Co-simulation of Automotive Cyber-Physical Systems

UNIVERSITÀ DI PISA

Maurizio Palmieri

Department of Information Engineering, University of Pisa, Pisa, Italy

WSCF 2022

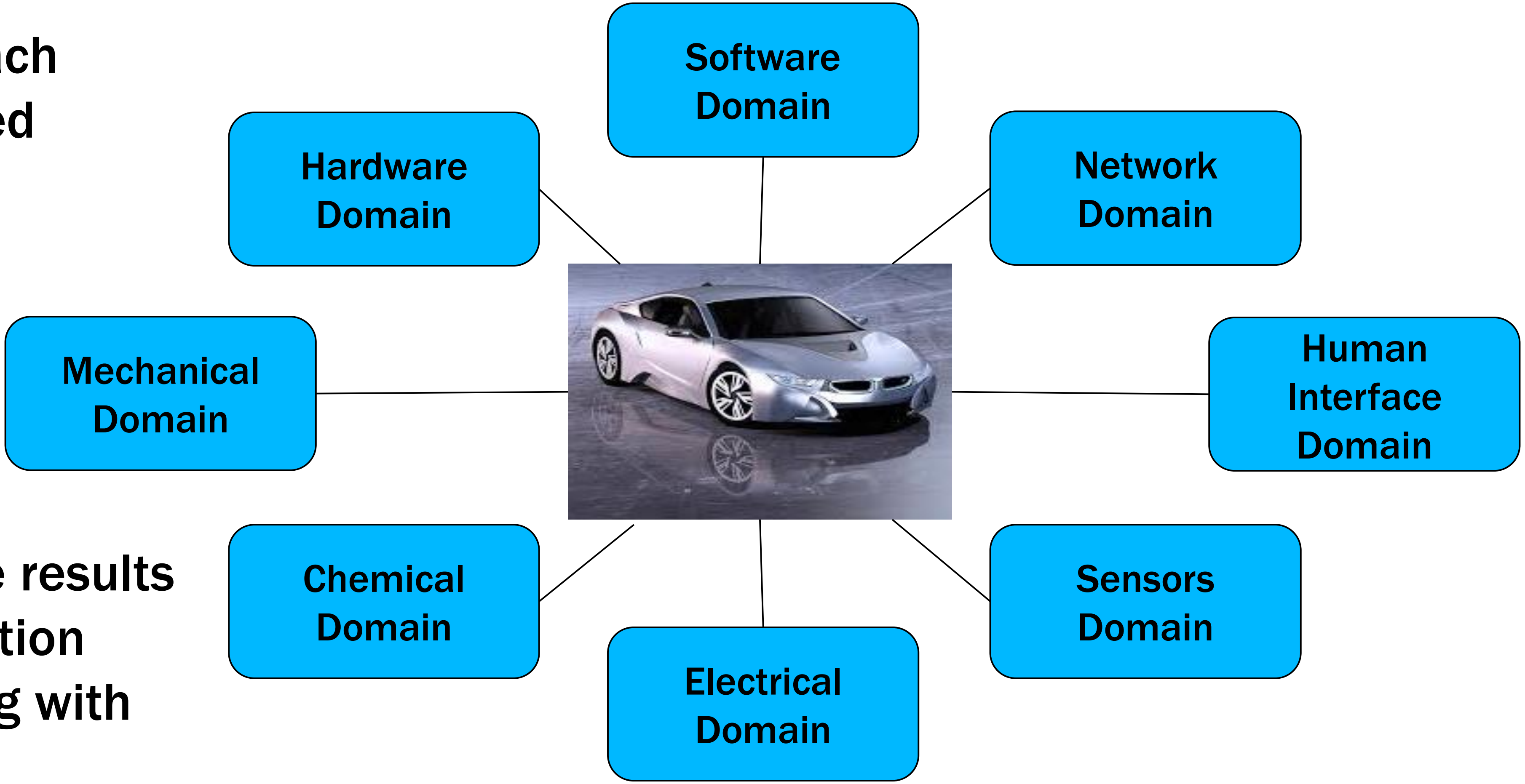
Digital Twins for Cyber Physical Systems

Montevideo, 10 November 2022

Outline

1. Introduction

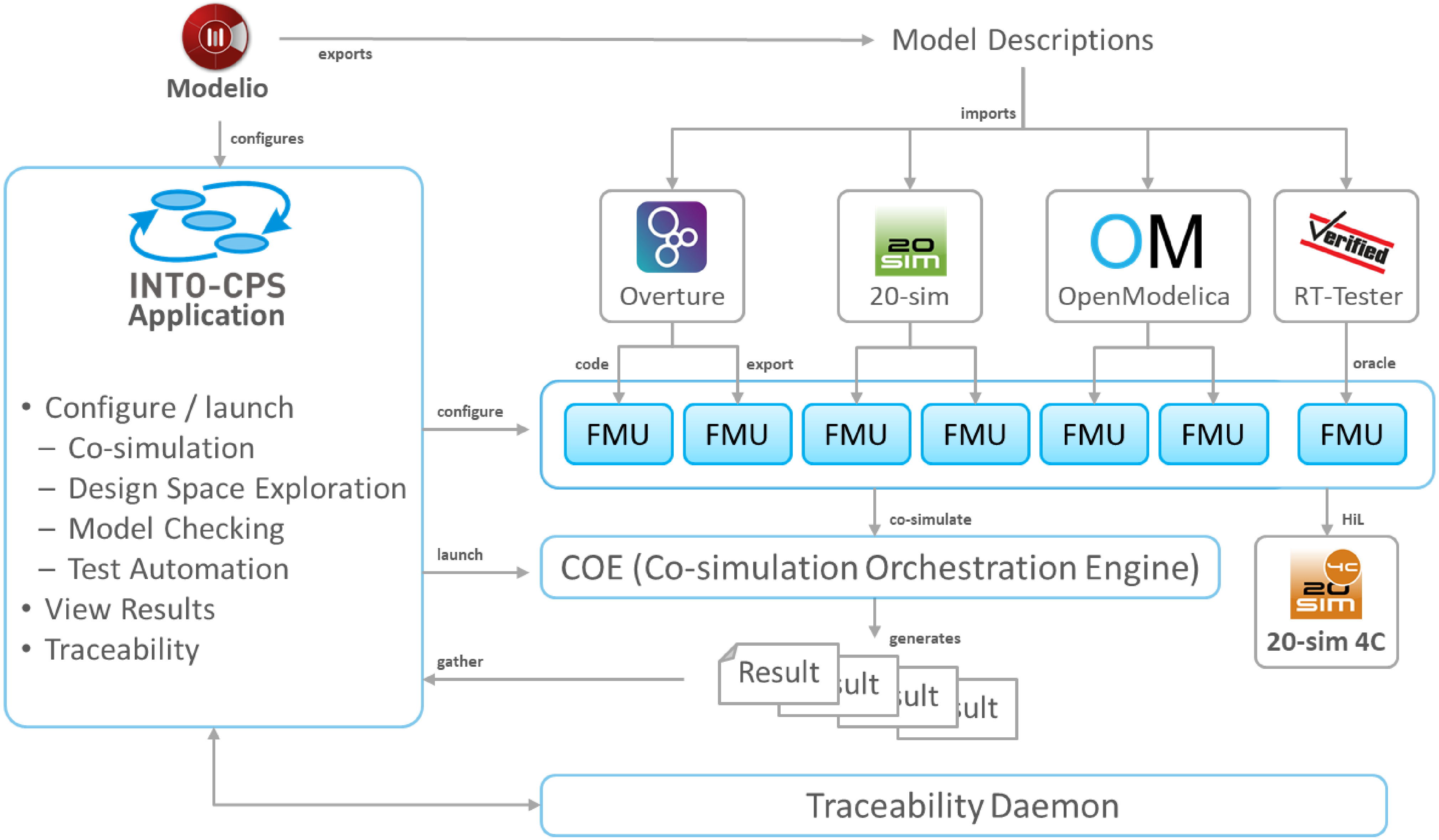
Digital Twin approach requires fine-grained modeling



Car modeling is the results of different application domains interacting with each other

Is the Co-simulation an enabler technology for car Digital Twin?

2. The INTO-CPS tool-chain



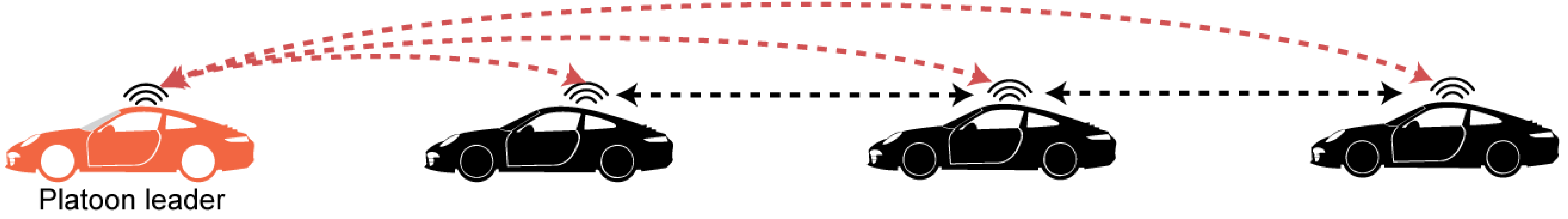
- Configure / launch
  - Co-simulation
  - Design Space Exploration
  - Model Checking
  - Test Automation
- View Results
- Traceability

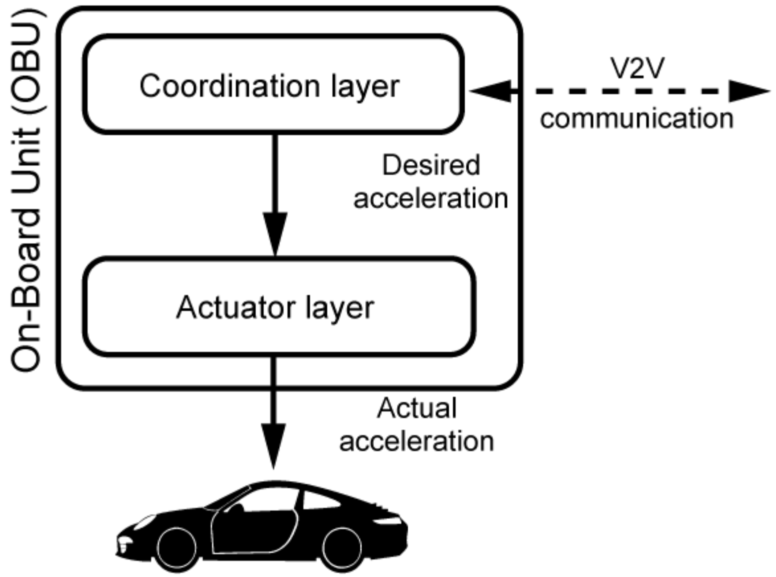
Outline

2. Car Platooning

- The platooning is the technique/strategy by which **two or more vehicles circulate on the road in a joint and coordinated manner.**
- A number of vehicles follows a leader, maintaining a safe inter-vehicle distance
- Assumptions:
  - All vehicles have same length and properties.
  - Fixed size of the platoon.

2. Vehicle to Vehicle Approach





The coordination layer computes the platoon control law using data read from on-board sensors and received from other vehicles through V2V communication

V2V communication options:

- **IEEE 802.11p**
- **LTE V2V (mode 3) → use base station scheduler**
- **LTE V2V (mode 4) → distributed scheduling (semi-persistent scheduling)**

2. Cooperative Adaptive Cruise Control

- ***CACC*** **is a class of controllers that result in *string-stable platoons***
  - perturbations at the head of the platoon propagate smoothly towards the tail
  - **Parameters:** *C_1, dampingRatio ξ, controllerBandwidth ω_n, vehicle length l_{i-1}, targetDistance d_{des}*
  - ***Inputs:*** x_i, x_{i-1}, $\dot{x}_i$, $\dot{x}_{i-1}$, $\dot{x}_0$, $\ddot{x}_{i-1}$, $\ddot{x}_0$
  - ***Output***: $\ddot{x}_{i\_des}$

$$\dot{\varepsilon}_i = \dot{x}_i - \dot{x}_{i-1}$$

$$\varepsilon_i = x_i - x_{i-1} + l_{i-1} + d_{des}$$

$$\alpha_1 = 1 - C_1, \alpha_2 = C_1$$

$$\alpha_3 = -\left(2\xi - C_1\left(\xi + \sqrt{\xi^2 - 1}\right)\right)\omega_n$$

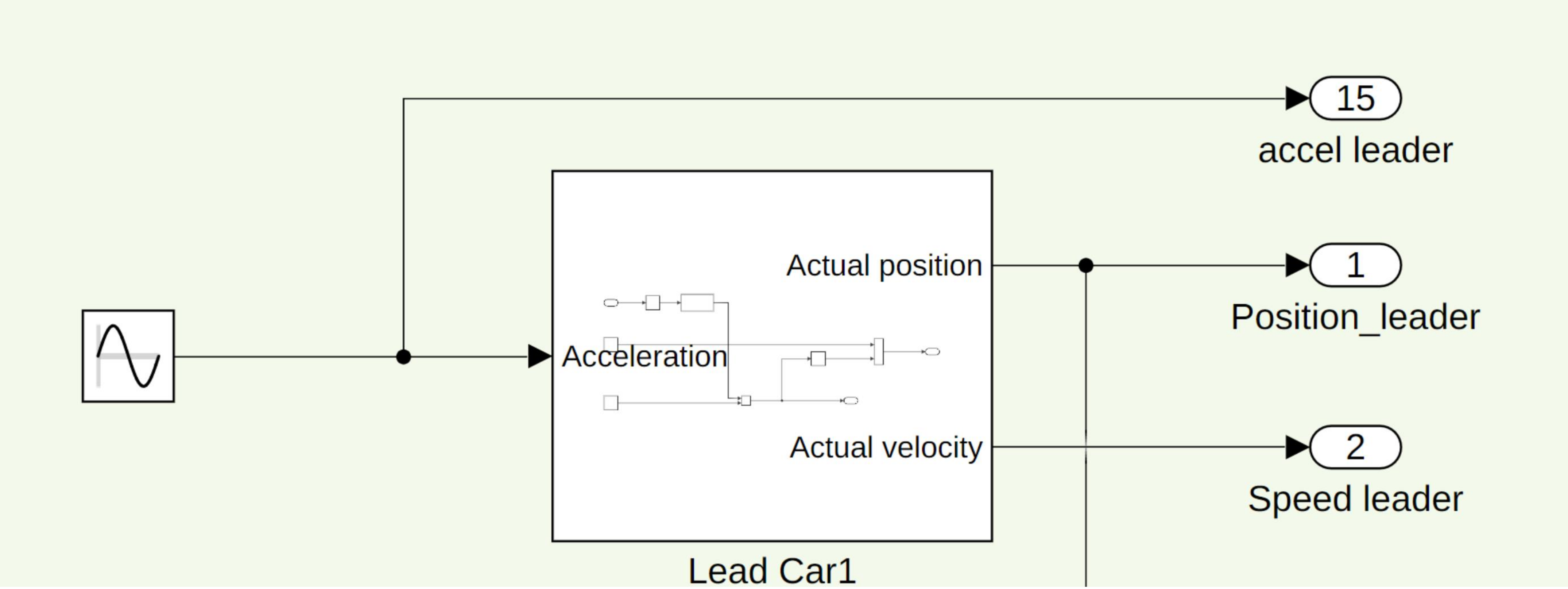
$$\alpha_4 = -C_1\left(\xi + \sqrt{\xi^2 - 1}\right)\omega_n$$

$$\alpha_5 = -\omega_n^2$$

$$\ddot{x}_{i\_des} = \alpha_1 \ddot{x}_{i-1} + \alpha_2 \ddot{x}_0 + \alpha_3 \dot{\varepsilon}_i + \alpha_4(\dot{x}_i - \dot{x}_0) + \alpha_5 \varepsilon_i$$

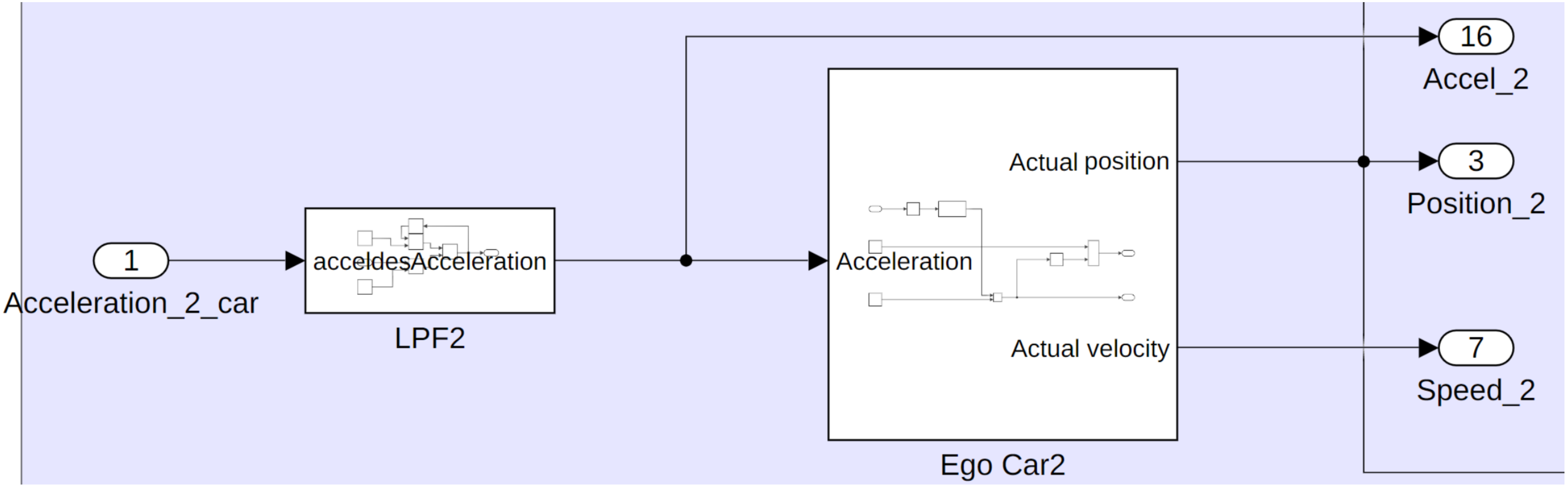
2. Simulink Model of the leader car

- Pre-defined Acceleration
- 3 outputs



2. Simulink Model of a following car

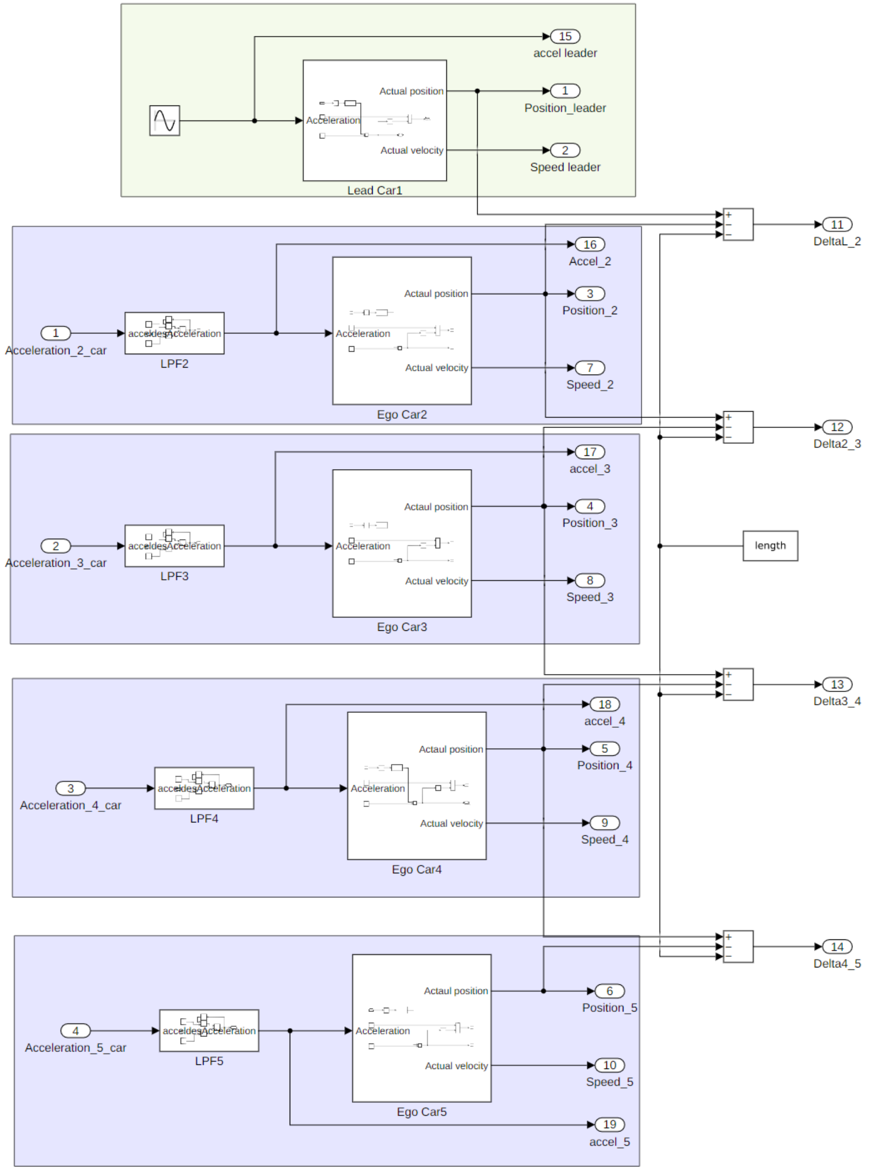
- Actuation lag, Low Pass Filter for realistic behavior of the acceleration
- 1 input, 3 outputs



2. Simulink Model of the platoon

- 1 Leader
- 4 Followers
- Outputs the distances among the vehicles

Parameter	Value
Platoon size	5 cars
Target distance	10 meters
C_1	0.5
ξ	1
ω_n	0.2
l_{i-1}	4 meters



2. Delay FMU

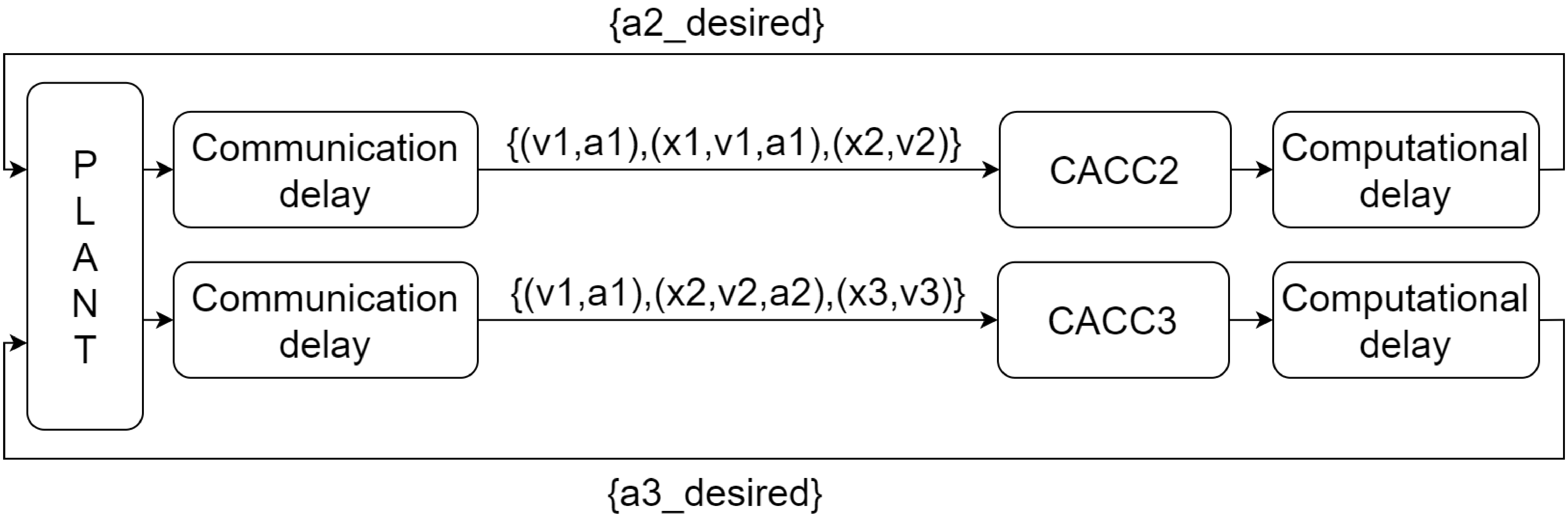
Main parameters:

- tick-size
- detDelay
- expDelay

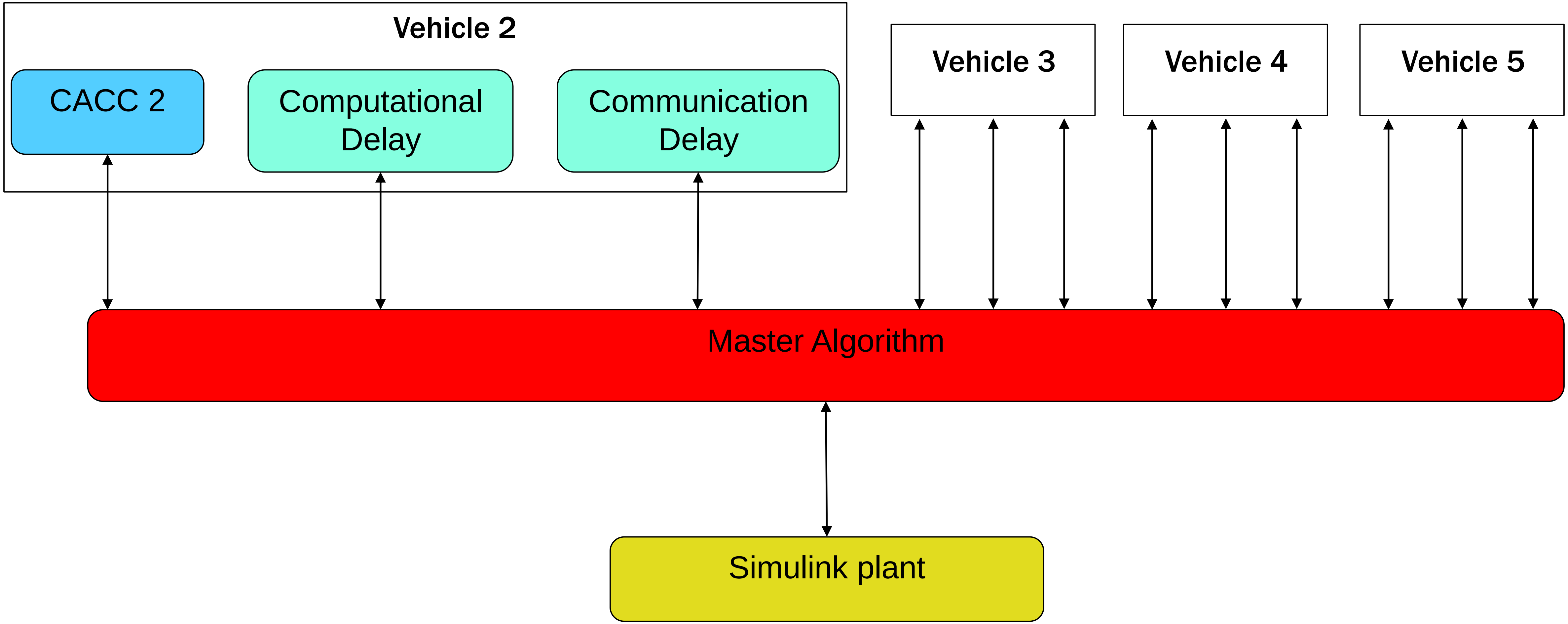
```
State* tick(State* st) {
    if (st->mode == X1 && ( st->time < st->delay )) {
        st->time = st->time + st->tickSize;

    } else if (st->mode == X1 && ( st->time >= st->delay )) {
        st->output = st->memory;
        st->memory = st->input;

        st->delay = st->time +
                        st->tickSize * ran_expo(st->expDelay) +
                        st->tickSize * st->detDelay;


        st->time = st->time + st->tickSize;
    }
    return st;
}
```

2. V2V: System model architecture

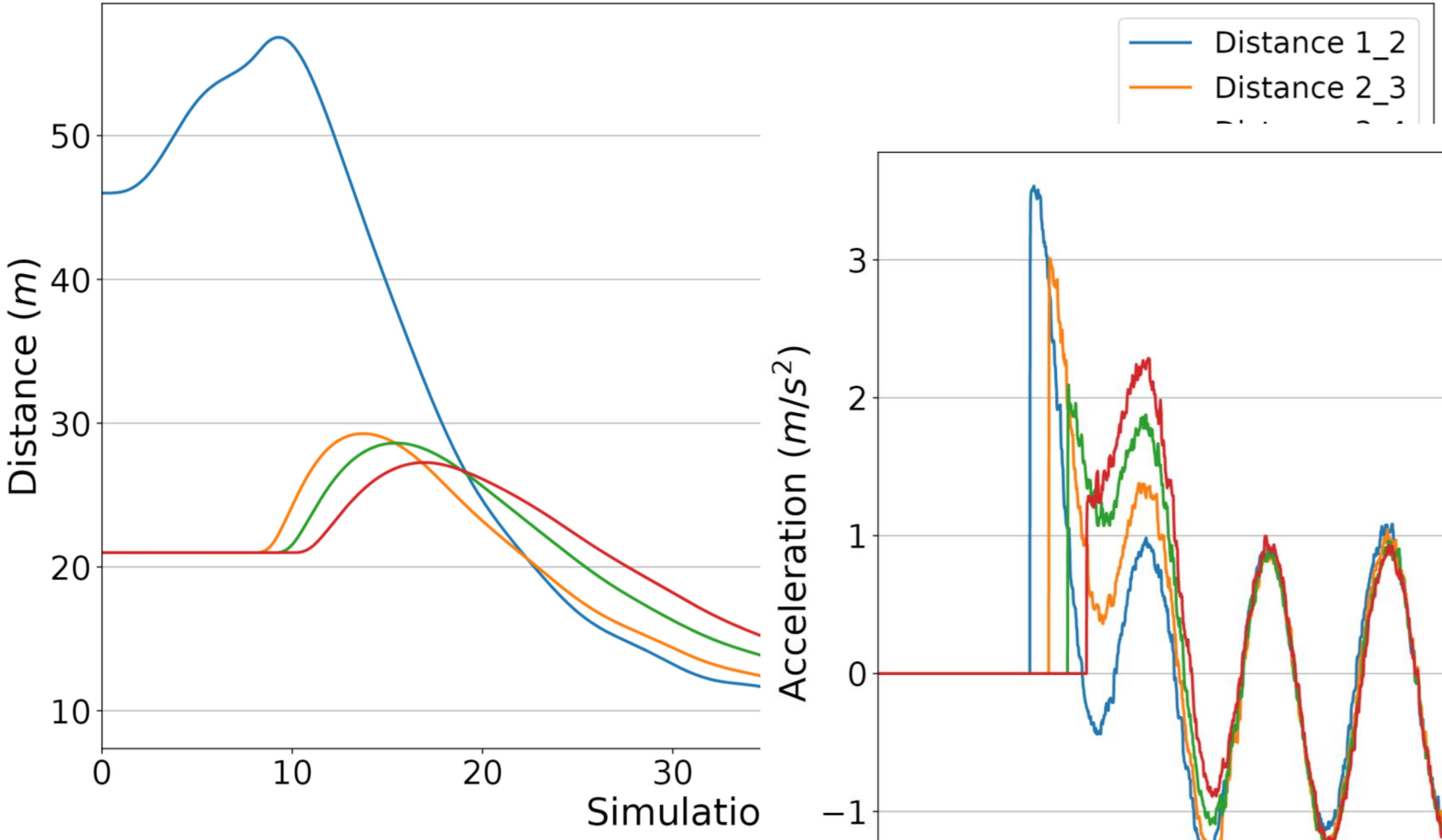
- Delay FMUS used to include delays in the scenario

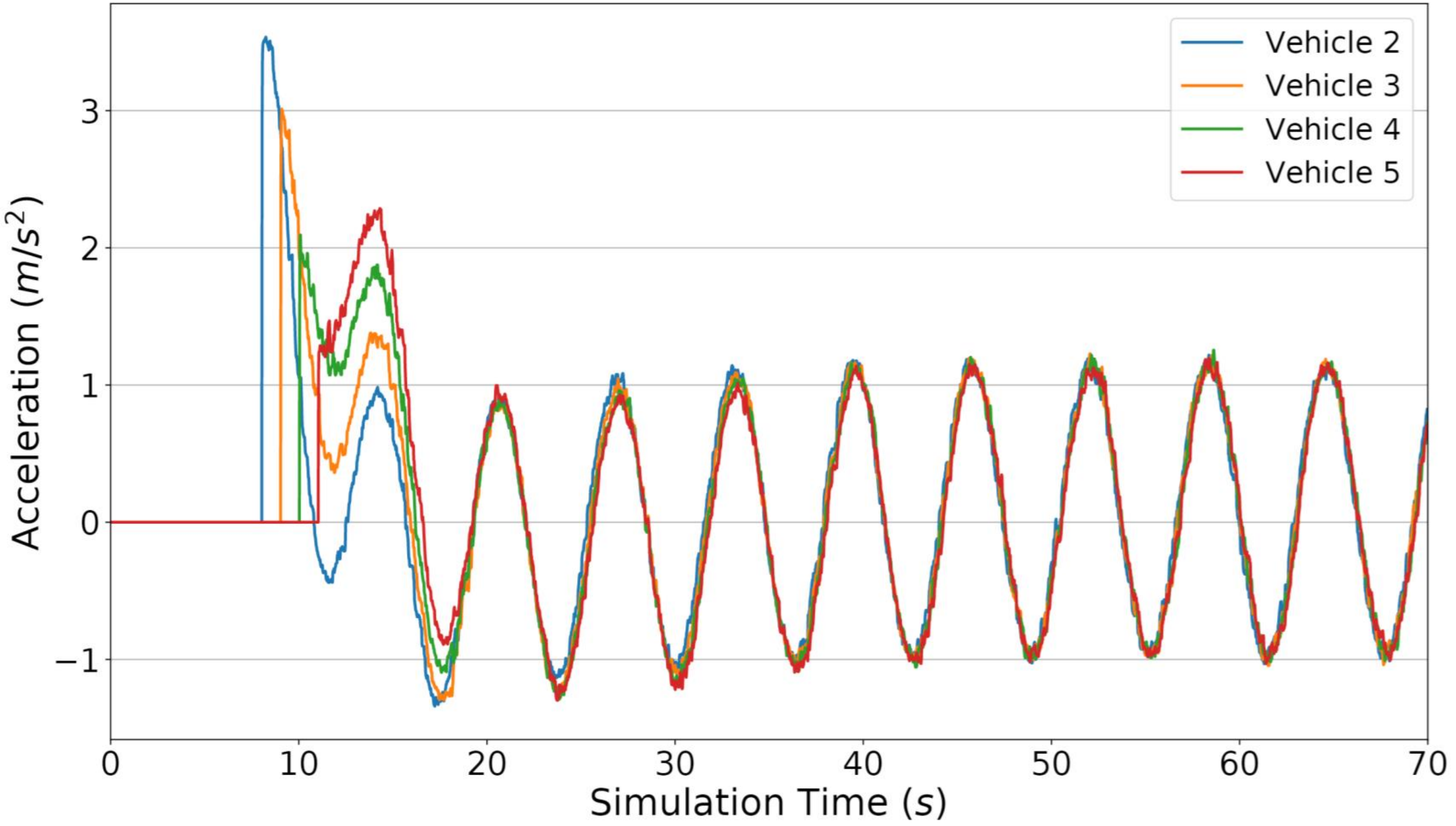


2. V2V: Co-simulation architecture



2. V2V: Co-simulation results





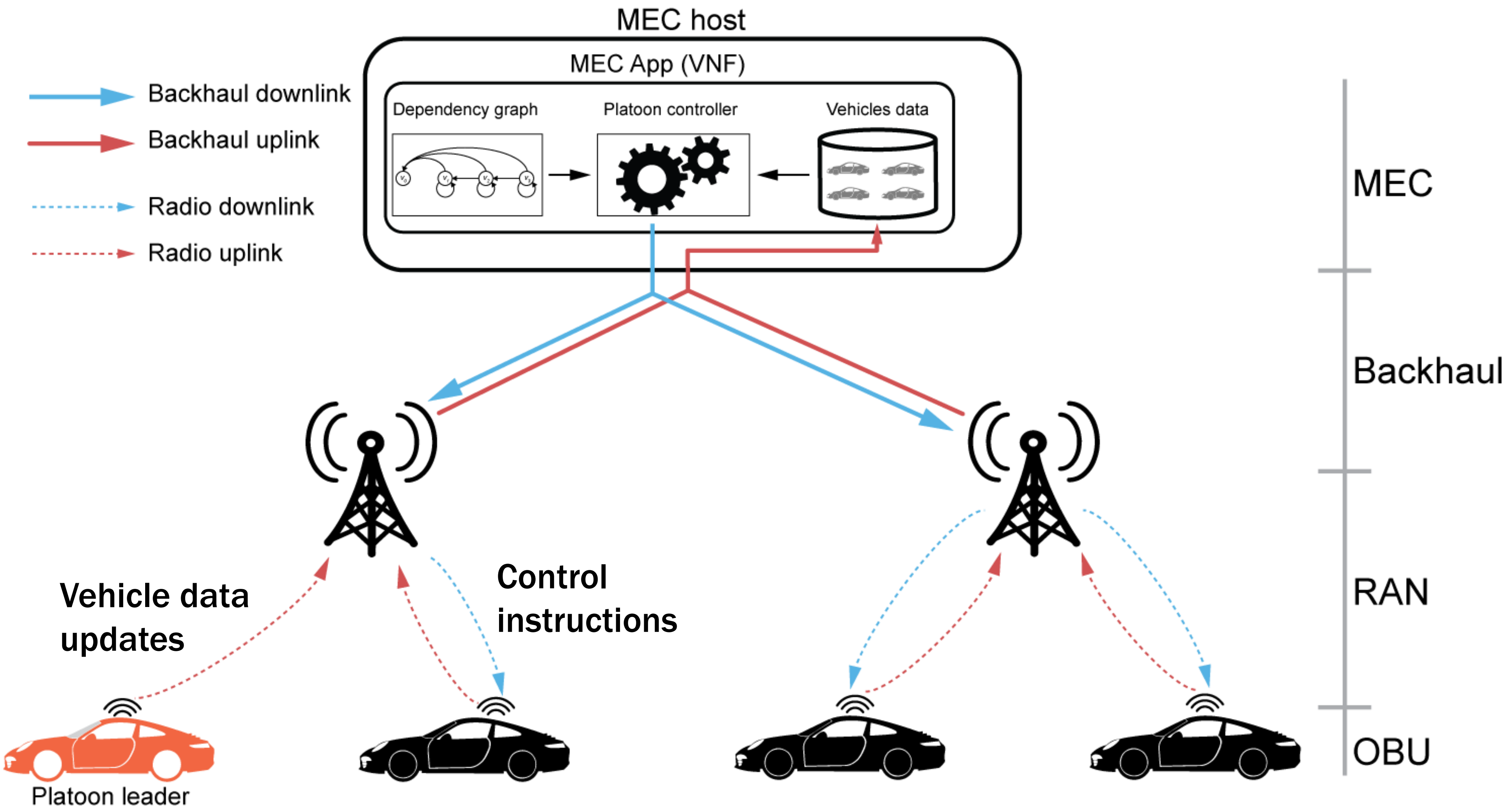
2. MEC/Edge

Multi-access Edge Computing (MEC) architecture by ETSI provides:

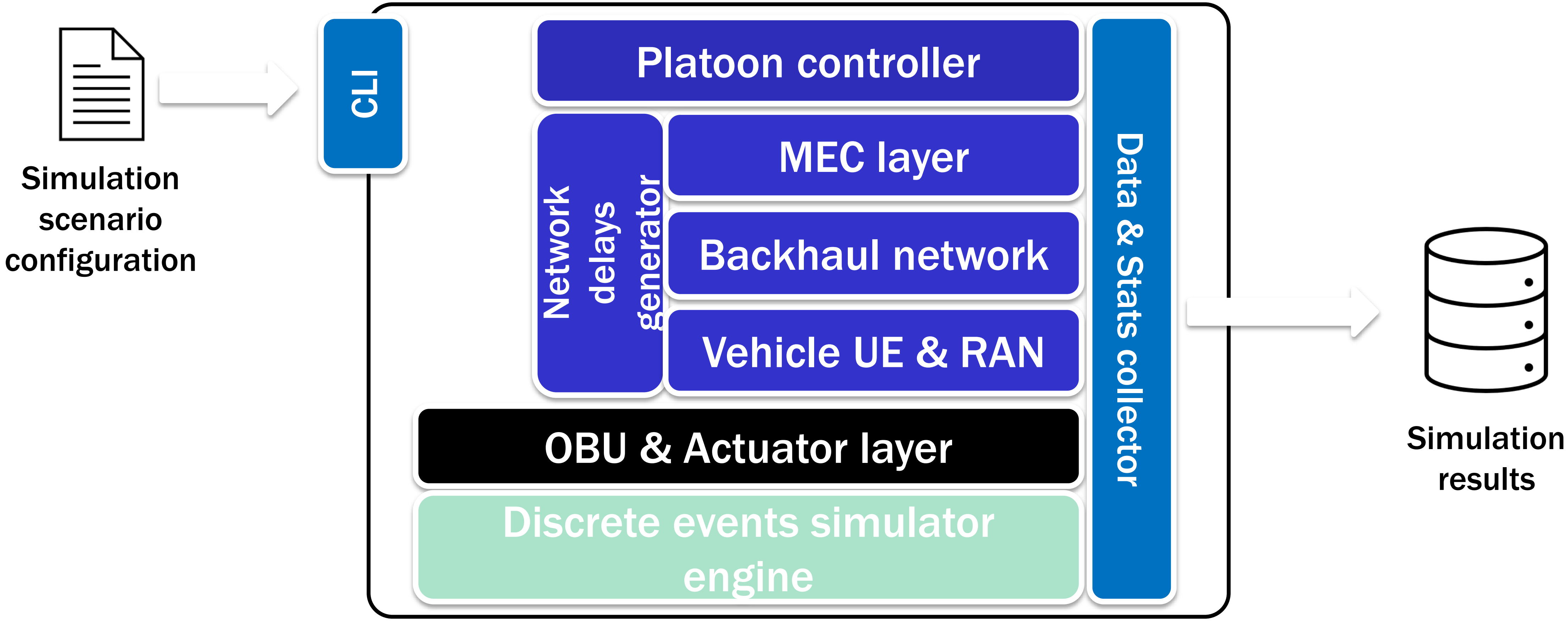
- Edge computing capabilities fully integrated with 5G Core network
- Low-latency communication
- Virtualized computing and storage resources (limited compared to Cloud)
- Distributed computing architecture

- **MEC is a suitable architecture for implementing CPS and supporting the stringent delay requirements of Digital Twin Paradigm**

2. MEC assisted platooning



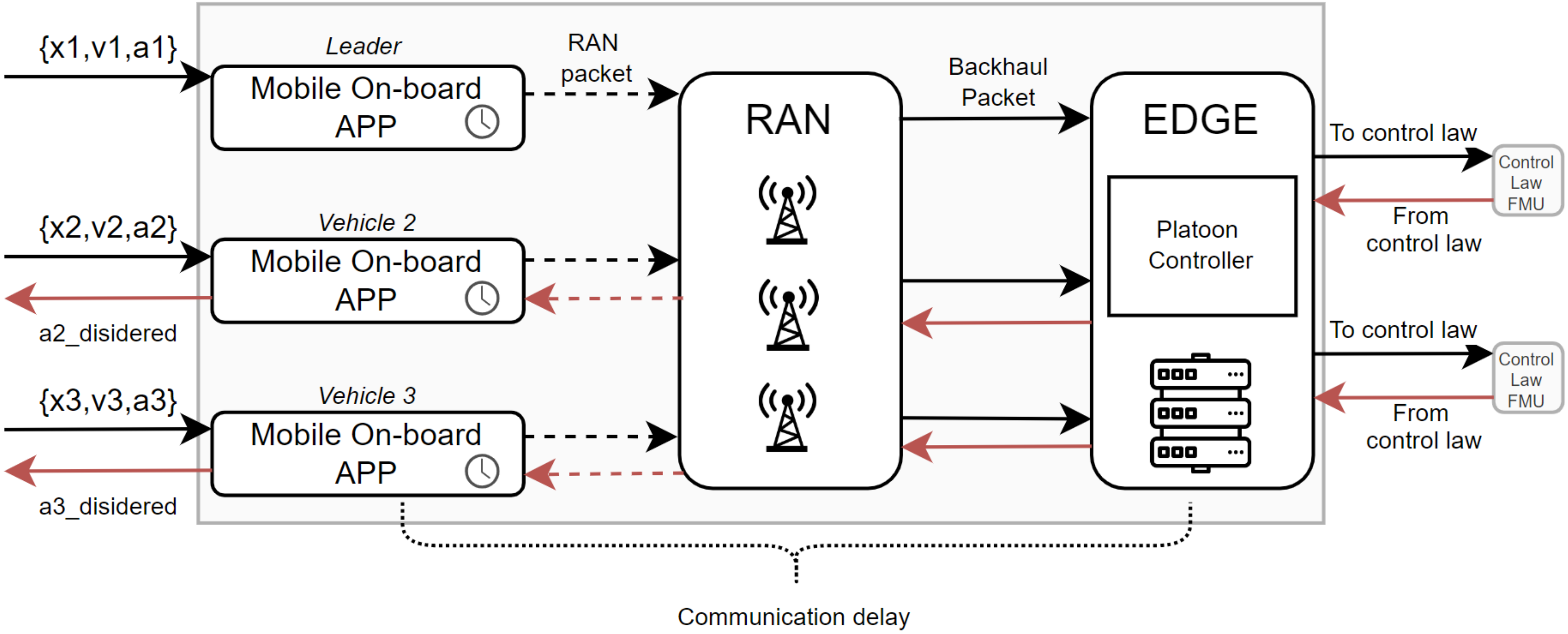
2. MEC Platoon Python simulator



2. MEC simulator as an FMU

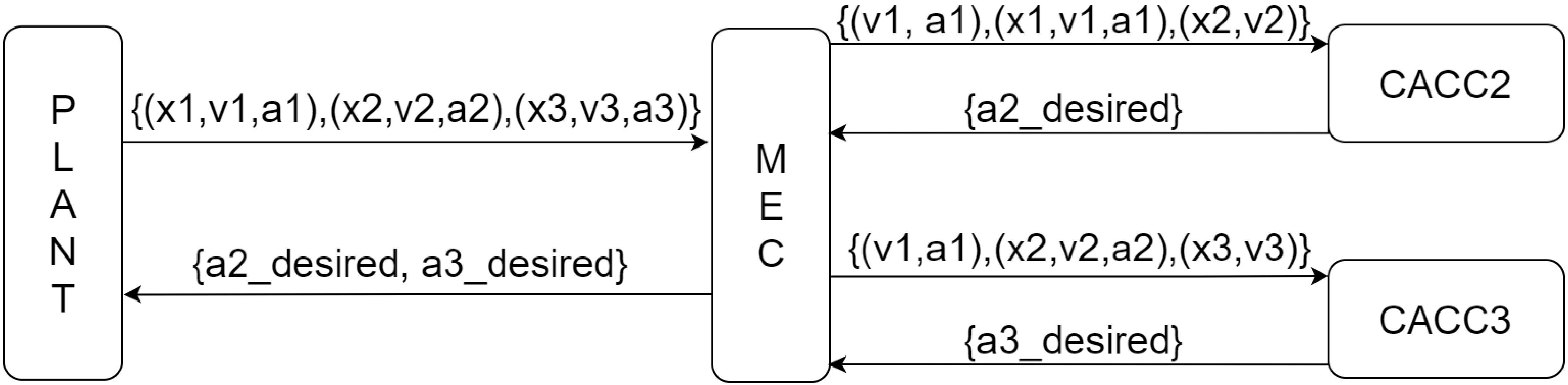
- FMU created using UniFMU

MEC FMU

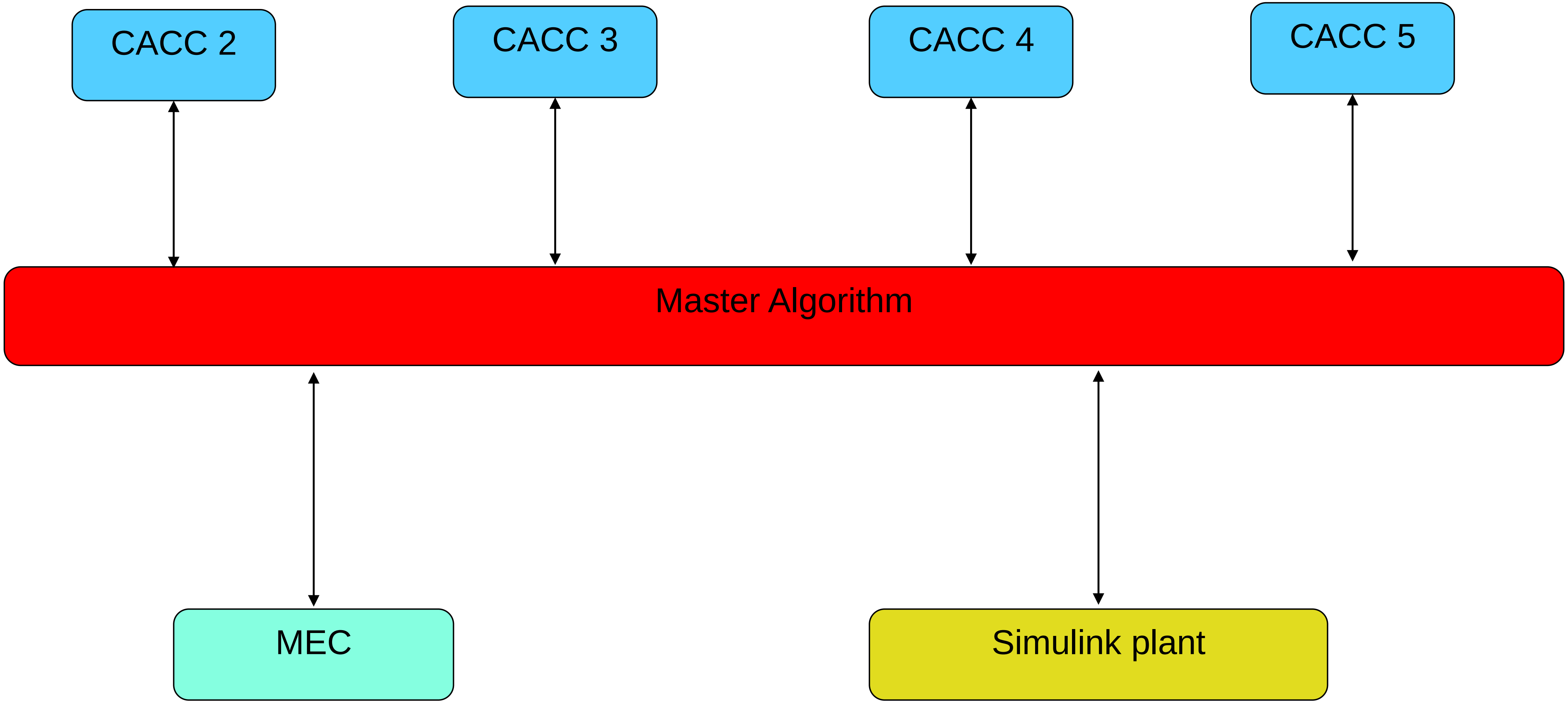


2. MEC: System model architecture

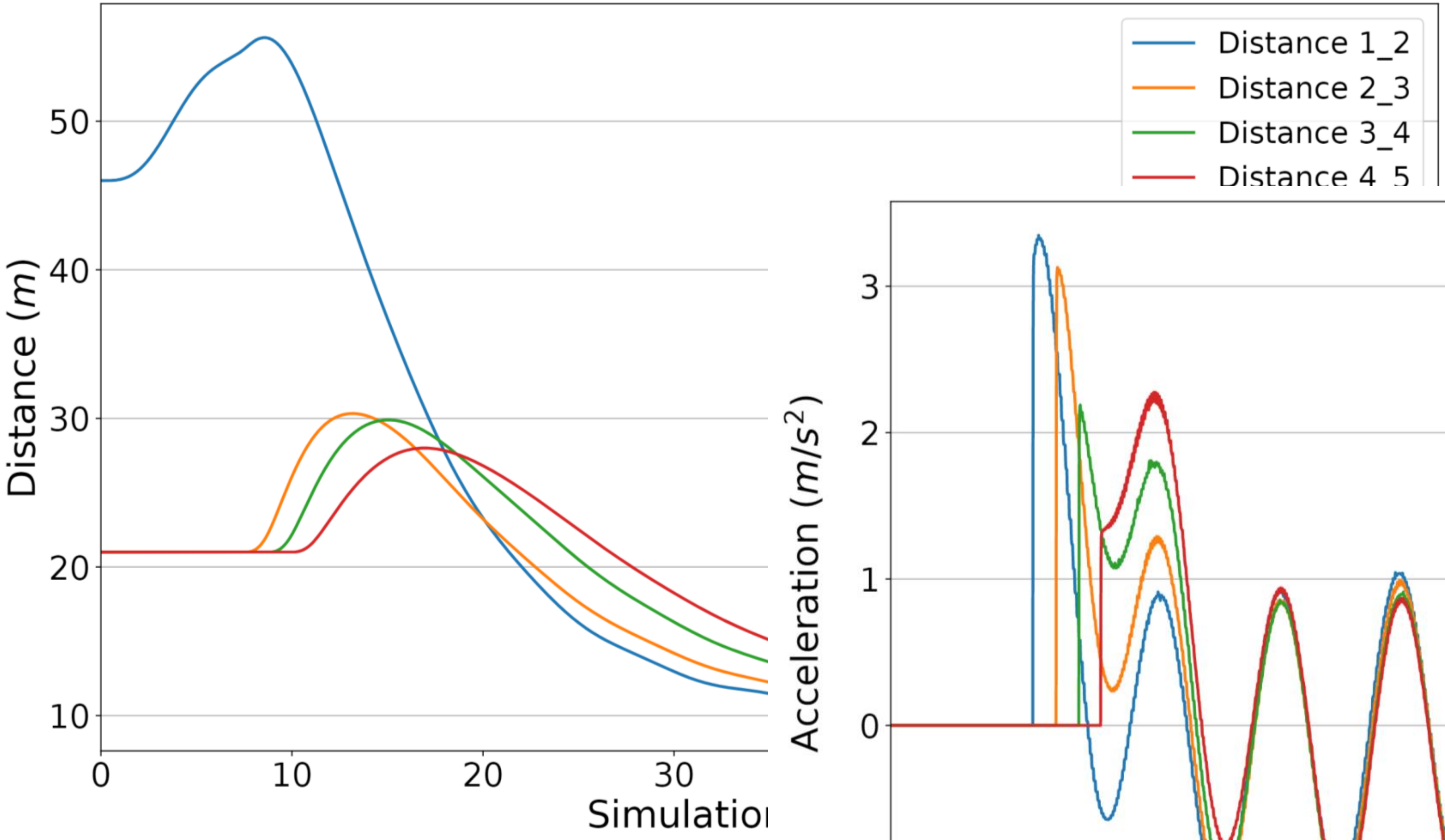
- Delays embedded in the MEC model

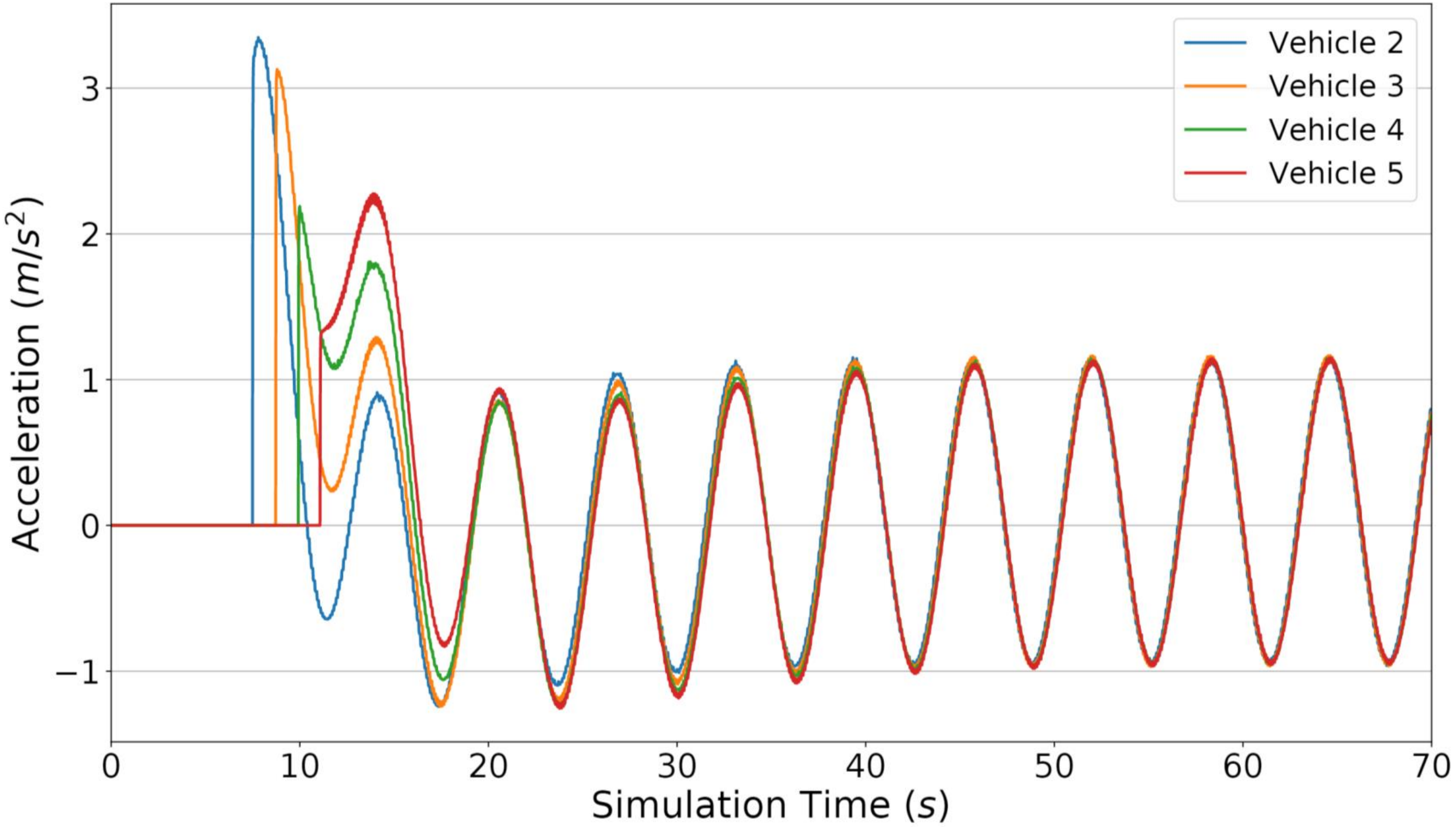


2. MEC: Co-simulation architecture



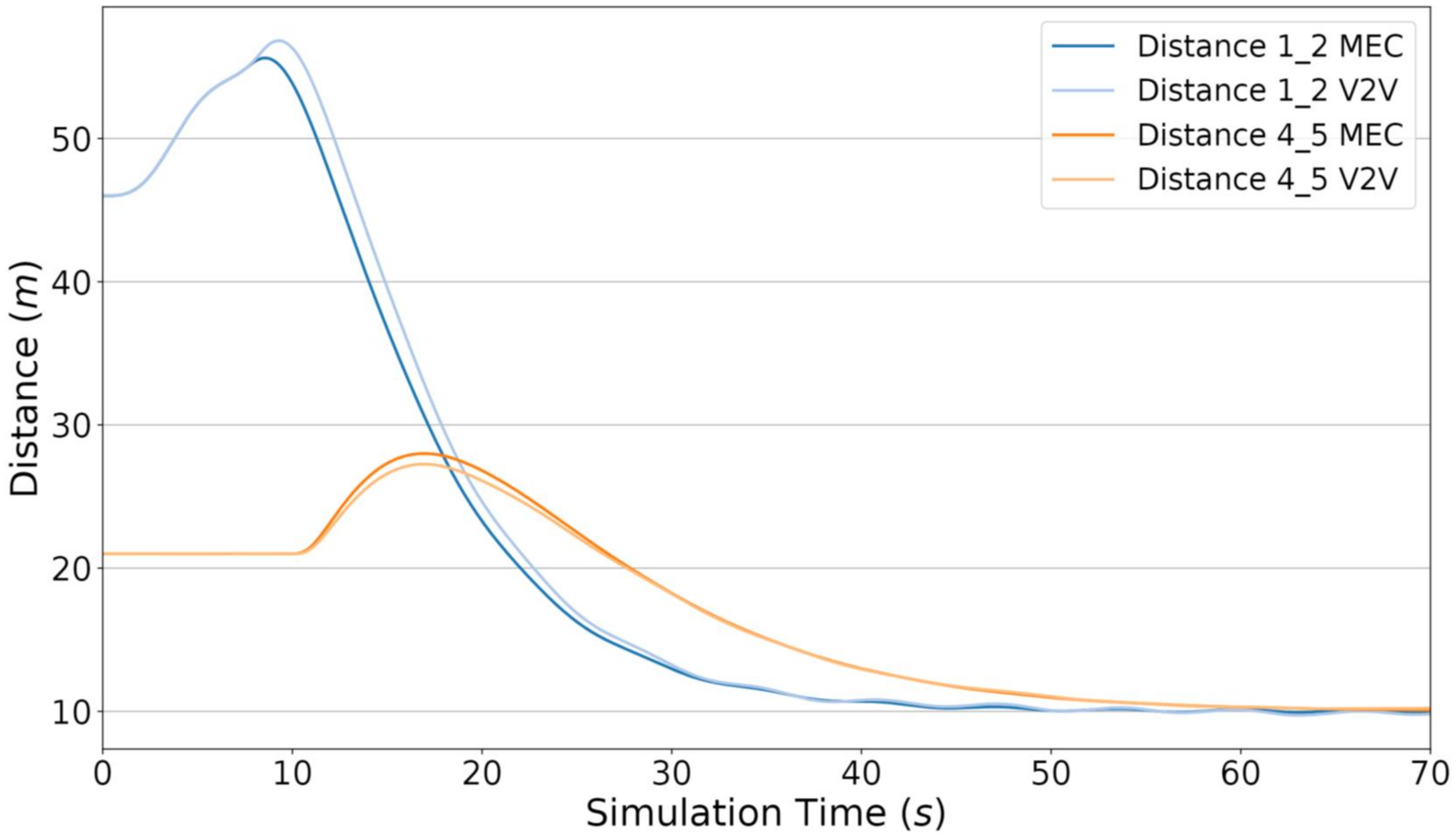
2. MEC: Co-simulation results





2. Comparison MEC and V2V

- Smooth propagation.
- MEC is more stable.



2. Platoon conclusions

- Platoon that includes an Edge based communication is co-simulated.
- Co-simulation enabled the analysis of different scenarios, MEC and V2V.
- CACC algorithm has performed in a correct manner, in both scenarios.
- The MEC scenario exhibits a more stable behavior.

Future work

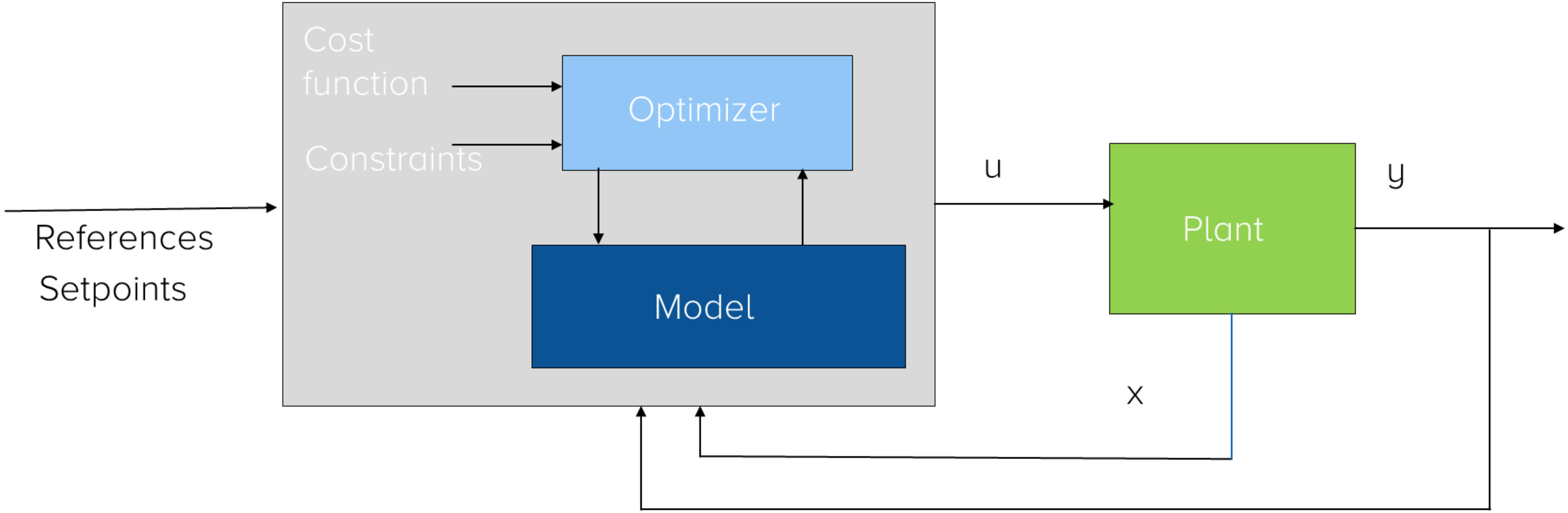
- Modeling sensor components and longitudinal dynamics.
- Enhancing the channel behavior modeling with OMNeT++ and Simu5G.
- Investigating on Design Space Exploration (DSE).
- Studying threats related to cyber-attacks.

Outline

1. Introduction
2. Car platooning co-simulation
    1. Modeling a car platoon with Simulink
    2. Including network communication inside an FMI co-simulation
    3. Improving the modeling of the network with a specific simulator
    4. Analysis of the results
3. ADAS co-simulation with Model Predictive Control
    1. Brief introduction to Model Predictive Control
    2. Extending a library for MPC to cope with FMI standard
    3. ADAS case study: following a trajectory while avoiding obstacles
    4. Analysis of the results
4. Conclusions

3. Model Predictive Control

- Use a prediction model to optimize the control value in a prediction horizon



3. Formal Model

- **Mathematical formulation of the optimization problem**

Cost function

$$\min_{\boldsymbol{u},\boldsymbol{p},T} \quad J(\boldsymbol{u},\boldsymbol{p},T;\boldsymbol{x}_0) = V(\boldsymbol{x}(T),\boldsymbol{p},T) + \int_0^T l(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t)\,\mathrm{d}t$$

System evolution

$$\text{s.t.} \quad \boldsymbol{M}\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t_0+t)\,, \quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0$$

Constraints

$$\boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t) = \boldsymbol{0}\,, \quad \boldsymbol{g}_T(\boldsymbol{x}(T),\boldsymbol{p},T) = \boldsymbol{0}$$

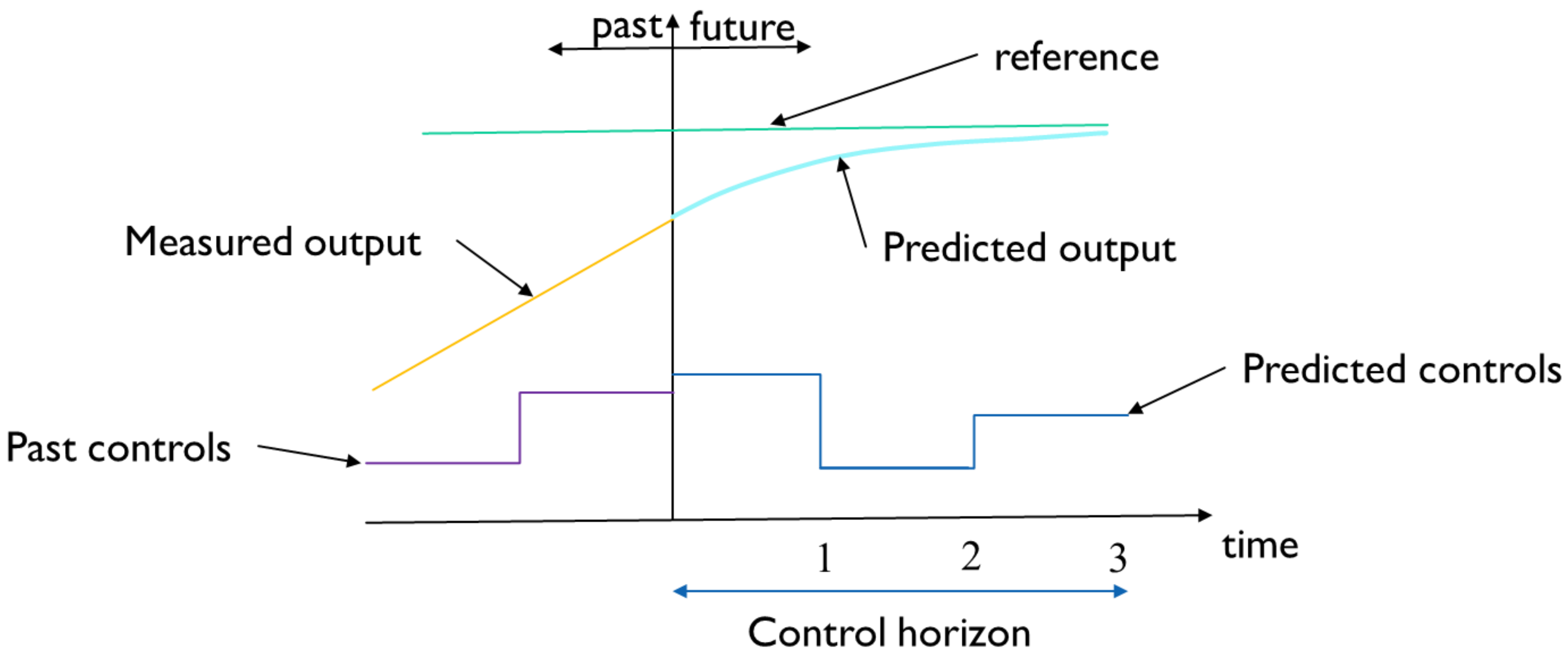
$$\boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t) \le \boldsymbol{0}\,, \quad \boldsymbol{h}_T(\boldsymbol{x}(T),\boldsymbol{p},T) \le \boldsymbol{0}$$

$$\boldsymbol{u}(t) \in [\boldsymbol{u}_{\min},\boldsymbol{u}_{\max}]$$

$$\boldsymbol{p} \in [\boldsymbol{p}_{\min},\boldsymbol{p}_{\max}]\,, \quad T \in [T_{\min},T_{\max}]$$

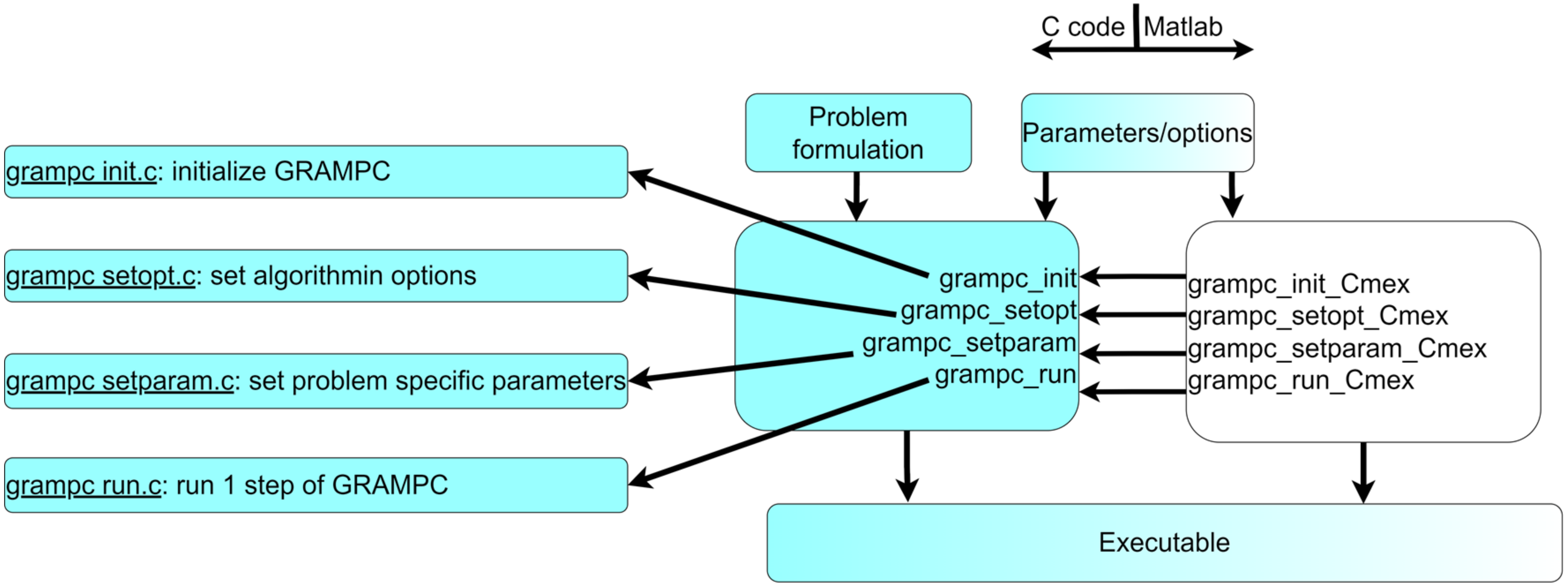
3. Prediction Horizon

- MPC optimizes the sequence of Predicted controls to produce the Predicted output that minimizes the error with the reference
- The control value provided to the controlled plant is the first of the sequence



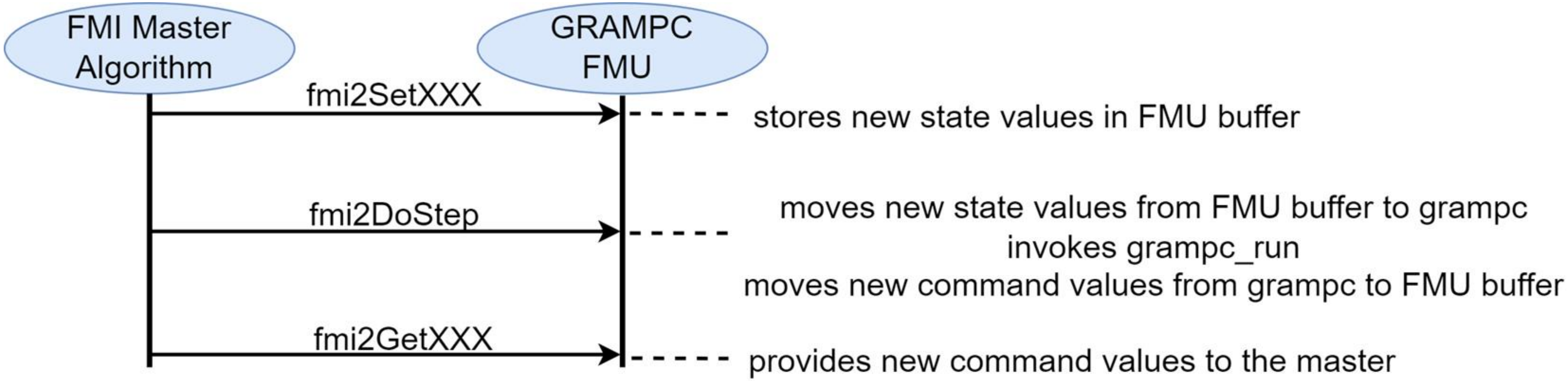
3. GRAMPC C library

- Problem formulation in C code
- Data visualization on Matlab



3. GRAMPC as an FMU

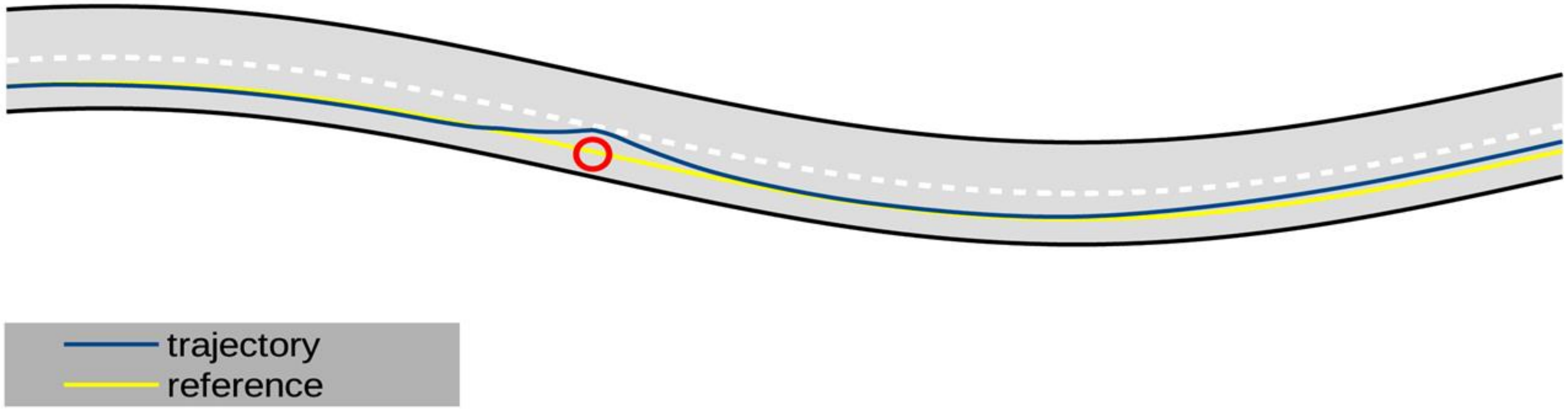
- **Main idea: GRAMPC model into an FMU written in C**



- **Initialization of GRAMPC called during the FMI initialization phase**
- **Semi-Automatic generation of the FMU based on templating system**

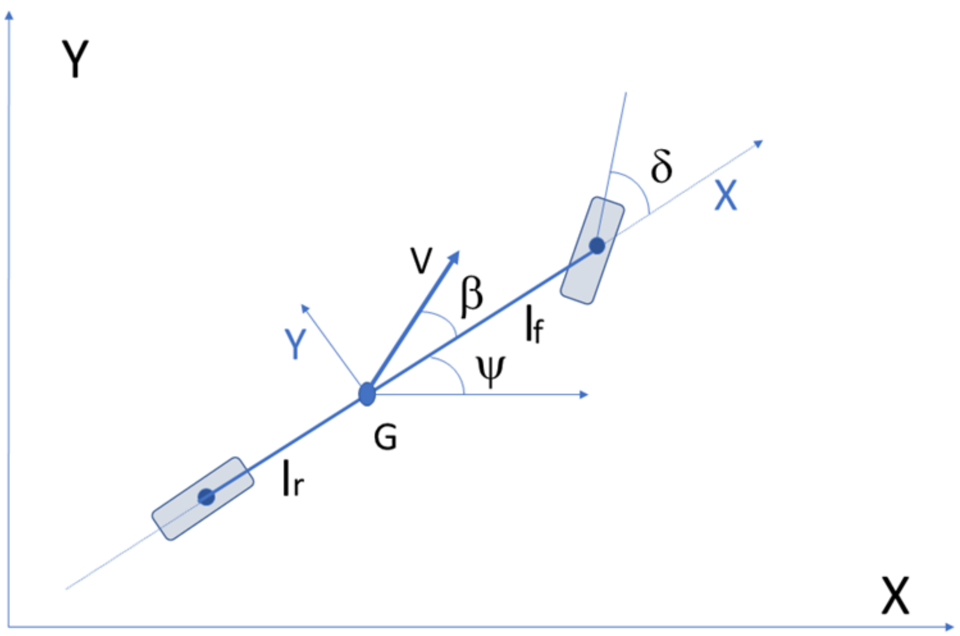
3. Automotive ADAS case study

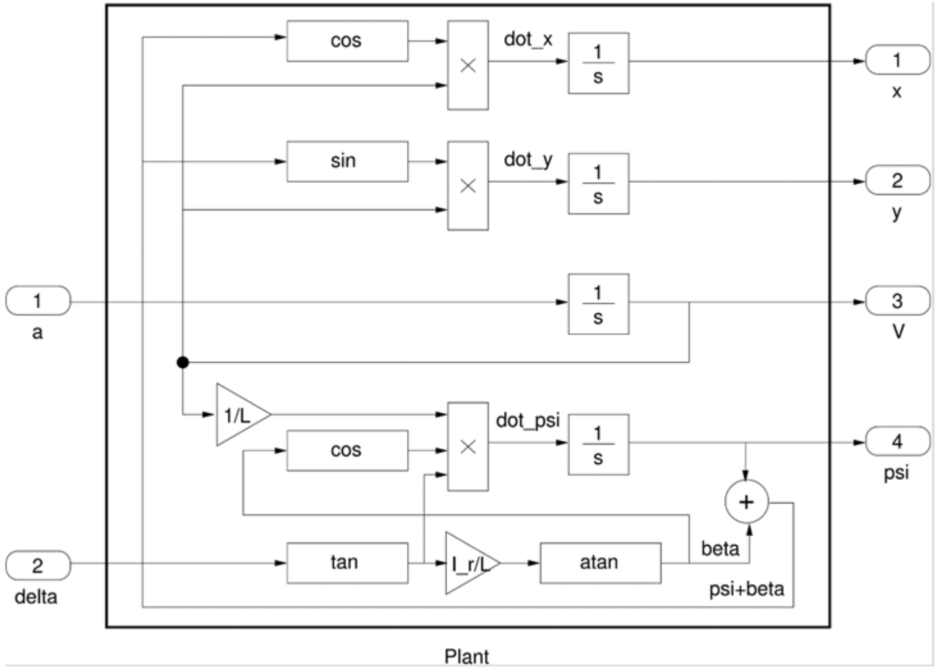
- Car following a reference trajectory
  - Reduce the error of the trajectory w.r.t. the reference
  - Avoid obstacles
  - Minimize GRAMPC execution time



3. Plant Model

- Bicycle model of a car modeled with Simulink
  - Simulink allows the automatic generation of the FMU





Coordinates of the center

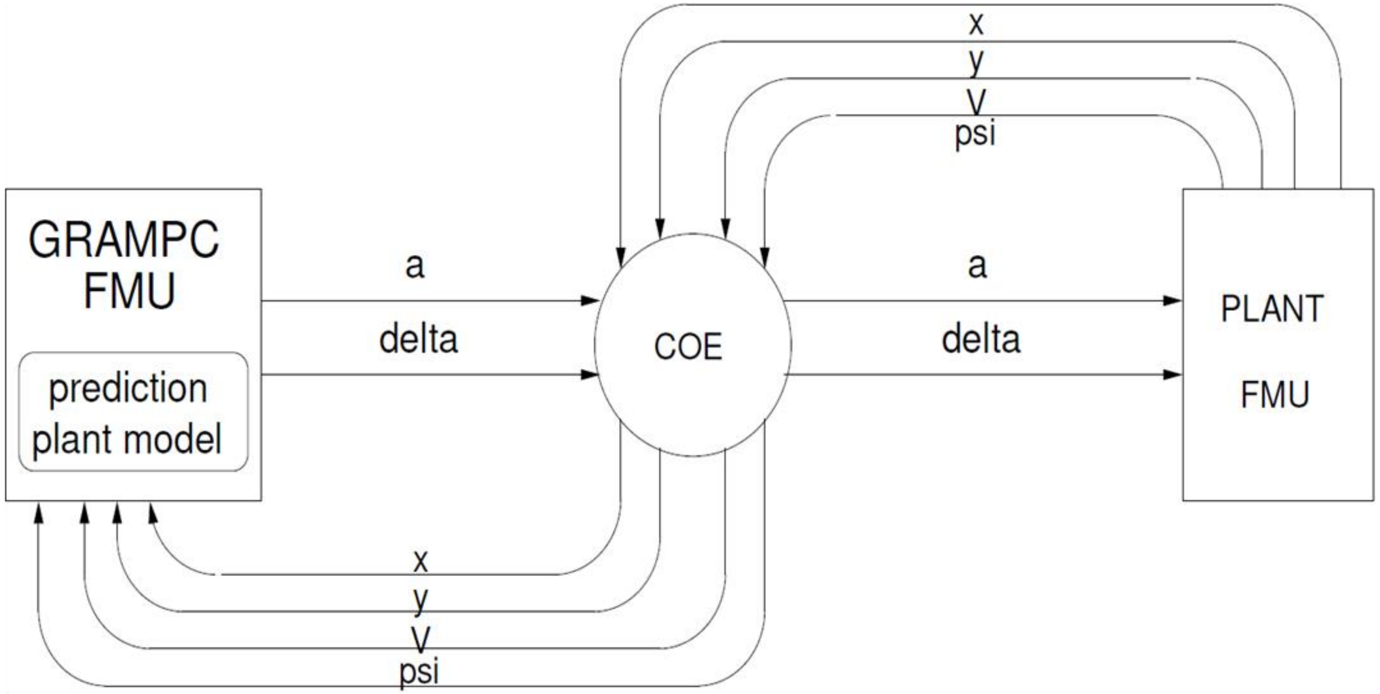
Param	Value
δ_{min}	-0.5 rad
δ_{max}	+0.5 rad
l_f	1.67 m
l_r	1.394 m

3. Co-simulation Architecture

- COE: Co-simulation Orchestration Engine
  - The INTO-CPS master algorithm

- Fixed step co-simulation algorithm
  - step-size = 1 ms

Parameter	Value
Time horizon	1 s
Solver	Euler
a_{max}	5.34 m/s^2
a_{min}	-11.2 m/s^2

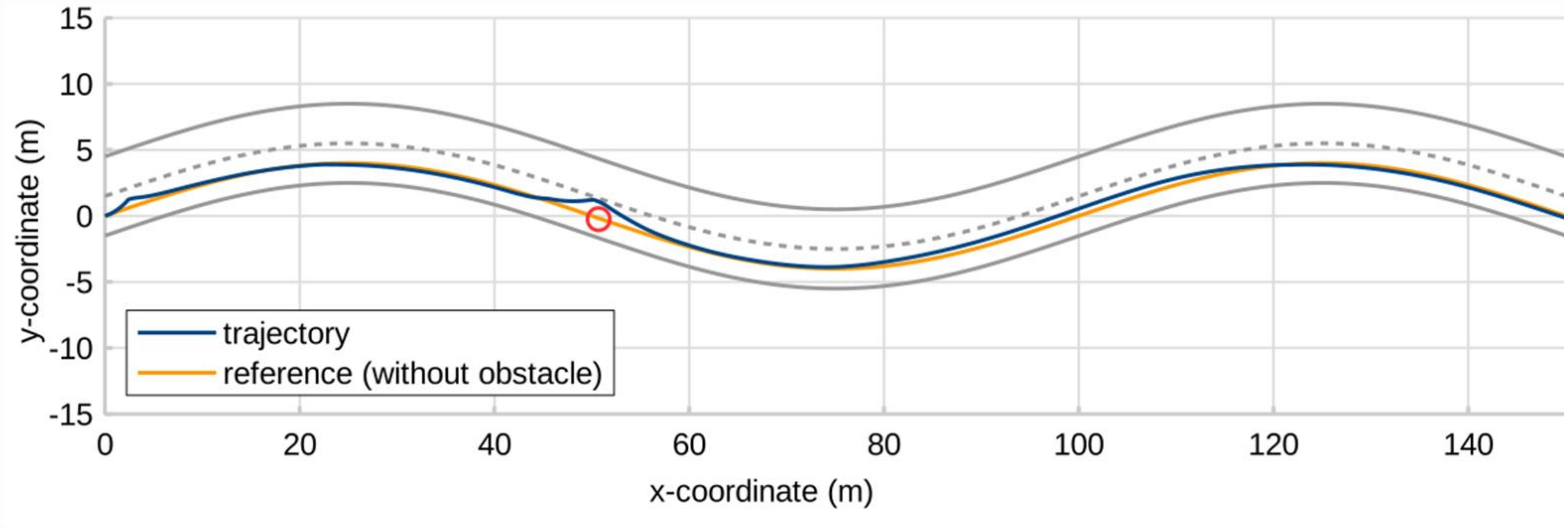


3. Results: Nominal Co-simulation

- **GRAMPC maximum execution time: < 1 ms**
  - Average error: 0.08 m

Executed on

Intel i7-7700 CPU @ 3.60GHz x8



3. Results: Parameter Variation

- INTO-CPS Design Space Exploration: measurement error in the car dimensions
  - 5% variation of the front track or rear track length

front track lf (m)	rear track lr (m)	Average error (m)
1.67	1.394	0.08
1.67	1.464	0.09
1.67	1.324	0.09
1.75	1.394	0.08
1.59	1.394	0.10

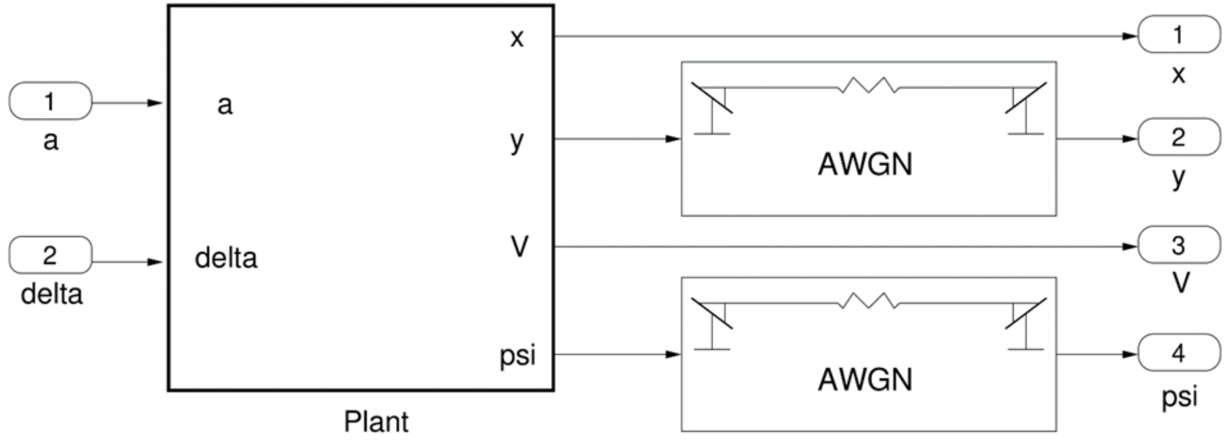
- Small impact on the average error

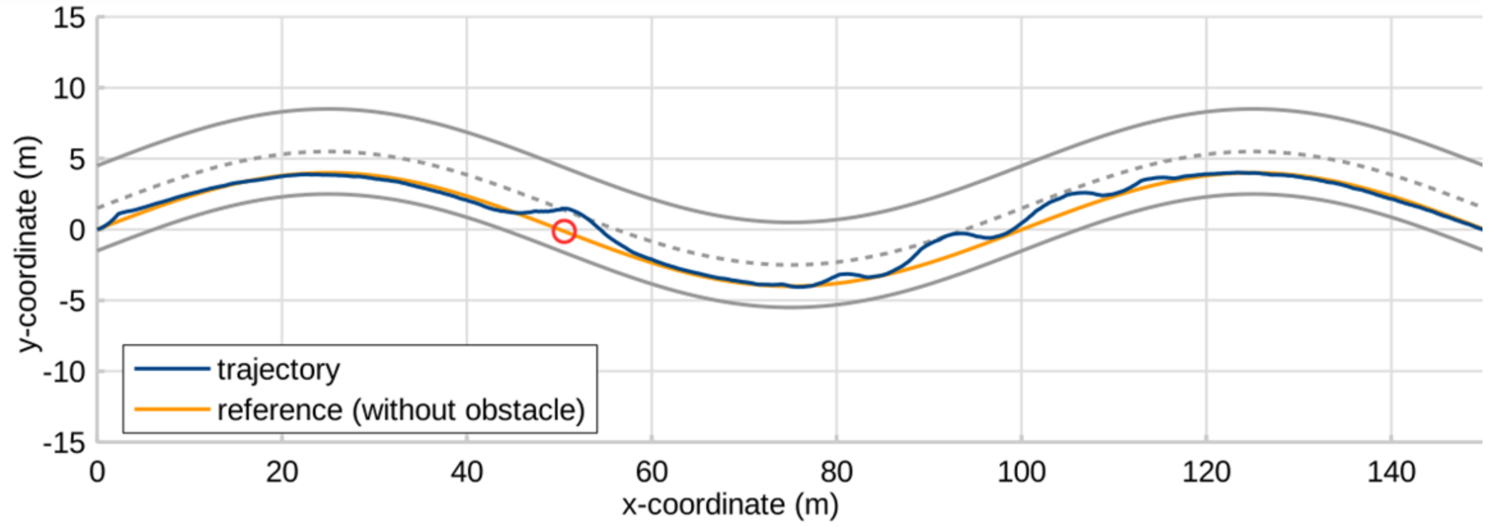
3. Results: Towards robustness analysis

- What happens with noisy sensors?

- Variance on y : 0.1 (10 cm)
- Variance on ψ: 0.01 (1°)

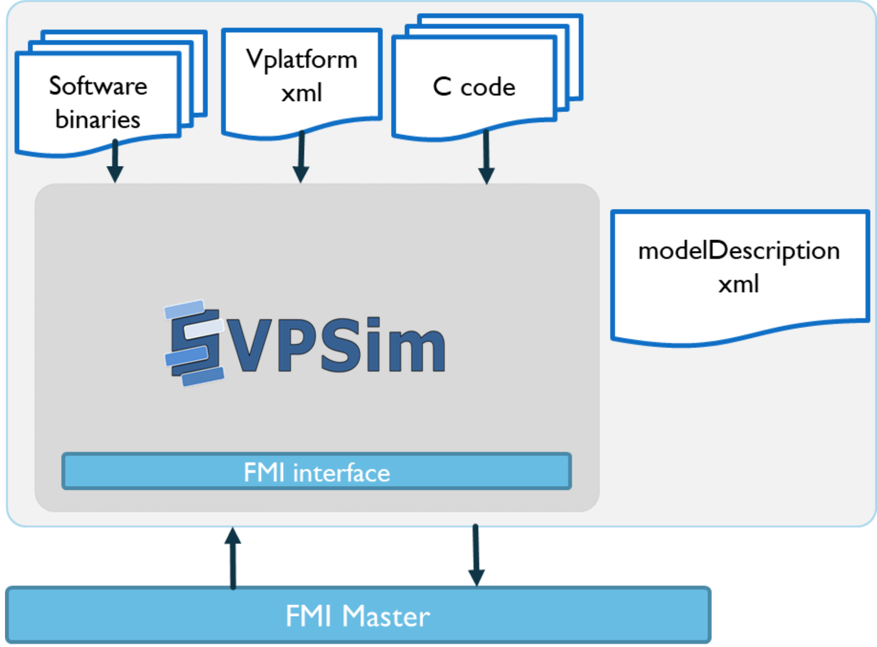
- average error: 0.240 m
- maximum error: 1.58 m

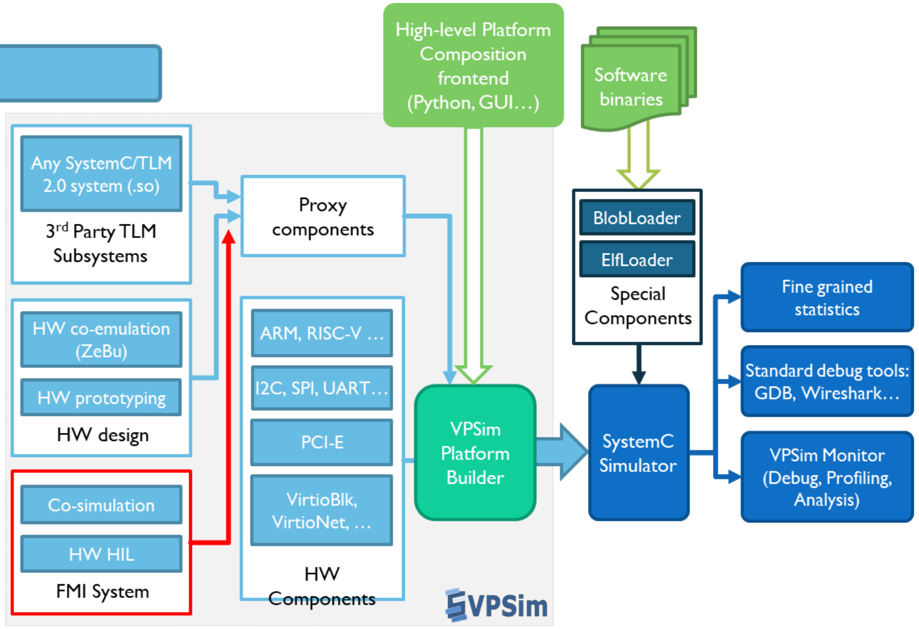




3. SESAM-VPSim

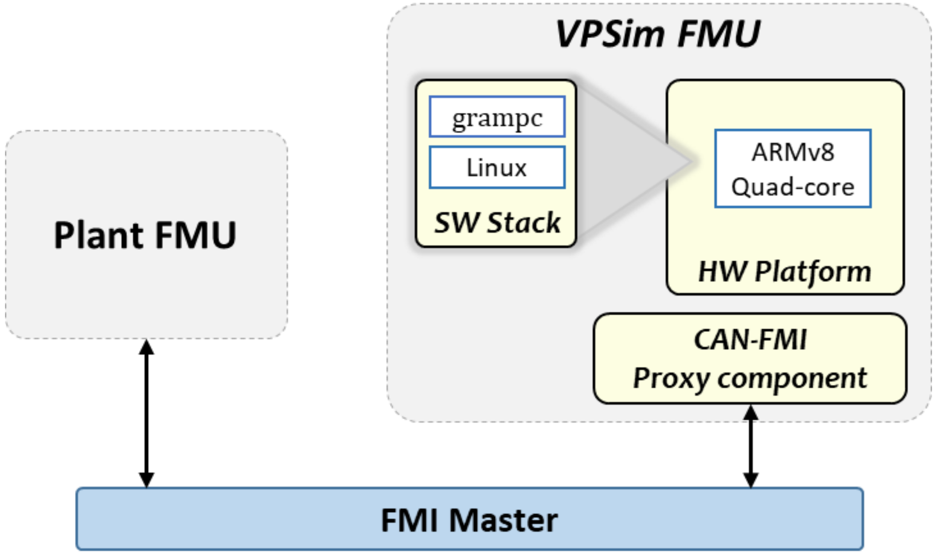
- Virtual prototyping Environment
  - Early Software development
  - Performance profiling and debug
  - Obtain fine-grained statistics
- Platform Composition
  - Rich model library
  - Load software binaries
  - High-level description with Python
- Rapid simulation able to run full software stacks
- Inter-operability with external models and tools
  - SystemC/TLM Interface
  - AXI-compatible RTL designs Interface
  - FMI 2.0 Co-Simulation





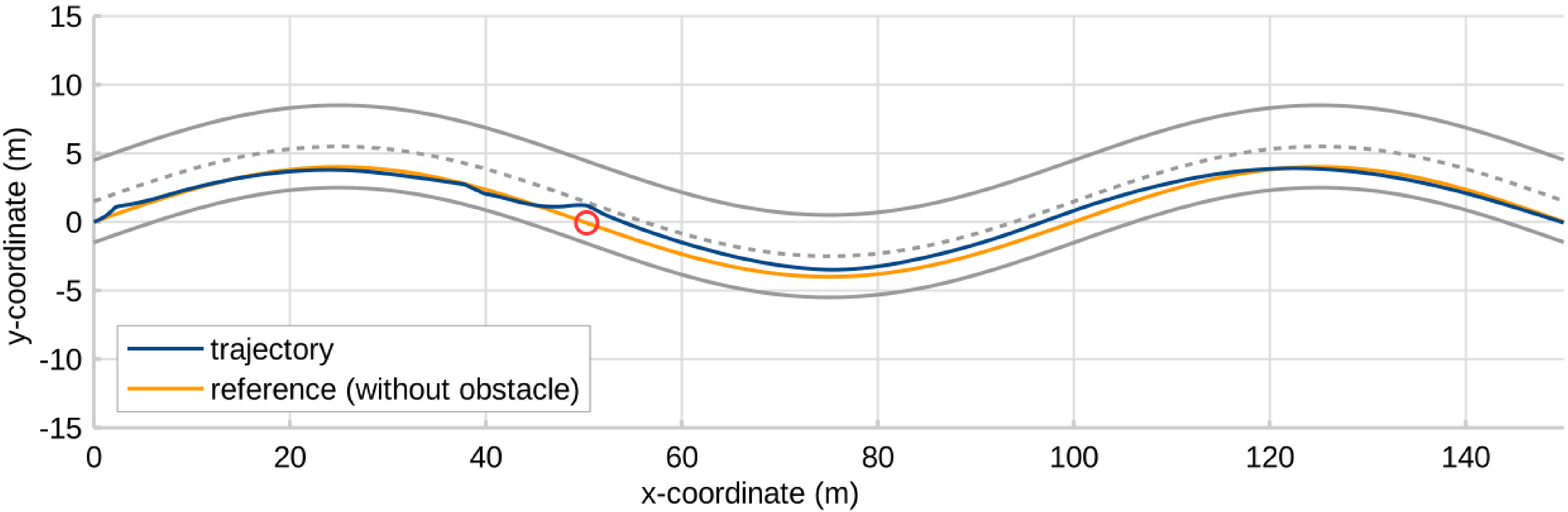
2. GRAMPC with SESAM-VPSim

- HW Platform Description
  - Quad-core ARMv8 64-bit processor architecture using QEMU
  - Each core has private L1 & L2 caches
  - The cores share four slices of LLC banks, which are connected to the NoC and peripheral devices
- The deployment of GRAMPC on a simulated architecture with VPSim
  - Evaluate the behavior of GRAMPC on the target hardware architecture



3. Results with SESAM-VPsim

- GRAMPC average execution time: 2.4 ms
- Mean error : 0.262 m



3. GRAMPC Conclusions

- The co-simulation of GRAMPC on a simulated architecture with VPSim
  - Comparison with the nominal co-simulation
  - Identify the impact of the processor architecture

- More realistic case study
  - add more obstacles
  - refine the vehicle model
  - add vision system

This work has been partially supported by the European Processor Initiative (EPI) project, which has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement n° 826647, and by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Department of Excellence).

4. Conclusions

Including network behavior and hardware components in the models is a key feature for enabling the usage of the Digital Twin paradigm.

Standard FMI co-simulation has allowed the integration of these features and simplifies their combination.

Many other features can be easily included, for example:

- Real-time monitoring of the real data
- Including control algorithms based on Artificial Intelligence

Co-simulation will play a relevant role in the modeling of the Digital Twin