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

Pratham Oza and Thidapat Chantem
Department of Electrical and Computer Engineering, Virginia Tech, USA. Email: {prathamo, tchantem} @vt.edu

***Abstract*—Safe and efficient traffic control remains a challenging task with the continued increase in the number of vehicles, especially in urban areas. This paper 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 paper 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. This work is also the first to provide worst-case bounds on wait time 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%.**

I. Introduction & Contributions

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 [1]. Congestion also negatively impacts the U.S. economy due to wasted time ($305 billion in 2017 [2]) and fuel costs (19 gallons of fuel per commuter is wasted solely due to idling in traffic per year [1]), as well as the environment (28% of CO_2 emissions in the U.S. come from the transportation sector [3]). 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 [4], 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 [5]. 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 [6]–[8] 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. In addition, urban, closely spaced intersections are often prone to queue *spillbacks* [9]. 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 obstruct the flow of emergency vehicles through the traffic, potentially impacting lives. Spillbacks can usually be observed when the separation between densely packed intersections is small ($<$170 m [10]) 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 avoiding spillbacks, reducing congestion, and improving 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, as will be shown in Section VI. While increased trip times may reduce the drivers' satisfaction, minimizing spillback has been shown to lead to fewer collisions and disruptions [11].

For autonomous vehicles, traffic control can be optimized by exploiting car-to-everything (C2X) connectivity, using virtual traffic lights [12] and/or by enforcing a set arrival pattern that avoids traffic deadlocks within an intersection [13]. These approaches have been shown to be effective at alleviating congestion and increasing fuel efficiency [14]. 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 [15], 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 wait times, and (ii) bound the delay associated with intersection crossings so that travel times can be accurately determined, which can help in precisely selecting the fastest routes and in trip planning. 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 queues, leveraging the traffic information at this and neighboring in-

tersections. Our approach aims to minimize queue spillbacks, and ensures 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 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.

II. Related Work

Recent work on intersection management has primarily focused on managing traffic flow for autonomous vehicles [16] or by assuming the vehicles are connected [17]. 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 [18]. A message request based connected vehicle traffic management is also presented [19], 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 [20], approaches that do not favor only the connected vehicles are needed. Our approach works for traditional, 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 [21] and queue estimation analysis [22], and then used to optimize signal timings. Existing solutions in this area are either fixed timing-based traffic systems, detector-based reactive traffic control systems, or adaptive traffic controllers [23]. 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 are not very adaptive in heavy traffic leading to resource starvation [24]. In addition, the sensors require calibrations and may malfunction [20], [25].

Adaptive traffic controllers used for conventional traffic [6], [7] 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 [25]. Field evaluations for such techniques also show that slightly inaccurate traffic data can result in significant performance drop in traffic management [26].

Similarly, prediction and learning based techniques using reinforcement learning and deep learning [27] 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 [28]. 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 [29], as will be shown in Sections VI and VII. We focus on minimizing spillbacks since they have been shown to have propagating effects on the network [30], [31]. In addition, existing works [16], [18], [27], [28] are best-effort approaches and do not provide worst-case delay guarantees.

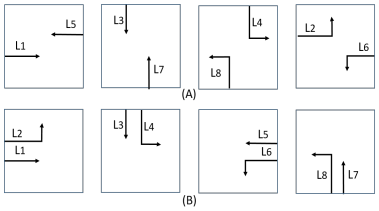


Fig. 1: Two possible phase sequences (A) and (B).

III. System Model and Assumptions

A. 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]. It is also assumed that vehicles in the system avoid performing lane changes. Hence, there are eight different traffic flow patterns inside the intersection, as shown in Figures 1 and 2. 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 lane L_2 are non-conflicting and do not interfere with lane L_1's flow. Thus, if lane L_1 is allowed access to the intersection, either lane L_2 or lane L_5 can simultaneously access the intersection as well. Considering this, Figure 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.

We assume that our system is managed by an *intersection manager* (IM), which aggregates information about the incoming traffic patterns from sensors and/or upstream intersections and forecast data, if any. The IM then calculates the required signal timing. In our approach, we only require (some) traffic information to be known and do not depend on how such data

[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.

may be acquired. This, however, implies that the intersection manager has minimal cloud connectivity.

B. 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 as 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 average length v_i of a 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

$$z_i = \frac{l_i}{v_i + s_i}. \tag{1}$$

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 [32].

An incoming lane L_i is characterized by a tuple $\{a_{i,j}, q_{i,j}, z_i\}$, where $a_{i,j}$ is the incoming vehicle flow rate during the j^{th} cycle, $q_{i,j}$ is the number of vehicles 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 lane capacity 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. Therefore, to avoid a spillback, this lane 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 (1), then, the spillback time $t_{sb_{i,j}}$ is*

$$t_{sb_{i,j}} = \frac{z_i - q_{i,j}}{a_{i,j}(t)}. \tag{2}$$

Proof: Based on the definitions, $a_{i,j}(t) \cdot t_{sb_{i,j}}$ vehicles will enter lane 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 paper. For rapidly changing flows, the worst case flow rate within T_c can be bounded and used in the analysis instead.

To determine the number of vehicles that can be dispatched in a given green time interval, we make use of *saturation headway* [33]. 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 $e_1, e_2 \ldots e_m$ as the queue moves forward where $e_1 \geq e_2 \geq \ldots \geq e_m$ since following vehicles react comparatively faster. After m vehicles, the time headway stabilizes to a value h and $e_{m+1} \ldots e_n = 0$ (usually after the 6^{th} vehicle [34]). 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,

$$T_{i,j} = h \times n_{i,j} + t_l. \tag{3}$$

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 are defined in Highway Capacity Manual (HCM) as 4 s and 2 s, respectively [35].

IV. Problem Statement

Let us assume an intersection with incoming traffic from four different directions, with a total of eight lanes, where access to the intersection is dictated by a traffic light for each direction of flow and according to the phase sequences. 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.

V. A Real-Time Task Scheduling Problem

The problem of traffic control at an intersection (Section IV) can 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

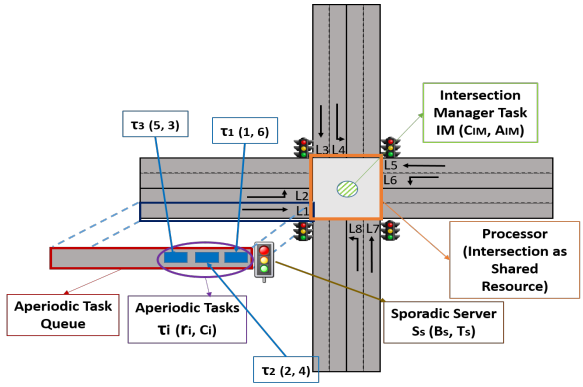


Fig. 2: An intersection as a real-time task scheduling problem.

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.

We model an intersection shown in Figure 2 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 servers are non-preemptive.

As explained in Section III, 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 [36]. As shown in Figure 2, 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 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.

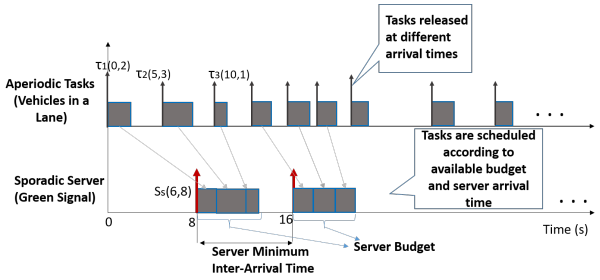


Fig. 3: Example sporadic server serving aperiodic tasks.

A. *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 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 an intersection. To serve these aperiodic tasks, we leverage the concept of sporadic servers [37]. In an environment with mixed real-time tasks, servers are often used to schedule aperiodic requests such that their response times are minimized while ensuring that the hard real-time tasks meet their deadlines. [38]. 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 inter-arrival time T_S as well as a budget B_S. Since there are eight lanes, and hence, eight aperiodic task queues, we 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. Note that our system can be trivially extended to situations where more than two lanes have non-conflicting flows, e.g., two lanes for left turns and one lane for going straight. In such cases, all the servers serving the non-conflicting lanes can be grouped to access the intersection at once. We will present an approach to assign the budget and the arrival time of these servers in Section V-C.

Figure 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 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 inter-arrival time has been reached, i.e., if an incoming lane is empty, the sporadic server is not activated again until a vehicle arrives. In Figure 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 1, respectively. As these tasks

TABLE I: Green time calculations using server based approach with cycle time of 90 sec

Phase	Lane	Cap (veh)	Queue (veh)	Flow 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		

arrive, they join the aperiodic task queue. When the server $S_S(6, 8)$, with a minimum inter-arrival 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 inter-arrival time is 8.

B. 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 pre-emptions can occur to ensure safety, since it is not possible to "preempt" a vehicle after it has entered an 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 Section III-B. It also sets its own next arrival time, depending on the cycle length obtained from the calculated budget (Section V-C). 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 inter-arrival times for all sporadic tasks in the system, i.e., the sporadic servers and the IM task are set to $t_{min} = h_1 + t_{cl}$. This is the minimum time required for the lead vehicle in a queue to cross an intersection as it will take h_1 time due to the reaction delay to the green light and take t_{cl} time to clear the intersection (as explained in Section III.) The time complexity of our signal timing calculation algorithm that the IM task performs is $\mathcal{O}(n)$. Since IM is a lightweight real-time task, it can easily be deployed using currently available traffic controllers that are capable of running a minimal real-time OS [39].

C. 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 [40]. That is, we determine the smallest spillback time among all lanes according to Equation (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 I 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 I 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., $T_c = 90$.

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 beginning of the next cycle and enforce 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 I, 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.

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

$$U_{i,j_{min}} = \frac{h \times (a_{i,j} \cdot T_c + q_{i,j} - z_i + 1) + t_l}{T_c}. \quad (4)$$

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,

$$U_{i,j_{max}} = \frac{h \times (a_{i,j} \cdot T_c + q_{i,j}) + t_l}{T_c}. \quad (5)$$

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 required 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.

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.

D. Correctness and Worst Case Delay Analysis

Once T_c is calculated (Section V-C), 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 utilization $\leq 100\%$), 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*

$$n_{out_{i,j}} \geq a_{i,j} \cdot T_c + q_{i,j} - z_i + 1. \tag{6}$$

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,

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

The integer 1 is added, as the number of vehicles must be an integer. ■

Theorem 1. *Assuming that the flow rate is equal to $a_{i,j}$, $i = 1, \ldots, n$ 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*

$$W_{k,i,j} \in \left[0, \sum_{j=1}^{\left\lfloor \frac{k}{n_{out_{i,j}}} \right\rfloor} (T_c - U_{i,j_{min}} \cdot T_c) \right). \tag{7}$$

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 (6). 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), and

$$U_{min_{i,j}} = \frac{h \times (a_{i,j} T_c + q_{i,j} - z_i + 1) + t_l}{T_c}. \tag{8}$$

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 bounded by

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

■

While the above worst-case delay analysis is based on the average length of the vehicles, a more accurate delay value can be obtained using information acquired on-the-fly from road-side detectors [32] (at the expense of an increase in time complexity associated with delay analysis), or the maximum vehicle length can be used instead of the average length for a more pessimistic but conservative estimate.

VI. Simulations

We compare our approach against (i) a widely deployed pre-timed traffic control technique [41], and (ii) an adaptive control technique based on real-time phase saturability [28]. 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 are adjusted according to the incoming flow from different phases by ensuring that the lanes with higher flow rate get a longer green time. More sophisticated techniques [6], [7], [27] consider network-wide optimization, which suffer from scalability issues as discussed earlier in Section II.

A. Simulation Setup

We used a tick-based simulator, written in Python, to simulate the desired vehicle flow (i.e., where the vehicles follow the arrival/departure patterns and safe distances as per the traffic control algorithms and the restrictions defined by the HCM [35]). A detailed description of both our simulation and hardware setups is available online [42]. In these simulations, 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 [43]. 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.

TABLE II: 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

B. Comparison with Pre-Timed Traffic Control Technique

Table II 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 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 II 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 because of which more vehicles are accumulated leading to larger queues, cycle times, 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 II). In contrast, our approach does not experience spillback and in fact reduces the average delays, except for the average case with flow rate of 7 veh/min. Even then, our approach does not perform worse than the existing approaches.

Figures 4 and 5(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 4 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(a) and (b) show the delays experienced by the vehicles with dashed line indicating queue spillbacks occurring in the case of average vehicle flow. It should also be noted that in most cases, the worst case flow results in lower delays compared to the average case flow, as long as the number of incoming vehicles is under the lane capacity. This is due to the fact that the worst case flow during a cycle (where not all vehicles can cross and thus must wait until the next cycle) becomes the best case flow for the next cycle (where the vehicles are waiting and

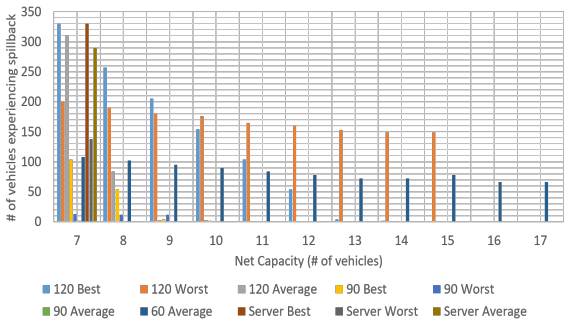
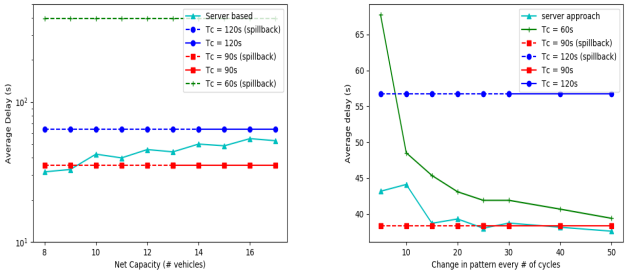


Fig. 4: Number of vehicles experiencing spillback under varying lane capacity with unbalanced flow rates.

ready to cross the intersection when the light turns green.

In addition, as seen in Figure 5(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.



(a) Constant unbalanced vehicle flow. (b) Frequently changing unbalanced vehicle flow.

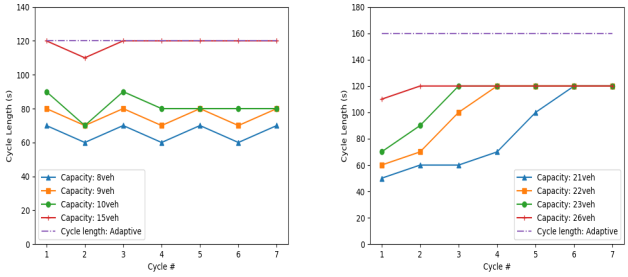
Fig. 5: Average delays incurred by vehicles with server based approach vs. pre-timed adaptive control.

Finally, to test the adaptability of our approach to 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(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, except with very frequent traffic surges (every 5 and 10 cycles).

C. Comparison with Adaptive Elastic Traffic Signal Control

Han et al. [28] 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" 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 III, 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 [28] 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, which was also observed in case of pre-timed control (Table II). The entries in Table III highlighted in red indicate that queue spillbacks were observed in that scenario. Table III shows that in case of the adaptive approach, queue spillbacks are observed with capacities less than 14 veh for medium traffic flow and 29 veh for 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 accommodate 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.



(a) Medium flow (b) Heavy flow

Fig. 6: Average delays incurred by vehicles with server based approach vs. pre-timed adaptive control.

Finally, Figure 6 shows 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

TABLE III: Average delay(s) experienced by vehicles using server based vs. adaptive elastic scheduling based approaches [28]

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

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 6 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 [28] 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 is slightly higher than the existing approaches, but none of the queues spillback. Queue spillbacks are known to cause disruptive effects on a transportation 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, it does not adversely affect the entire transportation network.

VII. Experimental Validation

This section 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. Due to space constraints, we only 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.

A. Setup

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 24 IR cameras and the Motive software that captures the vehicle positions. This position data is then streamed to a command computer where an interface application utilizing the Robot Operating System (ROS) [44] framework makes the gathered positions for each vehicle available to our controller application. This application processes the position data and sends control commands (left wheel and right wheel velocity) accordingly to each individual vehicle.

1) Controlling multiple vehicles: The controller application implemented on ROS works in the following manner:

- The raw position data from the software is processed using a 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 7) 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* [45] 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) [46].
- 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 [46].
- 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.

2) Emulating 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

Fig. 7: Hardware setup emulating an urban intersection.

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 has been used to emulate and study the behavior of vehicles in a realistic environment [47]. The delays pertaining to the traffic flow (t_l and h, explained in Section III) are incorporated in the controller application and the vehicles are commanded accordingly. The real-time server timing parameters are calculated in accordance to the ROS' timing framework. Figure 7 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 red). If the vehicles are joining an existing queue, they stop at a safe distance governed by the IDM parameters. Once the light turns green, the vehicles are commanded to accelerate as per the reaction delay parameters. A video of our experiment can be found online [48].

B. Results

Due to lack of space, we only show the results 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. Nevertheless, the data presented here are generally representative. Figures 8 and 9 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) 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.

Figure 8 shows 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 9 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

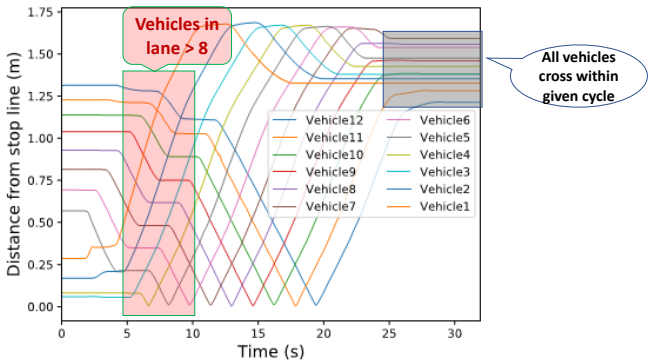


Fig. 8: Distance from stop line as a function of time for vehicles in lane L_3 using the elastic scheduling technique and medium flow.

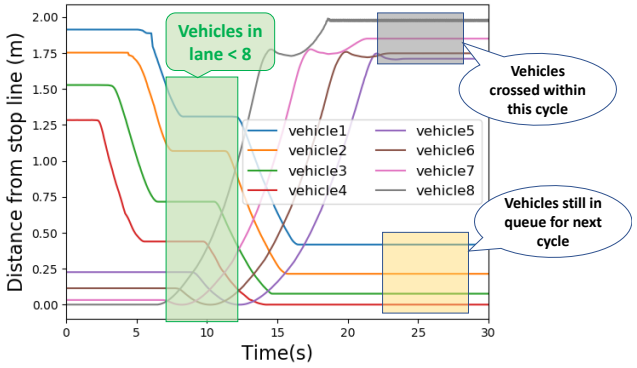


Fig. 9: Distance from stop line as a function of time for vehicles in lane L_3 using our server based approach.

four vehicles were not able to clear the intersection in this cycle, spillbacks do not occur.

VIII. Conclusions

In this paper, 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. In our future work, we plan on deriving how to best allocate the leftover budgets and consider more than one intersection.

IX. Acknowledgments

This work was supported in part by the National Science Foundation under grant number CPS-1618979.

References

[1] D. Schrank, B. Eisele, T. Lomax, and J. Bak, "2015 urban mobility scorecard," 2015.

[2] G. Cookson and B. Pishue, "Global traffic scorecard," Technical report, INRIX, Tech. Rep., 2017.

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

[4] T. F. Golob and W. W. Recker, "Relationships among urban freeway accidents, traffic flow, weather, and lighting conditions," *Journal of transportation engineering*, vol. 129, no. 4, pp. 342–353, 2003.

[5] D. Quinn, "A review of queue management strategies," *Traffic Engineering+ Control*, vol. 33, no. 11, pp. 600–5, 1992.

[6] A. G. Sims and K. W. Dobinson, "The Sydney coordinated adaptive traffic (scat) system philosophy and benefits," *IEEE Transactions on vehicular technology*, vol. 29, no. 2, pp. 130–137, 1980.

[7] P. Hunt, D. Robertson, R. Bretherton, and R. Winton, "Scoot-a traffic responsive method of coordinating signals," Tech. Rep., 1981.

[8] J.-J. Henry, J. L. Farges, and J. Tuffal, "The prodyn real time traffic algorithm," in *Control in Transportation Systems*. Elsevier, 1984, pp. 305–310.

[9] P. D. Pant, Y. Cheng, A. Rajagopal, N. 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.

[10] 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)*, Aug 2017, pp. 6–11.

[11] J. Kwon and P. Varaiya, "The congestion pie: delay from collisions, potential ramp metering gain, and excess demand," in *Proceedings of 84th Transportation Research Board Annual Meeting*, 2005.

[12] 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)*, July 2016, pp. 6227–6232.

[13] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Ballroom intersection protocol: synchronous autonomous driving at intersections," in *2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug 2015, pp. 167–175.

[14] 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*, Sep. 2012, pp. 1109–1114.

[15] P. Mirchandani and F.-Y. Wang, "Rhodes to intelligent transportation systems," *IEEE Intelligent Systems*, vol. 20, no. 1, pp. 10–15, Jan 2005.

[16] D. Carlino, S. D. Boyles, and P. Stone, "Auction-based autonomous intersection management," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, Oct 2013, pp. 529–534.

[17] C. Wuthishuwong, A. Traechtler, and T. Bruns, "Safe trajectory planning for autonomous intersection management by using vehicle to infrastructure communication," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, p. 33, Feb 2015. [Online]. Available: https://doi.org/10.1186/s13638-015-0243-3

[18] K. Dresner and P. 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*, ser. AAMAS '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 530–537. [Online]. Available: https://doi.org/10.1109/AAMAS.2004.190

[19] 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)*, May 2017, pp. 1–8.

[20] Y. Feng, K. L. Head, S. Khoshmagham, and M. Zamanipour, "A real-time adaptive signal control in a connected vehicle environment," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 460–473, 2015.

[21] M. Collotta, L. L. Bello, and G. Pau, "A novel approach for dynamic traffic lights management based on wireless sensor networks and multiple fuzzy logic controllers," *Expert Systems with Applications*, vol. 42, no. 13, pp. 5403–5415, 2015.

[22] 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*, vol. 16, no. 4, pp. 2129–2140, Aug 2015.

[23] L. Li, D. Wen, and D. Yao, "A survey of traffic control with vehicular communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 425–432, Feb 2014.

[24] N. A. of City Transportation Officials, "Urban bikeway design guide," *New York*, vol. 8, 2011.

[25] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.

[26] C. Kergaye, A. Stevanovic, and P. 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] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[28] S.-Y. Han, F. Ping, Q. Zhang, Y.-H. Chen, J. Zhou, and D. Wang, "Global adaptive and local scheduling control for smart isolated intersection based on real-time phase saturability," in *Intelligent Computing Theories and Application*, D.-S. Huang, K.-H. Jo, and J. C. Figueroa-García, Eds. Cham: Springer International Publishing, 2017, pp. 730–739.

[29] M. Ramezani and N. Geroliminis, "Queue profile estimation in congested urban networks with probe data," *Computer-Aided Civil and Infrastructure Engineering*, vol. 30, no. 6, pp. 414–432, 2015.

[30] X. Yang, Y. Lu, and G.-L. 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*, no. 2438, pp. 1–11, 2014.

[31] D.-f. Ma, D.-h. Wang, F. Sun, Y.-m. Bie, and S. Jin, "Method of spillover identification in urban street networks using loop detector outputs," *Journal of Central South University*, vol. 20, no. 2, pp. 572–578, Feb 2013. [Online]. Available: https://doi.org/10.1007/s11771-013-1520-0

[32] R. Wang, L. Zhang, K. Xiao, R. Sun, and L. Cui, "Easisee: Real-time vehicle classification and counting via low-cost collaborative sensing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 1, pp. 414–424, 2014.

[33] C.-q. Shao and X.-m. Liu, "Estimation of saturation flow rates at signalized intersections," *Discrete Dynamics in Nature and Society*, vol. 2012, 2012.

[34] P. Koonce *et al.*, "Traffic signal timing manual," United States. Federal Highway Administration, Tech. Rep., 2008.

[35] H. C. Manual, "Highway capacity manual, 2010," *Transportation Research Board, National Research Council, Washington, DC*, 2010.

[36] Y. Chen, "Model of traffic speed-flow relationship at signal intersections," *Open Journal of Applied Sciences*, vol. 7, no. 06, p. 319, 2017.

[37] D. Faggioli, M. Bertogna, and F. Checconi, "Sporadic server revisited," in *Proceedings of the 2010 ACM Symposium on Applied Computing*. ACM, 2010, pp. 340–345.

[38] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer Science & Business Media, 2011, vol. 24.

[39] M. 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].

[40] R. P. Roess, E. S. Prassas, and W. R. McShane, *Traffic engineering*. Pearson/Prentice Hall, 2004.

[41] F. Dion, H. Rakha, and Y.-S. Kang, "Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections," *Transportation Research Part B: Methodological*, vol. 38, no. 2, pp. 99–122, 2004.

[42] P. Oza, "A real-time server based approach for safe and timely intersection crossings," Master's thesis, Virginia Tech, 2019.

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

[44] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[45] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[46] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[47] A. Prorok, N. Hyldmar, and Y. He, "A fleet of miniature cars for experiments in cooperative driving," 2019.

[48] Anonymous, "Server based traffic control experiments," June 2018. [Online]. Available: https://www.youtube.com/channel/UCI-UGJKT7C5E_8bs391LCpA