12-05-04-0026

Real-Sim Interface: Enabling Multi-resolution Simulation and X-in-the-loop Development for Connected and Automated Vehicles

Yunli Shao[1], Dean Deter[1], Adian Cook[1], Chieh "Ross" Wang[1], Bradley Thompson[2], Nolan Perry[1]
[1]Oak Ridge National Laboratory, USA
[2]MathWorks, USA

Abstract

Connected and automated vehicles (CAVs) can bring safety, mobility, and energy benefits to transportation systems. Ideally, CAV applications would be fully evaluated and validated prior to real-world implementation. However, there are many technical challenges in both software and hardware that hinder the process. To comprehensively evaluate all aspects of CAV applications, an integrated evaluation environment is needed with various simulation tools from different domains. In the current literature, there lacks a well-developed interface to enable multi-resolution simulation of vehicle, traffic, virtual environment, and hardware-in-the-loop (HIL) simulation. In this work, a modular and flexible interface is developed to enable multi-resolution vehicle and traffic co-simulation for CAV applications. This interface is built upon Oak Ridge National Laboratory's Real-Sim approach, can support various simulation tools and real-time X-in-the-loop (XIL) simulations, and is based on network communication protocols. The network communication delays are analyzed for simultaneous connection of up to 20 simulators. Then, two example applications are studied to demonstrate the potential usage of the Real-Sim interface: (1) a centralized merging scenario where the dynamics of two ego vehicles are emulated with detailed vehicle and powertrain dynamics models; and (2) a signal-controller-in-the-loop (SCIL) evaluation where one intersection of the simulated traffic scenario is controlled by a physical signal controller. This Real-Sim interface is a tool that allows researchers to bring real, tangible hardware and software into simulated environments to comprehensively evaluate and support a wide variety of CAV applications.

Keywords: connected and automated vehicle, hardware-in-the-loop, evaluation, simulation, co-simulation

Introduction

The connected and automated vehicle (CAV) is an emerging technology that has the potential to improve the safety, energy efficiency, and mobility of transportation systems [1]. Vehicles equipped with vehicle-to-X (V2X) communication and perception sensors can obtain new traffic information regarding surrounding vehicles' position and speed, signal phase and timing (SPaT), driving situation (e.g., work zones, accidents, congestion), and other information in real-time. This information can greatly improve CAVs' awareness of both current and future traffic conditions. Leveraging vehicle automation, various CAV applications can be developed to control a vehicle in an intelligent way to benefit transportation systems. For example, energy-efficient vehicle speed control strategies can be developed for CAVs by focusing on minimizing energy consumption. These strategies require future traffic prediction and an optimization algorithm [2]. Traffic prediction is enabled by the new information from connectivity and sensors, and an optimal control problem is then formulated and solved for a look-ahead horizon to minimize vehicle energy consumption [3, 4]. To ensure safety, the vehicle is subjected to traffic constraints such as safe following distance, traffic lights, speed limits, and other traffic rules and regulations. Coupled with vehicle speed optimization, powertrain operation can be further optimized to increase the benefits [2, 5]. This optimization includes methods such as optimizing power-split between the engine and battery for hybrid electric vehicles [6, 7, 8] or optimizing shift schedules for conventional internal combustion engine-based vehicles [9, 10]. The literature has shown that 10%-20% energy and mobility benefits can be achieved by co-optimization of vehicle speed and powertrain operation [2, 11]. When coordinating the motion of multiple vehicles in proximity of each other, cooperative driving automation (CDA) applications can be developed, which further enable benefits at the traffic level. Described in details in SAE J3216 [12] and separated into Class A, B, C, and D, sample applications include coordinated merging [13, 14], cooperative intersection [15], and cooperative situational awareness [16].

Before actual implementation in the real world, CAV applications should be fully evaluated and validated. However, many inherent technical challenges exist [17, 18, 19] because CAV applications are complex and involve many technologies, such as sensing, localization, perception, communication, motion planning, and control. Existing simulation tools lack the ability to comprehensively cover all these aspects in a streamlined workflow. The fidelity of pure simulation-based approaches also depends on the accuracy of all simulated components, such as human driver behaviors, traffic dynamics, vehicle and powertrain dynamics, traffic signal lights, V2X communication, perception sensors, 3D road geometries, and other components that are difficult to model. Simulating all these systems together with high fidelity is challenging. Additionally, the simulated performance indexes such as mobility, energy efficiency, safety, and emissions may not truly reflect the real-world performance because of their complexities, unmodeled dynamics, and variations. Real-world vehicle testing provides accurate results and addresses variations in performance measurement but can be expensive, be time-consuming, and have safety concerns. Vehicle testing requires acquisition and maintenance of test vehicles with major modifications to equip all necessary sensors, actuators, and communication devices [20]. Also, CAV applications usually require a minimum penetration rate of connectivity to provide benefits to transportation systems [21, 22, 3]. Creating this penetration rate requires many test CAVs and evaluating them at scale, which further increases the level of challenges in real-world testing. An alternative approach is hardware-in-the-loop (HIL) simulation, which has attracted a great deal of attention from researchers [17, 23, 24]. HIL simulation incorporates actual physical systems, such as powertrain components or test vehicles, with simulation, such as a simulated traffic environment.

Also, HIL simulation has the advantage of keeping the actual dynamics of complex physical systems in the loop, which is otherwise challenging to model and simulate, while at the same time avoiding the complexity and risk of safety-related vehicle testing in the real world.

An integrated evaluation environment is needed to comprehensively evaluate CAV applications. Often, these evaluation environments require an integration of different existing simulation tools from various domains. Two critical aspects are the ego CAV simulation and microscopic traffic simulation. A vehicle simulator is typically used to develop and evaluate vehicle control software, such as an advanced driver-assistance systems (ADAS) or automated driving systems (ADS). Detailed dynamics of an ego vehicle are modeled (e.g., longitudinal, lateral, powertrain, and tire dynamics) with its subcomponents (e.g., steering systems and suspensions). A virtual 3D environment is typically included to visualize and test the ego vehicle. With the emergence of automated vehicles (AVs), many state-of-the-art virtual environments have incorporated high-fidelity photorealism into their simulations, as well as full perception sensor modeling and characterization. Additionally, HIL and software-in-the-loop (SIL) simulation are often offered to evaluate real hardware components (e.g., controller-in-the-loop, camera-in-the-loop, and sensor-in-the-loop) in the virtual environment. Vehicle simulators have been used by the automotive industry for many years and are designed to focus on the detailed simulation of a standalone ego vehicle at a high level of details. CAVs are different from conventional vehicles in that they proactively interact with surrounding vehicles and influence the traffic system in a feedback manner. A significant number of vehicles need to be considered to understand the impacts of CAVs to the traffic system. This characteristic is typically the focus of microscopic traffic simulation tools; the motion of each vehicle is described using behavioral models (e.g. car-following models and lane-changing models) without considering detailed vehicle dynamics. Therefore, traffic simulators can simulate the vehicle interactions and traffic patterns for a scenario with few road segments, corridors, or even a city with hundreds or thousands of vehicles. Thus, the overall traffic level performance can be understood using traffic level quantities, such as delay, travel time, and traffic flow. Considering the different focus and levels of detail for vehicle and traffic simulators, the integration of these tools to create a co-simulation environment for the development and evaluation of CAV applications is highly beneficial. Therefore, for ego CAVs, detailed dynamics and performance can be simulated while their concurrent impacts to the surrounding traffic can be fully understood.

Motivation and Contribution

Current vehicle simulators include, but are not limited to, open-source tools such as CARLA [25], LGSVL Simulator [26], and commercial tools such as IPG CarMaker [27], Vires VTD [28], and dSPACE ASM [29], etc. Traffic simulation tools include the open-source SUMO [30] and commercial software PTV VISSIM [31], Aimsun [32], among others. Generally, these tools do not have co-simulation interfaces and may have limited functionality if the necessary hooks exist. Ideally, one would choose a specific vehicle simulator as well as a traffic simulator and develop a corresponding co-simulation environment once for this structure. However, each tool has its benefits and drawbacks which may or may not be desired for the target applications. Typically, during the development of a CAV application, industry, government, or academia will leverage different tools to fully evaluate the CAV technology. Many considerations favor one tool over the other, for example: 1) Open-source versus commercial. Open-source tools are free, flexible, and usually update new functionalities frequently, but they can lack support/documentation or not be fully tested. Commercial tools are generally well developed but require paid licenses, which could be expensive; 2) Level of modeling fidelity versus computational time. Some tools have higher fidelity models (e.g., vehicle models, sensor models, traffic models, or the visualized 3D environments), but could have higher computational burdens; 3) Flexibility, modularity, and user-friendly application programming interfaces (APIs). Often, users need to implement customized models, controls, and applications. A flexible simulation tool with easy-to-implement APIs is always desirable. 4) Compatibility with standards, e.g., open-source map-generation packages such as OpenDRIVE [33]; 5) High performance computing (HPC) potential and scalability. Managing and evaluating CAV applications will be challenging if a separate integration interface is needed every time the simulation tool is changed. The literature lacks a well-developed interface for co-simulation of vehicle, traffic, and virtual environment for CAV applications. Eclipse MOSAIC [34] is a co-simulation framework designed for such purposes using high level architecture (HLA) concepts. However, currently, MOSAIC is only integrated with limited traffic and vehicle simulators and is not designed for real-time HIL implementation.

The major contribution of this work is the development of a flexible Real-Sim interface that enables multi-resolution vehicle and traffic simulation and X-in-the-loop (XIL) development for CAV applications. The interface is designed to be modular to support integration of various simulation tools and real-time XIL systems and can be a critical component to facilitate the development and evaluation of perception algorithms, motion planning/control strategies, and CDAs. Figure 1 shows the potential usages of the Real-Sim interface with example simulators and XIL systems that can be integrated. Existing simulation platforms usually are designed for a specific vehicle or traffic simulator and are not aimed for XIL integration with complete simulation capabilities. Due to different protocols, APIs and program languages used by different simulators, it is challenging for researchers to integrate various vehicle and traffic simulators and XIL systems in a flexible way. The Real-Sim interface is developed to address these challenges. The advantage is that the differences of simulators become transparent to the users and the integration becomes a streamlined process. Then users can select the relevant simulators and tools to “plug and play” with the help of the Real-Sim interface. Oak Ridge National Laboratory’s (ORNL’s) Real-Sim is simply defined as nearly any part of a system can be “in the loop,” either physically or virtually. This concept has become more evident as much of transportation research has expanded beyond the vehicle into the traffic networks and traffic control devices. Because of the nature of this expansion, ORNL has adapted Real-Sim into all of its XIL capable laboratories, as well as much of its simulation and model-based design. Real-Sim allows researchers to bring real, tangible hardware and software into simulated environments. This approach applies to a wide variety of applications, including Real controller – Simulated vehicle, Real vehicle – Simulated vehicle environment, Real SPaTs – Simulated traffic, Real time traffic – Simulated traffic flow results, etc. The Real-Sim interface presented in this work can be adapted to all these applications. The interface is modular and designed to be flexible to connect to other simulation components (e.g., traffic simulator, vehicle simulator, HIL system, signal controller) through network communication protocols. The Real-Sim approach and interface enables simulation feedback using relevant XIL platforms as well as actual on-road or on-track testing. Then, researchers can determine how well a simulation emulates or replicates the real world to enable realistic operations and results for facilitating CAV technology developments [35].

The remainder of this paper is organized as the follows: First, the overall framework of the Real-Sim interface and its design principles are discussed. Next, the performance of the Real-Sim interface is analyzed in terms of the interface computational efficiency and delays. Then, two example applications of the Real-Sim interface are demonstrated. Finally, the conclusions are presented.

Framework of the Real-Sim Interface

Overall Architecture

The Real-Sim interface is the bridge enabling co-simulation of vehicle, traffic, virtual environment, and XIL components. Real-Sim is implemented in a Server-Client fashion with coupling between each component through the network communication protocol TCP/IP. Thus, the interface is fully modular and flexible and can be used to integrate various vehicle and traffic simulation tools as well as XIL components.

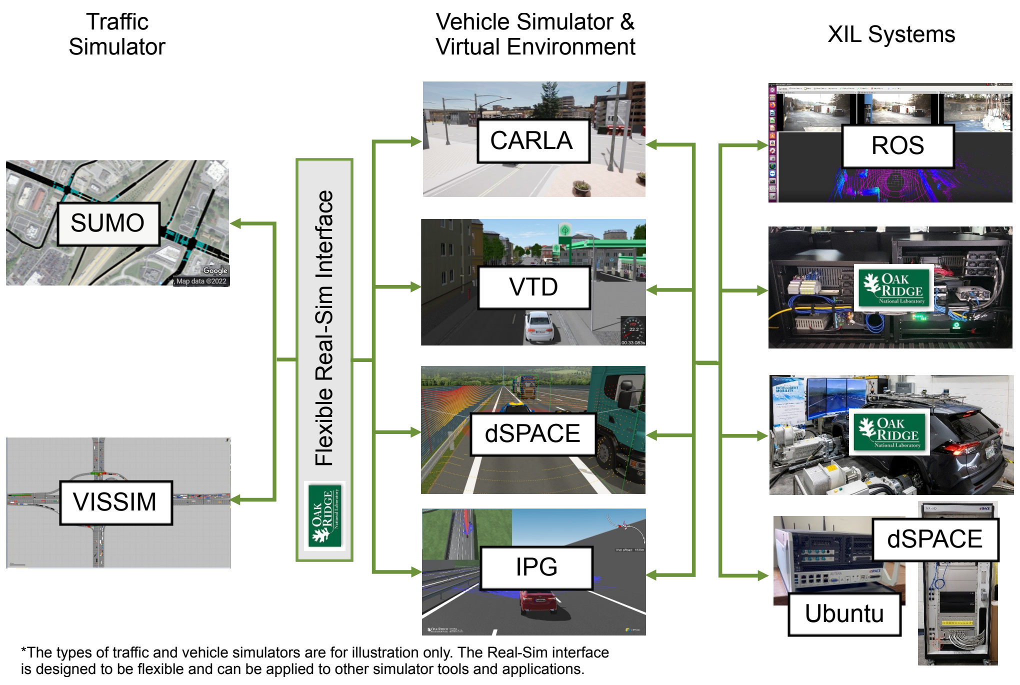

Traffic Simulator
Vehicle Simulator & Virtual Environment
XIL Systems
SUMO
VISSIM
Flexible Real-Sim Interface
CARLA
VTD
dSPACE
IPG
ROS
dSPACE
Ubuntu
*The types of traffic and vehicle simulators are for illustration only. The Real-Sim interface is designed to be flexible and can be applied to other simulator tools and applications.

Figure 1: Potential Usages of the Real-Sim Interface: modular and flexible integration of various simulation tools and XIL systems

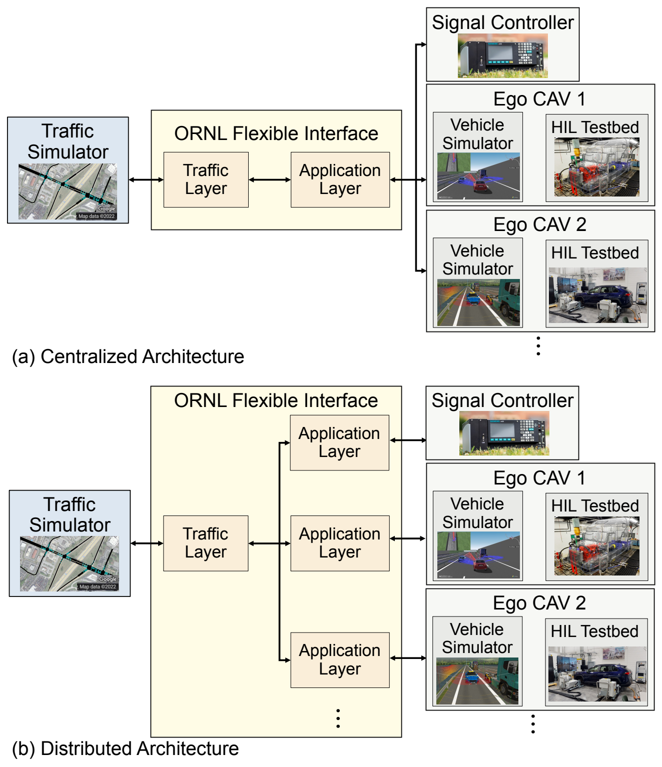


Figure 2: Architecture of the Flexible Real-Sim Interface

Figure 2 shows the overall architecture of the interface. The two major components are: 1) the Traffic Layer, which is the core that directly connects to the traffic simulator to extract traffic information; 2) the Application Layer, which contains the programs that enable CAV applications, such as coordinated driving control, optimal signal control, etc. The Traffic Layer acts as a *Server* that distributes the traffic information to all connected *Clients*, such as the Application Layer, and does not need to have any information about how Clients will use the information. Each Client only needs to subscribe to the information of interest (details are given in the "Message Flow" section). A Client is essentially the Application Layer that contains the CAV algorithms. This layer receives the traffic information and further processes it to generate control commands for the CAV Application Clients, which can be ego vehicle simulators, HIL testbeds, physical signal controllers, and so on.

Depending on the specific CAV application, the Application Layer could be applied in either the centralized architecture or distributed architecture. In the centralized architecture, a single Application Layer with a centralized controller that determines the control commands for all ego vehicles and passes the commands to the CAV Application Clients is used. This centralized Application Layer acts as the Server to all CAV Application Clients. Each Client subscribes to the Server to receive corresponding control commands, and the Server is not required to know the details of the Clients. This modular design facilitates integration with different vehicle simulators and XIL testbeds for various applications.

In the distributed architecture, each CAV Application Client is tied to one Application Layer. Each Application Layer only holds the control strategy for the corresponding CAV Application Client. The Traffic Layer is then connected to multiple Application Layers, where each subscribes to only information in proximity. The same Application Layer program can be applied to both centralized and distributed architectures. The only difference is the number of linked CAV Application Clients and the subscribed information. The "Example Applications" section demonstrates the actual implementation of this Real-Sim interface in two applications.

Message Flow

The Real-Sim interface is designed to integrate various simulators with different CAV applications, as the required traffic information for each application will be different. To maintain modularity and flexibility, the interface can subscribe to various information types. Figure 3 shows a typical message frame. It contains a *Header*, which indicates the current simulation status, simulation time, and the total length of the message frame; and the actual *Data Frame*. Currently, three types of *Data Frame* are implemented: vehicle data frame, traffic light data frame, and traffic detector data frame. The vehicle data frame contains a vehicle's state information, such as position, speed, acceleration, road link and lane in the traffic simulator, etc. The traffic data frame includes the

SPaT information of each intersection in the simulation. The detector data frame has the detector events for each detector (e.g. loop detectors or camera detectors) of each intersection. For a message frame, there is consistently one *Header* that may have multiple *Data Frames*. Each *Data Frame* can be any of the three defined types with a *Data ID* byte to distinguish each type and *Data Length* bytes. For example, every vehicle state will be parsed as one *Data Frame*. For each CAV application, the required message types need to be defined at the outset. Then, the Traffic Layer will be able to extract the traffic information from the traffic simulator, and proceed to the Application Layer. Example message subscriptions include: all vehicle states within an X meter radius of an ego vehicle; all detector calls at an intersection; all vehicle states at a given road link, etc.

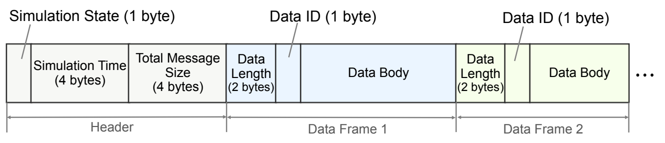


Figure 3: Typical Message Frame

Synchronization

Generally, the traffic simulator and CAV Application Clients (e.g., a vehicle simulator) have different simulation steps. Therefore, synchronization is critical for the success of co-simulation. Typically, CAV Application Clients are actors that researchers controlled such as ego vehicles and traffic signal controllers, while the traffic simulator simulates all the other actors in the traffic scenario. These two programs together simulate the complete performance of all traffic actors in a distributed simulation fashion. A traffic simulator typically has a longer time step than CAV Application Clients, which are typically implemented in real-time HIL simulation with a shorter time step. In a real-time HIL setup, CAV Application Clients often have a simulation time aligned with the real-world time clock. Therefore, the Real-Sim interface is designed to use the simulation time of CAV Application Clients to trigger the traffic simulation. Figure 4 shows the overall synchronization mechanism. As can be seen from the figure, the CAV Application Client has faster time step which is represented by those vertically aligned circles (this is for illustration only, the actual number of circles/time steps varies). Suppose the simulation time step of the traffic simulator is Δt; the current simulation time is t, which is a multiple of Δt (if the CAV Application Client runs on a real-time HIL testbed, then t is also the real-world clock time). Then, the synchronization occurs at every Δt time step. The CAV Application Client will send the actual state of the simulated traffic actor to the Real-Sim interface, which in turn passes it to the Traffic Simulator. This is represented by the top grey arrow pointing right to left. Upon receiving this actual state, the traffic simulator is triggered to perform one simulation step and get the traffic states at the next time step $t + \Delta t$. The next time step traffic states are forwarded to the CAV Application Client through the Real-Sim interface. Also, the Application Layer of the interface sends the control commands to the traffic actor simulated by the CAV Application Client. These are represented by the top blue arrow and orange arrow pointing left to right. Before receiving the traffic states and control commands, the previous traffic states and control commands were held. The CAV application kept on running as its time step is much faster than the traffic simulator. Once the updated traffic states and control commands are received, they are interpolated, and the CAV Application Client simulates until it reaches simulation time step $t + 2\Delta t$. The holding-interpolation mechanism acts as a filter to ensure a smooth control command and addresses any interface communication delays. The next section further demonstrates this synchronization and explains the delay handling mechanism.

Delay Handling Mechanism

Figure 5 shows what will occur with no delay and one step delay in the Real-Sim interface. In the "no delay" scenario, synchronization is performed as expected. The synchronization between Traffic Simulator

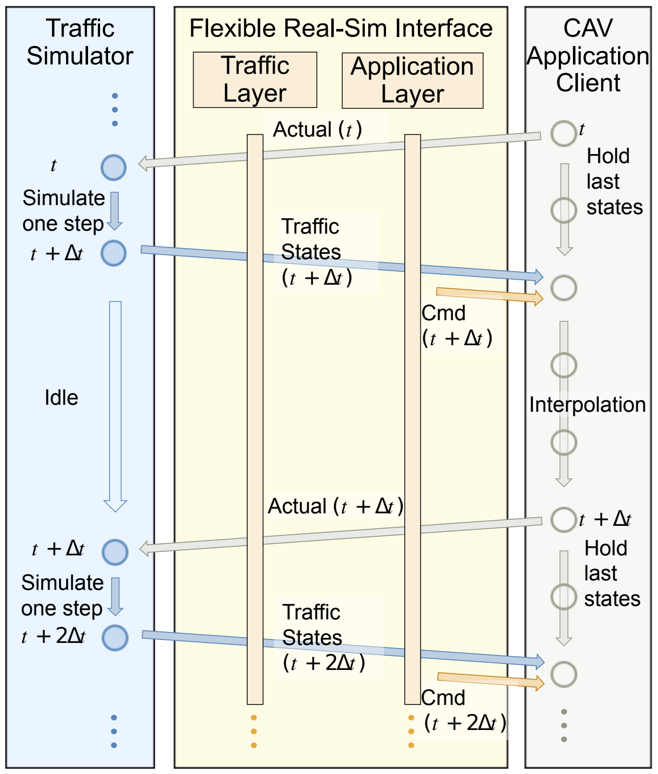


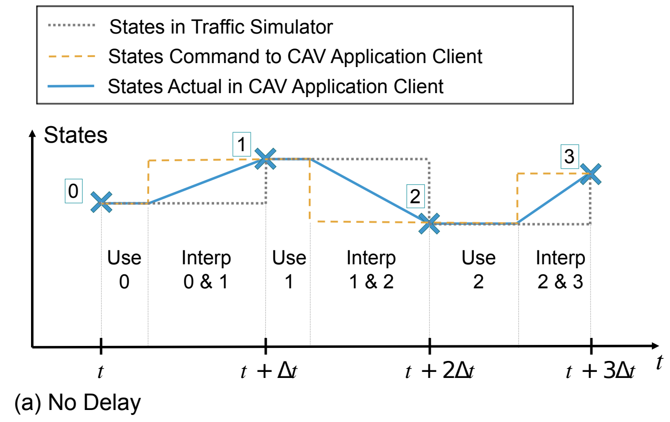
Figure 4: Synchronization Mechanism

and CAV Application Client is accomplished every Δt seconds, which is the time step of the traffic simulator. For example, at time $t + \Delta t$ simulation time in Figure 5(a), the CAV Application Client holds the previous traffic state (labelled as state "1") for a short period (labelled as "Use 1") before receiving the next traffic state (labelled as state "2"). Once state 2 is received, the CAV Application Client will interpolate the state command and simulate the corresponding traffic actor. During that short "hold last state" period, the CAV Application Client sends the actual state 1 to the traffic simulator, and receives the state 2. The duration of this period essentially characterizes the performance of the Real-Sim interface in terms of the round-trip network communication delays between the CAV Application Client and the Traffic Simulator, which is further discussed in "Performance Analysis" section.

In "1 step delay" scenario, because of the "hold last state" mechanism, the state commands will not break down, and there will be minimal concern in safety when operating a HIL testbed. As shown in Figure 5(b), the CAV Application Client waits for the state command "2" until it receives shortly after $t + 2\Delta t$ seconds. The state command is kept as the previous value for safety concerns. The received state command "2" is interpolated toward the next time step $t + 3\Delta t$. During the interpolation, since the state command "3" is received without delay, the CAV Application Client switches the target and interpolate command "3" toward $t + 3\Delta t$.

Performance Analysis

Because the Real-Sim interface is based on network communication protocols, understanding the network communication delay performance is critical. The network communication delay was analyzed for the Real-Sim interface for various numbers of CAV Application Clients. The Application Layer can contain various CAV control algorithms and the CAV Application Clients can include complex vehicle simulator or HIL systems. These algorithms or complex simulators could affect the computational efficiency and network communication delays. To enable throughput testing of the Real-Sim interface, simple Application Layer and CAV Application Clients were used. In other words, the simple Application Layer only relays the traffic informa-


States in Traffic Simulator
States Command to CAV Application Client
States Actual in CAV Application Client
States
0
1
2
3
Use 0
Interp 0 & 1
Use 1
Interp 1 & 2
Use 2
Interp 2 & 3
t
t + Δt
t + 2Δt
t + 3Δt
t
(a) No Delay

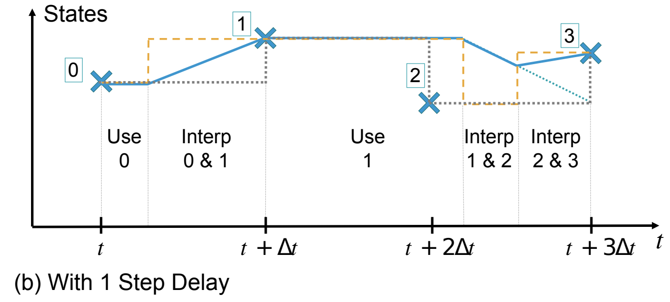
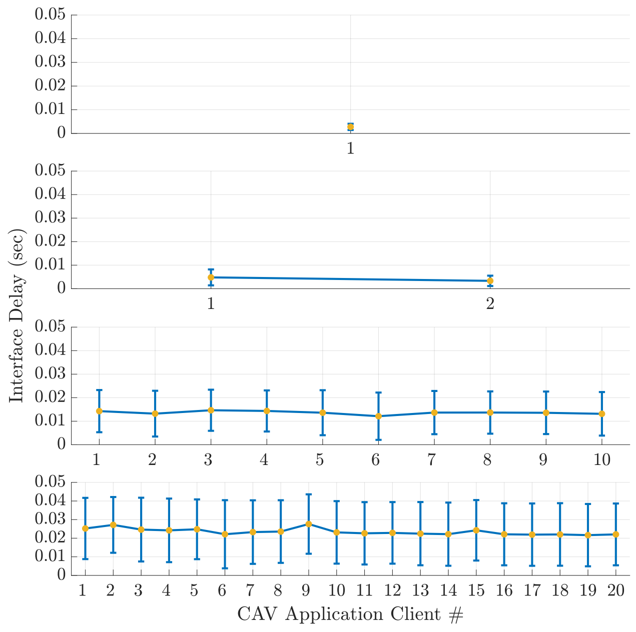


Figure 5: Delay Handling Mechanism

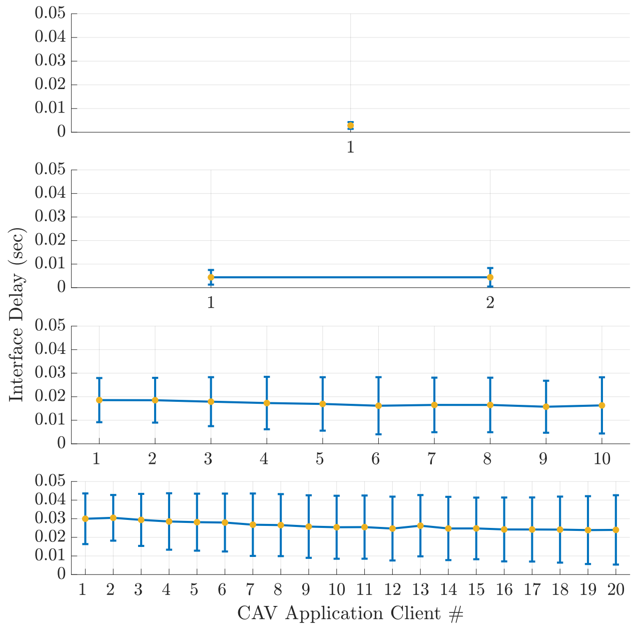
tion to the CAV Application Client without executing any control algorithms. Similarly, once the simple CAV Application Client receives a state command, it replies with the same value to the Application Layer and eventually to the Traffic Simulator. The network communication delay is defined as the round-trip time from the initial time a CAV Application Client sends out a message to the time it receives the next step commands. This amount of time is the duration of the "Use 0", "Use 1", "Use 2" portions shown in Figure 5.

The Real-Sim interface can be implemented in the centralized architecture or the distributed architecture as shown in Figure 2. Network communication delays were analyzed for both architectures for 1, 2, 10, and 20 CAV Application Clients. In the centralized architecture, a single centralized controller connects to all the CAV Application Clients. In the distributed architecture, each CAV Application Client is connected to a controller, and all controllers are connected to the traffic layer. Figure 6a shows the delay of the centralized architecture, and Figure 6b shows the delay of the distributed architecture. The simulation was conducted on a desktop PC with Intel Xeon W2145 CPU @ 3.70GHz and Intel I219-LM Ethernet Adapter for more than 200 seconds with a traffic simulator step size of 0.1 seconds. A total of more than 2,000 delay data points were recorded for every CAV Application Client. The computational burden of the interface is low and the size of the message sent through Ethernet is not significant. So it is anticipated to see similar level of delays on other standard PCs.

The yellow dots in Figure 6a and Figure 6b show the average delay, and the error bars show the 2-σ range, i.e., the upper bound is average delay plus two times standard deviation, and the lower bound is average delay minus two times standard deviation. For all cases, the delays are all below 0.05 seconds, which is well below the traffic simulator time step Δt. This verifies that the current Real-Sim interface can process at least 20 CAV Application Clients at the same time. An example application is that 20 ego vehicles can be simulated with detailed vehicle models and simulators and their benefits can be analyzed with higher fidelity. Figure 7 shows the average delay of each CAV Application Client scenario, e.g., for the 10-Client case, the average delays of all 10 Clients are averaged which is about 0.014 seconds for the centralized architecture and 0.017 seconds for the distributed architecture. It appears that the average delay increases in a near linear rate



(a) Centralized Architecture of the Real-Sim Interface



(b) Distributed Architecture of the Real-Sim Interface

Figure 6: Network Communication Delays: Using Centralized Architecture or Distributed Architecture of the Real-Sim Interface

as the number of clients increases.

The integration of a large volume of simulators with low latency is critical to the development of CAV control technologies. Typically, these technologies are based upon a wide array of inputs from many different sources, such as their onboard systems, the surrounding vehicles (i.e. CDA), and the larger traffic network. Therefore, developing these CAV controls relies on a methodical approach utilizing various rich development environments that enable the interaction of multiple high-fidelity vehicle and traffic models in simulation and hardware. Introducing a real-time constraint vets the operation of these complex CAV controls in a vehicle and traffic environment that supports real time decision making at various time scales. All in all, the Real-Sim interface provides a promising start to developing a framework with the appropriate scaling considerations for developing and evaluating CAV controls.

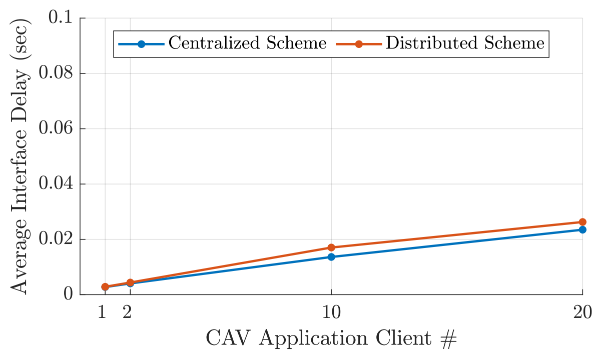


Figure 7: Average Network Communication Delays of Real-Sim Interface

This shows potential to apply and scale the Real-Sim interface to larger numbers of clients, which will be studied in future work.

Example Applications

The Real-Sim interface was applied to two applications. In the first application, the centralized architecture was applied where a centralized merging controller was applied and two dSPACE SCALEXIO racks were used to emulate the dynamics of two ego vehicles. In the second application, the distributed architecture was applied to implement signal-controller-in-the-loop (SCIL) simulation, in which one intersection in the traffic simulator was controlled by a physical signal controller. The two applications are only examples of many potential applications that can be integrated using the Real-Sim interface. They demonstrate the capability and potential usages of the interface. The development of the specific application algorithms are out of the scope of this work.

Application 1: Coordinated Merging with a Centralized Controller

In this application, a centralized controller coordinates the arrivals of vehicles around a merging ramp to improve energy efficiency and mobility. The algorithm is illustrated in Figure 8 and are based on existing work at ORNL [13]. It is a type of CDA algorithms where a centralized controller receives information of all vehicles' position inside a control zone near the merging area. The controller coordinates all vehicles' arrival times at the merging area based on the following principle: a vehicle enters the control zone earlier and closer to the merging area will have a higher priority, and hence enter the merging area earlier, regardless of whether it is on the main road or the ramp. For example, in Figure 8, the order will be blue car, white car, black car on the main road, black car on ramp, and the green car. This coordination principle can reduce potential queues and stops at both the main road and the ramp to optimize the traffic flow and capacity at the global level. Once the vehicle merging order is coordinated, the controller also optimizes the desired speed of all CAVs with an analytical solution obtained using the Pontryagin's minimum principle. The detailed algorithms can be referred to [13].

Scenario

A single on-ramp lane merging into a single lane main road scenario was studied with a road speed limit of $50km/h$. As shown in Figure 8, two vehicles from this scenario were selected as ego vehicle—one from the main road, and one from the merge road. These two ego vehicles were modeled as Class 8 trucks, and detailed dynamics models were simulated using two dSPACE SCALEXIO racks. ORNL has developed a library of high-fidelity vehicle dynamics models for various types of vehicles, and the Class 8 truck model leveraged in this scenario is in this model library. This model includes a full conventional vehicle plant model, high fidelity vehicle dynamics, and a TCP/IP hardware interface for communication with external tools. The engine is modeled based on torque and fuel consumption maps derived from experimental data collected at ORNL, and other components leverage representative data available for use in the Class 8 model.

When determining the controller commands, the centralized controller assumes simplified vehicle dynamics. The traffic simulator usually focuses on the traffic level performance (e.g., flow, travel time, delay, etc.) with simplified vehicle dynamics as well. It is challenging to evaluate the performance of the controller under detailed dynamics that is close to the real-world. The Real-Sim interface can help the inclusion of two detailed Class 8 vehicle models with the traffic simulator with small lift from the users. Figure 8 shows the information exchange among the ego vehicle simulators and the traffic simulator. The detailed Class 8 truck dynamics were simulated on the dSPACE SCALEXIO, which is the CAV Application Client in this application, whereas the remaining vehicles were simulated in the traffic simulator SUMO. The coordinated merging algorithm was implemented in the Application Layer of the Real-Sim interface. The merging algorithm receives states of all traffic inside the control zone around the merging point and optimizes the arrival time and desired speed. The desired speed commands for the two selected ego vehicles are forwarded to the Class 8 truck model simulated on the dSPACE boxes and the desired speed commands for the remaining vehicles in the control zone are sent to the traffic simulator. The Class 8 truck model uses a driver model to control the vehicle accelerator and brake pedal commands to follow the desired speed commands. The actual vehicle speed simulated by the Class 8 truck model with high fidelity vehicle dynamics is sent to the Real-Sim interface and then to the traffic simulator. Thus, the two ego vehicles inside the traffic simulator are synchronized with the dynamics simulated on the dSPACE boxes.

Results

Figure 9 shows the comparison of command speed, actual speed, and simulation speed for the two ego trucks, as well as the corresponding gear positions simulated by the detailed vehicle model on the dSPACE SCALEXIO. The simulation speed is the vehicle speed recorded by complete SUMO simulation without Real-Sim interface and the detailed models on dSPACE. The actual speed and command speed were obtained by co-simulation of the SUMO and two dSPACE with detailed vehicle models, as shown in Figure 9. Based on the coordination principle described earlier, the centralized controller would request deceleration of vehicles on either the main road or the ramp to avoid conflicts during the merging and maximize the overall traffic throughput. This is the reason why both ego vehicles have deceleration around 70-110 seconds. Figure 9 shows that the actual vehicle speed can generally follow the quadratic-shaped optimal speed command. However, the actual speed is different than the speed profile purely from simulation. The overshoot behaviors and the delayed deceleration of the actual speed profiles are because Class 8 vehicles possess large amount of inertia. Also, at around 95 seconds, there is a speed dip for both ego vehicles because of the transmission gear shift. This shows the differences between the simplified vehicle model used in SUMO and a well-developed detailed vehicle model. The Real-Sim interface can help establish such co-simulation of traffic simulator with external detailed vehicle models to improve the fidelity of the simulation and provide better evaluation on the CAV applications. The traffic scenario is defined by the traffic simulator and users can vary either the traffic flow conditions directly within the traffic simulator or replace the algorithm in the Application Layer of the Real-Sim interface to change the behavior of individual ego vehicle. The network communication delay for this two-client centralized architecture scenario is around 0.005 seconds, which is in the similar range as those shown in Figure 6a.

Application 2: Signal-controller-in-the-loop Evaluation

In this application, a SCIL simulation was performed with the Real-Sim interface. Existing microscopic traffic simulation uses software-based signal controllers to emulate the operations of modern con-

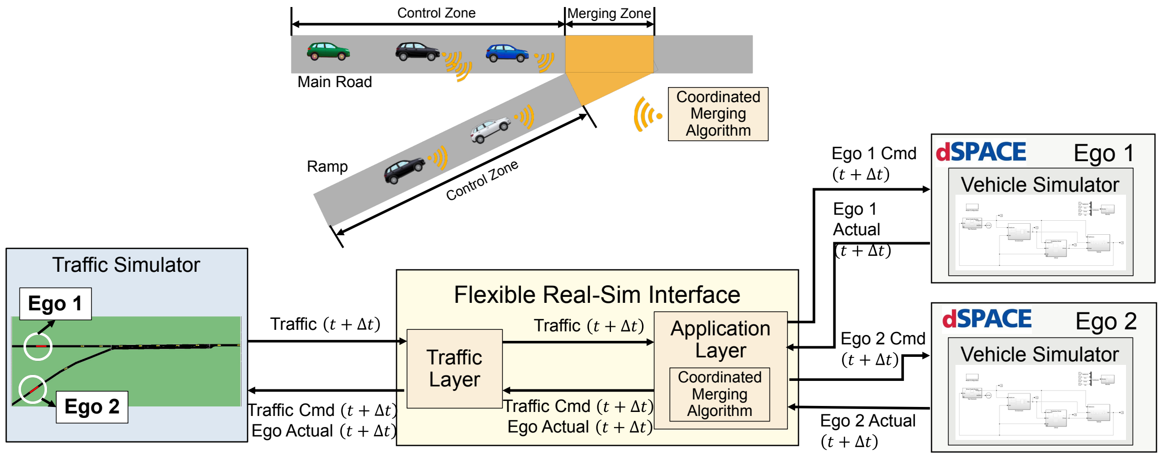

Control Zone
Merging Zone
Main Road
Ramp
Control Zone
Coordinated Merging Algorithm
Traffic Simulator
Ego 1
Ego 2
Traffic $(t + \Delta t)$
Traffic Cmd $(t + \Delta t)$
Ego Actual $(t + \Delta t)$
Flexible Real-Sim Interface
Traffic Layer
Application Layer
Coordinated Merging Algorithm
Ego 1 Cmd $(t + \Delta t)$
Ego 1 Actual $(t + \Delta t)$
Ego 2 Cmd $(t + \Delta t)$
Ego 2 Actual $(t + \Delta t)$
dSPACE Ego 1
Vehicle Simulator
dSPACE Ego 2
Vehicle Simulator

Figure 8: Diagram and Message Flow of the Coordinated Merging Application

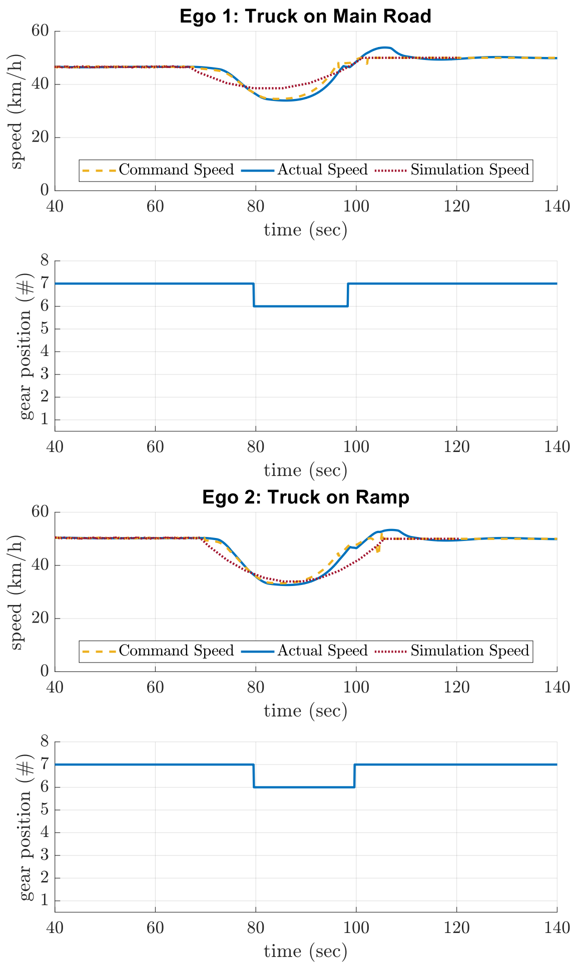


Figure 9: Simulation Results of Two Ego Trucks with Detailed Dynamics Models.

trollers. However, different signal controllers have complex built-in logic and advanced features that are often not well-captured by simulation software [36, 37]. For example, to implement coordination among multiple intersections, mechanisms and logic are needed to synchronize the controller clocks with a reference clock. Depending on the type of controllers and their configuration, ongoing automatic adjustment of the clocks takes place in the real-world and cannot be easily replicated in a software-based simulation approach. SCIL enables realistic testing and validation of applications, such as optimized signal control [38] and vehicle-signal coordination for eco-driving, using real-world advanced traffic signal controllers.

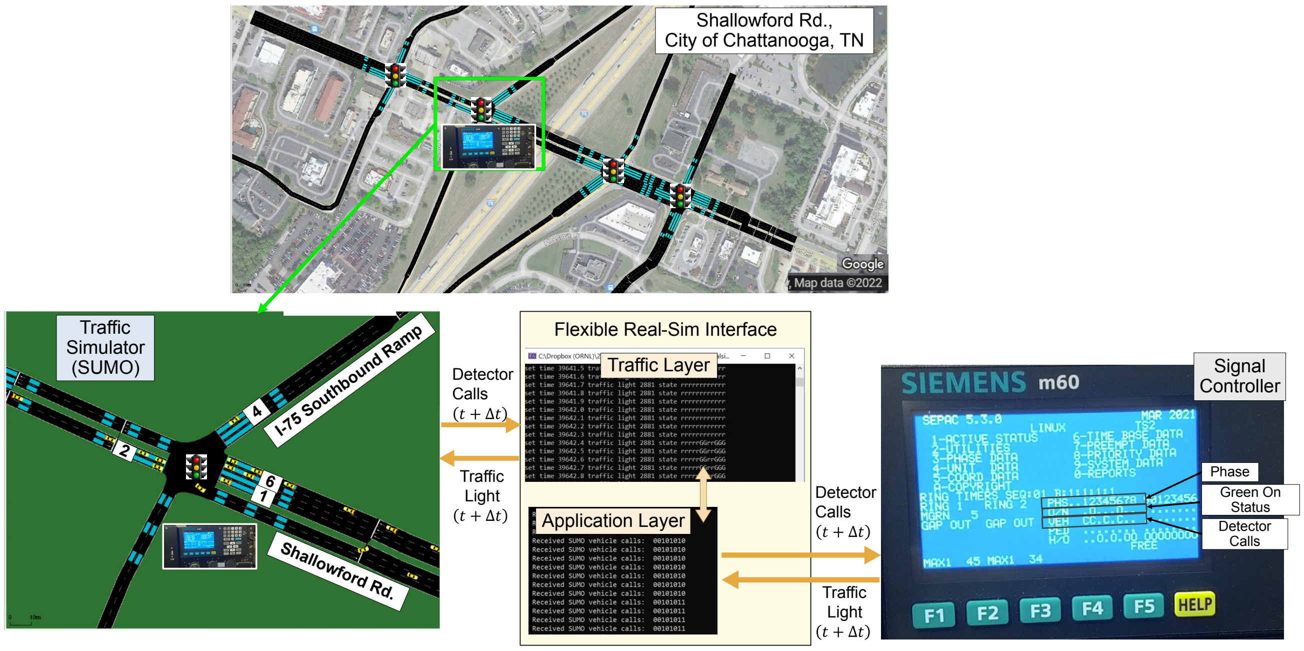
Scenario

The traffic scenario was conducted in SUMO and simulated based on real-world data from a corridor at Shallowford Rd. in Chattanooga, Tennessee [39, 40]. Figure 10 shows the traffic network in SUMO which contains total of four signalized intersections. It replicates traffic elements and operations of the real-world in a software environment. The components include (1) a road network that was created from geographic information systems (GIS) files of the network; (2) signal controls based on signal timing plans that are currently implemented in the field; and (3) traffic volumes and turning movements at different times of day based on data collected by smart traffic sensors deployed along the corridor.

In this application, one of the signal controllers in the traffic simulator SUMO was replaced with a physical signal controller using the SCIL setup, as shown in Figure 10. A Siemens M60 Advanced Traffic Controller identical to those in the field is set up in the laboratory and has two-way communications with the traffic simulator through the Real-Sim interface. The signal controller is the CAV Application Client to the Real-Sim interface. As the traffic simulation executes in SUMO, detector triggering events are sent as vehicle calls to the Real-Sim interface and then the Client signal controller. The signal controller responds to these calls as if it receives real vehicle calls from detectors in the actual field application. The real-time signal status is continuously shared with the Real-Sim interface and then the traffic simulator at 0.1 seconds intervals, which is the simulation step Δt of the traffic simulator. The Application Layer utilizes the NTCIP server, an existing effort between ORNL and Siemens, to communicate with the M60 controller.

Results

Figure 10 shows the screenshots during the actual SCIL evaluation. The screenshots of the traffic simulator (SUMO) shows the intersection controlled by the signal controller. The intersection has four signal phases: 1,2,4,and 6, which correspond to westbound left, eastbound through, westbound through, and southbound approaches, respectively. The numbers next to the stop bar detectors indicate the corresponding call phases. The Traffic Layer image shows that SUMO's real-time signal state is '`rrrrrGGrrGGG`', which is equivalent to a green light for phases 2 and 6. This status is the same as the active "Green On Status" of the signal controller. The Application Layer window displays the received detector calls for all 8 phases. Each digit indicates a phase detector call in which the rightmost one is phase 1. In this specific case, all 4 phases have vehicles on detectors, so the calls the Application Layer received were '`00101011`'. This phase detector call synchronizes with those shown in the signal controller screenshot on



```
C:\Dropbox (ORNL)\2                                     alsi...
set time 39641.5 tra                                    rr
set time 39641.6 tra                                    rr
set time 39641.7 traffic light 2881 state rrrrrrrrrrr
set time 39641.8 traffic light 2881 state rrrrrrrrrrr
set time 39641.9 traffic light 2881 state rrrrrrrrrrr
set time 39642.0 traffic light 2881 state rrrrrrrrrrr
set time 39642.1 traffic light 2881 state rrrrrrrrrrr
set time 39642.2 traffic light 2881 state rrrrrrrrrrr
set time 39642.3 traffic light 2881 state rrrrrrrrrrr
set time 39642.4 traffic light 2881 state rrrrrGGrrGGG
set time 39642.5 traffic light 2881 state rrrrrGGrrGGG
set time 39642.6 traffic light 2881 state rrrrrGGrrGGG
set time 39642.7 traffic light 2881 state rrrrrGGrrGGG
set time 39642.8 traffic light 2881 state rrrr  rGGG
```

```
Received SUMO vehicle calls:  00101010
Received SUMO vehicle calls:  00101010
Received SUMO vehicle calls:  00101010
Received SUMO vehicle calls:  00101010
Received SUMO vehicle calls:  00101010
Received SUMO vehicle calls:  00101010
Received SUMO vehicle calls:  00101010
Received SUMO vehicle calls:  00101011
Received SUMO vehicle calls:  00101011
Received SUMO vehicle calls:  00101011
Received SUMO vehicle calls:  00101011
```

Figure 10: Diagram of the Signal-controller-in-the-loop Implementation

the right. The signal controller receives these detector calls and react and control the signal status and changes accordingly. This SCIL application in this paper focuses on demonstrating the capability of the Real-Sim interface. For simplicity, the Application Layer only relays the information between traffic simulator and the signal controller. Future work is planned to implement advanced signal control algorithms (e.g., those in [38]) to the Application Layer.

Discussion

As deomonstrated, the current Real-Sim interface is capable of multi-resolution vehicle and traffic co-simulation. However, the evaluation and simulation of CAV applications involve many other aspects currently within the scope of the Real-Sim interface. In this section, several critical aspects are discussed for potential future expansion of the flexible Real-Sim interface.

These applications stress the importance of enabling communication network simulation or communication-hardware-in-the-loop. The performance of a CAV application depends on the information through connectivity, which can be affected by the quality of communication. The inclusion of a Communication Layer in the Real-Sim interface will help evaluate the impacts of communication delays, packet drops, and latency on CAV applications. A Communication Layer could potentially be implemented between the Traffic Layer and Application Layer. In this example, the Application Layer receives information from the Communication Layer rather than directly from the traffic simulator. Since all components are modular and connected through network communication protocol, adding another layer to the Real-Sim interface would be relatively straightforward and would not affect the current implementation of other components.

Perception sensors are another critical resource to collect and parse real-time traffic information. Sensors provide information to the Real-Sim interface's Application Layer, which contains the vehicle level controller strategies. The Application Layer interacts with the sensors mounted on a CAV operating within the virtual vehicle environment in any CAV Application Client. As a result, the CAV sensor data can be made available to the Application Layer through raw data streaming interfaces with the virtual vehicle environment. Additionally, data formatting services process the raw data of the virtual environment sensor models into a communications protocol that the real CAV sensors would output in a real test scenario. This formatting process ensures that as the CAV controller is developed it can interpret simulated data and real data with minimal added work on the controller interface layer. Research at ORNL is being conducted toward developing the data formatting services that interpret the raw CAV sensor data into standard protocols of various sensors. This work is proceeding on a sensor-by-sensor basis currently focused on lidar and camera integration.

Performing large-scale CAV evaluation is important to fully understand the impacts of CAV applications to the traffic flow or large areas of traffic networks (e.g., to a city). The Real-Sim interface is modular and designed to be capable of expanding to a large scale implementation, especially with the distributed architecture. The results shown in this work demonstrate that this interface can already simulate at least 20 ego vehicles or other CAV Application Clients. To further support large scale implementation or take advantage of high performance computing (HPC), the Real-Sim interface could integrate parallel computation methods or multi-threading programming. Therefore, for example, a traffic simulator could emulate the traffic in a city-level, and the Real-Sim interface could support hundreds of ego CAVs evaluation.

Conclusion

In this work, the flexible Real-Sim interface is developed to enable multi-resolution vehicle and traffic co-simulation for CAV applications. The interface is modular and can integrate various simulation tools and real-time XIL implementation. The integration with other simulators or XIL systems is based on network communication protocols and the network communication delays are analyzed. The results show that the interface can handle more than 20 simulators at the same time with less than 0.05 seconds of delay. This amount of delay is well below the typical traffic simulator update frequency which is 0.1 seconds. This shows that the Real-Sim interface provides a promising start to developing a framework with the appropriate scaling considerations for developing and evaluating CAV controls. Two example applications are studied to demonstrate the potential usage of the Real-Sim interface: a centralized merging application where two dSPACE SCALEXIO racks are used to emulate the dynamics of two ego vehicles; a signal-controller-in-the-loop (SCIL) simulation where one intersection of a digital twin of the Shallowford Rd. corridor in Chattanooga, TN is controlled by a physical signal controller. As future work, the Real-Sim interface will be expanded to integrate communication and sensors simulation as well as large scale simulations.

References

1. S. E. Shladover, "Connected and automated vehicle systems: Introduction and overview," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, vol. 22, pp. 190–200, may 2018.

2. Y. Shao and Z. Sun, "Energy-Efficient Connected and Automated Vehicles: Real-Time Traffic Prediction-Enabled Co-Optimization of Vehicle Motion and Powertrain Operation," *IEEE Vehicular Technology Magazine*, pp. 2–11, 2021.

3. Y. Shao and Z. Sun, "Eco-approach with traffic prediction and experimental validation for connected and autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1562–1572, 2021.

4. T. LaClair, Z. Gao, C. Wang, J. Rios Torres, J. Sanyal, R. Karthik, P. Nugent, S. Ravulaparthy, and A. Berres, "Development of a real-time mobility control and visualization system with predictive vehicle speed control for connected and automated vehicles (cavs)," in *The 32nd International Electric Vehicle Symposium (ESV32)*, 2019.

5. Y. Shao, M. A. Mohd Zulkefli, and Z. Sun, "Vehicle and Powertrain Optimization for Autonomous and Connected Vehicles," *Mechanical Engineering*, vol. 139, pp. S19–S23, 09 2017.

6. J. Hu, Y. Shao, Z. Sun, M. Wang, J. Bared, and P. Huang, "Integrated optimal eco-driving on rolling terrain for hybrid electric vehicle with vehicle-infrastructure communication," *Transportation Research Part C: Emerging Technologies*, vol. 68, pp. 228–244, 2016.

7. X. Qi, G. Wu, P. Hao, K. Boriboonsomsin, and M. J. Barth, "Integrated-connected eco-driving system for phevs with co-optimization of vehicle dynamics and powertrain operations," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 1, pp. 2–13, 2017.

8. M. R. Amini, Q. Hu, H. Wang, Y. Feng, I. Kolmanovsky, and J. Sun, "Experimental Validation of Eco-Driving and Eco-Heating Strategies for Connected and Automated HEVs," *SAE Technical Papers*, apr 2021.

9. Y. Shao and Z. Sun, "Vehicle Speed and Gear Position Co-optimization for Energy Efficient Connected and Autonomous Vehicles," *IEEE Transactions on Control Systems Technology*, pp. 1–12, 2020.

10. S. E. Li, S. Xu, X. Huang, B. Cheng, and H. Peng, "Eco-departure of connected vehicles with v2x communication at signalized intersections," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5439–5449, 2015.

11. A. Vahidi and A. Sciarretta, "Energy saving potentials of connected and automated vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 822–843, oct 2018.

12. SAE, "Taxonomy and definitions for terms related to cooperative driving automation for on-road motor vehicles," surface vehicle information report, SAE, May 2020.

13. J. Rios-Torres and A. A. Malikopoulos, "Automated and Cooperative Vehicle Merging at Highway On-Ramps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 780–789, apr 2017.

14. Y. Shao and J. Rios-Torres, "Traffic Prediction for Merging Coordination Control in Mixed Traffic Scenarios," vol. 2 of *Dynamic Systems and Control Conference*, 10 2020. DSCC2020-3219.

15. A. A. Malikopoulos, C. G. Cassandras, and Y. J. Zhang, "A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections," *Automatica*, vol. 93, pp. 244–256, jul 2018.

16. A. Tahmasbi-Sarvestani, H. Nourkhiz Mahjoub, Y. P. Fallah, E. Moradi-Pari, and O. Abuchaar, "Implementation and evaluation of a cooperative vehicle-to-pedestrian safety application," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 4, pp. 62–75, 2017.

17. Y. Shao, M. A. Mohd Zulkefli, Z. Sun, and P. Huang, "Evaluating connected and autonomous vehicles using a hardware-in-the-loop testbed and a living lab," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 121–135, 2019.

18. M. A. Mohd Zulkefli, P. Mukherjee, Z. Sun, J. Zheng, H. X. Liu, and P. Huang, "Hardware-in-the-loop testbed for evaluating connected vehicle applications," *Transportation Research Part C: Emerging Technologies*, vol. 78, pp. 50–62, 2017.

19. M. A. Mohd Zulkefli, P. Mukherjee, Y. Shao, and Z. Sun, "Evaluating Connected Vehicles and Their Applications," *Mechanical Engineering*, vol. 138, pp. S12–S17, 12 2016.

20. J. Zhao, C. F. Chang, R. Rajkumar, and J. Gonder, "Corroborative Evaluation of the Real-World Energy Saving Potentials of InfoRich Eco-Autonomous Driving (iREAD) System," *SAE Technical Papers*, vol. 2020-April, apr 2020.

21. J. Rios-Torres and A. A. Malikopoulos, "Impact of partial penetrations of connected and automated vehicles on fuel consumption and traffic flow," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 4, pp. 453–462, 2018.

22. Y. Shao and Z. Sun, "Optimal Eco-Approach Control With Traffic Prediction for Connected Vehicles," vol. 2 of *Dynamic Systems and Control Conference*, 10 2018. DSCC2018-9059.

23. Y. Feng, C. Yu, S. Xu, H. X. Liu, and H. Peng, "An Augmented Reality Environment for Connected and Automated Vehicle Testing and Evaluation," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2018-June, pp. 1549–1554, oct 2018.

24. J. Ma, F. Zhou, Z. Huang, C. L. Melson, R. James, and X. Zhang, "Hardware-in-the-loop testing of connected and automated vehicle applications: A use case for queue-aware signalized intersection approach and departure," *Transportation Research Record*, vol. 2672, no. 22, pp. 36–46, 2018.

25. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

26. G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta, *et al.*, "Lgsvl simulator: A high fidelity simulator for autonomous driving," *arXiv preprint arXiv:2005.03778*, 2020.

27. IPG, "IPG Automotive," 2019.

28. VIRES, "VTD-Virtual Test Drive kernel description," 2019.

29. dSPACE, "Automotive simulation models," 2019.

30. P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, IEEE, 2018.

31. PTV AG, Karlsruhe, Germany, *PTV VISSIM 11 User Manual*, 2019.

32. Aimsun, *Aimsun Next 20 User's Manual*. Barcelona, Spain, aimsun next 20.0.3 ed., 2021. [In software].

33. VIRES, "OpenDRIVE kernel description," 2019.

34. Eclipse MOSAIC, "Eclipse MOSAIC," 2020.

35. D. Deter, C. Wang, A. Cook, and N. K. Perry, "Simulating the autonomous future: A look at virtual vehicle environments and how to validate simulation using public data sets," *IEEE Signal Processing Magazine*, vol. 38, no. 1, pp. 111–121, 2020.

36. P. Li and P. B. Mirchandani, "A new hardware-in-the-loop traffic signal simulation framework to bridge traffic signal research and practice," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 9, pp. 2430–2439, 2016.

37. D. Wang, Z. Tian, and G. Yang, "Virtual controller interface device for hardware-in-the-loop simulation of traffic signals," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 2, pp. 201–216, 2021.

38. H. Wang, M. Zhu, W. Hong, C. Wang, G. Tao, and Y. Wang, "Optimizing signal timing control for large urban traffic networks using an adaptive linear quadratic regulator control strategy," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

39. A. S. Berres, T. J. LaClair, C. Wang, H. Xu, S. Ravulaparthy, A. Todd, S. A. Tennille, and J. Sanyal, "Multiscale and multivariate transportation system visualization for shopping district traffic and regional traffic," *Transportation Research Record*, vol. 2675, no. 6, pp. 23–37, 2021.

40. H. Xu, A. Berres, C. Wang, T. J. LaClair, and J. Sanyal, "Visualizing vehicle acceleration and braking energy at intersections along a major traffic corridor," in *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, pp. 401–405, 2021.