Journal of Artificial Intelligence Research 31 (2008) 591-656 Submitted 11/07; published 3/08

A Multiagent Approach to Autonomous Intersection Management

Kurt Dresner KDRESNER@CS.UTEXAS.EDU
Peter Stone PSTONE@CS.UTEXAS.EDU
Department of Computer Sciences, University of Texas at Austin
1 University Station [C0500], Austin, TX 78712 USA

Abstract

Artificial intelligence research is ushering in a new era of sophisticated, mass-market transportation technology. While computers can already fly a passenger jet better than a trained human pilot, people are still faced with the dangerous yet tedious task of driving automobiles. Intelligent Transportation Systems (ITS) is the field that focuses on integrating information technology with vehicles and transportation infrastructure to make transportation safer, cheaper, and more efficient. Recent advances in ITS point to a future in which vehicles themselves handle the vast majority of the driving task. Once autonomous vehicles become popular, autonomous interactions amongst *multiple* vehicles will be possible. Current methods of vehicle coordination, which are all designed to work with human drivers, will be outdated. The bottleneck for roadway efficiency will no longer be the drivers, but rather the mechanism by which those drivers' actions are coordinated. While open-road driving is a well-studied and more-or-less-solved problem, urban traffic scenarios, especially intersections, are much more challenging.

We believe current methods for controlling traffic, specifically at intersections, will not be able to take advantage of the increased sensitivity and precision of autonomous vehicles as compared to human drivers. In this article, we suggest an alternative mechanism for coordinating the movement of autonomous vehicles through intersections. Drivers and intersections in this mechanism are treated as autonomous agents in a multiagent system. In this multiagent system, intersections use a new reservation-based approach built around a detailed communication protocol, which we also present. We demonstrate in simulation that our new mechanism has the potential to significantly outperform current intersection control technology—traffic lights and stop signs. Because our mechanism can emulate a traffic light or stop sign, it subsumes the most popular current methods of intersection control. This article also presents two extensions to the mechanism. The first extension allows the system to control human-driven vehicles in addition to autonomous vehicles. The second gives priority to emergency vehicles without significant cost to civilian vehicles. The mechanism, including both extensions, is implemented and tested in simulation, and we present experimental results that strongly attest to the efficacy of this approach.

1. Introduction

Few concepts, if any, embody the goals and aspirations of artificial intelligence as well as fully autonomous robots. Countless films and stories have been made that focus on a future filled with such humanoid agents which, when not violently overthrowing their human masters, run errands, complete menial tasks, or perform duties that would be too difficult or dangerous for humans. However, machines that sense, think about, and take actions in the real world around us are no longer just the stuff of science fiction and fantasy. Research initiatives like Robocup (Noda, Jacoff, Bredenfeld, & Takahashi, 2006) and the DARPA Grand Challenge (DARPA, 2007) have shown that current AI can produce autonomous, embodied, competent agents for complex tasks like playing soccer or navigating the Mojave Desert, respectively. While certainly no small feat, traversing a barren desert devoid of pedestrians, narrow lanes, and multitudes of other fast-moving vehicles is not a typical daily task for humans. As Gary Bradski, a researcher at Intel Corp. said following the successful completion of the 2005 Grand Challenge by "Stanley," a modified Volkswagen Touareg, "Now we need to teach them how to drive in traffic" (Johnson, 2005). Since then, competitors in the 2007 DARPA *Urban* Challenge took significant strides towards this next milestone, though in the competition cars did not need to sense traffic signs or signals and traffic was relatively sparse—more characteristic of *suburban* than dense urban settings.

In modern urban settings, automobile traffic and collisions lead to endless frustration as well as significant loss of life, property, and productivity. A 2004 study of 85 U.S. cities by researchers at Texas A&M University estimated the annual time spent waiting in traffic to be 46 hours per capita, up from 16 hours in 1982 (Texas Transportation Institute, 2004). Americans burn approximately 5.6 billion gallons of fuel each year simply idling their engines. All told, the annual financial cost of traffic congestion has swollen from $14 billion to more than $63 billion (in 2002 US dollars) in this period. The cost of all the wasted time and fuel due to congestion pales in comparison to the costs associated with automobile collisions. In a 2002 report, the National Highway Traffic Safety Administration (NHTSA) put the annual societal cost of automobile collisions in the U.S. at $230 billion (National Highway Traffic Safety Administration, 2002).

Fully autonomous vehicles may be able to spare us much, if not nearly all of these costs. An autonomous driver agent can much more accurately judge distances and velocities, attentively monitor its surroundings, and react instantly to situations that would leave a (relatively) sluggish human driver helpless. Furthermore, an autonomous driver agent will not get sleepy, impatient, angry, or drunk. Alcohol, speeding, and running red lights are the top three causes of automobile collision fatalities. Autonomous driver agents—properly programmed—would eliminate all three.

A fully autonomous vehicle that will drive in traffic will have to do everything from obeying the speed limit and staying in its lane to detecting and tracking pedestrians or choosing the best route to the mall. While this is certainly a complex task, advances in artificial intelligence, and more specifically, Intelligent Transportation Systems (ITS), suggest that it may soon be a reality (Bishop, 2005). Cars can already be equipped with features of autonomy such as adaptive cruise control, GPS-based route planning (Rogers, Flechter, & Langley, 1999; Schonberg, Ojala, Suomela, Torpo, & Halme, 1995), and autonomous steering (Pomerleau, 1993; Reynolds, 1999). Some current production vehicles even sport these features. DaimlerBenz's Mercedes-Benz S-Class has an adaptive cruise control system that can maintain a safe following distance from the car in front of it, and will apply extra braking power if it determines that the driver is not braking hard enough. Both Toyota and BMW are currently selling vehicles that can parallel park completely autonomously, even finding a space in which to park without any driver input. In 2008, General Motors (GM) plans to release a nearly autonomous vehicle under its European "Opel" brand. The 2008 Opel Vectra will be able to drive itself at speeds up to 60 miles per hour, even in heavy traffic. Using a video camera, lasers, and a lot of processing power, the car will be able to identify traffic signs, curves in the street, lane markings, as well as other vehicles. By the end of the decade, GM hopes to incorporate the system into many other models.

Autonomous vehicles are coming. In this article, we present a well-defined multiagent framework to manage large numbers of autonomous vehicles at intersections. While there still exist many technical hurdles and rigorous safety tests, we show in simulation that this framework may someday dramatically improve the safety and efficiency of our roadways.

1.1 Multiagent Systems

As autonomous vehicles become more and more prevalent, the possibility of autonomous interactions among multiple vehicles becomes more interesting. Multiagent Systems (MAS) is the subfield of AI that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors (Stone & Veloso, 2000). Automobile traffic is a vast multiagent system involving millions of heterogeneous agents: commuters, truck drivers, pedestrians, cyclists, and even traffic-directing police officers. The mechanism that coordinates the behavior of these agents is a complex conglomeration of laws, signs, and signaling systems that vary slightly from state to state and widely from country to country. The mechanism is designed to work closely with the agents—the humans—that populate the multiagent system. Traffic lights leave time in between green lights to allow slower or perhaps impatient drivers to clear intersections. Street signs are colored brightly to make them easier to see and use simple designs to make them easy to understand. Drivers must maintain a sufficient following distance to make up for slow reaction times. Speed limits ensure that humans have enough time to process all the necessary information about the position and velocities of other vehicles. Safety buffers of myriad sorts are built into almost every part of the system to compensate for the limitations of humans.

The first generation of autonomous vehicles will undoubtedly need to work within this system. Processing-intensive vision algorithms will identify and extract semantic information from signs and signals, special subroutines will ensure that the vehicles do not exceed the speed limit, and in the middle of the night, with not another moving vehicle for blocks, an autonomous vehicle will come to a stop at a red light. However, once most vehicles are autonomous and the limitations are eliminated, it will not make sense to use a mechanism designed to control fundamentally different agents—it will be inefficient, both in terms of processing power and getting vehicles to their destinations quickly.

Replacing this soon-to-be-outdated mechanism is inherently a multiagent challenge for several reasons. First, there are no viable single-agent solutions; one computer cannot handle all the vehicles in the world. Second, with vehicles constantly entering and leaving countries, states, cities, and towns, any solution will have to be flexible and distributed. Third, the different agents have separate, and sometimes conflicting objectives. As with human-driven vehicles, autonomous vehicles will act in their own self-interest, attempting to minimize travel time, distance, and fuel use. Other types of agents may aim to maximize social welfare, minimizing these quantities for the average vehicle. Finally, even if a single computer could control a city's worth of traffic, it would be a very sensitive point of failure.

1.2 Intersections

On the open road, automobiles can be more or less completely autonomous. Furthermore, there is little need for more than a simple reactive behavior that keeps the vehicle in the lane, maintains a reasonable distance from other vehicles, and avoids obstacles. Even lane changing can be safely and efficiently accomplished by an autonomous vehicle (Hatipo, Redmill, & Ozguner, 1997). The algorithmic and AI aspects of open-road driving are essentially solved. The problem itself is not too difficult: there are no pedestrians or cyclists and vehicles travel in the same direction at similar velocities; relative movement is smooth and rare.

Intersections are a completely different story: vehicles constantly cross paths, in many different directions. A vehicle approaching an intersection can quickly find itself in a situation in which a collision is unavoidable, even when it has acted optimally. Traffic statistics support the sensitive nature of intersections. Vehicle collisions at intersections account for anywhere between 25% and 45% of all collisions. As intersections make up a very small portion of the roadway, this is a wildly disproportionate amount. Collisions at intersections tend to involve cars traveling in different directions, and thus they frequently result in greater injury and damage. Most modern-day intersections are controlled with traffic lights or stop signs, the former usually reserved for larger, busier intersections. At the busiest of intersections—freeway interchanges—large, extremely expensive cloverleaf junctions are built.

With the vastly improved precision control and sensing that autonomous vehicles will offer, there must be a more efficient and safe way to manage intersections. Imagine the scenario in which an autonomous vehicle stops at a red light in the middle of the night with no other vehicles nearby. At the very least, the vehicle should be able to communicate its presence to the intersection, which can verify that no other vehicles are nearby, and turn the light green for the stopped vehicle. In a more ambitious implementation, the intersection could turn the light green preemptively, obviating the stop altogether. In this article, we go a step further, allowing vehicles to "call ahead" and reserve space-time in the intersection.

The remainder of this article is organized as follows. In Section 2, we describe the problem of autonomous intersection management and a framework with which we will attempt to solve this problem. In Section 3, we describe the implementation of the solution framework. Section 4 presents our experiments and empirical results. In Section 5, we conduct a failure mode analysis of the proposed mechanism. Related work is discussed in Section 6. Section 7 briefly explores some avenues for future research and concludes.

2. Problem Statement and Solution Framework

Automobile traffic is already a huge multiagent system with millions of human driver agents, various signaling and control mechanisms, and a complicated protocol governing the actions of the driver agents, in the form of traffic laws. However, if human drivers are to be replaced by autonomous driving agents, the other elements of the multiagent system should be rethought. Traffic lights, stop signs, and our current traffic laws are all designed with human drivers in mind and fail to take advantage of the increased sensitivity and precision of computerized driver agents. If we want autonomous vehicles to operate with high efficiency and safety, we must design a new way to coordinate them. In this section, we formulate the problem we are trying to solve and present a framework within which we believe the problem can best be solved.

2.1 Desiderata

In designing a mechanism by which traffic is controlled at intersections, we aim to satisfy the following list of properties.

Autonomy Each vehicle should be an autonomous agent. If the entire mechanism were centrally controlled, it would be more susceptible to single-point failure, require massive amounts of computational power, and exert unnecessary control over vehicles in situations where they are perfectly capable of controlling themselves.

Low Communication Complexity By keeping the number of messages and amount of information transmitted to a minimum, the system can afford to put more communication reliability measures in place. Furthermore, each vehicle, as an autonomous agent, may have privacy concerns which should be respected. Keeping the communication complexity low will also make the system more scalable.

Sensor Model Realism Each agent should have access only to sensors that are available with current-day technology. The mechanism should not rely on fictional sensor technology that may never materialize.

Protocol Standardization The mechanism should employ a simple, standardized protocol for communication between agents. Without a standardized protocol, each agent would need to understand the internal workings of every agent with which it interacts. This requirement would forbid the introduction of new agents into the system. An open, standardized protocol would make adoption of the system easier and simpler for private vehicle manufacturers.

Deadlock/Starvation Avoidance Deadlocks and starvation should not occur in the system. Every vehicle approaching an intersection should eventually cross, even if it is better for the rest of the agents to leave that vehicle stranded.

Incremental Deployability The system should be incrementally deployable, in two senses. First, it should be possible to set up selected intersections to use the system, and then slowly expand to other intersections as needed. Second, the system should function even with few or no autonomous vehicles. At each stage of deployment, whether it is an increase in the proportion of autonomous vehicles or the number of equipped intersections, overall performance of the system should improve. At no point should a net disincentive to continue deploying the system exist.

Safety Excepting for gross vehicle malfunction or extraordinary circumstances (e.g. natural disasters), as long as they follow the protocol, vehicles should never collide in the intersection. Note that no stronger guarantee is possible—as with modern mechanisms, a suicidal human driver can always steer a vehicle into oncoming traffic. Furthermore, the system should be safe in the event of total communication failure. If messages are dropped or corrupted, the safety of the system should not be compromised. It is impossible to prevent all negative effects due to communication failures, but those negative effects should be isolated to efficiency. If a message gets dropped, it can make someone arrive 10 seconds later at their destination, but it should not cause a collision. In the rare but unpreventable case of gross vehicle malfunction, the system should react and attempt to minimize damage and casualties.

Efficiency Vehicles should get across the intersection and on their way in as little time as possible. To quantify efficiency, we introduce *delay*, defined as the amount of additional travel time incurred by the vehicle as the result of passing through the intersection.

2.2 The Reservation Idea

Of the desiderata, modern-day traffic lights and stop signs completely satisfy all but the last one. While many accidents take place at intersections governed by traffic lights, these accidents are rarely, if ever, the fault of the traffic light system itself, but rather that of the human drivers. However, as we will show, traffic lights and stop signs are terribly inefficient. Not only do vehicles traversing intersections equipped with these mechanisms experience large delays, but the intersections themselves can only manage a somewhat limited amount of traffic. Any stretch of open road can accommodate a certain level of traffic at a given velocity. The capacity of an intersection involving a road is trivially bounded above by the capacity of the road. As we will also show, the capacity of traffic lights and stop signs is much less than that of the roads that feed into them. The aim of this research is to create an intersection control mechanism that exceeds the efficiency of traffic lights and stop signs, while maintaining each of the other desiderata.

With the desiderata in mind, we developed a multiagent approach to direct vehicles through intersections more efficiently. In this approach, computer programs called *driver agents* control the vehicles, while an *arbiter* agent called an *intersection manager* is placed at each intersection. The driver agents "call ahead" and attempt to reserve a block of space-time in the intersection. The intersection manager decides whether to grant or reject requested reservations according to an *intersection control policy.* Figure 1 shows one interaction between a driver agent and an intersection manager. The system functions analogously to a human attempting to make a reservation at a hotel—the potential guest specifies when he or she will be arriving, how much space is required, and how long the stay will be; the human reservation agent determines whether or not to grant the reservation, according to the hotel's reservation policy. Just as the guest does not need to understand the hotel's decision process, the driver agents should not require any knowledge of the intersection control policy used by the intersection manager.

When a vehicle approaches the intersection, the vehicle's driver agent transmits a reservation request, which includes parameters such as time of arrival, velocity of arrival, as well as vehicle characteristics like size and acceleration/deceleration capabilities, to the intersection manager. The intersection manager then passes this information to the policy, which determines whether or not it is safe for the vehicle to cross the intersection. If the policy deems it to be safe, the intersection manager responds to the driver agent with a message indicating the reservation has been accepted and including any supplemental restrictions the driver must observe in order to guarantee the safety of the traversal. Otherwise, the intersection manager sends a message indicating that the reservation request has been rejected, possibly including the grounds for rejection. In addition to confirming or rejecting

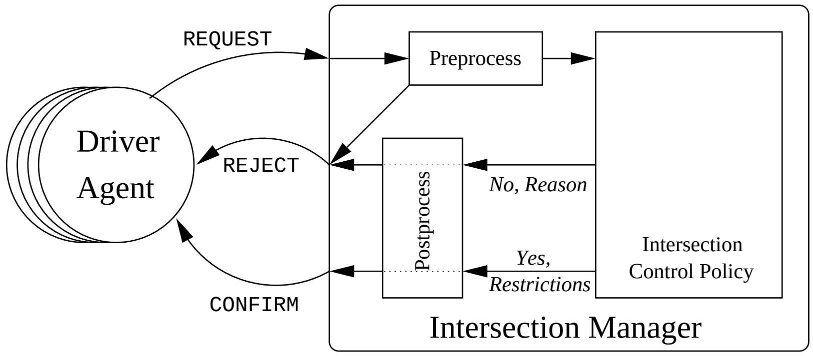


Figure 1: One of the driver agents attempts to make a reservation. The intersection manager responds based on the decision of an intersection control policy.

the request, the intersection manager may respond with a counter-offer. The driver agent may not pilot the vehicle into the intersection without a reservation. Even with a reservation, a driver agent may only proceed through the intersection according to the parameters and restrictions associated with the reservation. For the sake of brevity, we may refer to a vehicle having or obtaining a reservation, rather than specifically stating that the driver agent of that vehicle has or obtains a reservation.

3. Building The System

This section describes the realization of the reservation idea as an implemented algorithm. This process involved developing a simulator in which to run the algorithm, as well as creating behaviors for each of the agents and a protocol by which they can communicate.

3.1 Custom Simulator

In order to empirically evaluate the reservation idea, we built a custom time-based simulator. The simulator models an area that is 250 m $\times$ 250 m. The intersection is located at the center of that area, and its size is determined by the number of lanes traveling in each direction, which is variable. We assume throughout that vehicles drive on the right side of the road, however this assumption is not required for the system to work properly. Figure 2 shows a screenshot of the simulator's graphical display. During each time step, the simulator:

1. Probabilistically spawns new vehicles
2. Provides sensor input to all vehicles
3. Allows all driver agents to act
4. Updates the position of all vehicles according to the physical model
5. Removes any vehicles outside the simulated area that have completed their journey

3.1.1 Vehicles

Vehicles in the simulator have the following properties:

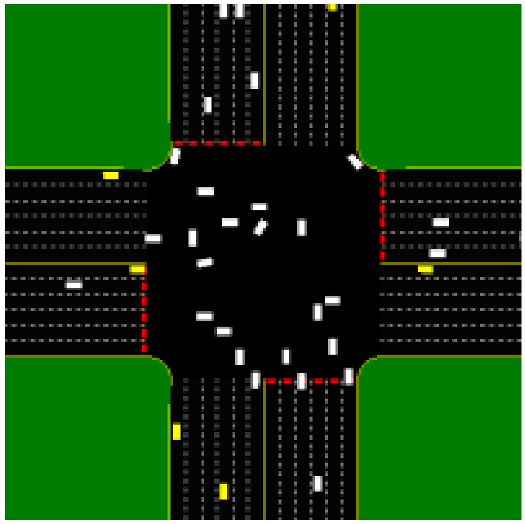


Figure 2: A screenshot of the simulator in action.

- Vehicle Identification Number (VIN)
- Length
- Width
- Distance from front of vehicle to front axle
- Distance from front of vehicle to rear axle
- Maximum velocity
- Maximum acceleration
- Minimum acceleration
- Maximum steering angle
- Sensor range

and the following state variables:

- Position
- Velocity
- Heading
- Acceleration
- Steering angle

The driver agent assigned to pilot the vehicle may access each of these quantities, with or without noise, depending on the configuration of the simulator. The driver agent may also access several simulated external sensors: a list of all vehicles within the sensor range, and a simplified laser range finder. A detailed description of the simplified laser range finder can be found in Appendix A.

The *steering angle* is the angle of the front wheels with respect to the vehicle. This angle can be changed by the driver agent, but the simulator limits the rate at which it can be changed. This limitation simulates the fact that even a computerized driver cannot move the steering wheel infinitely fast. By introducing this limitation, we more accurately approximate vehicle turning, including some of the more dangerous aspects. If the driver cannot turn the wheels instantaneously, it must ensure that it does not drive around corners at too high a velocity—it may not be able to straighten out quickly enough and wind up veering off the road instead.

The constants representing the distance from the front of the vehicle to the front and rear axles allow more accurate simulation of vehicle turning. Specifically, they allow the simulator to treat different styles of vehicle differently. The distance between the front and rear axles is known as the *wheelbase*. Vehicles with shorter wheelbases can turn more sharply than those with longer wheelbases—if the simulator is to accurately model turning, it needs access to these important parameters. Furthermore, a vehicle with a long hood will turn differently than a vehicle whose front wheels are located nearer to the front of the vehicle.

3.1.2 Lanes

Lanes in our system consist of a directed line segment, a width, left and right borders that vehicles may or may not be permitted to cross, and references to which lanes, if any, border on the right and left side. In a real-life implementation, this would be a software construct the vehicles and driver agents would use to perform lane following and changing. If a vehicle wants to change lanes to the left or right, it must first establish that the vehicle is allowed to cross the border between the lanes, after which it can feed its lane-following algorithm the reference to the desired lane.

3.1.3 Physical Model

At each time step, the simulator must update the position of every vehicle. Because we model only planar vehicle kinematics and not dynamics, we must make a few assumptions. First, we assume that vehicles do not skid on the road. Second, we assume that vehicles move according to the following differential equations for non-holonomic motion:

$$\frac{\partial x}{\partial t} = v \cdot \cos(\phi)$$
$$\frac{\partial y}{\partial t} = v \cdot \sin(\phi)$$
$$\frac{\partial \phi}{\partial t} = v \cdot \frac{\tan \psi}{L}$$

In these equations, x, y, and ϕ describe the vehicle's position and orientation, v represents the vehicle's velocity, ψ describes the vehicle's steering angle, and L is the vehicle's wheelbase. We solve these equations holding v and ψ constant for each time step.

3.1.4 Measuring Delay

In Section 2.1, we introduced *delay*—the increase in travel time for a vehicle due to the presence of the intersection. In the simulation, this is measured by first assuming that on the open road, a vehicle can maintain its velocity at the speed limit. Each vehicle is timestamped when it enters the simulation and keeps track of how far it has traveled. When the vehicle is removed from simulation, its total delay is calculated as the difference between how long it actually took to travel as far as it did and how long it would take were the vehicle to travel at the speed limit for the entire journey. By this measure, a zero delay is not possible when the vehicle is turning, as it needs to slow down in order to safely make the turn. In practice, we compare the delays of all vehicles to delays using a policy that allows vehicles through the intersection unhindered, which will also be non-zero if any of the vehicles turn or if the road is congested. In this way, we can quantify the effect of the intersection on the vehicle, both directly (not being able to go through the intersection because requests were rejected) and indirectly (having to decelerate because another vehicle cannot get through).

3.2 Communication Protocol

This section presents a detailed communication protocol by which vehicles and intersections can coordinate their behavior. The protocol as presented here offers three major benefits:

- All information between the agents goes through one monitorable channel, which makes reasoning about the communication straightforward.
- By limiting the interactions of the agents to a few message types, we can ensure that no agent has an unrealistic amount of control over another.
- The agents have a way to communicate that is identical for any intersection management policy or driver agent policy. Thus, a vehicle can cross an intersection without having any idea what policy the intersection manager is using—it simply sends and receives messages and obeys the rules.

The protocol consists of several message types for each kind of agent, as well as some rules governing when the messages should be sent and what sorts of guarantees accompany them. Driver agents can send Request, Change-Request, Cancel, and Done messages. Request and Change-Request are used when the driver agent wants to make a reservation or change an existing reservation, respectively. Both types of request message include all the relevant properties of the vehicle. Driver agents send a Cancel message when they want to cancel an existing reservation. When a vehicle has successfully crossed the intersection, its driver agent sends a Done message to the intersection manager. Both the Cancel and Done messages include the VIN of the vehicle, as well as an identifier for the reservation to be cancelled or reported as complete.

Intersection managers can send Confirm, Reject, and Acknowledge messages, as well as a special Emergency-Stop message, which is only used when the intersection manager detects a major problem in the intersection (see Section 5). Confirm is sent when the intersection manager approves a Request or Change-Request message. It includes information describing the reservation—a unique identifier for the reservation, a start time, a start lane, a departure lane (which will be identical to the start lane unless the vehicle is turning), and a list of constraints for the vehicle's acceleration while it is in the intersection. The Reject message is used to reject either a Request or Change-Request message. The intersection sends an Acknowledge message in response to Cancel and Done messages sent by the vehicles. A more detailed specification of the protocol including full syntax and semantics can be found in Appendix B.

3.2.1 Message Corruption and Loss

We assume that messages can be digitally signed, such that the possibility of an undetected message corruption is acceptably small. The protocol is designed specifically to be robust to message loss. If a message is sent but not received—or deemed corrupted—the worst thing that can happen is additional delay. No collisions can occur due to lost messages. When a vehicle makes a reservation request, it does not assume the space is reserved until it receives a confirmation from the intersection manager. If a Request message is dropped, no Confirm message will follow. If a Confirm or Reject message is dropped, the vehicle will simply try again—it won't assume that it has a valid reservation.

3.2.2 Enabling Policy Switching

The protocol hides the implementation of the policy from the driver agents — they have no idea how the intersection manager is making its decisions, they are just guaranteed that if they follow them, they will be safe. Thus, there are no stipulations that the policy must remain fixed. An intersection manager could use one policy one moment and then switch to a more appropriate policy later, provided it can still guarantee that vehicles following the protocol make it safely across the intersection.

3.2.3 Intersection Manager

The intersection manager acts as a stable communication interface between the driver agents and the intersection control policy and therefore does not contain a lot of functionality. However, regardless of how the policy makes its decision, the intersection manager must present the same interface to the driver agents. The general intersection manager algorithm is shown in Algorithm 1. In it, Cancel messages and Done messages are treated almost identically—when a Done message is received, the intersection manager knows that the policy can erase any information about the related reservation. However, the Done message also may contain information that is useful to the intersection manager and policy. For example, when a vehicle sends a Done message, it could include the delay it experienced crossing the intersection, providing the intersection manager with a sort of reward signal, by which it can judge its performance.

3.3 Driver Agent

The vast majority of this research focuses on how to make a better intersection manager and control policy. These parts are designed to work with any driver agent that follows the protocol. However, for testing purposes, a driver agent implementation is required. Despite the fact that a lot of work went into the driver agent (it is probably the most intricate part of the system), it is not the focus of this article. We refer the interested reader to Appendix C, which explains the driver agent in detail. In brief, the driver agent estimates the time and velocity at which it will reach the intersection, and requests an appropriate reservation. If granted a reservation, it attempts to arrive on schedule. If it determines that it is unable to keep the reservation, it cancels the reservation. If it believes it will be substantially early, it attempts to change to an earlier reservation. If it is unable to get a reservation, it decelerates (down to a minimum velocity) and requests again. It does not

Algorithm 1 The intersection manager algorithm. Vehicle V sends a message to the intersection manager, which responds according to policy P.

```
loop
  receive message from V
  if message type is Request then
    process request for new reservation with P
    if P accepts the request then
      send Confirm message to V containing the reservation returned by P
    else
      send Reject message to V
  else if message type is Change-Request then
    process request for change of reservation with P
    if P accepts the request then
      send Confirm message to V containing the reservation returned by P
    else
      send Reject message to V
  else if message type is Cancel then
    process cancel with P
    send Acknowledge message to V
  else if message type is Done then
    record any statistics supplied in message
    process cancel with P
    send Acknowledge message to V
```

enter the intersection without a reservation. On the open road, the driver agent employs a simple lane-following algorithm, and maintains a following distance of one second between its vehicle and the vehicle in front of it.

3.4 The FCFS Policy

To this point, we've described the substrate infrastructure that enables our research. The remainder of Section 3 introduces the core contribution of this article and the main payoff for creating this infrastructure, namely an intersection control policy that enables fine-grained coordination of vehicles at intersections, and a subsequent dramatic decrease in delays.

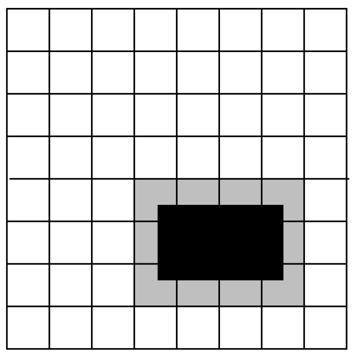
While the intersection manager communicates directly with the driver agents, the intersection control policy is the "brains" behind the operation. Here we describe an intersection control policy created from the reservation idea as discussed in Section 2.2. Because of the "First Come, First Served" nature of the policy, we name this policy FCFS. The main part of the policy—the request processing—is shown in Algorithm 2.

Recall that FCFS enables a car to reserve in advance the space-time it needs to cross the intersection. Planning ahead allows vehicles coming from all directions to traverse the intersection simultaneously with minimal delay. The policy works as follows:

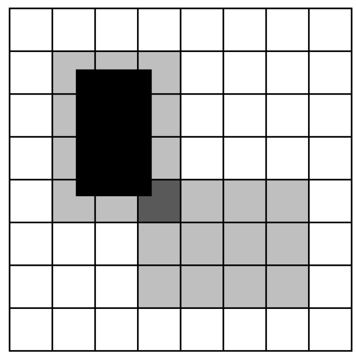
- The intersection is divided into an $n \times n$ grid of *reservation tiles*, where n is the *granularity* of the policy.

- Upon receiving the reservation parameters from an approaching driver agent, the policy runs an *internal simulation* of the trajectory of the vehicle across the intersection using these parameters.
- At each time step of the internal simulation, the policy determines which reservation tiles will be occupied by the vehicle
- If at any time during the simulation the requesting vehicle occupies a reservation tile that is already reserved by another vehicle, the policy rejects the request. Otherwise, the policy accepts the reservation and reserves the appropriate tiles for the times they will be required.

Figure 3 shows a graphical depiction of the concept behind the FCFS policy.



(a) Successful



(b) Rejected

Figure 3: The internal simulation of a granularity-8 FCFS policy. The black rectangles represent vehicles, and the shaded tiles are tiles that are currently reserved. In 3(a), a vehicle's request is accepted, and the intersection reserves a set of tiles at time t. In 3(b), a second vehicle's request is rejected because during the simulation of its trajectory, the policy determines that it requires a tile (darkly shaded) already reserved by the first vehicle at time t.

While the concept behind FCFS is sound, it requires some modifications before it will work reliably, safely, and efficiently—even in simulation. In the remainder of this section, we present these modifications, most of which were created in response to early experimental results documented in Section 4.

3.4.1 Determining the Outbound Lane

In our first implementation of the reservation system, vehicles were capable of traveling only in straight lines. Once we allowed vehicles to turn, it became apparent that the driver agents should not determine which lane they use to exit the intersection. Instead, the intersection manager, which has more information about the intersection, makes this decision. Driver agents indicate in their request message which way they intend to turn, or for more complicated intersections, which direction they intend to go. The intersection control policy then decides in which outbound lane to place the vehicle. For all experiments documented in this article, the FCFS policy chooses the most natural lane: for left and right turns, it chooses the nearest lane, whereas for vehicles that are not going to turn, it chooses the lane in which they are planning to arrive at the intersection. However, a policy could behave differently if configured to do so. For example, the policy can create a priority list of outbound lanes based on the inbound lane, and then run internal simulations using each of these lanes until it found an acceptable configuration. For turning vehicles, this list would be the set of outbound lanes in the correct direction, sorted from nearest to farthest. For vehicles not turning, it would "spiral" out from the lane in which they arrive—first the arrival lane, then the lane to the left, then the lane to the right, then two lanes to the left, and so forth. In this manner, a vehicle that might otherwise have had its request rejected can obtain a reservation for a different path through the intersection.

3.4.2 Acceleration In The Intersection

Given a set of reservation parameters, there are an infinite number of possible trajectories a vehicle can take, if it is allowed to accelerate in the intersection. This is because at each time step, the driver agent could set its vehicle's acceleration to any value within the limits of the vehicle's capabilities. Depending on the trajectory, the intersection manager may or may not be able to grant the reservation—one set of accelerations may cause it to collide with another vehicle, while a second set might let the vehicle through safely. For this reason, acceleration in the intersection must be constrained by the intersection control policy. Allowing driver agents to decide their own acceleration within the intersection would require the policy to be much more conservative in estimating vehicle trajectories, thereby reducing efficiency substantially. Instead, it is the responsibility of the intersection control policy to choose a safe and efficient acceleration schedule and include it in the Confirm message, if the driver agent's request is accepted.

Choosing the best acceleration schedule for the requesting vehicle, or on an even more basic level, finding a schedule for which the intersection manager can grant the reservation, is a difficult challenge for the intersection control policy. Our initial solution was to allow no acceleration within the intersection; driver agents were required to maintain the same velocity throughout the entire trajectory. This approach had several major flaws, the most severe of which was causing a deadlock scenario as vehicles traversed the intersection more and more slowly, unable to recover from the slightest decelerations. This scenario is described in much more detail in Section 4.2.

The FCFS policy, as we have implemented it still takes a fairly straightforward approach to the problem of determining acceleration schedules for reservation requests. It first attempts a trajectory in which the requesting vehicle accelerates as quickly as possible to maximum velocity as soon as it enters the intersection. If it cannot grant a reservation based on that trajectory, it tries one in which the requesting vehicle maintains a constant velocity throughout the intersection. If neither work, it rejects the request. Furthermore, if the request indicates that the vehicle will arrive at a sufficiently slow velocity—in our case 10 m/s—it does not grant a fixed-velocity reservation. Were it to grant arbitrarily slow reservations, a vehicle could use an excessively large amount of space-time in the intersection, causing other vehicles undue delay. By enforcing a minimum velocity for fixed-velocity reservations, the policy ensures that no vehicle will spend too long in the intersection. While more complex solutions exist, this solution is good for several reasons. First, it is compu-

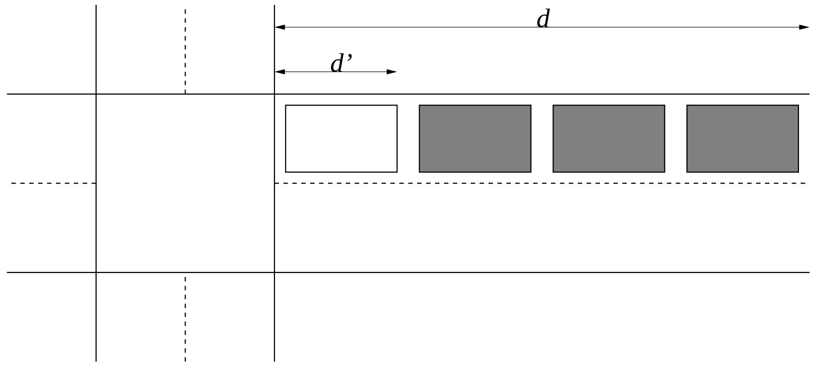


Figure 4: Several vehicles are waiting at the intersection. With a reservation distance of d, the front (white) vehicle is incapable of obtaining a reservation because the vehicles behind it (shaded) hold conflicting reservations. Once the white vehicle's request is rejected, the reservation distance is decreased to d'. Once the shaded vehicles cancel their reservations, the white vehicle can obtain a reservation uncontested.

tationally tractable: the policy runs at most two internal simulations per request. Second, it allows vehicles that are stopped or moving very slowly at the intersection to clear the intersection in a timely manner once they get a reservation. Third, it eliminates the deadlock scenario presented in Section 4.2 by allowing vehicles to recover from decelerating when they cannot obtain a reservation; even a vehicle that comes to a full stop at the intersection can accelerate back up to a reasonable velocity as it crosses the intersection.

3.4.3 Reservation Distance

Allowing accelerations in the intersection helps eliminate deadlocks, but other problems arose in our prototype implementation that significantly impaired the performance of the system. Frequently, a lane of traffic would become congested when many vehicles were spawned in that lane. Even when the simulator stopped spawning vehicles in that lane, the lane would remain congested. The problem is that FCFS, as first described, does nothing to control how vehicles in the same lane are alloted reservations. At best, the frontmost vehicle will get a reservation and make it through the intersection unhindered. However, this is often not the case. Sometimes the vehicle in front cannot obtain a reservation (due to congestion), and must decelerate. As shown in Figure 4, driver agents in vehicles further back may expect to accelerate soon and successfully reserve space-time in the intersection that the frontmost vehicle needs. While all vehicles will *eventually* make it through (a vehicle might get a reservation immediately after vehicles behind it cancel), this process can repeat many times before the frontmost vehicle gets a reservation. In the worst scenarios, a single vehicle can continue for quite some time to obtain reservations that prevent the front car from crossing the intersection.

If we could maintain the invariant that vehicles do not get reservations unless all cars in front of them (in their lane) have reservations, this scenario could be avoided entirely. A simple way to enforce this would be to insist that no vehicle can get a reservation unless the vehicle in front of it already has one. Unfortunately, there is no way to strictly enforce this: vehicles do not communicate their positions (and even if they did, they could be untruthful).

However, because the vehicles communicate the time at which they plan to arrive at the intersection, as well as what their velocity will be when they get there (quantities which the vehicles have no incentive to misrepresent), it is possible to approximate a vehicle's distance from the intersection, given a reservation request by that vehicle. We approximate this distance, which we call the *reservation distance*, as $v_a(t_a - t)$, where v_a is the proposed arrival velocity of the vehicle (at the intersection), t_a is the proposed arrival time of the vehicle, and t is the current time. This approximation assumes the vehicle is maintaining a constant velocity.

The policy uses the approximation as follows. For each lane i, the policy has a variable d_i, initialized to ∞. For each reservation request r in lane i, the policy computes the reservation distance, $d(r)$. If $d(r) > d_i$, r is rejected. If, on the other hand, $d(r) \leq d_i$, r is processed as normal. If r is rejected after being processed as normal, $d_i \leftarrow \min(d_i, d(r))$. Otherwise, $d_i \leftarrow \infty$.

While this does not guarantee that vehicles only get reservations if all vehicles in front of them already have reservations, it makes it much more likely. Two properties make the approximation particularly well-suited to this problem. First, if a vehicle is stopped at the intersection, its reservation distance will be approximated as zero. This means that no vehicle behind it will be granted a reservation before it is—no smaller reservation distance is possible. Furthermore, because the reservation distance is the product of the arrival velocity and the time until the vehicle arrives, as vehicles approach the intersection and slow down, the reservation distance gets smaller and more accurate. Thus, vehicles most susceptible to the problem described in Figure 4 are the most likely to be protected against it. The second property is that because the estimate uses the arrival velocity of the vehicle, it overestimates the distance of vehicles expecting to accelerate significantly before reaching the intersection. It is this expectation that causes driver agents to reserve space-time that is needed by vehicles in front of them. Note also that this heuristic only works within a single lane—each lane keeps track of its own reservation distance.

In the example of Figure 4, the white vehicle's rejected reservation request would shorten the maximum allowed reservation distance for its lane. This, in turn, would cause future requests by the shaded vehicles to be immediately rejected, giving the white vehicle exclusive access (within the lane) to the reservation mechanism. Once the white vehicle secured a reservation, the maximum allowed reservation distance would be reset to the maximum, and all vehicles would once again have equal priority.

3.4.4 Timeouts

Once a driver agent's reservation request is rejected, that driver agent may immediately make a new request. Unless the new request is significantly different, it will most likely be rejected as well. With the exception of the request made immediately after the first rejected request, a driver agent's estimate of its arrival at the intersection is not likely to change much in the instant between consecutive requests. Eventually, after the vehicle has decelerated enough or the driver agents with conflicting reservations have canceled, the vehicle will obtain a reservation and make it through the intersection. From the standpoint of the intersection manager, each of the requests before the successful one are wasted effort. While our policy runs at most two internal simulations per request, those simulations may be computationally expensive, especially if the FCFS policy has a high granularity. Furthermore, if each rejected vehicle makes a request at every possible instant, the work can add up very quickly.

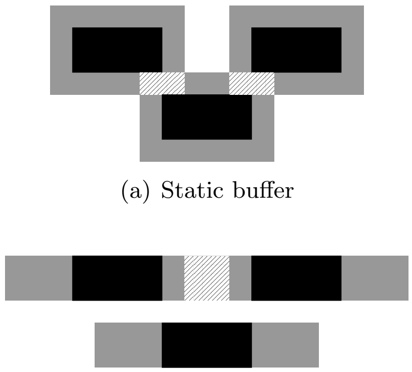
In order to keep the required amount of computation down and discourage driver agents from overloading the intersection manager with requests, the policy employs a system of *timeouts*. Once a driver agent's request is rejected, subsequent requests will not be considered until a period of time (determined by the reservation parameters) has elapsed. When rejecting a request, the policy includes in the rejection message the time after which it will consider further requests from the driver agent. In our implementation, this time is equal to $t + min(\frac{1}{2}, \frac{(t_a - t)}{2})$, where t is the current time and t_a is the time of arrival in the request message. This process serves two purposes. First, it dramatically reduces the amount of computation the policy needs to do, because the intersection manager receives fewer requests. Vehicles may not obtain reservations at the earliest moment possible, but the computational savings are more than worth it. Second, it gives preference to vehicles that will enter the intersection sooner. If a vehicle is stopped at the intersection, it can send requests as quickly as it wishes, giving it the best chance of getting a reservation approved. A vehicle farther away, however, may have to wait the full half-second before attempting to make another reservation. As a vehicle approaches the intersection, if it is unable to procure a reservation, the frequency of opportunities to send reservation requests increases. In practice, timeouts significantly improve the performance of the system, allowing it to handle much higher traffic loads while avoiding backups.

3.4.5 Buffers: Static vs. Time

In any system involving physical robots, noise in sensor readings and errors in actuators will inevitably manifest themselves. Even in simulation, artifacts resulting from the discretization of time are enough to weaken the reservation tiles' guarantees of exclusivity. In the intersection, where vehicles move at high speeds in all different directions, these potential sources of calamity cannot be ignored. For example, what happens when a driver agent realizes that it will not make its reservation exactly on time, close enough to the intersection that it is not possible to stop before entering the intersection? Some sort of safety buffer is required. Two types of buffers are most natural: static buffers and time buffers.

Static buffers—buffers whose size is constant—certainly suffice for safety purposes. If the intersection manager assumes each vehicle is ten times as large in each dimension, certainly no vehicle should even get close to another vehicle. However, this defeats the point of the intersection manager, which is to leverage the increased precision of autonomous vehicles. Furthermore, a static buffer does not take into account the direction of motion of the vehicle. Two vehicles whose paths would never intersect may begin to interfere with one another's reservation process if a large static buffer is used, as in Figure 5(a).

Time buffers, on the other hand, do take into account the motion of the vehicles. If the intersection manager instead assumes that the vehicle might be early or late, the actual area restricted by this buffer will shrink and grow with the vehicle's velocity, and only in the



(a) Static buffer (b) Time buffer, low velocity

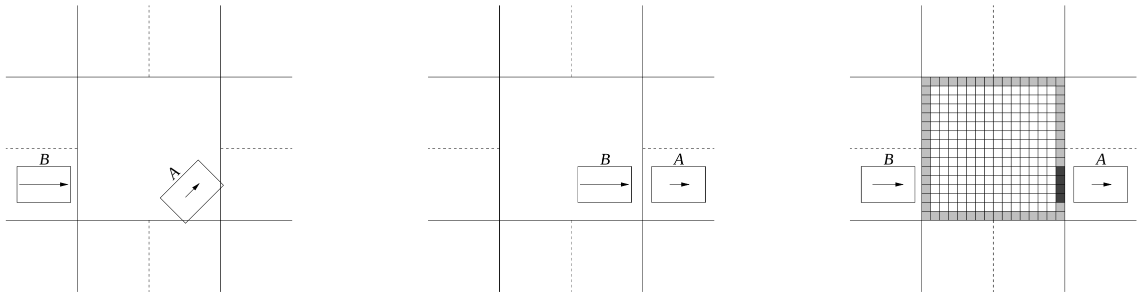
(c) Time buffer, high velocity (d) Hybrid buffer

Figure 5: Various styles of buffers designed to cope with sensor noise and actuator errors. The hatched areas show where buffers would cause reservation conflicts: only one of each pair of conflicting vehicles would be granted a reservation.

direction of movement. Figures 5(b) and 5(c) show how the buffer scales with the speed of the vehicle. Thus, if two vehicles are traveling along parallel lines, the time buffers for those vehicles should not interfere unless those vehicles could potentially collide (they are in the same lane or the lanes are too close together for the vehicles' width). Alone, time buffers are not sufficient to guarantee safety — a small error in lateral positioning (orthogonal to the direction of motion) may still cause a collision. Figure 5(d) shows the best solution: a *hybrid buffer*. The hybrid buffer has a time buffer that scales with velocity, as well as a small static buffer that protects against lateral positioning errors and serves as a minimum buffer for slow-moving vehicles.

3.4.6 Edge Tiles

When driving on the open road, vehicles must maintain a reasonable following interval (usually measured as an amount of time) between one another. If a vehicle decelerates suddenly, it puts the vehicle behind it in a dangerous situation—if the rear vehicle doesn't react quickly enough, it may collide with the front vehicle. In the intersection, following intervals are not very practical, because vehicles are traveling in many different directions. Vehicles in the intersection cannot react normally to their sensor readings, because the intersection manager may orchestrate some "close calls" that would look like a potential collision to a vehicle operating in "open road" mode. Instead, the vehicles trust the constraints given to them by the intersection manager. This does not pose a problem in the intersection, but when a vehicle exits the intersection, it may enounter a vehicle that also just left the intersection, but at a much slower velocity. As shown in Figures 6(a) and 6(b), this may lead to an unavoidable collision, with the later vehicle being unable to stop quickly enough. Even with autonomous vehicles, which can react almost instantaneously, some amount of following interval is required for vehicles leaving the intersection.



(a) A turns right in front of B.

(b) B cannot stop in time.

(c) B must slow down preemptively.

Figure 6: Edge tiles prevent collisions after vehicles leave the intersection. In 6(a), vehicle A turns in front of vehicle B, traveling slowly because it is making a right turn. In 6(b), vehicle B gets through the intersection without incident, but finds that once it leaves the intersection, it cannot stop before colliding with vehicle A. The extra buffers on edge tiles, as shown in 6(c), prevent vehicle B from obtaining a reservation which would cause it to exit the intersection too close to vehicle A. The shaded tiles are edge tiles, while the darkly shaded tiles are the specific tiles that would prevent the collision in 6(a) and 6(b).

A first-cut solution to this problem is simply to increase the time buffers on all reservation tiles to the desired following interval. Thus, if vehicles require a following interval of one second when exiting the intersection, then no vehicle will be able to reserve a tile within one second of another vehicle. This ensures that vehicles leaving the intersection in the same lane will not exit within one second of each other, and there will be a gap of at least one second between the vehicles. Unfortunately, this wreaks havoc with FCFS's ability to conduct vehicles efficiently through the intersection. The "close calls" from which the system gets its efficiency advantages will no longer be possible.

Instead, we divide the reservation tiles into two groups. *Internal tiles* are tiles that are surrounded on all sides by other reservation tiles. *Edge tiles*, which are shown shaded in Figure 6(c), are tiles that abut the intersection. At sufficiently high granularities, edge tiles are a relatively small fraction of the total number of tiles. It is only on these tiles that we increase the time buffer to the desired following interval. Because (at sufficiently high granularities) only vehicles leaving by the same lane will require the same edge tiles, this modification enforces the desired following intervals without otherwise preventing the intersection from exploiting its ability to interleave vehicles closely.

3.5 Other Policies

Because of the layer of abstraction provided by the protocol, the intersection manager can work in an emulation mode, imitating modern-day control mechanisms, such as the stop sign and traffic light. Here we briefly explain the implementation of two intersection control policies designed to mimic these mechanisms.

Algorithm 2 FCFS's request processing algorithm. FCFS has persistent state variables: *tiles*, a map from tiles and times to vehicles, *reservations*, a map from vehicles to sets of tiles, and *timeouts*, a map from vehicles to times.

1: $t_c \leftarrow$ the current time
2: **if** $timeouts$[vehicle id] $< t_c$ **then**
3:    reject the request
4: $t_a \leftarrow$ proposed arrival time
5: $timeouts$[vehicle id] $\leftarrow t_c + min(0.5, (t_a - t_c)/2)$
6: **for** $acceleration$ in {true, false} **do**
7:    $tile\_times \leftarrow \{\}$
8:    $t \leftarrow t_a$
9:    $V \leftarrow$ temporary vehicle initialized according to reservation parameters
10:    **while** V is in the intersection **do**
11:       $S \leftarrow$ tiles occupied by V and V's static buffer at time t
12:       $tile\_times \leftarrow tile\_times \cup \{(t, S)\}$
13:       **for all** $s \in S$ **do**
14:          **if** s is an edge tile **then**
15:             $buf \leftarrow$ edge tile buffer
16:          **else**
17:             $buf \leftarrow$ internal tile buffer
18:          **for** $i = -buf$ to buf **do**
19:             **if** $tiles[s, t + i]$ is reserved by another vehicle **then**
20:                **if** $acceleration$ **then**
21:                   goto line 29
22:                **else**
23:                   reject the request
24:       $t \leftarrow t +$ time step
25:       move V according to physical model
26:       **if** $acceleration$ **then**
27:          increase V's velocity by V's maximum acceleration
28:    break
29:
30: **if** request is a change **then**
31:    $old\_tile\_times \leftarrow reservations$[vehicle id]
32:    **for all** $(t_i, S_i) \in old\_tile\_times$ **do**
33:       **for all** $s \in S_i$ **do**
34:          clear reserved status of $tiles[s, t_i]$
35: **for all** $(t_i, S_i) \in tile\_times$ **do**
36:    **for all** $s \in S_i$ **do**
37:       $tiles[s, t_i] \leftarrow$ vehicle id
38: $reservations$[vehicle id] $\leftarrow tile\_times$
39: accept request, return reservation constraints (incl. accelerations)

Stop-Sign Stop signs are traditionally used at intersections with very light traffic. While they are much more cost-effective and reliable, they cannot provide the throughput and efficiency of a traffic light. Thus, there would never be a reason for our system to emulate a stop sign, however we include a description for completeness.

Stop-Sign is exactly like FCFS, except that it only accepts reservations from vehicles that are stopped at the intersection. Any other reservation requests are rejected with a message indicating the vehicle must stop at the intersection. The intersection determines whether a vehicle is stopped at the intersection by examining the difference between the current time and the arrival time in the request message.

Traffic-Light When the Traffic-Light policy receives a reservation request message, it calculates the next time after the proposed arrival time that the light for sending vehicle's lane will be green. It then responds with a confirmation message that reflects this information. Because confirmation messages have maximum tolerable errors associated with them, the intersection manager uses these errors to encode the beginning and end of the green light period.

3.6 Compatibility With Human Drivers

While an intersection control mechanism for autonomous vehicles will someday be very useful, there will always be people who enjoy driving. Additionally, there will be a fairly long transitional period between the current situation (all human drivers) and one in which human drivers are a rarity. Even if switching to a system comprised solely of autonomous vehicles were possible, pedestrians and cyclists must also be able to traverse intersections in a controlled and safe manner. For this reason, it is necessary to create intersection control policies that are aware of and able to accommodate humans, whether they are on a bicycle, walking to the corner store, or driving a "classic" car for entertainment purposes. In this section we explain how we have extended the FCFS policy and the reservation framework to incorporate human drivers. In order to accommodate human drivers, a control policy must be able to direct both human and autonomous vehicles, while coordinating them, despite having much less control and information regarding where and when the human drivers will be. The main concept behind our extension is the assumption that there is a human-driven vehicle anywhere one *could* be. While this may be less efficient than an approach which attempts to more precisely model human behavior, it is guaranteed to be safe, one of the desiderata on which we are unwilling to compromise. Adding pedestrians and cyclists follows naturally, and we give brief descriptions of how this would differ from the extensions for human drivers.

Compatibility with human drivers offers more than the ability to handle the occasional human driver once the levels of human drivers in everyday traffic reaches a steady state. It will also help facilitate the transition from the current standard—all human-driven vehicles — to this steady state, in which human drivers are scarce. In Section 2.1, we emphasized the need for incremental deployability. As we will show experimentally, human compatibility adds significantly to the incremental deployability of the reservation system. We will also show that the specifics of the implementation offer further benefits: incentives for both communities and private individuals to adopt autonomous vehicle technology.

3.6.1 Using Existing Infrastructure

A reliable method of communicating with human drivers is a prerequisite for including them in the system. The simplest and best solution is to use something human drivers already know and understand — traffic lights. Traffic light infrastructure is already present at many intersections and the engineering and manufacturing of traffic light systems is well developed. For pedestrians and cyclists, standard "push-button" crossing signals can be used that give enough time for a person to traverse the intersection. These can also serve to alert the intersection to their presence.

3.6.2 Light Models

If real traffic lights are to be used to communicate to human drivers, they must be controlled and understood by the intersection manager. Thus, we add a new component to each intersection control policy, called a *light model*. The light model controls the physical lights as well as providing information to the policy with which it can make decisions. In more complicated scenarios, the light model can be modified by the control policy, for example, in order to adapt to changing traffic conditions. The lights have the same semantics as modern-day lights: red (do not enter), yellow (if possible, do not enter; light will soon be red), and green (enter). Each control policy requires a light model so that human users know what to do. For instance, the light model for FCFS keeps all the lights red at all times, indicating to humans that it is never safe to enter. The Traffic-Light policy's light model, on the other hand, corresponds exactly to the light system the policy is emulating. Here, we describe a few light models used in our experiments.

All-Lanes In this model, which is very similar to some current traffic light systems, each direction in succession gets green lights in all lanes. Thus, all northbound traffic (turning and going straight) has green lights while the eastbound, westbound, and southbound traffic all have red lights. The green lights then cycle through the directions. As it is similar to some current traffic lights, this light model is particularly well-suited to controlling distributions of vehicles with significant contingents of human drivers. We demonstrate this fact experimentally in Section 4.5. Figure 7 shows a graphical depiction of this light model.

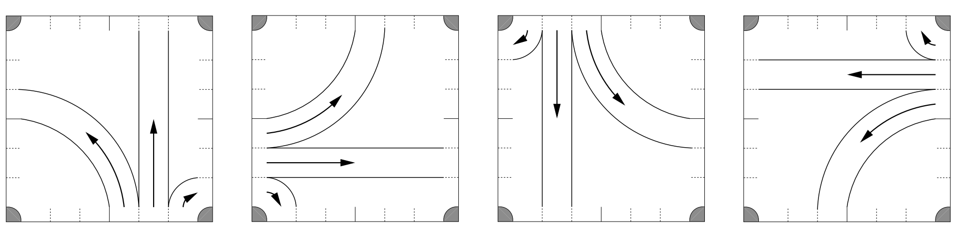


Figure 7: The All-Lanes light model. Each direction gets all green lights in a cycle: north, east, south, west. During each phase, the only available paths for autonomous vehicles with red lights are right turns.

Single-Lane In the Single-Lane light model, the green light rotates through the lanes one at a time instead of by direction. For example, the left turn lane of the northbound traffic would have a green light, while all other lanes would have a red light. Next, the straight lane of the northbound traffic would have a green light, then the right turn. Next, the green light would go through each lane of eastbound traffic, and so forth. A graphical description of the model's cycle can be seen in Figure 8. This light model does not work very well if most of the vehicles are human-driven, but as we will show, is very useful for intersections which control mostly autonomous vehicles but need also to handle an occasional human driver.

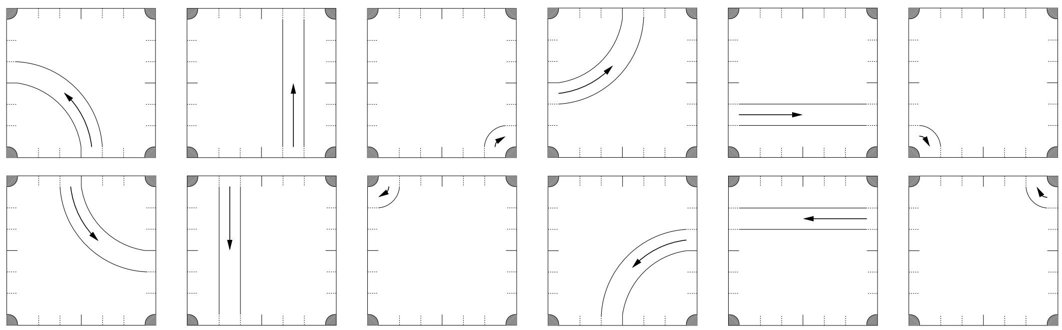


Figure 8: The Single-Lane light model. Each individual lane gets a green light (left turn, straight, then right turn), and this process is repeated for each direction. Note how a smaller part of the intersection is used by human vehicles at any given time. The rest of the intersection is available to autonomous vehicles.

3.6.3 The FCFS-Light Policy

In order to obtain some of the benefits of the FCFS policy while still accommodating human drivers, a policy needs to do two things:

1. If a light is green, ensure that it is safe for any vehicle (autonomous or human-driven) to drive through the intersection in the lane the light regulates.
2. Grant reservations to driver agents whenever possible. Autonomous vehicles can thus move through red lights (whereas humans cannot), provided they have a reservation—similar to a "right on red", but extended much further to other safe situations.

The policy FCFS-Light, which does both of these, is described as follows:

- As with FCFS, the intersection is divided into a grid of $n \times n$ tiles.
- Upon receiving a request message, the policy uses the parameters in the message to establish when the vehicle will arrive at the intersection.
- If the light controlling the lane in which the vehicle will arrive at the intersection will be green at that time, the reservation is confirmed.
- If the light controlling the lane will be yellow, the reservation is rejected.
- If the light controlling the lane will be red, the journey of the vehicle is simulated as in FCFS (Section 3.4).
- If throughout the simulation, no required tile is reserved by another vehicle or in use by a lane with a green or yellow light, the policy reserves the tiles and confirms the reservation. Otherwise, the request is rejected.

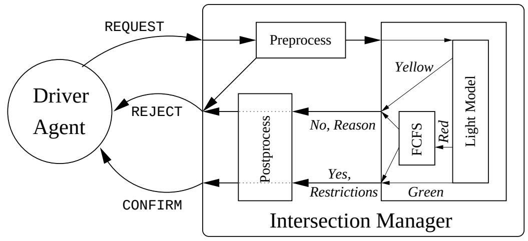


Figure 9: FCFS-Light is the combination of FCFS and a light model. When a request is received, FCFS-Light first checks to see what color the light will be. If it is green, it grants the request. If it is yellow, it rejects. If it is red, it defers to FCFS.

Off-Limits Tiles Unfortunately, simply deferring to FCFS does not guarantee the safety of the vehicle. If the vehicle were granted a reservation that conflicts with a vehicle following the physical lights, a collision could easily ensue. To determine which tiles are in use by the light system at any given time, we associate a set of *off-limits tiles* with each light. For example, if the light for the northbound left turn lane is green (or yellow), all tiles that could be used by a vehicle turning left from that lane are considered reserved for the purposes of FCFS. The length of the yellow light is adjusted so that vehicles entering the intersection have enough time to clear the intersection before those tiles are no longer off limits.

FCFS-Light Subsumes FCFS Using a traffic light–like light model (for example All-Lanes), FCFS-Light can behave exactly like Traffic-Light if all drivers are human. With a light model that keeps all lights constantly red, FCFS-Light behaves exactly like FCFS. In this case, if any human drivers are present it will fail spectacularly, leaving the humans stuck at the intersection indefinitely. However, in the absence of human drivers, it will perform exceptionally well. FCFS is just a special case of FCFS-Light. We can thus alter FCFS-Light's behavior to vary from strictly superior to Traffic-Light to exactly that of FCFS.

3.7 Emergency Vehicles

In current traffic laws there are special procedures involving emergency vehicles such as ambulances, fire trucks, and police cars. Vehicles are required to pull over to the side of the road and come to a complete stop until the emergency vehicle has passed. This is both because the emergency vehicle may be traveling quickly, posing a danger to other vehicles, and because the emergency vehicle must arrive at its destination as quickly as possible—lives may be at stake. Hopefully, once a system such as this is implemented, automobile accidents—a major reason emergency vehicles are dispatched—will be all but eradicated. Nonetheless, emergency vehicles will still be required from time to time as fires, heart attacks, and other emergencies will still exist. While we have previously proposed other methods for giving priority to emergency vehicles (Dresner & Stone, 2006), here we present a new, simpler method, which is fully implemented and tested.

3.7.1 Augmenting The Protocol

In order to accommodate emergency vehicles, the intersection manager must first be able to detect their presence. The easiest way to accomplish this is to add a new field to all request messages. In our implementation, this field is simply a flag that indicates to the intersection manager that the requesting vehicle is an emergency vehicle in an emergency situation (lights flashing and siren blaring). In practice, however, safeguards would need to be incorporated to prevent normal vehicles from abusing this feature in order to obtain preferential treatment. This could be accomplished using some sort of secret key instead of simply a boolean value, or even some sort of public/private key challenge/response mechanism. This details of the implementation, however, are beyond the scope of this project and are already a well-studied area of cryptography and computer security.

3.7.2 The FCFS-Emerg Policy

Now that the intersection control policy can detect emergency vehicles, it can process reservation requests while giving priority to the emergency vehicles. A first-cut solution is simply to deny reservations to any vehicles that are not emergency vehicles. However, this solution is not satisfactory, because if all the traffic comes to a stop due to rejected reservation requests, any emergency vehicles may get stuck in the resulting congestion. Instead, the FCFS-Emerg policy keeps track of which lanes currently contain approaching emergency vehicles. As long as at least one emergency vehicle is approaching the intersection, the policy grants reservations only to vehicles in those lanes. This ensures that vehicles in front of the emergency vehicles will also receive priority. Due to this increase in priority, lanes with emergency vehicles tend to empty very rapidly, allowing emergency vehicles to proceed relatively unhindered.

3.8 Summary

In this section, we have explained how we created a reservation-based intersection control mechanism in simulation. We described the construction of the simulator itself, as well as the communication protocol, the intersection manager, the driver agent, and several intersection control policies. The first policy, FCFS is only for fully autonomous vehicles. FCFS-Light extends FCFS to allow human interoperability using existing traffic light infrastructure. The last policy, FCFS-Emerg, extends FCFS to give priority to emergency vehicles without significant increasing delays for other vehicles.

4. Experimental Results

In this section, we fully test all of the features introduced in Section 3 and demonstrate that the reservation system can reduce delay by two orders of magnitude. Our experiments evaluate the performance of the reservation system using different intersection control policies, amounts of traffic, granularities, levels of human drivers, and the presence of emergency vehicles. We first compare the system using FCFS to traffic lights of varying cycle periods using a prototype simulator. We then show results from the full version, including the stop sign control policy as implemented under our protocol, comparing these results to those from the traffic light experiments. Next, we experiment with allowing vehicles to turn from any lane—something that would be extremely dangerous without the reservation-based mechanism. Finally, we evaluate the two extensions to FCFS: FCFS-Light and FCFS-Emerg. Videos of the simulator in action, including many scenarios from this section, as well as other supplementary materials can be found at `http://www.cs.utexas.edu/~kdresner/aim/`.

4.1 Low-Granularity FCFS vs. the Traffic Light

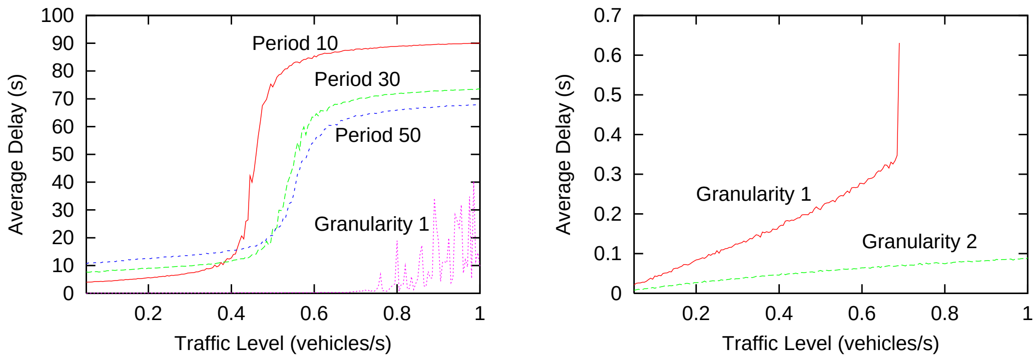
The simplest implementation of FCFS has granularity 1—the entire intersection is a single reservation tile. While only one vehicle may be in the intersection at a time, if that vehicle is traveling sufficiently fast, the total amount of time for which it will occupy the intersection is small. If we increase the granularity to 2, the intersection is no longer entirely exclusive. For example, non-turning vehicles traveling north no longer compete for the same reservation tiles as non-turning vehicles traveling south (similarly, eastbound and westbound non-turning vehicles no longer compete). Here we present our initial results comparing these two instances of the reservation mechanism and several incarnations of a traffic light.

4.1.1 Experimental Setup

These experiments were carried out using a prototype version of the simulator, which is fully described in an earlier publication (Dresner & Stone, 2004). In this version of the simulator, vehicles are not allowed to turn or accelerate while in the intersection. These restrictions do not detract from the core challenge of the problem, and the results are relevant even when the restrictions are relaxed. Each simulation contains one lane traveling in each direction, the speed limits of which are 25 meters per second. Traffic spawning probability varies from 0.0001 to 0.02 in increments of 0.0001, and each configuration runs for 500,000 steps in the simulator, which corresponds to approximately 2.5 hours of simulated time.

4.1.2 Results

Figure 10(a) shows delay times for traffic light systems with varying periods, ranging from extremely short (10 seconds) to fairly long (50 seconds). As expected from real-life experience, short-period traffic lights control light traffic well, while traffic lights with longer periods work better in heavy-traffic scenarios. When traffic is sparse, a short period allows vehicles to wait a shorter time before getting a green light. In many cities, traffic light periods are shortened during early hours of the morning to take advantage of this fact. In scenarios with more densely packed vehicles, the per-vehicle costs of slowing to a stop and accelerating back to full speed, as well as the intervals needed to clear out the intersection (the time during which there is a yellow light, or all lights are red), tend to dominate. This makes the longer-period lights better in these situations. In the Figure 10(a), above a certain traffic level, each of the traffic light systems reaches what appears to be a maximum delay level. This is an artifact of the simulator—when the traffic level gets high enough, the vehicles back up so far that the simulator cannot keep track of them (it cannot spawn new vehicles, for lack of a place to put them). At this point, vehicles are arriving at the intersection faster than the traffic lights can safely coordinate their passage. Thus, the point at which the delay spikes upwards indicates the maximum throughput of each traffic configuration.



(a) Reservation vs. Traffic Light

(b) Increasing Granularity

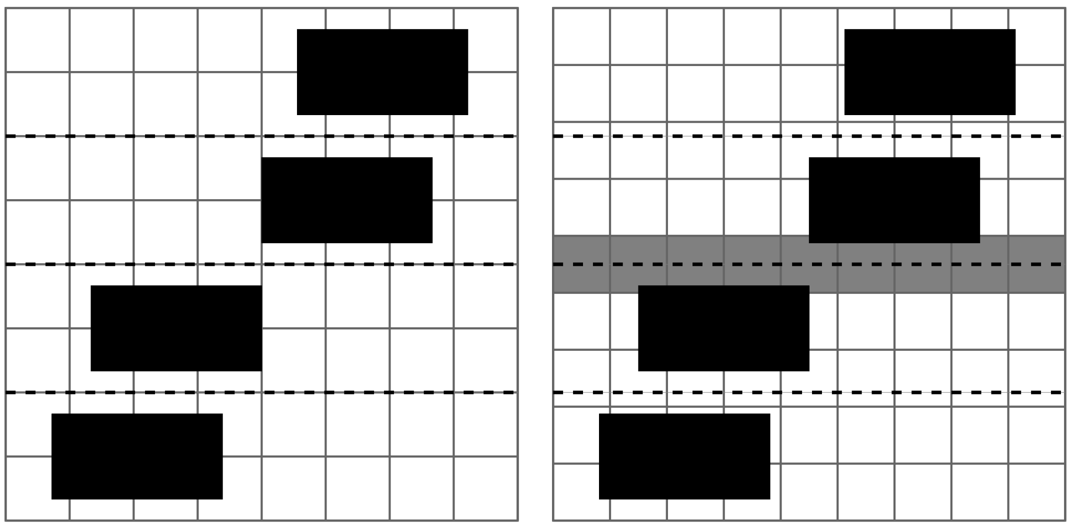
Figure 10: 10(a) shows average delays for traffic light systems with period 10, 30, and 50 seconds plotted against varying traffic levels along with a 1-tiled reservation-based system. 10(b) shows average delays for granularity 1 and 2 FCFS policies with varying traffic levels. Spawning probability was varied in increments of 0.0001, and each configuration was run for 1,000,000 steps of simulation (approximately 5.5 hours of simulated time). Each direction has 1 lane.

Also in Figure 10(a) are the delays for the granularity-1 and 2 FCFS policies. With the car spawning probability below about 0.013, the granularity-1 policy's delay is visually indistinguishable from the x-axis, while this is true for the granularity-2 reservation system for the whole graph. Figure 10(b) shows the bottom 0.7% of the graph, enlarged to show these results in more detail. At the vehicle spawning rate of 0.02, all of the traffic light systems are already beyond maximum capacity, while the granularity-2 system is allowing vehicles through without even adding a tenth of a second to the average vehicle's travel time.

4.2 Choosing Granularity

Of note in Figure 10(a) is the spike in delay for the granularity-1 FCFS policy. The system looks as though it is behaving chaotically—in Figure 10(b), delay slowly and steadily increases with the traffic level, until spiking off the graph when the probability of spawning a vehicle each time step reaches about 0.013.

With the granularity-1 system, vehicles traveling parallel to one another compete for the same tiles. This also happens to vehicles in the lanes closest to the middle of the road whenever the granularity is a small, odd number, as in Figure 11(b). Recall that in the prototype simulator, acceleration in the intersection is forbidden. Thus, if a vehicle slows down because it cannot obtain a reservation, when it finally does get a reservation it will be moving slowly for the entirety of the reservation and occupy the reservation tiles for a longer period of time. The next car to approach the intersection is therefore more likely to slow down as well. This process feeds itself and the vehicles slow down more and more. For small to average amounts of traffic, delays increase, but the system recovers during probabilistically generated periods of light traffic. However, for very heavy traffic, the intersection will eventually reach a deadlocked state. Because traffic is generated stochastically, this could happen early or late in the experiment. If it happens early, it will have a large effect on the average delay, whereas if it happens late, the effect will be smaller. Deadlocking is difficult to measure quantitatively, because as it progresses, driver agents make reservations for very long periods of time—so long, in fact, that they overflow the memory of the computer running the simulator. This effect can be seen in the rough line in Figure 10(a). To further explore the effects of granularity, we ran several more experiments, varying the granularity as well as the number of lanes.



(a) Granularity 8

(b) Granularity 9

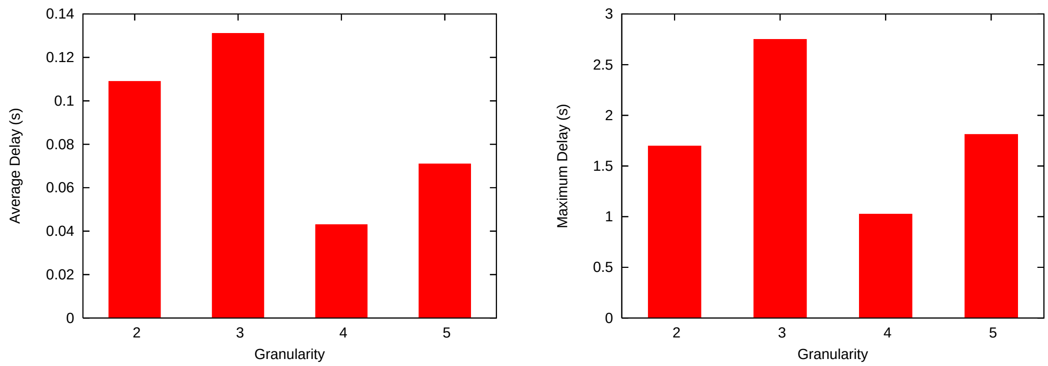
Figure 11: Increasing the granularity does not always improve performance. In 11(a), a granularity of 8 suffices. In 11(b), increasing the granularity to 9 actually hurts performance—vehicles traveling parallel to each other (but in opposite directions) are competing for the middle row of tiles.

4.2.1 Experimental Setup

These experiments were also used the prototype simulator as described in Section 4.1.1. Each data point represents 500,000 steps of simulation (approximately 2.5 hours of simulated time). The traffic level is fixed at 0.2 vehicles per second.

4.2.2 Results

As shown in Figure 12, with 2 lanes in each direction, a 2×2 grid performs better than a 3×3 grid. Increasing to a 4×4 grid is better than 2×2, but increasing it to 5×5 is again worse. An increase in granularity should correspond to a decrease in delay. However, for small granularities, incrementing the granularity from a small even number to a small odd number actually *increases* delay. In the case of maximum delay, even the granularity-2 system performs better than the granularity-5 system; the ill effects of odd granularities as shown in Figure 11 tend to slow down a few unfortunate vehicles.



(a) Average delay

(b) Maximum delay

Figure 12: Simulation statistics for FCFS policies with varying granularity. There are 2 lanes in each direction and the traffic level is 0.2 vehicles per second. Each experiment is run for 500,000 simulation steps. Note that increasing the granularity does not always improve performance.

This experiment suggests that FCFS should always be run with granularity high enough such that vehicles that never cross paths never compete for the same reservation tiles. As Figure 13 shows, more lanes require a higher granularity (though even with low granularity, the system out-performs the traffic light). However, because the computational complexity of the system increases proportional to the square of the granularity, the granularity should not be increased indiscriminately.

4.3 The Full Power of FCFS

While earlier experiments used a prototype simulator, these experiments use the full power of FCFS—turning, acceleration, and all the modifications from Section 3.4. Because vehicles turn, and thus do not always travel within a line of reservation tiles, increasing granularity beyond twice the number of lanes can improve performance even more. In addition to FCFS, we evaluate the stop sign policy as presented in Section 3.2.

Technically, the optimal delay for an individual vehicle is no delay at all. However, although a vehicle could experience delay as low as 0 seconds, turning vehicles may need to slow to avoid losing control. In order to create a worthwhile benchmark against which to compare the reservation system, we empirically measure the optimal *average* delay for an intersection manager. To do this, we use a special control policy that accepts all requests. We also deactivate each vehicle's ability to detect other vehicles, eliminating the interactions between them. These results are presented as the "optimal" control policy, which while optimal in terms of delay, provides no safety guarantees.

Small intersections with slow-moving traffic tend not to be amenable to control by traffic lights. Very light traffic can usually regulate itself fairly effectively. For example, consider an intersection with a stop sign—all vehicles must come to a stop, but afterwards may proceed

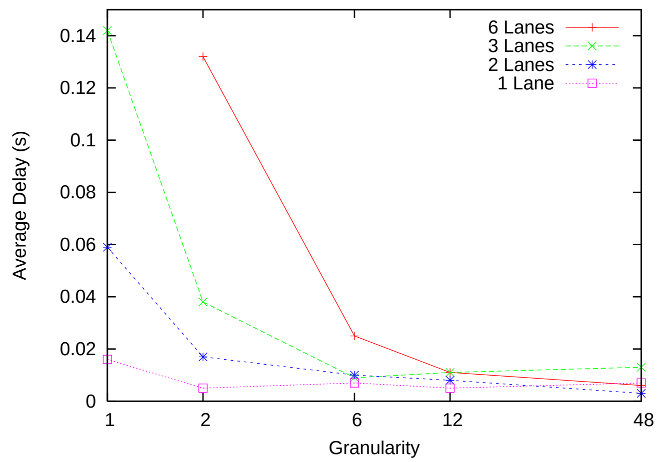


Figure 13: Average delays for the FCFS policy with independently varying numbers of lanes and granularity. Increasing the granularity beyond twice the number of lanes results in only marginal improvements. All simulations were run for at least 500,000 steps. 6 lanes with 1 tile deadlocks and overflows the system memory before 500,000 steps can complete.

if the intersection is clear. In these situations, a stop sign is often much more efficient than a traffic light, because vehicles are never stuck waiting for a light to change when there is no cross-traffic. The protocol enables us to define such a control policy, and we compare it experimentally to the other policies. Note that this policy is much more efficient than an actual stop sign, because once the vehicle has stopped at the intersection, the driver agent and intersection can determine when the car may safely proceed much more precisely and much less conservatively than a human driver.

4.3.1 Experimental Setup

The simulator simulates 3 lanes in each of the 4 cardinal directions. The speed limit in all lanes is 25 meters per second. Every configuration shown is run for at least 100,000 steps in the simulator, which corresponds to approximately half an hour of simulated time. Vehicles that are spawned turn with probability 0.1, and turning vehicles turn left or right with equal probability. Vehicles turning right are spawned in the right lane, whereas vehicles turning left are spawned in the left lane. Vehicles that are not turning are distributed probabilistically amongst the lanes such that the traffic in each lane is as equal as possible. FCFS and the stop sign (implemented as an extension of FCFS—see Section 3.5) both have a granularity of 24.

4.3.2 Results

The results for the experiments are shown in Figure 14. As expected, the average delay for the optimal system is positive and nonzero, but very small.

FCFS performs very well, nearly matching the performance of the optimal policy. At higher levels of traffic, the average delay for a vehicle gets as high as 0.35 seconds, but is never more than 1 second above optimal. Under none of the tested conditions does FCFS even approach the delay of the traffic light system from the previous experiment, shown in Figure 10(a).

The stop sign does not perform as well as FCFS, but for low amounts of traffic, it still performs fairly well, with average delay only about 3 seconds greater than optimal. However, as the traffic level increases, performance degrades. It is difficult to imagine a scenario in which this implementation of the stop sign would actually be used—it requires the same technology as the reservation system, but does not have any advantages over FCFS.

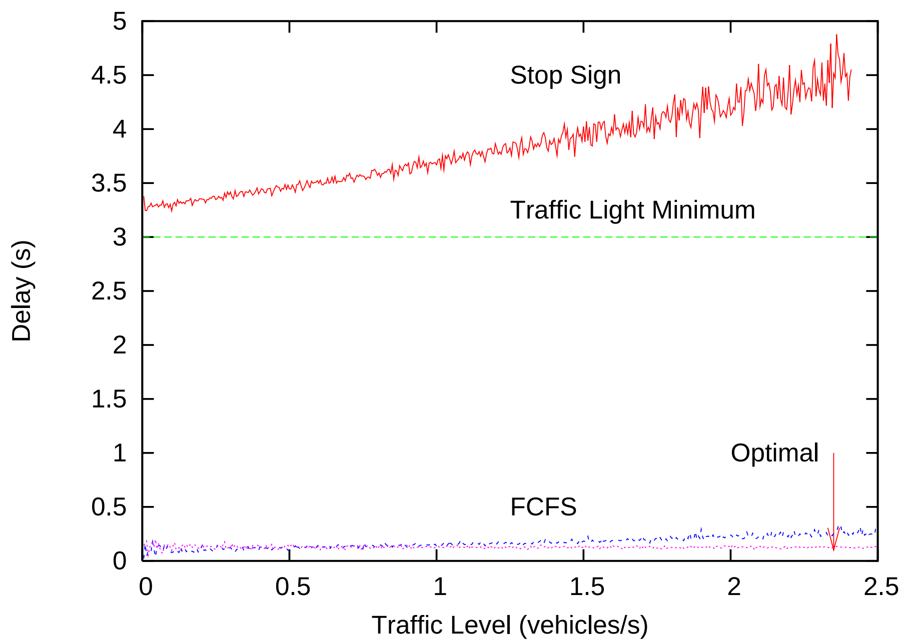


Figure 14: Delays for varying amounts of traffic for FCFS, the stop sign, and the optimal system.

4.4 Allowing Turns from Any Lane

In traditional traffic systems, especially those with traffic lights, vehicles wishing to turn onto the cross street must do so from specially designated turning lanes. This helps prevent cars that want to turn from holding up non-turning traffic. However, with a system like the reservation system, this restriction is no longer necessary. There is nothing inherent in the reservation system that demands vehicles turn from any specific lane. Investigating the effects of allowing turning from any lane produced some surprising results. As seen in Figure 15, relaxing the restriction actually hurts FCFS's performance slightly. While one might think this allows the vehicles more flexibility, on average it increases the resources used by any one turning vehicle. By making left turns from the left lane and right turns from the right lane, vehicles both travel a shorter distance and reserve reservation tiles that are less heavily used. However, these experiments may be misleading. Vehicles changing lanes to get into a designated turn lane could potentially delay vehicles behind them in the process. Because we do not currently model lane changing before the intersection, we have not been able to experimentally verify this conjecture.

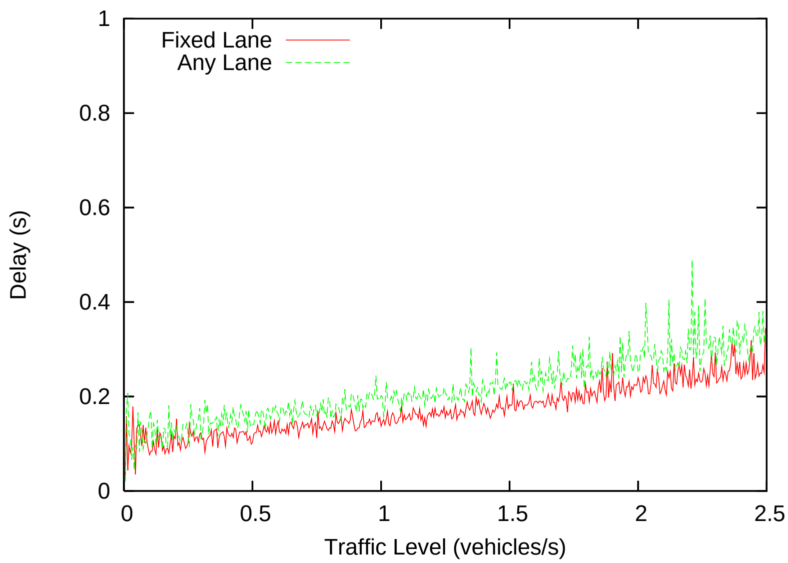


Figure 15: Comparison of an FCFS policy with traditional turns to one allowing turning from any lane. Allowing turns from any lane decreases performance slightly, producing longer delays.

4.5 The Effects of Human Interoperability

In Section 4.3, we showed that once all vehicles are autonomous, intersection-associated delays can be reduced dramatically. The following experiments suggest a stronger result: by using the two light models presented in Section 3.6.2, delays can be reduced at each stage of adoption. Furthermore, additional incentives exist at each stage for drivers to switch to autonomous vehicles.

4.5.1 Experimental Setup

For these experiments, the simulator models 3 lanes in each of the 4 cardinal directions. The speed limit in all lanes is 25 meters per second. For each intersection control policy with reservation tiles, the granularity is 24. The simulator spawns all vehicles turning left in the left lane, all vehicles turning right in the right lane, and all vehicles traveling straight in the center lane[1]. Unless otherwise specified, each data point represents 180000 time steps, or one hour of simulated time. Our simulated human-driven vehicles use a two-second following distance, but use the same lane-following algorithm as the autonomous drivers. They also employ a "point-of-no-return" mechanism for reacting to lights—if the vehicle can stop at a yellow or red light, it does, otherwise it proceeds.

4.5.2 Results

We present the experimental results for the human-compatible policies in two parts. The first focuses on how these policies can facilitate a smooth transition to an all-autonomous or mostly-autonomous vehicle system. The second focuses on the incentives throughout this process, both global and individual, to continue deployment of the system. Combined, these results suggest that an incremental deployment (one of the desiderata) is both technically possible and desirable.

Transition To Full Deployment The purpose of a hybrid intersection control policy is to confer the benefits of autonomy to passengers with driver-agent controlled vehicles while still allowing human users to participate in the system. Figure 16 shows a smooth and monotonically improving transition from modern-day traffic lights (represented by the Traffic-Light policy) to a completely or mostly autonomous vehicle mechanism (FCFS-Light with the Single-Lane light model). In early stages (100%-10% human), the All-Lanes light model is used. Later on (less than 10% human), the Single-Lane light model is introduced. At each change (both in driver populations and light models), delays are decreased. Notice the rather drastic drop in delay from FCFS-Light with the All-Lanes light model to FCFS-Light with the Single-Lane light model. Although none of the results is quite as close to the minimum as pure FCFS, the Single-Lane light model allows for greater use of the intersection by the FCFS portion of the FCFS-Light policy, which translates to higher efficiency and lower delay.

For systems with a significant proportion of human drivers, the All-Lanes light model works well—human drivers have the same experience they would with the Traffic-Light policy, but autonomous driver agents have extra opportunities to make it through the intersection. A small amount of this benefit is passed on to the human drivers, who may find themselves closer to the front of the lane while waiting for a red light to turn green. To explore how much the average vehicle would benefit, we ran our simulator with the FCFS-Light policy, the All-Lanes light model, and a 100%, 50%, and 10% rate of human drivers. This means that when a vehicle is spawned, it receives a human driver (instead of a driver agent) with probability 1, .5, and .1 respectively. As seen in Figure 17, as the proportion of human drivers decreases, the delay experienced by the average driver also decreases. While these decreases are not as large as those brought about by the Single-Lane light model, they are at least possible with significant numbers of human drivers.

Incentives For Individuals Even without any sort of autonomous intersection control mechanism, there are incentives for humans to switch to autonomous vehicles. Not having

1. This is a constraint we will likely relax in the future. It is included in this work to give the Single-Lane light model more flexibility and for a fair comparison to the FCFS policy, which performs even better in its absence.

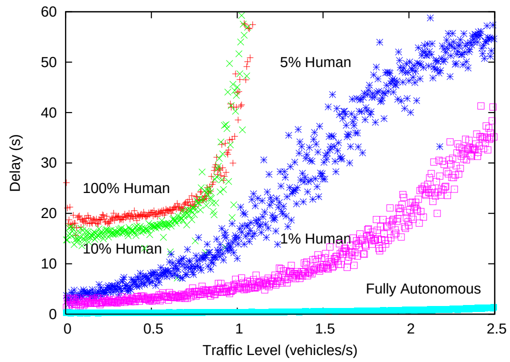


Figure 16: Average delays for all vehicles as a function of traffic level for FCFS-Light with two different light models: the All-Lanes light model, which is well-suited to high percentages of human-driven vehicles, and the Single-Lane light model, which only works well with relatively few human-driven vehicles. As adoption of autonomous vehicles increases, average delays decrease.

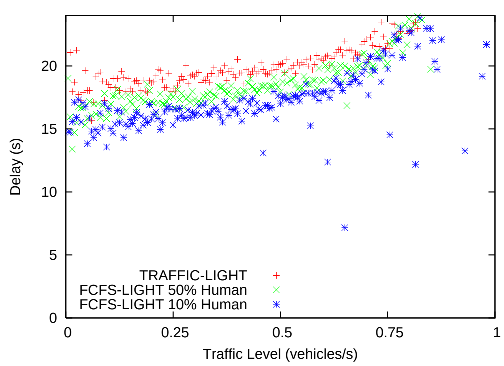


Figure 17: Average delays for all vehicles as a function of traffic level for FCFS-Light with the All-Lanes light model. Shown are the results for 100%, 50%, and 10% human-driven vehicles. The 100% case is equivalent to the Traffic-Light policy. Note that the average delay decreases as the percentage of human-driven vehicles decreases.

to do the driving, as well as the myriad safety benefits are strong incentives to promote autonomous vehicles in the marketplace. Our experimental results suggest additional incentives. Using our reservation system, autonomous vehicles experience lower average delays than human-driven vehicles and this difference increases as autonomous vehicles become more prevalent.

Figure 18 shows the average delays for human drivers as compared to autonomous driver agents for the FCFS-Light policy using the All-Lanes light model. In this experiment, half of the drivers are human. Humans experience slightly longer delays than autonomous vehicles, but not worse than with the Traffic-Light policy. Thus, by putting some autonomous vehicles on the road, all drivers experience equal or smaller delays as compared to the current situation. This is expected because the autonomous driver can do everything the human driver does and more.

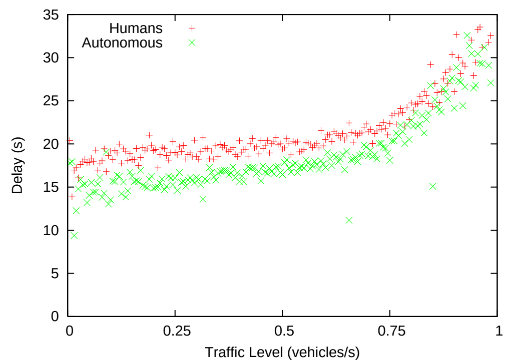


Figure 18: Average delays for human-driven vehicles and all vehicles as a function of traffic level for FCFS-Light with the All-Lanes light model. In this experiment, 50% of vehicles are human driven. Autonomous vehicles experience slightly lower delays across the board, and human drivers experience delays no worse than the Traffic-Light policy.

Once the reservation system is in widespread use and autonomous vehicles make up a vast majority of those on the road, the door is opened to an even more efficient light model for the FCFS-Light policy. With a very low concentration of human drivers, the Single-Lane light model can drastically reduce delays, even at levels of overall traffic that the Traffic-Light policy can not handle. Using this light model, autonomous drivers can pass through red lights even more frequently because fewer tiles are off-limits at any given time. In Figure 19 we compare the delays experienced by autonomous drivers to those of human drivers when only 5% of drivers are human and thus the Single-Lane light model can be used. While the improvements using the All-Lanes light model benefit all drivers to some extent, the Single-Lane light model's sharp decrease in average delays (Figure 16) comes at a high price to human drivers.

As shown in Figure 19, human drivers experience much higher delays than average. For lower traffic levels, these delays are even higher than those associated with the Traffic-Light policy. Figure 16 shows that despite this, at high levels of traffic, human drivers benefit relative to Traffic-Light. Additionally, intersections using FCFS-Light will still be able to handle far more traffic than those using Traffic-Light.

The Single–Lane light model effectively gives the humans a high, but fairly constant delay. Because the green light for any one lane only comes around after each other lane has had a green light, a human-driven vehicle may find itself sitting at a red light for some time before the light changes. However, since this light model would only be put in operation once human drivers are fairly scarce, the huge benefit to the other 95% or 99% of vehicles far outweighs this cost. A light model that detects and reacts to the presence of human

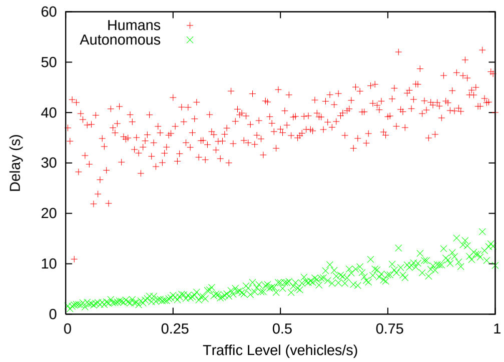


Figure 19: Average delays for human-driven vehicles and all vehicles as a function of traffic level for FCFS-Light with the Single-Lane light model. Humans experience worse delay than with Traffic-Light, but average delay for all vehicles is much lower. In this experiment, 5% of vehicles are human-driven.

drivers might be able to achieve even better overall performance, without causing the human drivers to wait as long.

These data suggest that there will be an incentive to both early adopters (persons purchasing vehicles capable of interacting with the reservation system) and to cities or towns. Those with properly equipped vehicles will get where they are going faster (not to mention more safely). Cities and towns that equip their intersections to utilize the reservation paradigm will experience fewer traffic jams and more efficient use of the roadways (along with fewer collisions and less wasted gasoline). Because there is no penalty to the human drivers (which would presumably be a majority at this point), there would be no reason for any party involved to oppose the introduction of such a system. Later, when most drivers have made the transition to autonomous vehicles, and the Single-Lane light model is introduced, the incentive to move to the new technology is increased—both for cities and individuals. By this time, autonomous vehicle owners will far outnumber human drivers, who will still benefit when traffic is at its worst.

4.6 Emergency Vehicle Experiments

While we have already shown that FCFS on its own can significantly reduce average delays for all vehicles, FCFS-Emerg helps reduce delays for emergency vehicles even further.

4.6.1 Experimental Setup

To demonstrate this improvement, we ran the simulator with varying amounts of traffic, while keeping the proportion of emergency vehicles fixed at 0.1% (that is, a spawned vehicle is made into an emergency vehicle with probability 0.001). Because of the very small number of emergency vehicles created with realistically low proportions, we ran each configuration (data point) for 100 hours of simulated time—much longer than the other experiments.

4.6.2 Results

As shown in Figure 20, the emergency vehicles on average experience lower delays than the normal vehicles. The amount by which the emergency vehicles outperform the normal vehicles increases as the traffic increases, suggesting that as designed, FCFS-Emerg helps most when more traffic is contending for space-time in the intersection.

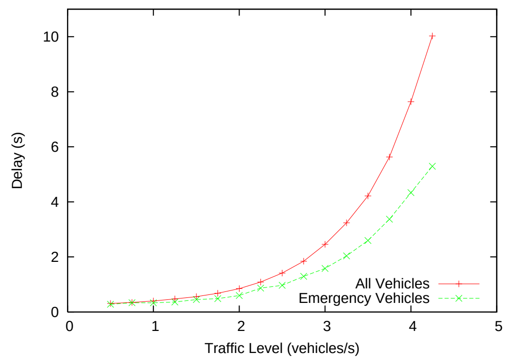


Figure 20: Average delays for all vehicles and emergency vehicles as a function of traffic level for the FCFS-Emerg policy. One out of a thousand vehicles (on average) is an emergency vehicle. Delays for the emergency vehicles are lower for all data points.

5. Performance in Failure Modes

Fully autonomous vehicles promise enormous gains in safety, efficiency, and economy for transportation. However, before such gains can be realized, a plethora of safety and reliability concerns must be addressed. In the previous sections, we have assumed that all vehicles perform without gross malfunctions. In this section, we relax that assumption and demonstrate how our reservation-based mechanism reacts to scenarios in which such malfunctions occur. Additionally, we intentionally disable some elements of the system in order to investigate both their necessity and efficacy.

5.1 Causes of Accidents

A collision in purely autonomous traffic can have any number of causes, including software errors in the driver agent, a physical malfunction in the vehicle, or even meteorological phenomena. In modern-day traffic, such factors are largely ignored for two reasons. First, the exclusively human-populated system, with its generous margins for error, is not as sensitive to small or moderate aberrations. Second, none of these are significant with respect to driver error as causes of accidents (Wierwille, Hanowski, Hankey, Kieliszewski, Lee, Medina, Keisler, & Dingus, 2002). However, in the future of infallible autonomous driver agents, it is exactly these issues which will be the prevalent causes of automobile collisions. The safety allowances explained in Sections 3.4.5 and 3.4.6 are adjustable—given some maximum allowable error in vehicle positioning, the buffers can be extended to handle that error—but no reasonable adjustment can account for gross mechanical malfunction like a blowout or failed brakes. Because these types of issues are infrequent, we believe the safety of the intersection control mechanism will be acceptable even if individual occurrences are slightly worse than accidents today.

5.2 Adding a Safety Net

One can easily imagine how badly an accident in such an efficient system could be without any reactive safety measures in place. Here, we explain how the system deals with these rare, but dangerous events. As we will show in Section 5.3, disabling the safety measures leaves the system prone to spectacular failure modes, sometimes involving dozens of vehicles. Intact, the measures make such events much more manageable.

5.2.1 Assumptions

In Section 5.3, we will show how our reactive safety measures can reduce the average number of vehicles involved in a crash from dozens to one or two. However, in order to employ these safety measures fully, we must make a few additional assumptions.

Detecting The Problem First, we assume that the intersection manager is able to detect when something has gone wrong. While this is certainly a non-trivial assumption, without it, no substantial mitigation is possible. Simply put, the intersection manager cannot react to something it cannot detect. There are two basic ways by which the intersection manager could detect that a vehicle has encountered some sort of problem: the vehicle can inform the intersection manager, or the intersection manager can detect the vehicle directly. For instance, in the event of a collision, a device similar to that which triggers an airbag can send a signal to the intersection manager. Devices like this already exist in aircraft to emit distress signals and locator beacons in the event of a crash. The intersection manager itself might notice a less severe problem, such as a vehicle that is not where it is supposed to be, using cameras or sensors at the intersection. However, this method of detection is likely to be much slower to react to a problem. Each has advantages and disadvantages, and a combination of the two would most likely be the safest. The specifics of the implementation are beyond the scope of this analysis. What is important is that whenever a vehicle violates its reservation in any way, the intersection manager should become aware as soon as possible. Because our simulations only deal with collisions, we assume that the colliding vehicle sends a signal and the intersection manager becomes aware of the situation immediately.

As described in Appendix B, our protocol includes a Done message that vehicles transmit when they complete their reservations. One way to reliably sense when a vehicle is in distress would be to notice a missing Done message. This approach has two drawbacks. First, the Done message is optional, mainly because there is no incentive for the driver agent to transmit it. Second, the intersection manager may not be able to notice the missing message until some time after the incident has occurred. We intend to investigate this alternative in future work.

Informing Other Vehicles We also assume that there exists a way for the intersection manager to broadcast the fact that something is wrong to the vehicles. Since the intersection manager can already communicate with the vehicles, this is not a big assumption.

However, the mode of communication is a bit different from that employed in the rest of the communication protocol (see Appendix B). Under normal operating conditions, individual messages each containing multiple pieces of information are transmitted between agents. Because we cannot verify the receipt of these messages without a response, the semantics of the protocol ensure that whenever a message is sent, the sending agent makes the most conservative assumption—in the case of a Request message, that it was not received; in the case of a Confirm message, that it was. In the event of a collision, however, the intersection manager needs to communicate one bit of information to as many vehicles as possible: that something is wrong. Because it is very important that all vehicles receive this message, it is transmitted repeatedly, to all vehicles, to the exclusion of all other messages. While we would like to assume that all vehicles receive this message, we will show in Section 5.3 that even when a significant number of vehicles do not, the safety measures in place still protect many vehicles that would otherwise wind up crashing.

5.2.2 Incident Mitigation

When a vehicle deviates significantly from its planned course through the intersection resulting in physical harm to the vehicle or its presumed occupants, we refer to the situation as an *incident*. Once an incident has occurred, the first priority is to ensure the safety of all persons and vehicles nearby. Because we expect such incidents to be very infrequent occurrences, re-establishing normal operation of the intersection is a lower priority and the optimization of that process is left to future work.

Intersection Manager Response As soon as the intersection manager detects or is notified of an incident, it immediately stops granting reservations. All subsequent received requests are rejected without consideration. Due to the nature of the protocol, the intersection manager cannot revoke reservations, as driver agents would have no incentive to acknowledge their receipt. However, the intersection manager can send a message to the vehicles that an incident has occurred. This message is the special Emergency-Stop message, which the intersection manager may only send in an emergency situation, and which (as with the rest of the protocol) it must assume has not been received.

The Emergency-Stop message lets vehicles know that an event has taken place in the intersection such that:

- no further reservations will be accepted
- vehicles able to come to a stop before entering the intersection should do so
- vehicles in the intersection should no longer assume that "near misses" will not result in collisions

For human-compatible policies, such as FCFS-Light, the intersection manager also turns all lights red. In a real-world implementation, a more conspicuous visual cue could be provided, but semantically it is only important that the intersection informs the human drivers that they may not enter.

Vehicle Response For the Emergency-Stop message to be useful in any way, driver agents must react to it. Here we explain the specific actions our implementation of the driver agent takes when it receives this message. Normally, when approaching the intersection, our driver agent ignores any vehicles sensed in the intersection. This is because what might otherwise appear to be an imminent collision on the open road is almost certainly a precisely coordinated "near-miss" in the intersection. However, once the driver agent receives the Emergency-Stop message from the intersection manager, it disables this behavior. If the vehicle is in the intersection, the driver agent will not blindly drive into another vehicle if it can help it. If the vehicle is not in the intersection and can stop in time, it will not enter, even if it has a reservation.

While our first inclination was to make the driver agent immediately decelerate to a stop, we quickly realized that this is not the safest behavior. If all vehicles that receive the message come to a stop, vehicles that would otherwise have cleared the intersection without colliding may find themselves stuck in the intersection—another object for other vehicles to run into. This is especially true if the vehicle that caused the incident is on the edge of the intersection where it is unlikely to be hit. Trying to stop all the other vehicles in the intersection just makes the situation worse.

If a driver agent does detect an impending collision, it should take evasive actions or apply the brakes. Since this is a true multiagent system with self-interested agents, we cannot prevent driver agents from doing so, even if it is detrimental to vehicles overall. Thus, our driver agent brakes if it believes a collision is imminent.

5.3 Experiments

In order to evaluate the effects of our reactive safety measures, we performed several experiments in which various components were intentionally disabled. The various configurations can be separated into three classes. An *oblivious* intersection manager takes no action at all upon detecting an incident. An intersection manager utilizing *passive* safety measures stops accepting reservations, but does not send any Emergency-Stop messages to nearby driver agents. Finally, the *active* configuration of the intersection manager—which corresponds to the full version of the protocol as specified in Appendix B—has all safety features in place. In addition to considering these three incarnations of the intersection manager, we also study the effects of unreliable communication in the active case. Note that when no vehicles receive the Emergency-Stop message, the active and passive configurations are identical.

5.3.1 Experimental Setup

With the great efficiency of the reservation-based system comes an extreme sensitivity to error. While buffering might protect against minute discrepancies, it cannot hope to cover gross mechanical malfunctions. To determine just how much of an effect such a malfunction would have, we created a simulation in which individual vehicles could be "crashed", causing them to immediately stop and remain stopped. Whenever a vehicle that is not crashed comes into contact with one that is, it becomes crashed as well. While this does not model the specifics of individual impacts, it does allow us to estimate how a malfunction might lead to collisions.

In order to ensure that we included malfunctions in all different parts of the intersection, we triggered each incident by choosing a random (x, y) coordinate pair inside the intersection, and crashing the first vehicle to cross either the x or y coordinate. This is akin to creating two infinitesimally thin walls, one horizontal and the other vertical, that intersect at (x, y). Figure 21 provides a visual depiction of this process.

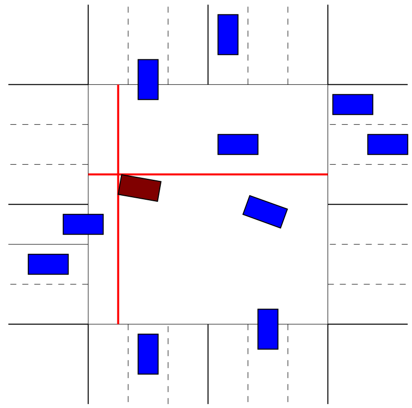


Figure 21: Triggering an incident in the intersection simulator. The dark vehicle turning left is crashed because it has crossed the randomly chosen x coordinate. If a different vehicle had crossed that x coordinate or the randomly chosen y coordinate earlier, it would be crashed instead.

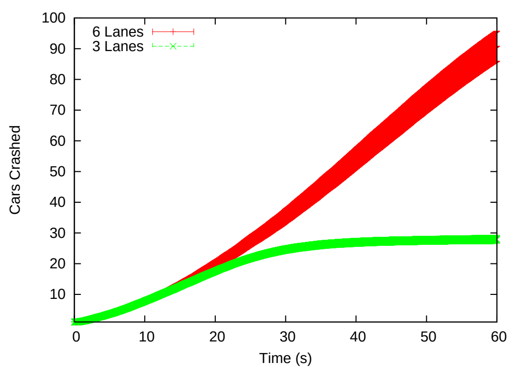
After initiating an incident, we ran the simulator for an additional 60 seconds, observing any subsequent collisions and recording when they occurred. Using this information, we constructed a *crash log*, which is essentially a histogram of crashed vehicles. For each step of the remaining simulation, the crash log indicates how many vehicles were crashed by that step. By averaging over many such crash logs for each configuration, we were able to construct an "average" crash log, which gives a picture of what a typical incident would produce.

Because our system is compatible with humans, we included experiments with a human-compatible intersection control policy. As demonstrated in Section 4.5, when a significant number of human drivers are present, the FCFS-Light cannot offer much of a performance benefit over traditional traffic light systems. As such, we limited our experimentation to scenarios in which 5% of the vehicles are controlled by simulated human drivers, and used a Single-Lane light model (see Section 3.6.2). With only 5% human drivers, an FCFS-Light policy can still create a lot of the precarious situations that are the focus of this investigation.

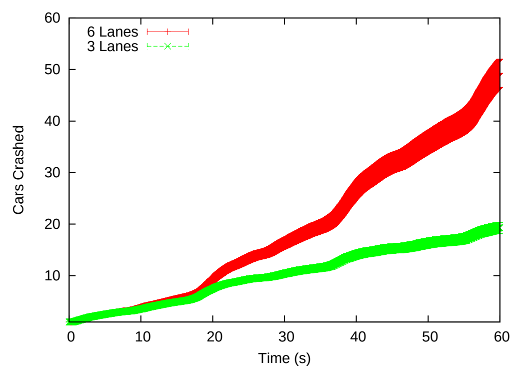
For these experiments, we ran our simulator with scenarios of 3, 4, 5, and 6 lanes in each of the four cardinal directions, although we will discuss results only for the 3- and 6-lane cases (other results were similar) for the sake of brevity. As with earlier experiments, vehicles are spawned equally likely in all directions, and are generated via a Poisson process which is controlled by the probability that a vehicle will be generated at each step. Vehicles are generated with a set destination—15% of vehicles turn left, 15% turn right, and the remaining 70% go straight. As before, the leftmost lane is always a left turn lane, while the right lane is always a right turn lane. Turning vehicles are always spawned in the correct lane, and non-turning vehicles are not spawned in the turn lanes. In scenarios involving only autonomous vehicles, we set the traffic level at an average of 1.667 vehicles per second per lane in each direction. This equates to 5 total vehicles per second for 3 lanes, and 10 total vehicles per second for 6 lanes. Scenarios with human-driven vehicles had one third the traffic of the fully autonomous scenarios—the intersection cannot be nearly as efficient with human drivers present. We chose these amounts of traffic as they are toward the high end of the spectrum of manageable traffic for the respective variants of the intersection manager. While we wanted traffic to be flowing smoothly, we also wanted the intersection to be full of vehicles to test situations that likely lead to the most destructive possible collisions.

5.3.2 How Bad Is It?

As we suspected, the average crash log of the oblivious intersection manager is quite grisly. As explained in Section 5.2.2, driver agents must ignore their sensors while in the intersection, because many of the "close calls" would appear to be impending collisions. Without any way to react the situation going awry, vehicles careen into the intersection, piling up until the entire intersection is filled and crashed vehicles protrude into the incoming lanes. Figure 22 shows that for both 6-lane cases—fully autonomous and 5% human drivers—the rate of collisions does not abate until over 70 vehicles have crashed. Even a full 60 seconds after the incident begins, vehicles are still colliding. In the 3-lane case, the intersection is much smaller and thus fills much more rapidly; by 50 seconds, the number of collided vehicles levels off.



(a) All autonomous



(b) 5% humans

Figure 22: Average crash logs (with 95% confidence interval) for 3- and 6-lane oblivious intersections. In 22(a), the intersection manages only autonomous vehicles, while 22(b) includes 5% human drivers.

In both of the scenarios with human drivers, shown in Figure 22(b), the number of vehicles involved in the average incident is noticeably smaller. This outcome is likely the result of two factors. First and foremost, the FCFS-Light policy must make broad allowances to accommodate the human drivers, and thus overall is inherently less dangerous. The characteristic "close calls" from the standard FCFS policy are less common. Second, the simulated human driver agents do not drive "blindly" into the intersection—trusting to the intersection manager—the way the autonomous vehicles do. Also of note in Figure 22(b) is the visible periodicity of the light model portion of the policy. As paths open up for autonomous vehicles due to changes in the lights, they drive unwittingly into the growing mass of crashed cars.

5.3.3 Reducing the Number of Collisions

There are two main components to the safety mechanism introduced in Section 5.2. First, the intersection manager stops accepting reservations. Second, the intersection manager sends messages informing the driver agents that an incident has taken place. There is a possibility that this second part might not always work perfectly; some vehicles might not receive the message. To investigate the effects of these potential communication failures, we intentionally disabled some of the vehicles' ability to receive the Emergency-Stop message. A parameter in our simulator controls the fraction of vehicles created with this property, and by varying this parameter, we could observe its subsequent effect on the average number of vehicles involved in incidents.

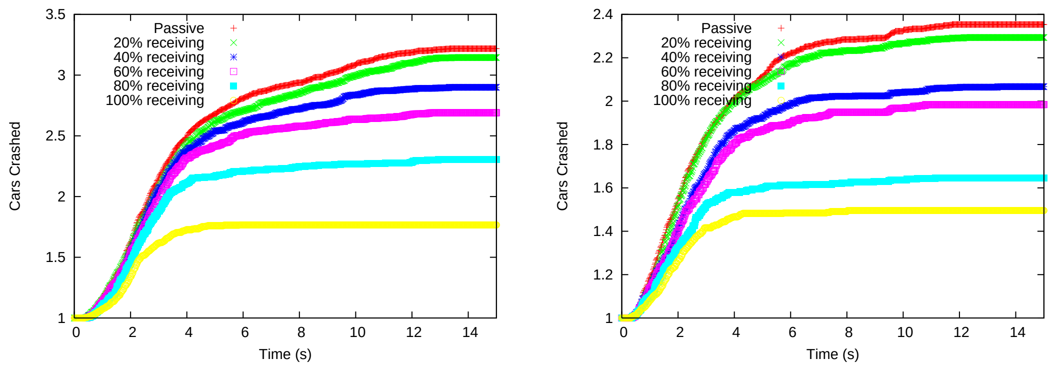
As compared to the oblivious intersection manager, the number of vehicles involved in the average incident for an active intersection manager decreases dramatically. Table 1 shows the numerical results for both the 3- and 6-lane intersections, along with a 95% confidence interval. The average crash logs for these runs are not shown in Figure 22, as they would be indistinguishable from one another at that scale. Instead, we present them in Figure 23.

	Fully Autonomous		5% Human	
	3 Lanes	6 Lanes	3 Lanes	6 Lanes
Oblivious	**27.9** $\pm$ 1.3	**90.9** $\pm$ 4.9	**19.3** $\pm$ 1.1	**49.3** $\pm$ 2.7
Passive	**2.63** $\pm$ .13	**3.23** $\pm$ .16	**2.23** $\pm$ .10	**2.35** $\pm$ .13
Active				
20% receiving	2.44 $\pm$ .13	3.15 $\pm$ .17	2.07 $\pm$ .10	2.29 $\pm$ .13
40% receiving	2.28 $\pm$ .12	2.90 $\pm$ .16	1.91 $\pm$ .10	2.07 $\pm$ .12
60% receiving	1.89 $\pm$ .10	2.69 $\pm$ .15	1.72 $\pm$ .09	1.98 $\pm$ .11
80% receiving	1.71 $\pm$ .08	2.30 $\pm$ .13	1.46 $\pm$ .07	1.65 $\pm$ .09
100% receiving	**1.36** $\pm$ .06	**1.77** $\pm$ .10	**1.22** $\pm$ .05	**1.50** $\pm$ .09

Table 1: Average number of simulated vehicles involved in incidents for 3- and 6-lane intersections. Even with only the passive safety measures, the number of crashed vehicles is dramatically decreased from the oblivious intersection manager. In the active configuration, as more vehicles receive the emergency signal, the number of crashed vehicles decreases further.

Figure 23 shows the effects of the reactive safety measures in intersections with 6 lanes, with the proportion of receiving vehicles varying from 0% (passive) to 100% in increments of 20%. Even in the passive configuration, the overall number of vehicles involved in the average incident decreases by a factor of almost 30 in the fully autonomous scenario, and a factor of over 20 in the scenario with 5% human drivers, as compared to the oblivious intersection manager. As expected in the active configuration, when more vehicles receive the emergency signal, fewer wind up crashing. The graphs in Figure 23 only show the first

15 seconds of the incident, because in no case did a collision occur more than 15 seconds after the incident started.



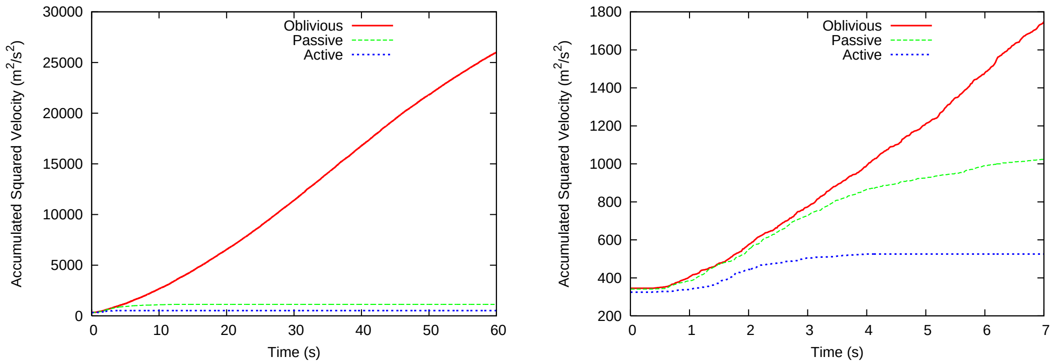
(a) All autonomous (b) 5% humans

Figure 23: The first 15 seconds of average crash logs for 6-lane passive and active intersections. As more vehicles react to the signal, safety improves.

5.3.4 Reducing the Severity of Collisions

While it is reassuring to know that the number of vehicles involved in the average incident can be kept fairly low, these data do not give the entire picture. For example, compare an incident in which 30 vehicles each lose a hubcap to one in which two vehicles are completely destroyed and all occupants killed. While we do not currently have any plans to model the intricate physics of each individual collision with high fidelity, our simulations do allow us to observe the velocity at which the collisions occur. In the previous example, we might notice that the 30 vehicles all bumped into one another at low velocities, while the two vehicles were traveling at full speed. To quantify this information, we record not only when a collision happens, but the velocity at which it happens. In a collision, the amount of damage done is approximately proportional to the amount of kinetic energy that is lost. Because kinetic energy is proportional to the square of velocity, we can use a running total of the squares of these crash velocities to create a rough estimate of the amount of damage caused by the incident. Figure 24 shows an average "damage log" of a 6-lane intersection of autonomous vehicles. Qualitatively similar results were found for the other intersection types.

As Figure 24(a) shows, the effect of our safety measures under this metric is quite dramatic as well. In the passive case the total accumulated squared velocity decreases by a factor of over 25. In the active case, with all vehicles receiving the signal, it decreases by another factor of 2. Of particular note is the zoomed-in graph in Figure 24(b). In the passive configuration, the total squared velocity accumulates as if the intersection manager were oblivious, until the first vehicles stop short of the intersection at around 3 seconds; without a reservation, they may not enter. In the active scenario, when all the vehicles receive the message, the improvement is almost immediate.

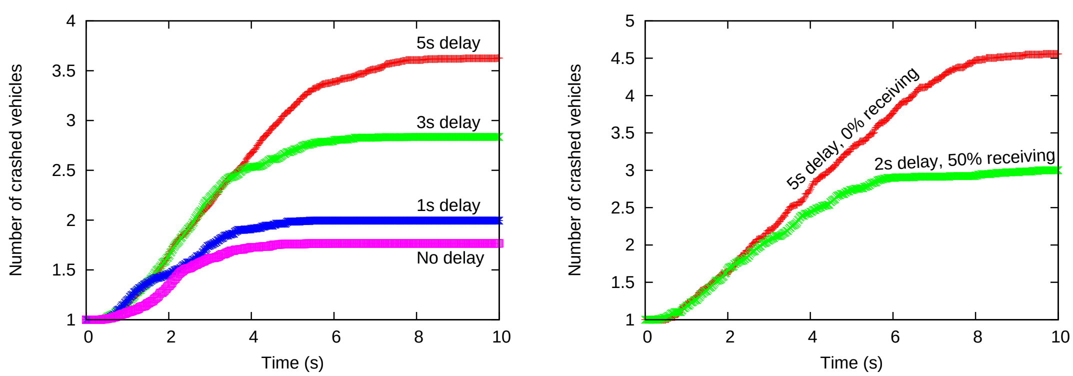


(a) the average incident (b) zoomed in

Figure 24: Average total squared velocity of crashed vehicles for a 6-lane intersection with only autonomous vehicles. Sending the emergency message to vehicles not only causes fewer collisions, but also makes the collisions that do happen less dangerous.

5.3.5 Delayed Incident Detection

Implicit in these results is the assumption that intersection managers become aware of incidents instantaneously. While this could be the case in many collisions—vehicles should communicate when they have collided—if a vehicle's communications are faulty, or if the vehicle does not realize it has collided, the intersection may not discover the problem for a few seconds, when another vehicle or sensor will detect the problem. To assess the effects of delayed incident detection, we artificially delayed the intersection manager's response in some of our simulations. Figure 25 shows the results from these experiments.



(a) delaying detection (b) delays and faulty communication

Figure 25: Crash logs showing the effects of delayed incident detection.

In Figure 25(a), the intersection manager's reaction was delayed 0, 1, 3, and 5 seconds. Note that the total number of crashed vehicles with a delay of 5 seconds is on par with the number in the experiment in which the intersection manager reacts immediately, but none of the vehicles receive the message, shown in Figure 23(a). Figure 25(b) shows what happens with both delayed detection and faulty communication. This graph, along with the earlier results, suggests that for small values, each second of delay is approximately equivalent to 20% of vehicles not receiving the Emergency-Stop message, and that when combined, delayed detection and faulty communication have an additive effect. For larger delays, the number of vehicles involved can be approximated using the data shown in Figure 22(a), because in these cases, the number of vehicles that crash after the intersection is much smaller than the number that crash before it reacts.

5.4 Safety Discussion

The results in this section suggest that it may be possible to improve efficiency while also improving safety. But of course before deployment in the real world, extensive testing with real vehicles would be needed in order to verify both the suggested efficiency benefits, as well as the safety properties of the system. People are often hesitant to put their well-being (physical or otherwise) in the hands of a computer unless they can be convinced that they will receive a significant safety benefit in exchange for surrendering precious control. Humans often suffer from the *overconfidence effect*, erroneously believing they are more skillful than others. In a 1981 survey of Swedish drivers, respondents were asked to rate their driving ability in relation to others. A full 80% of those asked placed themselves in the top 30% of drivers (Svenson, 1981). It is this effect that creates the high standard to which computerized systems are held. It is insufficient for such systems to be marginally safer, or safer for the *average* user; they must be the very paragon of safety.

In our experiments, we showed that the number of vehicles involved in individual incidents can be drastically reduced by utilizing a fairly straightforward reactive safety mechanism. In fact, in the active configuration with 3 lanes, 75% of the incidents involved only one vehicle: the one we intentionally crashed (60% for 6 lanes). Even in the passive case with 6 lanes of traffic, an average of only 3.23 vehicles were involved. But how does this compare with current systems? If we conservatively assume that accidents in traffic today involve only one vehicle, this represents a 223% increase per occurrence. Thus, all other things being equal, if the frequency of accidents can be reduced by 70%, these experiments suggest that an autonomous intersection management system will be safer overall. A 2002 report for the U.S. Federal Highway Administration blamed over 95% of all accidents on driver error (Wierwille et al., 2002). The remaining accidents were divided equally between vehicle failures and problems with roads. It is important to note that these numbers are for all driving, not just intersection driving. Accidents in intersections are even more likely to be caused by driver error, sometimes even by drivers willfully disobeying the law: running red lights and stop signs or making illegal "U"-turns.

Even if we make overly conservative assumptions—that all driving is as dangerous as intersection driving, and that driver error is no more accountable for intersection crashes than it is in other types of driving—our data suggest that automobile traffic with autonomous driver agents and an intersection control mechanism like ours will reduce collisions in intersections by over 80%. We believe that in reality, the improvement will be much greater.

The safety measures presented in this section constitute just one approach for mitigating the system's failure modes. More sophisticated methods involving explicit cooperation amongst vehicles may create an even safer system. We have not shown (or attempted to show) that this particular solution is the best possible. Rather we have demonstrated that even with a simple and straightforward response to accidents, the overall safety of the system can be maintained, without sacrificing the benefits of vastly improved efficiency.

6. Related Work

Traffic control is a vast area of research for computer scientists and engineers alike. The field of Intelligent Transportation Systems (ITS) is concerned with applying information, computing, and sensor technologies to solve problems in traffic and road management (Bishop, 2005). ITS includes intelligent vehicles (IV) as well as infrastructure, such as intersections. Unfortunately, while both aspects of ITS are heavily studied, relatively little current research considers how intelligent or autonomous vehicles and infrastructure can work together to improve the efficiency and safety of the overall traffic system. The Berkeley PATH project has produced a lot of interesting work, including work on a fully-automated highway (Alvarez & Horowitz, 1997).

In this section, we describe some work related to our own, both directly and tangentially. Some of this work is specifically concerned with intersection control, some takes a multiagent approach to other aspects of traffic management, and some represents work on the technologies necessary to bring fully autonomous vehicles into the mainstream.

6.1 Requisite Technology

Before autonomous vehicles can take over the roads, they will need to be able to interact with all the aspects of roadways, including pedestrians, other vehicles, and lanes. As early as 1991, a driver agent system named "Ulysses" had been developed in simulation (Reece & Shafer, 1991). While most systems currently under development for implementation on real vehicles are geared toward assisting human drivers, many of the technologies created through these efforts are applicable to the creation of a completely autonomous driver agent. Such a successful driver agent needs to do three main things: detect other entities on the road, keep its vehicle in the lane, and maintain safe distances from other vehicles. Fortunately, each of these three subtasks currently attracts an extensive amount of research.

6.1.1 Object Detection and Tracking

A fully autonomous vehicle must be able to reliably detect, classify, and track various objects that may be in the roadway. From pedestrians and bicycles to cars and trucks, autonomous vehicles will require robust sensors that can monitor the world around them in all manner of lighting conditions and weather. Without such abilities, any amount of higher reasoning a driver agent can do is irrelevant. Fortunately, researchers are attacking this problem with many techniques.

In 2004, Honda introduced an intelligent night vision system to the Japanese market capable of detecting pedestrians (Liu & Fujimura, 2003). The system uses two far-IR (FIR) cameras on the front of the vehicle to detect heat-emitting objects beyond the range illuminated by the vehicle's headlights. The two cameras allow the system to obtain distance information about the detected pedestrians and can then warn the driver. DaimlerChrysler is developing a similar system that also extrapolates the trajectories of classified objects in order to predict possible outcomes sooner (Gavrila, Giebel, & Munder, 2004). Mählisch et al. (2005) have developed a sensor fusion technique that can glean information about pedestrians reliably even from low-resolution images.

The Ford Motor Company has been investigating how to track vehicles using both color and shape information (She, Bebis, Gu, & Miller, 2004). Gepperth et al. (2005) have demonstrated that with only gray-valued videos (no color), a two-stage (initial detection and confirmation) mechanism using a simple neural network for confirmation can reliably and quickly classify other vehicles.

Vehicle and pedestrian classification and tracking is a well-studied area of IV research that is progressing quickly. A glance at any IV-related conference or symposium will reveal a plethora of articles aimed at using lidar, FIR, normal video, and any combination of these sensors with algorithms like Kalman filters, particle filters, and neural networks to track and classify other objects on the road.

6.1.2 Lane Following

As with pedestrian and vehicle detection and tracking, lane following is a heavily studied area of IV research. Varying from passive lane- and road-departure warning systems (LDWS/RDWS) to active lane keeping assistance (LKA), many systems are already showing up in production vehicles.

As far as RDWS go, Kohl et al. (2006) have used neuroevolution to create a warning system that can warn drivers of both road departure and impending crashes with other vehicles. The system was tested both in simulation and with a robotic vehicle. This work is sponsored by Toyota, who have also currently have an LDWS on the market in Japan. This system is unique in that it uses a rear-facing camera to predict and warn of impending lane departures. While LDWS and RDWS promise extensive benefits to drivers, they only warn of imminent road and lane departures, and do not provide information on what specific action should be taken. Autonomous vehicles will need to ensure they do not reach a point where a lane or road departure is imminent.

Lane keeping, on the other hand, provides and executes actions. For example, the "No Hands Across America" project in 1995 drove a vehicle 2,849 miles from Pittsburgh to Los Angeles. For 98.2% of the journey, the vehicle steered itself (Pomerleau, 1993). More recent projects have concentrated on making such systems robust to varying speed, inclement weather and poor lighting conditions such as beneath overpasses and in tunnels. Wu et al. (2005) have proposed and tested a vision-based lane-keeping system that can operate at varying speed while providing smooth human-like steering. Watanabe and Nishida (2005), working for Toyota, have developed a lane detection algorithm specifically designed for steering assistance systems that is extremely robust to varying road conditions and lighting.

While several LKA systems are on the market in Japan, these systems are not intended to allow autonomous driving. Rather, they attempt to reduce driver fatigue and make turning more stable (Bishop, 2005). Production systems that allow autonomous steering are almost invariably based on specially painted lines and are limited to special vehicles on closed courses.

Even without the benefit of explicitly designated lanes, autonomous vehicles can keep themselves on the roadway. In the 2005 DARPA Grand Challenge (DARPA, 2007), the winning vehicle, “Stanley”, used a technique fusing short-range laser range finders with long-range video cameras to follow a rough dirt path. First, the vehicle found smooth areas in front of it using the laser range finders. Then it mapped this information onto video images from forward-facing cameras. By determining the color of the area in the image corresponding to the smooth areas found by the laser range finder, Stanley was able to extrapolate using a flood-fill-type algorithm to find which areas of the video image were on the dirt path (NOVA, 2006). Ramström and Christensen (2005) achieved a similar goal by using a strategy based on a probabilistic generative model.

6.1.3 Adaptive Cruise Control

If lane-keeping systems represent the main lateral component of an autonomous vehicle’s driver agent, then adaptive cruise control (ACC) is the main longitudinal component. ACC allows a vehicle to maintain a safe following distance and can react quicker than a human driver in the case of sudden deceleration by the vehicle in front. ACC systems are already available on the market—DaimlerChrysler’s Mercedes-Benz S-class, for example, comes with a system that will automatically apply the brake if it detects that the driver is not slowing sufficiently fast. Jaguar, Honda, and BMW offer similar systems. Nissan and Toyota have recently begun offering “low-speed following” systems, which can follow other vehicles in slower, denser, urban traffic scenarios (Bishop, 2005). ACC relies on robust sensing and uses radar, lidar, and traditional machine vision algorithms. By combining various “flavors” of ACC — low speed, high speed, etc.—an agent could control the longitudinal motion of a vehicle in all situations. Recently, the notion of *cooperative* adaptive cruise control (CACC) has emerged (Laumônier, Desjardins, & Chaib-draa, 2006). This concept goes much further toward realizing the goal of fully autonomous vehicles. By allowing vehicles to collaborate and take advantage of the precision of autonomous driver agents, vehicles can use the existing road space much more efficiently.

6.2 Intersection Collision Avoidance

To date, much of the ITS work relating to intersections has focused on Intersection Collision Avoidance (ICA). This work seeks to warn the driver when the vehicle may be entering an intersection unsafely. With the aid of high-precision digital maps and GPS equipment, the vehicle detects and classifies the state of the traditional signaling systems placed at the intersection (Lindner, Kressel, & Kaelberer, 2004). ICA systems typically do not take any action on behalf of the driver, but simply provide a visual or auditory warning.

Rasche and Naumann (1997, 1998, 1997) have worked extensively on decentralized solutions to intersection collision avoidance problems, including those involving autonomous vehicles. This work is very similar to ours in that it uses “potential points of collision” to restrict access to the intersection. Only one vehicle may occupy any potential point of collision at a time. Vehicles attempt to obtain a token (similar to a token-ring in computer networking) for each point needed to cross the intersection. Once a vehicle has all the necessary tokens, it may cross. Rasche and Naumann’s system also includes a priority model that allows emergency vehicles to cross more quickly and prevents deadlocks amongst normal vehicles. However, the system fails to satisfy several of our desiderata. It does not make any guarantees, nor do the authors provide any results regarding the efficiency of the system as compared to a traditional system. Furthermore, the distributed algorithm is not shown to be resilient to unreliable communication. The authors also do not provide any insight into how the system could be adapted to work with a mixed human/autonomous vehicle population. The most striking difference, however, is that the mechanism does not seem to have any notion of planning ahead. Tokens for the potential points of collision are either taken or not taken—a vehicle can not seek to obtain a token for some point in the future, thus allowing it to proceed toward the intersection without slowing down while other vehicles have the tokens.

In the context of video games and animation, Reynolds (1999) has developed autonomous steering algorithms that attempt to avoid collisions in intersections that do not have any signaling mechanisms. Such a system would have the enormous advantage of not requiring any special infrastructure or agent at the intersection—vehicles equipped with such algorithms could operate at any intersection. Unfortunately, the two main drawbacks of the system make it unsuitable for use with real-life traffic. First, the algorithm does not let the agent choose which path it will take out of the intersection; a vehicle may even find itself exiting the intersection the same way it came in, due to efforts to avoid colliding with other vehicles. Second, the algorithm only *attempts* to avoid collisions—it does not make any guarantees about safety.

Cooperative intersection collision avoidance is a form of *cooperative vehicle-highway system* (CVHS) in which the intersection is allowed to participate in the ICA problem. ICA systems contained entirely in individual vehicles cannot account for gaps in sensor views or other sources of incomplete information. Thus, a CVHS approach is required. As with many other ITS technologies, production systems still assume a human driver and attempt to warn them when a violation is about to occur, or in some cases, punish them after the fact, as with cameras that detect when a vehicle has run a red light and automatically issues the driver a citation. The U.S. Department of Transportation is sponsoring several ICA projects including both infrastructure-only and cooperative approaches (USDOT, 2003). The intention is to first deploy the infrastructure-only systems, and then as the market penetration of ICA-equipped vehicles increases, to roll out the cooperative systems. Significant work on ICA is also underway in Japan (Bishop, 2005).

While these systems are a large step toward enabling autonomous vehicles to take to the roads, none are designed to work specifically with autonomous vehicles. With the exception of the algorithm designed for games, each assumes both a human driver and traditional signaling systems—a clumsy, inefficient interface that will find itself all but obsolete due to autonomous vehicle technology.

6.3 Optimizing Traffic Signal Timing

The vast majority of deployed technology for intersection control involves calibrating the timing of traditional traffic lights in order to create a "wave of green" such that once vehicles reach one green light, they continue through all subsequent intersections without having to stop. Unfortunately, in practice, such waves tend to be sporadic and short-lived due to rapidly changing traffic patterns. However, they do offer substantial benefits compared to systems without this coordination.

TRANSYT, the Traffic Network Study Tool, is an off-line system that, given average traffic flows, can determine optimum fixed-time coordinated traffic signal timings (Robertson, 1969). TRANSYT requires extensive data gathering and analysis, but is used very heavily all over the world. Unfortunately, this system is very brittle because it does not have the ability to react to unusual changes in traffic flow. For example, at the end of a major sporting event, thousands of vehicles may all be attempting to cross an intersection in a direction which under normal circumstances is rarely used. Because the light timings are set up to reflect these normal circumstances, the length of time for which the departing vehicles get a green light may be significantly less than the cross traffic, of which there may be little.

SCOOT, the Split, Cycle, and Offset Optimisation Technique, represents an advancement over TRANSYT (Hunt, Robertson, Bretherton, & Winton, 1981). SCOOT is an on-line adaptive traffic control system that can react to changes in traffic levels, give priority to vehicles such as buses, and even estimate vehicle emissions. While SCOOT has been shown to reduce traffic delays by an average of 20% over systems like TRANSYT, it still relies on traditional signaling systems and vehicles. Furthermore, SCOOT requires reliable traffic data in order to adapt, and thus may be slow to react to changes in traffic flow.

6.4 MAS and Traffic

Automobile traffic is a great example of a multiagent system, and it is not surprising that there is a lot of research into modelling and studying traffic using multiagent techniques. Many of these approaches consider systems consisting only of traffic-light-controlling agents or driver agents, as opposed to a heterogeneous multiagent system with many kinds of agents. Nevertheless, many of the ideas involved could potentially be adapted to work within the framework of the reservation system.

6.4.1 Cooperative Traffic Signals

Much of MAS traffic research focuses on improving current technology (systems of traffic lights). For example, Roozemond (1999) allows intersections to act autonomously while sharing the data they gather. The intersections then use this information to make both short- and long-term predictions about the traffic and adjust accordingly. This strategy attempts to overcome one of the weaknesses of SCOOT: the need for large amounts of reliable traffic data. If multiple intersections can share data, each intersection will get a more accurate picture of the current traffic situation.

Bazzan (2005) has used a decentralized approach combining MAS and evolutionary game theory. The approach models each intersection as an individually-motivated agent which must focus not only on local goals (getting vehicles through the intersection), but also on global goals (reducing travel times for all vehicles). Both Bazzan and Roozemond's techniques still assume traditional signaling mechanisms and human drivers.

6.4.2 Platoons

In addition to multi-intersection systems, multi-vehicle systems are the focus of a lot of research. Much of this research centers on creating *platoons* of vehicles in order to minimize the effects of stop-and-go driving. Consider a line of cars stopped at a red light. When the light turns green, the first car begins to move. Eventually, the car behind it notices that it has enough space to accelerate as well. Some time later, the vehicle at the back of the line will begin to move, but this may be too late to actually get through the intersection during the current green phase of the light. If, on the other hand, all the vehicles were to simultaneously and uniformly accelerate, more vehicles could make it through each green phase, because the vehicles would more efficiently use the space-time available to them to cross the intersection.

Clement (2002) has proposed a model called "Simple Platoon Advancement" (SPA), which addresses this exact problem. SPA boasts the ability to get nearly twice as many vehicles through a green light (increasing the light's throughput) as compared to normal human drivers, in addition to any safety and delay benefits associated with automated control. One the vehicles are through the intersection and dispersed to safe following distances, control is returned to the human driver.

Hallé and Chaib-draa (2005) have used the platoon approach to facilitate collaborative driving in general. They allow vehicles, which are controlled by separate agents, to form such platoons, with varying degrees of autonomy. Vehicles merge and split with platoons using carefully crafted maneuvers, during which each vehicle in the platoon has a specific responsibility. They present both centralized version, in which a *master* vehicle gives orders to the rest of the platoon, and a decentralized version, in which social laws dictate each agent's role, while the platoon's leader acts only as a representative to other platoons.

Both platooning systems assume automated control of vehicles, but use ordinary traffic lights for intersection control. By using platoons, these methods attempt to solve a problem inherent in the traffic lights themselves—they are designed for humans to use, and are not well suited to automated vehicle control. The work presented in this article attempts to free autonomous vehicles from the control of traffic lights and instead design a new system that specifically utilizes the capabilities of fully autonomous vehicles.

6.4.3 History-Based Traffic Control

Taking a different approach to intersection control, Balan and Luke (2006) use a history-based method to maximize *fairness* (all vehicles experience similar delays) as opposed to efficiency (the average vehicle experiences short delays). Under this paradigm, vehicles which have historically (previously in their journey) experienced long delays should be more likely to experience shorter delays at subsequent intersections. In addition to being a multi-intersection approach, this method uses a marketplace model involving a system of *credits* that can be given and taken in exchange for shorter and longer delays, respectively. Coordination at individual intersections is still done with traditional traffic lights, the timings of which are part of the mechanism. Interestingly, the fairness approach actually yields results that are also reasonably efficient.

6.5 Machine Learning and Traffic

Abdulhai et al. (2003) have used Q-learning, a simple, yet powerful form of reinforcement learning, to do on-line adaptive signal control. In the work, the authors explore both an isolated intersection as well as a linear chain of intersections. They demonstrate that Q-learning can significantly reduce delays for vehicles and quickly adapt to changing traffic patterns. Bull et al. (2004) have shown how Learning Classifier Systems (LCS) can also make traditional traffic signals more efficient. Wiering (2000) has demonstrated that multiagent, model-based reinforcement learning can also be used to optimize signal timings in more complex networks of intersections.

While not focusing on intersections, Moriarty and Langley (1998) have shown that reinforcement learning—specifically neuro-evolution—can train efficient driver agents for lane, speed, and route selection during freeway driving, all of which are critical components for a fully autonomous vehicle. Additionally, many of the object tracking and detection examples mentioned previously use neural networks to classify objects.

6.6 Physical Robots

On real autonomous vehicles, Kolodko and Vlacic (2003) have created a small-scale system for intersection control which is very similar to the granularity-1 FCFS policy. The authors developed the mechanism for small Cooperative Autonomous Mobile Robots (CAMRs), which are about 30 cm in diameter and have a top speed of 10 cm/s. The CAMRs were programmed to follow Australian traffic laws, and communicate with several different types of messages. Once demonstrated on the CAMRs, the mechanism was scaled up to use Imara vehicles, which are much larger (capable of carrying two human passengers) and faster (top speed of 30 km/h). The system is completely distributed and does not require extensive infrastructure at the intersection. However, it does assume that all vehicles cooperate with one another.

6.7 Safety Analysis

Section 5 includes a failure-mode analysis for our proposed intersection control mechanism. To the best of our knowledge, this is the first study of the impact of any other such autonomous intersection protocol on driver safety. However, there is an enormous body of work regarding safety properties of traditional intersections. This includes the general—correlating traffic level and accident frequency (Sayed & Zein, 1999) and analyses of particular types of intersections (Bonneson & McCoy, 1993; Harwood, Bauer, Potts, Torbic, Richard, Rabbani, Hauer, Elefteriadou, & Griffith, 2003; Persaud, Retting, Gardner, & Lord, 2001)—as well as plenty of more esoteric work, such as characterizing the role of Alzheimer's Disease in intersection collisions (Rizzo, McGehee, Dawson, & Anderson, 2001). However, because it concerns only human-operated vehicles, none of this work is particularly applicable to the setting we are concerned with.

7. Conclusion and Future Work

The reservation system as presented is a large step toward easing our traffic woes, in terms of both wasted time and injury or loss of life. However, substantial work must still be done before the system is ready to deploy. Some of this work represents possible future directions for this line of research. For example, more detailed studies of the safety properties of the system—how it reacts to various failures and whether the effects of those failures can be mitigated—are required. Another area ripe for improvement is the intersection manager. A manager that can switch among several different policies, learning from reservation histories which policy is best suited to particular traffic conditions, could significantly improve performance. Furthermore, a light model that could react not only to traffic conditions, but also to the presence of individual vehicles, might better be able to exploit the abilities of autonomous vehicles, without adversely affecting human drivers. Framing the intersection as a marketplace and space-time as a commodity could allow the system to handle vehicle priorities more intelligently or allow driver agents to exchange a long wait on one day for quick passage on some later, more important day. Finally, the driver agent itself may be able to benefit from some machine learning techniques, perhaps learning to make more accurate reservations and thus needing to cancel less frequently.

This article makes three main contributions. First, it defines the problem of autonomous intersection management, including a set of desiderata by which potential solutions can be evaluated. Second, it presents a framework that can meet all of these desiderata, and an algorithm (FCFS) that shows the advantages of the framework over current intersection control methods. Third, it demonstrates how the framework can be extended to allow human-driven (not autonomous) vehicles to use the system, while still exploiting the abilities of the autonomous vehicles to increase throughput and subsequently decrease delays.

Getting from where we are today to a future in which humans are no longer burdened with the mundane yet dangerous task of piloting automobiles will involve a vast amount of work in many different disciplines. While it does not extensively address the engineering or societal challenges involved in building and deploying such a system, this article suggests that it is both algorithmically feasible and worthwhile (in terms of decreasing delay) to do so.

Acknowledgments

This research is supported by NSF CAREER award IIS-0237699, and experiments were carried out on machines provided by NSF grant EIA-0303609.

Appendix A. Simplified Laser Range Finder

This appendix describes an implementation detail of the driver agent's sensor model. Recall from Section 3.1.1 that each driver has access to a set of simulated external sensors. In this set is a simplified laser range finder, which is intended to give the agent the same type of information as an actual laser range finder, but without the expensive computation required to fully simulate the sensor. Instead, the simplified laser range finder sensor examines each vehicle within sensor range and determines which is closest to the front of the sensing vehicle. Then, it records the point on that vehicle that is closest to the sensing vehicle and provides the distance and angle to this point.

Modern laser range finders and distance sensors can provide a large amount of distance and angle data to a mobile agent. In a real life setting, this information would definitely prove useful in fine-tuning a driver agent. However, in a simple simulation, we must process sensor information for *all* vehicles simultaneously, and accurately simulating a full laser-range finder is not feasible. Thus, we use this simple, yet pertinent sensor reading which the driver agent can use to control its actions with respect to the other vehicles. A purely straight-ahead sensor suffices when vehicles are traveling only in straight lines. However, when a vehicle turns, it must also take into account what is going on in the direction in which it is turning. To complicate matters, when a vehicle is turning it must still take into account what is going on directly in front of it because at any point it might straighten its wheels and continue on its current heading. A sensor that points in the same direction as the wheels will not be sufficient because vehicles coming out of turns may run into vehicles ahead of them. Instead, our sensor's scope widens in the direction of the turn, while narrowing slightly from the other side. Figure 26 shows a scenario that demonstrates the concept. A testament to the sensor's usefulness, vehicles equipped with only the sensor (i.e. no intersection manager is present) are able to avoid many collisions in the intersection, even with moderate amounts of traffic.

Appendix B. Communication Protocol

Section 3.2 gave a brief introduction to the communication protocol used by the agents in the reservation system. In this appendix, we specify the protocol with much greater detail. The protocol consists of several message types for each kind of agent, as well as some rules governing when the messages should be sent and what sorts of guarantees accompany them. In this section we present those aspects that are essential to understanding the remainder of the article.

B.1 Message Types

The vehicles and intersection manager are each restricted to a few types of messages with which they must coordinate.

B.1.1 Vehicle → Intersection

There are four types of messages that can be sent from vehicles to the intersection.

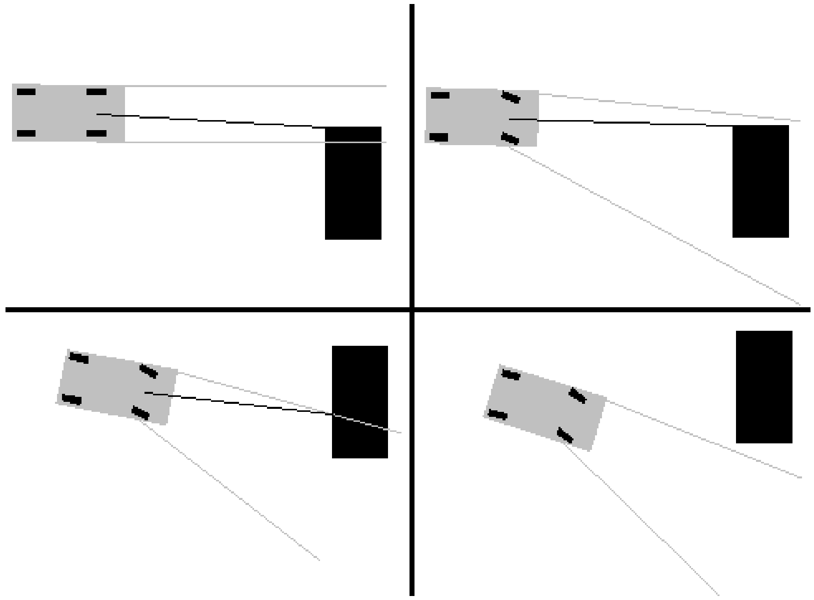


Figure 26: A depiction of the sensor model for the driver agents. The sensor is focused between the gray lines and does not provide information outside of them. The black line represents the reading provided to the driver agent.

1. Request — This is the message a vehicle sends when it does not have a reservation and wishes to make one. It contains the properties of the vehicle (ID number, performance, size, etc.) as well as some properties of the proposed reservation (arrival time, arrival velocity, type of turn, arrival lane, etc.). The message also communicates the vehicle's status as an emergency vehicle (in an emergency situation). In practice, this would be implemented using a secure method such that normal vehicles could not impersonate emergency vehicles. Such methods are well understood and the details of the implementation are beyond the scope of this research.

   This message has 15 fields:

   `vehicle_id` — a unique identifier for the vehicle.

   `arrival_time` — the absolute time at which the vehicle agrees to arrive at the intersection.

   `arrival_lane` — a unique identifier for the lane in which the vehicle will be when it arrives at the intersection.

   `turn` — which way the vehicle will turn when it reaches the intersection.

   `arrival_velocity` — the velocity at which the vehicle agrees to be traveling when it arrives at the intersection.

   `maximum_velocity` — the maximum velocity at which the vehicle can travel.

   `maximum_acceleration` — the maximum rate at which the vehicle can accelerate.

`minimum_acceleration` — the minimum rate at which the vehicle can accelerate (i.e. negative number representing maximum deceleration).

`vehicle_length` — the length of the vehicle.

`vehicle_width` — the width of the vehicle.

`front_wheel_displacement` — the distance between the front of the vehicle and the front axle.

`rear_wheel_displacement` — the distance between the front of the vehicle and the rear axle.

`max_steering_angle` — the maximum angle to which the front wheels can be turned for the purposes of steering.

`max_turn_per_second` — the rate at which the vehicle can turn its wheels.

`emergency` — whether or not this is an emergency vehicle in an emergency situation.

2. Change-Request — This is the message a vehicle sends when it has a reservation, but would like to switch to a different set of parameters. If the new parameters are not acceptable to the intersection, the vehicle may keep its old reservation. It is identical to the request message, except that it includes a unique reservation ID for the reservation the vehicle currently has.

   This message is identical to the Request message, except for one added field:

   `reservation_id` — an identifier for the reservation to be changed.

3. Cancel — This is the message a vehicle sends when it no longer desires its current reservation.

   It has 2 fields:

   `vehicle_id` — a unique identifier for the vehicle.

   `reservation_id` — an identifier for the reservation to be cancelled.

4. Done — This message is sent when the vehicle has completed its traversal of the intersection. While it communicates the same information as the Cancel message, there may be behavior tied to the Cancel message which should not occur when a vehicle successfully completes the trip across the intersection. Additionally, this message could be extended in order to communicate statistics for each vehicle, which could then be recorded in order to analyze the performance of the intersection manager. This message can be used to collect statistics for each vehicle, which can be recorded in order to analyze and improve the performance of the intersection manager.

   It has 2 fields:

   `vehicle_id` — a unique identifier for the vehicle.

   `reservation_id` — an identifier for the reservation that was just completed.

B.1.2 Intersection → Vehicle

There are four types of messages that can be sent from the intersection to the individual vehicles.

1. Confirm — This message is a response to a vehicle's Request (or Change-Request) message. It does not always mean that the parameters transmitted by the vehicle are acceptable. It could, for example, contain a counter-offer by the intersection. The reservation parameters in this message are implicitly accepted by the vehicle, and must be explicitly cancelled if the driver agent of the vehicle does not approve. Note that this is safe even with faulty communication—the worst that can happen is that the intersection reserves space that does not get used. Included in the message are acceleration constraints determined by the intersection. This is just a list of rates and durations. How the list is created depends on the intersection manager. However, the vehicle's safety must be guaranteed if it adheres to the list.

   This message has 7 fields:

   `reservation_id` — a unique identifier for the reservation just created.

   `arrival_time` — the absolute time at which the vehicle is expected to arrive.

   `early_error` — the tolerable error (early) in arrival time for the vehicle.

   `late_error` — the tolerable error (late) in arrival time for the vehicle. Note that the intersection manager must assume that the car could arrive and traverse the intersection at any time within the resulting bounds

   `arrival_lane` — a unique identifier for the lane in which the vehicle should be when it arrives at the intersection.

   `arrival_velocity` — the velocity at which the vehicle is expected to be traveling when it arrives at the intersection. A negative number signifies that any velocity is acceptable.

   `accelerations` — a run-length encoded description of the expected acceleration of the vehicle as it travels through the intersection. Here, a run-length encoded description is a sequence of order pairs of *acceleration* and *duration*—starting with the instant the vehicle enters the intersection, it should maintain each *acceleration* for the *duration* with which it is paired. If the sequence is empty, any accelerations are acceptable.

2. Reject — By sending this message, an intersection can inform a vehicle that the parameters sent in the latest Request (or Change-Request) were not acceptable, and that the intersection either could not or did not want to make a counter-offer. This message also indicates whether or not the rejection was because the reservation manager requires the vehicle to stop at the intersection before entering. This lets the driver agent know that it should not attempt any more reservations until it reaches the intersection.

   This message has 1 field:

`stop_required` — a boolean value indicating whether the vehicle must first come to a full stop before entering the intersection.

3. Acknowledge — This message acknowledges the receipt of a Cancel or Done message.

   It has 1 field:

   `reservation_id` — a unique identifier for the reservation just cancelled or completed.

4. Emergency-Stop — This message is only sent when the intersection manager has determined that a collision or similar problem has occurred in the intersection. This message informs the receiving driver agent that no further reservation requests will be granted, and if possible, the vehicle should attempt to stop instead of entering the intersection, even if it has a reservation. The specifics of how this message is used are discussed in Section 5.2.2. This message has no fields, as it only communicates a single bit of information.

B.1.3 Vehicle → Vehicle

There is currently no protocol for communication between vehicles.

B.2 Protocol Actions

In addition to message types, the agents involved (the vehicles and the intersection) must obey a set of rules. These are not entirely unlike the rules that human drivers follow when driving.

B.2.1 Vehicle Actions

These are the rules that the vehicles are expected to follow in order to allow the intersection to function efficiently.

1. A vehicle may not enter the intersection without a reservation.
2. If a vehicle is going to cross the intersection, it must do everything reasonable within its power to cross in accordance with the parameters included in the most recent Confirm message it has received from the intersection.
3. If a vehicle sends another message before the intersection manager has sent a response, the intersection manager may choose to ignore it. Thus, a vehicle should only send a message if it has received a response to its previous message.
4. If a vehicle has not yet entered the intersection and does not have a reservation, it may send a Request message. If it has not yet entered the intersection and does have a reservation, it may send either a Change-Request or Cancel message. If it sends any of these messages when it is not allowed to, the intersection may choose to ignore them.
5. If a vehicle has a reservation and has successfully crossed the intersection, it may send a Done message.

6. If a vehicle receives a Confirm message, it is considered to have a reservation.

B.2.2 Intersection Actions

These are the rules representing the obligations the intersection manager is expected to fulfill.

1. When an intersection receives a Request message, it must respond with either a Confirm or a Reject message. If it responds with a Confirm message, it is guaranteeing that no cross-traffic will interfere with the vehicle if it crosses the intersection in accordance with the parameters in the message.

2. When an intersection receives a Change-Request message, it must respond with either a Confirm or a Reject message. If it responds with a Confirm message, it is guaranteeing that no cross-traffic will interfere with the vehicle if it crosses the intersection in accordance with the parameters in the message. Any previous guarantees are nullified.

3. When an intersection receives a Cancel message, it must respond with an Acknowledge message. Any guarantee that had been made to the sending vehicle is nullified.

Appendix C. Driver Agent

As stated in Section 3.3, the main focus of this work is on improving the framework and algorithms for intersection control. However, in order to do this we require some sort of driver agent. Furthermore, the efficiency of the reservation framework depends on driver agents being reasonably intelligent, which is non-trivial. This appendix describes the driver agent implementation used in our experiments.

Containing both behaviors to control turning vehicles as well as optimizations to increase peformance of the system overall, the driver agent represents the single most intricate component of the reservation mechanism. Algorithm 3 gives a high-level pseudocode description of the driver agent.

C.1 Lane Following

Given the model of lanes, each driver must be able to drive the vehicles in those lanes. We accomplish this by means of a lane following behavior that acts only by modifying the steering angle of the vehicle. This behavior is entirely independent of the rest of the agent's behavior, which controls the vehicle's acceleration and communicates with the intersection manager. This behavior is active at all times—the vehicle is always attempting to stay in its current lane. The lane-following behavior is designed to be robust to sudden lane reassignment, and this is how both turning and lane changing are implemented: the driver agent simply changes which lane is its "current" lane, and the lane-following behavior steers the vehicle into the correct lane. This process is entirely smooth, provided the vehicle is traveling at a reasonable velocity—a condition enforced by other parts of the driver agent's behavior.

Because lanes are modeled as directed line segments, the lane-following behavior attempts to keep the vehicle evenly straddling the lane. The line segment represents the middle of the lane, and thus this condition is equivalent to keeping the vehicle centered in the lane. The driver agent accomplishes this by turning the front wheels toward a point on the segment. This point, which we call the *aim point* is farther along the segment than the vehicle. The aim point is computed by first projecting the point at the front and center of the vehicle onto the line segment, and then displacing this point in the direction of the line segment by an amount we call the *lead distance*. For the most part, the lead distance is proportional to the velocity of the vehicle. The proportion is smaller inside the intersection so that vehicles will pull more strongly into their new lane if they are turning—they must be entirely in the correct lane before they leave the intersection so that they do not collide with other vehicles outside the intersection. The proportional lead distance is necessary because otherwise at high velocities, the required steering angle may change faster than the driver agent can steer, resulting in either wildly erratic steering or the vehicle driving in circles. The lead distance also has a minimum value of 1 meter. If the lead distance gets too small, the effect is the same as if the velocity were too large—by ensuring the aim point is at least a meter farther down the lane, we can ensure that the vehicle will end up in a stable configuration traveling in the proper direction. Figure 27 depicts how the driver agent determines the lead distance (and subsequent aim point) for different velocities.

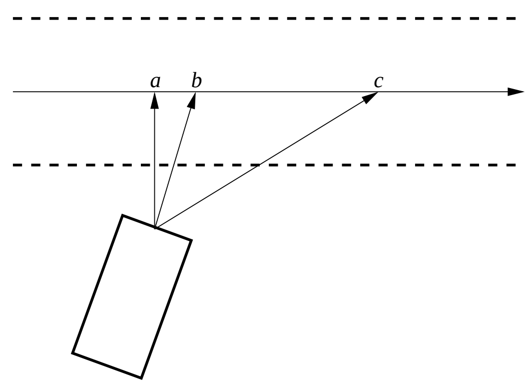


Figure 27: A vehicle is attempting to follow the lane. To do so, it first calculates the point that represents the projection of its position onto the directed line segment running down the center of the lane (a). Then, depending on its velocity, it displaces the resulting point in the direction of travel by a small or large amount to obtain the point at which it should aim its front wheels. For low velocities, the point will not be displaced much—only enough to ensure the vehicle moves in the correct direction (b). For higher velocities, the aim point must be farther along the lane, so that the vehicle's steering will be more gradual and thus more stable (c).

This method of lane following is only one possible method, and was selected because it is sufficient for our purposes. Furthermore, the reservation system's functionality does not depend on the driver agent using this particular method, but will work with any method, provided the driver agent turns within some mutually understood constraints.

C.2 Optimistic and Pessimistic Driver Agents

A naïve driver agent can perform poorly when, for example, it makes a reservation while stuck behind a slower-moving vehicle. If the vehicle in front eventually accelerates, it would ideally accelerate as well (possibly switching to an earlier reservation).

To account for situations like this, we introduce the notion of an *optimistic* or *pessimistic* driver agent. An optimistic agent makes a reservation assuming it will arrive at the intersection in the minimum possible time. An agent which finds itself no longer stuck behind a slower vehicle will become optimistic and attempt to make a new, earlier reservation. A pessimistic agent assumes it will be stuck at its current velocity until it reaches the intersection. If an agent has to cancel its reservation because there is no way for it to arrive on time, it becomes pessimistic. Due to the relatively infrequent and smooth transitions through these “moods”, our driver agent can take advantage of improving circumstances without causing it to send excessive numbers of messages when things change.

As shown in Figure 2, the addition of optimism and pessimism to the driver agent reduced both the average number of reservations made as well as the average number of messages transmitted. As expected, the effect was less pronounced for lower amounts of traffic.

	Messages	**Reservations**
With	560.85	165.89
Without	5.97	1.02

Table 2: For a moderate amount of traffic, the average number of messages sent and reservations made by driver agents with and without the optimism/pessimism heuristic.

C.3 Estimating Time To Intersection

A driver agent’s estimate of how long it will take to get to the intersection must be very precise so that vehicles can arrive on time for their reservations. If, at this point, the vehicle is not certain whether or not it will arrive on schedule, it cannot safely continue. The pessimistic driver agent simply divides the distance to the intersection by its current velocity—it assumes it will not be able to accelerate. An optimistic driver first determines what its velocity will be when it arrives. If it is turning, for example, this velocity may be lower than the speed limit. Otherwise, it may be limited by the amount the vehicle can accelerate before reaching the intersection. It then computes the minimum possible time to reach the intersection at that velocity, that is, it assumes it can accelerate as much as possible before decelerating to its arrival velocity.

Algorithm 3 The driver agent behavior. All driver agents are initialized as optimistic.

```
determine aim point and attempt to point wheels at it
t ← current time
if Velocity is below speed limit then
   Accelerate
if Before the intersection then
   if Optimistic then
      t_i ← optimisitic estimate of time to intersection
   else
      t_i ← pessimistic estimate of time to intersection
   if Do not have a reservation then
      if t + t_i is after scheduled arrival then
         Cancel reservation
         become pessimistic
      else if t + t_i is significantly before scheduled arrival then
         Become optimistic
         Attempt to change reservation to earlier time
   else
      Try to make reservation according to t_i
      if Reservation request rejected then
         Decelerate
else if In the intersection then
   Set acceleration according to parameters of reservation
if Not in the intersection and less than 1 second behind car in front then
   Decelerate
```

References

Abdulhai, B., Pringle, R., & Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, *129*(3), 278–285.

Alvarez, L., & Horowitz, R. (1997). Traffic flow control in automated highway systems. Tech. rep. UCB-ITS-PRR-97-47, University of California, Berkeley, Berkeley, California, USA.

Balan, G., & Luke, S. (2006). History-based traffic control. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 616–621, Hakodate, Japan.

Bazzan, A. L. C. (2005). A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, *10(2)*, 131–164.

Bishop, R. (2005). *Intelligent Vehicle Technology and Trends*. Artech House.

Bonneson, J. A., & McCoy, P. T. (1993). Estimation of safety at two-way stop–controlled intersections on rural highways. *Transportation Research Record*, *1401*, 83–89.

Bull, L., Sha'Aban, J., Tomlinson, A., Addison, J. D., & Heydecker, B. G. (2004). Towards distributed adaptive control for road traffic junction signals using learning classifier systems. In Bull, L. (Ed.), *Applications of Learning Classifier Systems*, pp. 276–299. Springer.

Clement, S. (2002). The SPA model with smooth acceleration. In *24th Conference of Australian Institutes of Transport Research (CAITR-2002)*, Sydney, Australia.

DARPA (2007). The DARPA urban challenge.. `http://www.darpa.mil/grandchallenge`.

Dresner, K., & Stone, P. (2004). Multiagent traffic management: A reservation-based intersection control mechanism. In *The Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 530–537, New York, NY, USA.

Dresner, K., & Stone, P. (2006). Multiagent traffic management: Opportunities for multiagent learning. In K. Tuyls et al. (Ed.), *LAMAS 2005*, Vol. 3898 of *Lecture Notes In Artificial Intelligence*, pp. 129–138. Springer Verlag, Berlin.

Gavrila, D. M., Giebel, J., & Munder, S. (2004). Vision-based pedestrian detection: The PROTECTOR+ system. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2004)*, Parma, Italy.

Gepperth, A., Edelbrunner, J., & Bücher, T. (2005). Real-time detection and classification of cars in video sequences. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV2005)*, pp. 625–631, Las Vegas, NV, USA.

Hallé, S., & Chaib-draa, B. (2005). A collaborative driving system based on multiagent modelling and simulations. *Journal of Transportation Research Part C (TRC-C): Emergent Technologies*, *13*, 320–345.

Harwood, D. W., Bauer, K. M., Potts, I. B., Torbic, D. J., Richard, K. R., Rabbani, E. R. K., Hauer, E., Elefteriadou, L., & Griffith, M. S. (2003). Safety effectiveness of intersection left- and right-turn lanes. *Transportation Research Record*, *1840*, 131–139.

Hatipo, C., Redmill, K., & Ozguner, U. (1997). Steering and lane change: A working system. In *IEEE Conference on Intelligent Transportation Systems*, pp. 272–277.

Hunt, P. B., Robertson, D. I., Bretherton, R. D., & Winton, R. I. (1981). SCOOT - a traffic responsive method of co-ordinating signals. Tech. rep. TRRL-LR-1014, Transport and Road Research Laboratory.

Johnson, R. C. (2005). Steady pace takes DARPA race. EE Times. Accessed at `http://www.eetimes.com`.

Kohl, N., Stanley, K., Miikkulainen, R., Samples, M., & Sherony, R. (2006). Evolving a real-world vehicle warning system. In *Proceedings of the Genetic and Evolutionary Computation Conference 2006*, Seattle, WA, USA.

Kolodko, J., & Vlacic, L. (2003). Cooperative autonomous driving at the intelligent control systems laboratory. *IEEE Intelligent Systems*, *18*(4), 8–11.

Laumônier, J., Desjardins, C., & Chaib-draa, B. (2006). Cooperative adaptive cruise control: a reinforcement learning approach. In *The Fourth Workshop on Agents in Traffic and Transportation*, Hakodate, Hokkaido, Japan.

Lindner, F., Kressel, U., & Kaelberer, S. (2004). Robust recognition of traffic signals. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2004)*, Parma, Italy.

Liu, X., & Fujimura, K. (2003). Pedestrian detection using stereo night vision. In *IEEE International Conference on Intelligent Transportation Systems*, Shanghai, China.

Mählisch, M., Oberländer, M., Löhlein, O., Gavrila, D., & Ritter, W. (2005). A multiple detector approach to low-resolution FIR pedestrian recognition. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV2005)*, Las Vegas, NV, USA.

Moriarty, D., & Langley, P. (1998). Learning cooperative lane selection strategies for highways. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp. 684–691, Madison, WI. AAAI Press.

National Highway Traffic Safety Administration (2002). Economic impact of U.S. motor vehicle crashes reaches $230.6 billion, new NHTSA study shows. NHTSA Press Release 38-02. `http://www.nhtsa.dot.gov`.

Naumann, R., & Rasche, R. (1997). Intersection collision avoidance by means of decentralized security and communication management of autonomous vehicles. In *Proceedings of the 30th ISATA - ATT/IST Conference*.

Naumann, R., Rasche, R., & Tacken, J. (1998). Managing autonomous vehicles at intersections. *IEEE Intelligent Systems*, *13*(3), 82–86.

Noda, I., Jacoff, A., Bredenfeld, A., & Takahashi, Y. (Eds.). (2006). *RoboCup-2005: Robot Soccer World Cup IX*. Springer Verlag, Berlin.

NOVA (2006). The great robot race.. Originally aired 28 March 2006 on PBS, available online at `http://www.pbs.org/wgbh/nova/darpa`.

Persaud, B. N., Retting, R. A., Gardner, P. E., & Lord, D. (2001). Safety effect of roundabout conversions in the united states: Empirical bayes observational before-after study. *Transportation Research Record*, *1751*, 1–8.

Pomerleau, D. A. (1993). *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers.

Ramström, O., & Christensen, H. (2005). A method for following umarked roads. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV2005)*, pp. 650–655, Las Vegas, NV, USA.

Rasche, R., Naumann, R., Tacken, J., & Tahedl, C. (1997). Validation and simulation of decentralized intersection collision avoidance algorithm. In *Proceedings of IEEE Conference on Intelligent Transportation Systems (ITSC 97)*.

Reece, D. A., & Shafer, S. (1991). A computational model of driving for autonomous vehicles. Tech. rep. CMU-CS-91-122, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA.

Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In *Proceedings of the Game Developers Conference*, pp. 763–782.

Rizzo, M., McGehee, D. V., Dawson, J. D., & Anderson, S. N. (2001). Simulated car crashes at intersections in drivers with Alzheimer disease. *Alzheimer Disease and Associated Disorders*, *15*(1), 10–20.

Robertson, D. I. (1969). TRANSYT — a traffic network study tool. Tech. rep. TRRL-LR-253, Transport and Road Research Laboratory.

Rogers, S., Flechter, C.-N., & Langley, P. (1999). An adaptive interactive agent for route advice. In Etzioni, O., Müller, J. P., & Bradshaw, J. M. (Eds.), *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pp. 198–205, Seattle, WA, USA. ACM Press.

Roozemond, D. A. (1999). Using intelligent agents for urban traffic control systems. In *Proceedings of the International Conference on Artificial Intelligence in Transportation Systems and Science*, pp. 69–79.

Sayed, T., & Zein, S. (1999). Traffic conflict standards for intersections. *Transportation Planning and Technology*, *22*(4), 309–323.

Schonberg, T., Ojala, M., Suomela, J., Torpo, A., & Halme, A. (1995). Positioning an autonomous off-road vehicle by using fused DGPS and inertial navigation. In *2nd IFAC Conference on Intelligent Autonomous Vehicles*, pp. 226–231.

She, K., Bebis, G., Gu, H., & Miller, R. (2004). Vehicle tracking using on-line fusion of color and shape features. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, Washington, DC, USA.

Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, *8*(3), 345–383.

Svenson, O. (1981). Are we all less risky and more skillful than our fellow drivers?. *Acta Psychologica*, *47*(2), 143–148.

Texas Transportation Institute (2004). 2004 urban mobility report.. Accessed at `http://mobility.tamu.edu/ums` in December 2004.

USDOT (2003). Inside the USDOT's 'intelligent intersection' test facility. Newsletter of the ITS Cooperative Deployment Network. Accessed online 17 May 2006 at `http://www.ntoctalks.com/icdn/intell_intersection.php`.

Watanabe, A., & Nishida, M. (2005). Lane detection for a steering assistance system. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV2005)*, pp. 159–164, Las Vegas, NV, USA.

Wiering, M. A. (2000). Multi-agent reinforcement learning for traffic light control. In Langley, P. (Ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'2000)*, pp. 1151–1158.

Wierwille, W. W., Hanowski, R. J., Hankey, J. M., Kieliszewski, C. A., Lee, S. E., Medina, A., Keisler, A. S., & Dingus, T. A. (2002). Identification and evaluation of driver errors: Overview and recommendations. Tech. rep. FHWA-RD-02-003, Virginia Tech Transportation Institute, Blacksburg, Virginia, USA. Sponsored by the Federal Highway Administration.

Wu, S.-J., Chiang, H.-H., Perng, J.-W., Lee, T.-T., & Chen, C.-J. (2005). The automated lane-keeping design for an intelligent vehicle. In *Proceedings of the IEEE Intelligent Vehicle Symposium (IV2005)*, pp. 508–513, Las Vegas, NV, USA.