A Real-Time Server Based Approach for Safe and Timely Intersection Crossings

Pratham R. Oza

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

May 02, 2019

Arlington, Virginia

Keywords: real-time systems, intelligent transportation, sporadic servers, intersection management, ROS

A Real-Time Server Based Approach for Safe and Timely Intersection Crossings

Pratham R. Oza

(ABSTRACT)

Safe and efficient traffic control remains a challenging task with the continued increase in the number of vehicles, especially in urban areas. This manuscript focuses on traffic control at intersections, since urban roads with closely spaced intersections are often prone to queue spillbacks, which disrupt traffic flows across the entire network and increase congestion. While various intelligent traffic control solutions exist for autonomous systems, they are not applicable to or ineffective against human-operated vehicles or mixed traffic. On the other hand, existing approaches to manage intersections with human-operated vehicles, cannot adequately adjust to dynamic traffic conditions. This manuscript presents a technology-agnostic adaptive real-time server based approach to dynamically determine signal timings at an intersection based on changing traffic conditions and queue lengths (i.e., wait times) to minimize, if not eliminate, spillbacks without unnecessarily increasing delays associated with intersection crossings. We also provide timeliness guarantee bounds by analyzing the travel time delays, hence making our approach more dependable and predictable. The proposed approach was validated in simulations and on a realistic hardware testbed with robots mimicking human driving behaviors. Compared to the pre-timed traffic control and an adaptive scheduling based traffic control, our algorithm is able to avoid spillbacks under highly dynamic traffic conditions and improve the average crossing delay in most cases by 10–50%.

A Real-Time Server Based Approach for Safe and Timely Intersection Crossings

Pratham R. Oza

(GENERAL AUDIENCE ABSTRACT)

Safe and efficient traffic control remains a challenging task with the continued increase in the number of vehicles, especially in urban areas. This manuscript focuses on traffic control at intersections, since urban roads with closely spaced intersections are often prone to congestion that blocks other intersection upstream, which disrupt traffic flows across the entire network. While various intelligent traffic control solutions exist for autonomous systems, they are not applicable to or ineffective against human-operated vehicles or mixed traffic. On the other hand, existing approaches to manage intersections with human-operated vehicles, cannot adequately adjust to dynamic traffic conditions. This work presents a technology-agnostic adaptive approach to dynamically determine signal timings at an intersection based on changing traffic conditions and queue lengths (i.e., wait times) to minimize, if not eliminate, spillbacks without unnecessarily increasing delays associated with intersection crossings. We also provide theoretical bounds to guarantee the performance of our approach in terms of the travel delays that may incur on the vehicles in the system, hence making our approach more dependable and predictable. The proposed approach was validated in simulations and on a realistic hardware testbed which uses robots to mimic human driving behaviour in an urban environment. Comparisons with widely deployed and state-of-the-art traffic control techniques show that our approach is able to minimize spillbacks as well as improve on the average crossing delay in most cases.

Dedication

To my parents for their endless love and support.

Acknowledgments

I would like to thank my advisor, Dr. Tam Chantem for providing me with the opportunity to pursue the masters program at Virginia Tech and her invaluable guidance throughout this time. Her continuous constructive advise and support has steered me to grow as a researcher and as an individual. I would also like to thank Dr. Ryan Gerdes for providing time to entertain my silly questions and lending his expertise in making the robots work for the RTX lab. I would also like to thank Dr. JoAnn Paul for serving on my committee and giving shape to this thesis.

Last few years would not have been this productive and enjoyable without Mahsa, Tanmaya, Gaurang and Anway, among others. The tea sessions, banters and some helpful discussions with them and their support during the testing times kept me going through some long nights. I am grateful to Mitali, Walia, Animesh, Indu, Parth, Abhinandan, Sanket and Nakul, among others, who've stood by me since my undergraduate days. Above all, I would like to thank Rohini for being an unceasing source of motivation, encouragement and happiness through all these years.

Finally, I would like to express gratitude to my family who have supported me through all my endeavours. I would specially like to thank my parents for their limitless love, belief and support, their hard work and sacrifice, without which none of this would have been possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As traffic continues to increase, congestion remains an everyday challenge for commuters in urban areas who face an average delay of 42 hours a year [48]. Congestion also negatively impacts the U.S. economy due to wasted time (\$305 billion in 2017 [7]) and fuel costs (19 gallons of fuel per commuter is wasted solely due to idling in traffic per year [48]), as well as the environment (28% of CO_2 emissions in the U.S. come from the transportation sector [14]). Development of additional roads is not feasible due to shortage in land resources.

While it is important to manage traffic in general, congestion at one intersection can (i) result in stop-and-go traffic and potentially collisions [18], and (ii) cause queue spillbacks in multiple lanes which can lead to a huge cog in the entire network, resulting in unexpected delays and long recovery time [45]. Thus, a traffic control system that manages the intersection efficiently can have a significant impact on the overall transportation network. Various adaptive traffic control systems [21, 22, 51] employ different optimization techniques to adjust the pre-timed signal policy to adapt to varying traffic patterns. However, such methods are not easily tuned online, and urban roads with closely spaced intersections do not necessarily follow the modeled traffic scenarios. Such intersections are often prone to queue spillbacks [40]. A spillback occurs when there is a standing queue downstream of an intersection that disrupts the discharge of vehicles even when the light is green. Such congestion can even obstruct the flow of the emergency vehicles through the traffic, thereby increasing their response time. Spillbacks can also be observed when the separation between densely packed intersections is small ($< 170\,\mathrm{m}$ [50]) and when the lights are locally managed without considering the flow of traffic at upstream intersections. Hence, controlling traffic flow in accordance to changing traffic conditions and queue lengths is key to avoid spillbacks, reduce congestion, and improve the overall trip time of the vehicles. Due to its adverse effects on the entire traffic network, considering queue spillback is important while adapting to dynamic traffic patterns, even if it results in slightly increased delays. While increased trip times may reduce the drivers' satisfaction, minimizing spillback has been shown to lead to fewer collisions and disruptions [30].

For autonomous vehicles, traffic control can be optimized by exploiting car-to-everything (C2X) connectivity and using virtual traffic lights [57]. These approaches have been shown to be effective at alleviating congestion and increasing fuel efficiency [59]. However, the vehicles occupying our roadways today are still predominantly conventional vehicles. Therefore, in the short term, an efficient and effective traffic control infrastructure that exploits real-time traffic data (e.g., time of day, temporary road blocks, and social or unexpected events) and various sensing capabilities (e.g., image processing using cameras, loop detectors and on-road sensors, etc. [39]), and which is applicable to both human-operated as well as autonomous vehicles is crucial.

The goal of this work is to design a traffic control system at a given intersection that is able to quickly adapt to changing traffic patterns without relying on complex traffic models, while accommodating both conventional vehicles and future intelligent transportation systems. Our objectives are to (i) minimize, if not altogether eliminate, spillbacks in order to improve travel times without inducing long delays associated with intersection crossings, and (ii) bound the delay associated with the intersection crossings so that travel times can be accurately determined, which can help in selecting fastest routes precisely. Our main contributions are:

1. We formulate the traffic control problem at an intersection as a real-time task scheduling problem and develop a server-based approach to adapt to changing traffic flows and queue lengths, leveraging the traffic information at this and neighboring intersections. Our approach ensures that queue spillbacks are minimized, if not eliminated, and that vehicles can cross intersections in a timely manner without collisions.

2. We provide a bound on the worst case wait time experienced by the vehicles at an intersection by exploiting the real-time properties of our proposed model.

3. We analyze different traffic arrival patterns and gauge the adaptability of our algorithm in simulations. Data show that our approach avoids spillbacks even under extremely unbalanced traffic flows from different directions and changing flow rates while reducing the average crossing delay by 10% to 50% in most cases.

4. We validate our approach and assess its performance on a hardware testbed with robots representing vehicles and emulating realistic human driving response in a typical urban traffic environment.

Chapter 2

Literature Review

Recent work on intersection management has primarily focused on managing traffic flow for autonomous vehicles [4] or by assuming the vehicles are connected [55]. Reservation-based schemes were introduced where autonomous vehicles benefit from their communication features and make reservations in the intersection, while the human-driven vehicles follow the standard traffic light [11]. A message request based connected vehicle traffic management is also presented [58], where the effect of communication delays in traffic control is studied. Since it will be 25–30 years before all vehicles will have connected vehicle technology [13], approaches that do not favor only the connected vehicles are needed. Our approach works for both mixed and fully-automated traffic.

Gathering traffic information to optimize and improve signalization at an intersection has also been a key research area. Traffic pattern analysis can be conducted using wireless sensor networks [6] and queue estimation analysis [52], and then used to optimize the signal timings. Existing solutions in this area are either fixed timing-based traffic systems, detector-based reactive traffic control systems, or adaptive traffic controllers [32]. Fixed timing controllers use offline databases to adjust signal timings according to the hourly usage pattern generally observed. Sensor based traffic detection, i.e., loop detectors and cameras, are often used for real-time traffic control. However, such actuation based controls require sensor activation and are not very adaptive in heavy traffic leading to resource starvation [19]. In addition, the sensors require calibrations and may malfunction [13, 41].

Adaptive traffic controllers used for conventional traffic [22, 51] rely on mathematical models and optimization techniques. These control techniques provide centralized control over the entire urban network, which can achieve optimal traffic control but is not scalable in practice [41]. Field evaluations for the centralized control techniques have shown that slight inaccuracies in control validation, which may occur due to inaccurate traffic data estimation, can result in significant performance drop in traffic management [26].

Similarly, prediction and learning based techniques using reinforcement learning and deep learning [33] can be used to optimize traffic at an intersection, but require high computational capabilities, along with large amount of data containing different traffic patterns and dynamics. Such computational and data intensive algorithms require costly infrastructure, which is not readily available. In a closely related work, an adaptive traffic control technique that maximizes the green times for real-time traffic signalization was presented [20]. While the authors proposed a local intersection control approach by introducing a weight factor to the phases as per the incoming traffic flow, they did not consider queue spillbacks, which can cause disruptive effects, especially in urban networks [46], as will be shown in Chapters 5.2 and 5. We focus on minimizing spillbacks since they have been shown to have propagating effects on the network [37, 56]. To the best of our knowledge, existing works in traffic control [4, 11, 20, 33] are best-effort approaches and do not provide worst-case delay guarantees.

Chapter 3

System Model, Assumptions and Problem Statement

3.1 Road, Infrastructure, and Vehicle Models

We consider an intersection with incoming traffic from four different directions and which is a typical crossroad where the incoming traffic enters the intersection to either go straight or take a left turn[1]. Hence, there are eight different traffic flow patterns inside the intersection, as shown in Figures 3.1 and 4.1. It is important to notice that every entering traffic flow has a non-conflicting traffic flow which neither disrupts nor hampers the former's ongoing flow. For instance, for vehicles going straight from lane L_1, vehicles from lane L_5 or vehicles from lane L_2 are non-conflicting and do not interfere with lane L_1's flow. Thus, when signalizing the intersection, and lane L_1 is allowed access to the intersection, either lane L_2 or lane L_5 can simultaneously access the intersection. Considering this, Figure 3.1 shows that there can be two different usage patterns known as phase sequences of the intersection that indicate which two non-conflicting lanes can access the intersection and that the eight incoming lanes can be combined into four different phases that need exclusive access to the intersection.

[1]Right turning vehicles are not considered for simplicity, as they do not necessarily enter into the intersection and usually have a special channelized turning lane dedicated for a free right turn. However, right turning vehicles can be considered in our approach by considering them as independent incoming lanes into the system.

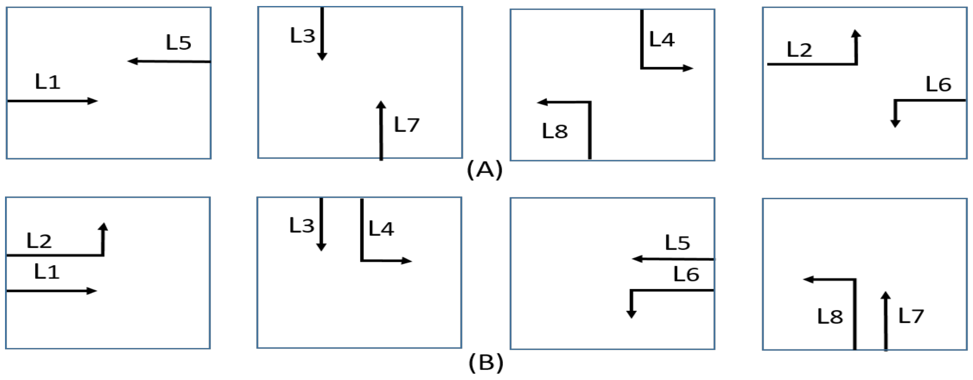


Figure 3.1: Two possible phase sequences (A) and (B).

Each phase consists of two lanes.

Currently deployed adaptive traffic control techniques make use of the traffic data collected from various sources and sensors that are placed through out the road network to measure and estimate traffic flow. Depending on the sensors and the data collection methodology employed, different parameters regarding the flowing traffic are measured. Most widely deployed data recorders and sensors are:

- Pneumatic road tubes, piezoelectric sensors, magnetic loops:
  - Intrusive methods that require vehicles to come in physical contact with the sensors to perform measurements.
  - Use pressure difference, electrical energy and magnetic field generation respectively to provide vehicle count, speeds and weight information to the traffic controller.
  - A sensor type may be beneficial or detrimental depending on the weather, temperature and traffic conditions, which may show variations in lane coverage and measurement accuracy.
  - Combination of these sensors are placed at various locations on a lane to overcome

drawbacks of one another and still provide accurate traffic information.

- Infrared sensors, microwave radar sensors, acoustic sensors and video image detection:
  - Non-intrusive methods which remotely observe vehicle presence and its characteristics.
  - Detect presence, speed and vehicle type using infrared radiations, Doppler effect, acoustic signals and captured image frames respectively.
  - Multiple such sensors can be combined to increase lane coverage and measure various vehicle characteristics.
  - Such approaches are specifically sensitive to meteorological conditions.

All the above mentioned sensors are then connected over a wired or a wireless network to transmit the collected data to the traffic controller located at all intersections to process the data and run the traffic control algorithms [31]. Small interconnected groups of the controllers are then used to coordinate traffic through the network. Urban traffic control system (UTCS) software with various versions is used by these group of traffic controllers to perform computations More advanced techniques such as using Floating Car Data (FCD) locates vehicles via cellular network or GPS network of a mobile phone located inside the vehicle. Traffic controllers currently available are capable of using such high quality data to complement the data processed from the road-side sensors. Detailed information on sensors, controllers and traffic infrastructure can be found in Traffic Detector Handbook, provided by Federal Highway Federation [15]. We assume that our system is managed by an intersection manager (IM), which aggregates information about the incoming traffic patterns through sensors and/or upstream intersections and forecast data, if any. The intersection manager then calculates the required green-yellow-red light timing. In our approach, we only require (some) traffic information to be known, which can be collected using the approaches discussed earlier. This, however, implies that the intersection managers are deployed along with the existing traffic controllers while ensuring that the controllers have minimal cloud connectivity to share the data. These controllers can also be enabled with communication technologies that are dedicated to vehicle-to-infrastructure (V2I) such as CV2X or DSRC, for efficient data sharing.

3.2 Traffic Model

All the traffic indicators are assigned the green-yellow-red light timings. One cycle is said to have completed, when all the traffic indicators of the intersection have completed one rotation of lights. The time taken to complete an entire cycle is called the cycle time (T_c), after which the signals repeat the pattern. Typically, the cycle time is divided into smaller chunks of signal times which are distributed among the vehicular as well as pedestrian traffic that need to use the intersection. Each chunk of signal times would allow the incoming traffic from a particular phase sequence to cross the intersection. (In our model, though we focus on the incoming vehicular traffic, pedestrian traffic can trivially be incorporated.)

We use lane capacity as a measure to detect a spillback. The lane capacity is defined by the maximum number of vehicles that a lane can handle on average. Since urban traffic usually consists of passenger vehicles, if the length l_i of a lane L_i, the length v_i of an average passenger vehicle, and the average safe spacing distance s_i between two consecutive vehicles are known, then the lane capacity z_i can be estimated as

$$z_i = \frac{l_i}{v_i + s_i}. \tag{3.1}$$

Figure 3.2 describes the calculation of lane capacity. The capacity of a lane is a constant and can be found prior to deployment by surveying vehicle types accessing the urban areas or from real-time classification approaches [54].

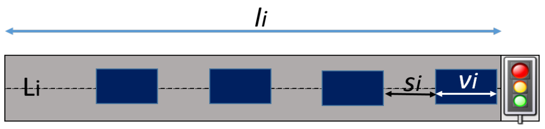


Figure 3.2: Lane capacity calculation

An incoming lane L_i is characterized by a tuple $\{a_{i,j}, q_{i,j}, z_i\}$, where $a_{i,j}$ is the flow rate of incoming vehicles during the j^{th} cycle, $q_{i,j}$ is the number of vehicles already queued in the lane at the beginning of the j^{th} cycle, and z_i is the lane capacity, which does not change over time. For a given flow rate, the amount of time a lane needs access to the intersection to avoid spillbacks can be calculated if we know the number of vehicles that must be dispatched during T_c before the lane reaches its capacity. For example, for a closely spaced intersection with the incoming lane length of 120 m, an average vehicle length of 5 m, and a spacing of 1 m, the capacity of this lane is 20 veh. With an incoming traffic flow rate of 5 veh/min, queue in this lane will spillback in $20/5 = 4$ min if no vehicle from this lane crosses the intersection. In other words, to avoid a spillback, the lane under consideration must receive a green light within 4 min. This spillback time $t_{sb_{i,j}}$ is defined as follows.

Property 1 (Spillback Condition). For a lane L_i during the j^{th} traffic cycle, assuming that $a_{i,j}(t)$ is the flow rate of vehicles that varies with time t, $q_{i,j}$ is the existing queue length, i.e., number of vehicles already in L_i, at the start of the j^{th} cycle, and z_i is the capacity calculated using (3.1), then, the spillback time $t_{sb_{i,j}}$ is

$$t_{sb_{i,j}} = \frac{z_i - q_{i,j}}{a_{i,j}(t)}. \tag{3.2}$$

Proof. Based on the definitions, $a_{i,j}(t) \cdot t_{sb_{i,j}}$ vehicles will enter L_i during the time interval of length $t_{sb_{i,j}}$. Hence, the total number of vehicles in L_i during $t_{sb_{i,j}}$ is

$$n_{i,j} = a_{i,j}(t) \cdot t_{sb_{i,j}} + q_{i,j}.$$

If none of the vehicles is dispatched during $t_{sb_{i,j}}$, then a spillback will occur after $n_{i,j}$ equals the capacity, i.e., $a_{i,j}(t) \cdot t_{sb_{i,j}} + q_{i,j} = z_i$ and the property holds. □

We assume that the flow rate $a_{i,j}(t)$ varies with time, but is fixed within a given cycle T_c. Hence, we refer to $a_{i,j}(t)$ as $a_{i,j}$ for the rest of the manuscript. For rapidly changing flow rates, the worst case flow rate within T_c can be bounded, given the starting flow rate at the beginning of T_c, and can be used in the analysis.

To determine the number of vehicles that can be dispatched in a given green time interval, we make use of saturation headway [49]. When the traffic light for a lane turns from red to green, the leading vehicle takes a longer time (h_1) to react to the change in traffic lights. This headway difference slowly reduces to e_1, $e_2 \dots e_m$ as the queue moves forward where $e_1 \geq e_2 \geq \dots \geq e_m$ since following vehicles react comparatively faster. After m vehicles, the time headway stabilizes to a value h and $e_{m+1} \dots e_n = 0$ (usually after the 6^{th} vehicle [27]). Hence, in one cycle of green-yellow-red, there is start-up lost time denoted by t_s when none of the vehicles utilize the intersection due to the delayed reaction of the leading vehicles in the queue and $t_s = h1 + \sum_{k=1}^{m} e_k$. Along with t_s, there exists clearance lost time, t_{cl} that represents the time between the signal phases during which an intersection is not assigned to any of the lanes. During t_{cl}, the vehicles that have entered the intersection just before the light turned red are allowed to clear the intersection. The total lost time t_l comprises of both t_{cl} and t_s; $t_l = t_s + t_{cl}$. Accordingly,

$$T_{i,j} = h \times n_{i,j} + t_l. \tag{3.3}$$

In other words, $T_{i,j}$ shows the amount of time required to discharge $n_{i,j}$ vehicles from a lane during the j^{th} traffic cycle when the saturation headway h and the lost time t_l are considered. The constants h and t_l do not change with traffic condition and the Highway Capacity Manual defines t_l and h as 4 s and 2 s, respectively [38].

3.3 Problem Statement

Let us assume an intersection with incoming traffic from four different directions, with a total of eight lanes (directions of flow), where access to the intersection is dictated by a traffic light for each direction of flow and according to the phase sequences, each of which consists of two lanes. Further, each lane is characterized by a capacity, an instantaneous flow rate, and a queue length as described earlier. The goal is to determine the cycle time T_c and the signal timing assignment within T_c for the different phase sequences to minimize spillbacks.

Chapter 4

Intersection Management as a Real-Time Task Scheduling Problem

Real-time systems comprise of computing systems that need to make precise timing decisions, perform accurate computations and provide correct outcomes in a time constrained environment. A real-time system distinguishes itself from other systems by ensuring that the system not only provides the logical outcome post computation, but also provides the result within a defined time deadline. Depending on the repercussions of missing these deadlines and how time-critical the system is, a real-time system can be distinguished in three categories:

- Hard - when there are catastrophic consequences on the system, upon a missed deadline
- Firm - when the results calculated post the defined deadline are deemed useless for the system, but such deadline misses do not cause any damage to the system
- Soft - when a missed deadline may cause performance degradation, but the calculated result can still be used by the system for further computation and control.

While considering a system that consists of multiple concurrent tasks that are executed on a single processor or when a single critical resource is shared among multiple such tasks, real-time theories provide scheduling policies, which determines the order and times at which the tasks will access the resource or get executed on the processor. Real-time theories are thus exploited to provide robust scheduling algorithms to ensure all concurrent tasks get their fair share of the shared resource and none of the tasks are starved of processor time. Real-time task scheduling is often used to achieve timeliness guarantees or to increase responsiveness [2]. In an environment with multiple tasks sharing a common resource, real-time scheduling policies can provide (i) guaranteed level of service for each task (ii) isolation among the tasks such that the overall performance of all tasks in the system is maintained [35].

Efficiently managing traffic at an intersection also relies on precise and timely traffic signal parameter calculations. Using precise traffic information, timing decisions for the traffic lights need to be calculated such that none of the lanes experience heavy congestion and queue spillbacks. It is also expected that no two vehicles with conflicting paths cross the intersection at the same time. Such behavior may cause collisions, further worsening the traffic. This requires multiple vehicles to be scheduled efficiently through an intersection such that collisions are avoided, while the performance of the system is still maintained. Such requirements can be effectively satisfied by using real-time properties and scheduling algorithms. Detailed understanding on real-time theories and task scheduling policies can be found in [3, 9, 36].

The problem of traffic control at an intersection (Chapter 3.3) can thus be transformed into a real-time task scheduling problem while ensuring that there is no conflict in accessing the intersection and the traffic remains under the capacity without causing spillbacks. We will also leverage a real-time based analysis to provide bounds on the delays incurred in the traffic associated with intersection crossings. Such worst case delay analysis can be useful in providing predictable wait-times for the vehicles, make searching for faster routes more reliable and leads to a more accurate trip time estimation.

4.1 Overview

We model an intersection shown in Figure 4.1 as follows:

- Vehicles waiting to cross an intersection–aperiodic tasks that needs to be executed.
- Incoming lanes in each phase sequence–aperiodic task queues.
- Intersection–resource, e.g., processor, that is shared among all the incoming lanes with conflicting flows.
- Traffic lights for each direction of flow–sporadic server responsible for executing the aperiodic tasks, i.e., the incoming vehicles.
- Intersection manager (IM)–a sporadic task that gathers the incoming traffic data to calculate the cycle time T_c as well as the signal timing within T_c. The IM ensures that the serverrs are non preemptive.

As explained in Chapter 3, an intersection can be divided into four phases, each of which consists of two lanes with non-conflicting flows. These four phases need to be effectively scheduled to utilize the intersection while ensuring no collisions occur by disallowing vehicles from multiple phases to enter the intersection at the same time. We thus consider a lane in each phase as an aperiodic task queue. Vehicles entering into the lanes are represented as aperiodic soft real-time tasks having a known execution time (C_i), i.e., time to cross the intersection, and an arrival time (r_i), defined as $\tau_i(r_i, C_i)$. The execution time, C_i is a function of the traveling speed of the vehicles, which is relative to the traffic flow rate and vehicle's position in the queue [5]. As shown in Figure 4.1, vehicle $\tau_1(1, 6)$ arrives at time 1, and has an execution time of 6. The tasks are considered soft real-time as they do not necessarily have a deadline before which they need to be scheduled, i.e., enter and exit the

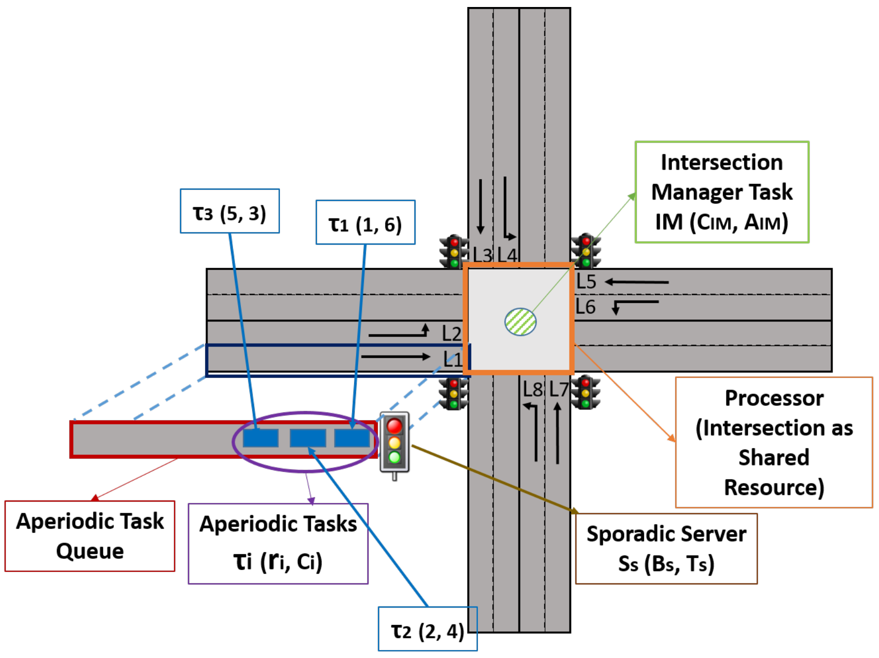

τ3 (5, 3)
τ1 (1, 6)
Intersection Manager Task IM (CIM, AIM)
L3 L4
L5
L6
L2
L1
L8 L7
Processor (Intersection as Shared Resource)
Sporadic Server Ss (Bs, Ts)
Aperiodic Task Queue
Aperiodic Tasks τi (ri, Ci)
τ2 (2, 4)

Figure 4.1: An intersection as a real-time task scheduling problem.

intersection. However, to reduce wait times and improve traffic flow, the vehicles should be allowed to cross the intersection as soon as possible. An intersection is thus, a shared resource. In all, we have eight task queues corresponding to eight lanes that combine to form four phases which hold the incoming aperiodic tasks. Each task queue is served using a sporadic server. Figure 4.2 shows the steps followed by our algorithm.

4.2 Sporadic Servers to Execute Aperiodic Tasks

As discussed earlier, vehicles in a given lane that need to access the intersection are modeled as aperiodic tasks that are queued in a task queue (associated with that lane) until they are ready to be executed on the processor, i.e., until the vehicles are permitted to enter and cross an intersection. To serve these aperiodic tasks, we leverage the concept of sporadic servers [12]. In our setting, a sporadic server is responsible for executing the queued aperiodic requests. Such a server executes until all aperiodic requests have been served or it has exhausted its execution budget during that period, whichever comes first. Hence, a sporadic server job S_S is represented by its budget B_S and an arrival time a_S. We do not distinguish between a sporadic server and a sporadic server instance when the context is clear. That said, a sporadic server is characterized by a minimum interarrival time T_S as well as a budget B_S. Since there are eight incoming lanes, and hence, eight aperiodic task queues, we will have eight sporadic servers, each of which is paired with another (corresponding to its non-conflicting lane in a given phase) and is served along with its pair. We will present an approach to assign the budget and the arrival time of these servers in Chapter 4.4.

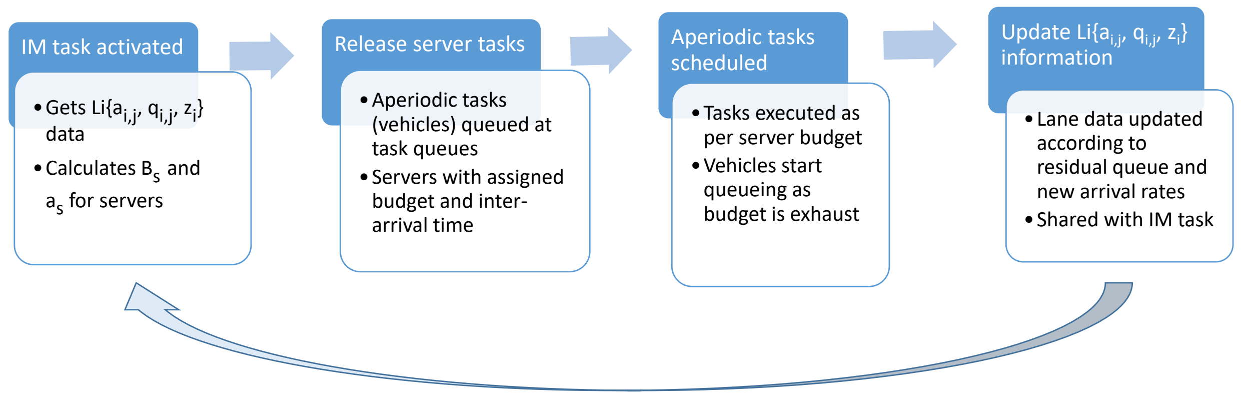

IM task activated
Gets Li{$a_{i,j}$, $q_{i,j}$, z_i} data
Calculates B_s and a_s for servers
Release server tasks
Aperiodic tasks (vehicles) queued at task queues
Servers with assigned budget and inter-arrival time
Aperiodic tasks scheduled
Tasks executed as per server budget
Vehicles start queueing as budget is exhaust
Update Li{$a_{i,j}$, $q_{i,j}$, z_i} information
Lane data updated according to residual queue and new arrival rates
Shared with IM task

Figure 4.2: Flowchart depicting the steps of our algorithm.

Figure 4.3 shows the operation of a typical sporadic server serving aperiodic tasks. Aperiodic tasks (vehicles) arrive and join the aperiodic queue (lane). The sporadic server (green light) is invoked according to its arrival time and execute the aperiodic task(s) at the head of the queue. The duration for which the server serves aperiodic requests (green light) is equal to the server's budget (maximum green light time) unless the aperiodic queue is empty, in which case the server suspends itself (red light). A sporadic server will be activated again, and its budget replenished, no sooner than after the set minimum interarrival time has been reached, i.e., if an incoming lane is empty, the sporadic server is not activated again until

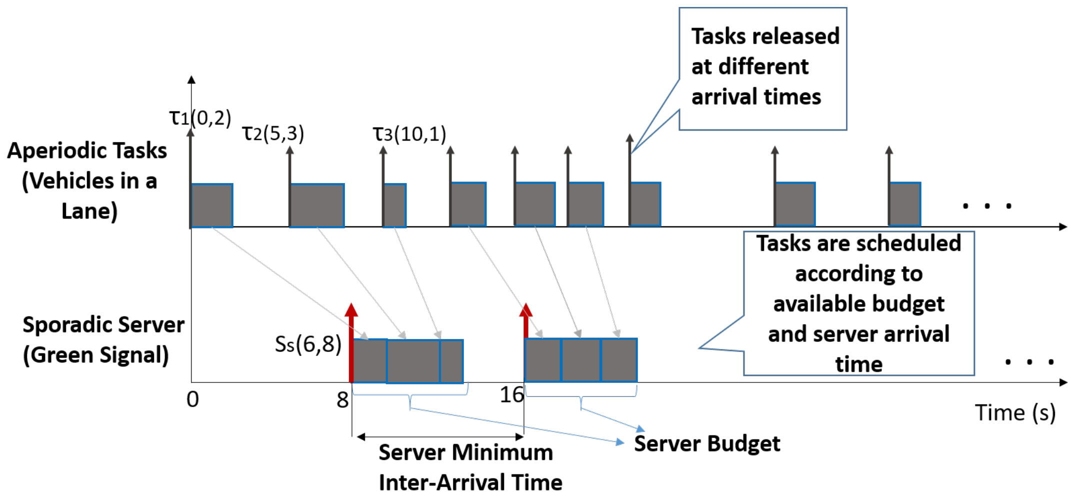

Tasks released at different arrival times
τ1(0,2)
τ2(5,3)
τ3(10,1)
Aperiodic Tasks (Vehicles in a Lane)
Tasks are scheduled according to available budget and server arrival time
Sporadic Server (Green Signal)
Ss(6,8)
0
8
16
Time (s)
Server Budget
Server Minimum Inter-Arrival Time

Figure 4.3: Example sporadic server serving aperiodic tasks.

a vehicle arrives. In Figure 4.3, aperiodic tasks τ_1, τ_2 and τ_3 are released at time 0, 5, and 10, with execution times of 2, 3, and 10, respectively. As these tasks arrive, they join the aperiodic task queue. When the server $S_S(6, 8)$, with a minimum interarrival time of 8 and budget of 6 is invoked at time 8, τ_1, τ_2, and τ_3, whose combined execution times are 6 (equal to the server's budget) are executed and the server suspends itself at time 14. The server is activated again at time 16 since its minimum interarrival time is 8.

4.3 Intersection Manager

Unlike a quintessential sporadic server that replenishes its capacity and sets its next activation time in accordance to its consumption, the budget replenishment rules for the sporadic servers used in our work are governed by a monitoring task, i.e. the intersection manager (IM), which is also responsible for guaranteeing that the collective server budget is no greater than the maximum budget, i.e., that the "system" is not overloaded. In addition, the IM assigns budgets in such a way that no preemptions can occur to ensure safety, since it is not possible to "preempt" a vehicle after it has entered the intersection. The IM is a lightweight sporadic task, with an execution time C_{IM} and an arrival time a_{IM}. The arrival time of the IM task coincides with the traffic cycle time (T_c), i.e. IM task is activated at the end of each traffic cycle. Once activated, the IM aggregates traffic information for the incoming lanes and calculates the budget and the arrival time of each sporadic server based on the spillback concept discussed in Chapter 3.2. It also sets its own next arrival time, depending on the cycle length obtained from the calculated budget (Chapter 4.4). Note that the budgets and the arrival times are expected to change over time due to time of day, traffic pattern etc., and calculated in such a way that all sporadic servers and the IM task itself are schedulable.

The minimum interarrival times for all sporadic tasks in the system, i.e., the sporadic servers and the IM task are set to $t_{min} = h1 + t_{cl}$. This is the minimum time required for the lead vehicle in a queue to cross an intersection as it will take $h1$ time due to the reaction delay to the green light and take t_{cl} time to clear the intersection (as explained in Chapter 3.)

4.4 Parameter Assignment for Sporadic Servers

Our goal is to assign the cycle time, as well as the budget and the arrival time of a given sporadic server instance within each cycle such that spillbacks are minimized. To calculate the cycle time T_c, we use a concept that is similar to critical lane analysis [47]. That is, we determine the smallest spillback time among all lanes according to (3.2). The idea is that, in this way, none of the lanes experiences a spillback before they receive their green time in a given cycle. This least spillback time is used as the cycle length T_c for the next traffic cycle. Since the cycle length is distributed among all incoming lanes, some vehicles will be dispatched from every lane and hence none of the queues will spillback within this traffic cycle. Consider an example in Table 4.1 where lanes $(L_1, L_5), (L_2, L_6), (L_3, L_7)$ and (L_4, L_8)

form the four phases with non-conflicting flows. Table 4.1 shows the spillback time for each queue. The capacity, existing queue length, and flow rate of each lane are also listed. In this case, the critical lane is L_3 and the cycle time is set to the smallest spillback time, i.e., $T_c = 90$.

Table 4.1: Green time calculations using server based approach with cycle time of 90 sec

Phase	Lane	Cap (veh)	Queue (veh)	Arr Rate (veh/min)	Spillback time (s)	Budget (%)	Green Time (s)
1	1	10	4	3	120	24.5	0-20
	5	13	2	2	300		
2	2	13	3	1	600	23.39	21-40
	6	12	3	4	135		
3	3	14	5	6	90	26.71	41-66
	7	15	3	4	135		
4	4	15	5	2	300	25.38	67-89
	8	13	4	5	108		

To calculate the arrival time of the eight sporadic servers, we make the following observation. Since the time when these servers execute coincides with a green time for a given lane, we set the arrival time to be the same as the cycle time T_c and enforces isolation, i.e., only lanes in a given phase receive green light at a time, by assigning a fixed priority to each server in an arbitrary but consistent manner. In our work, we assume that $\pi_{S_{S_i}} \geq \pi_{S_{S_j}}$, $i < j$, where $\pi_{S_{S_i}}$ denotes the priority of server S_{S_i}. For servers whose lanes belong in the same phase, their priorities are the same and their corresponding servers can execute at the same time with the same budget.

From Table 4.1, the cycle time of 90 will be distributed among the four phases. Thus, all eight lanes will dispatch some number of vehicles during this time period, thereby avoiding spillbacks not only in the third lane but in all other lanes as well. After the cycle time has elapsed, the IM task will be activated again to calculate the new value for T_c and the budget for the servers. We now discuss how to assign the budgets.

4.4.1 Calculating Minimum and Maximum Budgets

Let us define the utilization of a sporadic server S_{S_i} as $U_i = \frac{B_i}{T_c}$. Depending on the flow rate, queue, capacity, and arrival time, every server S_{S_i} (phase) will have a minimum utilization demand, $U_{i,j_{min}}$, to avoid spillback within j^{th} cycle, which is given by

$$U_{i,j_{min}} = \frac{h \times (a_{i,j} \cdot T_c + q_{i,j} - z_i + 1) + t_l}{T_c}. \tag{4.1}$$

As explained later in Lemma 1, $h \times (a_{i,j} \cdot T_c + q_{i,j} - z_i + 1) + t_l$ indicates the minimum number of vehicles that need to be dispatched from lane L_i. Similarly, the maximum utilization demand $U_{i,j_{max}}$ of each server (phase) can also be calculated. The maximum utilization demand is defined as the amount of utilization that a lane requires to dispatch every vehicle currently in the queue as well as every vehicle expected to arrive during T_c. Therefore,

$$U_{i,j_{max}} = \frac{h \times (a_{i,j} \cdot T_c + q_{i,j}) + t_l}{T_c}. \tag{4.2}$$

To avoid spillbacks in any phases, the assigned utilization should be between $U_{i,j_{min}}$ and $U_{i,j_{max}}$. The closer it is to $U_{i,j_{max}}$ for each phase, the more we are dispatching than the minimum requirement to avoid spillback and hence the system performs better with shorter queue buildup for the next cycle. To start, each server is initialized with its respective minimum utilization, which is recalculated with updated traffic information at the beginning of each cycle. We next discuss how to distribute the leftover budget (green time) if the combined minimum utilization for the four phases is less than the length of T_c. Note that if such combined minimum utilization is greater than T_c, spillbacks cannot be avoided.

4.4.2 Distributing Leftover Budgets

Once the minimum budget demands of all the incoming lanes have been satisfied, we aim to maximize the assigned budget for each lane. We adopt a simple heuristic where the leftover budget is divided among the phases inversely proportional to the spillback time; less budget is allocated to the lane with a higher spillback time.

4.5 Correctness and Worst Case Delay Analysis

Once T_c is calculated (Chapter 4.4), the server budgets are distributed such that the total assigned budget does not exceed 100% of the available bandwidth (T_c). Hence, the total green times assigned to the servers will never exceed T_c, thereby preventing the occurrence of conflicting green lights. In addition, since the servers execute in a fixed-priority, round robin fashion, and server budgets are calculated in such a way that preemption cannot occur, no two or more lanes with conflicting traffic flow (servers) will have green lights at the same time, as the lower priority servers will not get to run as long as a higher priority server is being executed and has not exhausted its budget. As the system is not overloaded (total bandwidth $\leq$ 100%), all incoming lanes will be able to access the intersection, enough to satisfy the minimum execution demand for all lanes, thereby avoiding spillback if possible.

Now, we analyze the worst-case intersection crossing delay that a vehicle may be subjected to under our approach.

Lemma 1. For a lane L_i during the j^{th} cycle, assuming that $a_{i,j}$ is the flow rate of vehicles, $q_{i,j}$ is the existing queue length, i.e., number of vehicles already in L_i, at the start of the j^{th} cycle, and z_i is the capacity. Let $n_{out_{i,j}}$ be the number of vehicles that are dispatched from

L_i in a cycle of length T_c. Then, a spillback is avoided if

$$n_{out_{i,j}} \geq a_{i,j} \cdot T_c + q_{i,j} - z_i + 1. \tag{4.3}$$

Proof. If $a_{i,j}$ is the flow rate of vehicles in Lane L_i, z_i and $q_{i,j}$ are the lane capacity and the existing queue length during the j^{th} cycle, then in T_c time, $a_{i,j} \cdot T_c$ vehicles will enter the lane L_i. If $n_{out_{i,j}}$ vehicles are dispatched in T_c time, then total number of vehicles in lane L_i in the j^{th} cycle will be $a_{i,j} \cdot T_c - n_{out_{i,j}} + q_{i,j}$. To avoid spillback, the total number of vehicles have to be less than the capacity z_i. Therefore,

$$\begin{gathered} a_{i,j} \cdot T_c - n_{out_{i,j}} + q_{i,j} < z_i \\ \Rightarrow n_{out_{i,j}} > a_{i,j} \cdot T_c + q_{i,j} - z_i \\ \Rightarrow n_{out_{i,j}} \geq a_{i,j} \cdot T_c + q_{i,j} - z_i + 1. \end{gathered}$$

(The integer 1 is added on the last step, since the number of vehicles is an integral value.)

□

Theorem 1. Assuming that the flow rate $= a_{i,j}$, $i = 1, \ldots, n$ and $\forall j' > j$, the wait time $W_{k,i,j}$ for a vehicle at the k^{th} position in lane L_i at the j^{th} cycle is bounded by

$$W_{k,i,j} \in \left[0, \sum_{j=1}^{\left\lfloor \frac{k}{n_{out_{i,j}}} \right\rfloor} (T_c - U_{i,j_{min}} \cdot T_c) \right]. \tag{4.4}$$

where $n_{out_{i,j}}$ denotes the minimum number of vehicles dispatched from L_i in j^{th} cycle, and all other variables are as defined previously.

Proof. A vehicle entering lane L_i will have the longest wait time when it joins the queue exactly when the light turns from green to red. Since our underlying approach follows a round robin policy, lane L_i will get to access the intersection after all the other three phases have consumed their allotted budget. As discussed earlier, during the period T_c in the j^{th} cycle, the entire budget is distributed among the four phases only. Hence this vehicle will have to wait for at most $T_c - U_{min_{i,j}} \cdot T_c$ time, before its lane is able to utilize the intersection again. $U_{min_{i,j}}$ can be calculated by finding the minimum number of vehicles, $n_{out_{i,j}}$, to be dispatched in T_c duration from lane L_i as in (4.3). Hence, the time required to dispatch n_{out_i} vehicles is: $C_{min_{i,j}} = h \times n_{out_{i,j}} + t_l$ from Equation (3.3), and

$$U_{min_{i,j}} = \frac{h \times (a_{i,j}T_c + q_{i,j} - z_i + 1) + t_l}{T_c}. \tag{4.5}$$

The worst case wait time will occur when the lane under consideration L_i receives minimum utilization assignment $U_{min_{i,j}}$. Since, only $n_{out_{i,j}}$ number of vehicles in front of the k^{th} vehicle will be able to enter the intersection. Thus, the total worst case wait time for the vehicle at the k^{th} position in lane L_i at the j^{th} cycle is

$$W = \sum_{j=1}^{\left\lfloor \frac{k}{n_{out_{i,j}}} \right\rfloor} (T_c - U_{i,j_{min}} \cdot T_c).$$

Clearly, the best case wait time for a vehicle is zero, which will occur when the vehicle arrives at the intersection when the light is already green and there is no queue in front of it. The vehicle then proceeds to cross the intersection immediately.

The wait time $W_{k,i,j}$ for a vehicle at the k^{th} position in lane L_i at the j^{th} cycle is hence

$$W_{k,i,j} \in \left[0, \sum_{j=1}^{\left\lfloor \frac{k}{n_{out_{i,j}}} \right\rfloor} (T_c - U_{i,j_{min}} \cdot T_c)) \right].$$

$\square$

Chapter 5

Hardware Experimentation and Large Scale Simulations

This chapter describes the validation of our approach on a hardware testbed consisting of small robots that emulate vehicular traffic for an urban intersection. Pre-timed traffic control, adaptive elastic traffic control and our proposed approach are implemented on the testbed. Here, we show the comparison between our approach and the elastic adaptive traffic control. Similar results were obtained when pre-timed traffic control technique was implemented, with increased spillbacks.

5.1 Hardware Setup

Hardware setup and the flow of data and commands through our experimental setup is described briefly in Figure 5.1. Our experimental testbed consists of 30 small size robots, each representing a vehicle. Each robot, henceforth referred as vehicle, is affixed with multiple IR markers. These markers are tracked by the Optitrack motion capture system consisting of 22 IR cameras. All 22 Optitrack IR cameras in the environment are connected to a Windows 10 machine running the proprietary Optitrack Motive software where the captured camera frames are processed and the IR marker positions are made available [42]. This position data is then streamed to a command computer running Ubuntu 16.04 that runs our

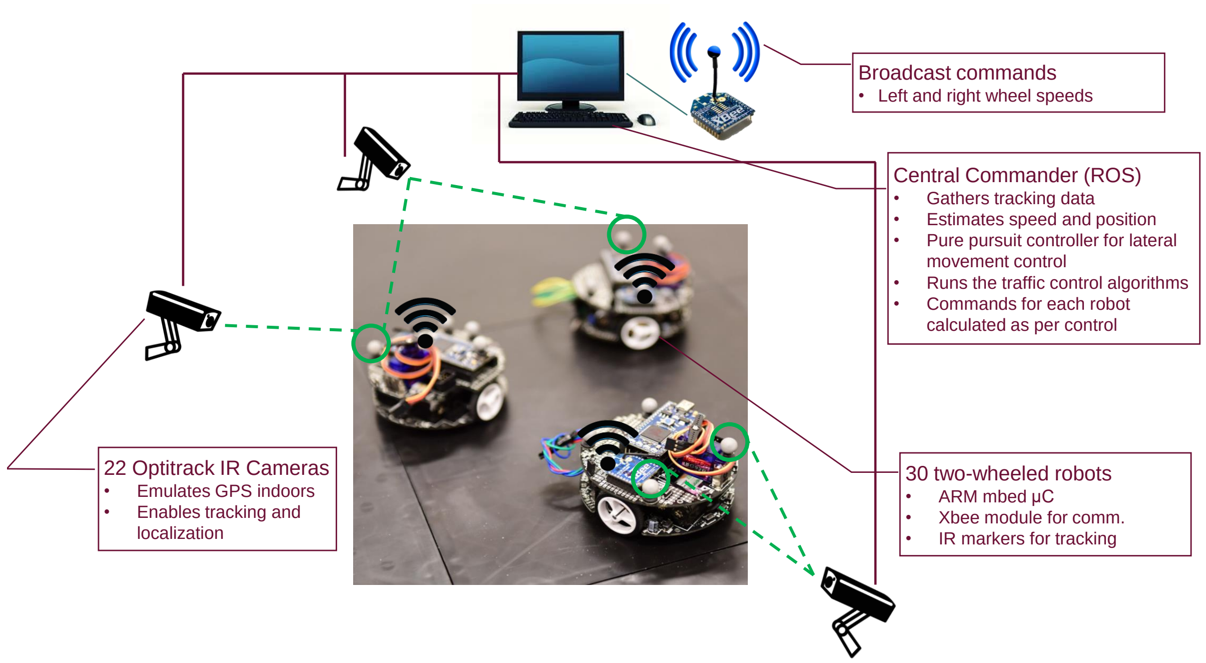

Broadcast commands
• Left and right wheel speeds
Central Commander (ROS)
• Gathers tracking data
• Estimates speed and position
• Pure pursuit controller for lateral movement control
• Runs the traffic control algorithms
• Commands for each robot calculated as per control
22 Optitrack IR Cameras
• Emulates GPS indoors
• Enables tracking and localization
30 two-wheeled robots
• ARM mbed μC
• Xbee module for comm.
• IR markers for tracking

Figure 5.1: Hardware setup description.

interface application. This interface application utilizes Robot Operating System (ROS) [44] and its publisher-subscriber framework to make the position data available on nodes called ”topics”. One topic is dedicated to each vehicle’s position data. Our interface application also consists of our controller application implemented on ROS that further processes this position data by subscribing to these topics. The controller application processes the position data and sends control commands (left wheel and right wheel velocity) accordingly to each individual vehicle.

5.1.1 Controlling multiple vehicles in the environment

The controller application uses ROS to command the vehicles in the following manner:

- The raw position data from the software is processed using an extended Kalman filter to reduce camera sensor noise and accurately estimate the position in 2-D space as well as the velocity.

- Pre-planned map with path coordinates resembling eight lanes entering and exiting an intersection (as shown in Figure 5.2) are stored in the database.
- Depending on the estimated position of the vehicle in the testbed, one path is assigned to it out of the eight available paths (lanes).
- A pure pursuit controller [8] utilizes the estimated location of the vehicle , as well as the assigned path coordinates to calculate the angular velocity command required for each vehicle to stay in its defined path.
- The estimated data of all vehicles is used to calculate the relative distance between the consecutive vehicles. This is then fed to a high level controller which implements the intelligent driver model (IDM) [53].
- IDM calculates the acceleration values for each vehicle depending on the relative distances. IDM is a widely used car following model used to emulate freeway and urban traffic driving, and the acceleration value output closely resembles human driving conditions and reaction delays as per various tuning parameters explained in [53].
- As the vehicles only act upon instantaneous velocity commands, these acceleration values along with current measured velocities are used to calculate the desired velocities for each vehicle. The desired linear velocities for the vehicles are achieved using a PI controller which acts as our low level controller. This controller calculates the linear velocity commands for each vehicle such that the measured and the desired velocities match.

5.1.2 Emulating human-driven vehicles at an intersection using robots

Each vehicle is represented by a robot consisting of a 32-bit ARM-based mbedNXP LPC1768 microcontroller on the Pololu m3pi platform interfaced with Digi Xbee receivers. The corresponding Xbee transmitter is connected to the command computer. These Xbee modules establish a wireless communication channel using the Zigbee protocol over which the angular and linear velocity commands calculated for each vehicle using our controller application are broadcast. The firmware on these vehicles receive the broadcast messages and calculate the left and right wheel speeds from the received angular and linear velocities as per the differential drive kinematics model. Similar setup have been used to emulate and study the behaviour of vehicles in a realistic environment [43].

To replicate human driving behavior using robots, the reaction delays pertaining to traffic flow and changing traffic lights (t_l and h, explained in Chapter 3) are incorporated in our controller application. By combining the time headway for dispatching traffic and IDM, vehicle traffic flow at an intersection can be emulated. Similar approach is presented in [29], where the saturation headway model for dispatching traffic at a signalized intersection is used to tune the parameters of IDM. IDM is also used to study drivers' intent in terms of left and right turn speeds at an urban intersection [34]. Using the position and speed data for each vehicle, IDM can thus ensure that the robots can replicate vehicles flowing through a signalized intersection.

5.1.3 Replicating a signalized intersection in the environment

Figure 5.2 shows our hardware setup with lane markings superimposed on the image for clarity. While our setup does not have physical traffic lights, the vehicle positions are tracked and are commanded to stop if the lane does not have access to the intersection (traffic light for the lane is expected to be red). To keep the vehicles on their desired path, the start point, end point and 100 equispaced path coordinates in between for all eight possible lanes in our setup are stored in a database. Before the start of experiments, the vehicles are placed

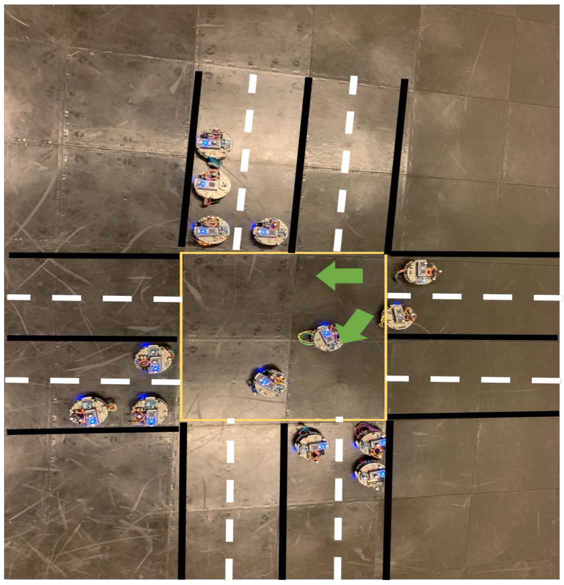


Figure 5.2: Hardware setup emulating an urban intersection.

in the setup. Depending on their start location and its proximity to a stored coordinate, a lane is assigned to that vehicle. Using the pure pursuit controller, it is ensured that once a lane is assigned, the vehicle does not deviate from its path and follows the pre-determined lane movements. If the vehicles are joining an existing queue, they stop at a safe distance governed by the IDM parameters. Once the light is expected to turn green, the vehicles are commanded to accelerate as per the reaction delay parameters. A video of our experiments is uploaded here [1].

5.2 Observations from Hardware Experiments

The results obtained for the best case flow where all the vehicles arrive and join the queue right at the beginning of the green time with medium traffic flow are discussed. Nevertheless, the data presented here are generally representative. Figures 5.3 and 5.4 show the distance of all the vehicles in lane L_3 from L_3's stop line as a function of time. A decreasing (increasing)

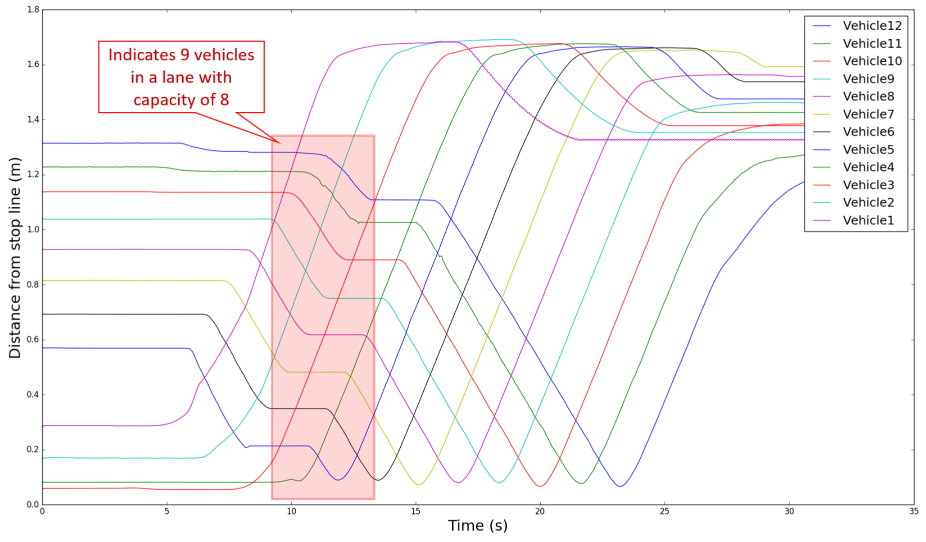


Figure 5.3: Distance from stop line as a function of time for vehicles in lane L_3 using the elastic scheduling technique and medium flow.

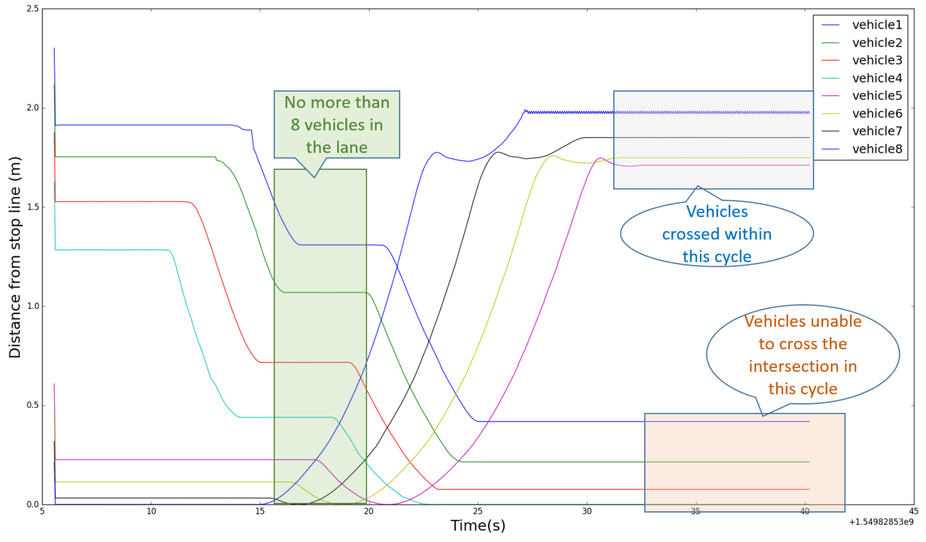


Figure 5.4: Distance from stop line as a function of time for vehicles in lane L_3 using our server based approach

distance over time indicates that a vehicle is moving closer to the stop line to cross the intersection (away from the intersection). A constant distance over time is an indication that the vehicle is stationary.

From Figure 5.3, it can be observed that using the elastic adaptive control technique results in more than eight vehicles queuing up in L_3, between time 9 s and 13 s, causing a spillback, when the lane capacity is set to 8 vehicles. In contrast, from Figure 5.4 it can be seen that, since the vehicle flow rate is 6 veh/min, and the existing queue length is 4 veh in L_3, the server based approach selects the cycle time of 40 s, such that the number of vehicles in L_3 do not exceed the capacity of 8 vehicles. Even though four vehicles were not able to clear the intersection in this cycle, spillbacks do not occur. These hardware experiments prove that our proposed server based approach can function in a realistic vehicular environment. Due to spatial restrictions and limited number of vehicles in our hardware setup, we also perform large-scale simulations to ensure that our approach can be used even with continually varying traffic conditions and patterns.

5.3 Large Scale Simulation

We compare our approach against (i) a widely deployed pre-timed traffic control technique [10], and (ii) an adaptive control technique based on real-time phase saturability [20]. The pre-timed control uses the commonly recommended cycle lengths of 60, 90, and 120 s. In addition, we assumed a more intelligent pre-timed technique where the green times within a cycle length are adjusted according to the incoming flow from different phases by ensuring that the lanes with higher vehicle flow get a longer green time. More sophisticated techniques [22, 33, 51] consider network-wide optimization, which suffer from scalability issues and are not practical as discussed earlier in Chapter 2.

Three different types of incoming traffic flow patterns with varying traffic flow rates were considered to validate the adaptability of our approach to varying flow patterns.

- Best case flow: all the vehicles arrive and join the queue right at the beginning of the green time
- Average case flow: all the vehicles have uniform arrival times
- Worst case flow: all the vehicles that are expected to arrive join the queue right at the end of the green time

To permit a comprehensive comparison of varying traffic flow through different incoming lanes of the intersection, we simulate vehicle flow rates varying from 1–7 veh/min (60–420 veh/hr). These flow rates illustrate different realistic critical volume-to-capacity ratios for a signalized intersection, as provided by the Federal Highway Administration [16]. We have,

- Flow rate of 1–4 veh/min (60–240 veh/hr) – an intersection running under capacity with reduced delays
- Flow rate of 4–6 veh/min (241–360 veh/hr) – an intersection running near capacity where delays and queue buildups are expected
- Flow rate up to 7 veh/min (420 veh/hr) – an intersection with unstable flows and wide range of delays

Vehicle flow rate of 8 veh/min or more entirely disrupts the intersection as the demand exceeds the capacity. Hence, our simulation results compare vehicle flows up to 7 veh/min.

5.3.1 Simulation Specifications

Our traffic simulator is implemented in Python 3, which takes the following parameters for all eight lanes as input:

- Lane capacities (number of vehicles)
- Initial residual queues (number of vehicles)
- List of flow rates (vehicles per minute)
- Cycle time (s)
- Simulation time (s)
- Flow type - Best, Average, Worst
- Traffic Algorithm - Pre-timed, Elastic scheduling, Server based approach

Once these parameters and variables are set as per user input, the green time intervals and cycle times are calculated as per the expected traffic flow and the traffic control algorithms are then used to manage the traffic.

- Defining cycle time - The list of flow rates contains the vehicle flow rate per minute for all the lanes. If the simulation is to be executed for one hour (60 minutes), each lane will have a list of flow rates associated to it, such that each list has 60 entries. The flow rate changes every minute as per the next entry in the list. Using the above mentioned parameters, the simulator then performs calculations as per the traffic control algorithm. While the cycle time for the pre-timed and elastic scheduling approach are fixed, the server based algorithm uses setCycleTime method that calculates the spillback times for each phase and sets the minimum spillback time as the cycle time for the next cycle.

- Calculating green time intervals for each phase - The simulator calculates the load of vehicles expected for the next traffic cycle. If the arrival rate of a lane is constant and set to 4 veh/min, residual queue is 2 veh/min, and cycle time set to 120 s, the load in this lane for the next cycle is $4 * (120/60) + 2 = 10$ vehicles. Once the vehicle load is obtained for each lane, the simulator sets the green time for the traffic signals in each phase according to the traffic control algorithm being simulated. In case of fixed timing approach, the cycle time is directly divided into four green time intervals. The smarter pre-timed approach distributes green time as per the traffic intensity (vehicle load). Hence, the green time interval for each phase is found by dividing the cycle time in proportion to the maximum vehicle load in each phase. For the elastic scheduling approach, the green time intervals are mentioned in the paper [20] and are used directly in accordance with the flow rates mentioned. For the server based approach, as mentioned in Chapter 4, using the saturation headway model to calculate the time required to dispatch the incoming vehicle load as well as the spillback times for each phase, the minimum as well as the maximum demands are calculated per phase. With these bounds, the budget for each phase is calculated such that the assigned budget is inversely proportional to its minimum spillback time. This budget represents the proportion of cycle time that will be assigned to a given phase which is equal to the green time interval for that phase.

- Traffic Flow - Once the green time intervals for each phase and the cycle time is obtained, the simulator then switches to a tick based discrete time processing, where the traffic patterns change as per the green-yellow-red timings at every tick (second) until the end of the current traffic cycle. Once the cycle ends, the residual queue and other system variables are updated, and green time interval and cycle time are recalculated for the new cycle.

- The queue inside the lane is continuously tracked at every tick during the simulation and a flag is set every time the queue inside any lane exceeds the defined lane capacity. This way, the simulator tracks the queue spillbacks occurring in the system. This entire process is repeated until the end of the set simulation time.

5.3.2 Comparison with Pre-Timed Traffic Control Technique

Table 5.1: Average delay in seconds experienced by vehicles using server based approach vs. pre-timed adaptive approach with cycle lengths of 60, 90 and 120 s

		Flow Rate (veh/min)						
	Type	1	2	3	4	5	6	7
Server Based	Avg	61.19	63.86	53.66	43.86	40.11	36.79	67.09
	Best	2.94	1.6688	1.411	1.307	1.255	1.19	1.15
	Worst	45.82	46.13	31.29	23.83	19.35	16.35	46.17
60	Avg	23.69	32.125	32.74	34.88	41.09	580.55	334.06
	Best	2.98	3.92	2.78	2.7	15.02	234.3	28.6
	Worst	67.67	45.9	34.84	23.72	20.52	600.52	32.58
90	Avg	37.63	38.44	34.34	35.46	46.46	44.73	141.76
	Best	36	1.87	1.58	1.36	1.31	1.24	1.299
	Worst	67.67	45.9	34.84	23.72	20.52	16.29	15.559
120	Avg	61.19	63.85	54.4	55.32	60.06	64.54	67.09
	Best	2.94	1.6688	1.411	1.307	1.255	1.215	12.6
	Worst	45.82	46.13	31.29	23.83	19.35	16.35	15.35

Table 5.1 shows the the average delay experienced by the vehicles when the capacity of the lane is 10 veh and the incoming flow varies from 1–7 veh/min (60–420 veh/hr across all lanes). Since there are 8 incoming directions, 8–56 veh aim to cross the intersection per minute, i.e., flow of up to 3,360 veh/hr through the intersection from different directions. A constant flow of 56 veh/min can saturate the intersections when the lane capacities are low, causing the queues to spillback, as is observed in both approaches when the flows from all directions exceed 7 veh/min. The highlighted cell entries in Table 5.1 indicate that queue spillbacks were observed.

For lighter flow rates (less than 5 veh/min), our approach does not reduce delays associated with intersection crossings, as we aim to avoid spillbacks. As a result, more vehicles are accumulated in their lanes before they are dispatched with a larger cycle and green times. However, for heavier flow rates (5–7 veh/min), the pre-timed adaptive approaches lead to spillbacks (marked in red in Table 5.1). In contrast, our approach does not experience any spillback and in fact always reduces the average delays. The only exception is the average-case with flow rate of 7 veh/min. Even then, our approach does not perform worse than the pre-timed adaptive approaches.

Figures 5.5 and 5.6(a) show the average delay experienced by the vehicles when the incoming flows from different directions are unbalanced. For this case, we considered low traffic volumes (1–3 veh/min) in two phases and medium to heavy traffic volumes (6–8 veh/min) in the other two phases, to simulate minor and major arterials. By varying lane capacity, it can be seen in Figure 5.5 that our approach experiences spillbacks only when the net capacity is very small (7 veh) and the incoming flow rate is high (8 veh/min), i.e., when a spillback cannot be avoided. For capacity of 8–14 veh, all pre-timed adaptive approaches experience heavy spillbacks in at least one of the three types of flow patterns. As the capacity increases, spillbacks become less of an issue, as expected. Figures 5.6(a) and 5.6(b) show the delays experienced by the vehicles with dashed line indicating queue spillbacks occuring in the case of average vehicle flow.

In addition, as seen in Figure 5.6(a), our approach ensures that the average delays remain reasonable and are comparable to the delays when using the pre-timed adaptive approach with 90 and 120 s cycle lengths. It is also clear that the cycle length of 60 s is not able to handle the load imbalance with incoming traffic of 8 veh/min and hence shows large delays due to spillbacks. Our simulations show that our server based approach ensures that a spillback is avoided in cases when it is possible to do so, while also showing 10–50%

improvement in average delays experienced by the vehicles in most cases.

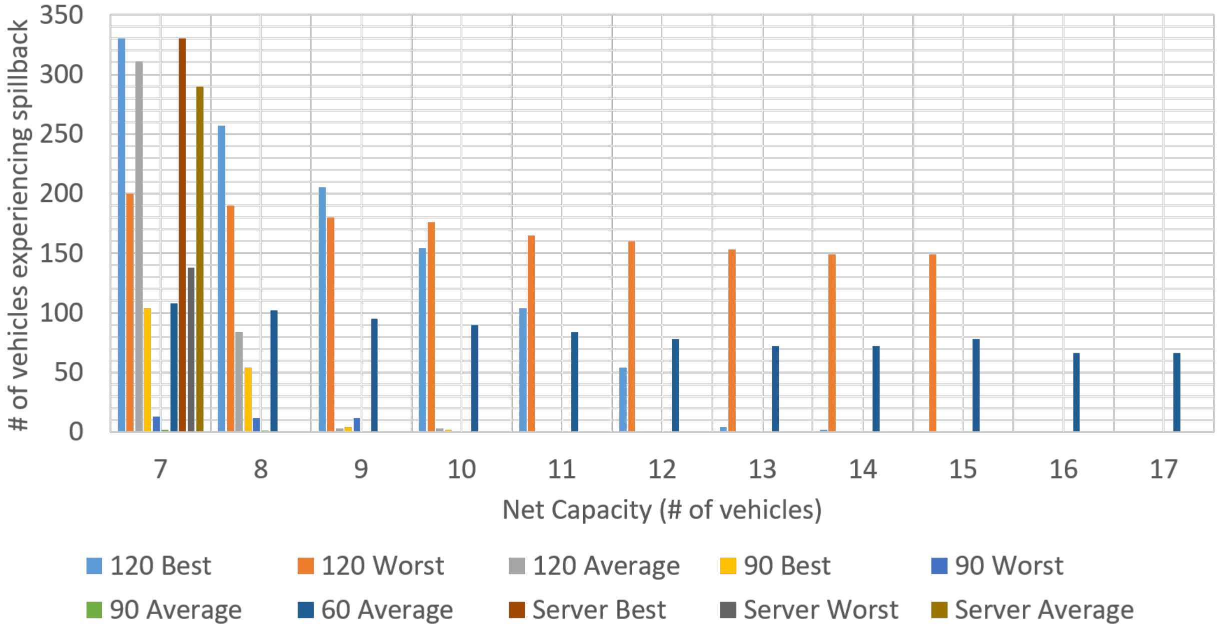
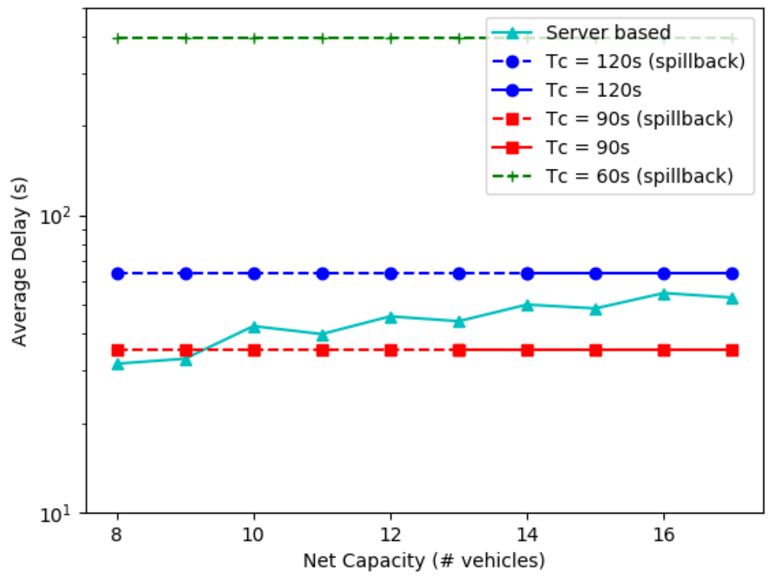
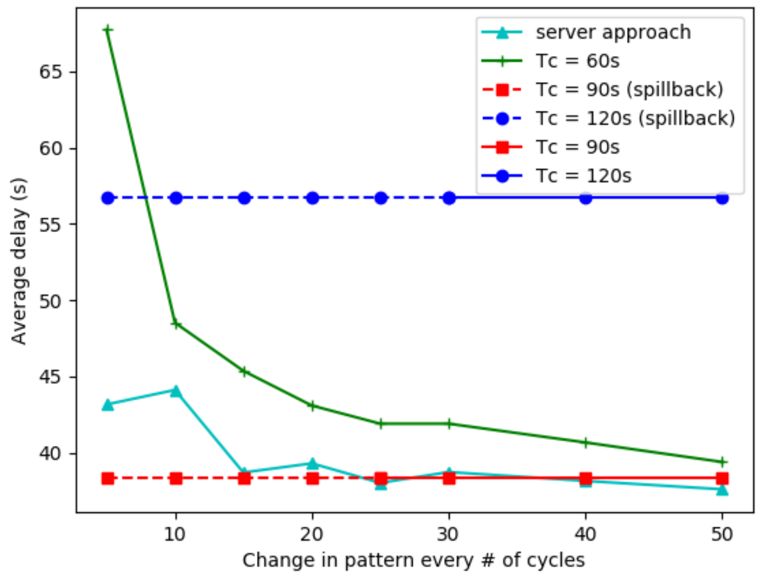


Figure 5.5: Number of vehicles experiencing spillback under varying lane capacity with unbalanced flow rates.

Finally, to test the adaptability of our approach to varying and fluctuating traffic, we ran the simulations for 100 traffic cycles with an average case traffic flow of 4 veh/min and a sudden increase of traffic to 8 veh/min between every 5–50 cycles. Figure 5.6(b) shows that the average delay experienced by the vehicles in the case of traffic controllers with 120s and 90s cycle length is constant, since the green intervals are long enough to dispatch all vehicles from the queue. However spillbacks were observed since more vehicles are accumulated during the red time, thereby exceeding the lane capacity. While spillbacks were not observed when the cycle length is 60 s, long average delays of 67.72 s were observed, especially when the flow rate frequently changes. When using our approach, no spillbacks were observed and the average delay is below 40 s in most cases, except with very frequent traffic surges (every 5 and 10 cycles).



(a) Constant unbalanced vehicle flow.



(b) Frequently changing unbalanced vehicle flow.

Figure 5.6: Average delays incurred by vehicles with server based approach vs. pre-timed adaptive control.

5.3.3 Comparison with Adaptive Elastic Traffic Signal Control

Table 5.2: Average delay(s) experienced by vehicles using server based vs. adaptive elastic scheduling based approaches [20]

Flow Property		Lane Capacities (veh)													
		8		9		10		11		12		13		14	
Type	Pattern	Server	Adaptive	Server	Adaptive	Server	Adaptive	Server	Adaptive	Server	Adaptive	Server	Adaptive	Server	Adaptive
Medium Flow	Best	12.0	8.0	3.4	8.0	3.89	8.0	3.89	8.0	4.3	8.0	4.33	8.0	4.84	8.0
	Avg	23.6	28.7	26.3	28.7	32.3	28.7	35.1	28.7	28.6	28.7	30.0	28.7	30.3	28.7
	Worst	35.9	50.5	39.5	50.5	43.5	50.5	47.5	50.5	51.5	50.5	55.5	50.5	49.5	50.5

Flow Property		Lane Capacities (veh)													
		20		21		22		23		24		25		29	
Type	Pattern	Server	Adaptive	Server	Adaptive	Server	Adaptive	Server	Adaptive	Server	Adaptive	Server	Adaptive	Server	Adaptive
Heavy Flow	Best	22.4	27.11	20.96	27.11	18.0	27.11	13.33	27.11	8.33	27.11	8.27	27.11	8.29	27.11
	Avg	70.1	57.2	42.2	57.2	51.45	57.2	63.3	57.2	50.44	57.2	48.1	57.2	42.63	57.2
	Worst	28.19	11.24	22.14	11.24	17.13	11.24	15.31	11.24	12.54	11.24	10.74	11.24	10.12	11.24

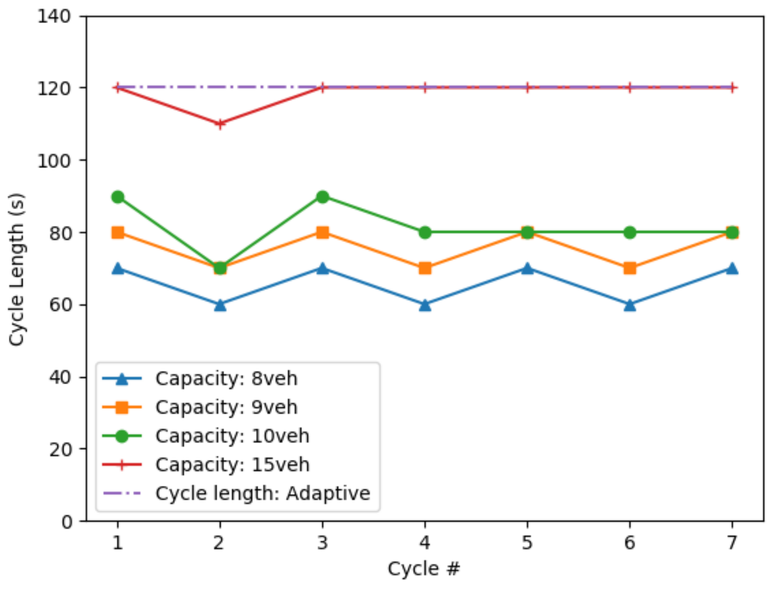
Han et al. [20] provided an adaptive technique using elastic scheduling to fine-tune the timing parameters of a signalized intersection. The authors implement their approach on three different types of flows, i.e. light, medium and heavy. Their scheduling technique is called "elastic" scheduling since they provide a range of allowable green times and select an optimum green time for different phases depending on the variations in the traffic pattern. They also calculate an optimum cycle length for a traffic signal, for a given number of vehicles entering the system per lane (for three different flow types).

As shown in Table 5.2, we compare our proposed server based approach with the adaptive technique by calculating the average delay incurred by the vehicles in the system under medium and heavy vehicle flow types. Based on our results, queue spillbacks occur when the optimal green times and cycle times provided in [20] are used under varying capacities. The results from the light traffic flow type are not shown as with such low traffic flow rate, the both approaches perform similarly and queue spillbacks do not occur. Similar trend was also be observed in case of pre-timed control as shown in Table 5.1, where lower flow rates result in low delays without queue spillbacks. The cell entries in Table 5.2 highlighted in red indicate that queue spillbacks were observed in that scenario.

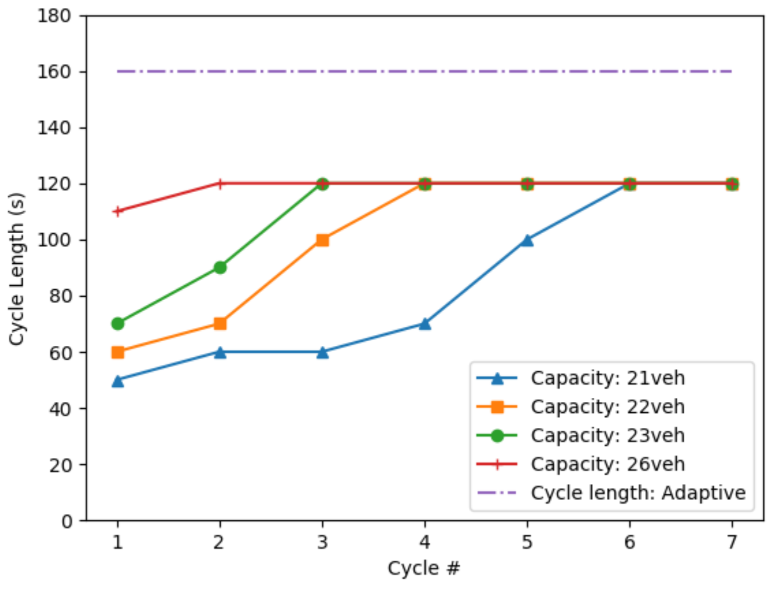
Table 5.2 shows that in case of the adaptive approach, queue spillbacks are observed with capacities less than 14 veh in case of medium traffic flow and 29 veh in case of heavy traffic in all three flow patterns, i.e. best, worst and average. While in case of our server based approach, spillbacks are avoided except when the lane capacity is too small to fit the heavy incoming traffic (capacity of 20 veh just falls short of accommodating an incoming traffic of 24-28 veh per traffic cycle). Even though in this case all three approaches fail at avoiding spillbacks, the queues experience spillback within 47 s and 68 s under the pre-timed control and the elastic adaptive control techniques respectively, while the server based approach was able to delay the spillback to 173 s.

Finally, Figure 5.7(a), 5.7(b) show the variable cycle lengths in the server based approach vs. fixed cycle length of adaptive approach, under medium and heavy flow type and average vehicle arrival pattern. In our proposed approach, the cycle length is recalculated according to the incoming traffic as well as the residual queues in the lanes change. It can be noticed that as the lane capacities increase, the server based approach has more laxity in managing the traffic since the possibility of queue spillback reduces and hence has less variations in cycle lengths. Due to this changing cycle lengths, along with varying green times, our server based approach is able to avoid spillbacks while still guaranteeing performance in terms of wait times of the vehicles. Since there is a constant flow of incoming traffic, the cycle lengths for both cases shown in Figure 5.7(a), 5.7(b) tend to settle at a stable value. This happens once the residual queues are dispatched and the lanes have balanced vehicle occupancy. Since, the adaptive approach presented in [20] does not continually update the cycle lengths with the incoming traffic, residual queue and lane capacity, frequent spillbacks are observed.

It is important to note that in a few cases the delay experienced by the vehicles in the server based approach are slightly higher than the existing approaches, but none of the queues spillback. Queue spillbacks are known to cause disruptive effects on a transportation



(a)



(b)

Figure 5.7: Change in cycle lengths as capacities and incoming vehicles change in server based approach vs. adaptive approach

network which can also impede the emergency response vehicles' paths and cause delays in their response times. Hence, it is a fair tradeoff to have a small increase in travel time than to experience spillbacks, as while an increase in travel time reduces drivers' satisfaction but does not adversely effect the entire transportation network.

Chapter 6

Applicability in a Realistic Environment

The proposed server based approach requires updated traffic information from all neighboring intersections to be made available at each traffic controller that runs the Intersection Manager (IM). This can be fulfilled using currently deployed infrastructure that includes road-side sensors and data recorders, as well as traffic controllers that have the capability of running minimal traffic management softwares like UTCS [15]. All intersections nowadays have a traffic controller which provides an interface with the sensors, detectors as well as the traffic lights (I/O). These controllers have a configuration software upon which a modular traffic management solution like our server based approach can be implemented, that can process the information available from the I/O modules and control the lights as per the algorithm implemented. Intersection controllers such as ATC eX 2070 [24] and 2070EN1 [25] are equipped with processor capabilities to support a full Linux distribution and a Microware OS-9 Real-Time OS respectively. Such intersection controllers showcase 32-bit, up to 667MHz processors with 16MB Flash memory, 128MB RAM with options to add memory expansions for additional storage. They support open architecture and real-time platforms with dedicated multitasking abilities to handle complex software applications. Since our proposed approach employs a simple budget calculation and distribution algorithm, the available memory and the processor capabilities in these controllers are sufficient to program our control technique. Such controllers already support flexible operational parameters such as cycle times, phase sequences and timing plans to be continuously updated and monitored by an intersection control software. Since the server based approach requires changes in the cycle times and green times, such controllers can be used to deploy our approach in a realistic setup. Traffic controllers at multiple intersections currently are clustered into smaller groups in various places where the traffic controllers optimize traffic for multiple intersections at once. Newer upgraded controllers are also enabled with intersection configuration software [23] which allows modular controller applications to have web browser support and peer to peer I/O sharing between intersection controllers. While the available centralized control approaches have shown significant drawbacks in optimizing traffic, the available data sharing capabilities can be exploited in implementing our IM in a real-world setting.

Our server based approach requires updated traffic data at the end of each traffic cycle. A traffic cycle in our technique will not be less than 16 s, since this is the minimum amount of time required to dispatch at least one vehicle from all four incoming phases, considering the response delays of the human-drivers (as per saturation headway model discussed in Chapter 3). This is also an extreme case where the traffic from all directions is extremely low. During our simulations, the cycle length is usually found to be above 40 s (shown in Figure 5.7). Sensors in the intersection network usually have a sampling rate of 20 Hz. Even after accounting the transmission and processing delays, the sampling rate of the sensors and video detectors currently deployed will be sufficient to make the data available for processing at the end of each cycle for our server based approach.

The intersection controllers [24, 25] being developed for the smart traffic market also enable direct connectivity to video detection mechanisms (such as cameras) as well as up to 128 detectors and sensors per intersection to detect and estimate traffic flow. Along with a dedicated connection with the data recorders, the configuration software application also provide robust data validation and estimation modules that ensure data accuracy and consistency [23]. While the available sensors do have drawbacks and limitations with changing weather and traffic conditions, the data still can be precisely measured using various advanced approaches that employ cellular and GPS connectivity of devices to accurately estimate traffic flow. Newer vehicles in the market also have cellular connectivity which can be used to improve on the accuracy of the data [28]. The proposed real-time server based approach requires per phase calculations and control of green time intervals. The available controllers support up to 16 flexible phase sequences making it possible to modify our model as per the phase sequence requirements of a real intersection. Precise timing measurements are required when a real-time software is being implemented. On-board real-time clocks (RTCs) with GPS synchronization for accurate timing characteristics ensure that real-time approaches can be implemented using such controllers.

Using advanced vehicular communication protocols like DSRC and CV2X that are currently being researched and rigorously tested, these communication networks can be further stabilized and made robust to ensure that our approach received accurate traffic information in a timely manner. Since the traffic controllers already have processing capabilities and web connectivity support, a minimal cloud connectivity can also be established among a small group of intersections to share this traffic information [17]. Using the existing data collection methods and by also employing the advanced approaches, it can be ensured that the proposed approach receives the relevant input parameters timely and accurately to perform the calculations and make timing decisions at an intersection.

The extensive hardware experiments and large scale simulations prove that our approach will show promising improvements in traffic congestion, as can be seen in Chapter 5. The command computer in our hardware setup tracks the position of each vehicle in the environment individually. In a realistic scenario, a human driver only considers the vehicle in front to make decisions in terms of speed and safe spacing between the vehicles. This is replicated using our controller application that performs individual isolated calculations for each vehicle by only considering the position data for the current vehicle and the vehicle in front. By addressing each vehicle in the system individually and using IDM which is tuned to replicate human driving behavior that maintains a desired safe spacing between the vehicles, the controller depicts realistic vehicle flow for an urban signalized intersection. IDM along with pure pursuit controller that assigns pre-planned path coordinates to each vehicle also exactly mimics a human-driver that is driving to a destination whose path is already known to them or is using GPS to navigate to the destination. This, along with the results observed from large scale simulations that indicate significant reduction in queue spillbacks while also improving travel time delays show that the proposed real-time server based approach can be deployed in a realistic environment to alleviate traffic congestion issues.

Chapter 7

Conclusion

In this manuscript, we modeled traffic control at a signalized intersection as a real-time scheduling problem. By using sporadic servers to schedule vehicles needing to cross an intersection, we formulated an approach to calculate the cycle length and distribute budget for the servers such that queue spillbacks are minimized, if not avoided. We also provided a worst-case delay analysis for the vehicles crossing the intersection when using our approach. With the help of simulations, we compared our approach with an intelligent pre-timed traffic control as well as an adaptive elastic traffic control technique. It is observed that, with the proposed approach, spillbacks were avoided when possible, even with unbalanced incoming traffic and reduced lane capacities. Results also showed a 10–50% improvement in average delay experienced by the vehicles as compared to the pre-timed control and the adaptive elastic control techniques. Our experiments on a hardware testbed provide similar conclusions proving that our approach can be deployed in a realistic urban traffic environment and can show significant reduction in traffic congestion. In our future work, we plan on deriving how to best allocate the leftover budgets and consider more than one intersection.

Bibliography

[1] Anonymous. Server based traffic control experiments, June 2018. URL https://www.youtube.com/channel/UCI-UGJKT7C5E_8bs391LCpA.

[2] A. Biondi, G. C. Buttazzo, and M. Bertogna. Schedulability analysis of hierarchical real-time systems under shared resources. IEEE Transactions on Computers, 65(5): 1593–1605, May 2016. ISSN 0018-9340. doi: 10.1109/TC.2015.2444833.

[3] Giorgio C Buttazzo. Hard real-time computing systems: predictable scheduling algorithms and applications, volume 24. Springer Science & Business Media, 2011.

[4] D. Carlino, S. D. Boyles, and P. Stone. Auction-based autonomous intersection management. In 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), pages 529–534, Oct 2013. doi: 10.1109/ITSC.2013.6728285.

[5] Yixin Chen. Model of traffic speed-flow relationship at signal intersections. Open Journal of Applied Sciences, 7(06):319, 2017.

[6] Mario Collotta, Lucia Lo Bello, and Giovanni Pau. A novel approach for dynamic traffic lights management based on wireless sensor networks and multiple fuzzy logic controllers. Expert Systems with Applications, 42(13):5403–5415, 2015.

[7] Graham Cookson and Bob Pishue. Global traffic scorecard. Technical report, Technical report, INRIX, 2017.

[8] R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[9] Umamaheswari C Devi and James H Anderson. Soft real-time scheduling on multiprocessors. University of North Carolina at Chapel Hill, 2006.

[10] Francois Dion, Hesham Rakha, and Youn-Soo Kang. Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections. Transportation Research Part B: Methodological, 38(2):99–122, 2004.

[11] Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS '04, pages 530–537, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 1-58113-864-4. doi: 10.1109/AAMAS.2004.190. URL https://doi.org/10.1109/AAMAS.2004.190.

[12] Dario Faggioli, Marko Bertogna, and Fabio Checconi. Sporadic server revisited. In Proceedings of the 2010 ACM Symposium on Applied Computing, pages 340–345. ACM, 2010.

[13] Yiheng Feng, K Larry Head, Shayan Khoshmagham, and Mehdi Zamanipour. A real-time adaptive signal control in a connected vehicle environment. Transportation Research Part C: Emerging Technologies, 55:460–473, 2015.

[14] FHWA. Intersection Safety, Safety Data and Analysis. https://highways.dot.gov/research-programs/safety/intersection-safety, 2018. [Online; accessed 18-January-2019].

[15] FHWA. Traffic detector handbook. https://www.fhwa.dot.gov/publications/research/operations/its/06108/03.cfm, 2018. [Online; accessed 18-January-2019].

[16] FHWA-USDoT. Signalized intersections: Informational guide, April 2019. URL https://www.fhwa.dot.gov/publications/research/safety/04091/07.cfm.

[17] Mario Gerla, Eun-Kyu Lee, Giovanni Pau, and Uichin Lee. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In 2014 IEEE world forum on internet of things (WF-IoT), pages 241–246. IEEE, 2014.

[18] Thomas F Golob and Wilfred W Recker. Relationships among urban freeway accidents, traffic flow, weather, and lighting conditions. Journal of transportation engineering, 129 (4):342–353, 2003.

[19] Urban Bikeway Design Guide. National association of city transportation officials. New York, 8, 2011.

[20] Shi-Yuan Han, Fan Ping, Qian Zhang, Yue-Hui Chen, Jin Zhou, and Dong Wang. Global adaptive and local scheduling control for smart isolated intersection based on real-time phase saturability. In De-Shuang Huang, Kang-Hyun Jo, and Juan Carlos Figueroa-García, editors, Intelligent Computing Theories and Application, pages 730–739, Cham, 2017. Springer International Publishing.

[21] Jean-Jacques Henry, Jean Loup Farges, and J Tuffal. The prodyn real time traffic algorithm. In Control in Transportation Systems, pages 305–310. Elsevier, 1984.

[22] PB Hunt, DI Robertson, RD Bretherton, and RI Winton. Scoot-a traffic responsive method of coordinating signals. Technical report, 1981.

[23] McCain Inc. Omni ex intersection control. https://www.mccain-inc.com/products/software/intersection-control/omni-ex-intersection-control-software, 2018. [Online; accessed 18-January-2019].

[24] McCain Inc. Omni ex atc controllers. https://www.mccain-inc.com/products/controllers/atc-ex-controllers/atc-ex-2070-controller, 2018. [Online; accessed 18-January-2019].

[25] McCain Inc. 2070e controller. https://www.mccain-inc.com/products/controllers/2070-controllers/2070e-controller, 2018. [Online; accessed 18-January-2019].

[26] Cameron Kergaye, Aleksandar Stevanovic, and Peter T Martin. An evaluation of scoot and scats through microsimulation. In International Conference on Application of Advanced Technologies in Transportation, Transportation and Development Institute, Athens, Greece, 2008.

[27] Peter Koonce et al. Traffic signal timing manual. Technical report, United States. Federal Highway Administration, 2008.

[28] Milan Kovac and Andrea Leskova. Innovative applications of cars connectivity network–way to intelligent vehicle. Journal of Systems Integration, 3(4):51–60, 2012.

[29] Tamás Kovács, Kálmán Bolla, Rafael Alvarez Gil, Edit Csizmás, Csaba Fábián, Lóránt Kovács, Krisztián Medgyes, József Osztényi, and Attila Végh. Parameters of the intelligent driver model in signalized intersections. Tehnički vjesnik, 23(5):1469–1474, 2016.

[30] Jaimyoung Kwon and Pravin Varaiya. The congestion pie: delay from collisions, potential ramp metering gain, and excess demand. In Proceedings of 84th Transportation Research Board Annual Meeting, 2005.

[31] Guillaume Leduc. Road traffic data: Collection methods and applications. 01 2008.

[32] L. Li, D. Wen, and D. Yao. A survey of traffic control with vehicular communications. IEEE Transactions on Intelligent Transportation Systems, 15(1):425–432, Feb 2014. ISSN 1524-9050. doi: 10.1109/TITS.2013.2277737.

[33] Li Li, Yisheng Lv, and Fei-Yue Wang. Traffic signal timing via deep reinforcement learning. IEEE/CAA Journal of Automatica Sinica, 3(3):247–254, 2016.

[34] M. Liebner, M. Baumann, F. Klanner, and C. Stiller. Driver intent inference at urban intersections using the intelligent driver model. In 2012 IEEE Intelligent Vehicles Symposium, pages 1162–1167, June 2012. doi: 10.1109/IVS.2012.6232131.

[35] G. Lipari and S. Baruah. Greedy reclamation of unused bandwidth in constant-bandwidth servers. In Proceedings 12th Euromicro Conference on Real-Time Systems. Euromicro RTS 2000, pages 193–200, June 2000. doi: 10.1109/EMRTS.2000.854007.

[36] Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM (JACM), 20(1):46–61, 1973.

[37] Dong-fang Ma, Dian-hai Wang, Feng Sun, Yi-ming Bie, and Sheng Jin. Method of spillover identification in urban street networks using loop detector outputs. Journal of Central South University, 20(2):572–578, Feb 2013. ISSN 2227-5223. doi: 10.1007/s11771-013-1520-0. URL https://doi.org/10.1007/s11771-013-1520-0.

[38] Highway Capacity Manual. Hcm2010. Transportation Research Board, National Research Council, Washington, DC, 2010.

[39] P. Mirchandani and Fei-Yue Wang. Rhodes to intelligent transportation systems. IEEE Intelligent Systems, 20(1):10–15, Jan 2005. ISSN 1541-1672. doi: 10.1109/MIS.2005.15.

[40] Prahlad D Pant, Yizong Cheng, Arudi Rajagopal, Nagaraju Kashayi, et al. Field testing and implementation of dilemma zone protection and signal coordination at closely-spaced high-speed intersections. Rep. No. FHWA/OH-2005/006, Ohio Dept. of Transportation, Columbus, OH, 2005.

[41] Markos Papageorgiou, Christina Diakaki, Vaya Dinopoulou, Apostolos Kotsialos, and Yibing Wang. Review of road traffic control strategies. Proceedings of the IEEE, 91 (12):2043–2067, 2003.

[42] Natural Point. Optitrack motive software documentation. https://v20.wiki.optitrack.com/index.php?title=OptiTrack_Documentation_Wiki, 2018. [Online; accessed 18-January-2019].

[43] Amanda Prorok, Nicholas Hyldmar, and Yijun He. A fleet of miniature cars for experiments in cooperative driving. 2019.

[44] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In ICRA workshop on open source software, volume 3, page 5. Kobe, Japan, 2009.

[45] DJ Quinn. A review of queue management strategies. Traffic Engineering+ Control, 33 (11):600–5, 1992.

[46] Mohsen Ramezani and Nikolas Geroliminis. Queue profile estimation in congested urban networks with probe data. Computer-Aided Civil and Infrastructure Engineering, 30 (6):414–432, 2015.

[47] Roger P Roess, Elena S Prassas, and William R McShane. Traffic engineering. Pearson/Prentice Hall, 2004.

[48] David Schrank, Bill Eisele, Tim Lomax, and Jim Bak. 2015 urban mobility scorecard. 2015.

[49] Chang-qiao Shao and Xiao-ming Liu. Estimation of saturation flow rates at signalized intersections. Discrete Dynamics in Nature and Society, 2012, 2012.

[50] A. Shu, Q. Fu, D. Liu, and L. Dai. Optimization of signal coordination for closely spaced intersections. In 2017 4th International Conference on Transportation Information and Safety (ICTIS), pages 6–11, Aug 2017. doi: 10.1109/ICTIS.2017.8047713.

[51] Arthur G Sims and Kenneth W Dobinson. The sydney coordinated adaptive traffic (scat) system philosophy and benefits. IEEE Transactions on vehicular technology, 29 (2):130–137, 1980.

[52] K. Tiaprasert, Y. Zhang, X. B. Wang, and X. Zeng. Queue length estimation using connected vehicle technology for adaptive signal control. IEEE Transactions on Intelligent Transportation Systems, 16(4):2129–2140, Aug 2015. ISSN 1524-9050. doi: 10.1109/TITS.2015.2401007.

[53] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. Physical review E, 62(2):1805, 2000.

[54] Rui Wang, Lei Zhang, Kejiang Xiao, Rongli Sun, and Li Cui. Easisee: Real-time vehicle classification and counting via low-cost collaborative sensing. IEEE Transactions on Intelligent Transportation Systems, 15(1):414–424, 2014.

[55] Chairit Wuthishuwong, Ansgar Traechtler, and Torsten Bruns. Safe trajectory planning for autonomous intersection management by using vehicle to infrastructure communication. EURASIP Journal on Wireless Communications and Networking, 2015 (1):33, Feb 2015. ISSN 1687-1499. doi: 10.1186/s13638-015-0243-3. URL https://doi.org/10.1186/s13638-015-0243-3.

[56] Xianfeng Yang, Yang Lu, and Gang-Len Chang. Dynamic signal priority control strategy to mitigate off-ramp queue spillback to freeway mainline segment. Transportation Research Record: Journal of the Transportation Research Board, (2438):1–11, 2014.

[57] Y. J. Zhang, A. A. Malikopoulos, and C. G. Cassandras. Optimal control and coordination of connected and automated vehicles at urban traffic intersections. In 2016 American Control Conference (ACC), pages 6227–6232, July 2016. doi: 10.1109/ACC.2016.7526648.

[58] B. Zheng, C. Lin, H. Liang, S. Shiraishi, W. Li, and Q. Zhu. Delay-aware design, analysis and verification of intelligent intersection management. In 2017 IEEE International Conference on Smart Computing (SMARTCOMP), pages 1–8, May 2017. doi: 10.1109/SMARTCOMP.2017.7946999.

[59] I. H. Zohdy, R. K. Kamalanathsharma, and H. Rakha. Intersection management for autonomous vehicles using icacc. In 2012 15th International IEEE Conference on Intelligent Transportation Systems, pages 1109–1114, Sep. 2012. doi: 10.1109/ITSC.2012.6338827.