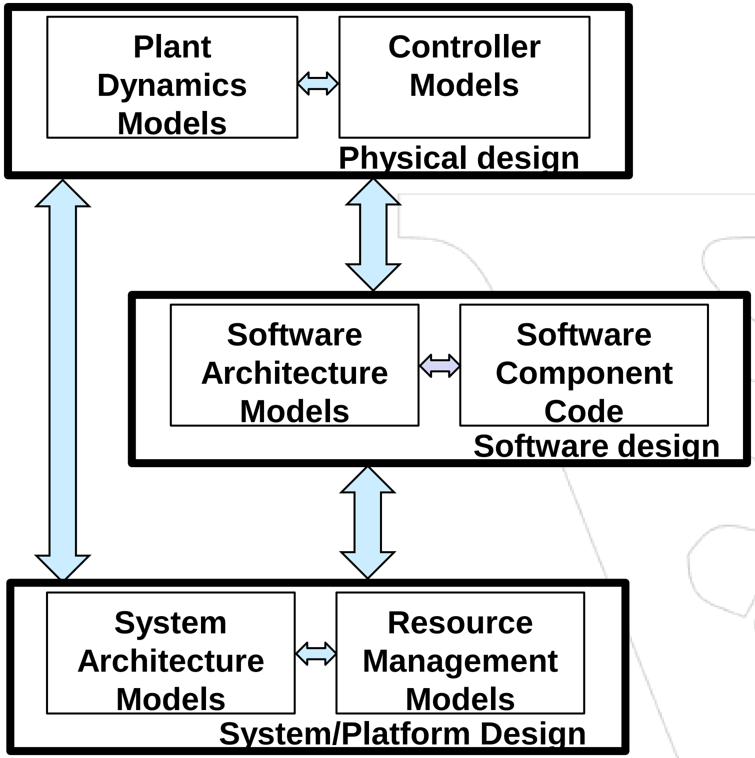
Model-Based Control and Integration of Automotive CPS

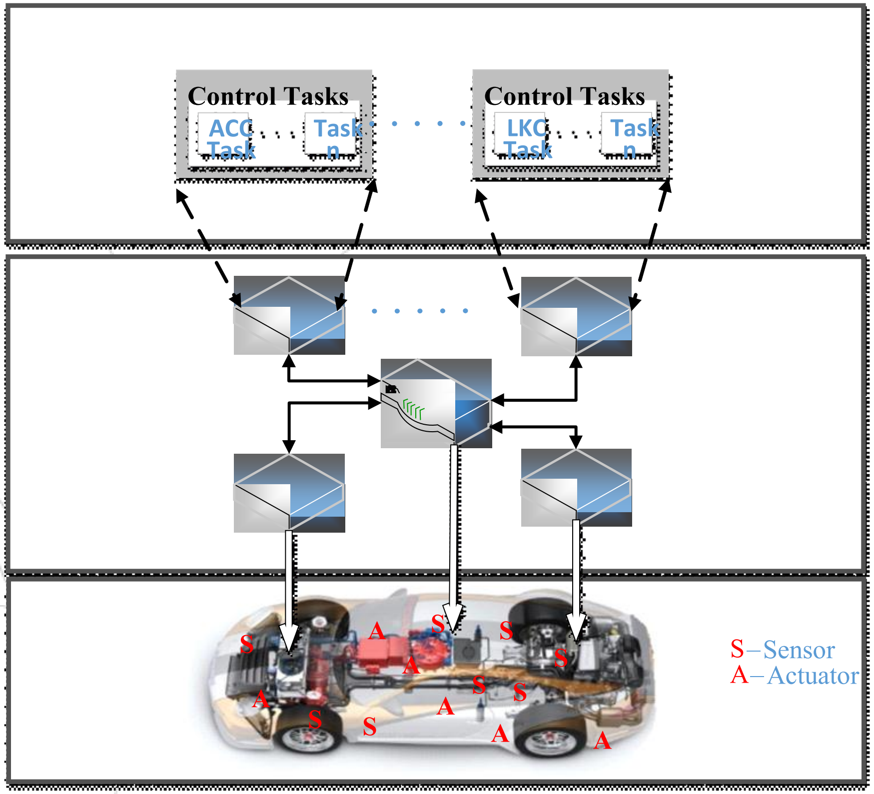
Xenofon Koutsoukos

Emeka Eyisi, Zhenkai Zhang, Di Shang, Joe Porter,
Gabor Karsai, Janos Sztipanovits

SoI Review Meeting
Nashville, TN, April 23, 2013

Control in Automotive CPS





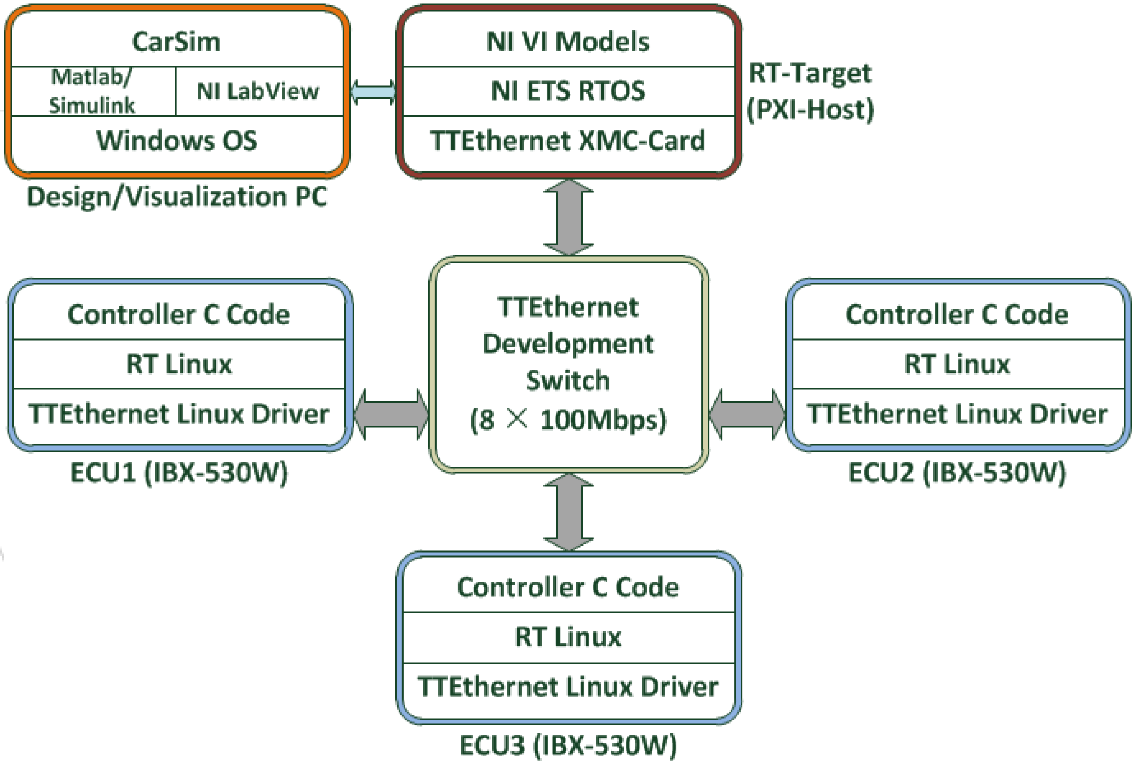
Passivity-based design:
Decouple stability from implementation side effects

Overview

- Hardware-in-the-loop simulation
- Virtual prototyping of time-triggered CPS
- Passivity-based design of adaptive cruise controller
- Model-based control and integration: Adaptive cruise controller and lane keeping controller

Hardware-in-the-Loop Simulation Platform

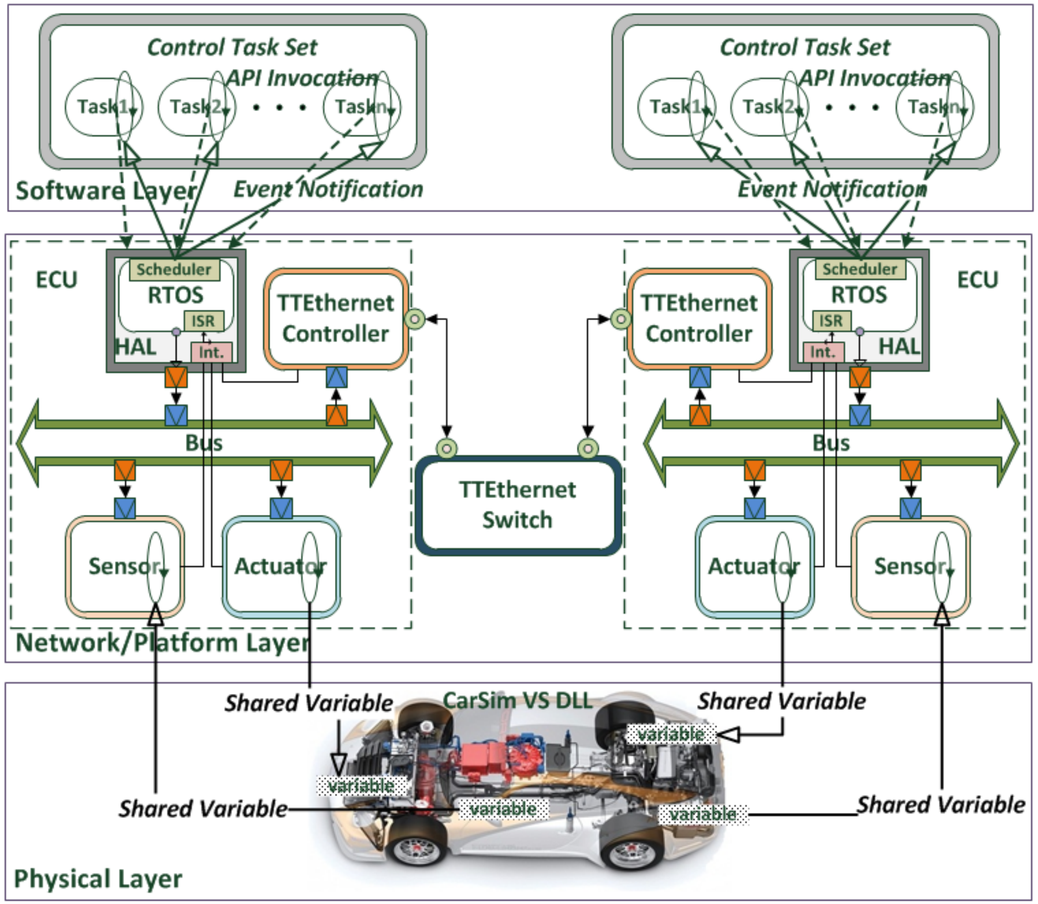
- Design/Visualization PC
  - Vehicle modeling using CarSim
  - Controller design
- RT-Target
  - Represents automotive vehicle
  - CarSim model is deployed via VI models
  - NI ETS 2011 RTOS
  - TTTech PCIe-XMC card
- 8 × 100Mbit/s TTTech TTEthernet Development Switch
- ECU – IBX-530W boxes
  - Controller C code is deployed
  - RT Linux kernel
  - TTEthernet timer driver
  - TTEthernet deriver for Realtek NIC

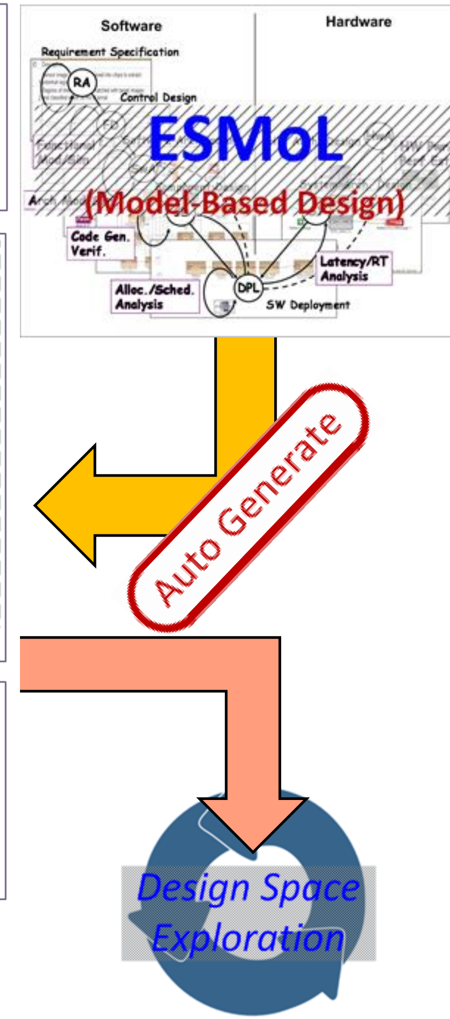


Overview

- Hardware-in-the-loop simulation
- Virtual prototyping of time-triggered CPS
- Passivity-based design of adaptive cruise controller
- Model-based control and integration: Adaptive cruise controller and lane keeping controller

Virtual Platform

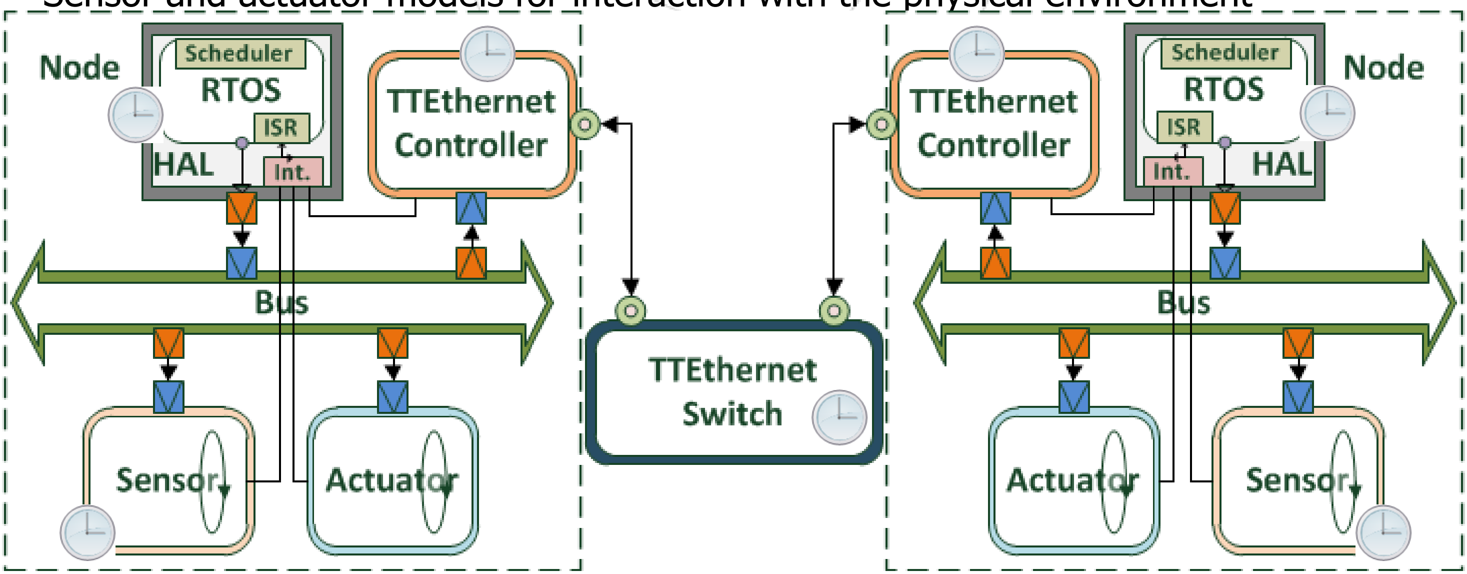




[Zhang et al. 2013] Co-Simulation Framework for Design of Time-Triggered CPS, ICCPS 2013.

Network/Platform Layer

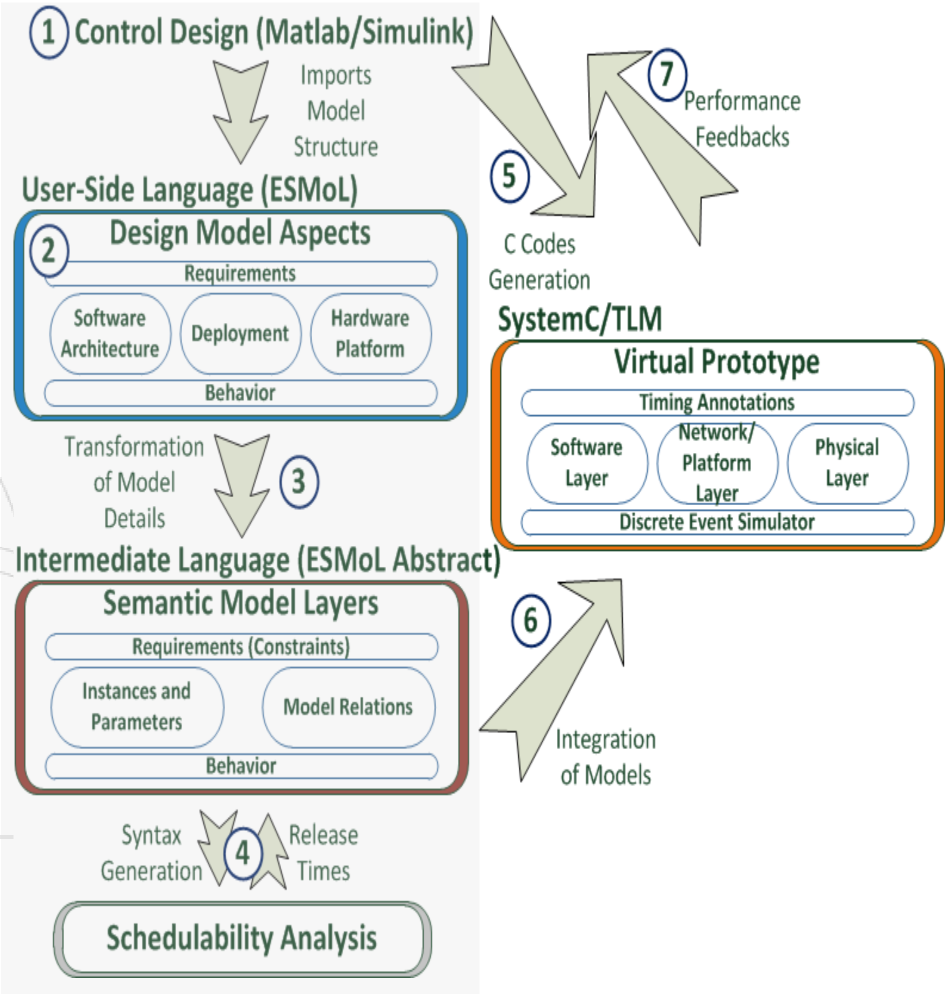
- As the backbone of virtual prototyping of CPS, the network/platform layer bridges the software layer and the physical layer.
- The behavior of this layer is captured by several models in SystemC:
  - A clock model for driving TT operations and synchronization
  - A processing element (PE) model in the form of RTOS model for TT computation
  - A network model compliant with the TTEthernet protocol for TT communication
  - Sensor and actuator models for interaction with the physical environment



[Zhang et al. 2013] Modeling Time-Triggered Ethernet in SystemC/TLM , IESS 2013.

Model-Based Approach

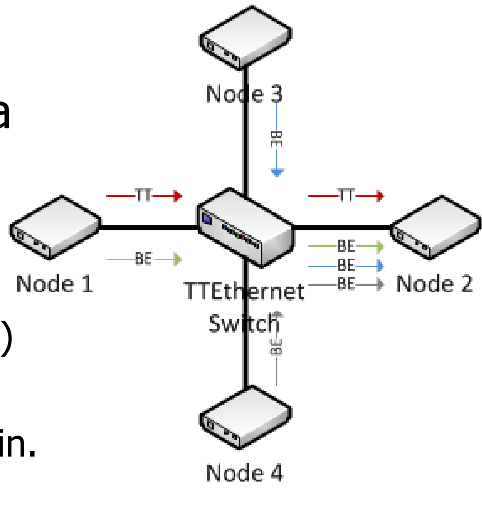
- The first 4 steps corresponds to ESMoL design [Porter et al. 2010].
- From the designed ESMoL model we can generate the executable co-simulation model:
  - Takes C code generated by RTW of MATLAB/Simulink to realize control functionalities.
  - Uses UDM model navigation APIs to traverse the ESMoL_Abstract model
  - Uses Google Ctemplate to fill in the configuration templates
    - A template for PE's task set: each PE's task set is generated as a SystemC module in which tasks are thread processes.
    - A template for sc_main() function in which different parts of different nodes are instantiated and connented.



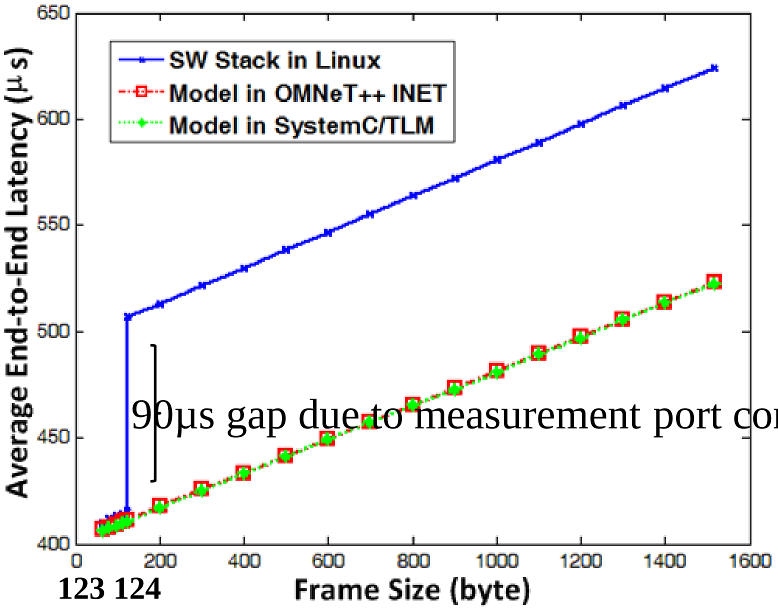
[Porter et al. 2010] The ESMoL Language and Tools for High-Confidence Distributed Control Systems Design. Technical Report, Vanderbilt University, 2010.

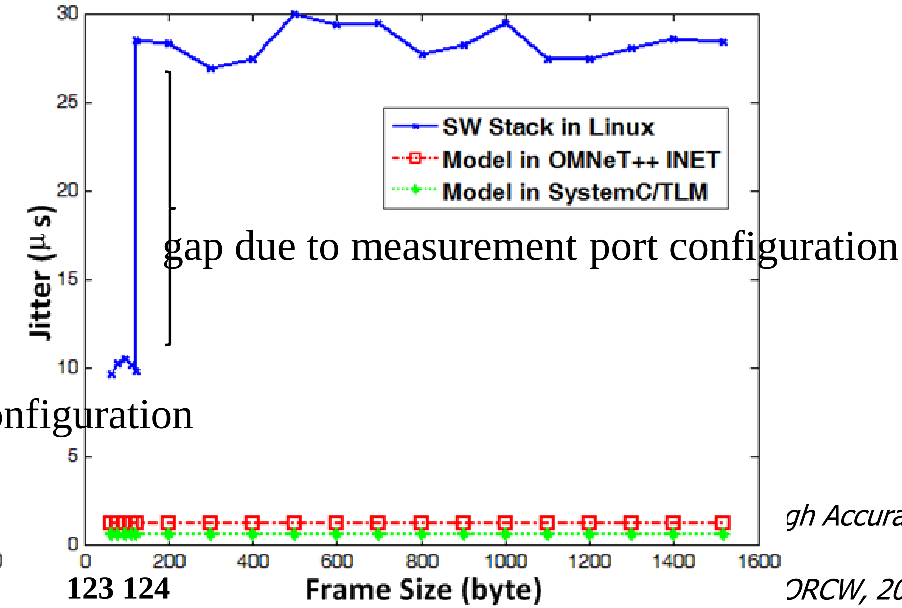
TTEthernet Model Validation

- We set up a star topology having 4 nodes connected to a central switch with 100Mbits/s links.
  - Communication period $T_{comm} = 10\text{ms}$, and time slot $ts = 200\mu s$.
  - Maximum clock drift is 200ppm.
  - Node 1 sends both TT and BE traffic to Node 2. (TT is at 1ms offset)
  - Node 3 and Node 4 send only BE traffic to Node 2.
  - Configuration files are generated by the TTTech TTEthernet toolchain.
    - Switch dispatches TT frame sent by Node 1 at 1.4ms offset.
- The metrics of this validation are



Validation Setup





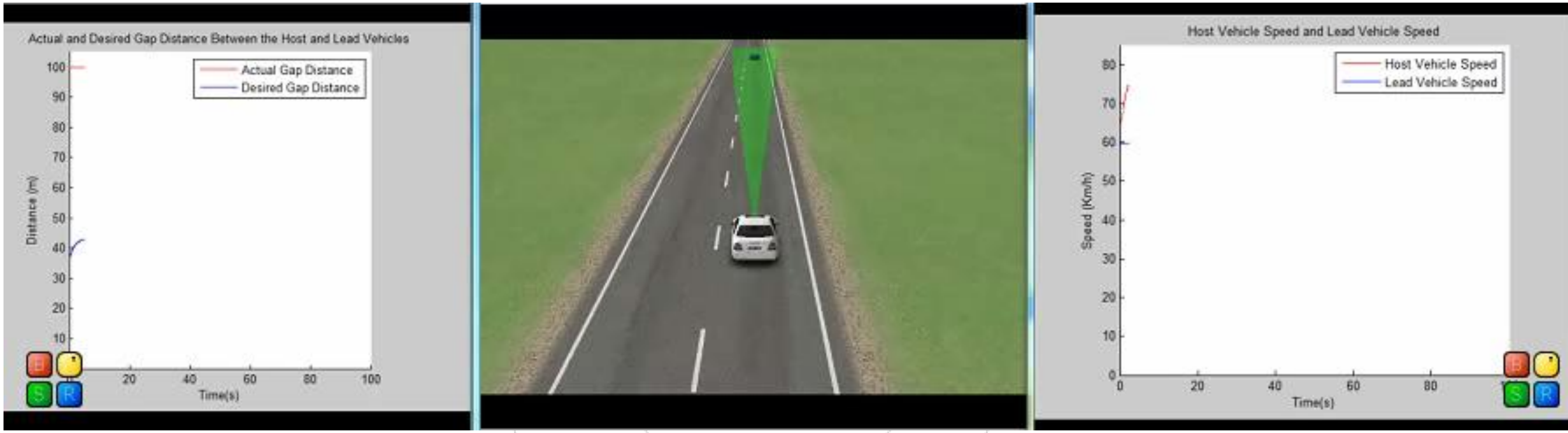
gh Accuracy.

ORCW, 2011

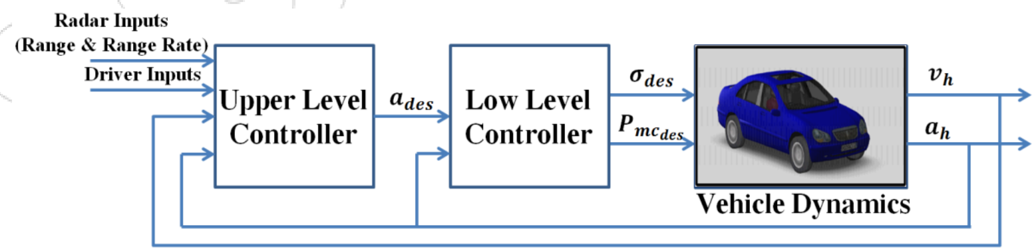
Overview

- Hardware-in-the-loop simulation
- Virtual prototyping of time-triggered CPS
- Passivity-based design of adaptive cruise controller
- Model-based control and integration: Adaptive cruise controller and lane keeping controller

Adaptive Cruise Controller (ACC)



- Longitudinal vehicle dynamics
- Upper level controller
  - Switches between cruise control model and distance spacing mode
- Low level controller
  - Switches between throttle or brake controllers



a_{des} : desired acceleration v_h : host car velocity
σ_{des} : desired throttle a_h : host car acceleration
$P_{mc_{des}}$: desired master cylinder pressure

[Eyisi et al. 2013] Model-Based Control Design and Integration of CPS, JCSE 2013.

Passivity-Based Design (VU/ND)

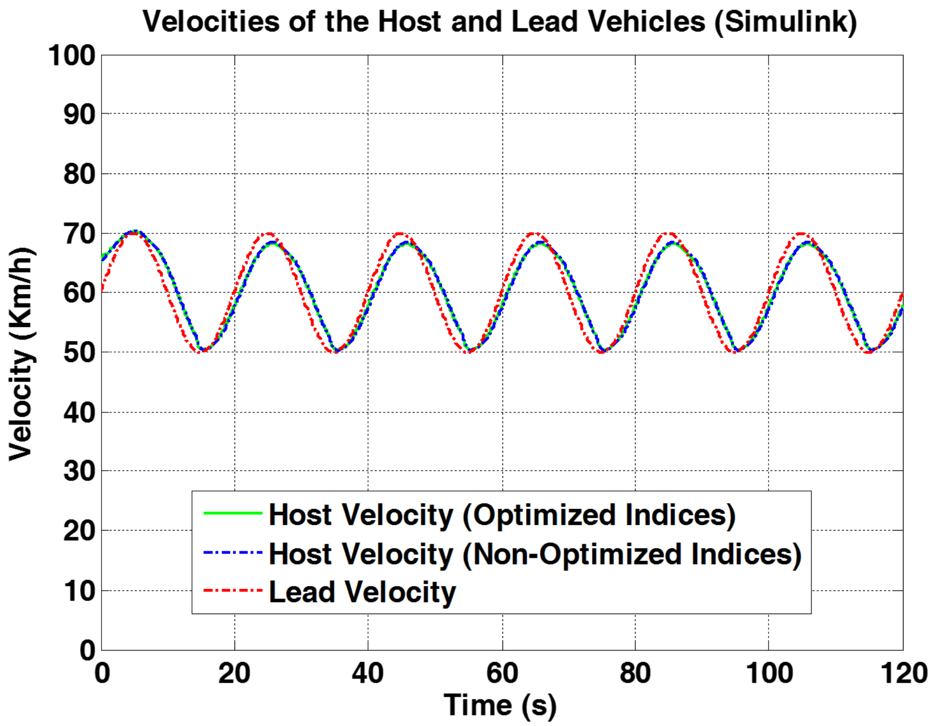
- Passivity Indices
  - Quantifies the level of passivity rather than the typical binary characterization of passive or not passive.
- Application of model-free passivity indices evaluation
- Application of Passivity Indices to ACC
  - Input-Output Mapping (Leading Vehicle Velocity → Host Vehicle Velocity)
  - Focused on the PI throttle controller of the ACC

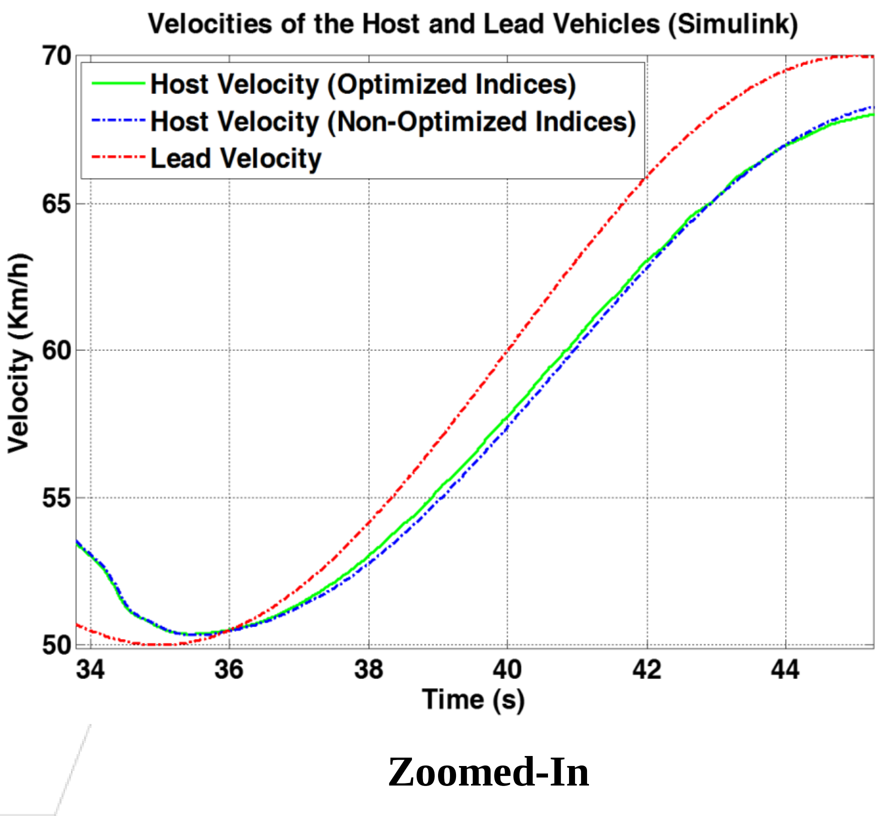
- Throttle Controller PI Gains
  - Non-optimized (Manually tuned gains)
  - Indices optimized Gains
    - Automatically generated using Hookes and Jeeves search
    - Non-optimized gains are used for initial values
    - Varies for each velocity profile

Experiments

- Control gains
  - Manually tuned
  - Optimized passivity indexes
- Dynamic speed profiles
- Performance in the presence of disturbance
  - Nominal
  - 10% increase in Vehicle Mass
  - 25% increase in Vehicle Mass
- Platform
  - Matlab/Simulink
  - Virtual platform
  - Hardware-in-the-loop simulation platform
- Design space exploration (virtual platform)
  - 100Mbit/s TTEthernet
  - 1Gbit/s TTEthernet

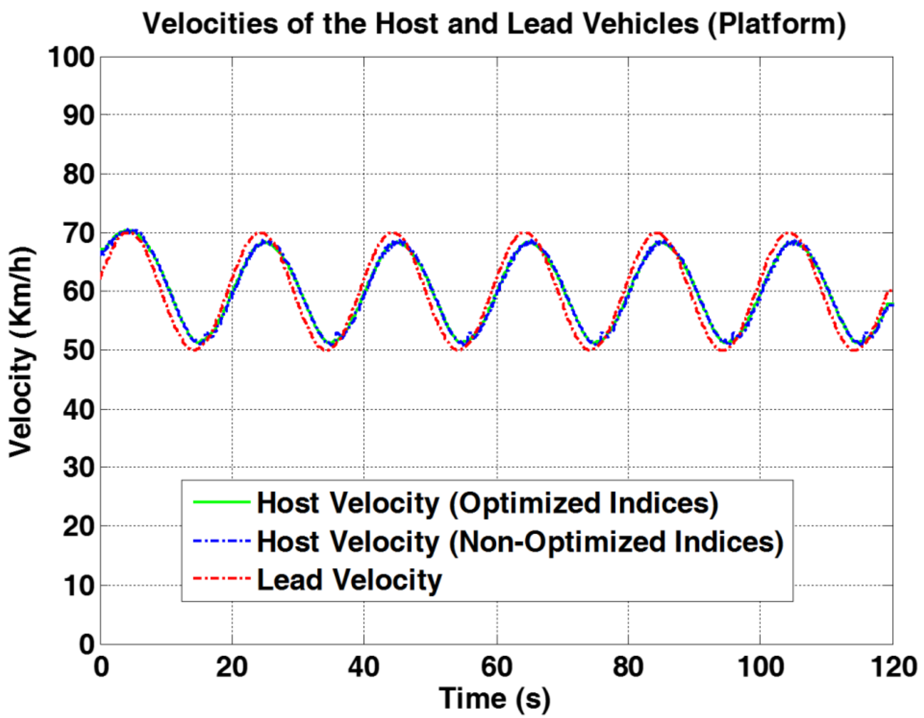
Sinusoidal Lead Velocity Profile (Matlab /Simulink)

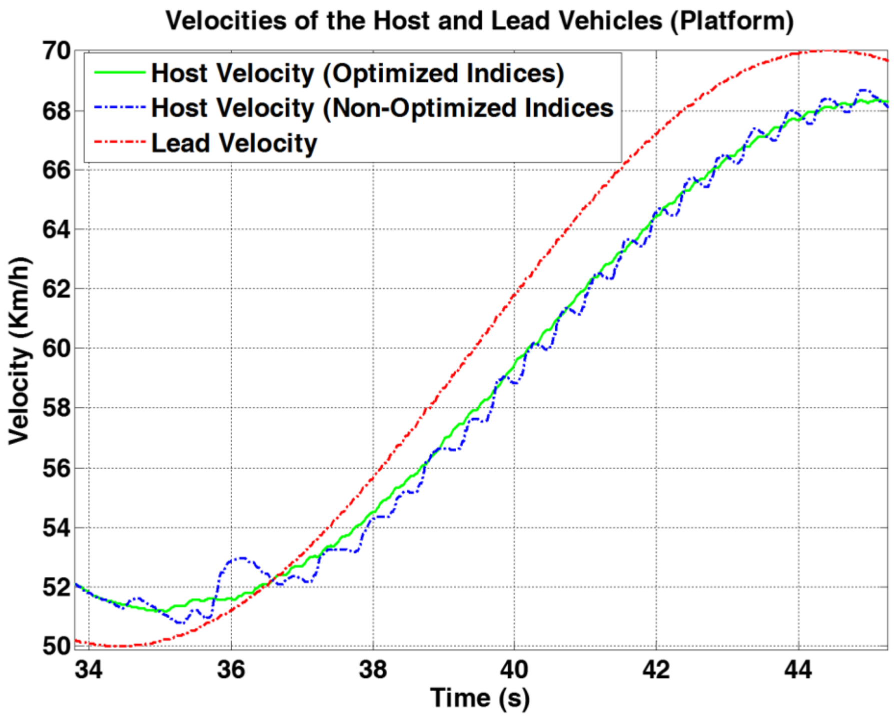




Zoomed-In

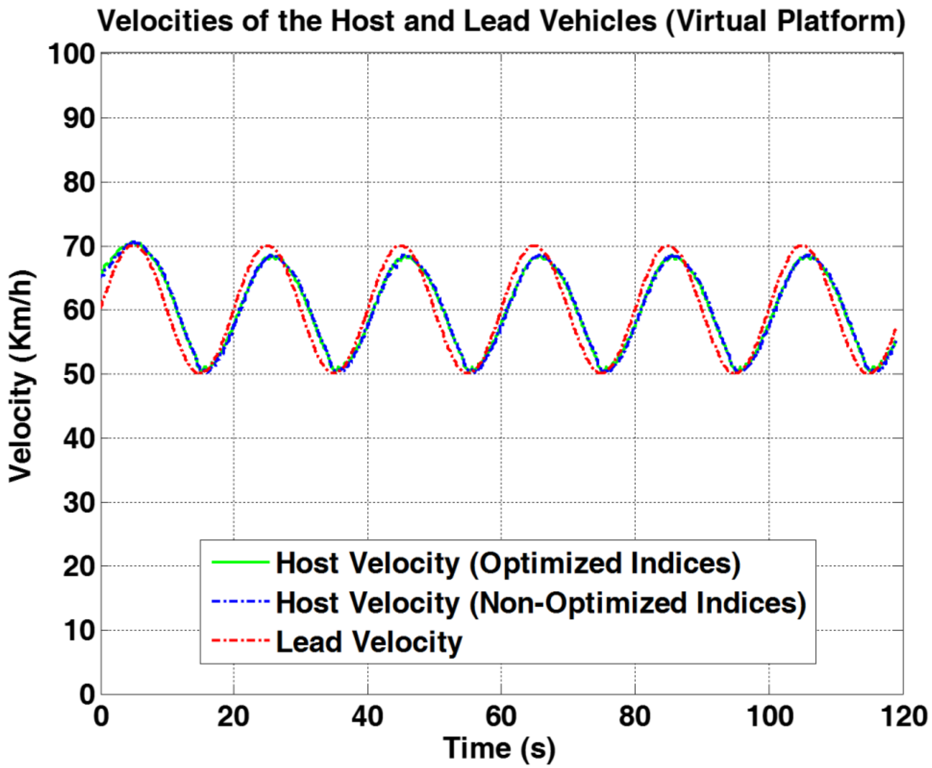
Sinusoidal Lead Velocity Profile (HIL)

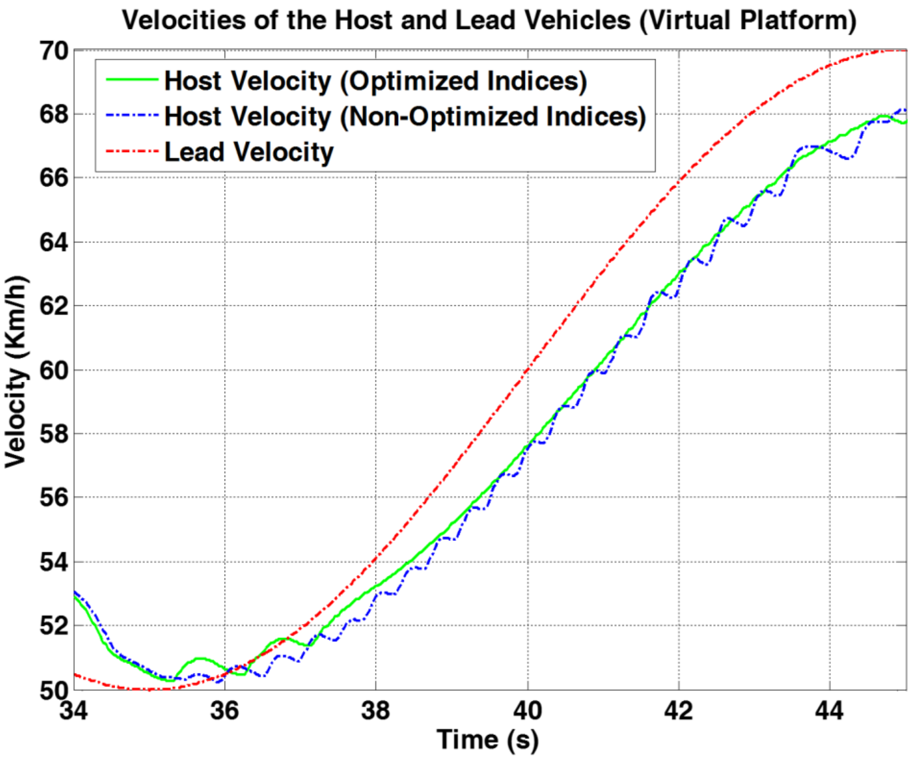




Zoomed-In

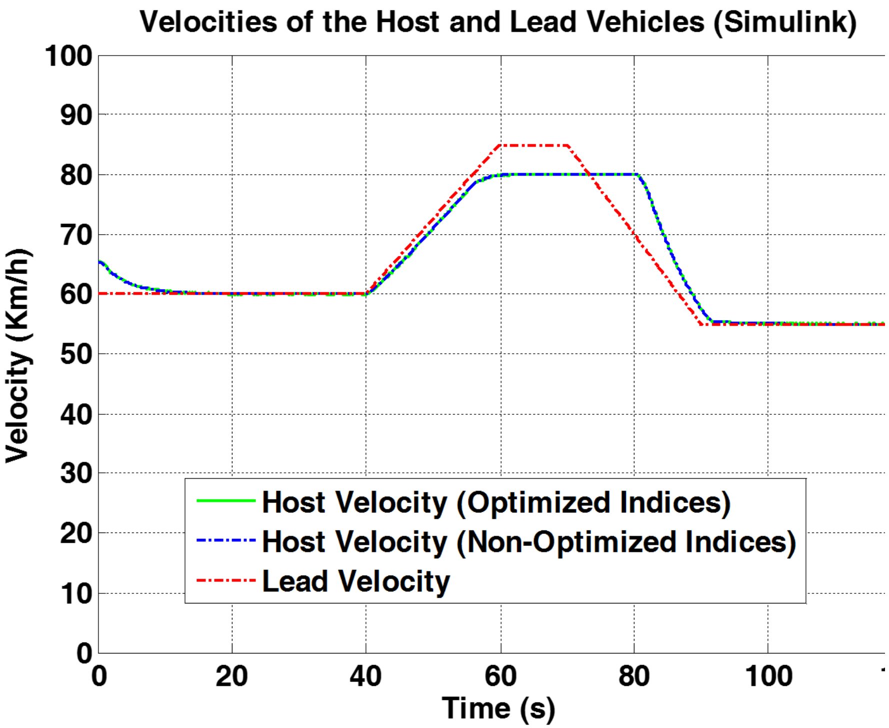
Sinusoidal Lead Velocity Profile (VP)

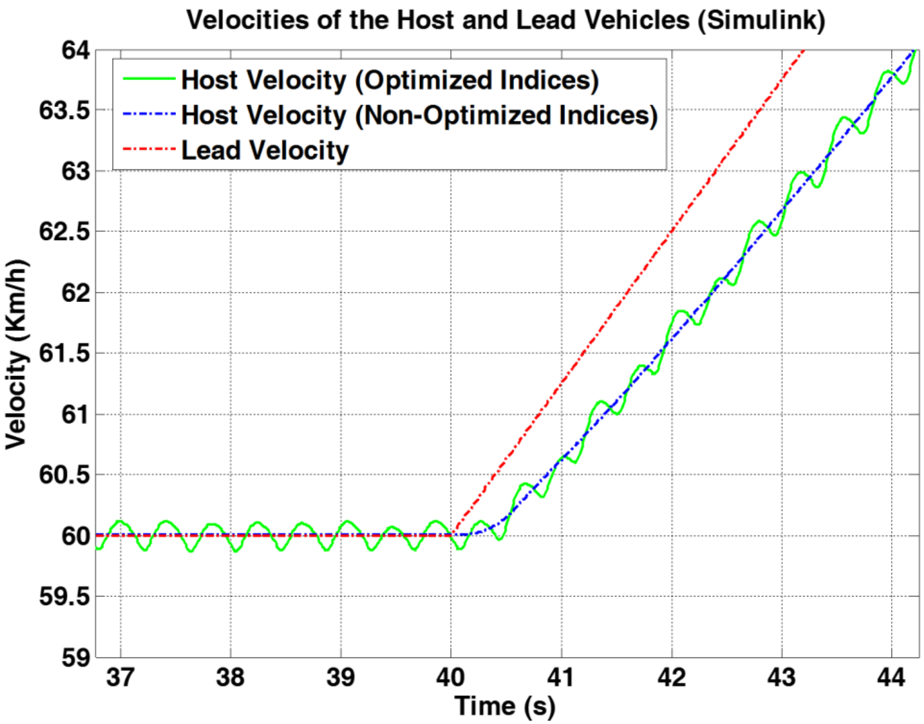




Zoomed-In

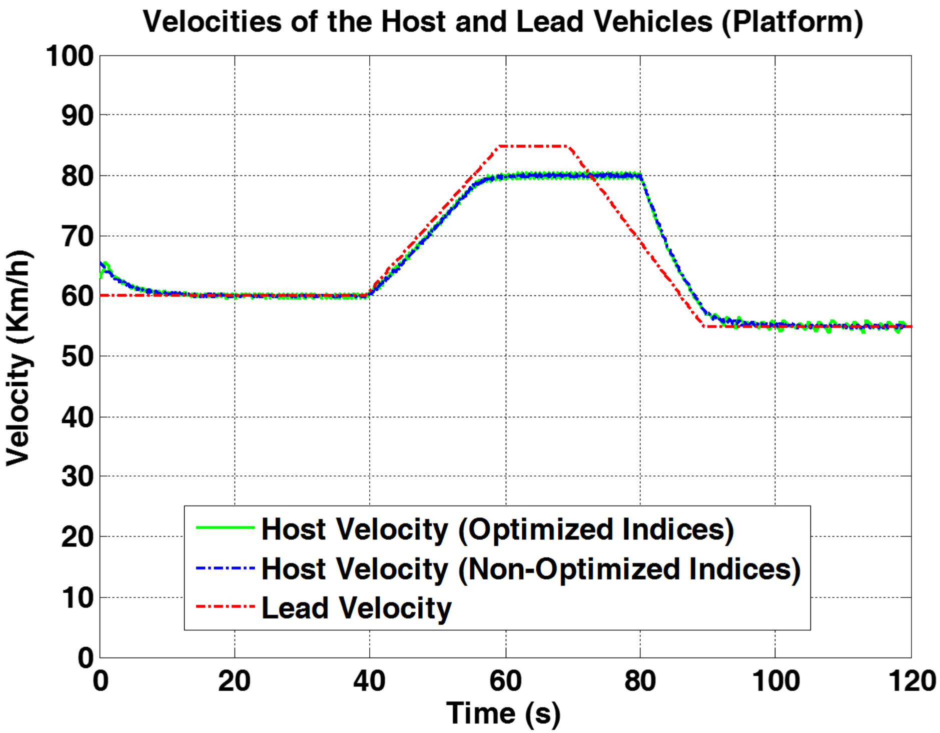
Step-wise Lead Velocity Profile (Matlab /Simulink)

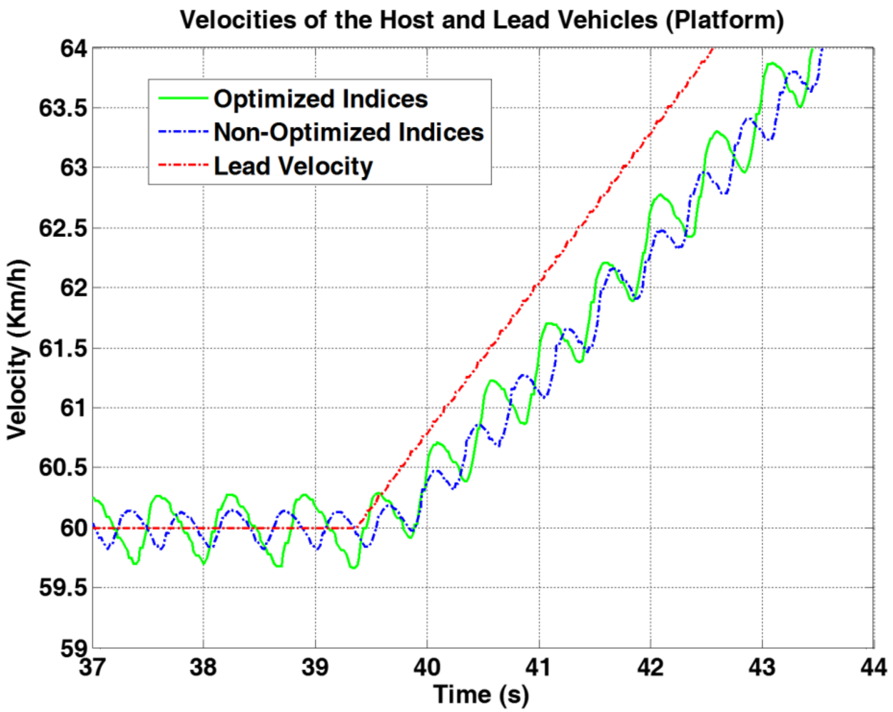




Zoomed-In

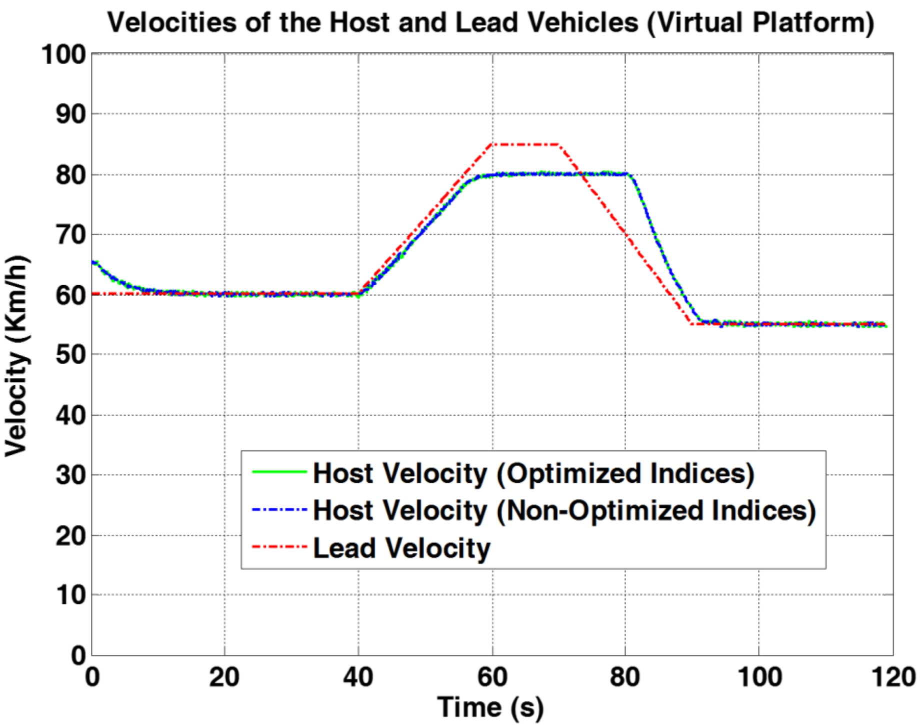
Step-wise Lead Velocity Profile (HIL)

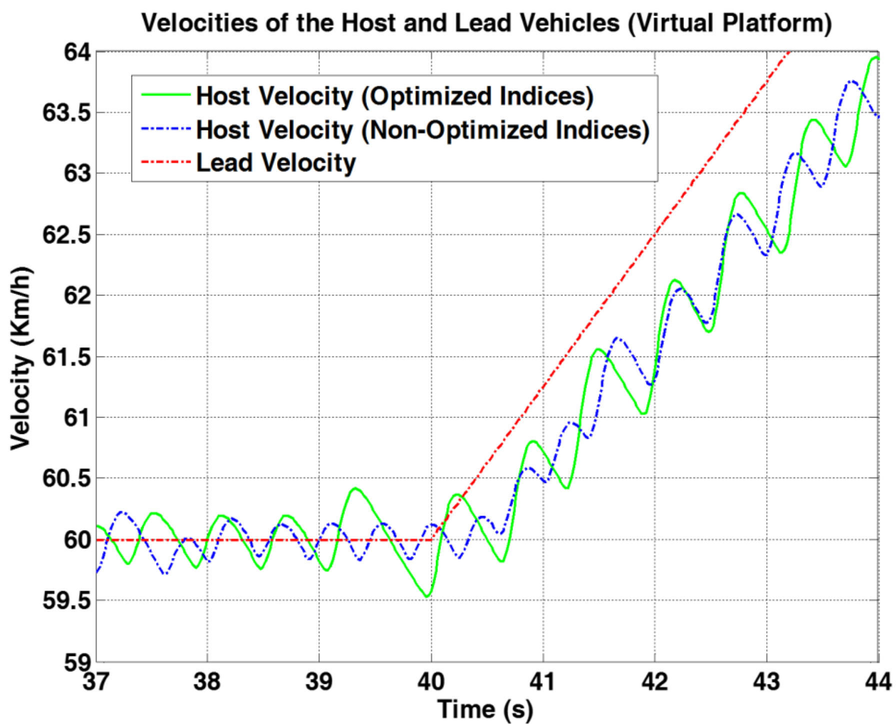




Zoomed-In

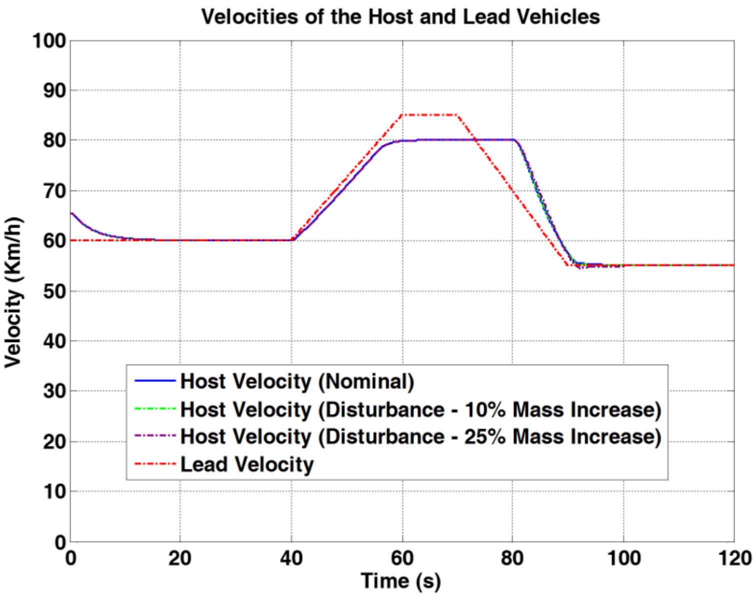
Step-wise Lead Velocity Profile (VP)

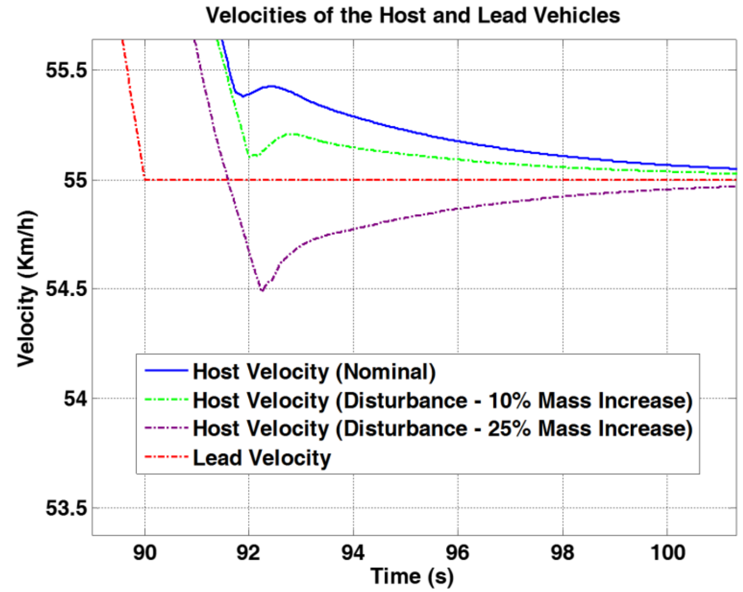


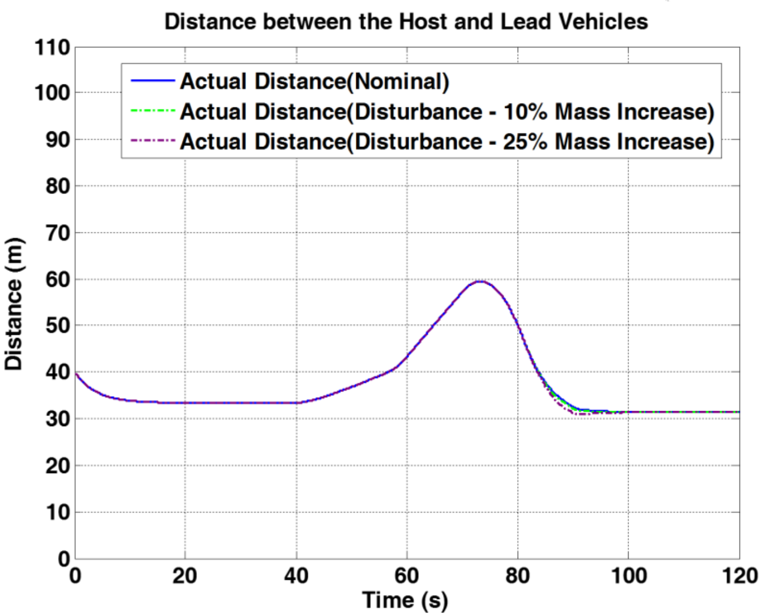


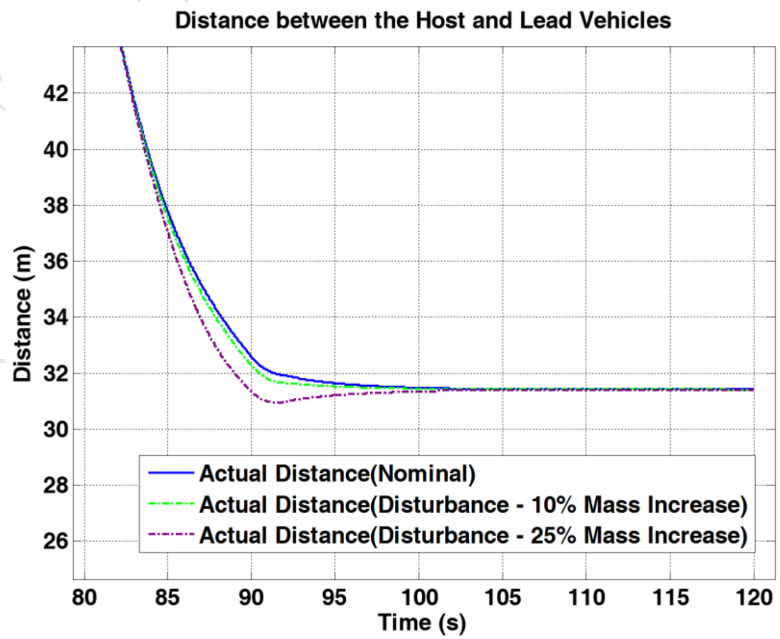
Zoomed-In

Physical Disturbance



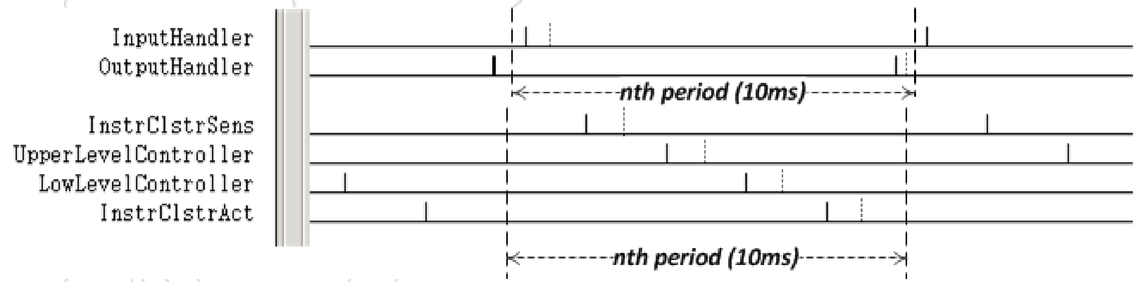


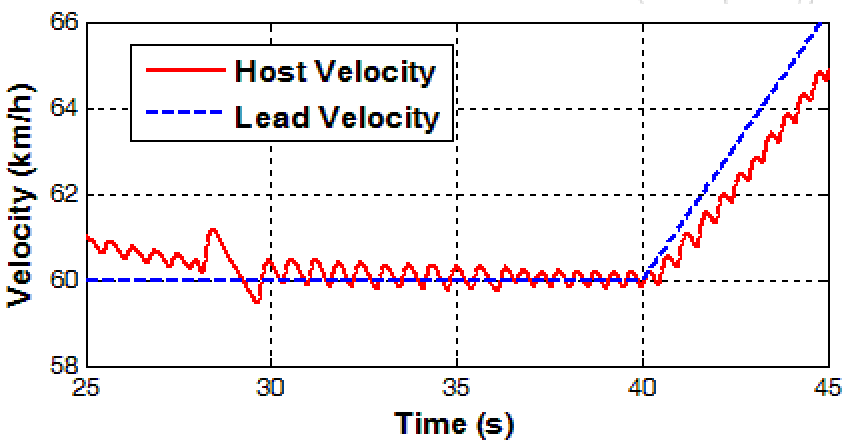




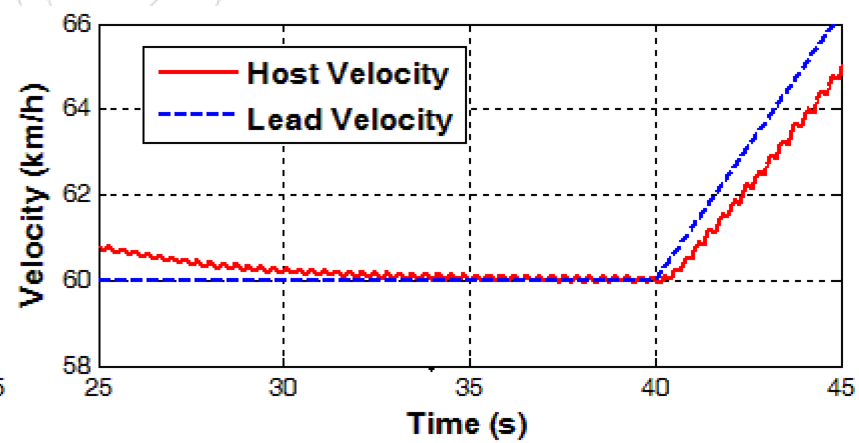
Design Space Exploration

- In order to improve the control performance, we can increase the sampling rate.
- ACC control software on HIL simulator is not computationally intense:
  - InputHandler: 200ns
  - InstrClstrSens: 100ns
  - UpperLevelController: 300ns
  - LowLevelController: 1.7μs
  - InstrClstrAct: 100ns
  - OutputHandler: 200ns





10ms sampling period

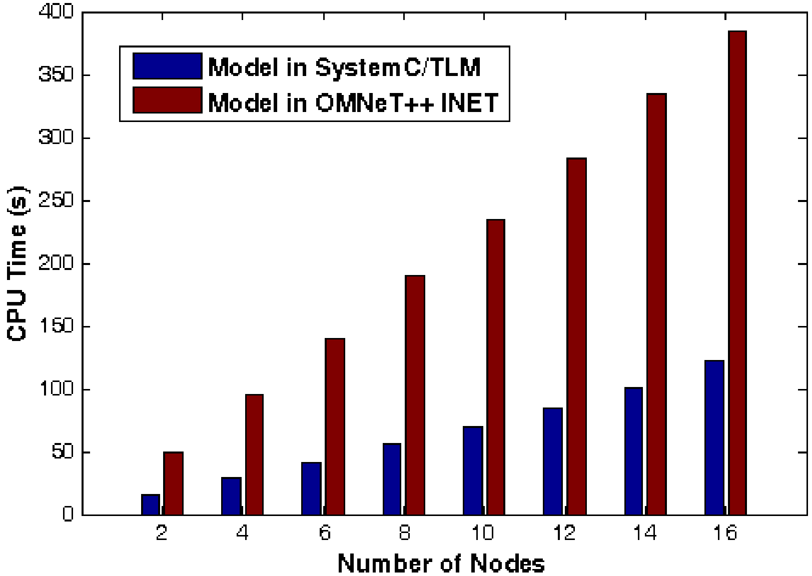


5ms sampling period

21

Simulation Efficiency

- TTEthernet model simulation efficiency:



- Event number of nodes are connected with a central switch.
- Increasingly add a pair of nodes into network
- Each pair of nodes communicates with each other using TT, RC, and BE traffic.
- Each node sends out a TT frame, a RC frame, a BE frame every 10ms.
- 300,000 × #nodes frames totally
- Simulation time is 1000s

- 100s simulation time of ACC under a machine with 3.40GHz and 8GB memory:
  - 102s CPU time for 10ms sampling period
  - 194s CPU time for 5ms sampling period

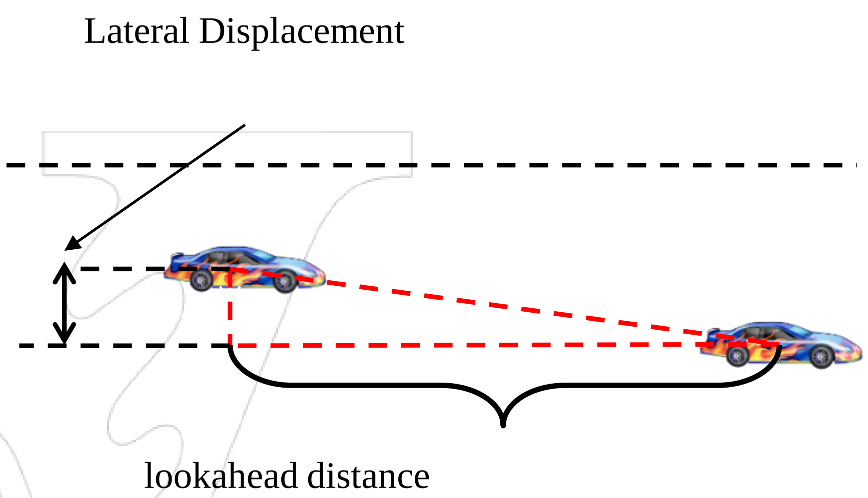
22

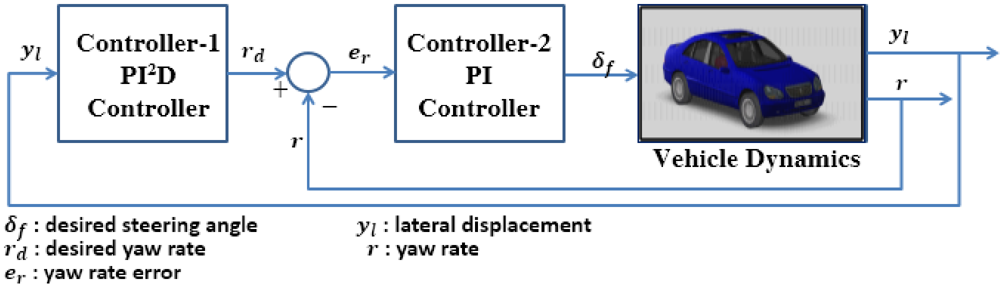
Overview

- Hardware-in-the-loop simulation
- Virtual prototyping of time-triggered CPS
- Passivity-based design of adaptive cruise controller
- Model-based control and integration: Adaptive cruise controller and lane keeping controller

Lane Keeping Controller (LKC)

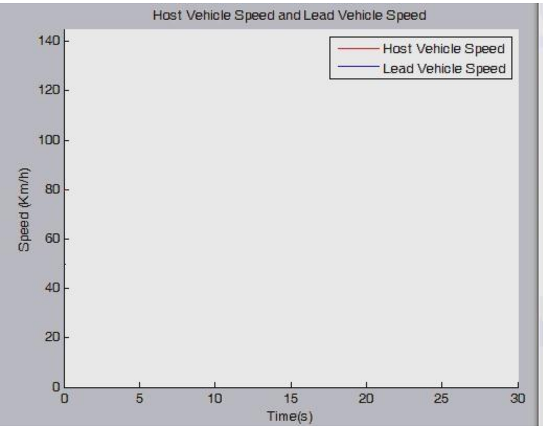
- Lateral vehicle dynamics
- Controller 1: PI^2D
  - Computed desired yaw rate
- Controller 2: PI
  - Computes desired steering angle, δ_f, to achieve zero lateral displacement at a lookahead distance

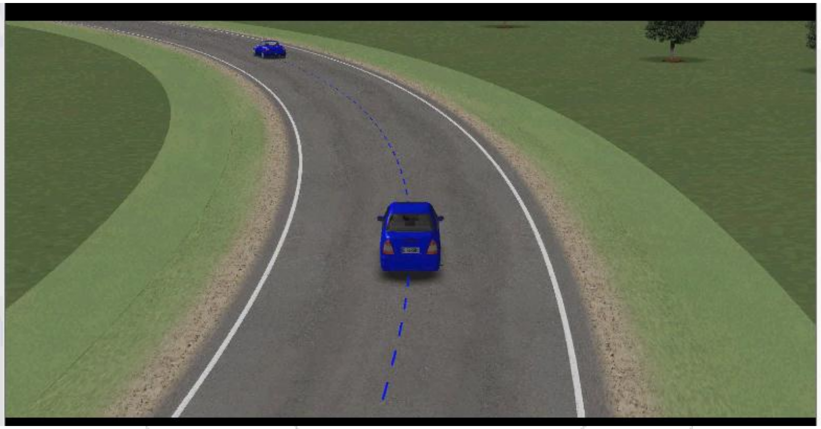


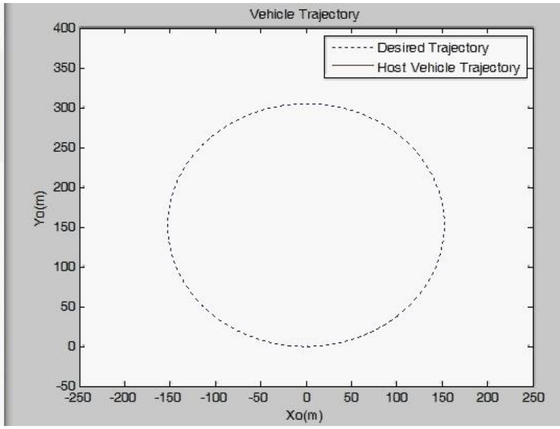


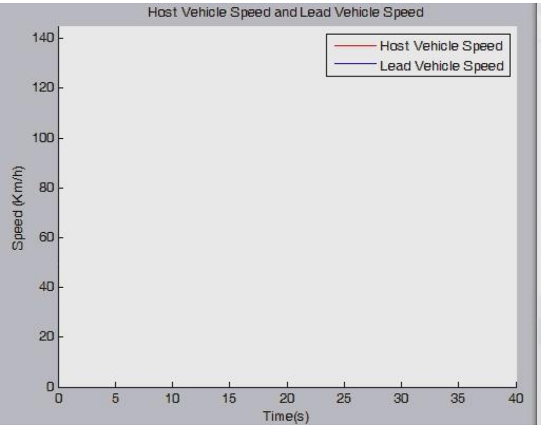
[Shang et al. 2013] ACC and LKC Integration, MED 2013.

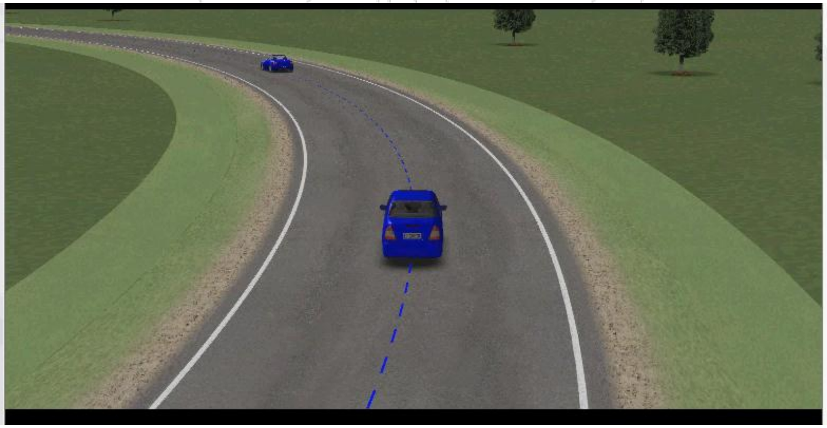
ACC/LKC Interactions

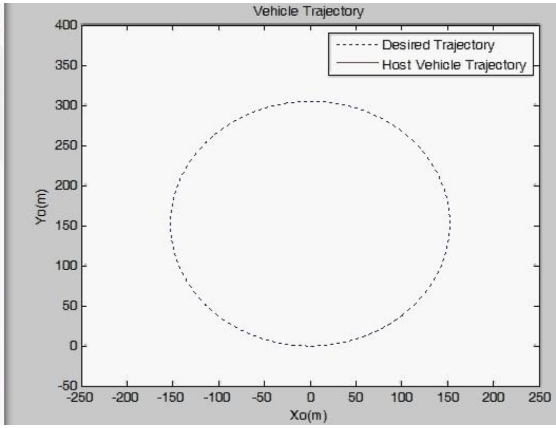




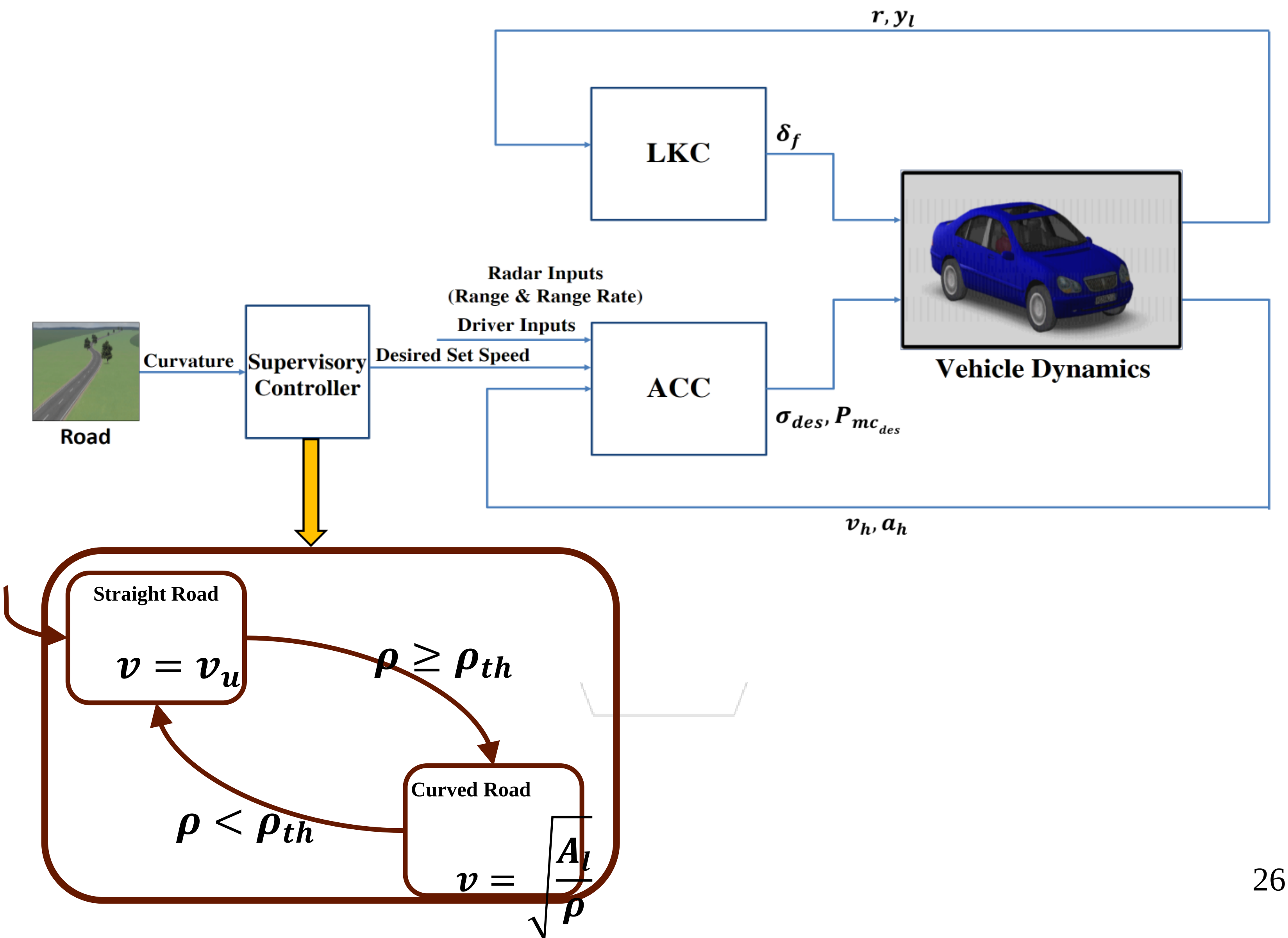




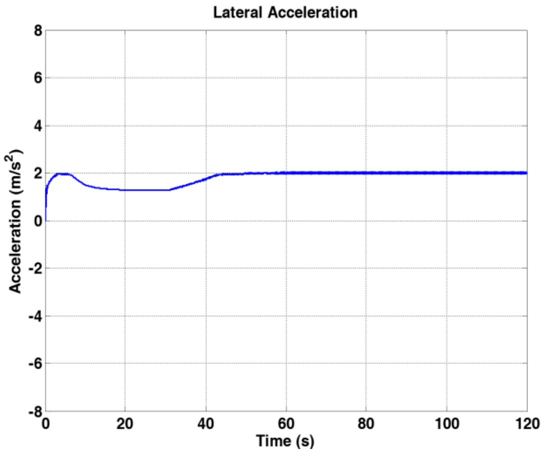


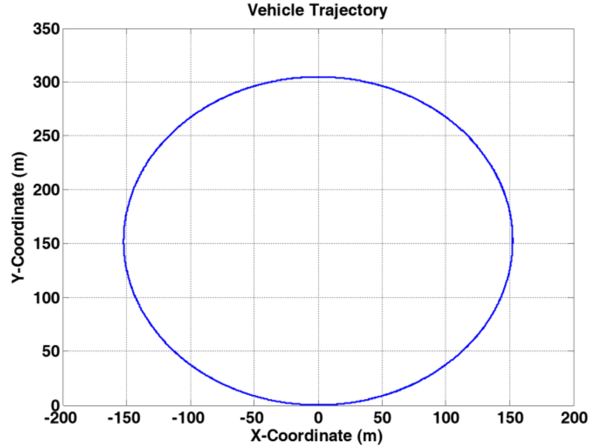


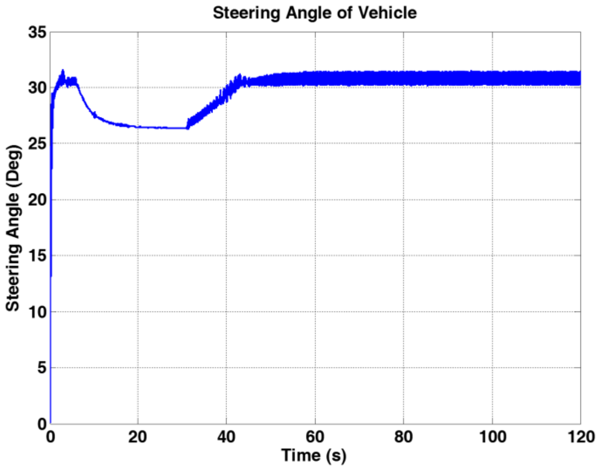
Integrated ACC/LKC Controller

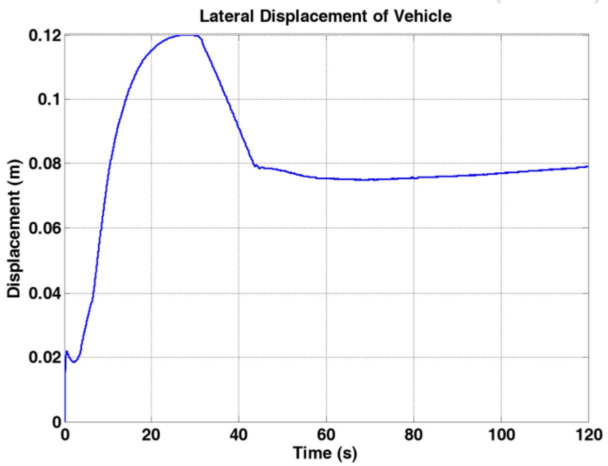


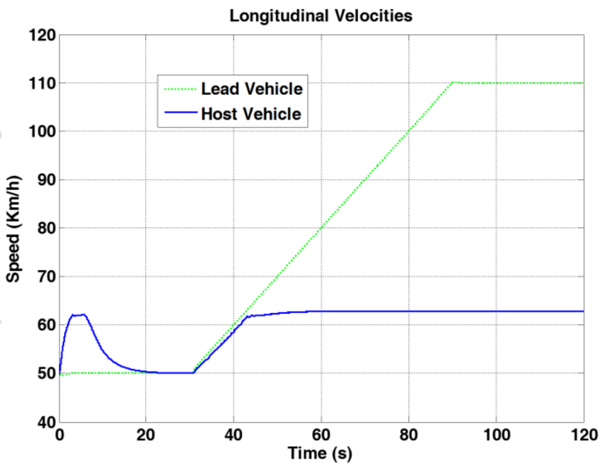
Integrated ACC/LKC Controller



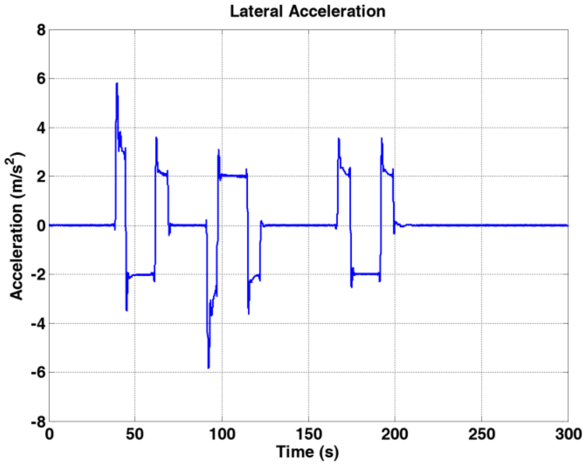


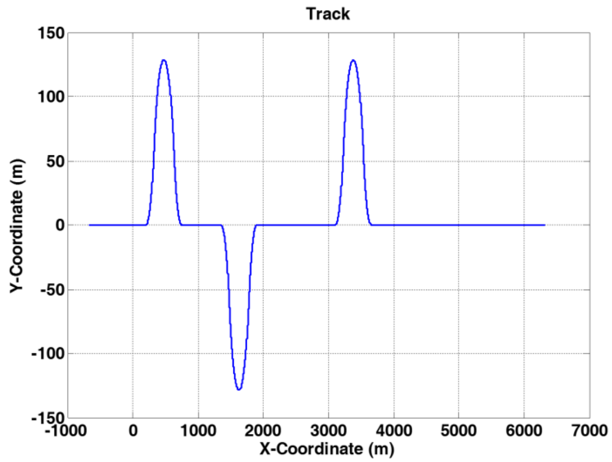


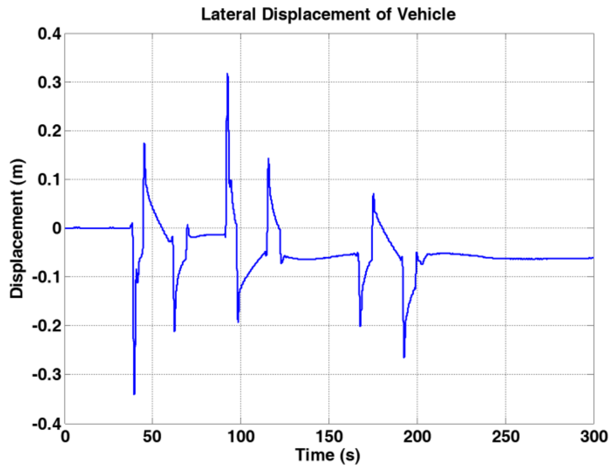


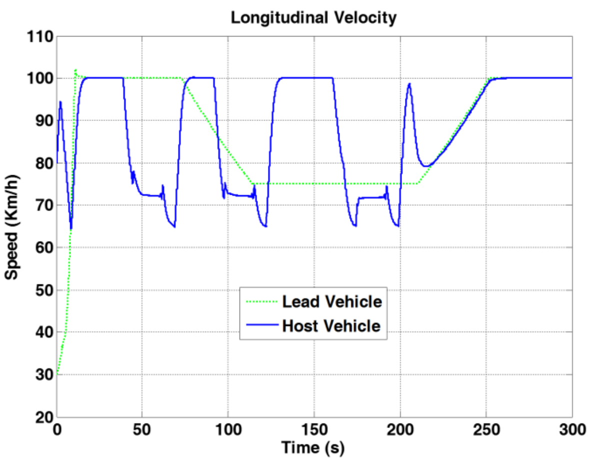


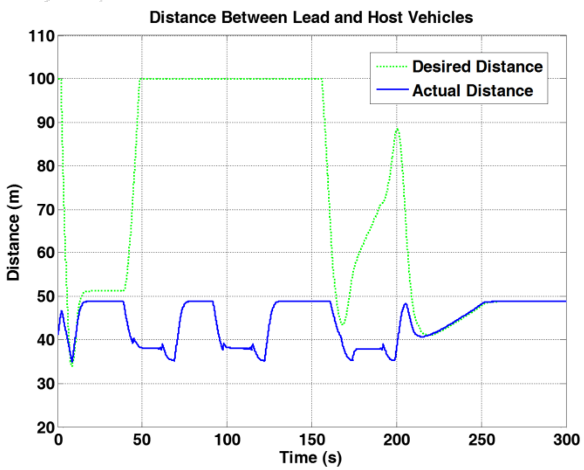
Integrated ACC/LKC Controller











Future Work

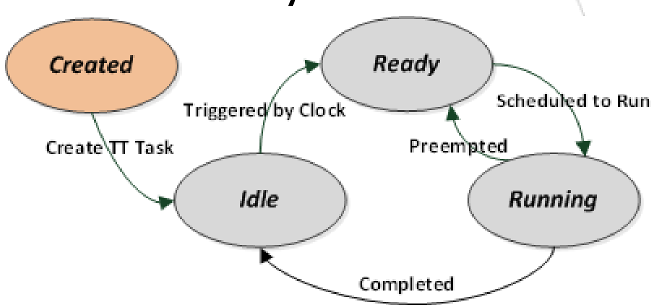
- Passivity-based design
  - Control design using model-free optimization
  - Switching between multiple modes of operation
  - Passivity design for LKC
  - Passivity design for integrated ACC/LKC
- Virtual prototyping of CPS
  - Verification of virtual platform model
  - Design space exploration

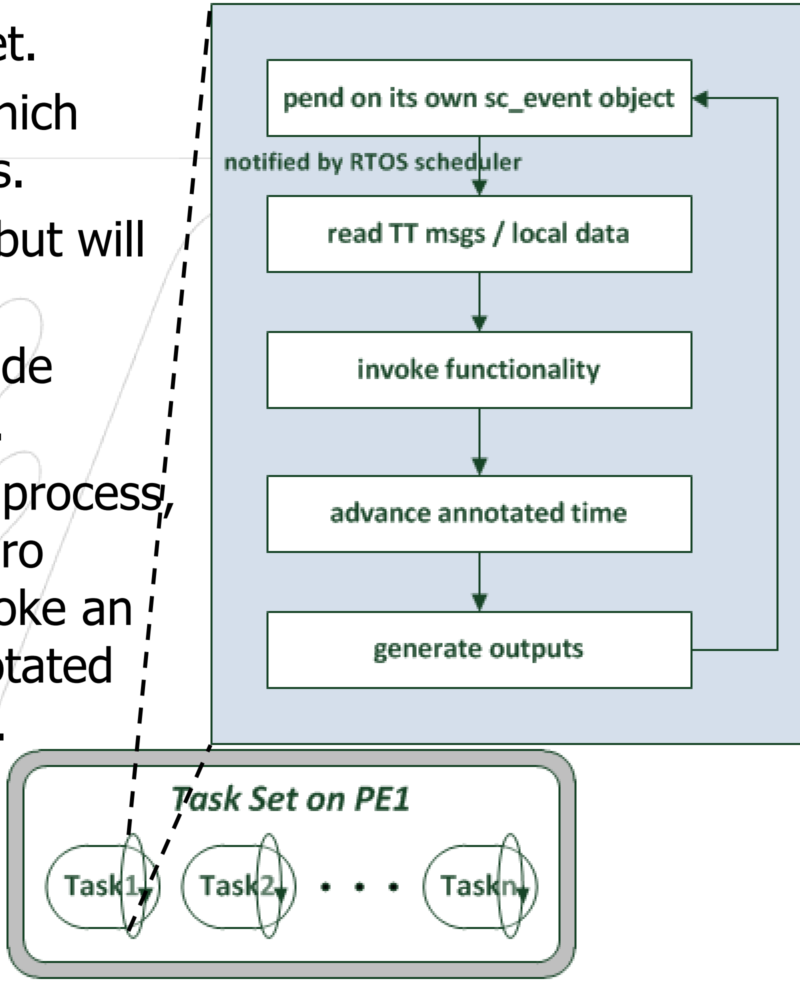
Back-up Slides

Virtual Platform

Software Layer

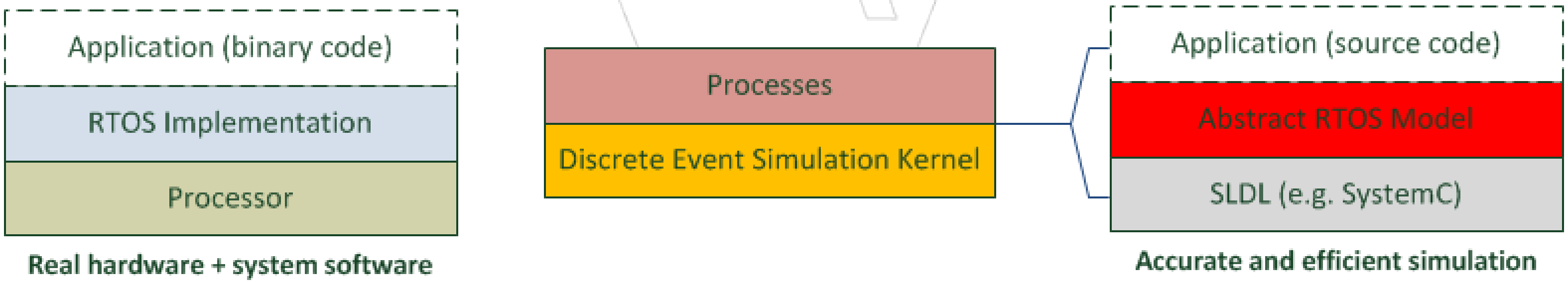
- Each node of the CPS has its own task set.
- Each software component is a TT task which corresponds to a SystemC thread process.
- The processes are concurrent in nature, but will be scheduled to run serially.
- The functionality of each task is the C Code generated from MATLAB/Simulink model.
- Between two synchronization points in a process, the execution of a piece of code takes zero simulation time, so the task needs to invoke an RTOS primitive to delay itself for its annotated execution time before generates outputs.
- A TT task mainly runs in three states:





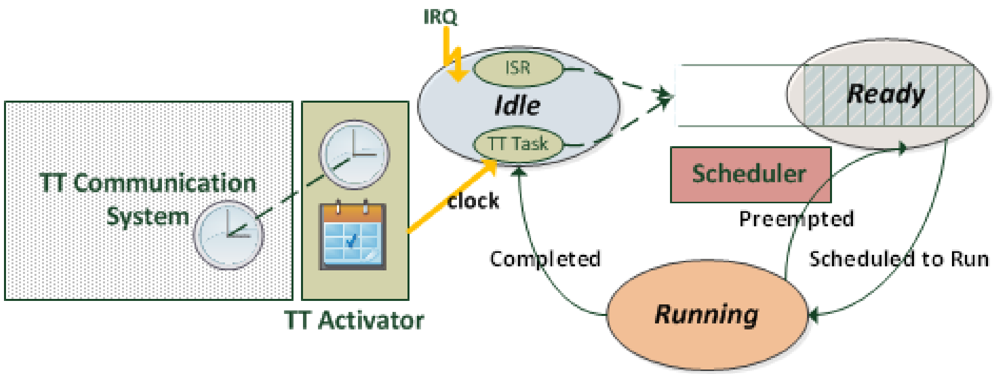
Processing Element Models

- A processing element (PE) corresponds to the underlying computational environment in which the control application software runs.
- In order to simulate the computation efficiently at early stages, we can model the PE at a high level of abstraction – at RTOS level – to take into account the effect of serializing tasks on a processor.
- We use a TT RTOS model that abstracts away the underlying hardware and provides TT computation services to the upper control application.
- The control application tasks, abstract RTOS model and other models will be converted to communicating concurrent processes running on a discrete event simulator.



TT RTOS Model (1/2)

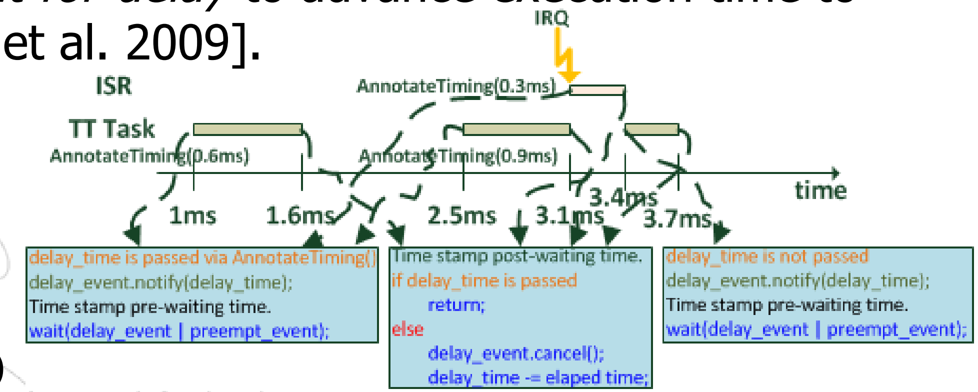
- TT tasks are activated by the TT activator at the predefined times according to an *a priori* schedule table.
- TT activator's clock can be independent or synchronized with TT communication system's clock.
- When activated, a TT task does not run immediately but is put into a ready queue waiting for being scheduled to run.



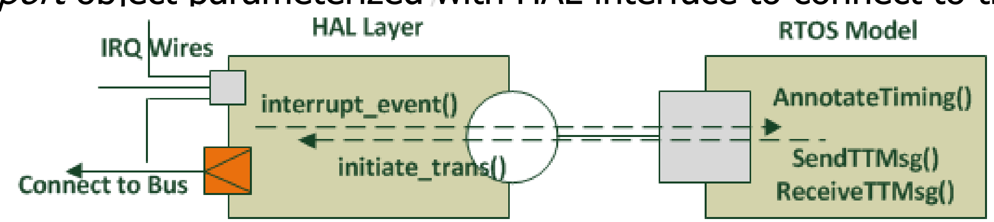
- The scheduler can have a specific scheduling policy to schedule the ready queue, which consists of TT tasks and ISRs.
- This mechanism is useful for the design of mixed time-/event-triggered systems.

TT RTOS Model (2/2)

- Using *wait-for-event* other than *wait-for-delay* to advance execution time to deal with interrupt handling [Zabel et al. 2009].
  - Inter-task communication is achieved by:
    - Shared variables within a PE
    - Message passing between multi-PEs
    - Overwritable and sticky state messages (not consumed by reading)



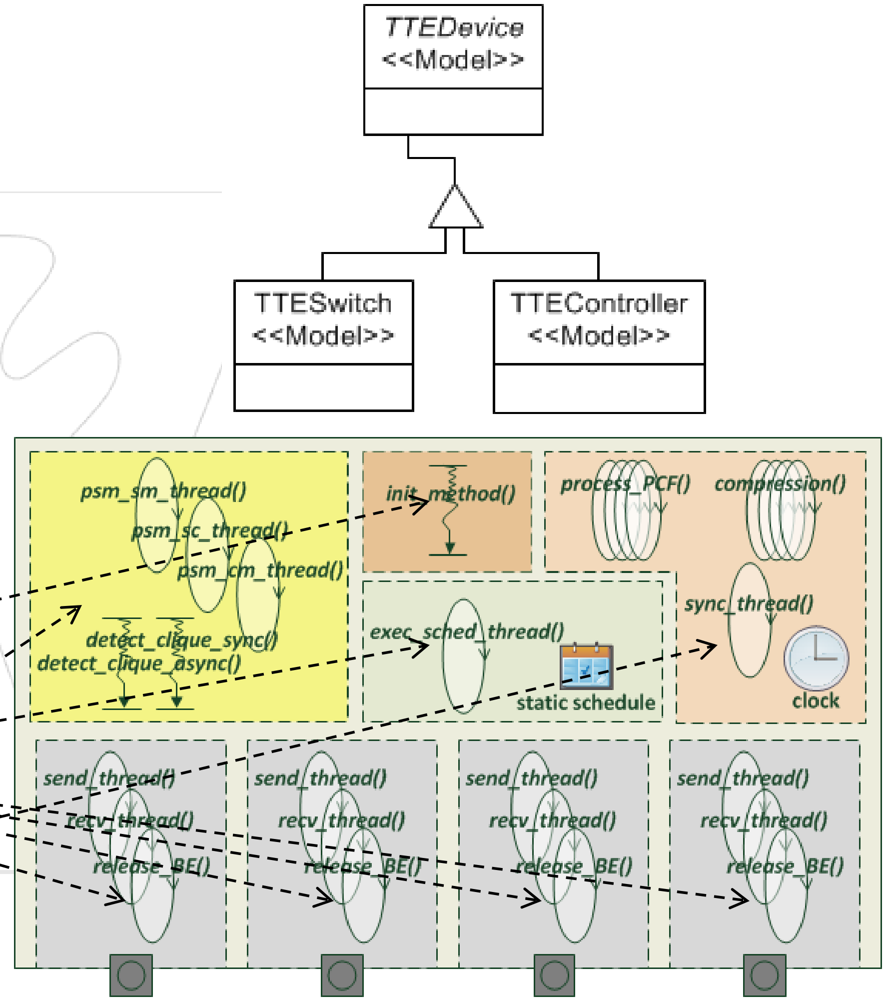
- A HAL (hardware abstraction layer) model is added to wrap the TT RTOS model for PE integration with a bus and other peripherals.
  - Has a multi-port *sc_port* object used to collect all the IRQs of peripherals.
  - Implements the pure virtual functions of a HAL interface (a hierarchical channel).
  - RTOS model has a *sc_port* object parameterized with HAL interface to connect to the HAL layer model.



[Zabel et al. 2009] Accurate RTOS Modeling and Analysis with SystemC. Hardware-dependent Software, Chapter 9, 2009.

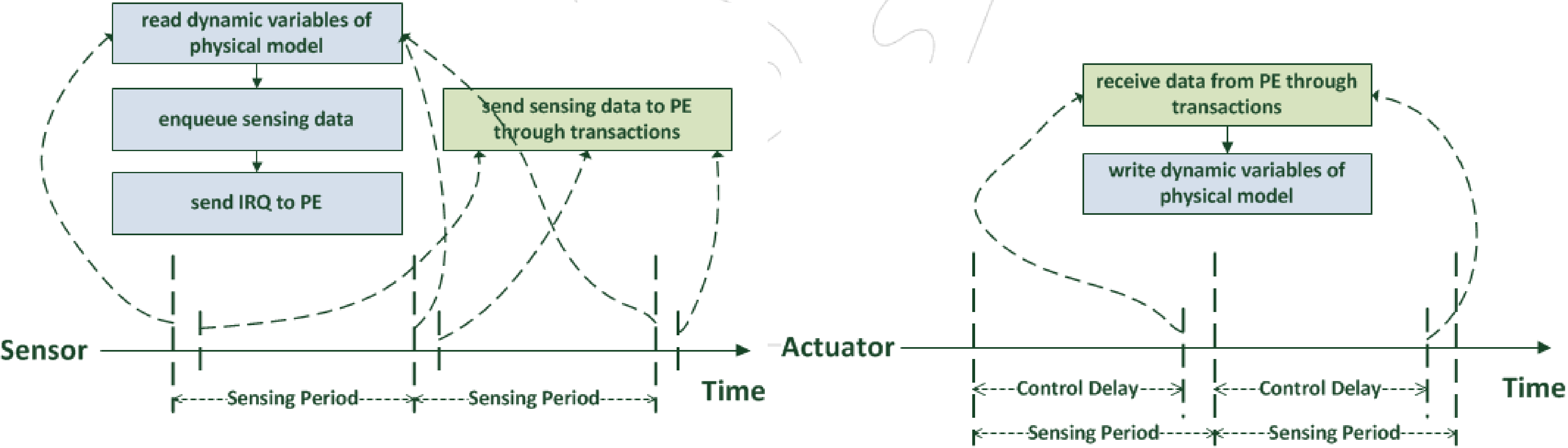
TTEthernet Model

- The network topology is star or cascaded star – switches segment the collision domains:
  - Blocking transport interface of TLM-2.0 is efficient and accurate enough to model the Ethernet frame transmission.
- TTEthernet controller and switch are derived from an abstract base class.
- An abstract *TTEDevice* base module realizes common functions of switch and controller.
  - Initialization
  - Bidirectional ports
  - Scheduler
  - Protocol state machines
  - Synchronization



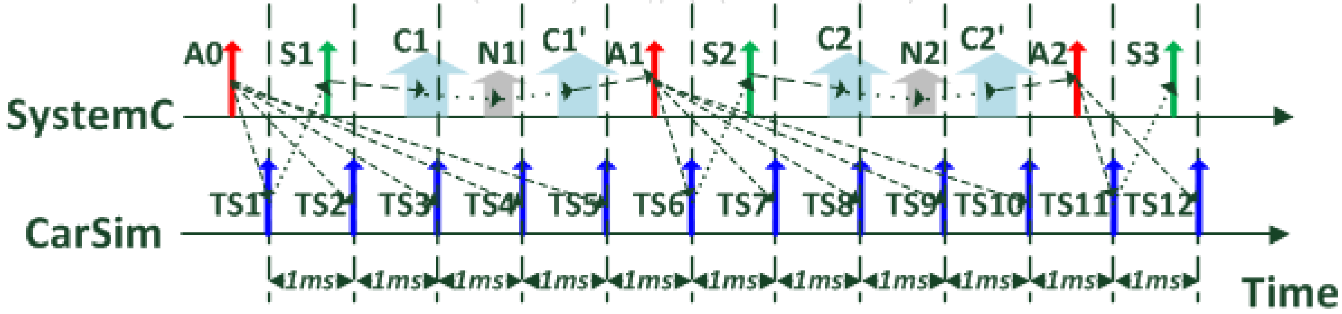
Sensors and Actuators

- The cyber part interacts with the physical part via sensors/actuators.
- Each sensor/actuator is a thread process.
- Each sensor periodically reads data from physical model and generates IRQ to let PE initiate a transaction.
- Each actuator passively receives data from PE periodically or sporadically and writes them to the physical model.

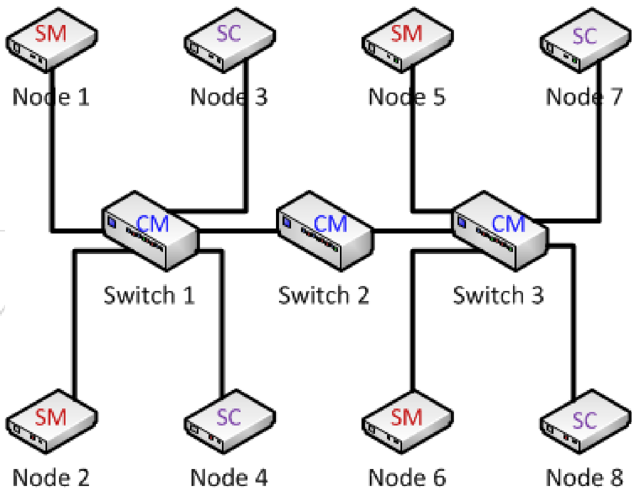


Integration in Time Domain

- SystemC uses discrete event simulator: an event can happen at any time point (the time granularity is small).
- CarSim uses a fixed-step solver: the interval I between two successive mathematical model updates is fixed (e.g. 1ms).
- Sensing period $T_S^S > I$ and control delay $\delta > I$. (not strong in reality)
- After an *Actuation*, next *Sensing* should at least be separated by an interval boundary. (not strong by using TTA)



TTEthernet Model Evaluation



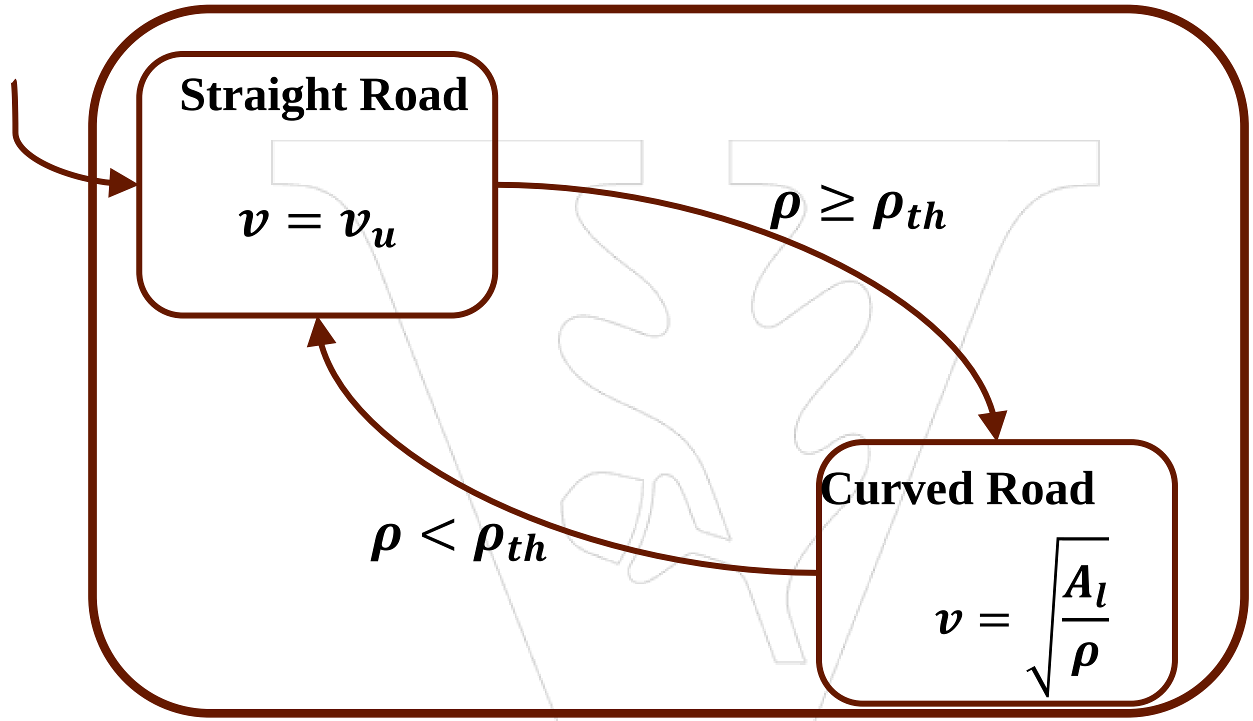
Evaluation Setup

- We set up a cascaded star network with different power-on times to evaluate the model's synchronization services.
  - Node 1, 2, 5, and 6 are SMs.
  - Switch 1, 2, and 3 are CMs.
  - Node 3, 4, 7, and 8 are SCs.
  - The integration cycle is 10ms.
  - Configuration files are generated by the TTTech toolchain.
- Two-step synchronization mechanism:

N1 & N2 & N5 & N6	SW1 & SW2 & SW3	Sync	Resync
0s/0s/0s/0s	0s/0s/0s	29.834ms	-
0.1ms/1ms/0.5ms/1.2ms	1.1ms/0.8ms/1.5ms	30.845ms	-
2ms/4ms/8ms/6ms	30ms/10ms/40ms	79.856ms	-
0s/0s/0s/0s	0s/30ms/0s	38.677ms	-
0s/0s/0s/0s	0s/50s/0s	29.776ms	50.0256s

ACC/LKC

Supervisory Controller



A_l: Max Desired Lateral Acceleration

v_u : User Set Speed

ρ : Curvature (1/Curve radius)

v : Desired Set speed