A case for heterogenous co-simulation of cooperative and autonomous driving

Tobias Hardes[*†‡], Ion Turcanu[§], and Christoph Sommer[*]

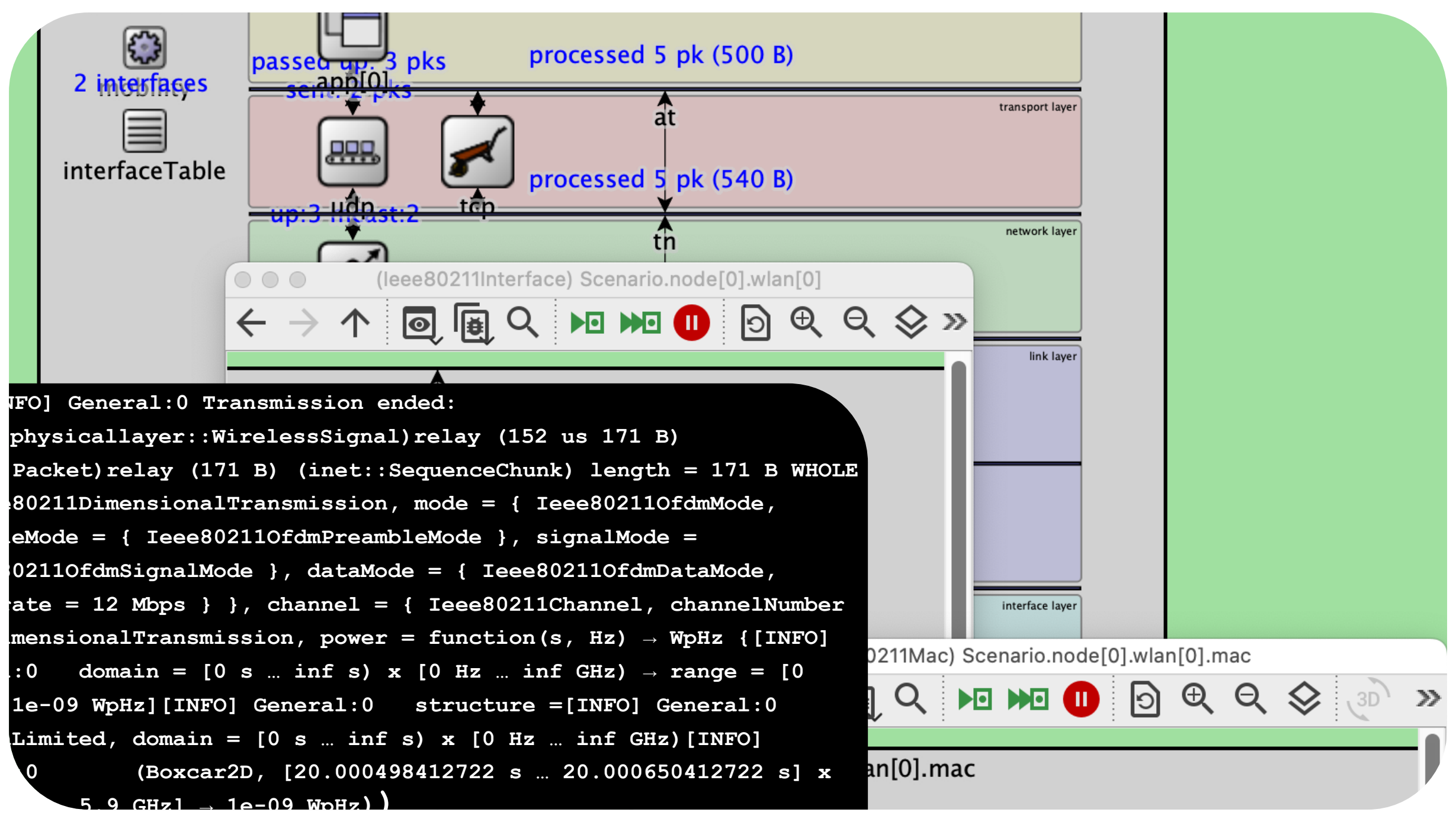
[*]TU Dresden, Faculty of Computer Science, Germany
[†]Paderborn University, Dept. of Computer Science, Germany
[‡]Software Innovation Campus Paderborn, Germany
[§]Luxembourg Institute of Science and Technology (LIST), Luxembourg

Abstract

Many exciting future research topics in the field of Cooperative Autonomous Vehicles (CAVs) require the simulation of both connectivity and automation components. However, existing simulation tools focus on only one of these two aspects while making idealistic assumptions about the other. In this work, we motivate the use of established libraries such as gRPC to couple existing independent simulation tools tailored to either connectivity or automation, and demonstrate the feasibility of such an approach. We also describe an Open Source reference implementation coupling CARLA and Veins.

Motivation

"Networks" Community



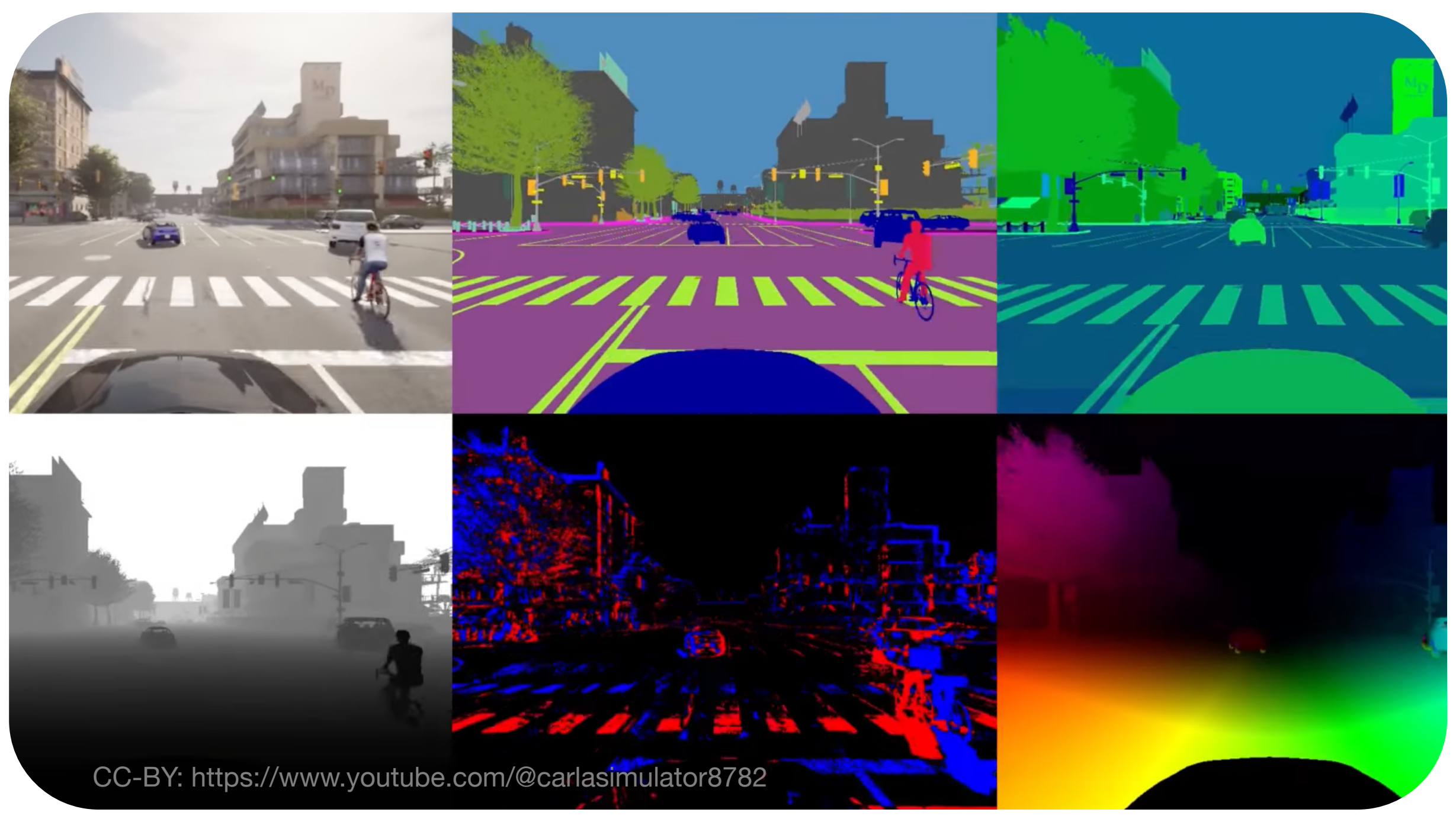
```
NFO] General:0 Transmission ended:
physicallayer::WirelessSignal)relay (152 us 171 B)
Packet)relay (171 B) (inet::SequenceChunk) length = 171 B WHOLE
80211DimensionalTransmission, mode = { Ieee80211OfdmMode,
eMode = { Ieee80211OfdmPreambleMode }, signalMode =
02110fdmSignalMode }, dataMode = { Ieee80211OfdmDataMode,
ate = 12 Mbps } }, channel = { Ieee80211Channel, channelNumber
mensionalTransmission, power = function(s, Hz) → WpHz {[INFO]
:0   domain = [0 s … inf s) x [0 Hz … inf GHz) → range = [0
1e-09 WpHz][INFO] General:0   structure =[INFO] General:0
Limited, domain = [0 s … inf s) x [0 Hz … inf GHz)[INFO]
0        (Boxcar2D, [20.000498412722 s … 20.000650412722 s] x
   5.9 GHz] → 1e-09 WpHz))
```

+

```
sensor_data = [
  "car_1",
  "car_2"]
return sensor_data
```

VS.

"Control" Community



CC-BY: https://www.youtube.com/@carlasimulator8782

```
can_receive =
  dist(car1, car2)
  < d_max
return can_receive
```

+

Concept and First Results

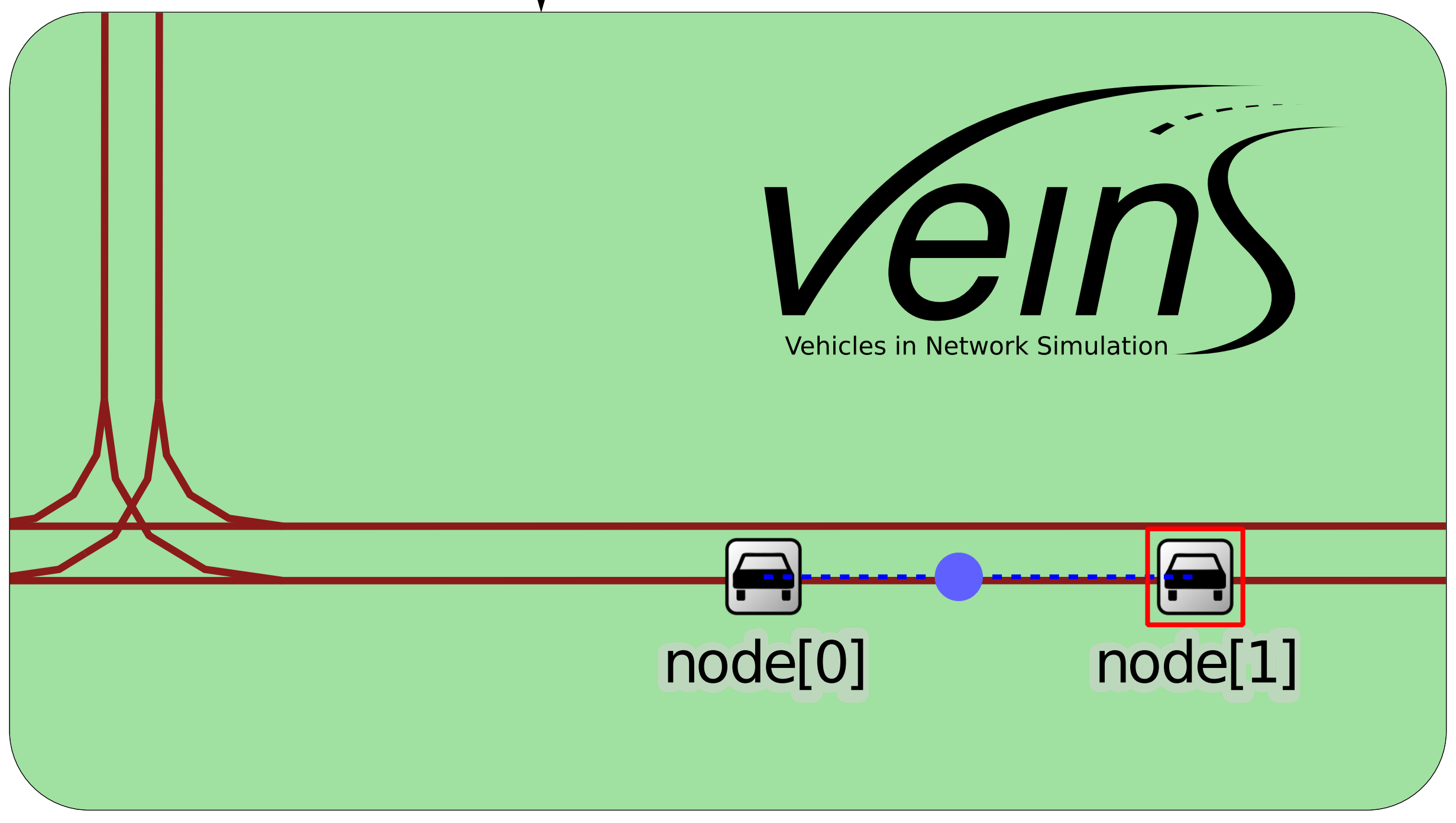
```
syntax = "proto3";

import "google/protobuf/empty.proto";

package carla;

service CarlaAdapter{
    rpc ExecuteOneTimeStep (google.protobuf.Empty)
    rpc Finish (google.protobuf.Empty) returns (go
    rpc GetManagedActorsIds(google.protobuf.Empty)
    rpc GetManagedActorById(Number) returns (Vehic
    rpc InsertVehicle(Vehicle) returns (Number);
```

gRPC



+

Source Code:
https://github.com/veins/veins_carla

Contact:
Tobias Hardes https://www.cms-labs.org/people/hardes/