Polling-systems-based Control of High-performance Provably-safe Autonomous Intersections

David Miculescu  Sertac Karaman

Abstract—The rapid development of autonomous vehicles spurred a careful investigation of the potential benefits of all-autonomous transportation networks. Most studies conclude that autonomous systems can enable drastic improvements in performance. A widely studied concept is all-autonomous, collision-free intersections, where vehicles arriving in a traffic intersection with no traffic light adjust their speeds to cross through the intersection as quickly as possible. In this paper, we propose a coordination control algorithm for this problem, assuming stochastic models for the arrival times of the vehicles. The proposed algorithm provides provable guarantees on safety and performance. More precisely, it is shown that no collisions occur surely, and moreover a rigorous upper bound is provided for the expected wait time. The algorithm is also demonstrated in simulations. The proposed algorithms are inspired by polling systems. In fact, the problem studied in this paper leads to a new class of polling systems where customers are subject to differential constraints, which may be interesting in its own right.

I. INTRODUCTION

Autonomous systems technology holds the potential to provide substantial performance increases in transportation systems, independently of the application domain. For example, robotic vehicles servicing packaging requests in large Amazon warehouses already increase efficiency [1], [2]. Similar autonomous vehicles provide material handling services in seaports [3]–[5]. In both application domains, the vehicles coordinate their motion in order to share common resources, in this case roads and intersections, as effectively as possible, for example to minimize transportation times.

Encouraged by the rapid development of autonomous, driverless cars, similar technologies may be considered for urban transportation networks, where cars coordinate their motion to reduce delays while enhancing safety. Consider, for example, vehicles arriving near a traffic intersection where a central control system adjusts their speeds to lead them through the intersection as quickly as possible. In the absence of traffic lights, when the only limiting factor is avoiding collision with other vehicles, the vehicles’ motion can be carefully adjusted, to ensure that the intersection is crossed as quickly as possible. This may require, for instance, that the vehicles traveling in the same lane form series of small platoons, whereas the vehicles from different lanes cross the intersection in close proximity to one another, while traveling at high speeds.

The importance of this problem has not gone unnoticed. Coordination control systems based on auction algorithms [6], multi-agent simulation [7], [8], genetic algorithms [9] and token-based approaches [10] were proposed very recently, and implementations using vehicle-to-vehicle communication networks were considered [11]. Similar problems were also studied from the perspective of hybrid control systems [12], [13] and air traffic management [14], [15].

All of the aforementioned approaches use computational experiments to show that the proposed algorithms are safe and they provide good performance. In many cases, the improvement on all-autonomous intersections is so drastic that it leads to several orders of magnitude reduction in average delay for vehicles crossing the intersection. Although the results of the computational experiments are very encouraging, to the best of our knowledge, optimal algorithms and mathematically rigorous performance bounds are not known.

The main contribution of this paper is a motion coordination algorithm for all-autonomous intersections. Our algorithm is based on the polling systems literature, and it provides provable guarantees on safety and performance. In fact, our algorithm uses a particular polling policy at its core, and its performance is tied to the performance of this polling policy. Roughly speaking, we show that no collisions occur at all times, and moreover, the delay each customer experiences is bounded by the delay in a corresponding polling policy executed on a traditional polling system. The latter result implies an analytical bound on the average delay.

Let us note that the polling systems literature is fairly rich [16]–[20]. Motivated by applications in communication systems, transportation systems, and manufacturing, the literature has flourished during the last few decades. Analytical expressions were derived for a range of polling policies [16]–[18], and these foundational results have been utilized in a variety of application domains [19], [20]. The traditional applications of polling systems include urban traffic flows [19].

However, to the best of the authors’ knowledge, the applications of polling systems in the context of all-autonomous traffic intersections is novel. Furthermore, the problem formulation presented in this paper can be generalized leading to a new class of polling systems where the customers are subject to differential constraints, which may be interesting on its own right. In this case, the customers must be “steered” to a suitable state before they can be serviced. Our results imply that, in a certain class of such polling systems, the differential constraints can be managed, i.e., the differentially-constrained polling system can achieve the performance that its counterpart with no differential constraints achieves.
Although the present paper focuses on applications in urban transportation, let us emphasize that potential application areas also include air transportation as well as warehouse automation and manufacturing. In particular, trajectory-based operations considered for the NextGen air transportation system (see, e.g., [21]) by the Federal Aviation Administration in the U.S. will enable trajectory planning and precision execution for aircraft. An effective use of the shared airspace may be possible by setting up virtual roads and intersections, where the aircraft coordinate their motion for increased performance. Furthermore, autonomous robotic vehicles servicing warehouses, factories, and transportation hubs (e.g., airports, seaports, train stations, etc.) may enable efficient transportation of goods and people with the help of effective motion coordination algorithms.

The paper is organized as follows. The problem formulation is presented in Section II. The coordination algorithm is presented in Section III, where queuing systems are polling systems are also briefly introduced. The proposed algorithm is analyzed in Section IV, and the results of computational experiments are presented in Section V. The paper is concluded with remarks in Section VI.

II. PROBLEM DEFINITION

Consider a traffic intersection where two orthogonal lanes intersect. Suppose each vehicle is subject to second order dynamics of the following form:

\[ \ddot{x}(t) = u(t), \]  

where \( x(t) \) denotes the position of the front bumper of the vehicle, \( 0 \leq \dot{x}(t) \leq v_m \) is the maximum velocity constraint, and \( |u(t)| \leq a_m \) is the maximum acceleration constraint. The region where the two lanes intersect is called the intersection region. The portion of the road within a distance of \( L \) to the intersection is called the control region. We assume that, once a vehicle is inside the control region, it is controlled by a central control system. In other words, the input signal \( u(t) \) is directly determined by a central control system for all vehicles that are in the control region.

This central control system does not know a priori the precise times that each vehicle will arrive at the control region. More precisely, we model the arrival times \( \{t_{i,k} : i \in \mathbb{N}\} \) as a suitable stochastic process, where \( t_{i,k} \) is the time that the \( i \)th vehicle enters the control region (front bumper is exactly a distance of \( L \) away from the end of the intersection) from lane \( k \in \{1, 2\} \). In other words, the \( i \)th vehicle entering the control region from lane \( k \) is at position \(-L\) at time \( t_{i,k} \), i.e., \( x_i(t_{i,k}) = -L \), where \( x_i(t) \) denotes the position of this vehicle at time \( t \). From this point on, the position \( x_i(t) \) of the same vehicle is governed according to the dynamics given in Equation (1). If the lane number of the \( i \)th vehicle is clear from context, we simplify the notation and drop the lane subscript, i.e., \( x_i \) denotes the trajectory of the \( i \)th vehicle, \( t_i \) denotes the time the \( i \)th vehicle enters the control region, et cetera. We assume that the vehicles enter the control region with maximum speed, i.e., \( \dot{x}(t_{i,k}) = v_m \) for all \( i \in \mathbb{N} \) and all \( k \in \{1, 2\} \). To simplify notation, the \( i \)th vehicle that enters lane \( k \) is sometimes denoted as vehicle \( (i, k) \).

We represent each vehicle as a two-dimensional, rectangle-shaped rigid body with length \( l \) and width \( w \). The position of this rigid body is encoded with respect to one of the corners of the intersection area. See Figure 1. The orientation of this rigid body depends on which lane the vehicle is traveling in. To formalize, let us define the rigid body at position \( z \in \mathbb{R} \) in lane \( k \) with \( R(z, k) \subset \mathbb{R}^2 \), i.e., \( R(z, 1) := \{(y_1, y_2) \in \mathbb{R} : z - l < y_1 < z, 0 < y_2 < w\} \), and \( R(z, 2) := \{(y_1, y_2) \in \mathbb{R} : 0 < y_1 < w, z - l < y_2 < z\} \). Then, the \( i \)th vehicle entering lane \( k \) at time \( t \) is represented by \( R(x_{i,k}(t), k) \).

Let \( I_k(t) \) denote the indices for all vehicles that are inside the control region at time \( t \) and in lane \( k \). For instance, if the 3rd, 4th, and 5th vehicles are in lane 1 at time \( t \), then we have \( I_1(t) = \{3, 4, 5\} \). Clearly, \( I_k(t) \) is a set of consecutive natural numbers for all \( t \geq 0 \) and all \( k \in \{1, 2\} \).

Definition II.1 (Safety) The control region is said to be safe at time \( t \in \mathbb{R}_{\geq 0} \), if there are no pairwise collisions among the vehicles, i.e., \( R(x_{i,k}(t), k) \cap R(x_{j,l}(t), l) = \emptyset \) for all \( i \in I_k \), all \( j \in I_l \) and all \( k, l \in \{1, 2\} \).

For each vehicle, we define the delay incurred in transitioning through the intersection as follows. Recall that \( t_{i,k} \) is the time that the \( i \)th vehicle enters lane \( k \). Let \( T_{i,k} \) denote the time that the same vehicle leaves the control region, that is, the rear bumper of the vehicle is outside the intersection region. More precisely, \( T_{i,k} \) is the instance for which \( x_i(T_{i,k}) = l + w \). Then, the time it takes for the same vehicle to transition the control region is \( T_{i,k} - t_{i,k} \). We define the delay as the difference between this time and the time it takes the vehicle to go through the control region if this region were empty:

\[ D_{i,k} := (T_{i,k} - t_{i,k}) - \frac{L + l + w}{v_m}, \]

where \( D_{i,k} \) denotes the delay and \( (L + w + l) / v_m \) is the time it would take the vehicle to traverse the control region if there were no other cars in the control region.

We are interested in developing coordination algorithms that govern this central controller such that both performance (for example, in terms of delay) and safety (avoiding collisions between vehicles) are simultaneously guaranteed. We will discuss such algorithms in the next section.

At this point, we tacitly leave out an important part of the problem formulation: we do not specify the distribution of the stochastic processes \( \{t_{i,k} : i \in \mathbb{N}\} \). We will identify this as a part of our model and assumptions in Section IV.
III. CONTROL POLICY

In this section, we propose a control policy for the intersection coordination problem we introduced in Section II. This policy is based on the control policies for polling systems [18]. Before describing the policy, we briefly introduce queueing and polling systems and their analysis below.

A. Queueing and Polling Systems

Loosely speaking, queueing theory studies the behavior of wait lines and queues. One of the widely studied mathematical models can be described as follows. Suppose customers arrive at the system, where their requests are processed by one server. Let $t_i$ denote the arrival time of the $i$th customer. Let $s_i$ denote the time it takes for this customer to be serviced by the server, once it gets the server’s attention. Often, both $\{t_i : i \in \mathbb{N}\}$ and $\{s_i : i \in \mathbb{N}\}$ are stochastic processes. Suppose the customers are serviced by the server one at a time, in the order that they arrive. Then, given the statistics of arrival times and service times, what is the time average queue length (number of customers who have arrived, but have not yet been serviced) or what is the average wait time for a typical customer? Queueing theory aims to answer these questions for a variety of queueing models. The queueing theory literature has found profound applications in a number of domains [22], including urban traffic [23].

Polling systems are extensions of queueing systems, where the servers service multiple sets of customers arriving in different queues. The server may choose to serve a customer from any queue. However, the server must pay a set-up cost each time it serves customers coming from a queue that is different than the queue of the previous customer.

Central to polling systems is a controller that decides which queue to serve next. This decision is a determining factor for the performance of the system, for instance, in terms of the average delay or the queue length. In either case, the control policy must trade off the following two. On the one hand, it should switch between different queues often enough, in order to ensure customers in one queue do not wait too long for the server to process customers in a different queue; on the other hand, the server should not switch too often in order not to incur too much set-up cost.

Consider a polling system with two queues and one server. Let $t_{i,k}$ denote the time that customer $i$ arrives in queue $k$, and let $s_{i,k}$ denote the amount of time it takes the server to service this customer, where $k \in \{1,2\}$. The customers are serviced by the server one at a time. Switching from one queue to the other requires a set-up time, say $r$, which is also a random variable. That is, if the server last serviced a customer from queue 1 and it decides to service a customer from queue 2 next, then a set-up time of $r$ time units is incurred in addition to the service time of the customers; no set-up time is required, if the server decides to continue servicing customers from queue 1. After each service completion, the server must make a decision: continue serving customers from the same queue or switch to the next queue.

Polling systems theory analyzes the performance of various polling policies [18]. Some popular examples are the following. In the exhaustive policy, the server continues to service customers from the same queue until that queue is empty. In the gated policy, right after the server switches over to a new queue, it takes a snapshot of this queue; and the server services only those customers in the snapshot. In the $k$-limited policy, the server services the customers in the same queue until either $k$ customers are serviced or all customers in the queue are serviced, whichever comes first. Once these customers are serviced, the server switches to the next queue. These policies can be formalized easily. We refer the reader, for example, to [18].

Polling systems literature has been active for several years [19]. The existing literature, including the analysis of these policies for a large class of similar polling models [17], focuses on the case when the interarrival times of the customers have independent identical memoryless distribution, i.e., the arrival times process $\{t_{i,k} : i \in \mathbb{N}\}$ is a Poisson process for all queues $k \in \{1,2\}$. For example, the necessary and sufficient conditions for stability, the expected delay, and the steady state queue length are known for all of the three policies considered above [18]. In fact, these values can be computed when the number of queues is more than two and the intensity of arrival times is different across queues [18].

Unfortunately, it is analytically challenging to find optimal policies [18]. However, approximation results for limiting cases are available. For example, on the one hand, the exhaustive policy is known to induce lower delay when compared to the gated polling system in a light-load regime, i.e., when the intensities of the arrival times are close to zero for both queues [18]; on the other hand, the gated policy is known to be better than the exhaustive one in the heavy-load regime, i.e., when the load is close to instability [17], [18].

B. Simulating Polling Systems Behavior

The coordination algorithms we present below heavily rely on the polling systems policies. In particular, in a number of places we simulate the behavior of a polling policy forward in time. We devote this section to formalizing this procedure.

Consider a polling system with two queues and deterministic service and set-up times. That is, the customers arrive at times $\{t_{i,k} : i \in \mathbb{N}\}$, where $k \in \{1,2\}$; their service requires $s$ time units, and the server requires $r$ time units to switch queues. The coordination algorithm we present in Section III-C relies on simulating the behavior of a polling system that has fixed service time $s$ and fixed set-up time $r$.

For notational convenience, we represent a polling system with the symbol $\mathcal{P}$. The algorithms we describe below interact with a polling system through two procedures. The procedure $\mathcal{P}.AddToQueue(k)$ adds one customer to queue $k$. The $\mathcal{P}.Simulate()$ procedure simulates the behavior of the polling system, assuming no additional customers arrive. More precisely, the $\mathcal{P}.Simulate()$ returns two sequences of time instances, namely $T_1 = (t_{i_1,1}, t_{i_2,1}, \ldots, t_{i_n,1})$ and $T_2 = (t_{j_1,2}, t_{j_2,2}, \ldots, t_{j_m,2})$, where $t_{i,k}$ is the time that the server is scheduled to begin servicing the $i$th customer...
in queue \( k \). Note that the behavior of the polling system \( \mathcal{P} \) depends on its polling policy (e.g., exhaustive, gated, \( k \)-limited, et cetera), the service time \( s \), the set-up time \( r \), the number of customers in the two queues, and the time the server began serving the current customer (if the server is currently serving any customers) or the time that the set-up operation started (if the server is currently switching over to the next queue). Given the values of all these variables, the \( \mathcal{P} \).Simulate() procedure is well-defined in the sense that the sets \( T_1 \) and \( T_2 \) are uniquely determined, if the service time \( s \) and the set-up time \( r \) are fixed (not random).

C. The Intersection Coordination Algorithm

We propose an event-triggered coordination algorithm that plans the motions of all vehicles that enter the control region described in Section II. More precisely, the control algorithm computes \( x_{i,k} \), for all vehicles that are in the control region, such that the trajectories \( x_{i,k} \) are dynamically feasible and no two vehicles collide. The coordination algorithm is event-triggered in the sense that the trajectories \( x_{i,k} \) are updated each time a new vehicle arrives at the control region.

The core procedure embedded in this event-triggered coordination algorithm is presented in Algorithm 1. Each time a new customer arrives in queue \( k \), this procedure is triggered. The procedure computes and returns the trajectories \( x_{i,1} \) for all \( i \in \{i_1, i_2, \ldots, i_{n_1}\} \) and \( x_{i,2} \) and all \( j \in \{i_1, i_2, \ldots, i_{n_2}\} \), each time a new vehicle arrives at the control region.

Before presenting the coordination algorithm, let us introduce the following motion planning procedure. The MotionSynthesize procedure generates a trajectory for each vehicle given the time this vehicle must reach the intersection region and the trajectory of the vehicle in front of it. Furthermore, the trajectory is designed such that the vehicle stays as closely as possible to the intersection region at all times. Suppose the time to reach the intersection is denoted by \( \tau \), and trajectory of the vehicle in front is denoted by \( x' : [t_0', \tau'] \to \mathbb{R} \). Then, MotionSynthesize\((x', \tau) := \arg \min_{x : [t_0, \tau] \to \mathbb{R}} \int_{t_0}^{\tau} |x(t)| dt \)

subject to

\[ \begin{align*}
\dot{x}(t) &= u(t), & \text{for all } t \in [t_0, \tau]; \\
0 &\leq x(t) \leq v_m, & \text{for all } t \in [t_0, \tau]; \\
|u(t)| &\leq a_m, & \text{for all } t \in [t_0, \tau]; \\
|x(t) - x'(t')| &\geq L, & \text{for all } t \in [t_0, \tau']; \\
x(t_0) &= -L; & \dot{x}(t_0) = v_m; \\
x(\tau) &= 0; & \dot{x}(\tau) = v_m,
\end{align*} \]

where \( v_m \) and \( a_m \) are the maximum velocity and the maximum acceleration of the vehicles, \( L \) is the length of the vehicle, and \( L \) is the length of the road in the control region.

Then, the coordination algorithm works as follows. The algorithm emulates the behavior of a polling system denoted by \( \mathcal{P} \). We set this policy to have fixed (non-random) service time \( s = 1/v_m \) and set-up time \( r = w/v_m \). Note that the service time \( s \) is precisely the amount of time from when a vehicle’s front bumper enters the intersection region to when its rear bumper leaves the control region assuming the vehicle is traveling at \( v_m \). The set-up time on the other hand is the amount of time from when the vehicle’s rear bumper leaves the control region to when its rear bumper exits the intersection region. (See Figure 2.) This polling system may be governed by almost any polling policy that satisfies some mild technical assumptions. (See Assumption IV.1.) Each time a new customer arrives in the control region from lane \( k \), the procedure AddToQueue\((k \) (see Algorithm 1) is triggered. This procedure first adds into queue \( k \) of polling system \( \mathcal{P} \) one customer that represents the newly arriving customer (Line 1). Then, the procedure simulates the polling system forward in time assuming no new customers will arrive (Line 2). The result of the simulation is a sequence of times, namely \( T_1 \) and \( T_2 \). Let us denote the element \( \tau_{i,k} \) of \( T_k \) by \( \tau_{k}(i) \). Finally, the procedure generates a trajectory using the MotionSynthesize procedure for each vehicle such that the \( i \)th vehicle in lane \( k \) is scheduled to reach the intersection at time \( \tau_{i,k} + L/v_m \) while avoiding collision with the vehicle in front (Line 6). To be precise, define \( x_{0,k}(t) = L \) for all \( t \in \mathbb{R}_{\geq 0} \) and \( k \in \{1, 2\} \); and the MotionSynthesize procedure is used to compute \( x_{i,k} \) for all \( i \neq 0 \).

IV. ANALYSIS

In this section, we show that the coordination algorithm presented in Section III has two important properties. Firstly, the algorithm is safe in the sense that no collisions occur in the control region. (See Definition II.1.) Second, the coordination algorithm provides good performance. Recall that the coordination algorithm simulates polling system. We show that the additional delay incurred in traversing the control region is no more than the delay that would incur in servicing a customer in the corresponding polling system, independently of the polling policy that governs the system.

These guarantees hold under certain assumptions. In what follows, we present a class of stochastic process models

---

```
1 \( \mathcal{P} \).AddToQueue\((k) \); 
2 \((T_1, T_2) \leftarrow \mathcal{P} \).Simulate(); 
3 for \( k = 1, 2 \) do 
4 \( \tau_{i,k} \leftarrow T_k(i); \) 
5 \( x_{i,k} \leftarrow \text{MotionSynthesize}(x_{i-1,k}, \tau_{i,k} + L/v_m); \) 
6 end
```

Algorithm 1: NewArrival\((k) \) procedure is triggered when a new vehicle arrives in the control region from lane \( k \).

---

![Control region and intersection region](image)

Fig. 2: We show the three time instances that are described in the text. In (a), the front bumper of the vehicle is entering the intersection region. In (b), the rear bumper is leaving the control region. In (c), the rear bumper is leaving the intersection region. In the figure above, the control region (shaded light grey) and the intersection region (shaded in dark grey) are shown.
for vehicle arrival times and two important assumptions. Subsequently, we state and prove our theoretical results.

a) Arrival time model: First, we model the arrival times \{t_{i,k} : i \in \mathbb{N}\} as a hard-core stochastic point process on the non-negative real line \([24]\) such that the \(t_{i+1,k} - t_{i,k} \geq l/v_i\) for all \(i \in \mathbb{N}\) and \(k \in \{1, 2\}\).\(^1\) The inequality guarantees that no two vehicles are in collision at the time of arrival. A widely studied hard-core stochastic process, which we also utilize here, is the Matérn process \([25]\)–\([27]\) generated by thinning a Poisson process.

In addition, we model the effect of over-crowding in the control region to rule out cases that trivially contradict safety. More precisely, we tacitly assume that, if vehicle \(i\) arrives in lane \(k\) at time \(t_{i,k}\) and at this time there is no input that saves it from hitting the car in front, then the same vehicle chooses not to enter the intersection.\(^2\) Let us emphasize that this assumption does not guarantee safety immediately. That there exists some path that does not lead to a collision does not immediately imply that there is a path that is both safe and provides performance guarantees. In other words, this assumption does not render the problem trivial. Without such an assumption, however, the system is (trivially) unsafe. That is, without it, one is faced with trivial cases where any coordination algorithm is unsafe with a non-zero probability. However, clearly, the chance that a vehicle must choose not to enter the intersection is very small, when the road length \(L\) is very large or the intensity of the arrival times is very small (i.e., the light load case). Hence, our analysis applies in these limiting cases, even without this assumption.

b) Assumptions: Recall that the behavior of the coordination algorithm depends on the policy of the polling system it simulates. We introduce a regularity assumption on this policy. Roughly speaking, we assume that the policy does not respond to the arrival of a customer in queue 1 by favoring the servicing of customers in queue 2, and vice versa.

**Assumption IV.1 (Regular polling policies)** Let \(\mathcal{P}\) denote the polling system that the coordination algorithm is using. The polling policy of \(\mathcal{P}\) satisfies the following. At any given time instance, suppose we simulate the polling system to get \((T_1, T_2) \leftarrow \mathcal{P}.Simulate()\). Suppose, right at that time instance we first add one customer to queue 1 and simulate the queue afterwards, i.e., \(\mathcal{P}.AddToQueue(1); (T'_1, T'_2) \leftarrow \mathcal{P}.Simulate()\) to obtain \((T'_1, T'_2)\). Then, policy does not schedule customers in queue 2 to an earlier time, i.e., \(T'_2(j) \geq T_2(j)\) for all \(j \in \{j_1, j_2, \ldots, j_{n_2}\}\), and moreover, the policy does not change the scheduled times of the customers in queue 1, i.e., \(T'_1(i) = T_1(i)\) for all \(i \in \{i_1, i_2, \ldots, i_{n_1-1}\}\). The same result also holds for queue 1, whenever a new customer is added to queue 2.

This assumption is satisfied by many polling policies.

**Proposition IV.2** The exhaustive, gated, and the \(k\)-limited polling policies (see Section III) all satisfy Assumption IV.1.

**Proof:** This result can be verified easily for each policy.

We verify only for the exhaustive policy. Without loss of generality, suppose a new customer arrives in queue 1. First, suppose the server was processing a customer from queue 1. This would cause the scheduled times for queue 2 customers to be shifted back in time, while it does not affect the scheduled times of customers in queue 1. Second, suppose the server was serving a customer from queue 2 at the time of arrival of the new customer. Then, the servicing times of the customers in queue 1 and 2 do not even change. Hence, Assumption IV.1 holds for the exhaustive policy. \(\blacksquare\)

We provide our main assumption that the length of the control region is bounded from below by a certain number.

**Assumption IV.3 (The length of the control region)** The length \(L\) of the roads in the control region is bounded from below as follows: \(L \geq 2v_{m1}/a_m + l + w\).

As we will show later, this assumption guarantees that the control region is long enough to allow the safe coordination of vehicles while guaranteeing good performance.

c) Main theoretical results: Our main theoretical results are presented below in Theorems IV.5 and IV.6. These results are enabled by a key lemma, which we present below.

**Lemma IV.4** Suppose Assumptions IV.1 and IV.3 hold. Then, each time a new customer arrives and Algorithm 1 is called, every call to the \(\text{MotionSynthesize}\) procedure (Line 6 of Algorithm 1) yields a feasible optimization problem.

Before providing the proof of this lemma, let us point out two important corollaries, which are our main results.

First, the algorithm guarantees safety surely. In other words, it is guaranteed that no collisions occur as new customers arrive at the control region.

**Theorem IV.5** (Safety) Suppose Assumptions IV.1 and IV.3 hold. Then, the control region is safe at all times \(t \geq 0\) in the sense of Definition II.1.

Second, the delay experienced by the vehicles is bounded by the delay of the corresponding polling system.

**Theorem IV.6** (Performance) Suppose Assumptions IV.1 and IV.3 hold. Recall that the additional delay incurred in transitioning through the control region for vehicle \(i\) in lane \(k\) is denoted by \(D_{i,k}\). (See Section II.) Let \(W_{i,k}\) denote the wait time of the corresponding customer added to the polling system (see Algorithm 1, Line 1) when vehicle \(i\) arrives in the control region from lane \(k\). Then, we have the following: \(D_{i,k} \leq W_{i,k}\), almost surely.

In essence, Theorem IV.6 states that the differential constraints that bind the customers as they traverse the control region are irrelevant. The delay is no more than the delay that is incurred in the corresponding polling system. Hence, one can employ any polling policy in the coordination algorithm and guarantee the same performance provided by this policy.
We emphasize that the expected delay is known for many polling policies, in particular the exhaustive and gated policies, when the arrival process is Poisson [18]. Then assuming that the vehicle arrival times is a stochastic Matérn point process, the expected delay time can be bounded as follows.

Let $E[D]$ denote the average delay for a typical vehicle while traversing the control region, when the arrival times of the vehicles is a Matérn process with parameter $\lambda$, for some polling policy. Let $E[W]$ denote the average delay for a typical customer in a polling system with Poisson arrivals with intensity $\lambda$, for the same polling policy. Then, $E[D] \leq E[W]$.

The proof of this simple corollary of Theorem IV.6 follows directly from the fact that the Matérn process with parameter $\lambda$ is obtained by thinning the corresponding Poisson.

Finally, we provide the proof of the main lemma. Due to space limitations, we provide a sketch, omitting a few details.

Proof: [Lemma IV.4] (Sketch) The following definitions are used in the proof. An incoming vehicle is admissible at time $t$, if the control region is safe at time $t$ (in the sense of Definition II.1) with the new vehicle added to the control region. We say that vehicle $(j, k)$ with schedule time $\tau_{j,k}$ is committed at time $t$ if for any $t' \geq t$ and $k \neq k$, an admissible vehicle at time $t$ added to polling system $P$ does not change the schedule time of vehicle $(j, k)$, i.e., $\tau_{j,k} = \tau_{j,k}$, where $\tau_{j,k}$ is the new schedule time of vehicle $(j, k)$. If $\tau_{j,k} \neq \tau_{j,k}$, we say that vehicle $(j, k)$ is uncommitted at time $t$.

Claim 1. Suppose vehicle $(j, k)$ is committed at time $t$ and has trajectory $x_{j,k}$ at time $t$. Then, any call to Algorithm 1 after time $t$ will assign trajectory $x_{j,k}$ to vehicle $(j, k)$.

Claim 2. Suppose vehicle $(j, k)$ enters control region at time $t_{j,k}$. Define $\xi$ to be the distance that vehicle $(j - 1, k)$ has traveled since entering the control region until time $t_{j,k}$, i.e., $\xi = L + x_{j-1,k}(t_{j,k})$. Let $v$ denote the velocity of vehicle $(j - 1, k)$ at time $t_{j,k}$, i.e., $v = \dot{x}_{j-1,k}(t_{j,k})$. Then the following is satisfied: $\xi + v^2/(2a_m) \geq v_m^2/(2a_m) + L$.

Claim 3. Let $\tilde{\tau}_{j,k}$ be the schedule time of vehicle $(j, k)$ at time $t_0$. Suppose at a later time $t_1 \geq t_0$, Algorithm 1 is called and updates the schedule time of vehicle $(j, k)$ to a new schedule time $\tilde{\tau}_{j,k}$, i.e., $\tilde{\tau}_{j,k} \neq \tau_{j,k}$. Then, there exists a dynamically feasible trajectory $\tilde{x}_{j,k}$ such that $\tilde{x}_{j,k}(\tilde{\tau}_{j,k} + L/v_m) = 0$.

Define $d_{j,k}(t)$ as the delay of vehicle $(j, k)$ at time $t$, which can be computed as $d_{j,k}(t) = \tau_{j,k}(t) - t_{j,k}$, where $\tau_{j,k}(t)$ is the schedule time of vehicle $(j, k)$ at time $t$.

Claim 4. If vehicle $(j, k)$ is uncommitted at time $t$, then we have that $d_{j,k}(t) \geq \delta t_{j,k} + t - t_{j,k}$ for any time $t$, where $n_j(t)$ is the number of uncommitted vehicles in the same lane ahead of vehicle $j$ at time $t$.

Claim 5. Let $x_{j,k}$ be the trajectory of vehicle $(j, k)$ as last computed by Algorithm 1 before time $t$. There exists $t_0$ such that $\dot{x}_{j,k}(t_0) = 0$ if and only if $d_{j,k}(t) \geq v_m/a_m$.

Now, we begin the proof of the lemma. Recall that Algorithm 1 computes the trajectory of every vehicle in the control region each time a new vehicle enters the control region. Suppose vehicle $(j, k)$ enters the control region. We show that whenever Algorithm 1 updates the trajectory of that vehicle, it does so safely. Since the following discussion is only concerned with one lane at a time, we drop the subscript denoting which lane we are referring to.

If vehicle $j$ is the first vehicle to arrive in its lane, Algorithm 1 always gives a feasible solution. Now suppose vehicle $j$ is not the first vehicle to arrive in its lane. Denote $x_j$ and $x_{j-1}$ the trajectories of vehicle $j$ and vehicle $j - 1$, respectively, as computed by Algorithm 1 at time $t_j$. The following three cases can occur. (Note that if vehicle $j - 1$ is uncommitted then by definition of committedness, vehicle $j$ must also be uncommitted).

Case 1. Suppose both vehicle $j$ and vehicle $j - 1$ are committed. The result immediately follows from Claim 1 and our over-crowding assumption, i.e., $x_j$ and $x_{j-1}$ are safe.

Case 2. Suppose vehicle $j$ is uncommitted and vehicle $j - 1$ is committed. By Claim 1, vehicle $j - 1$ will never change its trajectory. If the schedule time of vehicle $j$ does not change in any future call to Algorithm 1, then its trajectory also does not change, and, thus, $x_j$ and $x_{j-1}$ are safe by our over-crowding assumption. Suppose the schedule time $\tau_j$ of vehicle $j$ changes to $\tau_j$ in a future call to Algorithm 1. Using Assumption IV.1, we have that $\tau_j > \tau_j$. By Claim 3, a dynamically feasible trajectory $\tilde{x}_j$ that satisfies the new schedule time exists. Since trajectory $x_{j-1}$ has not changed, one can create trajectory $\tilde{x}_j$ that is also safe with $x_{j-1}$.

Case 3. Suppose both vehicle $j$ and vehicle $j - 1$ are uncommitted. Define $t_{dec}$ such that $t_{dec} - t_j = (L/v_m^2/a_m - l_1/(t_{dec} - v_m))$, where $l_n(t_{dec})$ is the number of uncommitted vehicles in the same lane ahead of vehicle $j$ at time $t_{dec}$. Note that $t_{dec} - t_j \geq 0$ follows directly from Claim 2. If vehicle $j$ is committed at $t_{dec}$, then its current trajectory $x_j$ will not be changed for any subsequent calls to Algorithm 1 by Claim 1, and the last computed trajectory assigned to vehicle $j$ is safe by construction. Suppose vehicle $j$ is uncommitted at $t_{dec}$. By Claim 4 and construction of $t_{dec}$, we have that $d_j(t_{dec}) \geq \delta t_{j,k}(t_{dec}) + t_{dec} - t_j = (L - v_m^2/a_m)/v_m$. Since $L \geq 2v_m^2/a_m + l + w$ (by Assumption IV.3), we have that $d_j(t_{dec}) \geq v_m/a_m$, and therefore, by Claim 5, vehicle $j$ must come to a full stop. By construction of $t_{dec}$, vehicle $j$ can safely come to a full stop behind vehicle $j - 1$. Thus, a call to Algorithm 1 at time $t_{dec}$ yields a feasible trajectory for vehicle $j$, namely, the trajectory in which vehicle $j$ decelerates from $t_{dec}$ as quickly as possible to a full stop. All subsequent calls to Algorithm 1 at time $t \geq t_{dec}$ will yield safe trajectories for vehicle $j$ when vehicle $j$ is uncommitted at time $t$, since the new trajectory of vehicle $j$ will just extend the length of time that vehicle $j$ is stopped. If vehicle $j$ becomes committed at some later time $t \geq t_{dec}$, then by Claim 1 the trajectory that Algorithm 1 previously computed for vehicle $j$ will be assigned to vehicle $j$ for all future calls to Algorithm 1. Since this trajectory was safe by construction, we have that vehicle $j$ will be safe for all future calls.

V. SIMULATIONS

In this section, we evaluate the proposed coordination algorithm in computational simulations. We used Matlab to generate the simulation results presented in this section.
Incoming vehicle arrival times were generated by a Matérn process [27]. The vehicle length, width, maximum velocity, and maximum acceleration were taken to be 2 meters, 1 meter, 10 meters/sec, and 4 meters/sec², respectively.

Define the Poisson process with parameter λ as the Poisson point process in the line that has intensity λ. Define the Matérn process with parameter λ as the Matérn process obtained by thinning a Poisson process with parameter λ.

In Figure 3, we show the trajectories of the vehicles with increasing load. It can be seen that the platooning behavior emerges when the load becomes substantial. In Figure 4, we compare the performance of the proposed algorithm with the performance of the corresponding polling systems, which is in line with our result in Theorem IV.6.

VI. CONCLUSIONS

We proposed a coordination algorithm that provides provable guarantees on both safety and performance in all-autonomous traffic intersections. The proposed algorithm is based on polling systems policies. In that regard, this work established novel connections between the polling systems literature and the motion planning, in the context of coordination problems in all-autonomous intersections.

REFERENCES