2012 IEEE/ACM Third International Conference on Cyber-Physical Systems

A Platform for Evaluating Autonomous Intersection Management Policies

Chien-Liang Fok, Maykel Hanna, Seth Gee, Tsz-Chiu Au,
Peter Stone, Christine Julien, and Sriram Vishwanath
University of Texas at Austin
{liangfok,michael.jean,seth.gee,c.julien,theory}@mail.utexas.edu, {chiu, pstone}@cs.utexas.edu

***Abstract*—There is a significant push towards greater vehicular autonomy on roads to increase convenience and improve overall driver experience. To enable this autonomy, it is imperative that cyber-physical infrastructure be deployed to enable efficient control and communication. An essential component of such road instrumentation is intersection management. This paper develops an intersection management platform that provides the sensing and communication infrastructure needed to enable efficient intersection management policies. The testbed, located in a indoor laboratory, consists of an intersection and multiple robotic vehicles that can sense and communicate. Whereas traditional approaches to intersection management rely on simulations, this testbed enables the first realistic evaluation of several intersection management policies. Six simple but practical centralized and distributed policies are evaluated and compared against the current state of the art, i.e., traffic signals and stop signs. Through extensive experimentation, this paper concludes that, in the scenario tested, even a simple coordinated management policy can halve vehicular delay, while improving the aggregate traversal time of the intersection by 169%.**

***Keywords*-Autonomous vehicles, Intersection management**

I. INTRODUCTION

Imagine driving down a deserted street and approaching a traffic signal at a four-way intersection. Suppose there are no obstructions preventing you from seeing cross-traffic. As you approach the light turns red, forcing you to stop despite no conflicting traffic. Throughout the entire red cycle the vehicle is idling — wasting time and energy.

The above scenario, while simplistic, illustrates a fundamental problem with our current vehicular transportation system. In particular, the delays and waits at intersections can be extremely frustrating and inefficient. Increasing the degree of autonomy in how vehicles are scheduled to cross intersections can lead to shorter drive-times and enhanced overall driving experience. Although vehicle autonomy is desirable at all times and not just at intersections, complete autonomy is a highly complex problem that requires considerable effort and thorough testing before it becomes a reality. Indeed, there is a vast and growing body of work on enabling full vehicular autonomy [1], [2]. In this paper, we focus on a critical component in the goal of fully autonomous vehicles—the ability to autonomously and safely navigate intersections.

A majority of existing research on autonomous vehicles focuses on enabling vehicles to operate on current roads, which are designed for human-operated vehicles. These include techniques from machine learning and artificial intelligence to enable vehicles to recognize road markings, road signs, traffic signals, other vehicles and pedestrians, and, in general, the laws of the road. While this is essential for a gradual transition from human-controlled vehicles to autonomous ones, it limits the efficiency gains that can be afforded by the switch since all of these constructs are tailored to the relatively sluggish response times and low communication expressiveness of human drivers (i.e., we honk, "nudge forward", flash the lights, etc.). Going back to the scenario, an autonomous vehicle at an empty stoplight would still be left idling, wasting time and energy, despite the ability to safely cross. Thus, research on vehicular autonomy should be combined with a re-thinking of conventional vehicular-management systems. For example, replacing traffic signals with an intersection manager could enhance intersection throughput. Developing an understanding of whether this throughput increase is indeed possible in a safe and reliable manner is the main goal of this paper.

We reiterate that autonomous intersection management does not require that all vehicles be fully autonomous at all times. It only requires that intersections and vehicles be instrumented such that vehicular management at intersections can be conducted autonomously. Once the intersection is safely navigated, the vehicle can be returned to human-control to perform other complex tasks (such as merging, lane changes etc.). In this way, autonomous intersection management can be restricted to select (critical) intersections that are particularly accident-prone and/or are traffic bottlenecks.

We investigate mechanisms by which road intersections can be improved by instrumenting the intersection with advanced management schemes and the vehicle with the ability to sense properties of the intersection and communicate with a scheduler and/or other vehicles over a wireless network. Instead of forcing every vehicle to stop, as is the case of stop signs, or to cycle among non-conflicting paths through the intersection at predetermined intervals, as is the case with traffic signals, new forms of intersection management entail a system that dynamically reacts to and communicates with traffic, enabling vehicles to cross the intersection with lower delay. We refer to such a system as an *autonomous intersection*.

978-0-7695-4695-7/12 $26.00 © 2012 IEEE
DOI 10.1109/ICCPS.2012.17

Underlying our investigation is the requirement that vehicles detect that they are approaching, entering, and exiting an intersection. Vehicles must coordinate with each other and/or the intersection itself to ensure safe and efficient passage, potentially without stopping or even slowing down. Such ideas have previously been investigated using simulations [3], or mixed reality simulations involving a single vehicle [4]. Our work differs from these in the following ways:

- It is a multi-vehicle robotic testbed located in an instrumented laboratory; our results are based on system measurements and not (simplistic) simulated models for vehicular motion and communication.
- We devise simplified intersection management policies that account for the limits and nuances of a real cyber-physical system. These policies are low in complexity and found to be robust in practice.

By evaluating an actual cyber-physical system, we account for scalability, robustness, safety, and overall performance of these policies in terms of the physical characteristics of the vehicle and surrounding environment, all of which are near-impossible to accurately replicate in a simulator.

This paper is organized as follows. The next section discusses related work. Section III provides the problem definition. It is followed by our approach (Section IV) which also details the intersection management policies we evaluate. Section V presents our implementation, followed by an evaluation and experimental results in Section VI. The paper ends with conclusions and future work in Section VII.

II. RELATED WORK

Research on vehicular autonomy has made significant progress in recent years. This was in part due to a series of robotic car competitions like the *DARPA Grand Challenges* [5]. These competitions accelerated the development of autonomous vehicles to the point where the technical problem of open-road autonomous driving is considered by some to be essentially solved [3]. The non-technical barrier for the adaptation of autonomous vehicles are largely traffic laws and regulations, though this is also being overcome [6].

The vast majority of research on autonomous vehicles focuses on how to ensure they run on existing road infrastructure; there is limited literature on understanding changes to road infrastructure that can facilitate vehicular autonomy. One such project on jointly optimizing autonomous vehicles and road infrastructure is the PATH program, which relies on magnetic markers in the roadway for measuring steering angle and vehicle movements [7]. The Autonomous Intersection Management (AIM) protocol [3], [4], [8] is a vehicle-to-infrastructure (V2I) mechanism in which vehicles request space-time in the intersection for their trajectories prior to arriving at the intersection; a server at the intersection handles these requests, granting or rejecting reservations using a grid-based collision detection scheme. This protocol is enhanced to reduce network traffic and increase safety using spatial-temporal buffers surrounding the vehicles [8]. While AIM is feature rich, it was evaluated either purely through simulations [3] or mixed reality simulations involving one vehicle [4]. AIM is one of the many intersection management schemes that can be evaluated using our testbed. In this paper, we implement several intersection management schemes that have lower computational complexity than (and are sometimes simplified versions of) AIM and demonstrate that such policies increase intersection efficiency over traditional traffic-signal-based schemes.

Vehicle-to-Vehicle (V2V) forms of autonomous intersection management have also been investigated [9], [10]. In this form, no centralized server is required (i.e., there is no single point of failure) and vehicles coordinate in a peer-to-peer fashion when crossing the intersection. Naumann *et al.* investigated a distributed policy that uses virtual "tokens" that a vehicle must possess to cross certain contested areas of the intersection [9] and formally evaluated it using petri-nets. VanMiddlesworth *et al.* developed a protocol that enables vehicles to "call ahead" to reserve space-time in the intersection [10]. Their protocol outperformed the traditional stop sign in light traffic. We implemented a slightly simplified version of the protocol in [10] that does not use estimated time of arrival and demonstrated that it also outperforms a stop sign in light traffic.

Other researchers have investigated autonomous intersections using real systems involving multiple mobile vehicles. For example, Kolodko and Vlacic used golf-cart-like Imara vehicles in evaluating an autonomous intersection [11]. In their study, all vehicles must come to a complete stop at the intersection irrespective of traffic conditions. This is analogous to the stop sign policy in this paper. Our work differs in being a framework for evaluating many different intersection management policies.

Finally, many other mobile wireless network testbeds exist [12], [13]. Our work differs in it focus on autonomous intersections and not purely on wireless communication.

III. PROBLEM DEFINITION

In this section, we describe the key challenges addressed, our assumptions, and desiderata of our testbed.

A. Challenges

The first challenge is how to design and implement a testbed and software infrastructure for evaluating a wide range of intersection management policies. The framework must be flexible to account for V2I and V2V policies that require different context information at different times.

The second challenge relates to using the testbed to evaluate actual management policies, thereby demonstrating our framework's efficacy. This is challenging because it requires actual system deployment. The goal in doing this is to gain insight into the potential real-world efficiency gains alternative intersection management policies can provide relative to existing traffic signal and stop sign-based policies.

B. Assumptions

We assume *all* vehicles crossing the intersection are autonomous, will actively participate in the employed management policy, and travel straight through the intersection without turning or switching lanes. Handling occasional non-autonomous vehicles, pedestrians, and cyclists, turning, lane switches, and hardening the system against faults and adversaries are essential. However, they are left as future work to reduce the complexity of our initial implementation, and enable us to focus on the aforementioned primary challenges. Solutions for handling many of these challenges exist, though they have only been evaluated in simulation.

In addition, a few more assumptions are made. First, we require the autonomous vehicles to communicate with each other and the surrounding infrastructure over wireless network links. Second, we require the autonomous vehicles to be equipped with sensors that can precisely detect when the vehicle is approaching, entering, and exiting the intersection. For example, overhead markers may be installed that can be detected by the vehicles as they pass underneath (similar to existing toll collection points on our highways). For now, the detection of these points need to be absolutely reliable, though this may be relaxed in the future as more advanced fault-tolerant intersection management policies are developed. Note that depending on the management policy employed, the vehicle may *not* need to know additional details of the intersection like its size, location, and orientation, though the testbed should support the delivery of such information. Finally, we assume that the vehicles have sufficiently powerful brakes or that the lanes have a sufficiently-wide buffer zone on either side (i.e., a shoulder), to enable a vehicle traveling at the speed limit to stop upon detecting the entrance of the intersection, and not end up blocking any of the cross traffic lanes. In our current system, the vehicles have a stopping distance of about 21cm when traveling at 0.5m/s, and a buffer of 69cm is used to ensure safety when vehicles stop at the intersection's entrance.

Finally, we assume that the employed policy remains constant. Investigating how the policy can be dynamically changed on-line based on context like amount of traffic, time of day, and the weather is an area of future work.

C. Desiderata

Real system. The key components of the testbed, which include the vehicles, the intersection, and the infrastructure for managing the intersection, should all be real and not virtual entities in a simulator. All of the sensing necessary for the vehicles to participate in the intersection should be done using real sensors. This includes sensors that detect the key points of the intersection (i.e., the points of approach, entry, and exit), and the sensors that enable the vehicles to follow lanes that traverse the intersection. All of the computation and communication necessary to coordinate safe traversal of the intersection should be done live and not *a prori* in an off-line manner. The goal is to keep the testbed as realistic as possible to capture the highly complex interactions that occur between the cyber and physical elements in a real autonomous intersection.

Extensible and flexible. The testbed should be easily extensible to support evaluating new intersection management algorithms, types and numbers of vehicles, and types of intersections. An important contribution of this testbed is to provide a foundation on which different intersection management policies can be evaluated and compared. The testbed should support a wide variety of vehicles with different physical properties like width, height, weight, turning radius, rate of acceleration, and stopping distance since future autonomous vehicles will likely exhibit such diversity, just as how our human-controlled vehicles do today. The cost of each vehicle should be low enabling large scale experiments involving many vehicles. Finally, the testbed should enable testing different types of intersections, e.g., with different numbers of roads or different numbers of lanes in each road.

Safety. Since we are working with a real system that contains many dynamic components and parts, maintaining absolute safety is critical. Ideally, intersection management policies should prevent collisions, even in the face of unpredictable system behavior like wireless disconnection. However, in case they fail, the resulting collision should not endanger anyone's life. We achieve this in our testbed by scaling down the system and using lighter-weight vehicles that cause little permanent damage when they collide.

IV. APPROACH

We approach the problem of evaluating intersection management policies for autonomous vehicles from both a cyber and physical perspective.

A. Cyber-Approach

From a cyber-perspective, we developed a software framework that provides infrastructure for rapidly implementing and evaluating a highly diverse set of intersection management policies. The key cyber-components of this infrastructure are shown in Figure 1. They consist of an experiment manager running on a central control station, an autonomous intersection client running on each vehicle, and an optional server located at the intersection; the latter is used by V2I intersection management policies.

Since autonomous intersections are naturally distributed systems and at a minimum consist of numerous mobile vehicles, coordinating the start of an experiment is not trivial. Our experiment manager communicates with each of the participating vehicles prior to the start of the experiment. It takes as input an experiment configuration file that specifies the experiment name, type, and vehicles used. If the experiment is evaluating a V2I policy, it also specifies the address of the central server managing the intersection.

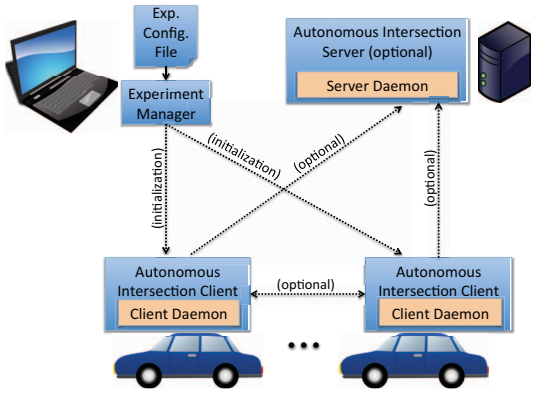

Figure 1. The cyber-elements of our autonomous intersection testbed

Upon receiving the experiment configuration, the experiment manager wirelessly connects to the autonomous intersection clients running on each vehicle, informing them of the experiment parameters. It then coordinates the start of the experiment ensuring all vehicles begin moving at approximately the same time.

Prior to running the experiment manager, an autonomous intersection client is started on each vehicle. This client is a software process that executes the autonomous intersection management protocols to safely navigate across the intersection. To keep this component generic and support the evaluation of a diverse set of intersection management policies, the client simply defines an abstract client daemon and provides the supporting infrastructure needed by specific instances of the daemon that implement the actual intersection management protocols. Specifically, the client provides the daemon a network interface for both single and multi-cast communication, a vehicular kinematics interface for controlling the speed and steering of the vehicle, an event interface for informing the daemon when the vehicle is at critical points around the intersection, and relative state information like from which points the vehicle will enter and exit the intersection. The client selects which client daemon to instantiate and use based on the experiment configuration message from the experiment manager.

To support V2I autonomous intersection management schemes, we provide an autonomous intersection server, which is a software process that runs on a machine at the intersection. As the vehicles approach and cross the intersection, they communicate with this server using a protocol set by a specific intersection management policy. Like the client, the server is designed to be generic. It simply defines an abstract server daemon and provides the infrastructure for supporting instances of the daemon. Each instance of the server daemon implements the server-side of a specific autonomous intersection management policy. The server provides the daemon specifications of the intersection, a network interface, and interfaces for accessing relevant protocol-specific sensors. The selection of which server daemon to use is done when the server is started, which must occur prior to the beginning of the experiment. We for now assume that the server runs the same protocol throughout its lifetime. As mentioned previously, the ability to dynamically change intersection management protocols based on context is left as future work.

B. Physical Approach

The physical design of the autonomous intersection testbed impacts how the vehicles move and detect the intersection and the repeatability of experiments. To simplify aspects of the system not directly related to the evaluation of intersection management protocols, we create a "clean-room" environment for the intersection. There are no obstacles or unexpected debris in the road that need to be detected by the vehicles. In addition, markers are installed at the critical points along the intersection as shown in Figure 2. They include the starting locations of the vehicles and the points of approach, entry, and exit from the intersection. These markers ensure the robots begin at the same location across experiments (i.e., that the initial physical state of the system is consistent across runs) and are easily detected by the vehicle using simple sensors and do not require complex computer vision object-recognition algorithms. In the real world, a reasonable analog may be RFID tags mounted above the lanes that can be read by vehicles as they pass under it.

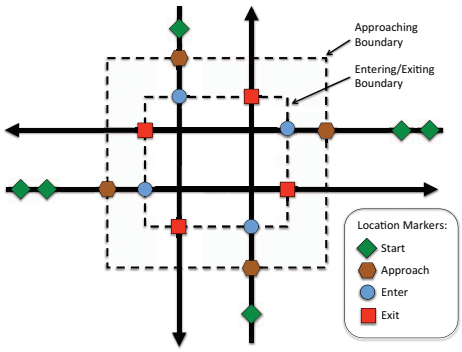

Figure 2. The physical markers placed around our autonomous intersection testbed. A four-way two-lane intersection is shown here as an example. Other intersection configurations are supported, but the relevant markers remain the same.

Prior to conducting an experiment, the vehicles are physically placed in their starting locations, which are shown by the green diamonds in Figure 2. They are oriented to face towards the intersection. Note that a lane may contain multiple vehicles. Lanes are demarcated by lines on the ground that the robot can follow using a simple

vision sensor. When the client receives the start message from the experiment manager, it moves the vehicle forward while following the lane. Upon detecting the marker at the approaching point, the client's daemon initiates the intersection management protocol that it implements. The protocol should ideally grant the vehicle permission to cross the intersection by the time it reaches the entry point. If it fails to do this, the vehicle must stop and wait for permission. Upon gaining permission to cross, the vehicle travels across the intersection and eventually past the exit marker. This illustrates our basic approach at establishing a cyber-physical testbed for evaluating intersection management schemes for autonomous vehicles. The next sub-section provides details on the actual intersection management policies we developed. It is followed by a detailed discussion of the system's implementation.

C. Autonomous Intersection Management Policies

We developed eight different policies for managing autonomous intersections to demonstrate the flexibility of our testbed. They are failsafe against wireless communication failure since the vehicles will not enter the intersection due to the lack of a grant message. Being simple, these policies are not flawless, but they demonstrate some naive and intuitive approaches to autonomous intersection navigation and illustrate the capabilities of our general purpose intersection testbed. In the future, more nuanced policies may be developed, perhaps by extending one of the following protocols, and evaluated using our testbed. The policies are discussed below.

V2I-Sequential. In this policy, the intersection is managed by a central server, which grants vehicles permission to cross the intersection on an opportunistic basis. Initially, the server is idle waiting for vehicles to approach. After reaching the approaching point to the intersection, the vehicle sends the server a `RequestAccess` message asking for permission to cross. This message contains the ID of the transmitting vehicle.[1] The vehicle periodically retransmits this message until it receives a `GrantAccess` message from the server, at which point it can cross the intersection. If the vehicle reaches the entrance to the intersection without receiving a `GrantAccess` message, it stops at the entrance until such a message is received. When the vehicle reaches the exit marker, it sends an `Exiting` message to the server telling it that it has exited the intersection.

The server is relatively simple. It maintains a single value, `grantedVehicle`, which records which vehicle is currently granted permission to cross the intersection. This value is initialized to `null`. Each time a `RequestAccess` message is received, if `grantedVehicle` is `null`, the server changes the value to be the ID of the sender and replies with a `GrantAccess` message. Otherwise, it ignores the request. Ignoring the request is acceptable since the vehicle will periodically retransmit the request until a `GrantAccess` message is received.

Since wireless communication will not be 100% reliable in any real-world system, it is possible for the `GrantAccess` message to be lost. This will result in an inconsistent state where the server mistakenly thinks a vehicle is in the intersection. To account for this, when a `RequestAccess` message is received, the server checks whether the ID contained within this message is equal to `grantedVehicle`. If it is, the message is a duplicate and the server replies with a `GrantAccess` message. Similarly, to account for lost `Exiting` messages, the vehicle periodically transmits this message to the server until the server replies with an acknowledgement. If the vehicle moves out of range prior to the exiting message getting through, a lengthy timeout can be set in the server that indicates the vehicle is mostly likely out of the intersection.

V2I-Parallel. This policy is also managed by a central server. It is the same as V2I-Sequential except it attempts to increase the throughput of the intersection by allowing more than one vehicle to cross the intersection at a time. To do this, the `RequestAccess` message contains not only the ID of the vehicle, but also specifications on where the vehicle will enter and exit the intersection. Using this information, the server can determine the path the vehicle will travel through the intersection and whether it will conflict with any vehicles already in the intersection. Thus, in this scheme, the server maintains a list of vehicles that have been granted permission to cross the intersection. Note that while the core logic of the client on the vehicle is the same between V2I-S and V2I-P, different client daemons must be used since they need to transmit different `RequestAccess` messages.

V2I-Reservation. This is an enhanced version of V2I-Parallel that allows vehicles to obtain reservations for future times when they can enter the intersection. To do this, the `RequestAccess` message is extended to include the amount of time the sending vehicle thinks it will need to cross the intersection. The server uses this information to determine the earliest time the vehicle can cross and responds with this "reservation time." Upon receiving this, the client adjusts the speed of the vehicle to arrive at the entrance just-in-time. Ideally, this would allow the vehicle to avoid coming to a complete stop. This policy is very similar to AIM, except it does not perform fine-grain spatiotemporal allocation of grid locations within the intersection.

V2V-Sequential. This scheme achieves the same semantics as V2I-Sequential except without the use of a central server. Requiring every intersection to have a server may not be feasible in reality due to cost, and the server represents a single point of failure. In addition, this approach highlights our framework's flexibility in evaluating both ad hoc and centralized intersection management schemes. In the V2V-

[1]The ID must be globally unique, e.g., it could be the vehicle's VIN.

Sequential management scheme, the vehicles communicate amongst themselves to negotiate when they can each cross the intersection.

Each vehicle maintains a neighbor list that records the state of the other vehicles that want to cross the intersection and the last time a message was received from each vehicle. This list is initially empty but is populated as the vehicle approaches the intersection based on wireless beacons it receives. The potential states of neighbors include idle, requesting, crossing, and exiting. All vehicles initially start in the idle state.

When a vehicle reaches the approaching point of the intersection, it begins to periodically broadcast a beacon indicating that it is in a requesting state. Simultaneously, it receives beacons from other vehicles approaching the intersection. Since in this management scheme only one vehicle can be in the intersection at a time, if it detects that other vehicles are requesting, it must decide whether to yield to another vehicle. To determine which vehicle can proceed and which must stop, the IDs of the vehicles are compared. The vehicle with the highest ID is given priority to cross the intersection first.[2] If a vehicle decides to yield the intersection to another vehicle, it stops at the entrance to the intersection.

While waiting, the vehicle periodically checks its neighbor list to determine whether it can potentially cross the intersection. It can potentially cross the intersection if there are no other requesting vehicles, no vehicle in the intersection (i.e., in the crossing state), or if it has the highest ID among the requesting vehicles. After determining that it is potentially safe to cross, the vehicle first waits a minimum safe duration that is a multiple of the beaconing rate. This is to gain higher confidence that it is indeed safe to cross. The period may be adjusted to account for the necessary level of safety. The longer the period, the more likely the vehicle can safely cross the intersection; the trade-off is, of course, unnecessarily delaying vehicles. After this period expires, if the vehicle still concludes that it is safe to proceed, it changes its beacons to indicate that it is crossing the intersection and proceeds to cross. When it reaches the exit point, it changes its beacon to indicate that it has finished crossing the intersection. It continues to broadcast this beacon for a pre-set period of time.

V2V-Parallel. This intersection management scheme is similar to V2V-Sequential except it enables multiple nodes to cross the intersection. It does this by including specifications on where the vehicle is entering and exiting the intersection in the beacons. Using this information, each node can determine whether it will interfere with the crossing node. If a waiting or requesting vehicle determines that it can safely cross the intersection simultaneously with a vehicle that is already crossing, it immediately changes it state to crossing and begins to cross the intersection. The assumption is that the vehicles are traveling along parallel lanes, though not necessarily in the same direction. If vehicles can turn, additional coordination steps are necessary to ensure the vehicles that follow the one that is already crossing do not collide.

V2V-Reservation. This extends V2V-Parallel to support reservations. It works by having the vehicles broadcast their self-selected entry time (if determined) and how long they expect to take crossing the intersection. Using this information along with the neighbor list and aforementioned vehicle ID-based ordering, each vehicle computes when it should enter the intersection and arrive just-in-time.

Stop Sign. This management scheme is designed to model the behavior of the traditional stop sign. It is a centrally managed scheme where the vehicles ignore the approaching marker and only send a request upon reaching and stopping at the entrance marker. In this scheme, the same server is used as in the V2I-Parallel scheme.

Traffic Signal. As the name implies, this scheme mimics the behavior of a traffic signal. The server runs in a cycle periodically granting access to vehicles traveling along nonintersecting lanes. When several vehicles approach the intersection almost at the same time, they can be granted access to enter the intersection (i.e., given green signals) if their lanes are nonintersecting and no vehicles are occupying their lanes the intersection; otherwise, only a subset of these vehicles on nonintersecting lanes can enter the intersection and the rest will have to wait until next cycle. Note that in this scheme, the clients run the same daemon as in V2I-Parallel, but the server runs a special daemon that implements the semantics of a traffic signal.

Collectively, the stop sign and traffic signal policies provide a baseline performance against which the other more flexible schemes can be compared.

V. IMPLEMENTATION

We implemented the autonomous intersection testbed in Pharos [14], a general mobile computing platform consisting of approximately thirty highly modular Proteus mobile robots. These robots serve as the autonomous vehicles in our system. For this work, we used the hardware configuration shown in Fig. 3. It consists of a modified Traxxas Stampede mobile chassis and a module containing computational elements, a CMUCam2 vision sensor, and a Sharp GP2Y0A02YK0F Short Range IR range finder. The computational elements include a general-purpose x86 computer and a Freescale 9S12 microcontroller (MCU). The x86 is a VIA EPIA Nano-ITX motherboard that contains a 32-bit 1GHz VIA C7 CPU, 1GB of DDR2 RAM, a 16GB compact flash drive, and a CM9-GP IEEE 802.11g WiFi mini-PCI module based on the Atheros AR5213A chipset.

[2]To prevent starvation (i.e, a vehicle waiting at the entrance forever), more advanced vehicle selection policies can be employed that, for example, consider how long a vehicle has been waiting at the intersection when deciding which can go first.

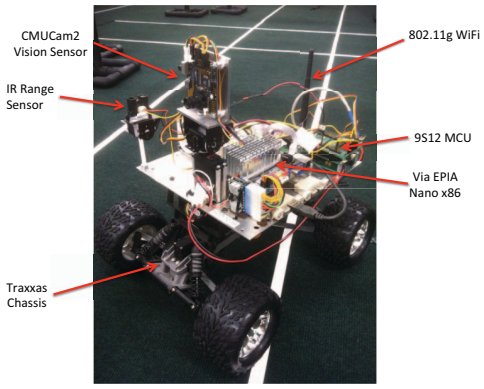

Figure 3. The Proteus Node.

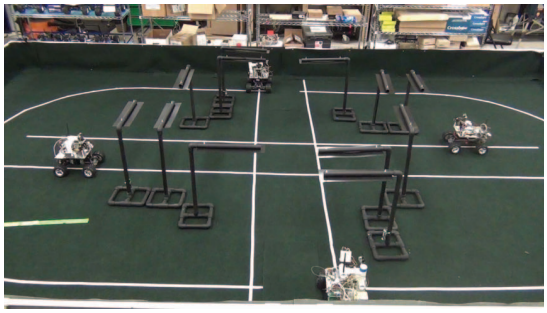

Figure 4. The Autonomous Intersection Testbed configured with a 4-way intersection of two 2-lane roads. The vehicles are positioned at their starting points.

Each Proteus runs Ubuntu Linux 11.04 server and Player 3.02 [15]. Custom Player drivers provide programming abstractions for vehicle movement and sensing; it is through these interfaces that we provide the resources needed by the client daemons on the vehicles. The WiFi interfaces form an ad hoc network among the robots and with the central server should one be used.

While scaled down, the Proteus robots are real systems that move with the same nonholonomic motion as most real vehicles (i.e., the front wheels steer while the rear wheels remain straight). In addition, their size (they weigh 6kg and have dimensions of 40x32x35cm) and low cost (less than $2,500 USD each) prevent catastrophes in case intersection management schemes fail and collisions occur. Given the robots' modularity, where the mobile chassis and computational plane are decoupled, the Traxxas mobile chassis can be easily swapped with something larger like a golf cart. By exploiting this modularity, our testbed enables users to first evaluate autonomous intersection management policies using small scale vehicles and to move onto larger vehicles after gaining confidence in the safety and correctness of a particular policy. For this study, we limit our experiments to the Traxxas mobile chassis due to limits in our lab's physical dimensions and the fact that we are testing our implementations of various intersection management policies for the first time.

Using Pharos, we implemented one of the first real-world managed autonomous intersections that involves multiple vehicles. Figure 4 shows the testbed configured to evaluate the performance of intersection management policies in a four-way intersection of two 2-lane roads. White vinyl tape denotes lanes. The ground is dark-green outdoor carpet, which provides a smooth moderate friction surface for the vehicles to accelerate without excessive tire slippage. The high contrast between the white tape and dark-green carpet is necessary for the vehicle's CMUCam2 to reliably detect it. The CMUCam2 is mounted on two medium torque HiTec HS-322HD Deluxe Servos, which pan and tilt the camera to ensure a vehicle can follow the lane around curves. As a vehicle follows the line, it passes under overhead markers that denote the critical points of approach, entrance, and exit of the intersection. These markers are constructed of 1 inch PVC pipe and ABS plastic. They rise 66 cm above the vehicles and span 51 cm across the lane. A 14 cm wide sheet of ABS plastic facing down ensures the vehicle reliably detects the marker. The markers are painted black to prevent confusing the vision sensor, which is searching for a bright white line that denotes the lane. The vehicle uses its short range IR range finder to detect these overhead markers as it follows the lane. Figure 3 shows how this IR sensor is mounted facing up towards the front of the vehicle to ensure rapid detection when the vehicle passes under the marker. The vehicle's 9S12 MCU samples this sensor at 25Hz, which is sufficient for the vehicles to reliably detect the markers. To reduce the chance of false positives, the client on the vehicle monitors the Traxxas' wheel encoder and ensure that consecutive markers are at least 15cm apart (they are separated by at least 36cm in our test configuration). While this critical point detection system suffices for our testbed, the development of alternative more reliable mechanisms is future work.

In addition to detecting the critical points around the intersection, it is also important to detect which lane the vehicle is in. Our system supports several options to achieve this. The first is to manually specify this in the experiment configuration file passed to the experiment manager. This is acceptable since we need to manually place the robots at their starting locations anyway, meaning we know *a priori* how the vehicle will travel through the intersection. The second option is to use an active range sensor like Cricket motes installed at the base of markers at the approaching points of the intersection and on the vehicle. As a vehicle drives past an approaching marker, the cricket mote on the vehicle determines the distance to the cricket mote on the marker and its ID. Together, this information can identify the lane the vehicle is in. Specifically, if the distance is

less than a threshold, the vehicle assumes it is in the lane corresponding to the ID of the cricket on the marker. Other technologies like RFID may also be used. In the experiments presented in this paper, we use the first approach since it is the simplest.

The software framework that implements the cyber-portion of our system is primarily written in Java. We use Java sockets and multicast sockets for network communication. A Java client interfaces with the Player robotics framework, enabling our framework to control the vehicle's speed and steering and to receive short range IR readings and wheel encoder information for detecting markers. Object-oriented programming provides a highly extensible and efficient implementation. All client daemons extend a master abstract `ClientDaemon` that implements the core services including lane following, critical point detection, and entry/exit point detection mechanisms. Likewise, all server daemons extend an abstract `ServerDaemon` that provides the basic network and sensing interfaces. The same client daemon is used in the V2I-Parallel, V2I-Sequential, and Traffic Signal management schemes. This is because the client does not need to know whether the server is allowing simultaneous traversals of the intersection, or even implementing traffic-signal semantics. In addition, the Stop Sign client daemon *extends* the V2I-Parallel client daemon by simply ignoring the approaching marker (all other behavior in the Stop Sign client matches that of the V2I-Parallel client daemon). In all, by exploiting the object-oriented nature of the Java programming language, we provide an extensible framework for evaluating autonomous intersections with significant code reuse.

VI. EVALUATION

We used our testbed to evaluate our eight different intersection management policies. Four vehicles are configured to cross a four-way intersection between a two 2-lane roads. The starting locations of the vehicles and the dimensions of the intersection are shown in Figure 5. The dimensions are only shown for one lane since the other four lanes are the same. During the experiment, the speed limit is set to 0.5m/s. Note that 0.5m/s is a moderate speed since the distance between the entrance and exit is 199cm, meaning a vehicle may cross in only 4 seconds. For the Traffic Signal policy, a 2-phase traffic signal was emulated with a 30s cycle time, meaning up to $\lfloor \frac{30}{4} \rfloor = 7$ vehicles may cross per enabled lane per cycle. After each experiment, we manually reset the system by physically moving the vehicles back to their start states and terminating and restarting all vehicle and intersection software processes.

Each intersection management scheme was executed until it ran flawlessly ten times (i.e., all four vehicles successfully crossed the intersection without any collision). The reasons for failure vary, but are usually due to factors unrelated to the

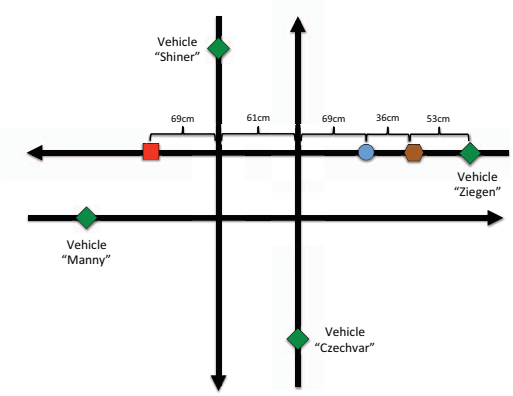

Figure 5. The starting configuration of all experiments performed, and the dimensions of the intersection used in the evaluations.

intersection management policy.[3] For example, most failures include loss of the start experiment message sent by the experiment manager to a vehicle, loose wires connecting the IR or wheel encoder sensors to the MCU, out-of-focus CMUCam2 vision sensor causing the vehicle to loose the lane, and the vehicle's batteries running out of power. Logical errors in the implementation of an intersection management policy would sometimes cause failure. However, such errors could usually be identified and replicated using our testbed, enabling quick resolution, demonstrating how our testbed can assist in debugging intersection management policies.

For the V2I tests, the server was running on the same machine as the experiment manager, an Apple Macbook Pro laptop residing next to the testbed. The experiment manager only runs prior to the start of the experiment and thus does not consume any computational resources during the actual experiment. In addition, both entities reside entirely within the cyber-domain and thus are relatively agnostic to the actual machine on which they run. The V2I client daemon was configured to have a request timeout of two seconds, meaning it would retransmit requests to the server at a rate of 0.5Hz. Recall that in the current V2I schemes, the server will simply ignore a request if it cannot be granted.

For the V2V tests, the broadcast period was randomly selected between 100ms and 1s. This helps avoid repeated collisions between synchronized beacon transmitters. The maximum number of consecutive beacons that can be lost before concluding that a node is disconnected is five. Thus, if more than five seconds pass and no beacon is received from a particular vehicle, the vehicle is removed from the local vehicle's neighbor list. The minimum safe duration was set to 2.1s, meaning a node must wait 2.1s after detecting a potential opportunity to cross the intersection

[3]For full details of all experiments including raw data and videos, see: http://pharos.ece.utexas.edu/wiki/index.php/AutoInt#Experiments.

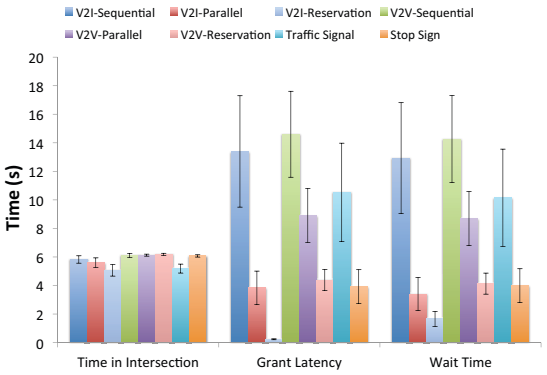

Figure 6. The performance of the intersection management policies.

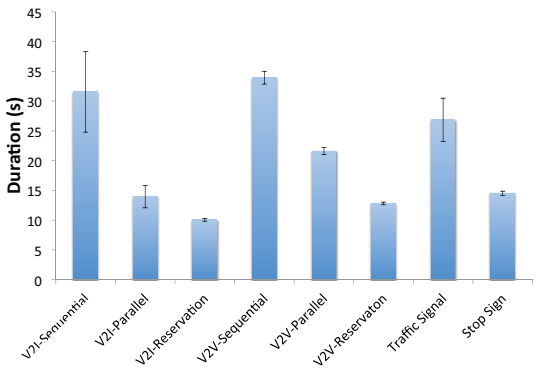

Figure 7. Total duration of four vehicles crossing the intersection.

before concluding that it is safe. This period was selected to enable all other competing vehicles to announce their state at least twice during this period.

We used extensive logging to enable off-line analysis of performance. In addition, all experiments were recorded by an overhead camcorder enabling us to verify the experiment physically worked (i.e., that all vehicles stopped where they should and that there were no potential collisions).

The results are shown in Figure 6. They present the average times over 10 successful executions of each type of intersection management policy. The error bars denote 95% confidence intervals. We used three values to evaluate the management policies: time in intersection, wait time, and grant latency. Time in intersection is the time a node spends between leaving the entrance and arriving at the exit. In our scenario, the different intersection management policies do not significantly impact this duration, which is about 6s. Since the width of the intersection is 199cm, the average speed was $\frac{199cm}{6s} = 0.33m/s$, which is lower than the 0.5m/s speed limit. The slower speed and longer time in the intersection is due to some vehicles stopping at the entrance and having to accelerate through the intersection.

The wait time is the duration a vehicle stops at the entrance of the intersection prior to crossing. Among our metrics, wait time is the most critical since it is a direct cost in terms of increasing the time to cross the intersection; it is the source of the frustration in the example scenario in Section I. The wait time was much smaller in the V2I-Parallel and V2I-Reservation management policies because two vehicles will have a wait time of near zero, meaning they can immediately cross the intersection upon arrival without slowing, and the other two vehicles will only wait the amount of time it takes one vehicle to cross the intersection since the other two are crossing in parallel. The wait times are not zero in the reservation-based policies because the vehicles could not decelerate fast enough between the approaching and entering points to the intersection. Clearly, separating these points more may further improve reservation-based policy performance. Wait times are higher in the V2V policies than in the V2I policies due to the overhead of distributed decision making, specifically the 2.1s minimum safe duration a vehicle must wait before it an be sure its decision is correct.

In V2I experiments, the grant latency is the time between when a vehicle first asks for permission to cross the intersection and when it is granted permission. For V2V experiments, it is the time between when a vehicle first wants to gain access to the intersection and when it decides it has access. The grant latency is clearly more variable across schemes; it is significantly lower in the parallel schemes, and the V2I-Reservation scheme is by far the best at only 0.23 ± 0.03s since the server always grants a vehicle access, though potentially at a future point in time.

Of particular interest is a comparison between the various new management schemes relative to traditional traffic signals and stop signs. In these experiments, traffic signals have on average higher grant latencies and wait times than stop signs since traffic signals essentially allow "batches" of vehicles traveling along the same lane through, meaning our use of four vehicles dispersed across all four lanes does not play to this policy's strength. Instead, such a traffic pattern better fits a stop sign. More interestingly, V2V-Reservation had a wait time nearly identical to that of Stop Sign (4.1 ± 0.7s versus 4.0 ± 1.2s). This implies that the amount of time needed to perform V2V coordination equaled that of having every node stop and query a central stop sign server for directions. Finally, the results indicate that V2I-Reservation is more than twice as efficient as a traditional stop sign in terms of the wait time (1.7 ± 0.5s versus 4.0 ± 1.2s).

Finally, we estimate the aggregate traversal times of all vehicles of the intersection by measuring the total duration of intersection traversal, which is the time between the first vehicle entering and the last vehicle exiting. The results are shown in Figure 7. They indicate that among all of the management policies, V2I-Reservation had the lowest (best) duration of 10.0 ± 0.2s. This is lower than the stop sign, which has a duration of 14.5 ± 0.4s. The V2V-Reservation

policy had a duration of 12.8 ± 0.2s, which also beats the stop sign. When comparing the traditional traffic signal, which has a duration of 26.9 ± 3.6s, to the newer V2I-Reservation management policy, the aggregate traversal time for four vehicles to cross the intersection is improved by approximately $\frac{26.9-10.0}{10.0} \cdot 100 = 169\%$ when using V2I-Reservation.

VII. CONCLUSIONS AND FUTURE WORK

A rethink of the conventional infrastructure for our roadways is an essential counterpart to increasing autonomy being incorporated into our vehicles. By instrumenting our roads with cyber-physical infrastructure, the entire traveling experience can be made safer, faster, more energy efficient and more enjoyable. This paper represents a significant step in this direction by focusing on instrumenting intersections, which when combined with vehicular autonomy and coordination, can enable faster intersection traversal times. In particular, we construct a cyber-physical testbed for evaluating new intersection management policies. Our investigation resulted in the design and implementation of one of the first autonomous intersection testbeds with multiple real vehicles. Unlike previous testbeds that are based partly or entirely on simulation, our testbed can test the physical properties and intersections of multiple physical vehicles entirely in reality. An evaluation of numerous intersection protocols illustrates the efficacy of our testbed and indicates that a V2I-Reservation intersection management policy is superior to both the traditional traffic signal and stop sign mechanisms that we use today.

In the future, we intend to run larger experiments and evaluate new intersection management policies using our testbed. Past simulations indicate that more sophisticated policies that perform advanced reservations and fine-grain control of space-time within the intersection will result in even greater efficiency [3], [8]. Additional investigation is also needed to study the impact of 1) vehicles with varying speeds and dynamics, 2) traffic patterns with different lane or intersection configurations, 3) pedestrians and non-autonomous vehicles, 4) sensory inputs other than location. To this end, we will extend our testbed to support turning, switching lanes, swerving to avoid disabled vehicles, prioritizing emergency vehicles, avoiding locations that may contain non-autonomous entities, and experimenting with additional sensors like laser range finders and cameras. Ultimately we would like to scale up the testbed to use life-size vehicles and mixed-reality simulations [4], and to support multiple sequential autonomous intersections.

REFERENCES

[1] E. Guizzo, "How google's self-driving car works," http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works, October 2011.

[2] C. Squatriglia, "Audi's robotic car drives better than you do," http://www.wired.com/autopia/2010/03/audi-autonomous-tts-pikes-peak, March 2010.

[3] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, March 2008.

[4] M. Quinlan, T.-C. Au, J. Zhu, N. Stiurca, and P. Stone, "Bringing simulation to life: A mixed reality autonomous intersection," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[5] "DARPA grand challenge," http://en.wikipedia.org/wiki/DARPA_Grand_Challenge.

[6] R. Calo, "Nevada bill would pave the road to autonomous cars," http://cyberlaw.stanford.edu/node/6663, April 2011.

[7] S. Shladover, C. Desoer, J. Hedrick, M. Tomizuka, J. Walrand, W.-B. Zhang, D. McMahon, H. Peng, S. Sheikholeslam, and N. McKeown, "Automated vehicle control developments in the path program," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 1, pp. 114–130, 1991.

[8] D. Fajardo, T.-C. Au, T. Waller, P. Stone, and D. Yang, "Automated intersection control: Performance of a future innovation versus current traffic signal control," *Transportation Research Record (TRR)*, 2011.

[9] R. Naumann and R. Rasche, "Intersection collision avoidance by means of decentralized security and communication management of autonomous vehicles." in *Proceedings of the 30th ISATA - ATT/IST Conference*, 1997.

[10] M. VanMiddlesworth, K. Dresner, and P. Stone, "Replacing the stop sign: Unmanaged intersection control for autonomous vehicles," in *AAMAS Workshop on Agents in Traffic and Transportation*, Estoril, Portugal, May 2008, pp. 94–101.

[11] J. Kolodko and L. Vlacic, "Cooperative autonomous driving at the intelligent control systems laboratory," *Intelligent Systems, IEEE*, vol. 18, no. 4, pp. 8 – 11, jul-aug 2003.

[12] P. De, A. Raniwala, R. Krishnan, K. Tatavarthi, J. Modi, N. A. Syed, S. Sharma, and T.-c. Chiueh, "Mint-m: an autonomous mobile wireless experimentation platform," in *Proceedings of the 4th international conference on Mobile systems, applications and services*, ser. MobiSys '06. New York, NY, USA: ACM, 2006, pp. 124–137. [Online]. Available: http://doi.acm.org/10.1145/1134680.1134694

[13] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, "Mobile emulab: A robotic wireless and sensor network testbed," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, april 2006, pp. 1 –12.

[14] C. Fok, A. Petz, D. Stovall, N. Paine, C. Julien, and S. Vishwanath, "Pharos: A testbed for mobile cyber-physical systems," Univ. of Texas at Austin, Tech. Rep. TR-ARiSE-2011-001, 2011.

[15] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *ICAR*, 2003.