Performance Analysis of Embedded Platoon Controllers

Amr Ibrahim*, Iñaki Martín Soroa*, Hong Li†, Dip Goswami*, Twan Basten*‡
*Eindhoven University of Technology, Eindhoven, The Netherlands
†Car Infotainment & Driving Assistance, NXP Semiconductors, Eindhoven, The Netherlands
‡ESI, TNO, Eindhoven, The Netherlands
Email: {a.ibrahim, d.goswami, a.a.basten}@tue.nl, i.martin.soroa@gmail.com, hong.r.li@nxp.com

***Abstract*—Vehicle platooning is a technology capable of reducing the distance between vehicles, which in turn increases the road capacity and reduces the fuel consumption. In vehicle platooning, vehicles exchange information through wireless Vehicle-to-Vehicle (V2V) communication. The maximum message rate is limited by the traffic of vehicles equipped with V2V capabilities and the communication protocols. It can vary between 1Hz and 10Hz in IEEE 802.11p. Many platoon control strategies in the literature do not consider the limited message rate and are not usable in real-life scenarios. One of the strategies capable of dealing with lower message rates uses Model Predictive Control (MPC), a type of optimal controller with a high computational cost. In this work, we analyze the performance of an MPC platoon control over IEEE 802.11p using embedded platforms from Cohda Wireless and NXP Semiconductors. We consider a set of commonly used message rates – 1, 2, 5 and 10Hz as well as sensor noise. We analyze the string stability and fuel consumption to evaluate the performance of the platoon controllers. We show that MPC provides satisfactory performance with a message rate as low as 1Hz and it outperforms the platoon control state-of-the-art techniques. Our results clearly show the need for taking into account message rate restrictions in the control algorithms.**

I. Introduction

Vehicle platooning is a cooperative driving technique in which autonomous vehicles follow each other closely maintaining a safe inter-vehicle gap. This increases the road capacity and reduces the air drag on the vehicles, reducing the fuel consumption. Cooperative Adaptive Cruise Control (CACC) is an enabling technology for vehicle platooning; it is an extension of ACC and enhances its functionality by integrating Vehicle-to-Vehicle (V2V) wireless communication between vehicles, along with other on-board sensors. Vehicles communicate with each other using a V2V communication standard e.g. ETSI-ITS. In ETSI-ITS (which uses IEEE 802.11p), vehicle platooning applications use the control channel (CCH) for sending CAM (Cooperative Awareness Messages) messages, achieving a maximum message rate of 10Hz. Depending on the vehicular congestion level, the message rate can be as low as 1Hz. We consider message rates of 1, 2, 5, 10Hz. It should be noted that more message rates can be used [1].

Most theoretical works do not consider the computational constraints and real-time issues that arise from the implementation of different control strategies on embedded platforms. Model Predictive Control (MPC) is a type of optimal controller that requires solving an optimization problem in every iteration. The embedded implementation of MPC for vehicle platooning is challenging due to the large computational requirements of MPC.

For this paper we implemented the control strategy proposed in [2], a distributed multi-rate control strategy with an upper layer using an MPC controller and a lower layer using a state-feedback controller, on the Cohda Wireless MK5 embedded platform, developed by Cohda Wireless and NXP. We have developed an evaluation setup with four devices communicating wirelessly using the ETSI-ITS protocol, with the aim to explore the challenges that arise from the implementation and show that the theoretical performance of the approach also translates to a good performance in a real system. Each of the devices simulates one platoon vehicle and runs in real time. The setup simulates movement of the vehicles using a vehicle dynamic model and simulates the measurement devices by adding noise to the predecessor's reported position.

To evaluate the performance of the embedded controllers, we consider two metrics – string stability and fuel consumption. String stability is the property by which disturbances are attenuated in the upstream direction of the platoon (the rear of the platoon). Without string stability, any disturbance in the head of the platoon would be amplified as it propagates through the platoon (reducing the inter-vehicle distance) which might affect safety, driving comfort and fuel savings. Fuel consumption reduction is one of the main incentives for realizing vehicle platooning. It is achieved by allowing the vehicles to drive very close to each other, reducing the aerodynamic drag force. We compare our algorithm with the state-of-the-art vehicle platooning control algorithm, Ploeg's algorithm presented in [3], which is a PD-based multi-layer modular vehicle platooning control system. It was re-implemented in [4] for the MK5 platform and is available in [5].

MPC has been implemented for embedded platforms for several applications [6], [7], [8], [9]. A number of works propose using specialized hardware such as FPGAs or ASICs [7], [8]. Applications using conventional embedded platforms are generally limited to controlling systems with low sampling rates [6], [9]. In [10], it is proven that embedded MPC for vehicle platooning using conventional hardware is feasible.

For solving the embedded MPC optimization problem we use the Fast Gradient Method (FGM) that was first developed in 1983 by Nesterov [11] and was proposed in [12] for solving

MPC problems with quadratic cost function and convex input constraints set. In [13], FGM was proposed for embedded applications. Other methods can be used to solve the optimization problem such as active set method or interior point method. FGM is chosen here because it does not require solving a linear set of equations at every step.

This paper is organized as follows. Section II introduces the considered platoon scenarios. In Section III, vehicle, platoon and fuel consumption models are introduced. Section IV introduces the multi-layer control structure of the platoon vehicles and the embedded FGM solver. The embedded platform architecture is introduced in Section V and in Section VI we analyse the performance of the platoon scenarios. Section VII draws conclusions.

II. Platoon Scenarios under Consideration

As a case study we consider a platoon of four vehicles circulating on a highway occupying the right lane. We simulate the platooning system on four MK5 devices (see Fig. 1). Each device simulates one vehicle and communicates with the other devices (vehicles) using IEEE 802.11p. The movement of the vehicles is simulated using models while the software runs in real-time using the ETSI-ITS communication stack and the full control stack. We analyze two different platooning scenarios:

- **Scenario-1**: the vehicles start with zero initial speed, the leader accelerates gradually until reaching 80km/h and then brakes decreasing the speed to 45km/h. With this profile we observe the behaviour of the platoon in normal conditions, without sudden changes in the acceleration.
- **Scenario-2**: The vehicles quickly accelerate to reach 80km/h and maintain the speed. We analyze the fuel consumption of the platoon when it runs at 80km/h on a flat highway (no road inclination) and it has already reached steady state. We compare it to the fuel consumption of vehicles moving with a 50m gap (measured as the distance between the front of a vehicle to the rear end of its preceding vehicle) between them that follow the same behaviour as the leader.

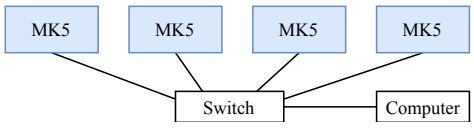

Fig. 1. Emulation hardware setup

III. Modelling

In this section we establish the vehicle, platoon and fuel consumption models required for control and simulation.

A. Vehicle and platoon model

Each vehicle has a model of itself (vehicle model) and its relation with its predecessor (inter-vehicle dynamics), which together represent the distributed platoon model.

The model of vehicle i is obtained by combining the dynamics of the engine with a simplified model of the vehicle's longitudinal dynamics. The throttle angle is adjusted by the throttle actuator, which is modeled as a DC motor [14], obtaining the combined model (vehicle model):

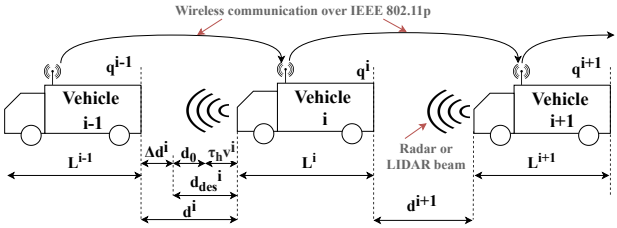

Fig. 2. Consecutive vehicles in a platoon

$$\dot{x}_v^i = A_v^i x_v^i + B_v^i u_v^i, \tag{1}$$

where $x_v^i = [a^i \quad \dot{a}^i]^T$ is the state vector, where a^i and $\dot{a}^i$ are the acceleration and the rate of change of acceleration of vehicle i, respectively. A_v^i and B_v^i are the state and input matrices respectively and u_v^i is the duty cycle of the motor's input signal. Moreover, the state matrix A_v^i and the input vector B_v^i of vehicle i are defined as:

$$A_v^i = \begin{pmatrix} 0 & 1 \\ \frac{-1}{\tau^i \tau_a^i} & \frac{-(\tau^i+\tau_a^i)}{\tau^i \tau_a^i} \end{pmatrix} \in \mathbb{R}^{2\times 2}, B_v^i = \begin{pmatrix} 0 \\ \frac{K^i K_a^i}{\tau^i \tau_a^i} \end{pmatrix} \in \mathbb{R}^{2\times 1}, \tag{2}$$

where K^i, τ^i, K_a^i and τ_a^i are parameters which depend on the characteristics of vehicle i. $\mathbb{R}$ is the set of real numbers.

The platoon follows the predecessor-follower (PF) topology, which is modelled through the inter-vehicle dynamics, relating vehicle i to vehicle $i-1$. The inter-vehicle dynamics are modelled with two new states, Δv^i and Δd^i:

$$\Delta v^i = v^{i-1} - v^i, \tag{3}$$

$$\Delta d^i = d^i - d_{des}^i, \tag{4}$$

where Δv^i is the velocity error between the vehicles i and $i-1$, where v^i is the velocity of vehicle i. Δd^i is the gap error, the difference between the actual gap (d^i) and the desired inter-vehicle gap (d_{des}^i) between vehicles i and $i-1$. d^i and d_{des}^i are defined as:

$$d^i = q^{i-1} - q^i - L^i, \tag{5}$$

$$d_{des}^i = d_0 + \tau_h v^i, \tag{6}$$

where L^i and q^i are the length and position of vehicle i, respectively. d_0 is the gap between vehicles at standstill and τ_h is the constant headway time (the time vehicle i needs to reach the position of vehicle $i-1$ when $d_0 = 0$), see Fig. 2.

Combining the partial models (vehicle model with the inter-vehicle dynamics) we obtain the platoon model:

$$\dot{x}_p^i = A_p^i x_p^i + B_p^i u_p^i + G_p^i a^{i-1}, \tag{7}$$

where $x_p^i = [a^i \quad \dot{a}^i \quad \Delta d^i \quad \Delta v^i]^T$ is the state vector, a^{i-1} is the acceleration of the preceding vehicle and $G_p^i = [0 \quad 1]^T$. A_p^i and B_p^i are the state and input matrices, respectively.

B. Fuel consumption model

There are numerous factors that affect fuel economy, but the most effective factors are: engine, smooth driving behavior, vehicle acceleration and air drag [15].

A simple fuel consumption model can be obtained based on the longitudinal vehicle dynamics. Using Newton's second law, the equilibrium of forces on a vehicle on a flat surface is [16]:

$$F^i_{engine} = m^i a^i + F^i_{aero} + F^i_{roll}, \tag{8}$$

where F^i_{engine}, F^i_{aero} and F^i_{roll} are the force exerted by the engine, the aerodynamic resistance and the rolling resistance respectively of vehicle i. m^i is the mass of vehicle i and a^i is its acceleration. For simplicity we assume that the transmission is ideal so gear changes are not taken into account. We also assume no road inclination and no wind. The rolling resistance can be defined as:

$$F^i_{roll} = f^i m^i g, \tag{9}$$

where g is the gravitational acceleration and f^i is a parameter depending on the vehicle and the tires. The aerodynamic resistance can be defined as:

$$F^i_{aero} = \frac{1}{2}\rho C^i_{d0} A^i_f (1-\phi^i)(v^i)^2, \tag{10}$$

where ρ is the air density, C^i_{d0} is the drag coefficient of the vehicle, ϕ^i is the attenuation of the drag force due to the proximity of other vehicles and A^i_f is the frontal area of the vehicle. We select the values of m^i, f^i and A^i_f as an average of the values shown in [17] for several cab-over trucks; these vary depending on the exact vehicle used. We approximate ϕ^i using the following rational formula, obtained by fitting a curve to the experimental data for truck platoons presented in [18]:

$$\phi^i = \frac{C^i_{d1} d^i + C^i_{d2}}{d^i + C^i_{d3}}, \tag{11}$$

where d^i is as defined in Fig. 2 except in the case of the leader, where we use the gap to its follower and different values of C^i_{d1}, C^i_{d2} and C^i_{d3} (see Table I).

We assume that F^i_{engine} and v^i are constant for a time interval dt equal to the rate at which information is logged (logging period), since information for shorter periods is not available. We can compute the energy consumption at every time instant (E) as:

$$Power^i(k) = F^i_{engine}(k)\frac{dq(k)}{dt} = F^i_{engine}(k) v^i(k), \tag{12}$$

$$E^i(k) = Power^i(k) dt. \tag{13}$$

We add together all the positive energy values to obtain the total consumed energy. Knowing the average engine efficiency (η) and the calorific power of the fuel (Q_c), we can estimate the fuel consumption as:

$$Fuel^i = \frac{1}{\eta Q_c} \sum_{k, E^i(k)\geq 0} E^i(k). \tag{14}$$

An example of the parameters used in this paper is shown in Table I.

TABLE I
VEHICLE PARAMETERS FOR FUEL CONSUMPTION

m^i	f^i	A^i_f	ρ	η	Q_c
40000kg	0.048	9.7m^2	1.225kg/m^3	0.45	45MJ/kg

	C^i_{d0}	C^i_{d1}	C^i_{d2}	C^i_{d3}
Leader	0.68	-10.81	171.4	11.530
1st follower		19.12	408.8	6.425
Others		28.89	420.7	6.034

IV. Control Structure of Platooned Vehicles

A multi-layer control structure is adopted for the platoon vehicles. The lower layer controller is a state-feedback controller with a sampling rate of 2ms. The output of this controller is the motor duty cycle which controls the vehicle acceleration [2]. The upper-layer controller uses Model Predictive Controller (MPC) [19] with a 100ms sampling period, matching the ETSI-ITS communication standard maximum message rate. By using the prediction of MPC, we extend the upper-layer controller to deal with higher sampling rate e.g. 1s (1Hz).

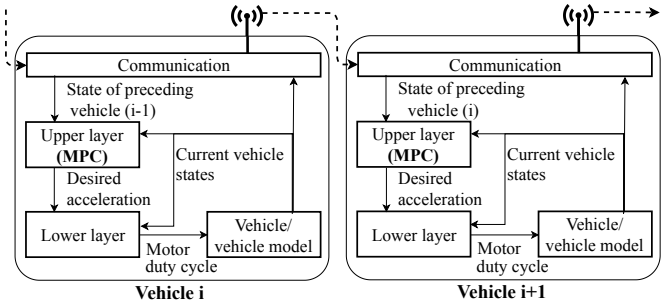

Fig. 3. Structure of the control strategy

A. Lower-layer controller

The state-feedback controller computes the motor duty cycle $u^i_v(k)$ as:

$$u^i_v(k) = \kappa^i x^i_v(k) + F^i a^i_{des}(k), \tag{15}$$

where $x^i_v(k)$ is the discretised vehicle state, a^i_{des} is the desired acceleration (given by the upper layer). The feedback (κ^i) and the feedforward (F^i) gains are designed such that the system is stable and reaches the desired acceleration.

B. Upper-layer controller using MPC

MPC solves an optimization problem by finding the minimum solution of the following quadratic cost function with respect to specific constraints on states and inputs:

$$J = {x^i_{N+k|k}}^T P x^i_{N+k|k} + \sum_{j=0}^{N-1} \left[\left({x^i_{j+k|k}}^T Q x^i_{j+k|k} + {u^i_{j+k|k}}^T R u^i_{j+k|k} \right) \right]$$

$$\textbf{s.t.:} \quad x^i_{j+k+1|k} = \Phi^i x^i_{j+k|k} + \Gamma^i u^i_{j+k|k} + \Psi^i a^{i-1}_{j+k|k},$$

$$x^i_{j+k+1|k} \in \mathcal{X},\ u^i_{j+k|k} \in \mathcal{V},\ j = 0, ..., N-1. \tag{16}$$

J is the cost function, vector $x^i_{j+k|k} = \begin{bmatrix} a^i_{j+k|k} & \delta a^i_{j+k|k} & \Delta d^i_{j+k|k} & \Delta v^i_{j+k|k} \end{bmatrix}^T$ is the predicted state vector of vehicle i after j steps computed at time k. $x^i_{k|k}$ is the measured state of vehicle i and N is the horizon length. Q, R and P are the weighting parameters. $u^i_{j+k|k}$, $j = 0, \ldots, N-1$ is the sequence of optimal control inputs that will be computed where only the first value will be applied before solving the MPC problem again at the next time step. $\mathcal{X}$ and $\mathcal{V}$ are the sets of admissible values (defined by box constraints) for the states and control inputs, respectively. A quadratic cost function is chosen so that the problem is convex, and a global minimum can be found.

MPC requires a discrete model, therefore we discretize the platoon model Eq. 7 using Zero-Order Hold (ZOH). The discretized model is the predictive model for vehicle i shown in Eq. 16. $a^{i-1}_{j+k|k}$ is the predicted acceleration of the preceding vehicle. We assume that the acceleration of the preceding vehicle is constant for the next N time steps. Therefore, it does not affect the optimization process.

C. MPC solver implementation

The optimization problem defined in Eq. 16 can be compactly rewritten in the following quadratic programming (QP) form [19],

$$\begin{aligned} \min_{U^i_k} \quad & \overbrace{\frac{1}{2} {U^i_k}^T G^i U^i_k + U^i_k \mathcal{F}^i}^{\mathcal{U}}, \\ \textbf{s.t.:} \quad & LU^i_k \leq C + wx^i(k), \end{aligned} \tag{17}$$

where the states in Eq. 16 are eliminated and expressed as a function of the current state $x^i(k)$ and the future (optimal) input sequence U^i_k. The optimization variable $U^i_k = \begin{bmatrix} u^i_{k|k} & u^i_{k+1|k} & \ldots & u^i_{k+N_p-1|k} \end{bmatrix}^T$ contains the optimal control inputs over the prediction horizon at time k for vehicle i. The $\mathcal{F}^i$ matrix is a function of the current state $x^i(k)$. $LU^i_k \leq C + wx^i(k)$ is a condition that encompasses input and state constraints.

The MPC problem is solved using a FGM (Fast Gradient Method) solver. The custom FGM solver used in this paper and implemented on MK5 devices is adopted from [13]. FGM computes the gradient of the cost function for the current sensed state, which using the compact formulation Eq. 17 can be computed as,

$$\nabla \mathcal{U} = U^i_k G^i + \mathcal{F}^i. \tag{18}$$

The algorithm used in the MATLAB implementation is described in Algorithm 1, where $P_V(\cdot)$ (Eq. 19) is a projection function which projects the gradient on the set V, the feasible constrained set of the control inputs.

$$P_V(w, h) = arg\ min_{q \in V} ||q - v||^2, \tag{19}$$

$$v = w - h \nabla \mathcal{U}, \tag{20}$$

where h is the step size chosen such that $J(U^i_{k+1}) \leq J(U^i_k)$. V is defined as,

$$V = \{a_{min} \leqslant u^i(k) \leqslant a_{max}\}, \tag{21}$$

where a_{max} and a_{min} are the upper and lower bounds of the control input, respectively. The maximum number of iterations is computed as,

$$\begin{aligned} i_{max} &\leq min\left(\frac{\ln 2\epsilon - \ln L\tilde{d}^2}{\ln(1 - \sqrt{\frac{\mu}{L}})}, \sqrt{\frac{2L\tilde{d}^2}{\epsilon}} - 2 \right), \\ \tilde{d}^2 &= N \frac{\left(a_{max} - a_{min}\right)^2}{2}, \end{aligned} \tag{22}$$

where μ and L are the minimum and maximum eigenvalues of the matrix G^i, and ϵ is the suboptimality level.

Algorithm 1: Fast Gradient Method algorithm

Data: state $x^i(k)$, initial guess $y \in V$, number of iterations i_{max}, maximum and minimum eigenvalues of G^i: L, μ
Set $U^{old} = y$, $w = y$
for $i = 1, \ldots, i_{max}$ **do**
  compute $U = P_V(w, \frac{1}{L})$;
  compute $w = U + \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}(U - U^{old})$
  set $U^{old} = U$
end
Result: optimal control inputs U

Note that this FGM solver can only constrain the control inputs. However, the states are guaranteed to be within reasonable limits through penalising the states in the cost function and running the simulation with feasible initial values.

V. Embedded Platform Architecture

We use the Cohda Wireless MK5 platform, a prototyping platform for V2V applications, such as CACC. The platform has one main processor, an NXP i.MX6 DualLite @ 800MHz, paired with a communication co-processor, NXP MARS. It is equipped with 1GB of volatile memory, see Fig. 4. The device uses an Ubuntu distribution of Linux as its Operating System (OS). The ETSI-ITS communication protocol is available as a system application. We use the embedded implementation presented in [4] and available in [5] as the base of our MPC-based control system implementation.

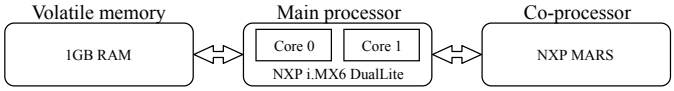

Fig. 4. Cohda Wireless MK5 block diagram

A. Hardware architecture

We use four MK5 devices connected via an Ethernet network and via a wireless IEEE 802.11p network, see Fig. 1. The Ethernet network is used as the interface between the devices and the computer and as a low latency network for time synchronization. In real vehicles the time synchronization can be done using GNSS (Global Navigation Satellite System) [20]. The wireless network is used for the platoon experiment, so that the experiment uses the same network that would be used in real vehicles. Each device simulates a different vehicle and performs all the control tasks of the vehicle in real-time.

The upper layer and lower layer controllers are executed in the main processor (NXP i.MX6 DualLite). Following the OSI

model [21], the host (upper) layers of the OSI protocol stack are executed in the main processor while the media (lower) layers are executed in the co-processor (NXP MARS) (see Fig. 5). This allows the main processor to run main platoon tasks while the co-processor takes care of other simple but time consuming tasks.

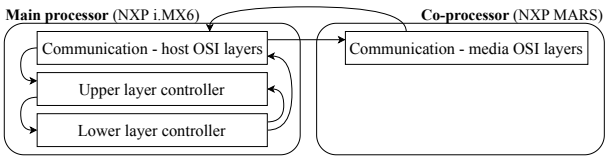

Fig. 5. Task distribution between the main processor and the co-processor

B. Software architecture

Task model: The platooning algorithm is divided into four main tasks executed on the main processor (although the communication is realized in the MARS co-processor): sending messages, receiving messages, upper layer controller, and lower layer controller. These tasks must execute independently from each other, either because they have different task periods (upper layer and lower layer) or because their execution depends on external factors (sending and receiving messages). A main function sets the V2X configuration and initializes all the threads. The upper layer is responsible for solving the MPC optimization problem and for that it requires some libraries. The sending and receiving threads make use of the V2X libraries. The threads are governed by a scheduler, see Fig. 6.

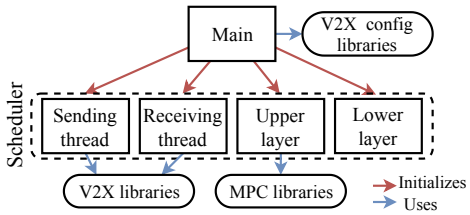

Fig. 6. General software architecture with the main tasks, libraries and interactions

The detailed tasks performed by each of the threads are as follows:

- **Receiving thread:** time synchronization (only initialization), receive and decode V2V messages, add noise to the received measurements (to simulate a realistic sensor), compensate for the message delay, and apply a filter to the noisy signal.
- **Sending thread:** time synchronization (only initialization), send V2V messages.
- **Upper layer thread:** perform MPC tasks, handle low message rates, log basic timing data.
- **Lower layer thread:** apply lower layer controller and update vehicle states, predict current predecessor state, log states (position, velocity, acceleration, jerk, etc).

Task scheduling: The selected device uses a standard Operating System (OS) instead of a Real-Time OS (RTOS); therefore, it cannot guarantee a strict task period or a bounded execution time. The execution frequency was implemented by making the thread sleep for a period of time defined as $t_s = \frac{1}{f_{des}} - t_{exec}$, where t_s is the sleeping time, f_{des} is the desired execution frequency, and t_{exec} is the execution time. In this way the length of the sleep adapts to the execution time of the tasks, which are variable due to non-real-time OS. Thus, we achieve a reasonably periodic execution of the tasks since there can still be some variation/jitter in between periods. While the thread sleeps, it releases the resources that it was using.

Although the execution time for the tasks is not bounded, in practice it is always smaller than the execution period except for the lower layer, which can be delayed after performing a system call to dump the cache where the logged data is stored. Those delays in the lower layer are ignored in Table II (do not have significant impact on performance or stability).

TABLE II
TASKS PRIORITIES AND EXECUTION TIMES

Task	**Priority**	**Average t_{exec}**	**Worst t_{exec} observed**
OS	Highest	-	-
Receiving thread	Medium	0.401ms	1.041ms
Sending thread	Medium	0.132ms	0.425ms
Lower layer	Medium	0.0166ms	0.226ms
Upper layer	Lowest	0.339 ms	8.418ms

The tasks are scheduled using a Fixed Priority Pre-emptive Scheduler (FPPS), which handles equal priority tasks in a First-In First-Out (FIFO) manner. In Table II, we see the priority, average t_{exec}, and worst t_{exec} of each thread. In this implementation, the sending thread and specially the receiving thread are slowed down due to the communication with the co-processor. The upper layer is discussed in Section V-C.

C. Embedded FGM solver and code generation

The embedded implementation of the FGM solver and the computation of the compact form of the matrices for MPC use automatic C code generation. The algorithms were implemented in MATLAB and converted to C code using MATLAB Coder. This tool was unable to to convert all the required operations, namely, continuous to discrete model transformation and eigenvalue computation of G^i (see Eq. 18). The model that needs to be discretised (Eq. 7) depends on several constant vehicle parameters and the headway time , which is variable. The continuous-to-discrete transformation has been approximated by finding a linear relation between the discrete model and the headway time (the only value involved in the continuous model that can vary without changing the vehicle). We built a parametric discrete model with headway time as the only parameter. For the eigenvalue computation (required by Algorithm 1), we used the GNU Scientific Library (GSL) compiled for the embedded devices.

For our MPC problem, this implementation of MPC using the FGM solver takes on average 0.339ms, while other embedded MPC implementations using automatic code generation require over 23ms for a similar problem [10]. The execution time is in principle not bounded, with the largest observed execution time being 8.418ms, while $< 0.2\%$ of samples take more than 1ms. Taking the worst execution time observed, the maximum achievable execution rate is 118Hz.

VI. Performance Analysis with HIL

A. Vehicle platooning performance

Using Scenario-1 (see Section II), we found the minimum allowable headway time (the minimum headway time that preserves string stability). In Fig. 7 and Fig. 8, we show the acceleration profiles of platoon vehicles for a message rate of 10Hz and 1Hz, respectively. Each of them shows a string stable behaviour for the minimum headway time (disturbances are attenuated, see the zoomed-in curves in Fig. 7 and Fig. 8).

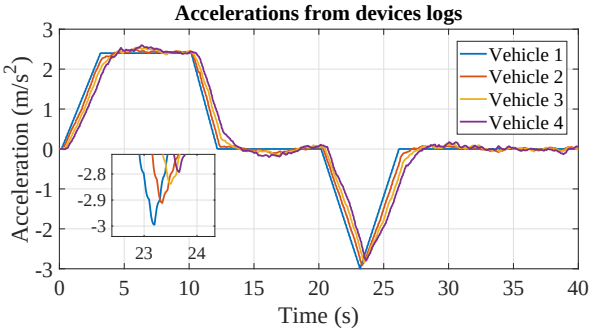

Fig. 7. Acceleration profile in Scenario-1, 10Hz message rate and headway time $\tau_h = 0.2$s

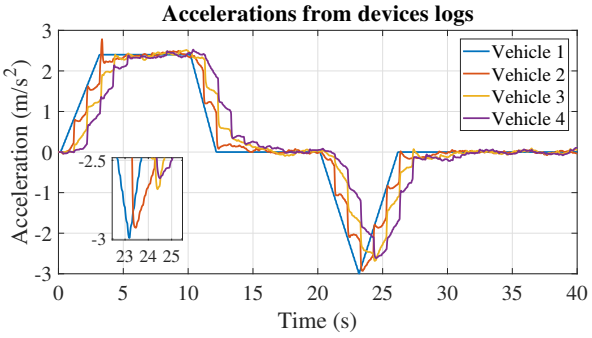

Fig. 8. Acceleration profile in Scenario-1, 1Hz message rate and headway time $\tau_h = 0.6$s

In Fig. 9, we can observe the gap error as detected by the vehicles (after adding noise and applying a filter). The profiles are still noisy but they remain within reasonable margins. With 10Hz message rate, the error is kept within ± 0.4m, while with 1Hz message rate, the gap error is larger, staying within ± 1m.

In Fig. 10, we show the minimum allowable headway time in comparison between MPC and the Ploeg-Zhu controllers [4], [3], which are PD controllers. This shows that the performance of MPC is superior to PD in embedded implementations.

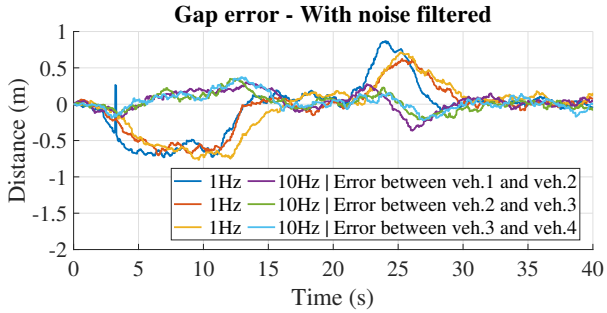

Fig. 9. Error in position between platoon vehicles in Scenario-1

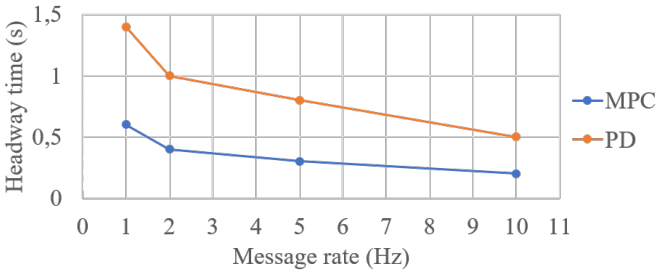

Fig. 10. Comparison between MPC and Ploeg-Zhu controllers [3], [4]

B. Fuel economy

Scenario-2 is used to estimate the fuel consumption. In Fig. 11, we can observe the acceleration profile of the platoon leader and the third follower in Scenario-2. The acceleration profile of the follower shows a noisy behaviour which affects the fuel consumption negatively, but its effect is smaller than the positive effect of having a smaller inter-vehicle distance (which reduces the air drag). Recall that the messages include information on the current acceleration but not on the future desired behaviour. Thus, when the leader acceleration is not constant (from 0s to 12s), the profile is more noisy when the message rate is lower (see also Fig. 7 and Fig. 8). This effect is not noticeable when the leader acceleration is constant and a steady-state is reached (from 17s to 80s), with all message rates having an equally noisy behaviour.

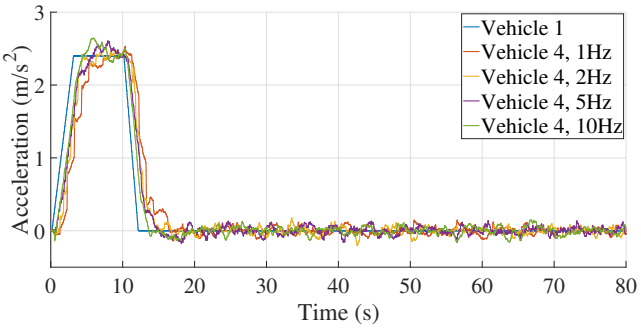

Fig. 11. Scenario-2 acceleration profiles for the leader and the third follower with different message rates. Following Fig. 10, a lower message rate uses a larger headway time (to preserve string stability)

In Fig. 12, we can observe the instantaneous power consumption of the platooning vehicles (see Eq. 14). The power

consumption ramps up at the beginning. Although the acceleration is constant, the speed increases and thus the air drag also increases. The vehicles at the rear of the platoon have a lower peak power consumption. After reaching the steady state, the power consumption is noisy for all the follower vehicles.

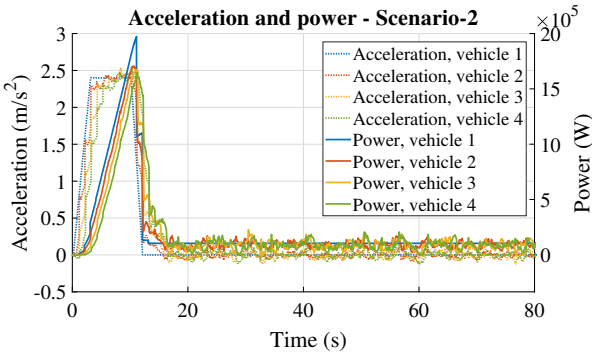

Fig. 12. Power and acceleration profile for 1Hz message rate and headway time $\tau_h = 0.6$s

In Fig. 13, we can observe the fuel savings for different positions in the platoon. At the rear of the platoon, it can reach up to 10%. Higher message rates allow for a smaller inter-vehicle gap, which achieves a larger reduction in the fuel consumption. The leader of the platoon also benefits from a smaller inter-vehicle gap, as the airflow behind the vehicle also affects its air drag. The third and fourth vehicles present a roughly equal fuel consumption. The differences in the fuel consumption for different message rates correspond to smaller inter-vehicle gaps.

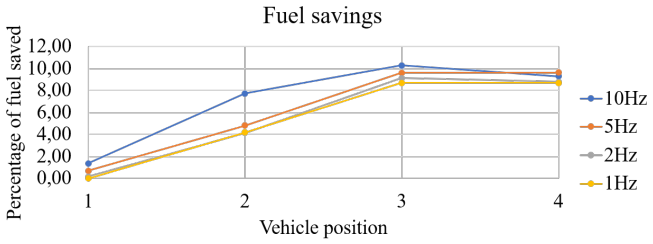

Fig. 13. Fuel savings under different message rates (average of 3 simulations)

VII. Conclusion

For the first time, we presented an embedded implementation of MPC-based multi-layer platoon control. The presented implementation takes into account the message rate restrictions imposed by the IEEE 802.11p V2V communication standard. We evaluated two performance metrics – string stability and fuel savings. We compared an MPC-based platoon controller with the state-of-the-art platoon control. MPC outperforms the state-of-the-art. We show the impact of message rate on platoon performance. The results clearly show that message rate should be considered in platoon control in real-life systems.

Acknowledgements

This research received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under grant agreement no 674875 (oCPS).

References

[1] European Telecommunications Standards Institute. Intelligent Transport Systems (ITS); Harmonized Channel Specifications for Intelligent Transport Systems operating in the 5 GHz frequency band, 2012.

[2] Ibrahim, Amr, et al. "Co-simulation framework for control, communication and traffic for vehicle platoons." 2018 21st Euromicro Conference on Digital System Design (DSD). IEEE, 2018.

[3] Ploeg, Jeroen, et al. "Design and experimental evaluation of cooperative adaptive cruise control." 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE, 2011.

[4] Sijie Zhu. Model-in-the-loop experiments and analysis of platoon control algorithms. Master's thesis, Technische Universiteit Eindhoven, 2018.

[5] Sijie Zhu. NXP Platoon Control Algorithm Evaluation Platform. URL: https://github.com/sijiezhu/NXP-Platoon-Control-Algorithm-Evaluation-Platform.

[6] Zometa, P., et al. "On Time versus Space and Related Problems." American Control Conference (ACC). 2012.

[7] Bleris, Leonidas G., et al. "A co-processor FPGA platform for the implementation of real-time model predictive control." 2006 American Control Conference. IEEE, 2006.

[8] Wills, Adrian G., Geoff Knagge, and Brett Ninness. "Fast linear model predictive control via custom integrated circuit architecture." IEEE Transactions on Control Systems Technology 20.1 (2011): 59-71.

[9] Currie, Jonathan, Arrian Prince-Pike, and David I. Wilson. "Auto-code generation for fast embedded model predictive controllers." 2012 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP). IEEE, 2012.

[10] Soroa, Iñaki Martín, et al. "Feasibility study and benchmarking of embedded MPC for vehicle platoons." 1st International Workshop on Autonomous Systems Design. 2019.

[11] Nesterov, Yurii E. "A method for solving the convex programming problem with convergence rate $O(1/k^2)$." Dokl. akad. nauk Sssr. Vol. 269. 1983.

[12] Richter, Stefan, Colin N. Jones, and Manfred Morari. "Real-time input-constrained MPC using fast gradient methods." Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference. IEEE, 2009.

[13] Kögel, Markus, and Rolf Findeisen. "A fast gradient method for embedded linear predictive control." IFAC Proceedings Volumes 44.1 (2011): 1362-1367.

[14] Ulsoy, A. Galip, Huei Peng, and Melih Çakmakci. Automotive control systems. Cambridge University Press, 2012.

[15] Zhou, Min, Hui Jin, and Wenshuo Wang. "A review of vehicle fuel consumption models to evaluate eco-driving and eco-routing." Transportation Research Part D: Transport and Environment 49 (2016): 203-218.

[16] Rajamani, Rajesh. Vehicle dynamics and control. Springer Science & Business Media, 2011.

[17] Henrik Stenvall. Driving resistance analysis of long haulage trucks at Volvo-Test methods evaluation. Master's thesis, Chalmers University of Technology, 2010.

[18] Wolf-Heinrich, Hucho, ed. Aerodynamics of road vehicles: from fluid mechanics to vehicle engineering. Society of Automotive Engineers, 1997.

[19] Maciejowski, Jan Marian. Predictive control: with constraints. Pearson education, 2002.

[20] Hofmann-Wellenhof, Bernhard, Herbert Lichtenegger, and Elmar Wasle. GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more. Springer Science & Business Media, 2007.

[21] Day, John D., and Hubert Zimmermann. "The OSI reference model." Proceedings of the IEEE 71.12 (1983): 1334-1340.