid arc $z : \operatorname{dom} z \mapsto \mathbb{R}^{\alpha \times 1}$ and a hybrid input $e : \operatorname{dom} e \mapsto \mathbb{R}^{\beta \times 1}$ form a *solution pair* $\phi = (z, e)$ to (6) if $\operatorname{dom} z = \operatorname{dom} e$, $\big(z(0, 0), e(0, 0)\big) \in \mathcal{C} \cup \mathcal{D}$, and

1) for all $j \in \mathbb{N}$ and almost all t such that $(t, j) \in \operatorname{dom} \phi$, $\big(z(t, j), e(t, j)\big) \in \mathcal{C}$, and $\dot{z}(t, j) = f\big(z(t, j), e(t, j)\big)$;
2) for all $(t, j) \in \operatorname{dom} \phi$ such that $(t, j + 1) \in \operatorname{dom} \phi$, $\big(z(t, j), e(t, j)\big) \in \mathcal{D}$, and $z(t, j+1) = g\big(z(t, j), e(t, j)\big)$.

Hereafter, we refer to the solution pair simply as the solution.

References

[1] *World Urbanization Prospects: The 2014 Revision, (ST/ESA/SER.A/366)*, United Nations, Dept. Econ. Social Affairs, Population Division, New York, NY, USA, 2014.

[2] Z. Ying-nian and H. Qi-fu, "Intersection approach road congestion index and application for Beijing, China," in *Proc. 2nd IEEE Int. Conf. Inf. Manage. Eng.*, Apr. 2010, pp. 263–266.

[3] B. Singh and A. Gupta, "Recent trends in intelligent transportation systems: A review," *J. Transp. Literature*, vol. 9, no. 2, pp. 30–34, Apr. 2015.

[4] L. Chen and C. Englund, "Cooperative intersection management: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 570–586, Feb. 2016.

[5] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *J. Artif. Intell. Res.*, vol. 31, pp. 591–656, Mar. 2008.

[6] M. Ahmane *et al.*, "Modeling and controlling an isolated urban intersection based on cooperative vehicles," *Transp. Res. C, Emerg. technol.*, vol. 28, pp. 44–62, Mar. 2013.

[7] A. Colombo and D. D. Vecchio, "Least restrictive supervisors for intersection collision avoidance: A scheduling approach," *IEEE Trans. Autom. Control*, vol. 60, no. 6, pp. 1515–1527, Jun. 2015.

[8] P. Tallapragada and J. Cortés, "Coordinated intersection traffic management," *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 233–239, 2015.

[9] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 81–90, Mar. 2012.

[10] J. Gregoire, S. Bonnabel, and A. de La Fortelle, “Priority-based coordination of robots,” 2014.

[11] H. Kowshik, D. Caveney, and P. R. Kumar, “Provable systemwide safety in intelligent intersections,” *IEEE Trans. Veh. Technol.*, vol. 60, no. 3, pp. 804–818, Mar. 2011.

[12] M. R. Hafner, D. Cunningham, L. Caminiti, and D. D. Vecchio, “Cooperative collision avoidance at intersections: Algorithms and experiments,” *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1162–1175, Sep. 2013.

[13] M. A. S. Kamal, J.-I. Imura, T. Hayakawa, A. Ohata, and K. Aihara, “A vehicle-intersection coordination scheme for smooth flows of traffic without using traffic lights,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1136–1147, Jun. 2015.

[14] D. Miculescu and S. Karaman, “Polling-systems-based control of high-performance provably-safe autonomous intersections,” in *Proc. IEEE 53rd Conf. Decision Control*, Dec. 2014, pp. 1417–1423.

[15] A. I. Morales Medina, N. van de Wouw, and H. Nijmeijer, “Cooperative intersection control based on virtual platooning,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1727–1740, Jun. 2018.

[16] R. Hult, M. Zanon, S. Gros, and P. Falcone, “Primal decomposition of the optimal coordination of vehicles at traffic intersections,” in *Proc. IEEE 55th Conf. Decision Control (CDC)*, Dec. 2016, pp. 2567–2573.

[17] A. Katriniok, P. Kleibaum, and M. Joševski, “Distributed model predictive control for intersection automation using a parallelized optimization approach,” *IFAC PapersOnLine*, vol. 50, no. 1, pp. 5940–5946, Jul. 2017.

[18] X. Qian, J. Gregoire, A. de L. Fortelle, and F. Moutarde, “Decentralized model predictive control for smooth coordination of automated vehicles at intersection,” in *Proc. Eur. Control Conf. (ECC)*, Jul. 2015, pp. 3452–3458.

[19] Y. Zheng *et al.*, “Research on cooperative vehicle intersection control scheme without using traffic lights under the connected vehicles environment,” *Adv. Mech. Eng.*, vol. 9, no. 8, pp. 1–13, Aug. 2017.

[20] L. Riegger, M. Carlander, N. Lidander, N. Murgovski, and J. Sjöberg, “Centralized MPC for autonomous intersection crossing,” in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 1372–1377.

[21] J. Ding, H. Xu, J. Hu, and Y. Zhang, “Centralized cooperative intersection control under automated vehicle environment,” in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2017, pp. 972–977.

[22] D. Heidemann and H. Wegmann, “Queueing at unsignalized intersections,” *Transp. Res. B, Methodol.*, vol. 31, no. 3, pp. 239–263, Jun. 1997.

[23] A. C. Soh, M. H. Marhaban, M. Khalid, and R. Yusof, “Modelling and optimisation of a traffic intersection based on queue theory and Markov decision control methods,” in *Proc. 1st Asia Int. Conf. Modelling Simulation (AMS)*, Mar. 2007, pp. 478–483.

[24] H. Y. Sutarto, M. Maulida, E. Joelianto, and A. Samsi, “Queue length optimization of vehicles at road intersection using parabolic interpolation method,” in *Proc. Int. Conf. Autom., Cognit. Sci., Opt., Micro Electro-Mech. Syst., Inf. Technol.(ICACOMIT)*, Oct. 2015, pp. 63–67.

[25] T. S. Babicheva, “The use of queuing theory at research and optimization of traffic on the signal-controlled road intersections,” *Procedia Comput. Sci.*, vol. 55, pp. 469–478, Jan. 2015.

[26] F. Creemers, A. I. M. Medina, E. Lefeber, and N. van de Wouw, “Design of a supervisory controller for cooperative intersection control using model predictive control,” in *Proc. 5th IFAC Conf. Anal. Control Chaotic Syst.*, Jan. 2018, pp. 63–68.

[27] E. F. Camacho and C. Bordons, *Model Predictive Control*. Berlin, Germany: Springer, 2007.

[28] A. Gupte, S. Ahmed, M. S. Cheon, and S. Dey, “Solving mixed integer bilinear problems using MILP formulations,” *SIAM J. Optim.*, vol. 23, no. 2, pp. 721–744, Apr. 2013.

[29] D. Gross, J. F. Shortie, J. M. Thompson, and C. M. Harris, *Fundamentals of Queueing Theory*. Hoboken, NJ, USA: Wiley, 2013.

[30] J. Ploeg *et al.*, “Cooperative automated maneuvering at the 2016 grand cooperative driving challenge,” *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1213–1226, Apr. 2018.

[31] S. T. G. Fleuren and A. A. J. Lefeber, “Data of real-life intersections for fixed-time traffic light control,” Eindhoven Univ. Technol., Eindhoven, The Netherlands, Tech. Rep., 2016.

[32] S. T. G. Fleuren, “Optimizing pre-timed control at isolated intersections,” Ph.D. dissertation, Dept. Mech. Eng. Faculty, Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2017.

[33] C. Cai and A. R. Teel, “Characterizations of input-to-state stability for hybrid systems,” *Syst. Control Lett.*, vol. 58, no. 1, pp. 47–53, Jan. 2009.

[34] R. Goebel, A. R. Teel, and R. G. Sanfelice, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton, NJ, USA: Princeton Univ. Press, 2012.

Alejandro Ivan Morales Medina received the B.Sc. degree in mechatronics from UPIITA-IPN, Mexico City, Mexico, in 2009, and the M.Sc. degree in automotive technology with a dynamics and control specialization from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 2013, where he is currently pursuing the Ph.D. degree. His current research is on the cooperative motion of vehicles through road intersections, including the control strategies to control the longitudinal and lateral motion of the vehicles, and the optimization of the traffic flow.

Falco Creemers received the B.Sc. and M.Sc. degrees from the Department of Mechanical Engineering, Eindhoven University of Technology, The Netherlands, in 2014 and 2017, respectively. He is currently employed at the R&D Department, Océ-Technologies B.V., Venlo, The Netherlands.

Erjen Lefeber received the M.Sc. degree in applied mathematics from the University of Twente, Enschede, The Netherlands, in 1996, and the Ph.D. degree from the University of Twente, in 2000, on the subject of tracking control of nonlinear mechanical systems. Since 2000, he has been an Assistant Professor at the Department of Mechanical Engineering, Eindhoven University of Technology. From 2000 to 2015, he worked on modeling and control of manufacturing systems, but in 2015, he joined the Dynamics and Control Group. His current research is nonlinear control theory. In particular, the control of drones and the control of platooning vehicles.

Nathan van de Wouw received the M.Sc. (Hons.) and Ph.D. degrees in mechanical engineering from the Eindhoven University of Technology, Eindhoven, The Netherlands, in 1994 and 1999, respectively. He currently holds a full professor position at the Mechanical Engineering Department, Eindhoven University of Technology. He also holds an adjunct full professor position at the University of Minnesota, USA, and a (part-time) full professor position at the Delft University of Technology, The Netherlands. He has been working at Philips Applied Technologies, Eindhoven, in 2000, and he has been working at The Netherlands Organisation for Applied Scientific Research (TNO), Delft, The Netherlands, since 2001. He has held positions as a Visiting Professor at the University of California Santa Barbara, USA, in 2006 and 2007, at the University of Melbourne, Australia, in 2009 and 2010, and at the University of Minnesota, in 2012 and 2013. He has published a large number of journals and conference papers and the books *Uniform Output Regulation of Nonlinear Systems: A convergent Dynamics Approach* with A.V. Pavlov and H. Nijmeijer (Birkhauser, 2005) and *Stability and Convergence of Mechanical Systems with Unilateral Constraints* with R.I. Leine (Springer-Verlag, 2008). His current research interests are the modeling analysis and control of nonlinear/hybrid systems, with applications to vehicular platooning, high-tech systems, resource exploration, smart energy systems, and networked control systems. In 2015, he received the IEEE Control Systems Technology Award “For the development and application of variable-gain control techniques for high-performance motion systems.” He is currently an Associate Editor for the journals *Automatica* and the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY.