R^2IM – Robust and Resilient Intersection Management of Connected Autonomous Vehicles

Mohammad Khayatian, Rachel Dedinsky, Sarthake Choudhary, Mohammadreza Mehrabian
and Aviral Shrivastava
Arizona State University

***Abstract*— Intersection management of Connected Autonomous Vehicles (CAVs) has the potential to significantly improve safety and mobility. While numerous intersection management designs have been proposed in the past few decades, most of them assume that the CAVs will precisely follow the directions of the Intersection Manager (IM) and prove the safety and demonstrate the efficiency based on this assumption. In real life, however, a CAV that is crossing the intersection may break down, accelerate out-of-control or lie about its information (e.g. intended outgoing lane) and cause an accident. In this paper, we first define a fault model called "rogue vehicle", which is essentially a CAV that either is dishonest or does not follow the IM's directions and then, propose a novel management algorithm (R^2IM) that will ensure safe operation, even if a CAV becomes "rogue" at any point in time. We prove that there can be no accidents inside the intersection, as long as there is no more than one "rogue vehicle" at a time. We demonstrate the safety of R^2IM by performing experiments on 1/10 scale model CAVs and in simulation. We also show that our approach can recover after the rogue vehicle leaves/is removed.**

I. Introduction

According to the American Automobile Association (AAA), more than two people are killed every day in the U.S. due to accidents caused by red lights runners [1]. When vehicles become autonomous and connected, however, the number of accidents caused by human errors can dramatically be reduced. In the past few decades, many works [2], [3] have been devoted to developing algorithms for safe and efficient management of connected autonomous vehicles (CAVs) at road intersections, in which CAVs do not have to come to a complete stop at the intersection, rather, they can just slow down a bit to pass each other safely.

Most broadly, existing algorithms for intersection management of CAVs [4], [5] can be classified into two categories: centralized and distributed. In centralized techniques, CAVs interact with an intersection manager (IM), which schedules the arrival time of CAVs while in distributed approaches, CAVs interact with each other to determine how to cross the intersection. Because of security concerns, centralized intersection management approaches are sometimes preferred. This is because the CAVs only have to communicate with the IM –which is a part of the infrastructure– rather than communicating with other CAVs. On the other hand, centralized approaches require support from the infrastructure and therefore, their deployment at all intersections may not be feasible.

Although various intersection management algorithms have been proposed, almost all of them make strong assumptions to ensure safety. First, it is assumed that a CAV shares correct and accurate information with the IM, while in reality; a CAV may unintentionally or deliberately send wrong information. For example, a CAV may send a wrong position, velocity, or even intended outgoing lane. Secondly, it is assumed that a CAV behaves as the IM expects and follows the assigned trajectory. However, a CAV may break down and suddenly stop in the middle of the intersection or accelerate out-of-control and enter the intersection earlier than expected which in turn can result in an accident. We believe that real-life intersection management techniques need to account for such failures and use a mechanism to avoid accidents if such scenarios happen.

This paper then presents R^2IM – a robust intersection management algorithm that is resilient against a rogue vehicle that may be present. Our approach uses an external surveillance system that can detect if a vehicle is not following the expected trajectory beyond a tolerance limit, and declares it as rogue. The safety is achieved essentially by scheduling the cross-time of approaching CAVs with enough temporal buffer, so that an accident becomes impossible, even if a CAV becomes "rogue" at any point of time. To evaluate the correctness of R^2IM, we built a 1/10 scale model intersection and injected faults on vehicles by forcing them to accelerate/decelerate and showcased the safety of the CAVs.

II. Related Works

In the past few decades, intersection management of CAVs has extensively been studied [4]. Many researchers have focused on distributed management techniques where CAVs communicate with each other and decide who should cross first [6]–[11]. On the contrary, other studies were focused on centralized intersection management approaches where CAVs communicate with the infrastructure to get a reservation. Some centralized techniques follow a query-based intersection management scheme where CAVs query safe crossing from the IM and the IM either accepts or rejects the request [12]–[15] while other centralized approaches follow an assignment-based management technique where CAVs share their information with the IM and the IM assigns a reservation to them [2], [16]–[19].

Although many research studies were focused on improving the throughput of the intersection, not much research is done on improving the safety and robustness of the

intersection management technique. Responsibility-Sensitive Safety rules are proposed to ensure the safety of CAVs in different scenarios [20] but they do not explicitly support intersection scenarios.

To tackle uncertainty issues for the intersection management problem, researchers have proposed methods to mitigate faults that can happen in the intersection management system. In Crossroads technique [18], [19], authors highlighted the need for having the same notion of time between IM and CAVs and safety issues that arises due to communication delay and IM's processing time. In RIM [16], authors showed that the existence of model mismatch and external disturbances can cause an error in the eventual arrival time of CAVs. In [21], [22], Bentjen *et al.* considered a scenario where a malicious CAV blocks the intersection. They have presented some initial thoughts on how to mitigate such vulnerabilities. Dedinsky *et al.* [23] has provided some initial thoughts on how to employ a surveillance system to detect rogue vehicles at the intersection. In almost all previous works, it was assumed that all CAVs share correct information with the IM and precisely follow IM's command.

In the next section, we first explain the interaction between IM and CAV when no rogue vehicle is present and then define the fault model for the rogue vehicle and discuss how R^{2}IM accommodates a rogue vehicle.

III. R^{2}IM WITHOUT ROGUE VEHICLES

When a CAV is within the communication range of the intersection, it synchronizes its internal clock with the IM and then sends a request to the IM by sharing its position, velocity, and corresponding timestamp as well as CAV's ID and the intended outgoing lane. Accordingly, the IM calculates a safe Time of Arrival (TOA) and Velocity of Arrival (VOA) and sends it back to the CAV. Upon receiving the VOA and TOA, the CAV determines an optimal reference trajectory and lets the IM know by sharing the trajectory parameters, A_0 and B_0 (explained later). Next, the IM adds CAV's information to its list of "active CAVs" and sends an acknowledgment (ACK) to the CAV. After receiving the ACK from the IM, the CAV follows its reference trajectory until it reaches the intersection where it continues at the constant velocity of VOA. If a CAV fails to synchronize its clock or at any stage, does not receive a response from the IM within the set timeout, it will apply break and starts over by synchronizing its clock as its clock may be out of sync. IM and CAV's algorithms are presented in Alg. 1 and Alg. 2.

Algorithm 1: Algorithm for CAVs

```
1:  while True do
2:    if (within the intersection range) then
3:      synchronize the clock
4:      if sync is not successful or timed out then
5:        apply brake and goto line 3
6:      end if
7:      send a request to IM
8:      receive the TOA and VOA from IM
9:      if response is timed out then
10:       apply brake and goto line 3
11:     end if
12:     calculate reference trajectory
13:     send trajectory information to the IM
14:     receive the ACK from the IM
15:     if ACK is timed out then
16:       apply brake and goto line 3
17:     end if
18:     inquiry emergency state from IM
19:     if emergency state is active or timed out then
20:       if (After point of no return (PONR)) then
21:         follow reference trajectory and goto line 18
22:       else
23:         apply brake and goto line 3
24:       end if
25:     else
26:       if (if entered the intersection) then
27:         drive at a constant velocity (VOA)
28:       else
29:         follow reference trajectory goto line 18
30:       end if
31:     end if
32:   end if
33: end while
```

A. Reference Trajectory Calculation and Tracking

When a CAV receives the VOA and TOA values from the IM, it needs to make a plan to arrive at the intersection at time TOA with speed VOA. The plan is essentially a position-vs-time graph that specifies where the vehicle should be at any point in time. For simplicity, we consider a double integrator model for the behavior of the CAVs before entering the intersection:

$$\begin{cases} \dot{p} = v \\ \dot{v} = a \end{cases} \tag{1}$$

where p is the longitudinal position of the vehicle, v is the velocity and a is the input acceleration. Since acceleration and deceleration rates of a CAV are bounded in real life, we consider limits for the acceleration as $a \in [a_{min}, a_{max}]$, where a_{max} and a_{min} are the maximum acceleration and deceleration rates of the CAV. Similarly, we consider an upper bound and a lower bound for the velocity of the CAV as $v \in [v_{min}, v_{max}]$, where v_{max} is the maximum velocity of the vehicle and is the same as speed limit and v_{min} is the minimum velocity of the vehicle. We determine the reference trajectory by minimizing the total amount of acceleration/deceleration for each CAV, which is linear:

$$a_r = A_0 t + B_0 \tag{2}$$

where A_0 and B_0 are constants that can be determined from initial and final conditions similar to [16]. Each CAV utilizes a PID (Proportional-Integral-Derivative) controller to track

Algorithm 2: Algorithm for the Intersection Manager

```
while True do
    check emergency state
    broadcast emergency state
    if a request is received then
        if emergency state is active then
            reject the request
        else
            calculate the optimal TOA and VOA
            send TOA and VOA to the vehicle
        end if
    end if
    if trajectory information is received then
        store the CAV's trajectory information
        send the ACK message
    end if
end while
```

the reference position trajectory:

$$a = k_P e + k_I e_I + k_D e_D \tag{3}$$

where a is the control input (acceleration), e is the position error defined as $e = p_r - p$, e_I is the integral of the error ($e_I = \int e$), e_D is the derivative of e ($e_D = v_r - v$) and k_P, k_I, and k_D are positive constants which are referred to as PID gains.

IV. R^{2}IM with Rouge vehicles

In this section, we first present the fault model and then show the interaction of IM with CAVs to handle rogue vehicles.

A. Fault Model – Rouge Vehicle

The Rogue vehicle is a CAV that intentionally or unintentionally is lying when sharing its information to the IM, or does not follow IM's directions. The rogue CAV may either accelerate or decelerates but it never drives outside the road boundary. To generalized the rogue vehicle's definition, we use the following definition:

Definition 1: A CAV is deemed rogue if it deviates from its expected position by a pre-set threshold.

This fault model covers many scenarios including following extreme cases:

Acceleration (ACC) Fault Scenario: The rogue vehicle suddenly accelerates with $a = a_{max}$ toward the intersection and enters the intersection earlier than it was scheduled.

Deceleration (DEC) Fault Scenario: The rogue vehicle breaks down and suddenly stops $a = -\infty$ inside the intersection.

Lying about outgoing Lane: The rogue vehicle lies about its outgoing lane and takes another path once it enters the intersection. Next, we define some of the terms that we use in the algorithm.

Definition 2: **Point of No Return (PONR)** is the farthest point from the intersection that if after passing this point a CAV starts applying full brake ($a = a_{min}$), it cannot fully stop without entering the intersection.

The distance of the PONR from the edge of the intersection (POA) is d_{PONR} and can be calculated as:

$$d_{PONR} = \frac{v_{PONR}^2}{2|a_{min}|} + \frac{VL}{2} \tag{4}$$

VL is vehicle length and v_{PONR} is the velocity of the CAV at the PONR. Since p represents the longitudinal location of the center of a CAV, $\frac{VL}{2}$ is added to account for the length of the CAV.

Definition 3: **Critical zone** for a CAV is defined as the area between its PONR and the point it exits the intersection.

Definition 4: **Critical time window** is the time it takes for a CAV to travel through the critical zone.

The critical window (Δt_{crit}) can be calculated as the summation of time to reach the intersection and time to travel inside the intersection:

$$\Delta t_{crit} = (TOA - t_{PONR}) + \frac{d_I + 0.5VL}{VOA} \tag{5}$$

where d_I is the traveled distance inside the intersection and can be determined from the dimensions of the intersection. For left and right turns, the d_I is $\frac{3\pi LW}{2}$ and $\frac{\pi LW}{2}$ respectively and for going straight d_I is $2LW$ where LW is the lane width.

Definition 5: **Safety Barrier (SB)** is the maximum distance that a CAV may travel when the previously scheduled CAV is in its critical zone.

Since the velocity of a CAV is bounded by v_{max}, the maximum distance that a CAV can travel corresponds to the case where its initial velocity is equal to v_{max}. As a result, the size of the safety barrier is:

$$d_{SB} = \Delta t_{crit} v_{max} \tag{6}$$

For a practical design, the IM should account for the worst-case execution time of the IM (C_{IM}) and CAVs (C_{CAV}), and the period of emergency inquiry by a CAV (T) to ensure safety. As a result, Eq. (4) and (6) is modified as follows to account for them:

$$d_{PONR} = \frac{v_{PONR}^2}{2|a_{min}|} + \frac{VL}{2} + \rho v_{max} \tag{7}$$

$$d_{SB} = (\Delta t_{crit} + \rho) v_{max} \tag{8}$$

where $\rho = T + C_{IM} + C_{CAV}$ is the worst-case end-to-end delay from the moment a CAV becomes rogue to the moment other CAVs are notified and react.

B. IM and CAV Interaction In Presence of A Rogue Vehicle

The IM periodically calculates the distance between the estimated position of CAVs – which is determined from a CNN-based perception system like [24] – and their expected position. If the distance is greater than a threshold, e_{th}, it set the emergency state to active. Whether the emergency state is active or not, the IM periodically broadcasts it. When a CAV notifies that the emergency state is active, it checks if its position is before its *Point of No Return* (PONR) (Eq. (4)) and can safely stop without entering the intersection. If the CAV is after its PONR, it ignores the emergency state and

continues with its trajectory. If a CAV does not receive the emergency state within the set timeout, it assumes that the emergency state is active and applies the brake. During the emergency state, the IM rejects all the requests that are received from CAVs. When the emergency is resolved the IM sets the emergency state to false and starts processing the requests that it receives.

The IM adopts a First-Come First-Served (FCFS) policy for scheduling. Given the request time, position, and velocity of the requesting CAV, and expected trajectories of other CAVs are known, the IM determines a VOA and TOA pair for the requesting CAV such that the earliest arrival time is achieved. To do so, the IM solves the following optimization problem:

$$\min TOA \tag{9}$$

subjected to following constraint:

$$\begin{cases} p_i(t_{PONR,i-1}) < p_{SB,i} \\ p_i(t_{exit,i-1}) < p_{PONR,i} + \rho v_{max} \\ p_i(t) > p_{i,front}(t) + d_{PONR,i} \qquad \forall i > 1 \\ a_{min} < a_i(t) < a_{max} \\ v_{min} < v_i(t) < v_{max} \\ VOA < v_{turn} \end{cases} \tag{10}$$

where $p_i(t_{PONR,i-1})$ is the position of the requesting CAV (i) when the last scheduled CAV $(i-1)$ is at its PONR and $p_{SB,i}$ is the safety barrier point that is $d_{SB,i}$ meters away from the intersection. The first constraint ensures that the requesting CAV (i) is far enough from the intersection when the last scheduled CAV $(i-1)$ is at its PONR and does not cause a conflict for the last scheduled CAV. $p_i(t_{exit,i-1})$ is the position of the CAV (i) when the last scheduled CAV $(i-1)$ leaves the intersection and $p_{PONR,i}$ is the position the PONR for the requesting CAV. The second constraint ensures that the requesting CAV has enough time to stop if the last scheduled CAV slows down and stops inside the intersection. $p_{i,front}(t)$ is the position of the last scheduled CAV in the same lane as the requesting CAV or simply CAV i's front CAV if any. The third constraint ensures the requesting CAV's trajectory has always a minimum safe distance from its front CAV. Fourth and fifth constraints are considered to ensure that the velocity and acceleration of the requesting CAV are within the feasible range. Finally, v_{turn} is the maximum safe velocity for making a turn to avoid rollover. For driving straight, $v_{turn} = v_{max}$. To ensure that constraints in Eq.(10) are met, the IM reconstructs the trajectory of the last scheduled CAV $(i-1)$ and the CAV's front CAV (if any) from the request information (t_0, v_0, p_0), schedule information (TOA, VOA, POA) and trajectory information (A_0, B_0).

V. Safety Proof

We assume that only one rogue vehicle is present at a time and prove that no accident will happen inside the intersection area. We are limiting our proof to provide safety for the intersection area only because a rogue vehicle may accelerates or steer to the opposite lane and hits another vehicle and in such cases, an accident may be unavoidable. We will show that the rogue vehicle cannot get involved in an accident with the last scheduled CAV or the next scheduled CAV (if any). For a better intuition, we have depicted these two corner cases in Figure 1.

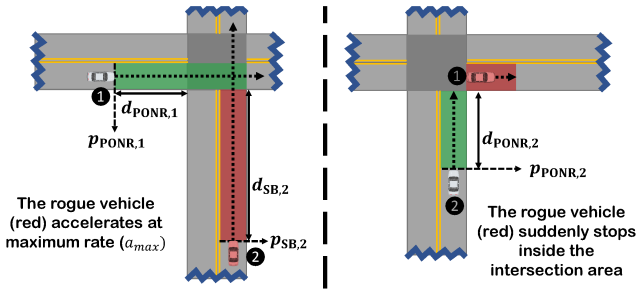

Fig. 1. Two corner cases where a CAV becomes rogue.

A. Interaction of Rogue Vehicle with Last Scheduled CAV

Let us assume that vehicle i becomes rogue and accelerates at time t_{rogue}. If the distance of the rogue vehicle from the intersection when it becomes rogue is more than d_{SB}, the earliest time that it can enter the intersection is after $t_{rogue} + \Delta t_{crit} + \rho$. Since the last scheduled CAV will reach and leave the intersection by Δt_{rogue}, the rogue vehicle enters the intersection when the last scheduled CAV has already left the intersection. If the distance of the rogue vehicle from the intersection when it becomes rogue is less than d_{SB}, the last scheduled CAV will be behind its point of no return according to Eq. (10). Therefore the last scheduled CAV can stop safely without entering the intersection.

B. Interaction of Rogue Vehicle with Next Scheduled CAV

If the rogue vehicle breaks down and suddenly stops before exiting the intersection, the next scheduled CAV will have at least $d_{PONR} + \rho v_{max}$ distance from the intersection (see Eq.(10)). As a result, even if it takes ρ milliseconds to notify the next scheduled CAV, it will have enough distance to stop without entering the intersection.

If a CAV lies about its outgoing lane and suddenly changes its direction inside the intersection, (e.g. makes a left turn instead of going straight), the next scheduled CAV will be beyond its PONR (see Eq. (10)) and therefore has enough distance to stop without entering the intersection similar to the deceleration case.

VI. Testbed and Results

We performed two types of experiments to validate the safety of our approach when a rouge vehicle is present: 1) systematically injecting a fault on a CAV and 2) randomly injecting a fault.

A. Safety Validation by Systematic Fault Injection

Our experimental testbed is an intersection with 1/10 scale model CAVs that are 57 cm long and 30 cm wide. An ESP8266 NodeMCU v3 board is utilized to enable wireless

communication of CAVs with the IM and perform real-time motion control by adjusting the steering angle and speed for the DC motors that run the CAV. A set of markers is installed on each vehicle and the OptiTrack monitoring system is used to track CAVs' positions. Figure 2 shows an overview of the intersection. The pose data and the emergency state packet

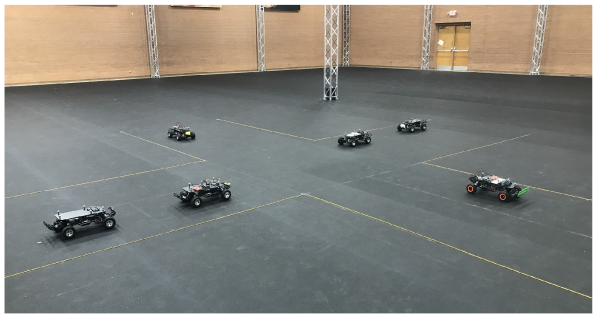

Fig. 2. Our testbed is a single-lane intersection with 1/10 scale model CAVs. CAVs position is tracked by the OptiTrack system.

is broadcast every 20 ms. Timing constraints of each AV are also monitored at the runtime [25]. CAVs start from arbitrary positions and track a set of way-points to drive within the lane and cross the intersection. The maximum velocity of CAVs is $3m/s$, and the maximum acceleration and deceleration rates are measured as $2m/s^2$ and $-1.5m/s^2$ respectively. We set the minimum velocity to be $0.2m/s$.

For the systematic fault injection, we created a scenario where two CAVs are scheduled to cross the intersection and the CAV that is scheduled to cross second becomes rogue after some time and accelerates. We created the same scenario and repeated our fault injection but at different times. Using the brute-force approach, we injected a set of ACC fault on the secondly scheduled CAV at every 0.1s where the fault injection time varies from 34.5 to 36. Figure 3 shows the distance between CAVs and the unsafe area. When

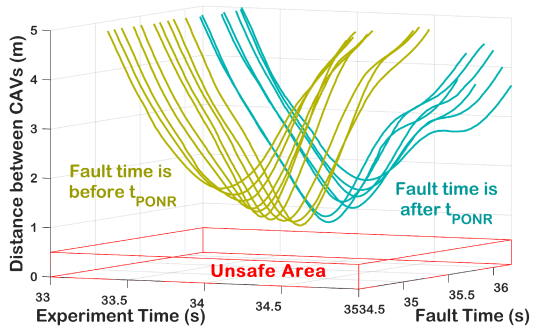

Fig. 3. Systematic fault injection of an ACC fault on our 1/10 scale model intersection. By increasing the fault time, the distance between CAVs decreases until the t_{PONR} and then it increases.

the fault is injected before the t_{PONR}, the other CAV stops but when the fault injection time is after the t_{PONR}, the other vehicle continues. It can be observed that for all experiments, the distance between vehicles is always greater than 0.5m.

To check the scalability of our approach for a real-size intersection with variable traffic patterns and also do fault injection more precisely, we built a simulator in Matlab and performed the same fault injection experiment. The length of CAVs is 5m and their width is 2m, the maximum velocity of CAVs is set to $20.1m/s$ (45mph), and maximum acceleration and deceleration rates are measured as $6m/s^2$ and $-7m/s^2$ respectively.

B. Randomized Fault Injection:

We performed another experiment where the fault injection is done randomly. This experiment was done on our simulator where the fault injection time was selected between [0, 10]s after the spawn time of a CAV, the traffic flow was randomly selected between [0.01, 0.15] car/second/lane and the probability of injecting a DEC fault was equal to an ACC fault. In total, 500 faults were injected during 20 hours of simulation, out of which 110 could cause an accident but they were avoided.

C. Recovery Analysis

To check if the intersection can recover after a rogue vehicle leaves the intersection area (for an ACC fault) or is removed (for a DEC fault) by a tow truck, we measured the average travel time of CAVs when a fault is injected. We first measured the average travel time for the normal operation of the intersection where no rogue vehicle is present and then repeated the same experiment and injected an ACC fault with the same traffic pattern and measured the average travel time. Next, we repeated the same experiment and injected a DEC fault with the same traffic pattern and measured the average travel time. These three experiments were repeated 20 times with different traffic patterns and the results were averaged. The final results for the average travel time of CAVs are shown in Figure 4. For the DEC fault, the rogue vehicle stops

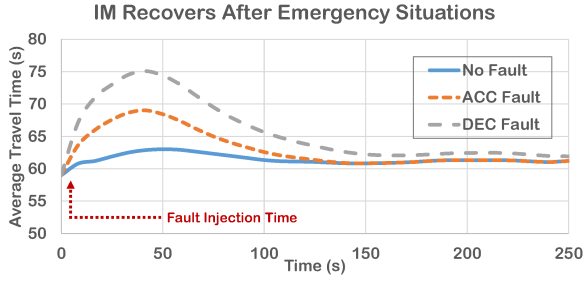

Fig. 4. Average travel time of CAVs increases when an ACC or DEC fault is injected but the intersection recovers.

for 10 seconds and then is removed. As can be expected, a DEC fault results in larger travel times compared to an ACC one since the duration of the emergency state is longer. Results show that the average travel time of CAVs increases when a fault is injected but the IM is able to recover and reduce the average travel time once the emergency situation is resolved.

D. Comparison with Traffic Light

To fairly compare the throughput of the R^{2}IM with other scheduling approaches, we developed a simulator for one of the state-of-the-art technique Robust Intersection Management (RIM) [16] and a simulator for an intersection that is managed by a traffic light. The green time of the traffic light is set to be 25 s, yellow time 5 s and red time 30 s. The same input flow and traffic patterns (spawn times) were used for all experiments to fairly compare the throughput of the intersection. Figure 5 shows the output flow rate of

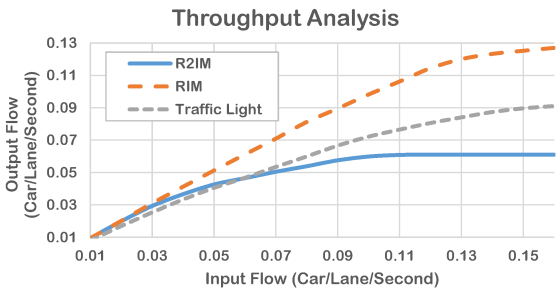

Fig. 5. Comparing the output flow rate of an intersection managed by a traffic light, R^{2}IM and RIM approaches.

CAVs vs the input flow rate for R^{2}IM, traffic light and the RIM approach when the input flow rate varies from 0.01 to 0.16 car/lane/second. The output flow is measured by counting the number of CAVs that leave the intersection in a 5-second time interval divided by the length of the interval (5). It can be observed that R^{2}IM always achieves a lower throughput compared to the RIM technique and this is because R^{2}IM allocates a larger temporal buffer between arrival times of CAVs for protecting CAV from a rogue vehicle. When comparing our approach with a traffic light, the R^{2}IM performs better than traffic light for low traffic flows ([0.01 to 0.05]) while traffic light performs better for higher flows. However, it should be noted that R^{2}IM can avoid accidents that cannot be avoided if the intersection is managed by a traffic light or RIM approach.

VII. Conclusion and Future Works

In this paper, we presented a resilient and robust intersection management algorithm that can detect rogue vehicles at the intersection which do not follow IM's command or are dishonest. Upon detection of a rogue vehicle, the IM broadcasts an emergency message to all CAVs. Enough space is considered between the cross-time of CAVs to ensure the safety of CAVs when a rogue vehicle is present. The resiliency of R^{2}IM is verified through both a formal proof and also conducting experiments on our 1/10 scale model intersection of CAVs and simulation. Future works include considering more sophisticated scheduling policies and extending the work to multi-lane intersections.

VIII. Acknowledgement

This work was partially supported by funding from NIST Award 70NANB19H144, and by National Science Foundation grants CNS 1525855 and CPS 1645578.

References

[1] AAA News Room. Red Light Running Deaths Hit 10 Year High. shorturl.at/bdjrH, 2019. [Online; accessed 03-March-2020].

[2] Joyoung Lee and Byungkyu Park. Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. *ITS Transactions*, pages 81–90, 2012.

[3] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008.

[4] Elnaz Namazi, Jingyue Li, and Chaoru Lu. Intelligent intersection management systems considering autonomous vehicles: a systematic literature review. *IEEE Access*, 7:91946–91965, 2019.

[5] Mohammad Khayatian et al. A survey on intersection management of connected autonomous vehicles. *ACM Transactions on Cyber-Physical Systems*, 2020.

[6] Li Li and Fei-Yue Wang. Cooperative driving at blind crossings using intervehicle communication. *IEEE Transactions on Vehicular technology*, 55(6):1712–1724, 2006.

[7] Reza Azimi et al. Stip: Spatio-temporal intersection protocols for autonomous vehicles. In *ICCPS'14: ACM/IEEE 5th International Conference on Cyber-Physical Systems*, pages 1–12. IEEE, 2014.

[8] Shunsuke Aoki and Ragunathan Raj Rajkumar. Dynamic intersections and self-driving vehicles. In *Proceedings of the ICCPS*, pages 320–330. IEEE, 2018.

[9] Changliu Liu, Chung-Wei Lin, Shinichi Shiraishi, and Masayoshi Tomizuka. Distributed conflict resolution for connected autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 3(1):18–29, 2018.

[10] Fethi Belkhouche. Collaboration and optimal conflict resolution at an unsignalized intersection. *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[11] Xiaoyuan Liang, Tan Yan, Joyoung Lee, and Guiling Wang. A distributed intersection management protocol for safety, efficiency, and driver's comfort. *internet of things journal*, 5(3):1924–1935, 2018.

[12] Chien-Liang Fok et al. A platform for evaluating autonomous intersection management policies. In *2012 IEEE/ACM Third International Conference on Cyber-Physical Systems*, pages 87–96. IEEE, 2012.

[13] Masoud Bashiri et al. Paim: Platoon-based autonomous intersection management. In *21st Conference on Intelligent Transportation Systems*, pages 374–380, Maui, HI, USA, 2018. IEEE.

[14] Qiu Jin et al. Platoon-based multi-agent intersection management for connected vehicle. In *ITSC*, pages 1462–1467, The Hague, Netherlands, 2013. IEEE.

[15] Luis Conde Bento et al. Intelligent traffic management at intersections supported by v2v and v2i communications. In *15th Conference on Intelligent Transportation Systems*, pages 1495–1502. IEEE, 2012.

[16] Mohammad Khayatian et al. RIM: Robust Intersection Management for Connected Autonomous Vehicles. In *IEEE Real-Time Systems Symposium*, pages 35–44, Nashville, TN, USA, 2018. IEEE.

[17] Muhammed O Sayin et al. Information-driven autonomous intersection control via incentive compatible mechanisms. *IEEE Transactions on Intelligent Transportation Systems*, (99):1–13, 2018.

[18] Mohammad Khayatian et al. Crossroads+: A time-aware approach for intersection management of connected autonomous vehicles. *ACM Transactions on Cyber-Physical Systems*, 4(2):20, 2019.

[19] Edward Andert et al. Crossroads: Time-sensitive autonomous intersection management technique. In *DAC*, page 50. ACM, 2017.

[20] Mohammad Hekmatnejad et al. Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In *MEMOCODE'19*, pages 1–11, 2019.

[21] Karl C Bentjen. Mitigating the effects of cyber attacks and human control in an autonomous intersection. Technical report, Air Force Institute Of Technology Wright-Patterson AFB OH, 2018.

[22] Karl Bentjen et al. Modelling misbehaviour in automated vehicle intersections in a synthetic environment. In *International Conference on Cyber Warfare and Security*, pages 584–XI. ACIL, 2018.

[23] Rachel Dedinsky et al. A dependable detection mechanism for intersection management of connected autonomous vehicles. In *Workshop on Autonomous Systems Design (ASD)*. Dagstuhl, 2019.

[24] Mohammad Farhadi et al. A novel design of adaptive and hierarchical convolutional neural networks using partial reconfiguration on fpga. In *High Performance Extreme Computing Conference*. IEEE, 2019.

[25] Mohammadreza Mehrabian et al. An efficient timestamp-based monitoring approach to test timing constraints of cyber-physical systems. In *Proceedings of the 55th Annual Design Automation Conference (DAC)*, pages 1–6, 2018.