Consensus with State Obfuscation: an Application to Speed Advisory Systems

Wynita Griggs, Giovanni Russo and Robert Shorten

Abstract— **In this paper, we present a new speed advisory system (SAS). The system provides indications to drivers which, if followed, guide the network of vehicles towards a common speed. The SAS we present is distributed and one of its key features is that the vehicles are guided towards the common speed by receiving *obfuscated* information. Essentially, this means that the desired network behaviour is achieved and, at the same time, there is no vehicle in the network that obtains clear knowledge of the state of any other vehicle. This results in a *privacy-preserving* feature which is particularly useful in smart cities applications, where drivers might not be willing to share their transient state with others.**

I. Introduction

Excessive or inappropriate speed plays a significant role in serious road accidents [1], thus motivating, over the years, the development of several Intelligent Speed Adaptation (ISA) systems. The original goal of such systems was to promote safe driving by alerting drivers when road speed limits were exceeded [2]. Since their first conceptualisation, however, ISA systems have considerably evolved into *smarter* systems, in an attempt to offer additional benefits to drivers and the surrounding environment. A remarkable example is given by speed advisory systems (SASs). The goal of such systems is to recommend a suggested speed to drivers, and they have proven useful in reducing general road traffic chaos and preventing travelling delays [3], [4]. Additional benefits of SASs include, but are not limited to: (i) ensuring that vehicles travel at safe speeds and at safe distances from vehicles ahead of them [5]; (ii) maintaining, as much as possible, the free flow of traffic, navigating optimally through bottlenecks when they occur, thus increasing overall throughput on roads; and (iii) helping to reduce factors such as pollution emissions and fuel consumption [3], [6].

The key goal of this paper is to propose a new architecture for a SAS with the objective of guiding a set of moving vehicles towards the same speed while driving in a common area. In contrast to previous algorithms, which strive to achieve consensus on a common driving speed via algorithmic innovation (see, for example, [7], [8] and the references therein), our work explores the benefits that can be achieved by allowing consensus protocols to evolve over multiple parallel network topologies. As with other consensus algorithms, our algorithm provides signals to the driver and, if such indications are followed, then all vehicles will achieve a common speed. However, by using a parallel architecture, we shall see that consensus can be achieved by exchanging noisy (obfuscated) information, which is important when privacy is a consideration.

This work was partially supported by Science Foundation Ireland grant 11/PI/1177.

W. Griggs and R. Shorten are with University College Dublin, School of Electrical, Electronic and Communications Engineering, Belfield, Dublin 4, Ireland. Emails: wynita.griggs@ucd.ie, robert.shorten@ucd.ie

G. Russo and R. Shorten are with IBM Research Ireland, Optimisation and Control Group, Dublin, Ireland. Email: grusso@ie.ibm.com

In our work, same speed driving is achieved by each vehicle sending its speed to a base station that controls a given geographical area, which then constructs a recommended input (i.e. an acceleration value) for all of the vehicles within this area. The recommended input is then sent to the vehicles and elaborated on, onboard the vehicles, to achieve an advised speed that is subsequently shown to the drivers. As we will see, the inputs sent to the cars are computed from a multi-layered network that *blends* together information regarding vehicles' speeds and some white noise. Thus, a single car in the network does not *see* the exact speed of its neighbours, but rather, it sees a signal that is obfuscated by noise.

The algorithm we propose relies on a solid theoretical background. In the appendix, we give a new sufficient condition for the consensus, [9]–[11], in multi-layer networks, [12], modelled by a set of continuous-time stochastic differential equations, [13]. Interestingly, with our result, we show that, if properly *designed*, noise can serve as a *distributed control input* and we show that it is indeed fundamental to algorithm convergence.

II. Problem Statement

In this paper, we will design a speed advisory algorithm for a network of N interconnected vehicles driving in the same *area*. The area might be the same stretch of a road (e.g. a highway), or a *mini-city* (e.g. a university campus). The goal of the algorithm is to allow vehicles to *collaboratively* converge towards an agreed speed.

As shown in Fig. 1, the key components of the algorithm are: (i) a *base station algorithm*, that continuously takes as input the speeds of the vehicles within the controlled area, say v_i, $i = 1, \dots, N$, and returns an acceleration to each vehicle, say $u_i(t)$, $i = 1, \dots, N$; and (ii) an *in-car system algorithm*, which sends a vehicle's current speed, v_i, to the base station. This algorithm also integrates the acceleration $u_i(t)$ provided by the base station algorithm and displays this suggested speed to the driver.

III. The Proposed Architecture

In this section, we provide a description of the consensus architecture and outline its key steps. Readers are referred to

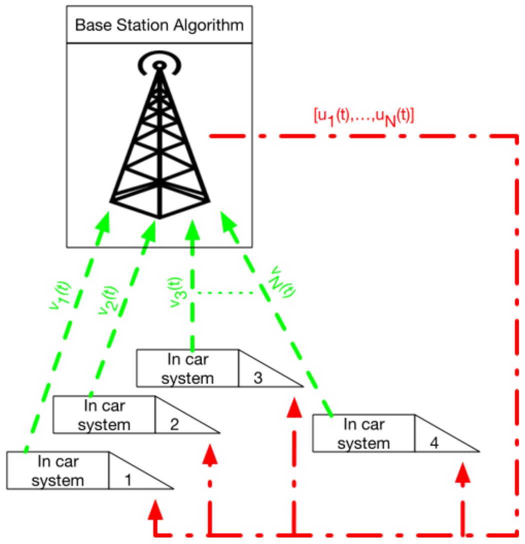

Fig. 1. Schematic diagram illustrating the set-up for the algorithm presented in this paper. Each vehicle communicates its speed to the road infrastructure (i.e. the base station) and receives, in return, a recommended speed.

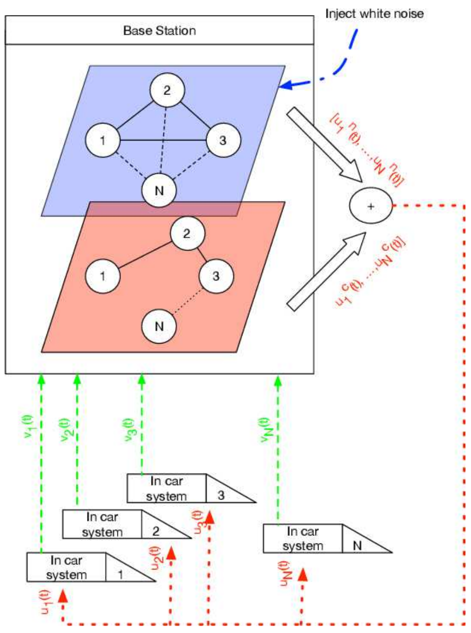

Fig. 2. Schematic diagram illustrating how the control input to the network of vehicles is computed.

the appendix for a theoretical result on network convergence.

A. *Overall architecture description*

The key idea for the algorithm is to suggest a speed to each driver within a given area controlled by a base station. Essentially, the base station receives speed data from each vehicle and constructs a recommendation (i.e. recommended input) by implementing a multi-layer network [12], where noise is injected into one of these layers. (Specifically, as outlined in Fig. 2, the multi-layer network consists of two layers in total.) The layers compute an acceleration for each vehicle. The algorithm uses the values of the vehicles' speeds to compute such, as outlined below:

• a first layer (i.e. the bottom layer in Fig. 2) that combines the speed of the last vehicle entering the area with the speed of the one that entered immediately before. This yields to the computation of a *clean* input for the vehicles, i.e. u_i^c, $i = 1,\dots,N$. Specifically, such a term is computed as

$$u_i^c(t) := v_{i+1}(t) + v_{i-1}(t) - 2v_i(t), \quad i = 2,\dots,N-1;$$
$$u_1^c(t) := v_2(t) - v_1(t);$$
$$u_N^c(t) := v_{N-1}(t) - v_N(t);$$

• a second layer (i.e. the top layer in Fig. 2) that combines all the vehicle speeds and corrupts this aggregated information with some white noise, say $w(t)$ (i.e. noise is injected in this layer). The result of this computation is a *noisy* input for the vehicles, i.e. u_i^n, $i = 1,\dots,N$. Namely, such a term is computed as

$$u_i^n(t) := w(t)\sum_{j=1}^{N}(v_j - v_i);$$

• the two components are then summed up and the recommended input that is returned to the i-th vehicle in the network is

$$u_i(t) := u_i^c(t) + u_i^n(t).$$

Physically, the input $u_i(t)$ is an acceleration. This acceleration value is received by an in-car system algorithm, which then shows to the driver the suggested speed.

We also remark here that the input received by each in-car system is an aggregate of information from vehicles within the controlled area. It is important to note that the vehicles do not know to whom they have been connected and that the signal received by each vehicle is also *corrupted* by the white noise injected at the base station. This implies that the i-th in-car system does not *see* the exact speed of its neighbouring vehicles and that, in this sense, the algorithm is privacy-preserving. In the appendix, we will give a result showing that the noise injected by the base station is crucial for the convergence of the algorithm.

B. *Pseudo-code for the base station algorithm*

The key steps of the base station algorithm are shown in Algorithm 1. Essentially, such an algorithm continuously checks for vehicles entering/exiting into/from the area controlled by the base station. This is done by creating a list that is updated: (i) every time a new vehicle entering the area is detected; (ii) whenever a vehicle exits from the controlled area. Once the list is updated, the algorithm gathers the speeds of the vehicles within the list and implements the two network layers of Section III-A. The control input, u_i, is then returned to the i-th vehicle.

C. *Pseudo-code for the in-car system algorithm*

The role of the in-car system is essentially that of facilitating the exchange of data with the base station and to provide the recommended speed to the driver. Specifically, as described in Algorithm 2, whenever a connection is established, the vehicle speed is sent to the base station. Analogously, whenever $u_i(t)$ is received, this is converted to show the recommended speed to the driver.

Algorithm 1 Pseudo-code for the base station algorithm

```
function $[u_1, \dots, u_N]$ = BASE-STATION($[v_1, \dots, v_N]$)
    Internal Variables:
    List = list of vehicles within the controlled area
    while True do
        List update
        if new vehicle within the controlled area then
            append vehicle to List
        else if vehicle exits from the controlled area then
            remove vehicle from List
        end if
        Gather data
        $N \leftarrow$ size of List
        for vehicles in List do
            get speeds, $v_i$
        end for
        Generate noise
        $w \leftarrow$ value from white noise
        Implement the bottom and top network layers
        $u_1^c \leftarrow v_2 - v_1$
        $u_N^c \leftarrow v_{N-1} - v_N$
        for $i$ in $[2, N-1]$ do
            $u_i^c \leftarrow v_{i-1} + v_{i+1} - 2v_i$
        end for
        for $i$ in $[1, N]$ do
            $u_i^n \leftarrow w \cdot \sum_{j=1}^{N} (v_j - v_i)$
        end for
        Set output
        for $i$ in $[1, N]$ do
            $u_i = u_i^n + u_i^c$
        end for
        return $[u_1, \dots, u_N]$
    end while
end function
```

Algorithm 2 Pseudo-code for the in-car system (car i)

```
function $v_i$ = IN-CAR($u_i$)
    Internal Variables:
    is-communicating: equal to 1 if communication is established
with a base station, 0 otherwise
    while True do
        update is-communicating
        if is-communicating == 1 then
            receive $u_i$ from base-station
            show recommended speed
            return $v_i$ (send to base station)
        else
            show: no connection to base station
            return Empty $v_i$
        end if
    end while
end function
```

TABLE I
SCENARIO 1: PRESET DEPARTURE SPEEDS

Passenger Vehicle No.	1	2	3	4	5
Depart Speed [m/s]	26	20	15	16	18
Passenger Vehicle No.	6	7	8	9	10
Depart Speed [m/s]	22	17	24	18	21

IV. SIMULATIONS

Emulations of some test cases were performed to provide preliminary testing and thus demonstrate the effectiveness of the algorithms introduced above. First, we performed a simplified analysis through Python 2.7.11 with the goal of quickly assessing the convergence of the algorithms. Specifically, a Python script was written which implemented Algorithms 1 and 2. In such a script, the in-car algorithm (one for each vehicle) receives an acceleration computed by the base station algorithm and converts it to a speed. In this first experiment, driver behaviour was neglected and the number of vehicles considered in the simulation was $N = 10$. All vehicles were released together at the beginning of the simulation and each vehicle had a pre-set departure speed (see Table I). The time step size used for the simulation was 0.01s and the total simulation time was 120s. The noise, w, injected by the base station algorithm was generated from a Gaussian distribution with mean equal to 0 and standard deviation equal to 0.5.

As shown in Fig. 3, the algorithms guided the speeds of the vehicles towards a common value. This confirmed the theoretical predictions presented in the appendix. Once the results obtained from the simplified Python script were assessed, we performed some more realistic simulations by means of introducing microscopic traffic simulation software, which allowed for the inclusion of some vehicular dynamics and driver behaviour. These experiments are described next.

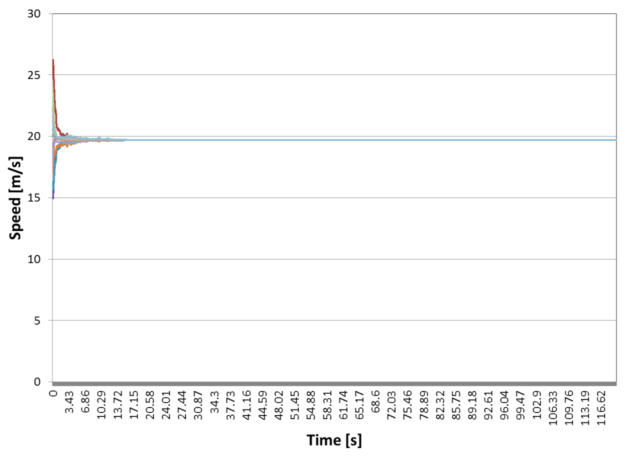

Fig. 3. Individual vehicles' speeds versus time.

A. SUMO

In order to add some more realism to the simulations, some vehicular dynamics and basic driver behaviours were incorporated via the use of the open source microscopic traffic simulation package SUMO [14]. SUMO is designed to handle large road networks and is being primarily developed at the Institute of Transportation Systems at the German Aerospace Centre (DLR). The package comes with a "remote control" interface, TraCI (short for Traffic Control Interface) [15], that allows one to adapt the simulation and to control singular vehicles on the fly. Two scenarios were identified. Descriptions of the scenario setups and results are provided next.

1) Scenario setups: A road to be used in all SUMO emulations was defined in XML, together with a base station. The road consisted of four lanes, the outermost lane being reserved for emergency vehicles, buses and taxes only. A maximum speed limit of 100 km/h was set on the road. The road is shown in Fig. 4.

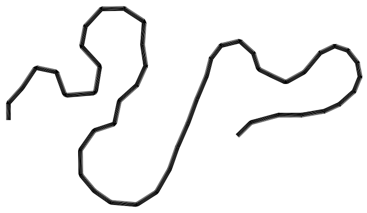

Fig. 4. Road design.

Two different scenarios were defined. In the first scenario (i.e. Scenario 1), 10 vehicles with pre-set departure speeds were released approximately every second from the starting position on the road, and then no further traffic was released while the simulation ran. The pre-set departure speeds of the 10 vehicles are listed in Table I. In the second scenario (i.e. Scenario 2), vehicles had random departure speeds and were randomly released during the simulation from the starting position on the road. The probabilities for vehicles' release are listed in Table II. Additional attributes were also set to characterise different types of vehicles (see Table III).[1] The time step set for the simulations was 0.1s and the total simulation time per simulation was 120s.

TABLE II
SCENARIO 2: RELEASE PROBABILITIES

	Passenger Vehicle	Bus	Taxi
Probability of Release [per 0.1s]	0.1	0.01	0.01

TABLE III
SCENARIOS 1 AND 2: OTHER ATTRIBUTES

	Passenger Vehicle	Bus	Taxi
Max. Speed [m/s]	70	50	60
Max. Acceleration [m/s^2]	2.6	2.2	2.8
Max. Deceleration [m/s^2]	4.5	4.0	4.6
speedFactor	1	1	1
speedDev	0.1	0.1	0.1

2) Algorithms and identifiers: Both Algorithm 1 and Algorithm 2 were implemented via a Python script, which interfaced with SUMO using TraCI. In order to better characterise our algorithms, we simulated for both Scenario 1 and Scenario 2, the case where the algorithms are *active* (i.e. provide speeds to the drivers and such speeds are followed), and the case where they are not implemented. Then, in order to make a comparison between such two cases, we recorded the following identifiers during the course of the simulations: (i) individual vehicle speeds; (ii) total fuel consumption of the network; and (iii) total particulate matter emissions.

[1]Attribute descriptions can be found in the user documentation on the SUMO website [16].

3) SUMO simulation results: Figs. 5 and 6 show the evolution of individual vehicle speeds over the course of the simulation runtimes in regards to Scenario 1. Specifically, Fig. 5 illustrates how the individual vehicles' speeds evolved in the case where the algorithms were not active, while Fig. 6 shows what occurred when the algorithms were active. Such figures clearly show that the algorithms allowed the vehicles' speeds to converge towards a common value. Note that this does not happen if the control is not active.

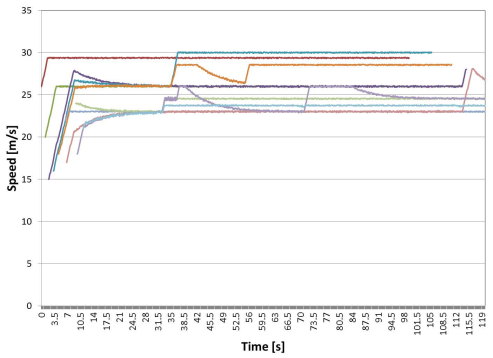

Fig. 5. Scenario 1: individual vehicles' speed evolutions without algorithms.

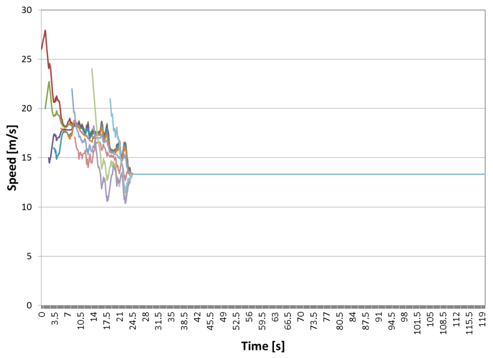

Fig. 6. Scenario 1: individual vehicles' speed evolutions with algorithms.

Analogously, Figs. 7 and 8 show the evolution of individual vehicle speeds over the course of the simulations in regards to Scenario 2. Fig. 7 demonstrates the progression of vehicles' speeds when the algorithms were not implemented, while Fig. 8 illustrates what occurred when the algorithms were applied. Again, the algorithms allowed the vehicles' speeds to converge towards a common value.

Note that every time a car enters the simulation, the vehicles automatically coordinate to achieve the common, average speed. This is due to the fact that there is no leader. We are currently developing a SAS with a leader, which will be presented in future work [17]. This work is omitted here due to page constraints.

Table IV compares total network fuel consumption and total network particulate matter emissions in regards to each of the previously described simulation runs. That is, the fuel consumption and particulate matter emissions of each

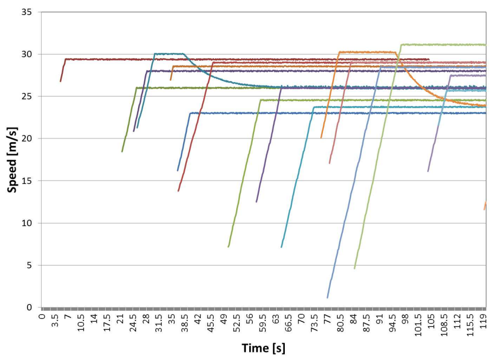

Fig. 7. Scenario 2: individual vehicles' speed evolutions without algorithms.

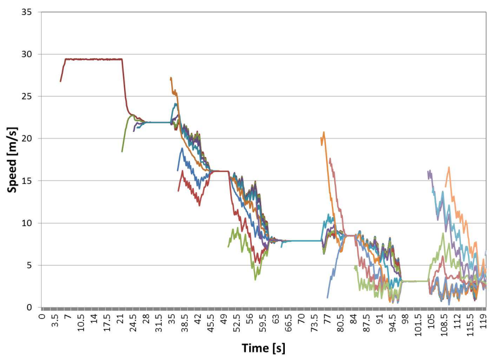

Fig. 8. Scenario 2: individual vehicles' speed evolutions with algorithms.

vehicle, at each time step, were summed to find the total network values. The result was that the activation of the algorithms reduced fuel consumption and particulate matter emissions in both Scenarios 1 and 2.

TABLE IV

ALGORITHMS SUCCESS MEASURES

Identifier	Fuel Consumption [ml]	PM$_x$ [mg]
Scenario 1 (No Algorithms)	15206	3707
Scenario 1 (Algorithms)	9134	1801
Scenario 2 (No Algorithms)	16842	4449
Scenario 2 (Algorithms)	7191	2135

B. The next step

Our next step will be to implement our speed advisory system in the hardware-in-the-loop platform described in [18] (and seen also in `https://www.youtube.com/watch?v=tCX2GLn1pnM`). The objective of the platform is to merge large-scale simulation and proof-of-concept by "embedding" a real, equipped vehicle, being driven by a real driver, into SUMO. As such, emulations consisting of the real vehicle and potentially thousands of simulated cars are able to be run in realtime, and the driver of the real vehicle is presented with an opportunity to experience first-hand what it feels like to travel in a large-scale, connected scenario, and to try out the new intelligent transportation technology that is being developed. In this way, we may gauge further how acceptable our speed advisory system is to real drivers.

V. CONCLUSIONS

In this paper, we presented a new SAS. The system is distributed and its key feature is that it makes use of noise in guiding the participating vehicles towards a common speed. Besides guaranteeing convergence of the algorithm, noise also corrupts the information seen by each of the vehicles. The effect of this is that there is no vehicle within the system knowing the exact speed of all the other vehicles. The results of this paper open up new, exciting directions for future work. Based on this work, the authors are currently working towards: (i) the development of hardware-in-the-loop test-beds to better validate our algorithms; (ii) the extension of the algorithms to consider multiple base stations; and (iii) the further development of the algorithms to drive all of the vehicles to a *desired* speed while optimising some global cost function.

REFERENCES

[1] Speed Statistics (Transport Accident Commission, Victoria State Government, Australia). (Last accessed: 23rd March 2016). [Online]. Available: `http://www.tac.vic.gov.au/road-safety/statistics/summaries/speed-statistics`

[2] S. Jamson, O. Carsten, K. Chorlton and M. Fowkes, *Intelligent speed adaptation literature review and scoping study*, Technical Report ISA-TfL D1, University of Leeds, MIRA Ltd. and Transport for London, 2006.

[3] S. Kundu, Flexible vehicle speed control algorithms for eco-driving, in *Proceedings of the 82nd IEEE Vehicular Technology Conference*, Boston, MA, USA, 2015.

[4] W. J. Schakel and B. van Arem, Improving traffic flow efficiency by in-car advice on lane, speed, and headway, *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1597-1606, 2014.

[5] R. H. Ordóñez-Hurtado, W. M. Griggs, K. Massow and R. N. Shorten, Intelligent speed advising based on cooperative traffic scenario determination, pp. 77-92, in: H. Waschl, I. Kolmanovsky, M. Steinbuch and L. del Re (Eds.), *Lecture Notes in Control and Information Sciences: Optimization and Optimal Control in Automotive Systems*, vol. 455, Springer, 2014.

[6] M. Liu, R. H. Ordóñez-Hurtado, F. Wirth, Y. Gu, E. Crisostomi and R. Shorten, A distributed and privacy-aware speed advisory system for optimizing conventional and electric vehicle networks, *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1308-1318, 2016.

[7] E. Adell, A. Várhelyi, M. Alonso and J. Plaza, Developing human–machine interaction components for a driver assistance system for safe speed and safe distance, *IET Intelligent Transport Systems*, vol. 2, no. 1, pp. 1–14, 2008.

[8] R. Gallen, N. Hautière, A. Cord and S. Glaser, Supporting drivers in keeping safe speed in adverse weather conditions by mitigating the risk level, *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1558-1571, 2013.

[9] R. Olfati-Saber and R. Murray, Consensus problems in networks of agents with switching topology and time-delays, *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[10] M. di Bernardo, D. Liuzza and G. Russo, Contraction analysis for a class of nondifferentiable systems with applications to stability and network synchronization, *SIAM Journal on Control and Optimization*, vol. 52, no. 5, pp. 3203–3227, 2014.

[11] G. Russo, M. di Bernardo and E. D. Sontag, Stability of networked systems: a multi-scale approach using contraction, in *Proceedings of the 49th IEEE Conference on Decision and Control*, Atlanta, Georgia, USA, 2010, pp. 6559-6564.

[12] D. Burbano and M. di Bernardo, Multilayer proportional-integral consensus of heterogeneous multi-agent systems, in *Proceedings of the 54th IEEE Conference on Decision and Control*, Osaka, Japan, 2015, pp. 4854-4859.

[13] X. Mao, *Stochastic Differential Equations and Applications*, Woodhead Publishing; 1997.
[14] D. Krajzewicz, J. Erdmann, M. Behrisch and L. Bieker, Recent development and applications of SUMO – Simulation of Urban MObility, *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3-4, pp. 128-138, 2012.
[15] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer and J.-P. Hubaux, TraCI: an interface for coupling road traffic and network simulators, in *Proceedings of the 11th Communications and Networking Simulation Symposium*, Ottawa, Canada, 2008, pp. 155-163.
[16] SUMO (Simulation of Urban MObility). (Last accessed: 2nd March 2016). [Online]. Available: www.dlr.de/ts/sumo/en/
[17] W. Griggs, G. Russo and R. Shorten, Lead and leaderless consensus with state obfuscation: an application to Speed Advisory Systems, in preparation.
[18] W. M. Griggs, R. H. Ordóñez-Hurtado, E. Crisostomi, F. Häusler, K. Massow and R. N. Shorten, A large-scale SUMO-based emulation platform, *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3050-3059, 2015.
[19] C. Godsil and G. Royle, *Algebraic Graph Theory*, Springer-Verlag, New York; 2001.
[20] A. F. Karr, *Probability*, Springer-Verlag, New York; 1993.
[21] V. K. Rohatgi, *An introduction to Probability Theory and Mathematical Statistics*, John Wiley & Sons; 1976.

APPENDIX

In this appendix, we provide a new result on the consensus of networks modelled by stochastic differential equations from which convergence of the SAS follows. For the sake of brevity, the full proof is omitted and will be presented elsewhere [17].

Notation

The Kronecker (or direct) product will be denoted by $\otimes$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with $\mathcal{V}$ being the set of nodes and $\mathcal{E}$ being the set of edges. We will denote by $\mathcal{N}_i$ the set of neighbours of node i. Let N be the number of nodes in the network. Then (see e.g. [19]) the Laplacian matrix associated to the graph, L, is symmetric and we will denote by λ_i, $i = 1, \ldots, N$, its eigenvalues.

Mathematical tools

Consider an n-dimensional stochastic differential equation of the form

$$dx = f(t,x)dt + g(t,x)db, \qquad (1)$$

where: (i) $x \in \mathbb{R}^n$ is the state variable; (ii) $f : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}^n$ belongs to $\mathcal{C}^2$; (iii) $g : \mathbb{R}^+ \times \mathbb{R}^n \to \mathbb{R}^n$ belongs to $\mathcal{C}$; (iv) b is a 1-dimensional Brownian motion. Throughout this paper, we will assume that for any given initial condition, (1) has a unique global solution; see e.g. [13]. We will also assume that $f(t,0) = g(t,0) = 0$ and the solution $x = 0$ will be said to be the *trivial solution* of (1). Following [20], [21], we say that a sequence of stochastic variables, $\{V_1, V_2, \ldots\}$, converges almost surely (a.s.) to the stochastic variable V if $\mathbb{P}\left(\{w : \lim_{n\to+\infty} V_n(w) = V(w)\}\right) = 1$. That is, convergence happens with probability 1 ($\mathbb{P} = 1$). We are now ready to give the following definition; see [13].

Definition 1: The trivial solution of (1) is said to be almost surely exponentially stable if, for all $x \in \mathbb{R}^n$, $\lim_{t\to+\infty} \sup \frac{1}{t} \log(|x(t)|) < 0$, $a.s.$

Let $V(t,x) \in \mathcal{C}^{1\times 2}$ (i.e. $V(t,x)$ is twice differentiable in x and differentiable in t) and let: (i) $LV(t,x) = V_t(t,x) + V_x(t,x)f(t,x) + \frac{1}{2}tr\left\{g(t,x)^T V_{xx} g(t,x)(t,x)\right\}$;(ii) $V_x = [V_{x_1}, \ldots, V_{x_n}]$; (iii) V_{xx} be the $n \times n$ dimensional matrix having as element ij $V_{x_i x_j}$ (where $V_{x_i} := \partial V(t,x)/\partial x_i$ and $V_{x_i x_j} := \partial^2 V(t,x)/\partial x_j \partial x_i$). Then, the following result from [13] holds.

Theorem 1: Assume that there exists a non-negative function $V(t,x) \in \mathcal{C}^{1\times 2}$ and constants $p > 0$, $c_1 > 0$, $c_2 \in \mathbb{R}$, $c_3 \geq 0$, such that $\forall x \neq 0$ and $\forall t \in \mathbb{R}^+$: **(H1)** $c_1 |x|^p \leq V(t,x)^p$; **(H2)** $LV(t,x) \leq c_2 V(t,x)$; **(H3)** $|V_x(t,x)g(t,x)|^2 \geq c_3 V(t,x)^2$. Then: $\lim_{t\to+\infty} \sup \frac{1}{t} \log(|x(t)|) \leq -\frac{c_3 - 2c_2}{p}$, $a.s.$ In particular, if $c_3 > 2c_2$, then the trivial solution of (1) is almost surely exponentially stable.

A result on consensus of stochastic networks

Consider a stochastic differential equation of the form

$$dx_i = \left[\sigma \sum_{j\in\mathcal{N}_i} (x_j - x_i)\right] dt + \sigma^* \sum_{j\in\mathcal{N}_i^*} (x_j - x_i)\, db, \qquad (2)$$

$x_i \in \mathbb{R}$, $x_i(t_0) = x_{i,0}$, $t \geq 0$, $i = 1, \ldots, N$. Such an equation corresponds to the dynamics of a network consisting of two layers [12]. Note that the two layers might have different topologies and that one of the layers is affected by noise (we will refer to the noise-free layer as *communication layer*, while the layer affected by noise will be termed as *noise-diffusion layer*). Network (2) can be written in compact form:

$$dX = -\sigma L X dt - \sigma^* L^* X db,$$

where $X := [x_1, \ldots, x_N]^T$, L is the Laplacian of the communication layer and L^* is the Laplacian of the noise-diffusion layer. We will denote by λ_N^* the largest eigenvalue of L^*, while λ_2^* will its algebraic connectivity.

Definition 2: Let $\bar{s} = \frac{1}{N}\sum_{i=1}^N x_{i,0}$. We say that (2) achieves stochastic consensus if $\lim_{t\to+\infty} \sup \frac{1}{t} \log(|x_i(t) - \bar{s}|) < 0$, $a.s.$, $\forall i = 1, \ldots, N$. Then, the following result holds.

Theorem 2: Assume that for network (2) the following condition is satisfied: $(\sigma^*)^2 \left((\lambda_2^*)^2 - \frac{(\lambda_N^*)^2}{2}\right) > -\sigma\lambda_2$. Then, (2) achieves complete stochastic consensus.

Sketch of the proof. The proof is based on the use of Theorem 1. Specifically, the function $V(e) = \frac{1}{2}e^T e$, with $S(t) := 1_N \otimes \bar{s}$ and $e = X - S$. Following Theorem 1, we need to show that there exists $c_2 \in \mathbb{R}$, $c_3 \geq 0$, such that $c_3 > 2c_2$.

Estimate of $LV(e)$. For this term, it is possible to show that:

$$LV(e) \leq \left(2\left(-\sigma\lambda_2\right) + (\sigma^*)^2(\lambda_N^*)^2\right) V(e).$$

Estimate of $\left|V_e(e)\tilde{G}(t,e)\right|^2$. For this term, it can be easily shown that $\left|V_e(e)\tilde{G}(t,e)\right|^2 \geq 4(\sigma^*)^2(\lambda_2^*)^2 V(e)^2$.

We can then conclude the proof by noticing that, by the hypotheses, $4(\sigma^*)^2(\lambda_2^*)^2 > 2\left(2\left(-\sigma\lambda_2\right) + (\sigma^*)^2(\lambda_N^*)^2\right)$. Therefore, by means of Theorem 1 we have that $\lim_{t\to+\infty} \sup \frac{1}{t} \log(|e(t)|) < 0$, $a.s.$, thus proving the result. ■