Interfacing Synchrono and NADS for Virtual Simulation of Conventional & Connected and Autonomous Vehicles

SAFETY RESEARCH USING SIMULATION

UNIVERSITY TRANSPORTATION CENTER

Dan Negrut, PhD
Professor
Department of Mechanical Engineering
University of Wisconsin-Madison

Chris Schwartz, PhD
Director of Engineering and Modeling Research
National Advanced Driving Simulation,
University of Iowa

Interfacing SynChrono and NADS for Virtual Simulation of Conventional & Connected and Autonomous Vehicles

Dan Negrut, PhD
Professor
Department of Mechanical Engineering
University of Wisconsin-Madison
https://orcid.org/0000-0003-1565-2784

Chris Schwarz, PhD
Director of Engineering and Modeling Research
National Advanced Driving Simulator,
University of Iowa
https://orcid.org/0000-0003-1565-2784

Radu Serban, PhD
Senior Scientist
Department of Mechanical Engineering
University of Wisconsin-Madison
https://orcid.org/0000-0002-4219-905X

Simone Benatti, PhD
Postdoctoral Research Associate
Department of Mechanical Engineering
University of Wisconsin-Madison
https://orcid.org/0000-0001-9449-1688

Asher Elmquist
Graduate Research Assistant Department of Mechanical Engineering University of Wisconsin-Madison
https://orcid.org/0000-0002-0142-1865

A Report on Research Sponsored by

SAFER-SIM University Transportation Center

Federal Grant No: 69A3551747131

September 2021

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. C-2-Y4	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle Interfacing SynChrono and NADS for Virtual Simulation of Conventional & Connected and Autonomous Vehicles		**5. Report Date** September 2021
		6. Performing Organization Code Enter any/all unique numbers assigned to the performing organization, if applicable.
7. Author(s) Dan Negrut, https://orcid.org/0000-0003-1565-2784 Chris Schwarz, https://orcid.org/0000-0003-1565-2784 Radu Serban, https://orcid.org/0000-0002-4219-905X Simone Benatti, https://orcid.org/0000-0001-9449-1688 Asher Elmquist, https://orcid.org/0000-0002-0142-1865		**8. Performing Organization Report No.** Enter any/all unique alphanumeric report numbers assigned by the performing organization, if applicable.
9. Performing Organization Name and Address University of Wisconsin-Madison, Madison, WI USA University of Iowa, Iowa City, IA USA		**10. Work Unit No.**
		11. Contract or Grant No. Safety Research Using Simulation (SAFER-SIM) University Transportation Center (Federal Grant #: 69A3551747131)
12. Sponsoring Agency Name and Address Safety Research Using Simulation University Transportation Center Office of the Secretary of Transportation (OST) U.S. Department of Transportation (US DOT)		**13. Type of Report and Period Covered** Final Research Report (September 2020 – September 2021)
		14. Sponsoring Agency Code

15. Supplementary Notes

This project was funded by Safety Research Using Simulation (SAFER-SIM) University Transportation Center, a grant from the U.S. Department of Transportation – Office of the Assistant Secretary for Research and Technology, University Transportation Centers Program.

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. government assumes no liability for the contents or use thereof.

16. Abstract

This project enabled the development of a software infrastructure that allows for the study for mixed human-driven and autonomously driven traffic scenarios, by interfacing the National Advanced Driving Simulator's driving simulator in Iowa with a Chrono simulation running on a compute cluster in Madison, WI. Since Chrono is capable of simulating multiple vehicles including vehicle dynamics and sensor, using local distribution, it is suitable for simulating convoys of autonomous vehicles. The interface with NADS additionally enables the investigation of traffic scenarios in which human-driven vehicles interact with autonomous vehicles.

17. Key Words Simulation; Computer simulation USE Simulation; Distributed Simulation; Human in the loop Simulation		**18. Distribution Statement** No restrictions. This document is available through the SAFER-SIM website, as well as the National Transportation Library	
19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 26	**22. Price**

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Table of Contents

List of Figures

List of Tables

Abstract

This project enabled the development of a software infrastructure that allows for the study for mixed human-driven and autonomously driven traffic scenarios, by interfacing the National Advanced Driving Simulator's driving simulator in Iowa with a Chrono simulation running on a compute cluster at the University of Wisconsin-Madison, WI. Since Chrono can simulate multiple vehicles including vehicle dynamics and sensor, it is suitable for simulating convoys of autonomous vehicles. The interface with NADS additionally enables the investigation of traffic scenarios in which human-driven vehicles interact with autonomous vehicles. This is a first step in a process that could lead to high-fidelity simulation scenarios that combine human in the loop with complex traffic scenarios. This could be used to understand the interplay between the automation and vehicle operator.

All software developed under this project is available free for use/distribution/modification under a BSD3 license in the Project Chrono GitHub repository.

1 Introduction

1.1 Project Goals

The main goal of the project was enabling geographically distributed simulation between the NADS simulator, an advanced driving simulator at the University of Iowa, and a computer cluster located at the University of Wisconsin-Madison, called Euler, to investigate the interaction between manned vehicles and convoys of autonomous vehicles.

1.2 Purpose

As the autonomy level and number of self-driving cars grow, so does the necessity of predicting the interplay of these with human driven vehicles. This project aims to simulate this interaction, reproducing in a virtual environment scenario that would be otherwise costly and dangerous to replicate in the real world. The accurate physics and sensor simulation provided by Project Chrono [7] and the immersive and realistic driving simulation offered by NADS allow to greatly reduce the simulation-reality gap. The simulation at UW-Madison relies on the physics engine Chrono to simulate the vehicle dynamics for all the AVs. A second software layer called SynChrono [6] manages the overall simulation of all Chrono vehicles so that they participate in a joint simulation that stay time-coherent and space-coherent. “Time coherence” means that the simulated vehicles perceive the same global time, which avoids some vehicles slipping into the future while others being laggards. “Space coherence” means that each vehicle can sense (see via camera, lidars, etc.) the other vehicles that participate in a joint traffic scenario.

1.3 Project Specific Communication Requirements

Simulating the human-in-the-loop interaction between a human-driven vehicle in the NADS simulator and AV convoy of vehicle simulated on the Euler High-Performance computer cluster poses a series of specific challenges:

1. Euler-NADS communication

   The state of the vehicles simulated by SynChrono must be passed to NADS and vice-versa.

2. Double Layer communication

   On SynChrono side, two distinct communication layers must coexist without interference: the first is responsible for passing information between NADS and

SynChrono; the second exchanges vehicle states between the vehicles participating in the SynChrono simulation experiment.

3. Head-Node Communication

While the SynChrono simulation processes happen on the compute nodes of a supercomputer, these nodes cannot communicate externally, thus all the information to be exchanged between SynChrono and NADS must pass through Euler's Head-Node.

2 The National Advanced Driving Simulator

2.1 NADS

The NADS-1 high-fidelity motion-base at the National Advanced Driving Simulator (NADS) was used for this study (see Figure 1). The NADS-1 utilizes an actual vehicle cab and projects scenery 360 degrees around the driver on the interior walls of a 24-foot diameter dome that houses the cab. The vehicle cab is mounted on four independent actuators that provide vibration associated with driving on varying road surfaces. The entire dome is mounted on a motion base that can independently provide surge, sway, heave, roll, pitch and yaw cues to the driver. The NADS-1 has a 13 degree-of-freedom large excursion motion base that can move around a space with dimension of 20 x 20 meters and generate accelerations up to 0.6 G in all directions. The NADS-1 features a full 2014 Toyota Camry cab with active steering and pedal feedback as well as a programmable center console. Sixteen high-definition (1920x1200) LED (light emitting diode) projectors display seamless imagery on the interior walls of the dome with a 360-degree horizontal field of view. Simulator data can be sampled at rates up to 240 Hz.

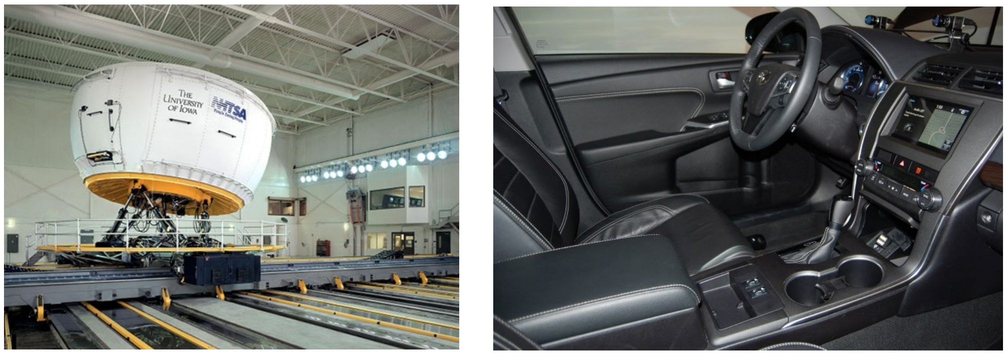

Figure 1 – Exterior and interior views of the NADS-1 simulator

3 SynChrono: distributing vehicle simulation

3.1 Distributed Vehicle Simulation Principles

In many scenarios involving simulating traffic scenarios, it is mandatory to simulate multiple vehicle. Although it is possible to simulate many simplified-physics vehicle on a single machine, this approach has some limitations:

- Physical model fidelity: while a single machine can simulate several reduced-DOF, Ackermann-style vehicles, this is not true for higher fidelity vehicle models. Simulating steering and suspension mechanisms, together with powertrain and tire models is computationally expensive, and real-time performance cannot be achieved on a single machine for multiple vehicles.
- Sensor simulation: when simulating AVs it is necessary to simulate the sensing process in order to provide sensor data to the autonomy stack. Sensor simulation is both computationally and memory demanding (particularly GPU memory).

For these reasons, we distribute the vehicle simulation over several *ranks*. A rank can be one or more cores of a CPU or a computer cluster. The i-th rank communicates the state of its AV to the other ranks, such that sensor simulation running on the other ranks can reconstruct the i-th vehicle presence.

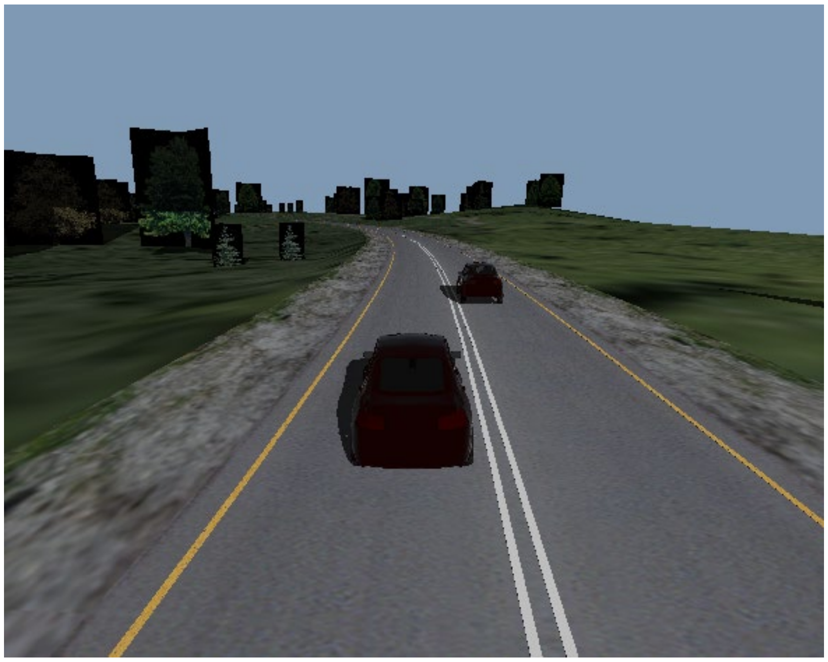

Figure 2 – Two cars in SynChrono

In Figure 2 we see two different vehicles. Only one of these is simulated by the process on which the sensor simulation is running. The other one is simulated by another process that passes position and orientation of the car parts such that it can be reconstructed. This approach allows both human and autonomous drivers to interact with several nearby vehicles while keeping real-time performance.

SynChrono is a Project Chrono module allowing for distributed simulation of vehicles by leveraging the Message Passing Interface (MPI) standard [1] or FastDDS [2] (these communication layers are both viable options in SynChrono). The physics and sensor simulation of each vehicle is simulated in a different rank (simulation process) and the information (the state of the bodies simulated within the rank) is packaged via the FlatBuffers [3] serialization library. While this offers high scalability and sensor-level interaction, it allows for no direct physical interaction between agents simulated on different ranks.

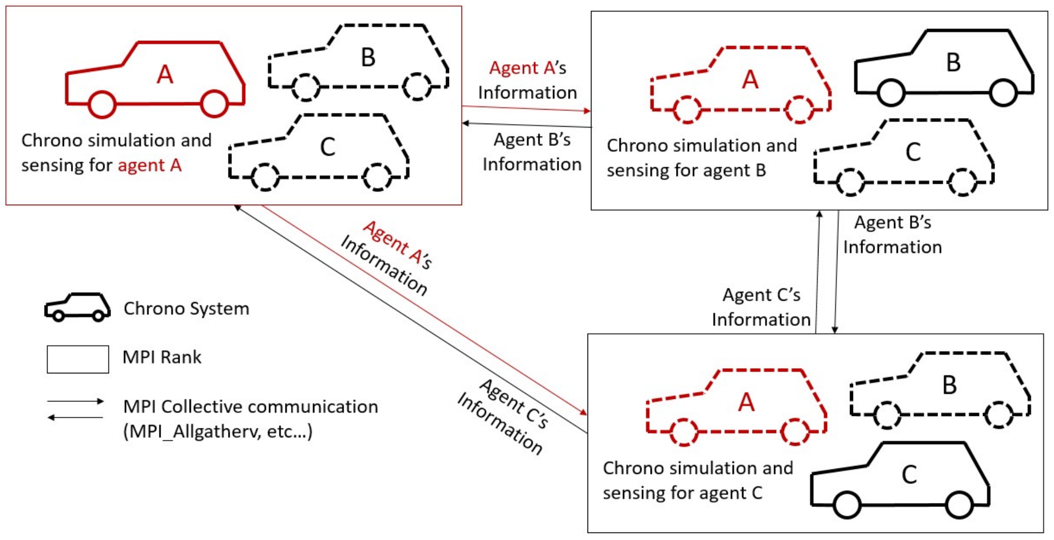
A
B
C
Chrono simulation and sensing for agent A
Agent A's Information
Agent B's Information
A
B
C
Chrono simulation and sensing for agent B
Agent A's Information
Agent B's Information
Agent C's Information
Chrono System
MPI Rank
MPI Collective communication (MPI_Allgatherv, etc...)
Agent C's Information
A
B
C
Chrono simulation and sensing for agent C

Figure 3 – SynChrono paradigm

3.2 FlatBuffers Library

Data serialization is a step required for data to be passed around. In this regard SynChrono relies on FlatBuffers, a serialization library that allows to define objects in a *schema* to generate code that is included in the project to serialize and de-serialize objects.

3.3 SynChrono Communication

While the serialization and de-serialization is achieved through FlatBuffers, it is agnostic with respect to how these buffers are passed. To this extent, SynChrono provides two possible communication layers:

- Message Passing Interface (MPI)
- Data Distribution Service (DDS) [4]

MPI is the de facto standard for distributed computing over homogeneous compute nodes while DDS is typically a more general communication paradigm between compute systems. In both cases, we enforce synchronism, forcing the simulation processes to step forward together at each *hearthbeat* (typically around 10 Hz). In other words, every tenth of second, the state of each vehicle is made known to all the processes contributing to the simulation. Thanks to this approach we ensure that the state of vehicles not simulated within the process is always up-to-date.

3.4 Euler HPC compute cluster

The Euler Cluster is a small, heterogeneous supercomputer which employs a number of different hardware configurations to facilitate research and education. Euler consists mostly of x86 CPU and NVIDIA GPU servers supplemented by outlier hardware including Arm and PowerPC CPUs and erstwhile accelerators such as Intel Xeon Phi and AMD GPU coprocessors. The servers share over 500 TB of high-availability storage delivered by a 100 Gbps low-latency network.

4 DDS Standard Principles

4.1 What is DDS

Data Distribution Service (DDS) is a data-centric, industry standard communication protocol. Being an industry standard, there are various DDS implementations that can interoperate with each other. This being said, each API is implementation-specific, thus a program using some DDS implementation cannot switch to a different one without modifying the code.

4.2 Publishers, Subscribers, Topics, Dynamic Discovery

In DDS each *Domain* has several Publisher and Subscribers: the first write data, the latter listen to data. The data are published in Topics. Each topic has a name, unique within the Domain, that is used to identify it. The participant discovery happens dynamically – the Subscribers look for Publishers matching their topic name and start listen to their data as soon as they find one. A single publisher might have multiple subscribers reading its data.

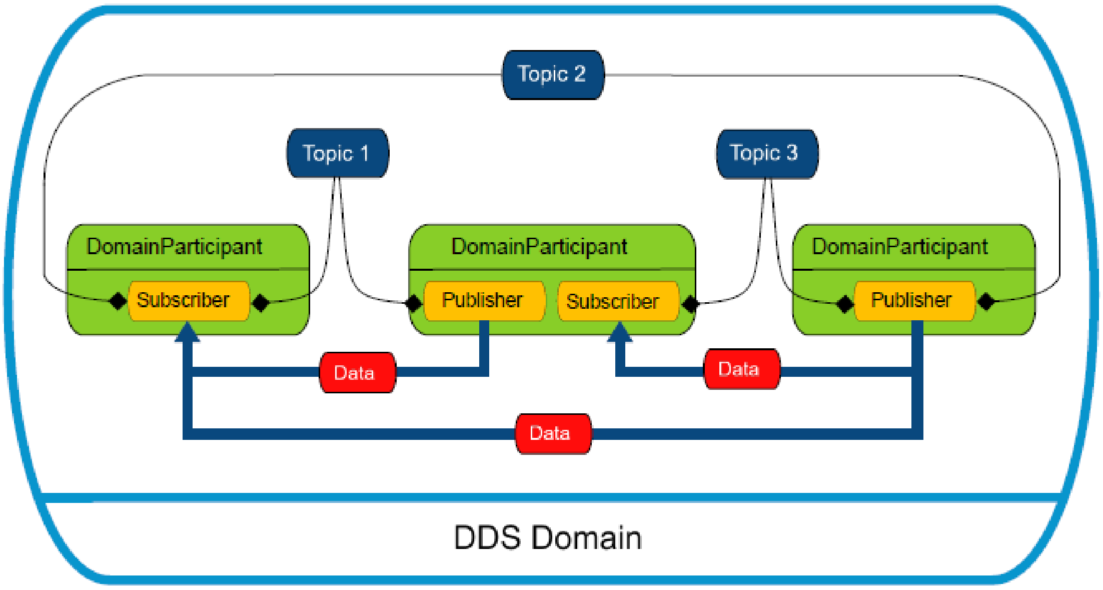

Figure 4 – DDS Communication ©eProsima

4.3 The Quality of Service

DDS is both hardware and protocol agnostic, for example it can run both on Shared Memory between processes on the same CPU, over Ethernet LAN, or over the Internet using UDP or TCP. According to the underlying connection, the DDS user can set the Quality of Service (QOS), adapting the expected reliability, latency, buffer size and protocol to the specific situation. Hence, QoS settings must match:

- The hardware and protocol of the underlying communication layer.

  A very low QoS latency, working perfectly fine for a Shared Memory application, will not work over a low-quality internet connection. In the same way, a reader set to reliable but working over a geographically distributed UDP will likely face issues, since packages might be lost.

- QoS setting of the corresponding reader/writer:

  In principle each participant within a domain can have a different QoS policy. This being said, for a Subscriber to read data from a Publisher, their QoS must be compatible.

4.4 Pros and Cons

DDS is arguably a lower level communication protocol compared to MPI; it is more flexible but requires more hand-tuning. In other words, DDS can allow for communication over different machines that do not share the same local network, on virtually any physical layer.

5 SynChrono-DDS: distributing simulation through DDS

5.1 Synchronous, Reliable, Local communication

SynChrono enforces synchronism even when running on DDS by getting the state of each rank (participant) at each heartbeat. This is achieved by waiting until every subscription has read its message before stepping forward. Since each simulation process, at each heartbeat, waits for an update from all the ranks before stepping forward, the QoS reliability policy of both the reader and the writer ensures robust behavior; in other words, packages that get lost will be re-sent. For this reason, enforcing both real-time and coherence requires low latency and high reliability connection, which is typically achievable on CPU shared memory or fast ethernet LAN.

Currently, SynChrono DDS can run several processes over multiple nodes on the Euler computer cluster and still maintain time-space coherence while achieving real-time performance.

6 The DDS-based NADS/Euler interface

6.1 Principles

1. Euler HPC infrastructure communication
    a. Simulation on Compute-nodes

       Euler is a computing infrastructure, composed of several *compute-nodes*, each one of those is equipped with modern CPUs and GPUs to perform HPC duties.

    b. Communication through head-node

       Compute nodes cannot communicate to eternal computer. The users of the HPC infrastructure connect to the Head-Node through SSH. The head-node is also responsible of managing the users job request across the compute-nodes.

   This poses a first obstacle: the SynChrono simulation processes, running on the compute-nodes, cannot directly communicate with NADS, while we cannot run the simulation in the head-node, which is not meant to perform HPC and whose use is shared by any user connecting to the infrastructure.

   To solve this, we adopted a two-level communication: on the head-node, we run a pure communication process, responsible for receiving the NADS vehicle state and sending to NADS the states of the vehicles simulated in Chrono. At the same time, it interacts with the SynChrono processes by collecting their vehicles' state and sending the NADS vehicle state. From an implementation point of view, the head-node runs a “fake” SynChrono simulation, since the vehicle is actually simulated by NADS. This setup allowed us to perform computationally demanding simulation on the compute-node and manage external communication through the head-node, while preventing it from computational duties at the same time.

In addition, NADS and SBEL labs rely on two different implementations of DDS, RTI Connext DDS and eProsima FastDDS, respectively. While DDS implementation are *not* interchangeable at the implementation level, DDS is a standard, thus two DDS participants can exchange messages regardless of the DDS implementation. In other words, DDS implementations are mutually compatible, if they agree on the message structure, which is defined in a message definition file (.idl file).

2. Two level DDS communication

   While solving the aforementioned issues, using this double layer communication poses a new series of complications:

   a. Guarding against interference

      The two different DDS processes must coexist without interfering. There are two critical phases during which DDS communication on different layer should not be affect each other:

      i. Barriers

         Each SynChrono rank waits for DDS to find a number of participants equal to the expected number of simulated vehicles before starting

ii. Subscription

SynChrono DDS ranks subscribe automatically to SynChrono publishers

To avoid both these issues, we changed SynChrono DDS such that each topic name has a */prefix/suffix* structure and both barrier and subscription phases check prefix correspondence before moving forward. For example, the NADS-Euler process prefix is "*/server-client/*", hence its barrier is not affected by SynChrono participants on compute-nodes, whose prefix is "*/synchrono/*"

b. Different QoS

Communication between NADS and Euler has higher latency and less reliability than internal Euler communication. For this reason, we:

i. Changed SynChrono API to accommodate custom QoS configuration

ii. Set the reader QoS of the NADS subscriber reliability to *BEST_EFFORT*, differently from SynChrono standard *RELIABLE*. This was necessary since enforcing all packages to be received on an internet UDP connection proved to be too strict to be accomplished, and block the NADS-Euler communication.

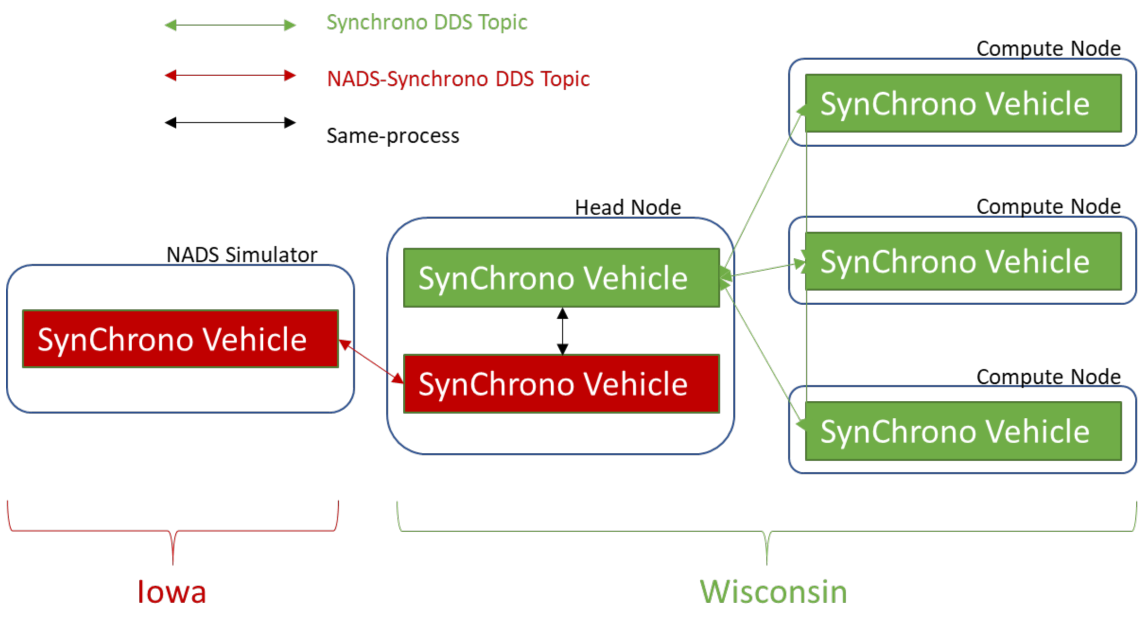

Figure 5 – DDS Layers Layout

6.2 Custom DDS messages

The messages exchanged between NADS and SynChrono are defined in an .idl file called *VehicleSet*. The data exchanged is a set of *Vehicle* messages (plus a time stamp), each one of those composed of a *body* and *tires* poses, which specify the state of the chassis and of the vehicle's wheels. The *Pose* message collects the position and orientation of a body and their derivatives (speed and acceleration).

Table 1 – NADS and SynChrono Message Format

Pose		
	Vec3	position
	Quat	rotation
	Vec3	position_dt
	Quat	rotation_dt
	Vec3	position_dtdt
	Quat	rotation_dtdt
Vehicle		
	u_long	id
	u_long	type
	Pose	body
	Pose<18>	Tires
	U_long	Light_state
Clock		
	u_long	secondsSinceEpoch
	u_long	nanoseconds
VehicleSet		
	Clock	sample_time
	<Vehicle,1024> data	data

6.3 Dead Reckoning

Not enforcing synchronism might lead to time discrepancies between the two simulations running on a geographically distributed system. It might happen that the information coming from the Iowa belongs to a previous time step in the SynChrono simulation. In this case, why apply Dead Reckoning. In other words, we use the clock information of the message to estimate the amount of delay and step the position and orientations forward using the velocities info of the Pose messages, as in an explicit Euler integration scheme.

7 Results

7.1 NADS-Euler demo

In our setup, a vehicle simulated in Iowa at NADS and driven by a human driver interacts with three vehicles simulated on Euler in Madison WI. The three SynChrono vehicles follow a prescribed path but are fully dynamic. In Figure 6, we see the vehicle simulated in Iowa, driven by Chris Schwartz, in the process of overtaking the first of the three vehicles. Below, in Figure 7, we see a vehicle simulated in Iowa and two SynChrono vehicles simulated on Euler in a scene as seen by Chrono (and as a camera sensor simulation running in SynChrono would see it). SynChrono simulation runs on a headless server. To visually reconstruct the simulation, we make use of a Chrono postprocessing rendering tool based on Blender. The output from this tool can be seen in Figure 7.

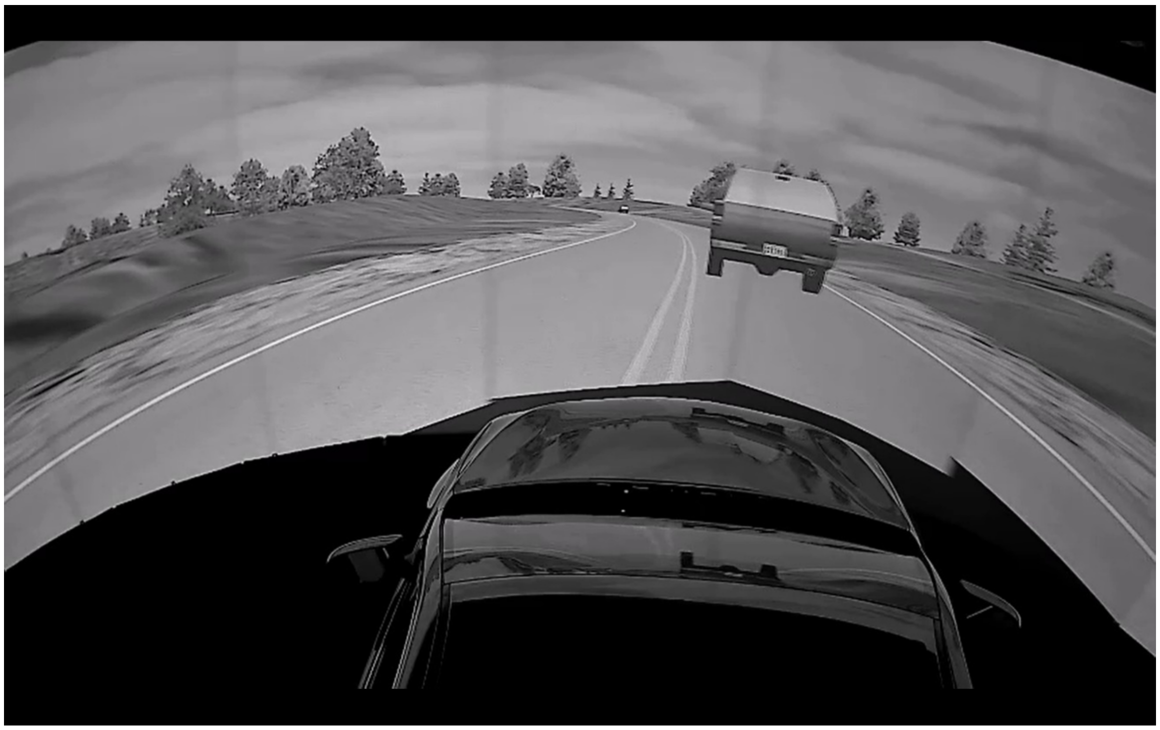

Figure 6 – NADS Vehicle overtaking maneuver

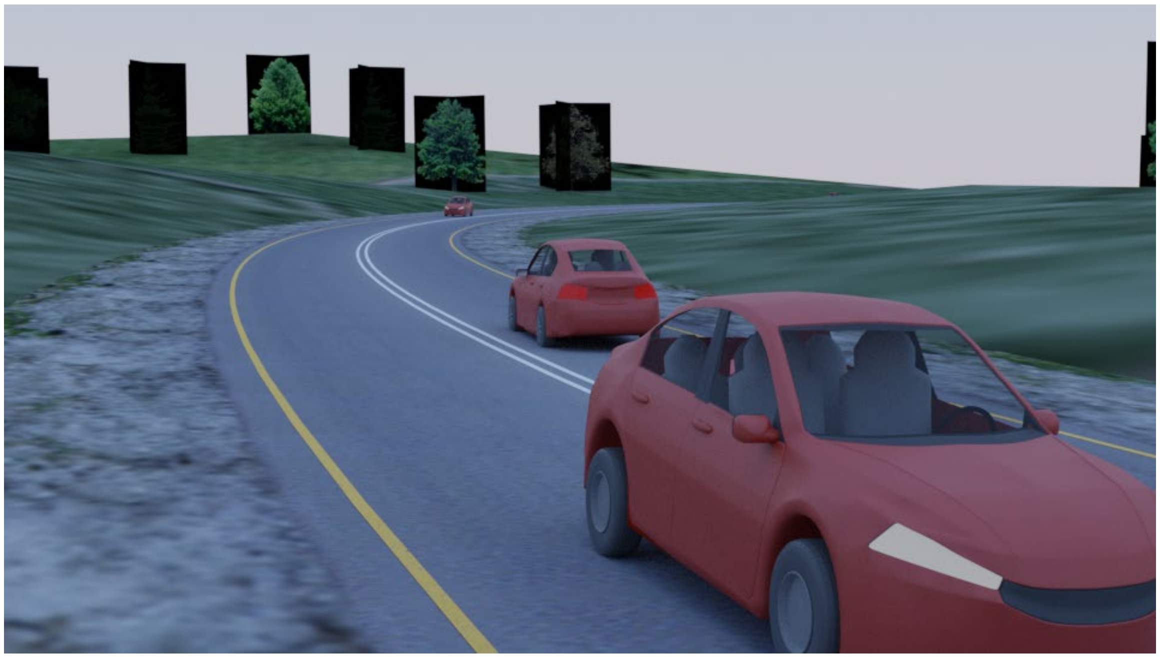

Figure 7 – Two SynChrono vehicles and one NADS vehicle, heading in different directions in a joint scene

8 Future Developments

8.1 Autonomy Stack

In this project, the SynChrono vehicles were forced to follow a prescribed path. In the future, we want to integrate Project Chrono with an industry-standard autonomy stack [5], in order to use realistic mixed-traffic scenarios where vehicle behavior is either governed by a human driver, or governed by a full, industry-standard autonomy stack.

8.2 Multi-node SynChrono DDS

Currently, to achieve real-time performance, we limit ourselves to distributing the simulation on a limited number (2 or 3) of compute-nodes. Due to communication protocol latencies, SynChrono DDS does not yet scale well above this number. This is an area of future work, as additional vehicles are required in a scenario. When a full autonomy stack is used alongside dynamics and sensor simulation, distributing the simulation to many nodes will become a requirement since each agent will require much of the resources of a compute node.

References

1. Gropp, W *MPI (Message Passing Interface)*. In: Padua D. (eds) Encyclopedia of Parallel Computing. Springer, Boston, MA.
2. eProsima FastDDS DDS (eprosima.com)
3. FlatBuffer FlatBuffers: FlatBuffers (google.github.io)
4. G. Pardo-Castellote *OMG Data-Distribution Service: architectural overview*. 23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings., 2003, pp. 200-206.

5. S. Kato et al. *Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems*. 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), 2018, pp. 287-296.
6. A. Tasora et al. *Chrono: An Open Source Multi-physics Dynamics Engine*. International Conference on High Performance Computing in Science and Engineering
7. J. Taves et al. *Chrono: SynChrono: A Scalable, Physics-Based Simulation Platform For Testing Groups of Autonomous Vehicles and/or Robots*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 2251-2256