R^{2}IM: Reliable and Robust Intersection Manager Robust to Rogue Vehicles

by

Rachel Dedinsky

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2019 by the
Graduate Supervisory Committee:

Aviral Shrivastava, Chair
Arunabha Sen
Violet Syrotiuk

ARIZONA STATE UNIVERSITY

August 2019

ABSTRACT

At modern-day intersections, traffic lights and stop signs assist human drivers to cross the intersection safely. Traffic congestion in urban road networks is a costly problem that affects all major cities. Efficiently operating intersections is largely dependent on accuracy and precision of human drivers, engendering a lingering uncertainty of attaining safety and high throughput. To improve the efficiency of the existing traffic network and mitigate the effects of human error in the intersection, many studies have proposed autonomous, intelligent transportation systems. These studies often involve utilizing connected autonomous vehicles, implementing a supervisory system, or both. Implementing a supervisory system is relatively more popular due to the security concerns of vehicle-to-vehicle communication. Even though supervisory systems are a step in the right direction for security, many supervisory systems' safe operation solely relies on the promise of connected data being correct, making system reliability difficult to achieve. To increase fault-tolerance and decrease the effects of position uncertainty, this thesis proposes the Reliable and Robust Intersection Manager, a supervisory system that uses a separate surveillance system to dependably detect vehicles present in the intersection in order to create data redundancy for more accurate scheduling of connected autonomous vehicles. Adding the Surveillance System ensures that the temporal safety buffers between arrival times of connected autonomous vehicles are maintained. This guarantees that connected autonomous vehicles can traverse the intersection safely in the event of large vehicle controller error, a single rogue car entering the intersection, or a sybil attack. To test the proposed system given these fault-models, MATLAB® was used to create simulations in order to observe the functionality of R^{2}IM compared to the state-of-the-art supervisory system, Robust Intersection Manager. Though R^{2}IM is less efficient than the Robust Intersection

Manager, it considers more fault models. The Robust Intersection Manager failed to maintain safety in the event of large vehicle controller errors and rogue cars, however R^2IM resulted in zero collisions.

DEDICATION

I dedicate my thesis work to my family and friends whose love and support are the reason I am here today. Also to my best friends Caffeine and Sugar, for a year of companionship through late nights of studying.

ACKNOWLEDGMENTS

Thank you to my Thesis Chair, Dr. Aviral Shrivastava, whose door was always open whenever I had questions about my research. Thank you also to my advising committee, Dr. Violet Syrotiuk and Dr. Arunabha Sen, for your support.

I am also very grateful to the Compiler Microarchitecture Lab for their guidance and insight, especially Mohammad Khayatian, Mohammadreza Mehrabian, and Sarthake Choudhary.

A very special gratitude goes out to all down at New American University Scholars' Fund, thank you for four wonderful years at Arizona State University.

Lastly, thank you to my wonderful parents, grandparents, boyfriend, and friends who inspired my love of engineering.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Rising traffic congestion has become inescapable, consuming our time and also endangering lives on the road. According to the Federal Highway Administration Office of Safety, the United States alone averaged 8,578 fatal crashes that occurred within an intersection between 2010 to 2017. Many of these accidents can be attributed to human error. In an effort to decrease the number of accidents and fatalities on the road, many autonomous, multi-agent frameworks have been studied that generally fall under two categories: cooperative traffic management, a vehicle-to-vehicle (many-to-many) communication, and centralized traffic management, an infrastructure-to-vehicles (one-to-many) communication between the Intersection Manager (IM)[1] and connected autonomous vehicles (CAVs), respectively. Prominent studies have been done by Li and Wang 2006, Neuendorf and Bruns 2004, Zohdy, Kamalanathsharma, and Rakha 2012, and Elhenawy et al. 2015 using a cooperative traffic management approach, however cooperative traffic management approaches are relatively less popular than centralized approaches due to the security concerns of vehicle-to-vehicle communication and their requirement for high network bandwidth. Therefore, centralized approaches have been more in the limelight in the research community. Figure 1 displays a direct comparison of the different kinds of centralized traffic management protocols. These centralized approaches include 1) Query-Based IM (QB-IM), 2) Velocity-Assignment IM (VA-IM), 3) Robust Intersection Management (RIM), and the method this thesis

[1]Intersection Manager and Intersection Management are interchangeable throughout the thesis, based on the context in which they read.

proposes 4) Reliable and Robust Intersection Management of Connected Autonomous Vehicles (R^{2}IM).

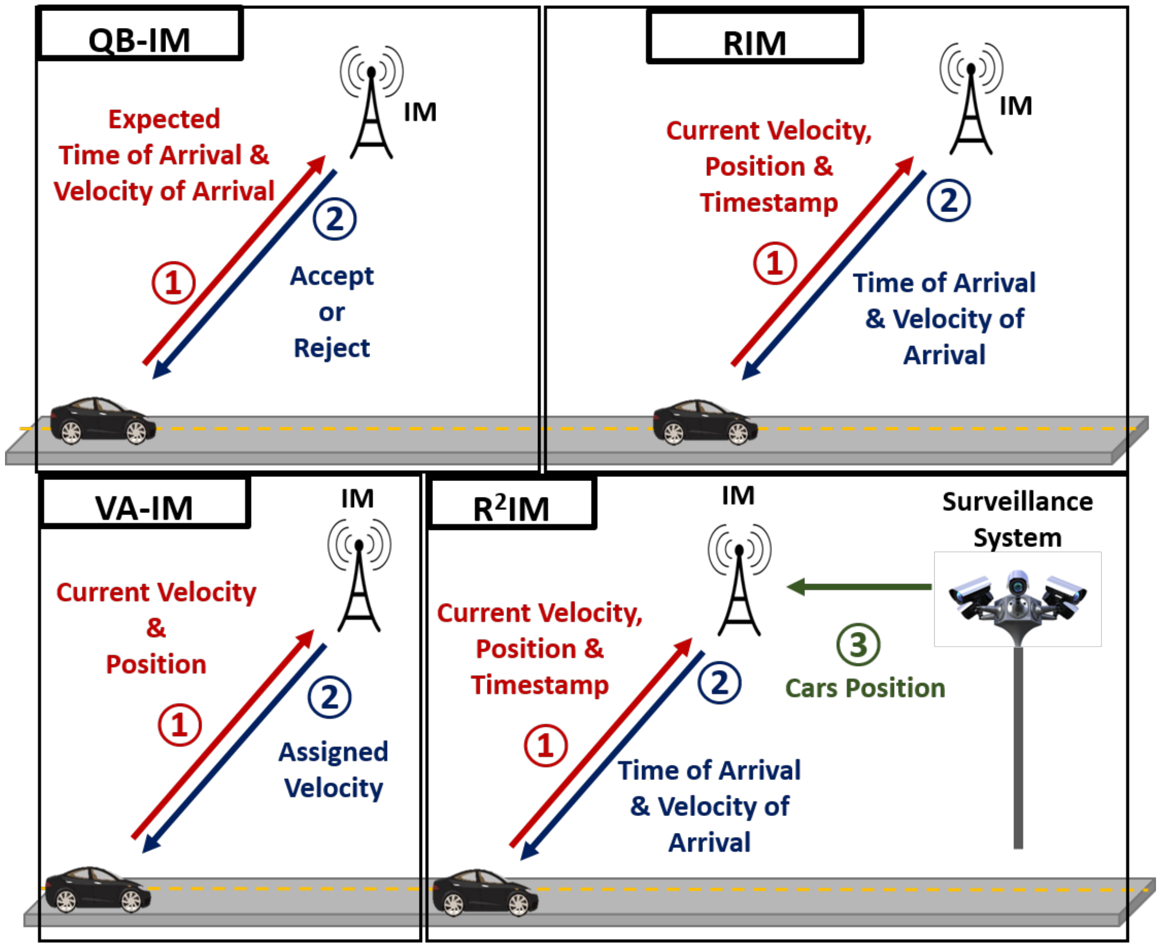

QB-IM
IM
Expected Time of Arrival & Velocity of Arrival
1
2
Accept or Reject
RIM
IM
Current Velocity, Position & Timestamp
1
2
Time of Arrival & Velocity of Arrival
VA-IM
IM
Current Velocity & Position
1
2
Assigned Velocity
R²IM
IM
Surveillance System
Current Velocity, Position & Timestamp
1
2
3
Cars Position
Time of Arrival & Velocity of Arrival

Figure 1. The centralized traffic management protocols are Velocity-Assignment Intersection Manager, Query-Based Intersection Manager, RIM, and finally the proposed Intersection Manager, R^{2}IM.

Cutting-edge examples of QB-IM were implemented by Dresner and Stone 2008 and Jin et al. 2012. In these studies, each CAV sends a computed Velocity of Arrival (VOA) and Time of Arrival (TOA) to the IM. The IM responds with either a success or failure message corresponding to whether or not the CAV trajectory is clear for the requested time interval. If the IM's response is a success message the car may proceed, but if IM's response is a failure message then the car must slow down and send the request again. This continues until the CAV receives a success message from the IM or the vehicle slows to a stop outside the intersection. Due to the high communication overhead of vehicles constantly communicating with the IM, QB-IM experiences high network delay which can lead to slower traffic and reduced efficiency. Another concern is the high computation required by the vehicles to schedule themselves.

Studies performed by Andert, Khayatian, and Shrivastava 2017 and Lee and Park 2012 iterated on QB-IM and developed VA-IM. In VA-IM, a CAV sends its current position and velocity to the IM and the IM responds with an assigned velocity. The CAV is then expected to approach and traverse the intersection at that velocity so as to avoid collisions in the intersection. In order to account for position uncertainty, VA-IM considers a safety buffer or an invisible barrier[2] around each car during scheduling. VA-IM is more consistent in scheduling down-time between CAVs because it can essentially specify arrival times instead of waiting for CAV requests as with QB-IM. Additionally VA-IM has less network traffic overhead because vehicles are not constantly requesting access to the intersection, aiding VA-IM to be more efficient than QB-IM. However, VA-IM is also flawed as it does not typically account for network delay, time to actuate to the target velocity, and delay caused by the IM's computation. VA-IM typically assumes that the vehicles can quickly reach their assigned velocity, follow a trajectory according to that velocity, and that the safety buffer absorbs all position/velocity inaccuracies. But this safety buffer can still be violated.

To solve many of VA-IM's issues, Khayatian, Mehrabian, and Shrivastava 2018 created RIM which receives the current velocity, position, and timestamp and assigns

[2]The length in meters of the safety buffer depends on the dimensions of the vehicle, relative accuracy of sensors, and maximum specified velocity of the vehicle. The safety buffer must also take into account the worst-case network delay and worst-case execution time.

a VOA and TOA instead of a fixed velocity. RIM solves the issue of safety buffer violations generally associated with VA-IM by providing CAVs with more flexibility in their trajectory; the IM calculates an optimal, feasible position trajectory for a requesting CAV and the CAV actuates with any feasible acceleration needed to achieve the assigned TOA and VOA. This makes RIM robust to external disturbance and model mismatches, situations which would cause VA-IM's safety buffer to be violated and scheduling[3] to fail.

Though RIM solves many security concerns, consider a vehicle which has large controller error, randomly accelerates or decelerates out-of-control, or is spoofing its information; this car will inevitably crash when RIM is implemented because the IM relies on connected vehicle data accuracy and is not equipped with security measures for these vehicle failure scenarios. To ensure vehicle failure does not result in an accident, it is necessary for the IM to have access to environmental data. By implementing an environmental sensing system, connected vehicle data can be verified and safety buffers can be monitored to ensure there are no erroneous scenarios occurring.

This thesis introduces the Reliable and Robust Intersection Manager, or R^{2}IM, an intersection management scheme that uses a surveillance system to increase data redundancy, enabling the IM to make more informed and accurate scheduling decisions for CAVs. R^{2}IM guarantees safety by utilizing position data from both the detection system and connected vehicle data to ensure vehicles are detected. Further, it proposes a new methodology for handling safety buffers temporally by ensuring it is physically impossible for vehicles to collide in the event that a vehicle has large controller error or goes rogue.

[3]Scheduling refers to the space-time schedule that the IM creates to guide CAVs safely and efficiently through an intersection.

In order to test the different aspects of the proposed system, MATLAB® was used to simulate the different CAV failure scenarios. A short study on cyber-attacks was performed, where the Surveillance System is used to verify position data to avoid vehicles spoofing data and encryption is used to ensure denial of service attacks were difficult to achieve. In RIM, this inevitably slows throughput as vehicles which are not present is scheduled in the intersection however it does not affect the throughput of R^{2}IM since cyber-attack vehicles will not be considered for scheduling. For testing the vehicle controller error, the set of acceptable position trajectories was used to determine whether or not randomly generated vehicles were within acceptable error bounds. This was coupled with testing tolerance to rogue vehicles since a vehicle with large vehicle controller error should be considered rogue since it isn't following the IM's instructions. In these tests, vehicles either accelerated or decelerated out-of-control at random points in time for a random time duration. The simulation showed that R^{2}IM was able to avoid all collisions in the event of these failures whereas RIM failed between approximately 3% and 16% of the time.

This thesis will discuss the current state of research in the field of intersection management of autonomous vehicles, the methodology proposed to create a more robust solution, the experiments performed to test the R^{2}IM's correctness and efficiency, and concluding with future iterations that would make R^{2}IM even more powerful in the future.

Chapter 2

RELATED WORKS

Many existing approaches have been proposed to manage connected autonomous vehicles through intersections. The first method to be researched is QB-IM. A great example of a QB-IM research is Autonomous Intersection Manager (AIM) introduced by Dresner and Stone 2008. Vehicles are scheduled using the First-Come-First-Served (FCFS) technique where vehicles are served based on the communication initiation time; those with earlier communication initiation times are served first. AIM implements QB-IM exactly as previously mentioned, and thus has the same problems to consider. The first problem is that the IM must process multiple requests and perform calculations to ensure the suggested VOA and TOA does not interfere with the schedule already in place, creating high computational load. The second problem is heavy network traffic; when many requests are received by the IM, it could cause scheduling delay and slow the overall throughput of the intersection. The third problem is that the scope of safety assumptions is limited to perfect operation of cars, meaning it is not fault tolerant for model mismatches[4], external disturbances, car malfunctions, or attacks[5].

Another QB-IM study performed by Jin, Wu, Boriboonsomsin, and Barth 2012

[4]In order for the IM to schedule a vehicle, it must have a pre-existing vehicle model by which it can determine the vehicle's motion. For example, a vehicle which is assigned a constant velocity has a linear motion and so the vehicle model is considered to be two-dimensional. A model mismatch occurs if the vehicle doesn't behave according to the assumed vehicle model.

[5]AIM also has mechanisms to control human-driven vehicles in addition to CAVs, however as the scope of this thesis is limited to full penetration of autonomous vehicles in the driving space only the fully autonomous system will be considered.

is “Advanced intersection management for connected vehicles using a multi-agent systems approach” which promises to improve vehicle emissions, fuel consumption, traffic control, and safety of automotive travel. Vehicle agents interact with intersection management agents inside a time-space, n x n grid. Reservations are made on a FCFS basis when vehicle agents successfully request from the the IM a VOA and TOA in which critical grid spaces are not occupied. This research was further extrapolated to use platooning, as done in the study Jin et al. 2013, to make further improvements. Platoons have the ability to reserve multiple grid spaces for traversing the intersection safely. However, Jin, Wu, Boriboonsomsin, and Barth have the same primary problems associated with AIM; this method is not fault tolerant, has high computational delay, and high network delay though lessened by implementing platoons.

In comparison to QB-IM, there is VA-IM which diminishes the problems of heavy network traffic and computational load. A great example of VA-IM is the study Crossroads by Andert, Khayatian, and Shrivastava 2017, where all vehicles first synchronize with the IM, then vehicles send a data packet with their current position and velocity. Finally the IM sends each vehicle an actuation timestamp and a target velocity to maintain from reception through intersection traversal. In Crossroads, vehicles have deterministic behavior but accidents can still occur. This is because CAVs track a constant velocity instead of a trajectory through the intersection. By tracking a constant velocity, the time it may take a vehicle to accelerate or decelerate to reach the target velocity must be absorbed by the error within the safety buffer. If the safety buffer is exceededd, accidents occur. Crossroads also depends on the assumption of perfect execution, and so is not fault tolerant to model mismatches, external disturbance, car malfunctions, or attacks.

RIM iterates on velocity assignment by solving the problems of model mismatches[6] and external disturbances[7]. RIM's methodology is unique in that it does not provide a strict trajectory to follow; instead the CAV communicates a data packet containing information including current velocity and position and the IM will respond with a feasible TOA and VOA. It is left up to the CAV to calculate an optimal trajectory and track it over time. Even though RIM accounts for small vehicle controller error, like other IMs it is prone to large car malfunctions and intentional attacks like cyber-attacks and rogue vehicles.

All of these related works have made assumptions about vehicle operation, however there are far more inputs to consider when modeling an intersection than perfect operation. Vehicle failures, rogue vehicles, and attacks can cause accidents in all the above scenarios. Methodologies to handle such scenarios need to be considered if such works are to actually be implemented in the real world. R^{2}IM proposes a new fault model that does not assume the ideal case of vehicle operation and provides a solution to managing traffic safely in the event this fault model occurs.

[6]Model mismatches occur when a vehicle is not behaving as expected.

[7]External disturbances include elements such as wind and bumps in the road.

Chapter 3

RELIABLE AND ROBUST INTERSECTION MANAGEMENT

R^{2}IM is resilient to cyber-attack and vehicle controller error by providing an extra source of data via the detection system, aiding the IM to make more informed scheduling decisions thus increasing system reliability. R^{2}IM is comprised of seven independent phases: 1) The detection mechanism is constantly monitoring the intersection to detect CAVs that go astray from the protocol and/or avoid communicating with the Intersection Manager, 2) The CAV synchronizes its clock with the IM's clock, 3) The detection mechanism checks if the CAV that synchronized its clock is actually physically present near the intersection to rule out the possibility of a potential cyber-attack, 4) If it is not present, the IM uses MAC blocking to ensure the vehicle does not increase network traffic, 5) If the vehicle is present, the car sends a request to the IM, passing its current velocity and position, 6) The IM responds with a TOA and VOA, and 7) The car actuates in order to reach the desired positions according to the VOA and TOA. In Dedinsky et al. 2019, we explained preliminary ideas of implementing an dependable intersection management system using a surveillance system.

3.1 Fault Model: Rogue Vehicle

A rogue car is an agent that starts to accelerate at an arbitrary and unknown acceleration in the range $[a_{min}, a_{max}]$, starting at an arbitrary time t. This may be a vehicle that breaks down, is a vehicle with malicious intent, is a human-driven vehicle, a vehicle which does not communicate with other vehicles and/or the IM, or a vehicle which does not follow the IM's instructions to name a few. We define the behavior of a rogue vehicle by the worst-case acceleration of a rogue vehicle or the worst-case deceleration of a rogue vehicle. In these cases, a rogue vehicle can be categorized as either a vehicle that accelerates or decelerates out-of-control and does not follow the IM's assigned trajectory or a vehicle whose vehicle controller error is large. The position trajectory of a vehicle which functions as expected has an associated acceptable error (ϵ) from the optimal trajectory. If a vehicle is within $\pm\epsilon$ distance from the optimal trajectory, then the vehicle is not marked rogue.

Consider the scenario where a vehicle is not following the calculated optimal trajectory. There may be two causes for this: 1) there is a large external disturbance and the vehicle does not actuate as expected or 2) there are delays in determining whether or not a position trajectory taken by the vehicle is valid (i.e. computation, communication, detection). If the position does not belong to the set of acceptable position trajectories, it will be considered rogue. The acceptable trajectories were experimentally determined where a chi-squared (χ^2) distribution was created with a maximum amplitude relating to the worst-case position error. The worst-case position error was found to be 15m by exhaustively testing VOAs with varying entry velocities. 15m error can only occur at the point where the vehicle transitions from actuating with the initial velocity to the VOA. This is because the expected position is calculated based on the instantaneous change in velocity, however since this is physically impossible the vehicle is allowed a large error to accelerate in order to recover to reach certain positions at specific times. Note the relationship in Equation Set 3.1 that calculates the χ^2 distribution used, where the distribution set the value of variable x to the considered time interval for the vehicle to cross the intersection,

k equals the point in time immediately after the time when the vehicle crosses the actuation line, and each element of the computed χ^2 probability distribution function (PDF) is multiplied by the maximum error ε=15m. Figure 2 shows the generated χ^2 PDF once it is multiplied with ε=15m.

$$\begin{aligned}
\varepsilon &= worst-case\ position\ error \\
Z_1, ..., Z_k &\leftarrow independent\ standard\ normal\ random\ variables \\
Q &= \sum_{i=0}^{k} Z_i^2 \ \leftarrow sum\ of\ squares \\
Q &\sim \chi_k^2 \leftarrow sum\ distributed\ according\ to\ \chi^2 \\
\mu &= k \leftarrow degrees\ of\ freedom \\
x &= t_i : t_f \leftarrow t_i = initial\ time, t_f = final\ time \\
PDF &= \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2} \\
PDF_i &= PDF_i \cdot \varepsilon
\end{aligned} \tag{3.1}$$

Once the χ^2 distribution is created and its maximum amplitude is related to the εm at time k, it is added and subtracted from the calculated optimal position trajectory to obtain the acceptable bounds for a normally operating vehicle as seen in Equation 3.2.

$$\begin{aligned}
trajectory_{optimal} &= expected\ position\ trajectory \\
trajectory_{optimal} - PDF \leq trajectory_{optimal} &\leq trajectory_{optimal} + PDF
\end{aligned} \tag{3.2}$$

The chi-squared distribution tapers off at the settling point, which defines the time at which vehicles must follow the position trajectory with 5% error. The settling time was also determined experimentally as the point in the majority of vehicle position

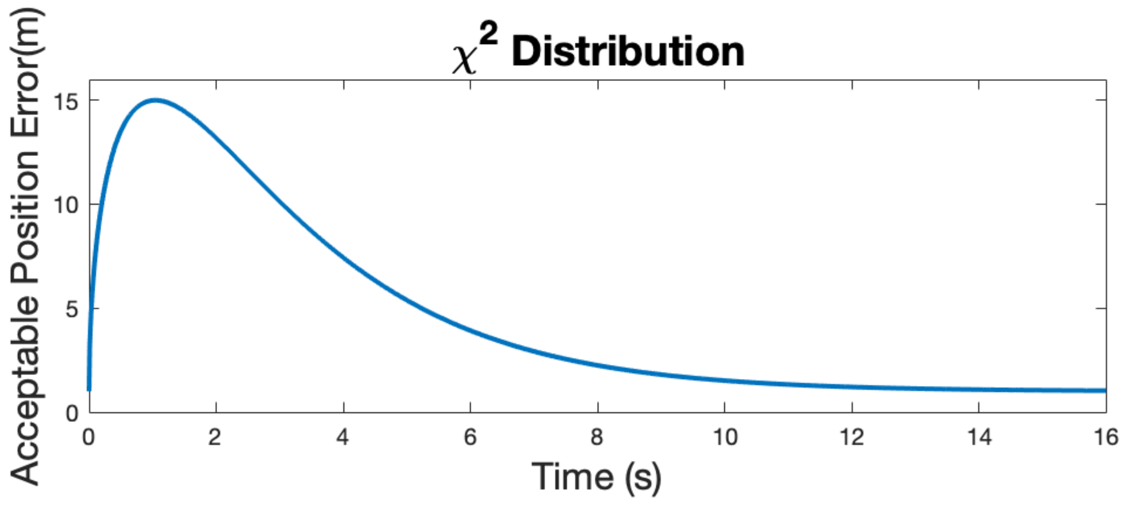


Figure 2. The resulting χ^2 PDF after multiplication with ε=15m will result in a distribution similar to this. Note the maximum amplitude occurs around the time when the target velocity changes from the initial velocity to the VOA.

trajectories where the error from the optimal path is less than or equal to 5%. The position error is a result of the Proportional-Integral-Derivative (PID)[8] controller behavior and system delay, since anything outside these error sources may result in dangerous vehicle behavior and the vehicle should therefore be marked rogue.

An example of acceptable position trajectories is shown in Figure 3, where there is some arrival velocity applied until the vehicle reaches the actuation line at -200m[9]. Once the actuation line is reached the vehicle should continue traveling with the assigned VOA toward the intersection. The expected position is calculated based on the expected velocity at a given timestamp. Since the expected velocity changes instantaneously from the arrival velocity to the VOA once the actuation line is crossed,

[8]This PID controller is a control feedback loop that modulates vehicle's speed in order to mitigate the position error. The position error is calculated based on the vehicle's actual position versus the vehicle's expected position at certain timestamps.

[9]The position of the actuation is negative because the vehicle's movement started from a negative position and moved toward the intersection which was at position 0m.

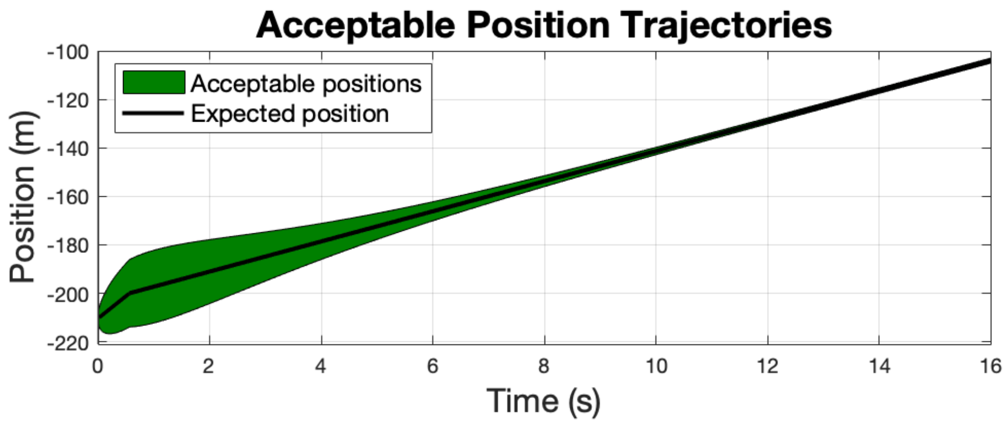


Figure 3. The expected position of the car is based on the VOA, initial velocity, and distance traversed over time. The acceptable bounds for error is a probability distribution with a maximum at the point in time where the position is likely furthest from the expected position. This probability distribution is added and subtracted from the expected position to obtain the lower and upper bounds of acceptable position trajectories.

the expected position around the time of the velocity change is very difficult to physically attain. The vehicle must accelerate and decelerate to reach the expected position at a given timestamp. Eventually the vehicle will catch up to its position trajectory and the vehicle's acceleration will approach zero, the actual velocity of the vehicle will be approximately the same as the expected VOA, and the expected positions will be reached at the correct timestamps.

The last fault model considered is a cyber-attack. One form of cyber-attack is a sybil attack. According to Bentjen 2018 a sybil attacker uses false data to authenticate with a system in order to cause harm. In this case, the cyber-attacker's malicious intents are directed toward the IM in order to cause jamming or spoof the position of a vehicle which is not actually at that location. R^{2}IM uses the surveillance system to check whether or not a requesting vehicle exists. If the vehicle does exist, normal operation ensues, however if the vehicle does not exist, the IM sends an emergency packet to all vehicles to proceed with caution and uses MAC blocking to prevent a malicious attacker from sending packets and slowing down network throughput.

3.2 Practical Assumptions

1. A Surveillance System should have the ability to detect anomalies that may occur inside or outside the intersection. Some of these anomalies may include communication failure, cyber-attack, rogue vehicles, and road cautions such as pedestrians, bicyclists, garbage, etc. The scope of this study assumes operation will not be interrupted by road cautions interfering with vehicle trajectories.
2. The IM cannot control rogue vehicles targeting CAVs maliciously. To avoid such collisions, it is assumed that all vehicles are well equipped with Adaptive Cruise Control (ACC). R^{2}IM was designed to be robust against only one rogue vehicle at a time in the intersection.
3. In R^{2}IM, vehicles turning inside the intersection follow the same methodology as RIM, where vehicles are not limited to slower speeds approaching the intersection and slow down only while they turn. This keeps throughput high compared to typical VA-IM strategies.

3.3 Vehicles

The vehicle interacts with the IM by first synchronizing[10] its clock with IM's clock. The vehicle decelerates to a stop if the vehicle does not synchronize with the IM on

[10]Synchronization is the process of aligning the clocks of the vehicle with clock of the IM. This makes the trajectory calculated by the vehicle the same as the trajectory calculated by the IM for that vehicle

time. Once synchronized, the vehicle transmits its position and velocity to the IM and waits for a timeout period to receive a VOA and TOA. If the VOA and TOA were not received within the timeout, the vehicle decelerates to a stop until it receives these values. If the vehicle receives the VOA and TOA within timeout, it calculates a reference trajectory and follows it using a typical PID controller calculated based on position error. The reference trajectory is followed after the actuation line is crossed. The actuation line's position is shown in Equation 3.3, where the position of the actuation line ($Pose_{actuation}$) is the position of the transmit line ($Pose_{transmit}$) plus the distance traveled if the car is moving with v_{max} over the time duration related to the the worst-case round-trip communication delay between the IM and a CAV. In this equation, Δt_{WCRTD} refers to the time duration of the worst-case round-trip delay.

$$Pose_{actuation} = Pose_{transmit} + \Delta t_{WCRTD} \cdot v_{max} \tag{3.3}$$

If a vehicle receives an emergency signal from the IM, the vehicle comes to a complete stop and waits for the IM to recalculate its VOA and TOA. Algorithm 1 in Appendix A displays the pseudocode for the vehicle controller. The reference trajectory, *Ref-erence_ Traj*, is calculated based on Equation 3.4, where $VOA \cdot \Delta t$ is the distance travelled given a time duration Δt and a vehicle traveling with velocity VOA.

$$D_{trajectory} = VOA \cdot \Delta t \tag{3.4}$$

Refer to Figure 4 for a visualization of the vehicle controller algorithm. The steps of the flow chart are followed until the vehicle crosses the intersection, at which point it continues through the intersection with the assigned velocity.

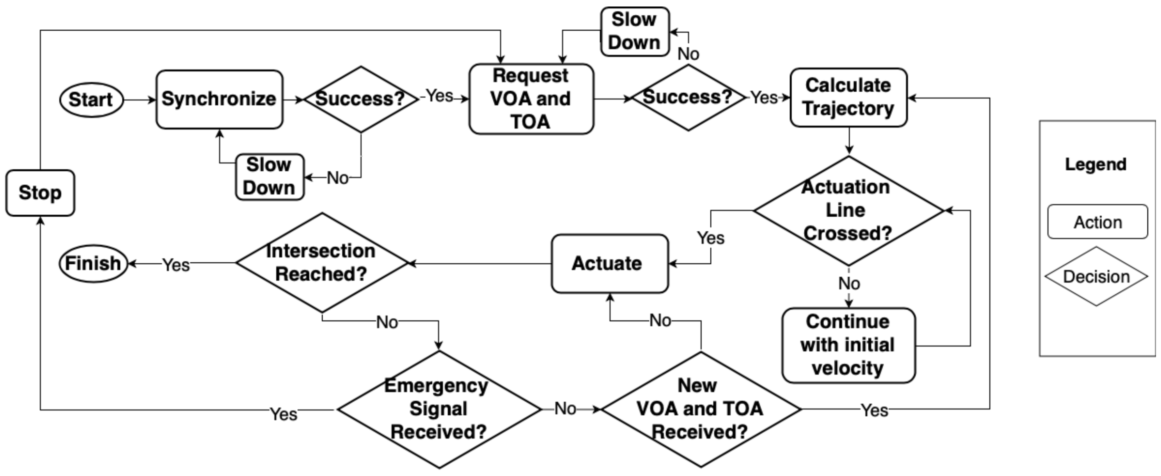


Figure 4. A flow chart describing the methodology behind a vehicle's behavior in R^{2}IM.

3.4 Intersection Manager

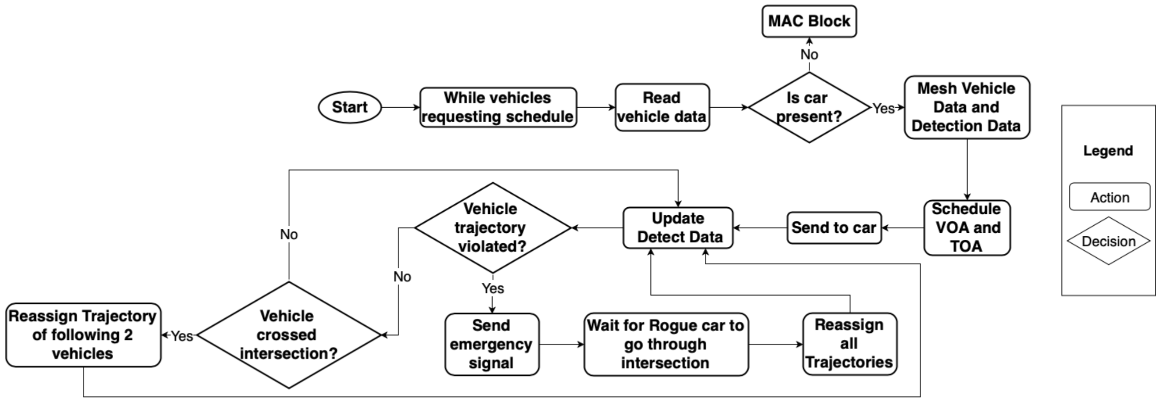
In order to calculate a vehicle's VOA and TOA, R^{2}IM depends on both the connected vehicle data and the detection system data. Algorithm 2 in Appendix B shows the pseudocode for the IM. The IM maintains its own data structure called IM_Data that stores each vehicle's data. Further, IM_Data provides a space-time schedule of previously calculated vehicle VOAs and TOAs. The IM uses IM_Data to ensure no two vehicles have conflicting schedules. V_Data is the vehicle data packet comprised of identification (ID), position, velocity and timestamp. Detect_Data is the detection data packet which is comprised of the ID, position, velocity, and timestamp of each vehicle. V_Data and Detect_Data mesh[11] together in a ratio related to the accuracy of the detection data in order to create IM_Data, as shown in Equation Set 3.5.

[11]Meshing combines two data sources. This allows the position of the vehicle as perceived by the IM to be in terms of the data received from both the surveillance system and the vehicle itself.

$$\begin{aligned} &\alpha = detection\ probability \\ &IM\_Data(V\_ID).position = V\_Data(1-\alpha) + Detect\_Data(\alpha) \end{aligned} \tag{3.5}$$

The control loop in Algorithm 2 starts by checking *RequestQueue*, which is a queue that holds vehicles that have not been added to the scheduling policy. If the queue is not empty, a vehicle is dequeued and the vehicle's status and position is checked using *Check-Vehicle-Behavior*(). In this function, if a vehicle tries to interact with the IM and the vehicle is not detected, IM_Data deletes all record of the vehicle and the vehicle's MAC address is blocked. However if the vehicle is present, the aforementioned meshing technique is used to create a new entry in IM_Data. Once the vehicle's status has been verified, a VOA and TOA for the vehicle is assigned by calling the function *Trajectory-Assignment*() which schedules the vehicle in IM_Data using a FCFS approach. Once the trajectory is calculated, the VOA and TOA are sent to the vehicle.

Algorithm 2 uses a timer interrupt which is necessary to update IM_Data. The interrupt updates Detect_Data with the newest data received from the Detection System and calls *Check-Detected*(). As the name suggests, *Check-Detected*() checks all the vehicles' (V_IDs') detected data from the Detection System for anomalies like a car accelerating or decelerating out-of-control. Based on this detected data and the associated accuracy of detection, IM_Data is updated. In the first part of *Check-Detected*(), a car is marked rogue if its position violates the reference trajectory by more than the allotted ϵ error. When a vehicle is marked rogue, the IM immediately calls *Broadcast-Alert*() function to broadcast an emergency stop packet to all the vehicles approaching the intersection. In *Broadcast-Alert*() after the rogue vehicle is safely outside the intersection, the IM reassigns all VOAs and TOAs of cars waiting to access the intersection and normal operation ensues. In the second part of *Check-Detected*(), if a vehicle is recognized with high detection accuracy and the IM has not scheduled it in IM_Data, the vehicle is added to IM_Data, is marked rogue, and the function *Broadcast-Alert*() is called. Lastly in the third part of *Check-Detected*(), if the Detection System recognizes a vehicle has exited the intersection, function *Trajectory-Reassignment*() is called. *Trajectory-Reassignment*() reschedules the following two vehicles (V_ID+1 and V_ID+2) after the vehicle which has just exited (V_ID) earlier TOAs and faster VOAs. This results in two reassignments per vehicle. The reassignment works by scheduling V_ID+1 with the earliest feasible TOA given velocity constraints (i.e., if the vehicle following the exited vehicle is 100m from the intersection, the earliest TOA is the time to accelerate and the time to cross the remaining 100m traveling with velocity v_{max}) and V_ID+2 is also rescheduled to arrive after V_ID+1 using the calculated temporal safety buffer which will be discussed in detail later in Chapter 3.6.



MAC Block
No
Start
While vehicles requesting schedule
Read vehicle data
Is car present?
Yes
Mesh Vehicle Data and Detection Data
Legend
Action
Decision
Schedule VOA and TOA
Send to car
Update Detect Data
Vehicle trajectory violated?
No
No
Yes
Reassign Trajectory of following 2 vehicles
Yes
Vehicle crossed intersection?
Send emergency signal
Wait for Rogue car to go through intersection
Reassign all Trajectories

Figure 5. A flow chart describing the methodology behind the IM's behavior in R^{2}IM.

For a visualization on how the IM algorithm works in R^{2}IM, refer to Figure 5. The IM is always running, thus never reaches a finish state. While there are vehicles requesting to enter the intersection, the IM is constantly checking the detected data, scheduling vehicles, and ensuring anomalous behavior does not result in a crash.

3.5 Surveillance System

The Surveillance System provides R^{2}IM with a layer of redundancy, thereby increasing reliability. Algorithm 3 shows pseudocode for the Surveillance System. Vehicles_Present[ ] will keep track of the detected vehicles entering and exiting the intersection. Vehicles_Present[ ] also has an associated percent accuracy depending on the belief distribution that the vehicle is actually in a specific location. This belief distribution is produced based on sensor filtering mechanisms, like Bayesian filtering as done by Geist, Pietquin, and Fricout 2008.

It is assumed the Detection System is an array of sensors meshed[12] together in order to detect vehicles. There are many solutions available to detecting and locating objects. Many studies such as Rodriguez Serrano and Larlus 2013, Csurka and Perronnin 2008, and Hu, Larlus, and Csurka 2012 have been done in computer vision research with localization and identification techniques. Sensors such as LIDAR, ultrasonic, radar, and magnetometer could aid the cameras by implementing passive tracking mechanisms for more precise positioning. Figure 6 displays the general flow of the Surveillance System's algorithm.

[12]Meshing sensors' data means combining the collected sensors' data in order to create a single source of data that has eliminated noisy measurements.

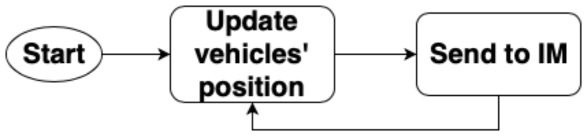


Figure 6. A flow chart describing the methodology behind the Surveillance System.

3.6 Mathematical Proof of Safety

The temporal safety buffer used for scheduling in RIM can be seen in Figure 7. If a vehicle goes rogue and accelerates or decelerates out-of-control, a collision is possible. In order to prevent such a collision from happening, there must be enough time between scheduling of the vehicles so that operational vehicles can come to a complete stop if needed or proceed through the intersection safely.

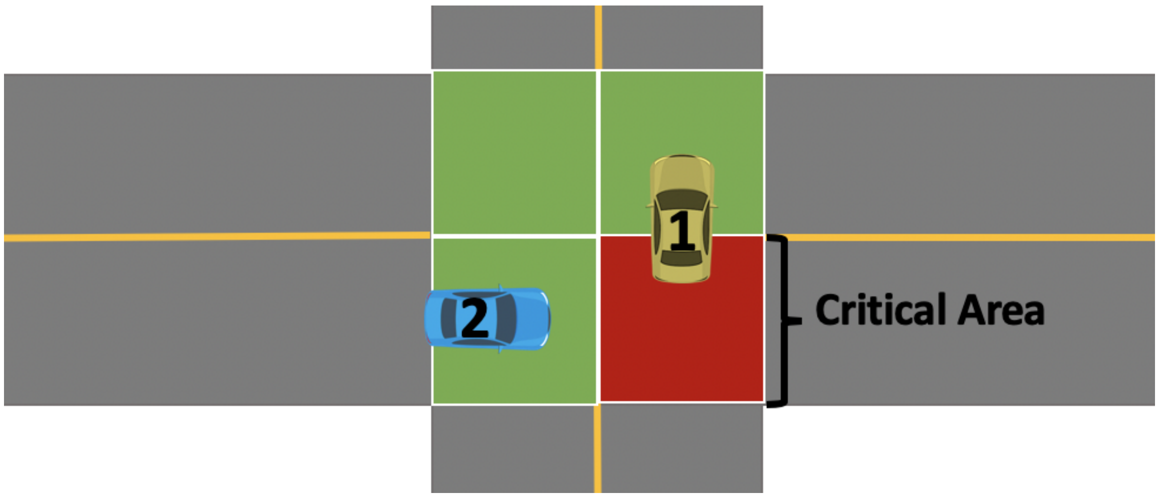


Figure 7. In the worst-case, RIM allows scheduling of vehicles directly after each other, as long as there is no conflict of trajectory.

For the remainder of this proof, projectile motion is mapped to a linear displacement equation. The equations of motion of a vehicle in two dimensions are as shown in Equation Set 3.6 where x, y are the position of the vehicle in Cartesian coordinate, θ is heading angle, v is velocity, a is acceleration, L is vehicle wheelbase and ψ is the steering angle, and u is the control input for the motor. K_p, K_i and K_d are PID controller gains, e, $\int e$ and $\dot{e}$ are the error between actual position and target position, its integral and derivative respectively and d(t) is the applied disturbance

$$\begin{aligned}
\dot{x} &= vcos(\theta) \\
\dot{y} &= vsin(\theta) \\
\dot{\theta} &= \frac{v}{L}tan(\psi) \\
\dot{v} &= u(t) \\
u(t) &= K_a(-K_pe - K_i\int e - K_d\dot{e} + d(t))
\end{aligned} \tag{3.6}$$

Since the expected vehicle motion has linear velocity, the 2-dimensional model is projected into a 1-dimensional model using Equation Set 3.7.

$$\begin{aligned}
\dot{x} &= v \\
\dot{v} &= a
\end{aligned} \tag{3.7}$$

In the case study portrayed by Figure 8, vehicle 2 accelerates out-of-control towards the critical area. In order to ensure vehicle 1 makes it safely through the intersection, it must be physically infeasible for vehicle 1 to collide with vehicle 2. A global standard for maximum and minimum acceleration and velocity is defined by Equation Set 3.8 in order to give context to the worst-case calculations. These global standards are based on the worst-case acceleration and deceleration (i.e., if a moving truck is in the same system as a Ferrari sports car, the global standard for maximum deceleration will be based on the moving truck's maximum deceleration since it will have a worse braking system than the Ferrari sports car will.)

$$\begin{aligned}
v &= [v_{min}, v_{max}] \\
a &= [a_{min}, a_{max}]
\end{aligned} \tag{3.8}$$

The amount of time it will take to stop a vehicle (Δt_{stop})[13] will reach a maximum when the vehicle is moving with maximum velocity. This time to stop a vehicle is derived in Equation 3.9. Note that the value of a_{min} is negative because a negative acceleration is applied to decelerate the vehicle.

$$v = u + at \tag{3.9}$$

$$\Delta t_{stop} = \left| \frac{v_{max}}{-a_{min}} \right| \tag{3.10}$$

The distance the vehicle travels in time Δt_{stop} in order to come to a complete stop is D_{stop} meters, as seen in equation set 3.11.

$$\begin{aligned}
d &= \frac{1}{2}at^2 + vt \\
D_{stop} &= \frac{1}{2}(-a_{min})(\Delta t_{stop})^2 + v_{max}(\Delta t_{stop}) \\
D_{stop} &= \frac{1}{2}(-a_{min})(\left|\frac{v_{max}}{-a_{min}}\right|)^2 + v_{max}(\left|\frac{v_{max}}{-a_{min}}\right|) \\
D_{stop} &= \frac{1}{2}\frac{v_{max}^2}{a_{min}}
\end{aligned} \tag{3.11}$$

When a vehicle is less than D_{stop} meters away from the intersection, it may not have time to stop before the intersection, provided another vehicle, with an intersecting trajectory, starts accelerating out-of-control. Take the previously discussed example where 2 vehicles are scheduled, vehicle 1 is schedule first and vehicle 2 is scheduled second; vehicle 2 goes rogue as soon as vehicle 1, a vehicle with an intersecting trajectory, is D_{stop} meters away from the intersection. The objective is to make sure that vehicle 1 crosses the intersection before vehicle 2 reaches the intersection edge to avoid accidents. Note that if vehicle 2 goes rogue before vehicle 1 has reached D_{stop} meters from the intersection, then vehicle 1 has enough time to stop outside the

[13]Time duration values are denoted with a Δ in front of the value, whereas timestamps are not represented with a Δ before the value.

intersection. The time it takes vehicle 1 to cross the braking distance $(\Delta t_{crossD_{stop}1})$ and the intersection width $((\Delta t_{crossIntersection1}))$ is shown in Equation Set 3.12 as $\Delta t_{crossCriticalArea1}$.

$$\begin{aligned}
time &= \frac{distance}{velocity} \\
\Delta t_{crossIntersection1} &= \frac{intersection\ width\ (meters)}{VOA_1} \\
\Delta t_{crossD_{stop}1} &= \frac{D_{stop}}{VOA_1} \\
\Delta t_{crossCriticalArea1} &= \Delta t_{crossD_{stop}1} + \Delta t_{crossIntersection1}
\end{aligned} \tag{3.12}$$

In that amount of time, a vehicle which has gone rogue and is travelling with maximum velocity can travel D_{Rogue} meters shown in Equation 3.13.

$$D_{Rogue} = v_{max} \cdot \Delta t_{crossCriticalArea1} + v_{max} \cdot \Delta t_{delay} \tag{3.13}$$

Note that Δt_{delay} refers to the experimentally determined worst-case detection delay related to the Surveillance System. In the time between the vehicle going rogue and the detection of the rogue vehicle, the car could have traveled $v_{max} \cdot \Delta t_{delay}$ distance.

If vehicle 1 is already inside the critical area when vehicle 2 goes rogue, vehicle 1 must be able to traverse the intersection safely before the vehicle 2 reaches the intersection edge. This can be guaranteed by ensuring vehicle 2 is at least D_{Rogue} meters away from the intersection at the time vehicle 1 is D_{stop} meters away from the intersection.

To ensure vehicle 2 is D_{Rogue} meters away from the intersection, the vehicles must follow specific timing constraints. The IM needs the entry time of vehicle 2 (t_{entry2}, i.e. the time at which vehicle 2 crosses the transmit line) and needs to calculate the time at which vehicle 1 is expected to D_{stop} meters away from intersection$(t_{arriveD_{stop}1})$. Their difference is the amount of time, $\Delta t_{crossD_{travelled}}$, vehicle 2 has to cover $D_{travelled}$ meters. *road length* is considered to be the distance between the actuation line and the intersection edge. Based on these constraints, vehicle 2's VOA and TOA is calculated as shown in Equation Set 3.14.

$$\begin{aligned}
D_{travelled} &= (road\ length) - (D_{Rogue}) \\
t_{arriveD_{stop}1} &= TOA_1 - \Delta t_{crossD_{stop}1} \\
\Delta t_{crossD_{travelled}} &= t_{arriveD_{stop}1} - t_{entry2} \\
VOA_2 &= \frac{D_{travelled}}{\Delta t_{crossD_{travelled}}} \\
\Delta t_{safety} &= \frac{D_{Rogue}}{VOA_2} - \Delta t_{crossD_{stop}1} \\
TOA_2 &= t_{entry2} + \frac{road\ length}{VOA_2} \\
&= TOA_1 + \Delta t_{safety}
\end{aligned} \tag{3.14}$$

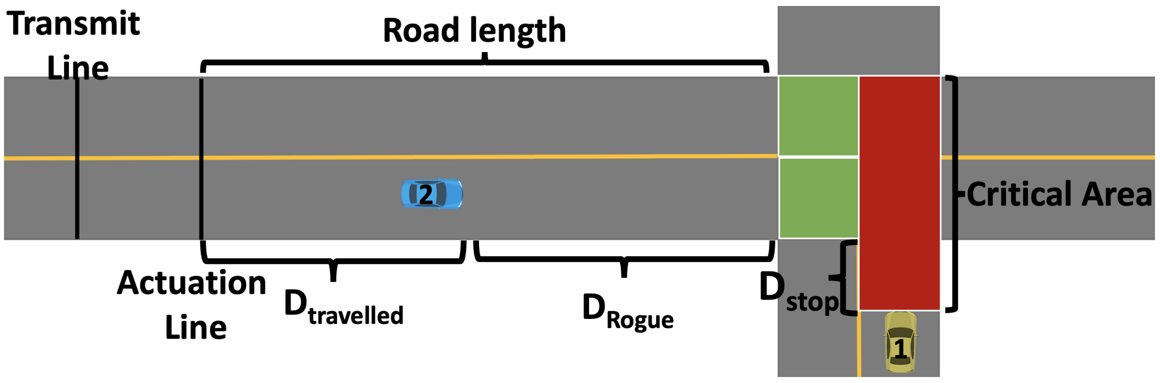


Figure 8. The relative position of the distances used in R^{2}IM.

In the case where vehicle 1 is decelerating out-of-control toward the intersection, vehicle 2 definitely has enough time to come to a complete stop outside the intersection and wait until the rogue vehicle 1 is safely outside the intersection. Once the rogue vehicle 1 is outside the intersection, vehicle 2 receives a new TOA and VOA from the IM and proceeds through the intersection.

Chapter 4

EXPERIMENTAL TESTBED

R^{2}IM provides a layer of data redundancy, making it resilient to vehicle controller error, rogue cars, and cyber-attack. Two experiments were conducted to ensure correctness: the first experiment was to guarantee no crashes occur in R^{2}IM in the presence of a rogue vehicle and the second experiment conducted was a throughput test. Throughput is calculated in two ways: 1) the inverse of average wait time of all the vehicles and 2) as the outgoing flow rate of vehicles. The flow rate is the number of cars that occur per lane per second. Finally a thorough study was conducted investigating cyber-attacks, which are intended to stop the intersection from working correctly by spoofing vehicle data to cause inaccuracies in the IM's scheduling policy. R^{2}IM detects these failures correctly and implements recovery scenarios to ensure accidents don't occur.

4.1 Rogue Car

Since the IM cannot determine the final destination of a rogue car, all vehicles must come to a complete stop and wait for the rogue car to complete its trajectory. The rogue vehicle's position is tracked using Detect_Data. Once the rogue car has exited the intersection, traffic flow resumes after reassigning each vehicle's VOA and TOA.

To test the efficacy of R^{2}IM, MATLAB® was used to build a 2-dimensional simulator. The simulator incorporates the vehicle's PID controller, the IM's instructions, random delay, random failures, random velocities, and random entry times. The simulator begins operation when the vehicle sends a request to the IM for the VOA and TOA, with the precondition that the vehicle and IM have synchronized. The vehicle receives the VOA and TOA from the IM and waits until the actuation line is crossed to actuate toward the expected trajectory. The distance between the transmit line and the actuation line is 10m since the maximum velocity considered was 20m/s and the minimum velocity considered was 0m/s. Consider the scenario where a vehicle is traveling 20m/s; it would take this vehicle 1s to travel 10m. According to Dedicated Short-Range CommunicationCommission Standards backed by the Federal Communication Commission and research completed by Savic, Schiller, and Papatriantafilou 2017, the worst-case round trip delay between vehicles and infrastructure is approximately 0.4s. This means the worst-case round-trip delay for communication between the IM and the CAVs is less than the time it takes the vehicle to traverse 10m even if the vehicle is traveling v_{max}m/s.

Two IMs were simulated: RIM and R^{2}IM. R^{2}IM's implementation differed from RIM in that it implemented an omnipotent detection system which was tolerant to failure scenarios such as rogue cars and large controller error. Once these situations were detected by the Surveillance System, the Reliable and Robust IM initiated resilience mechanisms. For the implemented resilience mechanism, the Surveillance System alerted the IM of erroneous vehicle behavior and the IM sent an emergency stop packet to all the incoming vehicles. Then recovery was implemented by reassigning the TOAs and VOAs of all stopped vehicles and normal operation ensued.

To maximize efficiency of the simulations, a study was done to determine the optimal values for the length of the road leading to the intersection and the number of assignments of [VOA,TOA] each vehicle received in normal operation. The average wait

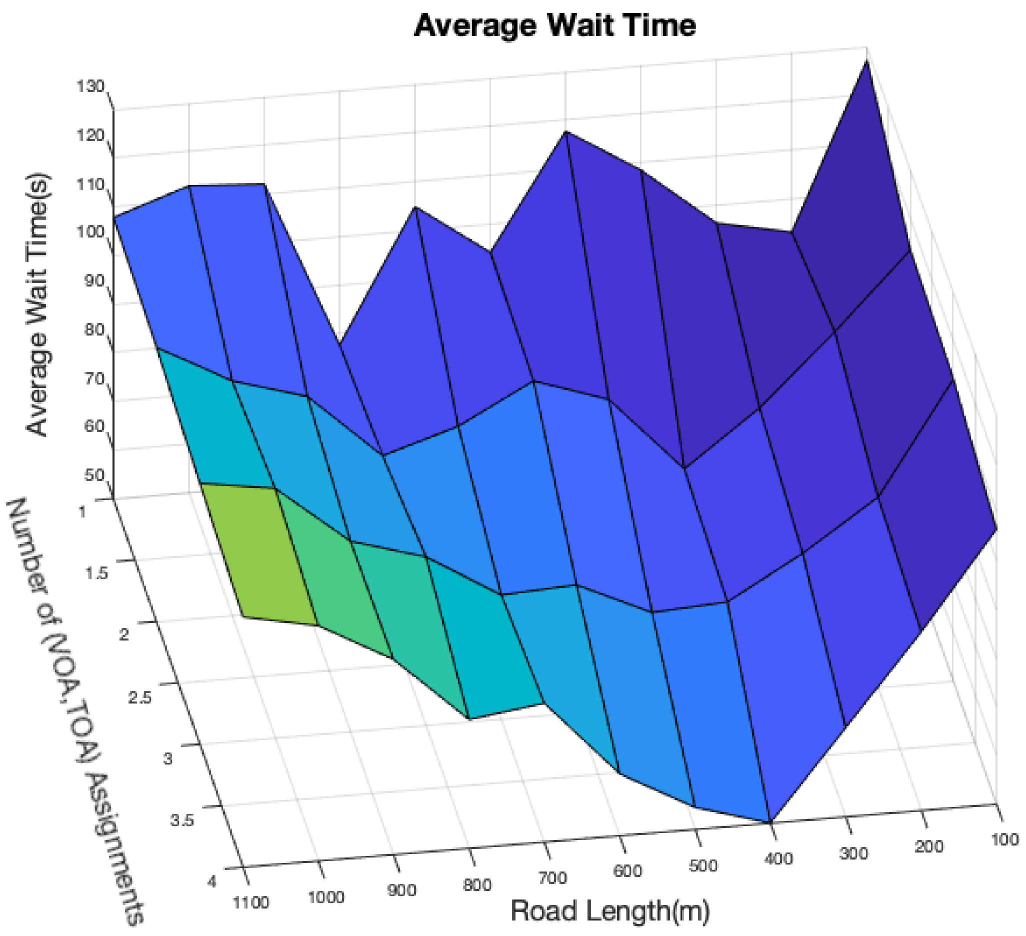


Figure 9. By varying different road lengths and number of assignments of [VOA,TOA], different average wait times are obtained. The color scheme shows the contours of the graph.

time was calculated to determine which combination was optimal as shown in Figure 9. Since three assignments was optimal in almost every case, one initial assignment and two reassignments were chosen. The road's length was chosen to be 200m since it had better average wait times than many other road lengths considered. Additionally, according to national averages posted by the Federal Transit Administration the distances between stops is at least 500m in most metropolitan cities. Since two intersections back-to-back will result in over 400m between potential stops when the road length leading to the intersection is 200m and the distance between the transmit and actuation line is 10m, 200m for the road length is the best option for coverage of most metropolitan cities.

According to "Reference", the average vehicle's length and width is about 5m x 2m so the simulated vehicles were 5m long and 2m wide. The simulated roads had a 12m road width, leaving the intersection to be of size 12m x 12m, as is standard according to the Transportation Engineering Agency. The maximum and minimum accelerations considered were 5m/s^2 and -8m/s^2 respectively. For testing if vehicles malfunctions would result in a collision, only two cars are simulated based on the following assumptions claimed by the following Proof by Induction. For testing throughput, 50 cars were simulated with varying flow rates(number of cars per second per lane).

Proof By Induction: Defense for Two Car Simulation

Only two safety critical interactions exist, where the first is when vehicle 2 starts accelerating out-of-control toward the intersection and the second is when vehicle 1 is decelerating out-of-control when moving towards or within the intersection. If another vehicle were added to the two-car proposed safety model, there are still only the two possible safety critical interactions that may happen - that is either a vehicle is traveling too fast or too slow. Therefore, in a multi-car intersection only the two safety critical scenarios are feasible. Since two cars can be used to model both these interactions, a multi-car simulation is not needed.

4.2 Cyber-Attack

R^{2}IM uses the Surveillance System to check whether or not a requesting vehicle exists, as shown in Figure 10. The vehicle's request is either accepted or rejected based on whether or not the vehicle is detected. In the case a cyber-attacker is trying to jam the communication network, Quigley and Peddoju, n.d. proposes a solution to ensuring jamming attacks like Denial of Service(DoS) attacks do not occur. Further, R^{2}IM assumes a high level of encryption exists between vehicles and the IM making cyber-attacks difficult.

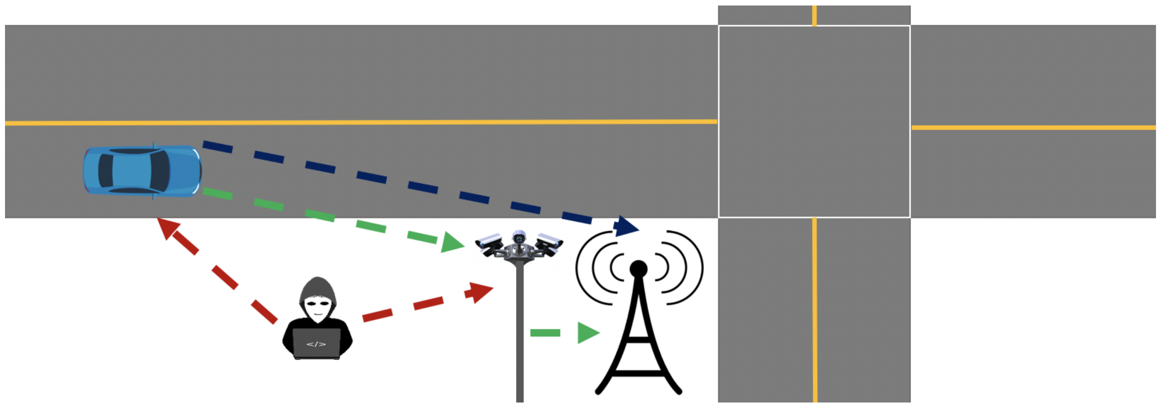


Figure 10. A cyber-attacker will spoof a car's position in order to cause an accident in the intersection. The Surveillance System can communicate to the IM whether or not the car exists.

Chapter 5

RESULTS

5.1 Safety

Safety experiments were conducted for both RIM and R^{2}IM. The resulting statistics can be seen in Table 1. The table breaks down the failure scenarios tested for safety into two categories: a rogue car that accelerates out-of-control and a rogue car that decelerates out-of-control. Unlike RIM, if a vehicle breaks down inside or outside the intersection the proposed system detects it through infrastructure sensing and takes preventative measures against collision. RIM on the other hand relies on other CAVs ACC to detect stalled, broken-down vehicles using ACC. However, ACC often will not allow enough time for the vehicle to come to a complete stop before its trajectory intersects with another vehicle.

Table 1. This table illustrates the statistics of the number of crashes, comparing RIM to R^{2}IM over 1000000 iterations.

IM	**Failure Scenario**	**Crashes**
RIM	Decelerating Rogue Car	159336
	Accelerating Rogue Car	38331
R^{2}IM	Deceleration Rogue Car	0
	Acceleration Rogue Car	0

Two vehicles were simulated, where one of the two vehicles goes rogue. To test the deceleration case, the first car, i.e., the car scheduled to cross the intersection first, goes rogue and randomly decelerates out-of-control. To test the acceleration case, the second car goes rogue and randomly accelerates out-of-control toward the intersection. The number of times these cases resulted in an intersection of the cars created the tally of number of crashes, as can be seen in Table 1. The input values to the simulation include:

- The initial velocity of the first car randomized between [5,v_{max}]m/s.
- The initial value of the second car randomized between [5,v_{max}]m/s.
- The assigned VOA of the first car randomized between [5,$v_{max} - 2$].
- The t_{entry2} of the second car is randomized between [0,TOA_1]s[14].
- The failure time of the vehicle.

The initial velocity values start at 5m/s because many states like Washington, California, and Pennsylvania in the United States have a minimum speed limit. Washington State Legislature notes, "No person shall drive a motor vehicle at such a slow speed as to impede the normal and reasonable movement of traffic". The upper limit of the range of initial velocity values is the global maximum velocity of the system, meaning the maximum velocity of the car which has the lowest maximum velocity value. The VOA of the first car is randomized between 5m/s and $v_{max} - 2$m/s to keep traffic flow efficient and to allow for position optimization. Position optimization is made possible by having a lower value for the upper bound on VOA than v_{max}; if a constant velocity is too large, the positions which can be obtained become infeasible to reach because the calculation for the optimal position trajectory does not account for time to accelerate. The bounds for t_{entry2} were chosen based on the times when the cars could feasibly cause a collision and times when it was infeasible. The failure time is the time at which a vehicle goes out-of-control; since a vehicle may go out-of-control at any time, the failure time of the vehicle has been randomized between a window

[14]All timing ranges listed are relative to 0s being the start time of the simulation, i.e. 5s means 5 seconds from the start of the simulation.

of values at which it is feasible for the vehicle to go out-of-control and affect other vehicles. The failure time range for both the acceleration and deceleration cases was randomized between [0,TOA_1]s. With these randomized input values, the simulation was run 1,000,000 times, where each time these input values were randomized. Further setup of the simulation parameters is described in Section 4.1. During 1,000,000 iterations of the simulation, RIM observed crashes and R^{2}IM maintained safety.

5.1.1 Case Study: Rogue Car Accelerates

This case study observes two vehicles, with intersecting paths inside the intersection, where the first vehicle (Car 1) is scheduled before the second vehicle (Car 2). Figure 11 shows the case where vehicle 2 is accelerating out-of-control towards the intersection. Since car 2 fails and is detected rogue before car 1 enters the critical area, car 1 comes to a complete stop. At approximately time 8s, car 2 accelerates out-of-control toward the intersection. The detection system recognizes car 2 has violated the set of acceptable position trajectories and at approximately 9s car 1 receives the emergency stop signal from the IM and begins to decelerate to a stop. Car 1 will remain stopped until car 2 reaches the end of the intersection, at which time it is safe for car 1 to traverse the intersection. Note that car 1 stays at approximately D_{stop}m from the intersection, where D_{stop}m equals 25m from the intersection[15] while the rogue car 2 completes its trajectory. Note that the vehicle controller is related to position and not velocity, so to correct for error in position the vehicle may accelerate or decelerate

[15]The value of the positions are negative because the simulated scenario had initial position -200m from the intersection and moved toward the intersection which was at position 0m.

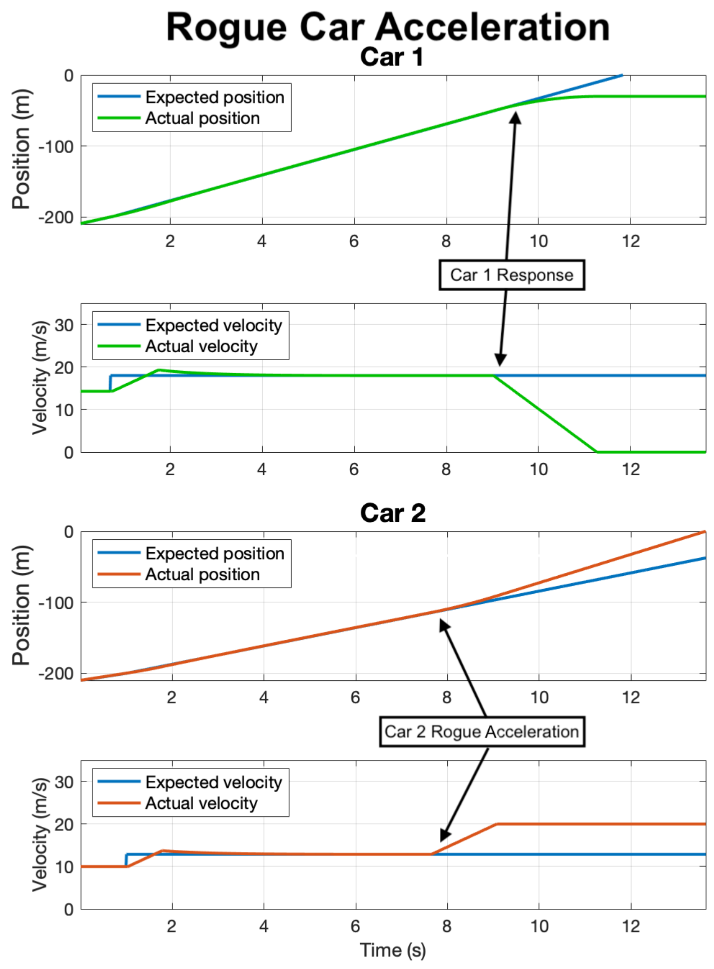


Figure 11. A vehicle is considered to be following the IM's instructions if it does not violate the acceptable set of trajectories. This case shows a car accelerating out-of-control. Car 1 is scheduled first and Car 2, the rogue car, is scheduled second. Car 1 responds to Rogue Car 2 by decelerating to a stop.

to achieve a certain position at a specific timestamp, resulting in velocity trajectories as seen in subplots 2 and 4 of the figure.

5.1.2 Case Study: Rogue Car Decelerate

This case study follows the same basic setup as in Chapter 5.1.1, where Car 1 is scheduled to enter the intersection before Car 2 and they have intersecting paths. Figure 12 depicts the scenario where Car 1 decelerates out-of-control and Car 2 must react by stopping. The first two sub-figures show Car 1's position and velocity. At approximately 17s, the first car decelerates out-of-control and maintains a 5m/s velocity, which is significantly less than the assigned VOA. In response, the Surveillance System recognizes Car 1 has gone rogue and alerts the IM; the IM then sends the emergency stop signal and Car 2 responds by decelerating to a stop at approximately 19s. This ensures Car 2 is safely outside the intersection while Car 1 crosses the intersection. Once Car 1 has crossed the intersection, Car 2 is reassigned to proceed through the intersection as quickly as possible at about 28s.

5.2 Performance

RIM's scheduling policy is different from R^{2}IM because the temporal safety buffer for R^{2}IM is far more conservative than RIM, as depicted in Figures 8 and 7 respectively. In order to analyze the performance of R^{2}IM compared to RIM, a MATLAB® simulation was used to compute the average wait time of both algorithms compared to a typical traffic light. The average wait time was calculated starting from the time the vehicle was requesting to traverse the intersection to the time the vehicle crossed the intersection. The simulation involved a four-way intersection with one lane of traffic per direction and used 50 cars. The input flow rate varied from 0.1 to 2 cars/lane/second. The simulation was run with the randomized initial values

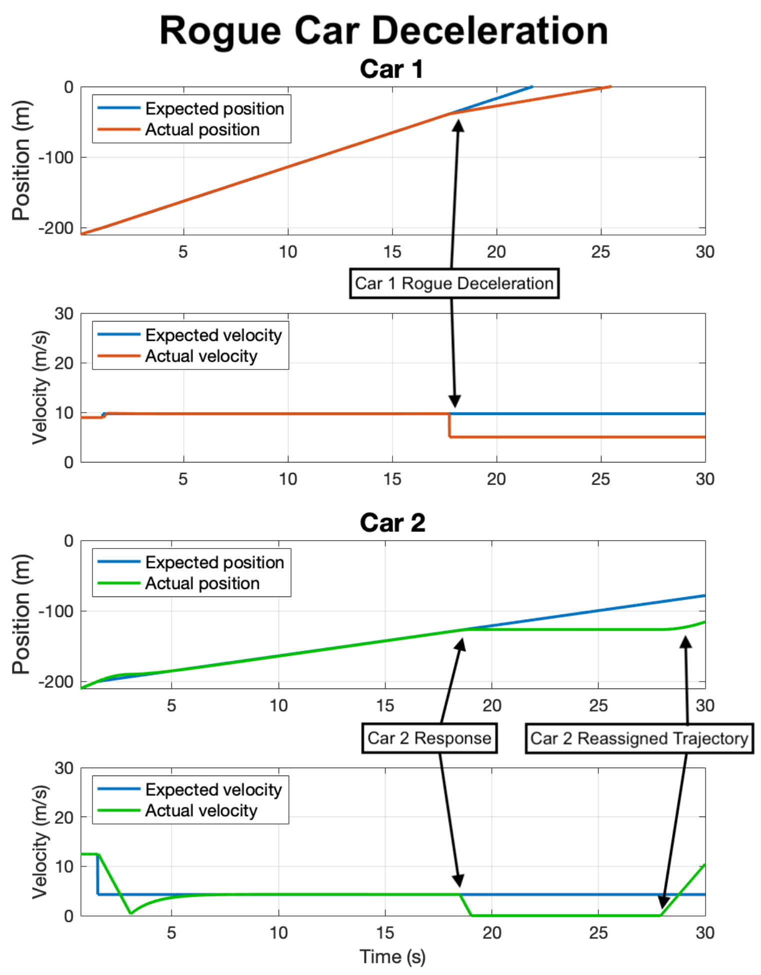


Figure 12. A vehicle is considered to be following the IM's instructions if it does not violate the acceptable set of trajectories. The case shows a car decelerating out-of-control. Car 1 is the rogue car which is scheduled first and Car 2 is scheduled second. Car 2 responds to Rogue Car 1 by stopping. Once Rogue Car 1 has finished its trajectory through the intersection, Car 2 is reassigned a VOA and TOA and continues toward the intersection.

as previously described. These inputs were randomized every time the simulation was run. The results can be viewed as shown in Figure 13. R^{2}IM does indeed have worse average wait times compared to RIM, however it does not compromise on safety precautions.

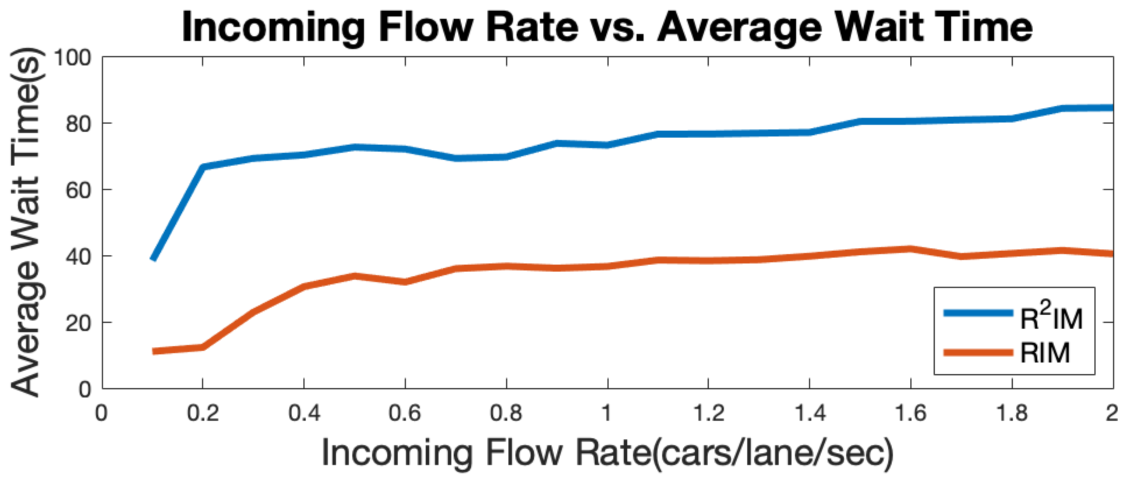


Figure 13. A one-lane, four-way intersection was simulated for RIM and R^{2}IM. As flow rate increases, the average wait the vehicles slowly increases.

Another way to look at throughput is the incoming flow rate versus the outgoing flow rate, as in Figure 14. RIM consistently has higher outgoing flow rate than R^{2}IM. Further, RIM's flow rate trend is essentially $x = y$, or incoming flow rate is equal to outgoing flow rate until it plateaus at approximately 0.2 cars/lane/second, however R^{2}IM plateaus at about 0.05 cars/lane/second.

Further analysis on factors affecting throughput of R^{2}IM shows that Δt_{safety} varies depending on VOA_1 and t_{entry2}. Assume two vehicles are simulated, where the first car scheduled is vehicle 1 and the second car scheduled is vehicle 2. The smaller Δt_{safety} is, the less the average wait time is for vehicle 2 and the higher the system throughput. Another simulation was created, where vehicle 1 is scheduled before vehicle 2. As VOA_1 and t_{entry2} increase, Δt_{safety} decreases. This simulation was tested multiple times with velocity increasing in steps of 0.1m/s from [5,20]m/s for VOA_1 and for t_{entry2} from [0,30]s with a step of 0.1s. The resulting Δt_{safety} from each combination of VOA_1 and t_{entry2} can be seen in Figure 15. As VOA_1 increases, vehicle 1 will reach

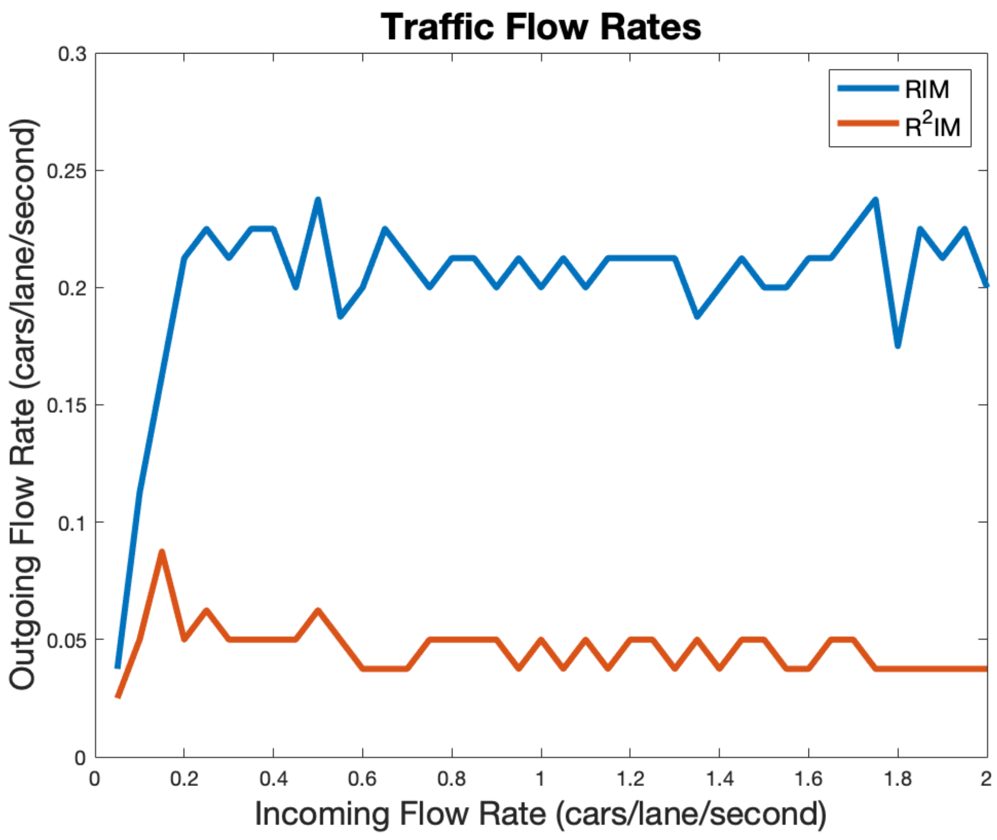


Figure 14. A one-lane, four-way intersection was simulated for RIM and R^{2}IM. As incoming flow rate increases, the outgoing flow rate decreases. Note that in every instance, RIM has higher outgoing flow than R^{2}IM.

the intersection edge in less time and so the downtime between scheduling vehicles can be less. Additionally, the later the second vehicle's t_{entry2}, the less wait time vehicle 2 has before entering the intersection.

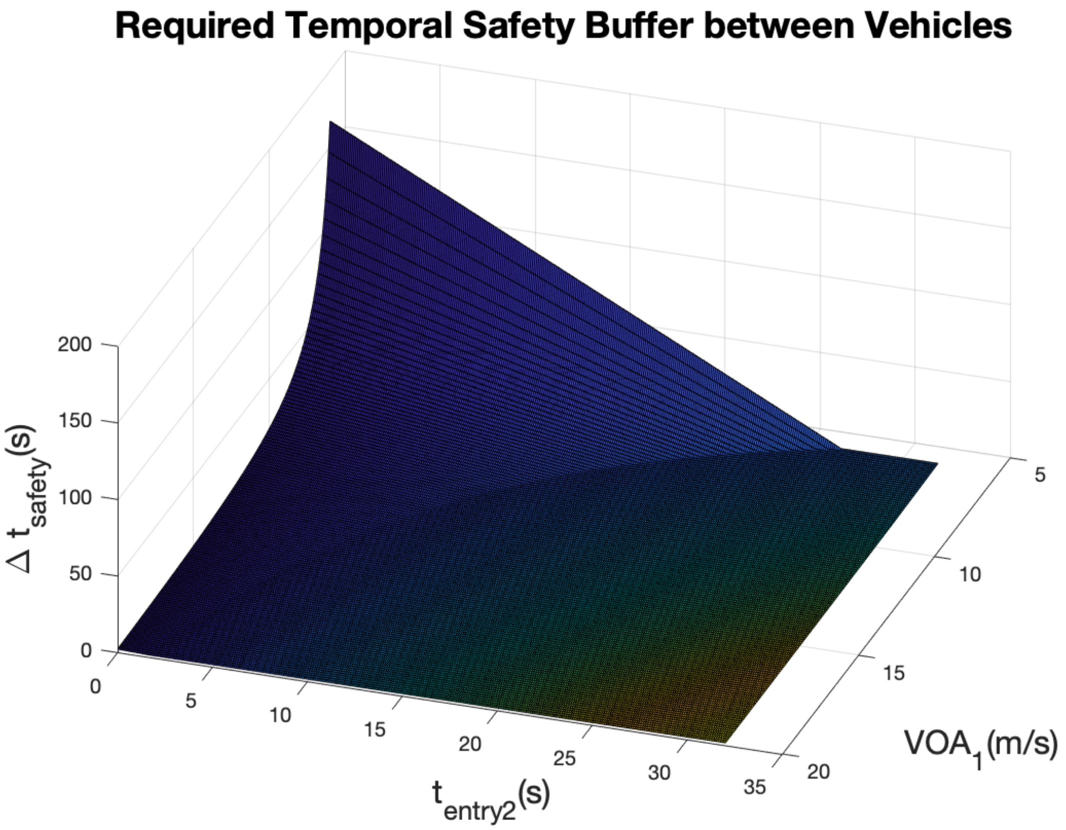


Figure 15. As VOA_1 and t_{entry2} increase, t_{safety} decreases and the wait time of vehicle 2 decreases. The worst-case is depicted when t_{entry2} is zero and VOA_1 is 5m/s.

Chapter 6

CONCLUSION

This thesis investigated a way to implement an intersection of autonomous vehicles which guarantees safety of controllable CAVs. This technique is both time and space aware, implementing a new centralized way to manage traffic by integrating vehicle data packets with environmental sensing data to increase data redundancy. The environmental sensing data obtained via the proposed Surveillance System ensures that each vehicle remains within reasonable bounds of the optimal trajectory. If the position of a vehicle violates the set of acceptable trajectories, the Surveillance System alerts the IM to take further actions to avoid a collision. Furthermore, R^2IM is robust to attack since the Detection System can determine the true location of a vehicle and implement blocking techniques in the event that a vehicle is spoofing its position or jamming the network. In order to prove reliability, R^2IM was tested using a MATLAB® simulation against the most robust model for CAV intersection management, RIM, and for each failure of RIM, R^2IM succeeded in avoiding collisions. Though conservative, in the event a single rogue car or a cyber-attack, R^2IM can guarantee safety.

6.1 Future Work

The fault model of the single rogue vehicle considered by R^2IM is small compared to the scope of possible errors that can occur within an intersection. This algorithm could be expanded to ensure safety of CAVs when there are multiple rogue agents interacting with the intersection. The way R^{2}IM would handle multiple rogue cars is the same as handling one rogue car, where all controlled vehicles would come to a complete stop and wait until the rogue vehicles have exited and it is safe to proceed. However if one rogue vehicle follows after another rogue vehicle, then all throughput through the intersection will halt until there are no rogue vehicles present, which is extremely harmful to graceful degradation. Another prospective area for extending R^{2}IM is to multiple lanes per direction. R^{2}IM was only tested in a four-directional, one-lane intersection. If each direction increased to two lanes, the total number of lanes in the intersection would then be eight. R^{2}IM's protocol only allows one of these eight lanes to access the intersection at one time, since R^{2}IM considers the rogue scenario of vehicles being able to collide with one another by swerving into different lanes inside the intersection. In addition to this, road cautions need to be considered in order to implement this system robustly in real-time. These road cautions include pedestrians, bicyclists, construction, etc. Lastly, a recovery mechanism should be considered if the IM is under a denial-of-service attack, in order to let controllable vehicles know an erroneous situation has occurred and to come to an emergency stop.

REFERENCES

Administration, Federal Transit. *Stops, Spacing, Location and Design.* https://www.transit.dot.gov/research-innovation/stops-spacing-location-and-design.

Agency, Transportation Engineering. *Standard Lane Widths.* https://www.sddc.army.mil/sites/TEA/Functions/SpecialAssistant/TrafficEngineeringBranch/BMTE/calcIntersections/intersectionsTurtorials/intersectionDesign/Pages/standardLaneWidth.aspx.

Andert, Edward, Mohammad Khayatian, and Aviral Shrivastava. 2017. "Crossroads: Time-Sensitive Autonomous Intersection Management Technique." In *Proceedings of the 54th Annual Design Automation Conference 2017,* 50. ACM.

Bentjen, Karl C. 2018. *Mitigating the Effects of Cyber Attacks and Human Control in an Autonomous Intersection.* AIR FORCE INSTITUTE OF TECHNOLOGY WRIGHT-PATTERSON AFB OH WRIGHT-PATTERSON ...

Commission, Federal Communications. *Dedicated Short Range Communications (DSRC) Service.* https://www.fcc.gov/wireless/bureau-divisions/mobility-division/dedicated-short-range-communications-dsrc-service.

Csurka, Gabriela, and Florent Perronnin. 2008. "A Simple High Performance Approach to Semantic Segmentation." In *BMVC,* 1–10.

Dedinsky, Rachel, Mohammad Khayatian, Mohammadreza Mehrabian, and Aviral Shrivastava. 2019. "A Dependable Detection Mechanism for Intersection Management of Connected Autonomous Vehicles (Interactive Presentation)." In *Workshop on Autonomous Systems Design (ASD 2019).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

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

Elhenawy, Mohammed, Ahmed A Elbery, Abdallah A Hassan, and Hesham A Rakha. 2015. "An intersection game-theory-based traffic control algorithm in a connected vehicle environment." In *2015 IEEE 18th International Conference on Intelligent Transportation Systems,* 343–347. IEEE.

Geist, Matthieu, Olivier Pietquin, and Gabriel Fricout. 2008. "Bayesian reward filtering." In *European Workshop on Reinforcement Learning,* 96–109. Springer.

Hu, Rui, Diane Larlus, and Gabriela Csurka. 2012. "On the use of regions for semantic image segmentation." In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing,* 51. ACM.

Jin, Qiu, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew Barth. 2012. "Advanced intersection management for connected vehicles using a multi-agent systems approach." In *2012 IEEE Intelligent Vehicles Symposium,* 932–937. IEEE.

———. 2013. "Platoon-based multi-agent intersection management for connected vehicle." In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013),* 1462–1467. IEEE.

Khayatian, Mohammad, Mohammadreza Mehrabian, and Aviral Shrivastava. 2018. "RIM: Robust Intersection Management for Connected Autonomous Vehicles." In *2018 IEEE Real-Time Systems Symposium (RTSS),* 35–44. IEEE.

Lee, Joyoung, and Byungkyu Park. 2012. "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment." *IEEE Transactions on Intelligent Transportation Systems* 13 (1): 81–90.

Legislature, Washington State. *RCW 46.61.425: Minimum Speed Regulation.* https://apps.leg.wa.gov/rcw/default.aspx?cite=46.61.425.

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

Neuendorf, Norbert, and Torsten Bruns. 2004. "The vehicle platoon controller in the decentralised, autonomous intersection management of vehicles." In *Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM'04.* 375–380. Ieee.

Peddoju, VikasChouhan&Sateesh Kumar. n.d. "Packet Monitoring Approach to Prevent DDoS Attack in Cloud Computing." *International Journal of Computer Science and Electrical Engineering (IJCSEE) ISSN,* nos. 2315-4209.

Reference. *What Is the Width and Length of the Average Car?* https://www.reference.com/vehicles/width-length-average-car-9eb7b00283fb1bd8.

Rodriguez Serrano, Jose A, and Diane Larlus. 2013. "Predicting an object location using a global image representation." In *Proceedings of the IEEE International Conference on Computer Vision,* 1729–1736.

Safety, FHWA Office of. *Roadway Safety Data Dashboards.* https://rspcb.safety.fhwa.dot.gov/Dashboard/Default.aspx.

Savic, Vladimir, Elad M Schiller, and Marina Papatriantafilou. 2017. "Aiding Autonomous Vehicles with Fault-tolerant V2V Communication." *arXiv preprint arXiv:1710.00692.*

Zohdy, Ismail H, Raj Kishore Kamalanathsharma, and Hesham Rakha. 2012. "Intersection management for autonomous vehicles using iCACC." In *2012 15th International IEEE conference on intelligent transportation systems,* 1109–1114. IEEE.

APPENDIX A

VEHICLE CONTROLLER PSEUDOCODE

Algorithm 1: Pseudocode for Vehicle Controller

```
1  Signal_ISR (VOA,TOA)   {
2      Receive new [VOA, TOA] from IM
3      Jump to line 29
4  }
5  Timer_ISR (every t seconds)  {
6      if  (emergency_variable is true)
7          Stop_car()
8          Jump to line 21
9      end
10 }
11 while(vehicle doesn't exit intersection)
12     if  (sync line crossed and not Synchronized)
13         Synchronize()
14         if  (Not Synchronized within timeout)
15             Apply $a_{min}$
16             Jump to line 13
17         end
18         Set sync_State to true
19     end
20     if  (transmit line crossed and sync_State is true)
21         V_Data ← Calc-V_Data()
22         Send V_Data to IM
23         Wait for response from IM
24         if  (response timed out)
25             Apply $a_{min}$
26             Jump to line 21
27         else
28             Receive [VOA, TOA] from IM
29             Ref_Traj ← Create-Ref-Traj(VOA,TOA)
30             if (actuation line crossed)
31                 Actuate using [VOA, TOA]
32                 Track Ref_Traj until Goal Pose is reached
33             end
34         end
35     end
```

APPENDIX B

INTERSECTION MANAGER PSEUDOCODE

Algorithm 2: Pseudocode for Intersection Manager

```
global  Detect_Data,  IM_Data, V_Data
Timer_ISR (every t seconds)   {
    Detect_Data ← data from Detection System
}
Trajectory-Reassignment ()   { ▷ Reassign TOA and VOA of next scheduled vehicle
    IM_Data(V_ID+1).TOA ← current time + time to cross remaining distance
    IM_Data(V_ID+1).VOA ← $\frac{remaining\ distance}{TOA-current\ time}$
    Send [VOA,TOA] to V_ID+1
    Schedule() [VOA,TOA] of V_ID+2
    Update IM_Data
}
Broadcast-Alert()   {
    Broadcast an alert to other automated vehicles
    Wait for Rogue Car to go through intersection
    Schedule() all vehicles again
    IM_Data[V_ID] ← Detect_Data[V_ID]
}
Check-Detected ()   {
    Process Detect_Data;
    while  (V_ID in Detect_Data not addressed and detection accuracy is high)
        if  (V_ID position is not in Ref_Traj $\pm \varepsilon$)
            IM_Data[V_ID].markRogue ← true
            Broadcast-Alert()
        else if (V_ID not in IM_Data)
            IM_Data.add(Detect_Data[V_ID])
            IM_Data[V_ID].markRogue ← true
            Broadcast-Alert()
        else if (V_ID achieves TOA+$t_{crossIntersection}$)
            Trajectory-Reassignment()
        end
    end
}
```

```
**Check-Vehicle-Behavior** () {
    V_Data ← *Read*(RequestQueue.*Dequeue*())
    **if** (A request is received but no vehicle is detected)
        Change Policy i.e. MAC blocking
    **end**
    IM_Data[V_ID] ← *Mesh*(V_Data, Detect_Data[V_ID],
      Detection_accuracy)
}
**while**(true)
    **if** (RequestQueue is not empty)
        *Check-Vehicle-Behavior*()
        [VOA, TOA] ← *Trajectory-Assignment*()
        Send [VOA, TOA] to V_ID
    **end**
    *Check-Detected*()
x **end**
```

APPENDIX C

SURVEILLANCE SYSTEM PSEUDOCODE

Algorithm 3: Pseudocode for Surveillance System

```
for(every $\Delta t$ seconds)
    if (new vehicle detected)
        vehicle = new Vehicle(V_ID, position, timestamp)
        Vehicles_Present.Add(vehicle, detection accuracy)
    end
    if (vehicle leaves)
        Vehicles_Present.Delete(vehicle)
    end
    Update positions in Vehicles_Present[ ]
    Update vehicles' detection accuracy
    Send Vehicles_Present[ ] to IM as Detect_Data[]
end
```