A Digital Twin prototype for traffic sign recognition of a learning-enabled autonomous vehicle

Mohamed AbdElSalam[1], Loai Ali[1,*], Saddek Bensalem[2], Weicheng He[2], Panagiotis Katsaros[3], Nikolaos Kekatos[3], Doron Peled[4], Anastasios Temperekidis[3] and Changshun Wu[2]

[1]*Siemens EDA, Cairo, Egypt*
[2]*Université Grenoble Alpes, VERIMAG, Grenoble, France*
[3]*Aristotle University of Thessaloniki, Thessaloniki, Greece*
[4]*Bar Ilan University, Ramat Gan, Israel*

Abstract

In this paper, we present a novel digital twin prototype for a learning-enabled self-driving vehicle. The primary objective of this digital twin is to perform traffic sign recognition and lane keeping. The digital twin architecture relies on co-simulation and uses the Functional Mock-up Interface and SystemC Transaction Level Modeling standards. The digital twin consists of four clients, i) a vehicle model that is designed in Amesim tool, ii) an environment model developed in Prescan, iii) a lane-keeping controller designed in Robot Operating System, and iv) a perception and speed control module developed in the formal modeling language of BIP (Behavior, Interaction, Priority). These clients interface with the digital twin platform, PAVE360-Veloce System Interconnect (PAVE360-VSI). PAVE360-VSI acts as the co-simulation orchestrator and is responsible for synchronization, interconnection, and data exchange through a server. The server establishes connections among the different clients and also ensures adherence to the Ethernet protocol. We conclude with illustrative digital twin simulations and recommendations for future work.

Keywords

digital twin, co-simulation, FMI, SystemC, lane keeping, perception, YOLOX, autonomous vehicle

1. Introduction

Digital Twins (DTs) [1] have gained significant attention in both academia and industry. Typically defined as a virtual representation of a physical asset, process, and system, a DT serves diverse purposes [2]. DT applications have been found in various domains, like product lifecycle management and manufacturing [3]. The main objective of a DT is to generate virtual/digital models of physical objects to simulate their behaviors [2]. Conventionally, DTs involve three elements: i) a physical entity, i.e. an artifact like a vehicle, a product, a process, or a system in real space, ii) a virtual entity, i.e. a virtual representation of the physical entity, and iii) a channel that links these two entities and can transfer information from one to the other.

Companion Proceedings of the 16th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling and the 13th Enterprise Design and Engineering Working Conference, November 28 – December 1, 2023, Vienna, Austria

*Corresponding author.

mohamed.abdelsalam@siemens.com (M. AbdElSalam); loai.ali@siemens.com (L. Ali); saddek.bensalem@univ-grenoble-alpes.fr (S. Bensalem); weicheng.he@univ-grenoble-alpes.fr (W. He); katsaros@csd.auth.gr (P. Katsaros); nkekatos@csd.auth.gr (N. Kekatos); doron.peled@gmail.com (D. Peled); anastemp@csd.auth.gr (A. Temperekidis); changshun.wu@univ-grenoble-alpes.fr (C. Wu)

The digital twin concept enables creating and refining a virtual counterpart before the physical one exists [4]. If the digital system meets specific requirements, the physical product can be manufactured and linked to its DT using sensors, often referred to as a digital twin prototype (DTP) [5]. The DTP includes all the necessary models and processes for realizing the physical entity [6].

Digital twins are increasingly applied in the automotive domain [7, 8, 9, 10], with ongoing development in new DT technologies and frameworks [11, 12, 13, 14, 15]. In this work, we present our design of a digital twin of a learning-enabled autonomous vehicle. The motivation for this DT stems from the FOCETA project, with its high-level schematic and intended functionality outlined in [16]. Our specific contributions in this paper involve:

- developing a complete digital twin prototype of a learning-enabled autonomous vehicle that seamlessly integrates and interconnects multiple components and subsystems. We leverage the PAVE360-VSI digital twin platform and Ethernet protocol.
- designing a 15 Degrees of Freedom (15DoF) vehicle model in Amesim. The model is exported as a Functional Mock-up Unit (FMU) for interoperability purposes.
- generating an environment model in Prescan that specifies the road path, the sensors, and the driving scenario.
- creating a perception module, based on the YOLOX algorithm, which can detect and classify traffic signs.
- implementing a steering controller in the Robot Operating System (ROS) for lane keeping.

The rest of the paper is structured as follows. In Section 2, we introduce the key technologies, standards, and tools employed in the co-simulation and digital twin architecture. In Section 3, we present all the DT components and building blocks designed to execute DT simulations. The paper concludes in Section 4.

2. Key enabling technologies for Digital Twins

Co-simulation. Co-simulation is a practical technique for simulating heterogeneous systems. It permits the modeling and simulation of different components of a system using various tools and methods. The global simulation of a coupled system can then be achieved by composing the simulations of its parts. Co-simulation also allows the joint simulation of loosely coupled stand-alone sub-simulators. To guarantee that the submodels and sub-simulators can work seamlessly together, standardized interfaces are commonly used [17, 18, 19].

Functional Mock-up Interface (FMI). A widely used standard for co-simulation is the FMI standard [20, 21]. The FMI standard defines an Application Programming Interface (API) and the model elements referred to as Functional Mock-up Units (FMUs), which adhere to this API. Each FMU can be seen as a black box that implements the methods defined in the FMI API. To run a group of interconnected FMUs, an orchestrator, also called a master algorithm or FMI Master, is required. Its purpose is to manage and synchronize their execution.

Transaction Level Modeling (TLM). The SystemC-TLM 2.0 standard simplifies communication and computation in industrial simulation models by using channels. These channels abstract unnecessary details, resulting in faster simulations and facilitating high-level design validation. Transactions are initiated through the functions in the channel model interface.

The primary focus of the TLM transaction API is the versatile *generic payload* transaction, which can be adapted to various application domains [22]. TLM is suitable for a wide range of domains, including digital and analog simulations, as well as digital communication protocols like CAN, Ethernet, AXI, and PCIe.

Behavior, Interaction, Priority (BIP). BIP is a modeling framework for rigorous system design [23]. In BIP, complex systems are constructed by coordinating the behavior of atomic components. BIP *atomic components* are transition systems with ports and variables. An atomic component may contain control locations, variables, communication ports, and transitions. Composite components are built by composing multiple atomic components. Interactions between components are defined by connectors, which specify sets of interactions.

During the execution of a BIP interaction, all components that participate in the interaction, i.e., have an associated port that is part of the interaction, must execute their corresponding transitions simultaneously. All components that do not participate in the interaction, do not execute any transition and thus remain in the same control location.

Digital Twin platform. PAVE360-Veloce System Interconnect (PAVE360-VSI) [24] is a digital twin platform with network interconnect and hardware emulation capabilities. It can be used for verifying hardware and software control systems of autonomous systems. It offers cyber-physical ports that support diverse client connections and allows both protocol-agnostic and protocol-aware connections between mixed-fidelity models. These connections can be shared for Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), and Hardware-in-the-Loop (HiL) verification. Digital Twins (DTs) are particularly valuable for pre-silicon verification [25].

PAVE360-VSI provides a co-simulation architecture for DTs, which complies with Functional Mock-up Interface (FMI) and Transaction-Level Modeling (TLM) standards. Simulations advance in discrete time steps, leveraging a server/client structure within the PAVE360-VSI architecture. This design ensures interoperability among DT components, synchronizes network communication and facilitates data transfer using multiple protocols.

The basic assumption is that every external simulator and foreign model can be integrated with a third-party API customized for the application's needs. This API is then linked to a TLM fabric portal interface known as the "Gateway" [26]. The Gateway serves as an entry point to the digital twin backplane, facilitating transactional communication and ensuring coordinated time advancement.

Each external simulator or foreign model is a client process connected to the shared DT backplane. This backplane acts as the timekeeper, overseeing all time advancement operations to maintain synchronization among client processes and ensure deterministic behavior.

PAVE360-VSI has been extended to facilitate formal modeling. A "BIP Gateway" has been created as part of the architecture in [27]. Runtime verification can be used within the architecture via FMI [28].

3. Developing a Digital Twin for traffic sign recognition and lane keeping

The digital twin comprises four heterogeneous components that communicate through the PAVE360-VSI platform. Figure 1 presents the overall DT prototype of our learning-enabled vehicle with lane-keeping and traffic sign recognition capabilities. Each component has a different role, i.e., vehicle modeling, environment modeling, lane keeping, traffic sign recognition and speed regulation, each detailed in separate sections.

Data exchange among these components occurs via the Ethernet protocol. Gateways play a crucial role in converting data into simulated Ethernet frames. These exchanged frames adhere to the structure of a typical (physical) Ethernet network connection, incorporating familiar fields such as MAC headers, payloads, and checksums. The payload of the exchanged Ethernet frames consists of a serialized bitstream representing the data to be exchanged between the communicating entities. This payload is designed to accommodate various data types, such as floats, integers, strings, and other relevant data structures. This flexibility enables the digital twin to exchange a diverse range of information, facilitating a comprehensive simulation of the ego vehicle's behavior and its interactions with the environment.

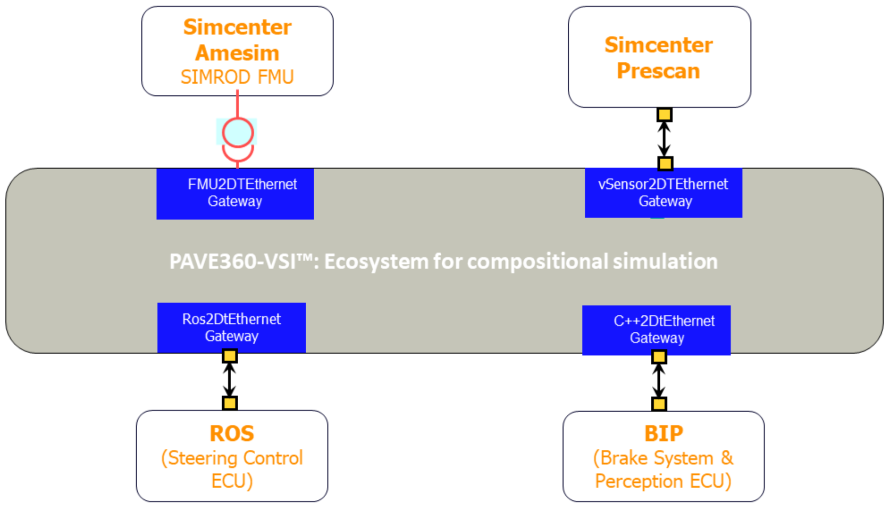


Figure 1: Digital twin prototype for traffic sign recognition of a learning-enabled vehicle. Architecture: PAVE360-VSI DT platform containing a server and four clients/components; a vehicle model modeled in AMESIM, a control ROS subsystem, a control module modeled in BIP, and a Prescan model. All tools are co-simulated and interconnected via gateways.

3.1. Vehicle model: Amesim

The vehicle is described by a high-fidelity dynamical model (with 15 degrees of freedom) in Amesim. It is adapted from the Simrod SIEMENS model [29]. It is encapsulated as an FMU to connect with the interconnect and we use an FMU to ethernet gateway.

The FMU to ethernet gateway has a dual role and it supports the bi-directional transfer

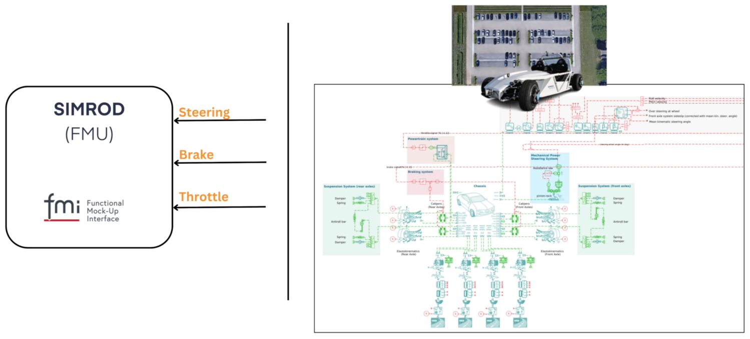


Figure 2: Vehicle Model in Amesim; the left part shows the inputs and outputs of the FMU client, while the right part displays a visual representation of the Simrod Amesim FMU.

of data from the FMU to the interconnect and vice versa. In this case, its primary task is to receive an ethernet frame that contains information about the ego vehicle's coordinates, brake, and throttle commands. The gateway decapsulates this information from the ethernet frame (keeping only the payload) and feeds it into the FMU model. The payload consists of three float variables representing the ego vehicle's coordinates, brake, and throttle commands. This model maintains a representation of the ego-vehicle dynamics. Within the FMU, we execute a step, advancing the simulation by a fixed time step. Figure 2 provides some details of the Amesim model and consists of two parts. The left part shows a visual representation at a top level for the FMU client representing its inputs/outputs, while the right part shows the Simrod Amesim FMU.

3.2. Steering control: ROS

The Robot Operating System (ROS) client plays a key role in the navigation of the ego vehicle. It receives real-time coordinates of the ego vehicle's position, providing a continuous update of its location within the Prescan environment. The ROS gateway works as follows. It receives an ethernet frame that encapsulates the current coordinates of the ego-vehicle. The payload of the received ethernet frame is a stream of bytes for three float variables representing the ego vehicle's coordinates, the ego vehicle's speed, and acceleration. The ROS gateway then decodes the data from the frame and sends this information to other ROS nodes. These nodes are designed with a specific algorithm for steering the car. They process the positional data and convert it into actionable steering commands. These steering commands are subsequently published by the ROS nodes. The ROS gateway receives these commands, encapsulates them within an Ethernet frame, and transmits this information. The payload of the transmitted Ethernet frame contains a float variable representing the new steering value of the ego vehicle. This ensures that the ego vehicle navigates smoothly and accurately through the virtual world, responding appropriately to the changing conditions and scenarios.

Figure 3 shows how ROS subsystem is structured and containerized in Docker. The left part shows a visual representation at a top level for the ROS client representing its inputs/outputs, while the right part shows the ROS subsystem.

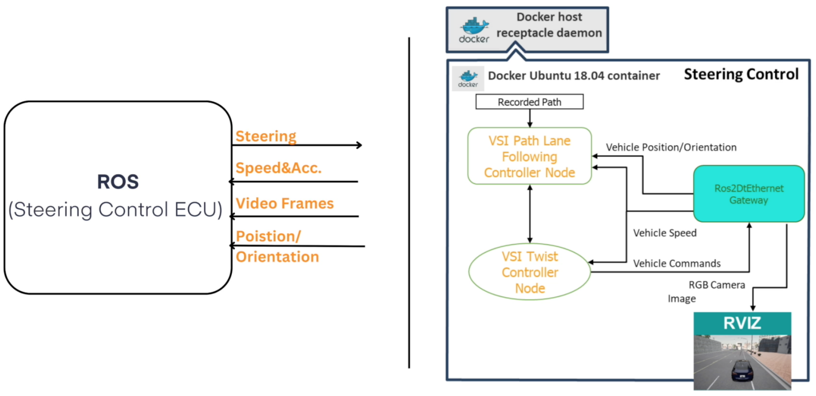


Figure 3: Digital Twin component - ROS client - steering control module.

3.3. Environment model: Prescan

The Prescan client serves as a digital representation of a real-world environment, designed to mirror actual driving conditions. This virtual world includes roads, traffic signs, and other elements typically encountered during a drive.

The main actor of this Prescan scenario is the ego vehicle, a digital representation of a real-world vehicle. This vehicle is equipped with an advanced RGB camera sensor, which captures video frames of the surrounding environment. These frames provide visual data that is disseminated to various clients within the network. This continuous stream of information forms the basis for decision-making processes in other clients, enabling them to react and adapt in real time to the changing conditions within the Prescan environment.

The vSensor (virtual sensor) to Ethernet gateway, implemented as an S-function within the Prescan client, plays a key role in this process. It encapsulates the video frame captured by the RGB camera sensor mounted on top of the ego-vehicle inside an Ethernet frame and transmits it. The video frames that are sent in the Ethernet frame are represented by a stream of bytes representing the RGB values of the pixels of the image that is captured by the RGB camera sensor. In addition, the gateway sends another Ethernet frame containing the current coordinates and speed of the ego-vehicle, conveyed as float variables in the payload. Finally, the gateway receives two Ethernet frames containing the steering angle, brake, and throttle values.

In Figure 4, we show the Prescan scenario under study. The left part shows a visual representation at a top level for the prescan client representing its inputs/outputs, while the right part shows the Prescan experiment setup and also the ego-vehicle used.

3.4. Perception and speed control module: BIP

The BIP client, another DT component, serves two essential functions for the ego vehicle. Firstly, it employs a YOLOX deep learning model [30] to process video frames received from the ego vehicle's RGB camera sensor. These frames, transmitted as an Ethernet frame from the Prescan client, pass through the C++ to Ethernet gateway within the BIP client. The video frame within the Ethernet frame is represented as a byte sequence capturing RGB values of the pixels. The

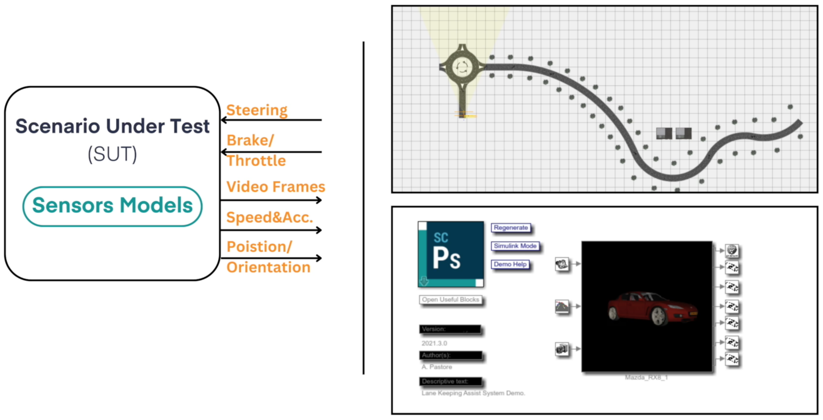


Figure 4: Digital Twin component - Prescan scenario - environment model and driving track.

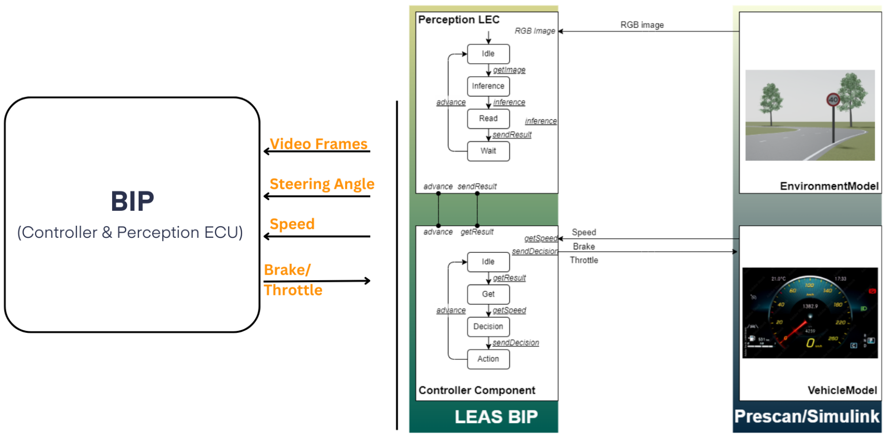


Figure 5: Digital Twin component - BIP module - speed control regulation and perception with YOLOX.

gateway decapsulates this data and forwards it to YOLOX. The YOLOX model has been trained to detect speed limits and identify Prescan traffic signs.

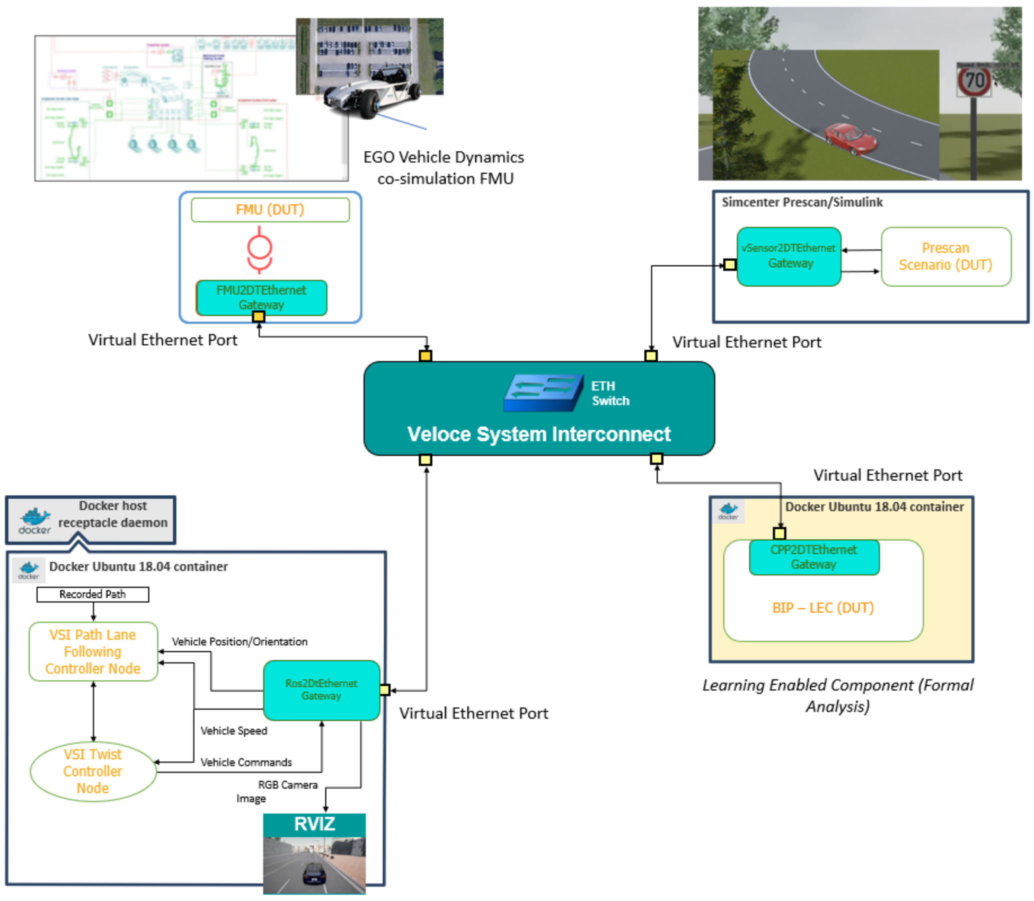
The second function of the BIP client is to regulate the velocity of the ego vehicle based on the detected speed limit and the current speed of the ego vehicle. The current speed of the ego vehicle is a float variable received via the C++ to Ethernet gateway. A Proportional-Integral (PI) controller then uses both pieces of information to apply either brake or throttle commands based on the difference between the detected speed limit and the vehicle's current speed.

Figure 5 provides a visual representation of the key functionalities of the BIP client. The left part illustrates the client's inputs/outputs, while the right part outlines the internal interactions within the BIP client.

3.5. Orchestration: PAVE360-VSI platform

The PAVE360-VSI platform serves as the cornerstone for co-simulation and digital twin orchestration, offering functions that seamlessly integrate all clients.

In this work, to realize the digital twin prototype, we employed a simulated Ethernet switch within the PAVE360-VSI digital twin platform for data exchange. All four clients connect to this switch through gateways, responsible for encapsulating and decapsulating data into/from Ethernet frames. The orchestrator's switch then takes control, routing data to the appropriate client (see Figure 6). The orchestrator is also responsible for synchronizing all clients based on a given message sequence diagram. It ensures that all components operate on the same simulated time, maintaining safe and stable operation and coordination within the system. Figure 7 depicts the message state sequence we designed and implemented.



EGO Vehicle Dynamics co-simulation FMU
FMU (DUT)
FMU2DTEthernet Gateway
Virtual Ethernet Port
Simcenter Prescan/Simulink
vSensor2DTEthernet Gateway
Prescan Scenario (DUT)
70
Virtual Ethernet Port
ETH Switch
Veloce System Interconnect
Virtual Ethernet Port
Docker Ubuntu 18.04 container
CPP2DTEthernet Gateway
BIP – LEC (DUT)
Learning Enabled Component (Formal Analysis)
Docker host receptacle daemon
Docker Ubuntu 18.04 container
Recorded Path
VSI Path Lane Following Controller Node
Vehicle Position/Orientation
Ros2DtEthernet Gateway
Vehicle Speed
VSI Twist Controller Node
Vehicle Commands
RGB Camera Image
RVIZ
Virtual Ethernet Port

Figure 6: Architecture of the digital twin showing all clients' connections.

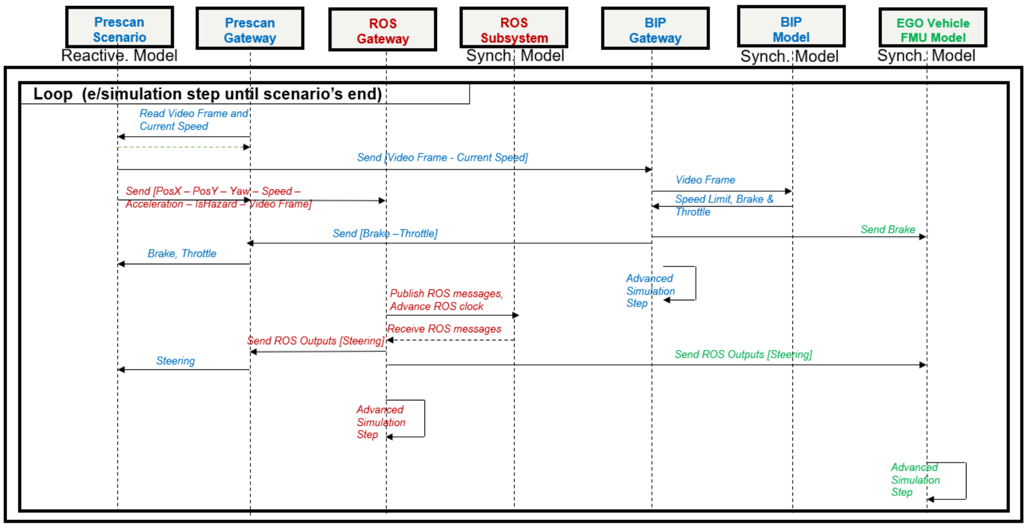
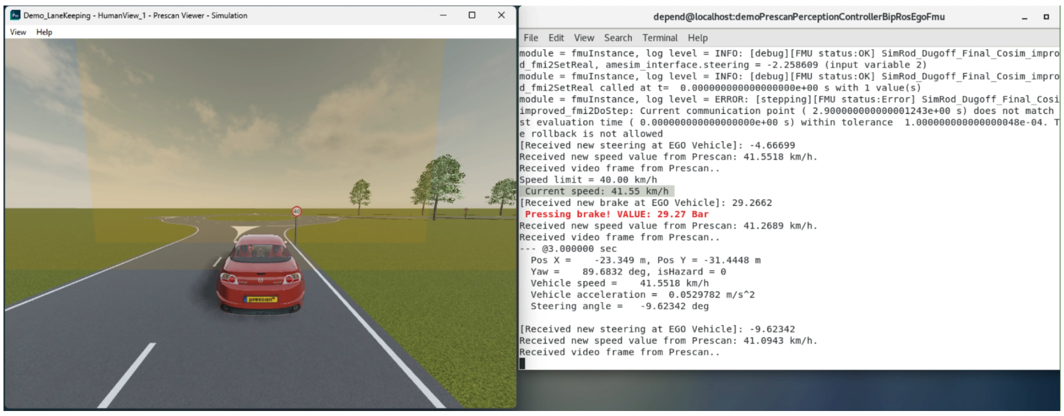

Prescan Scenario
Reactive. Model
Prescan Gateway
ROS Gateway
ROS Subsystem
Synch. Model
BIP Gateway
BIP Model
Synch. Model
EGO Vehicle FMU Model
Synch. Model
Loop (e/simulation step until scenario's end)
Read Video Frame and Current Speed
Send [Video Frame - Current Speed]
Video Frame
Send [PosX – PosY – Yaw – Speed – Acceleration – IsHazard – Video Frame]
Speed Limit, Brake & Throttle
Send [Brake –Throttle]
Send Brake
Brake, Throttle
Advanced Simulation Step
Publish ROS messages, Advance ROS clock
Receive ROS messages
Send ROS Outputs [Steering]
Steering
Send ROS Outputs [Steering]
Advanced Simulation Step
Advanced Simulation Step

Figure 7: Message sequence diagram of the digital twin showing the order of the component calls.


Demo_LaneKeeping - HumanView_1 - Prescan Viewer - Simulation
View Help
depend@localhost:demoPrescanPerceptionControllerBipRosEgoFmu
File Edit View Search Terminal Help

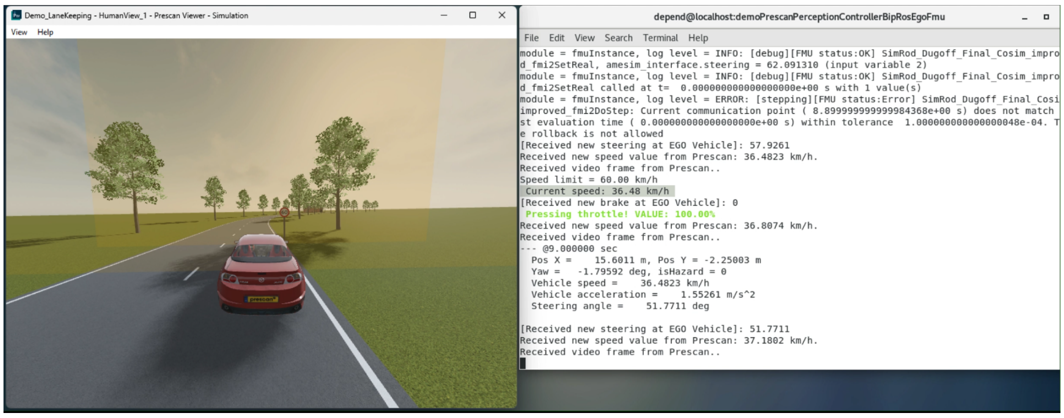
```
module = fmuInstance, log level = INFO: [debug][FMU status:OK] SimRod_Dugoff_Final_Cosim_impro
d_fmi2SetReal, amesim_interface.steering = -2.258609 (input variable 2)
module = fmuInstance, log level = INFO: [debug][FMU status:OK] SimRod_Dugoff_Final_Cosim_impro
d_fmi2SetReal called at t=  0.00000000000000000e+00 s with 1 value(s)
module = fmuInstance, log level = ERROR: [stepping][FMU status:Error] SimRod_Dugoff_Final_Cosi
improved_fmi2DoStep: Current communication point ( 2.90000000000001243e+00 s) does not match
st evaluation time ( 0.00000000000000000e+00 s) within tolerance  1.00000000000000048e-04. T
e rollback is not allowed
[Received new steering at EGO Vehicle]: -4.66699
Received new speed value from Prescan: 41.5518 km/h.
Received video frame from Prescan..
Speed limit = 40.00 km/h
 Current speed: 41.55 km/h
[Received new brake at EGO Vehicle]: 29.2662
 Pressing brake! VALUE: 29.27 Bar
Received new speed value from Prescan: 41.2689 km/h.
Received video frame from Prescan..
--- @3.000000 sec
  Pos X =    -23.349 m, Pos Y = -31.4448 m
  Yaw =    89.6832 deg, isHazard = 0
  Vehicle speed =    41.5518 km/h
  Vehicle acceleration =  0.0529782 m/s^2
  Steering angle =   -9.62342 deg

[Received new steering at EGO Vehicle]: -9.62342
Received new speed value from Prescan: 41.0943 km/h.
Received video frame from Prescan..
```

Figure 8: Digital twin simulation of a learning-enabled vehicle showcasing a successful detection of a *40 km/h* speed limit sign.

3.6. Digital Twin simulation

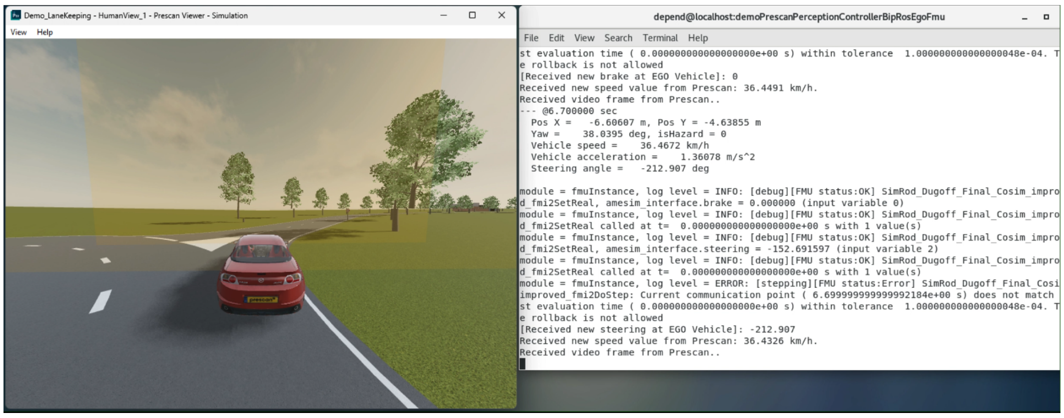
In this section, we conduct simulations of the digital twin, highlighting key moments through screenshots captured from a 3D-simulated Prescan video. Figure 8 shows the successful detection of a traffic sign and the specific $40km/h$ speed limit. Figure 9 shows the successful detection of another traffic sign, this time of and a $60km/h$ speed limit. Figure 10 captures a successful maneuver on a road with a sharp curvature.


Demo_LaneKeeping - HumanView_1 - Prescan Viewer - Simulation
View
Help
depend@localhost:demoPrescanPerceptionControllerBipRosEgoFmu
File
Edit
View
Search
Terminal
Help

```
module = fmuInstance, log level = INFO: [debug][FMU status:OK] SimRod_Dugoff_Final_Cosim_impro
d_fmi2SetReal, amesim_interface.steering = 62.091310 (input variable 2)
module = fmuInstance, log level = INFO: [debug][FMU status:OK] SimRod_Dugoff_Final_Cosim_impro
d_fmi2SetReal called at t=  0.000000000000000000e+00 s with 1 value(s)
module = fmuInstance, log level = ERROR: [stepping][FMU status:Error] SimRod_Dugoff_Final_Cosi
improved_fmi2DoStep: Current communication point ( 8.899999999999984368e+00 s) does not match
st evaluation time ( 0.000000000000000000e+00 s) within tolerance  1.000000000000000048e-04. T
e rollback is not allowed
[Received new steering at EGO Vehicle]: 57.9261
Received new speed value from Prescan: 36.4823 km/h.
Received video frame from Prescan..
Speed limit = 60.00 km/h
 Current speed: 36.48 km/h
[Received new brake at EGO Vehicle]: 0
 Pressing throttle! VALUE: 100.00%
Received new speed value from Prescan: 36.8074 km/h.
Received video frame from Prescan..
--- @9.000000 sec
  Pos X =    15.6011 m, Pos Y = -2.25003 m
  Yaw =   -1.79592 deg, isHazard = 0
  Vehicle speed =    36.4823 km/h
  Vehicle acceleration =    1.55261 m/s^2
  Steering angle =    51.7711 deg

[Received new steering at EGO Vehicle]: 51.7711
Received new speed value from Prescan: 37.1802 km/h.
Received video frame from Prescan..
```

Figure 9: Digital twin simulation of a learning-enabled vehicle showcasing a successful detection of a *60 km/h* speed limit sign.


Demo_LaneKeeping - HumanView_1 - Prescan Viewer - Simulation
View
Help
depend@localhost:demoPrescanPerceptionControllerBipRosEgoFmu
File
Edit
View
Search
Terminal
Help

```
st evaluation time ( 0.000000000000000000e+00 s) within tolerance  1.000000000000000048e-04. T
e rollback is not allowed
[Received new brake at EGO Vehicle]: 0
Received new speed value from Prescan: 36.4491 km/h.
Received video frame from Prescan..
--- @6.700000 sec
  Pos X =   -6.60607 m, Pos Y = -4.63855 m
  Yaw =    38.0395 deg, isHazard = 0
  Vehicle speed =    36.4672 km/h
  Vehicle acceleration =    1.36078 m/s^2
  Steering angle =   -212.907 deg

module = fmuInstance, log level = INFO: [debug][FMU status:OK] SimRod_Dugoff_Final_Cosim_impro
d_fmi2SetReal, amesim_interface.brake = 0.000000 (input variable 0)
module = fmuInstance, log level = INFO: [debug][FMU status:OK] SimRod_Dugoff_Final_Cosim_impro
d_fmi2SetReal called at t=  0.000000000000000000e+00 s with 1 value(s)
module = fmuInstance, log level = INFO: [debug][FMU status:OK] SimRod_Dugoff_Final_Cosim_impro
d_fmi2SetReal, amesim_interface.steering = -152.691597 (input variable 2)
module = fmuInstance, log level = INFO: [debug][FMU status:OK] SimRod_Dugoff_Final_Cosim_impro
d_fmi2SetReal called at t=  0.000000000000000000e+00 s with 1 value(s)
module = fmuInstance, log level = ERROR: [stepping][FMU status:Error] SimRod_Dugoff_Final_Cosi
improved_fmi2DoStep: Current communication point ( 6.699999999999992184e+00 s) does not match
st evaluation time ( 0.000000000000000000e+00 s) within tolerance  1.000000000000000048e-04. T
e rollback is not allowed
[Received new steering at EGO Vehicle]: -212.907
Received new speed value from Prescan: 36.4326 km/h.
Received video frame from Prescan..
```

Figure 10: Digital twin simulation of a learning-enabled vehicle showcasing successful lane keeping on a curvy road.

4. Conclusion

In this paper, we introduced a digital twin prototype of a learning-enabled self-driving vehicle. The prototype comprises four clients, each representing subsystems/models, collectively contributing to the simulation of real-world driving scenarios. The Amesim client describes the ego vehicle model, the Prescan client constructs a virtual environment, the ROS client guides the ego vehicle through this environment, and the BIP client manages both the perception module and the vehicle's speed control. The digital twin orchestrator ties all these components together, ensuring efficient data routing and synchronization. We opt for the PAVE360-VSI digital twin platform as it supports both FMI and SystemC-TLM open standards. Alternative platforms, such as VICO [31], MAESTRO [32], MAESTRO2 [33], DESYRE [34], DIGITBrain [35], and HUBCAP [36], also offer options for FMI-based co-simulation or SystemC co-simulation of digital twins.

This DT prototype system highlights the potential of digital twins of learning-enabled autonomous vehicles, demonstrating their ability to replicate and analyze complex real-world scenarios. A promising future direction involves the seamless integration of learning-enabled components in the DTs without imposing large computational burdens. Techniques such as neural network abstraction and compression could offer viable solutions, as discussed in works like[37, 38]. Another research direction entails the efficient integration of formal verification methods to facilitate reasoning about the conditions under which the DTs satisfy given safety, security, or performance specifications. Depending on the DT task and application, exhaustive methods [39, 40, 41, 42, 43] or non-exhaustive but lightweight methods [28, 27, 44, 45] can be considered.

References

[1] M. Grieves, Origins of the digital twin concept, Florida Institute of Technology 8 (2016).

[2] M. W. Grieves, Virtually intelligent product systems: Digital and physical twins, Complex Systems Engineering: Theory and Practice (2019). URL: https://api.semanticscholar.org/CorpusID:202478997.

[3] M. Grieves, Digital Twin: Manufacturing Excellence through Virtual Factory Replication-A Whitepaper by Dr. Michael Grieves, White Paper (2015) 1–7.

[4] M. Grieves, J. Vickers, Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems, Transdisciplinary perspectives on complex systems: New findings and approaches (2017) 85–113.

[5] M. W. Grieves, Digital twins: Past, present, and future, in: The Digital Twin, Springer, 2023, pp. 97–121.

[6] D. Jones, C. Snider, A. Nassehi, J. Yon, B. Hicks, Characterising the digital twin: A systematic literature review, CIRP Journal of Manufacturing Science and Technology 29 (2020) 36–52.

[7] J. P. Allamaa, P. Patrinos, H. Van der Auweraer, T. D. Son, Sim2real for autonomous vehicle control using executable digital twin, IFAC-PapersOnLine 55 (2022) 385–391.

[8] D. Piromalis, A. Kantaros, Digital twins in the automotive industry: The road toward physical-digital convergence, Applied System Innovation 5 (2022) 65.

[9] H. Esen, B. H.-C. Liao, Simulation-based safety assurance for an avp system incorporating learning-enabled components, arXiv preprint arXiv:2311.03362 (2023).

[10] M. F. Tøttrup, E. C. Hu, B. A. Kramer, H. D. Macedo, L. Esterle, Using INTO-CPS Tools in the Development of a Digital Twin for the F1TENTH Race Car, in: International Conference on Software Engineering and Formal Methods, Springer, 2022, pp. 200–209.

[11] A. Chaudhuri, G. Pash, D. A. Hormuth, G. Lorenzo, M. Kapteyn, C. Wu, E. A. Lima, T. E. Yankeelov, K. Willcox, et al., Predictive digital twin for optimizing patient-specific radiotherapy regimens under uncertainty in high-grade gliomas, Frontiers in Artificial Intelligence 6 (2023).

[12] D. Hartmann, H. Van der Auweraer, The executable digital twin: merging the digital and the physics worlds, arXiv preprint arXiv:2210.17402 (2022).

[13] M. Torzoni, M. Tezzele, S. Mariani, A. Manzoni, K. E. Willcox, A digital twin framework for

civil engineering structures, Computer Methods in Applied Mechanics and Engineering 418 (2024) 116584.

[14] M. Frasheri, H. Ejersbo, C. Thule, C. Gomes, J. L. Kvistgaard, P. G. Larsen, L. Esterle, Addressing time discrepancy between digital and physical twins, Robotics and Autonomous Systems 161 (2023) 104347.

[15] M. Deakin, M. Vanin, Z. Fan, D. Van Hertem, Smart energy network digital twins: Findings from a uk-based demonstrator project, arXiv preprint arXiv:2311.11997 (2023).

[16] S. Bensalem, P. Katsaros, D. Ničković, B. H.-C. Liao, R. R. Nolasco, M. AbdElSalam, T. Beyene, F. Cano, A. Delacourt, H. Esen, A. Forrai, W. He, X. Huang, N. Kekatos, B. Könighofer, M. Paulitsch, D. Peled, M. Ponchant, L. Sorokin, S. Tong, C. Wu, Continuous engineering for trustworthy learning-enabled autonomous systems, in: B. Steffen (Ed.), Bridging the Gap Between AI and Reality, Springer Nature Switzerland, Cham, 2024, pp. 256–278.

[17] C. Gomes, C. Thule, D. Broman, P. G. Larsen, H. Vangheluwe, Co-simulation: State of the art, arXiv preprint arXiv:1702.00686 (2017).

[18] C. Gomes, C. Thule, P. G. Larsen, J. Denil, H. Vangheluwe, Co-simulation of continuous systems: a tutorial, arXiv preprint arXiv:1809.08463 (2018).

[19] P. Talasila, C. Gomes, P. H. Mikkelsen, S. G. Arboleda, E. Kamburjan, P. G. Larsen, Digital Twin as a Service (DTaaS): A Platform for Digital Twin Developers and Users, arXiv preprint arXiv:2305.07244 (2023).

[20] T. Blockwitz, M. Otter, J. Akesson, M. Arnold, C. Clauss, H. Elmqvist, M. Friedrich, A. Junghanns, J. Mauss, D. Neumerkel, et al., Functional mockup interface 2.0: The standard for tool independent exchange of simulation models, in: Proceedings of the 9th International Modelica Conference, Munich, Germany, 2012, pp. 173–184.

[21] A. Junghanns, C. Gomes, C. Schulze, K. Schuch, R. Pierre, M. Blaesken, I. Zacharias, A. Pillekeit, K. Wernersson, T. Sommer, et al., The functional mock-up interface 3.0-new features enabling new applications, in: Modelica conferences, 2021, pp. 17–26.

[22] G. Frank, Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems, Springer (2005).

[23] A. Basu, S. Bensalem, M. Bozga, P. Bourgos, J. Sifakis, Rigorous system design: the BIP approach, in: International doctoral workshop on mathematical and engineering methods in computer science, Springer, 2011, pp. 1–19.

[24] Siemens EDA, Veloce®, Available at: https://eda.sw.siemens.com/en-US/ic/veloce/, 2023.

[25] M. AbdElSalam, K. Khalil, J. Stickley, A. Salem, B. Loye, Verification of advanced driver assistance systems and autonomous vehicles with hardware emulation-in-the-loop a case study with multiple ecus, IJAE (2019).

[26] Adam Erickson, John Stickley, UVM-Connect primer, Available at: https://verificationacademy.com/courses/uvm-connect, 2023.

[27] A. Temperekidis, N. Kekatos, P. Katsaros, W. He, S. Bensalem, H. AbdElSabour, M. AbdElSalam, A. Salem, Towards a digital twin architecture with formal analysis capabilities for learning-enabled autonomous systems, in: International Conference on Modelling and Simulation for Autonomous Systems, Springer, 2022, pp. 163–181.

[28] A. Temperekidis, N. Kekatos, P. Katsaros, Runtime verification for FMI-based co-simulation, in: International Conference on Runtime Verification, Springer, 2022, pp. 304–313.

[29] Siemens PLM Software, Simcenter Amesim, Available at: https://www.plm.automation.

siemens.com/en/products/lms/imagine-lab/amesim/, 2023.

[30] Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun, Yolox: Exceeding yolo series in 2021, arXiv preprint arXiv:2107.08430 (2021).

[31] L. I. Hatledal, Y. Chu, A. Styve, H. Zhang, Vico: An entity-component-system based co-simulation framework, Simulation Modelling Practice and Theory 108 (2021) 102243.

[32] C. Thule, K. Lausdahl, C. Gomes, G. Meisl, P. G. Larsen, Maestro: the into-cps co-simulation framework, Simulation Modelling Practice and Theory 92 (2019) 45–61.

[33] S. T. Hansen, C. Thule, C. Gomes, K. G. Lausdahl, F. P. Madsen, G. Abbiati, P. G. Larsen, Co-simulation at different levels of expertise with Maestro2, Journal of Systems and Software (2023) 111905.

[34] A. Mignogna, L. Mangeruca, B. Boyer, A. Legay, A. Arnold, Sos contract verification using statistical model checking, arXiv preprint arXiv:1311.3632 (2013).

[35] P. Talasila, D.-C. Crăciunean, P. Bogdan-Constantin, P. G. Larsen, C. Zamfirescu, A. Scovill, Comparison between the hubcap and digitbrain platforms for model-based design and evaluation of digital twins, in: International Conference on Software Engineering and Formal Methods, Springer, 2021, pp. 238–244.

[36] H. D. Macedo, C. Sassanelli, P. G. Larsen, S. Terzi, Facilitating model-based design of cyber-manufacturing systems, Procedia CIRP 104 (2021) 1936–1941.

[37] R. Mishra, H. P. Gupta, T. Dutta, A survey on deep neural network compression: Challenges, overview, and solutions, arXiv preprint arXiv:2010.03954 (2020).

[38] C. Eleftheriadis, N. Kekatos, P. Katsaros, S. Tripakis, On neural network equivalence checking using smt solvers, in: International Conference on Formal Modeling and Analysis of Timed Systems, Springer, 2022, pp. 237–257.

[39] T. Wright, C. Gomes, J. Woodcock, Formally verified self-adaptation of an incubator digital twin, in: International Symposium on Leveraging Applications of Formal Methods, Springer, 2022, pp. 89–109.

[40] L. D. Couto, S. Basagiannis, E. H. Ridouane, A. E.-D. Mady, M. Hasanagic, P. G. Larsen, Injecting formal verification in fmi-based co-simulations of cyber-physical systems, in: Software Engineering and Formal Methods, Springer, 2018, pp. 284–299.

[41] G. Frehse, N. Kekatos, D. Nickovic, J. Oehlerking, S. Schuler, A. Walsch, M. Woehrle, A toolchain for verifying safety properties of hybrid automata via pattern templates, in: 2018 Annual American Control Conference (ACC), IEEE, 2018, pp. 2384–2391.

[42] N. Kekatos, Formal verification of cyber-physical systems in the industrial model-based design process, Ph.D. thesis, Université Grenoble Alpes, 2018.

[43] F. Lisboa Malaquias, G. Giantamidis, S. Basagiannis, S. Fulvio Rollini, I. Amundson, Towards a methodology to design provably secure cyber-physical systems, ACM SIGAda Ada Letters 43 (2023) 94–99.

[44] J. Dobaj, A. Riel, T. Krug, M. Seidl, G. Macher, M. Egretzberger, Towards digital twin-enabled devops for cps providing architecture-based service adaptation & verification at runtime, in: Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2022, pp. 132–143.

[45] K. Havelund, D. Peled, D. Ulus, First-order temporal logic monitoring with bdds, Formal Methods in System Design 56 (2020) 1–21.