Dynamic Distributed Intrusion Detection for Secure Multi–Robot Systems

Adriano Fagiolini, Francesco Babboni, and Antonio Bicchi

***Abstract*— A general technique to build a dynamic and distributed intrusion detector for a class of multi–agent systems is proposed in this paper, by which misbehavior in the motion of one or more agents can be discovered. Previous work from the authors has focused on how to distinguish the behavior of a misbehaving agent in a completely distributed way, by developing a solution where agents act as local monitors of their neighbors and use locally sensed information as well as data received from other monitors at a particular time. In this work, we improve the system detection capability by allowing monitors to use information collected at different instants and thus realizing a dynamic state observer that is valid for any system in the considered class. Finally, we show through simulations the effectiveness of the proposed solution for a case study.**

I. Introduction

In the last decades, robotics has undergone a gradual yet constant migration of research interests from monolithic systems with a unique robot to distributed multi–agents composed of several semi–autonomous robots. A similar course happened earlier in computer science, where distributed algorithms were developed to provide better solutions for classical decision problems [1], [2]. Such multi–agents are normally meant to be used in application scenarios that lack of a centralized infrastructure, and where agents are not secured from malicious intervention. It is then reasonable to expect that one or more of these agents might be tampered, hijacked, or entered into the system as e.g. to degrade its QoS, or to compromise its safety. The term *intruders* will be used to denote these malicious agents that are not just faulty systems, in the same way malicious software is commonly denoted in computer science [3], [4]. As a matter of fact, the actual achievement of the system goal is theoretically guaranteed only under the hypothesis that all agents harmoniously act and cooperate [5]. This motivates the emerging interest toward techniques that robustify existing multi–agent systems by detecting the presence of intruders in various different settings [6]–[9].

In this context, we consider a class of multi–agent systems, where agent cooperation is obtained by sharing a set of rules. More precisely, we assume that agents are assigned with different tasks that require their motion within a shared physical environment, but they are also supposed to cooperate as to guarantee a given desirable system property. We further assume that cooperation rules are encoded as *decentralized logical conditions*, i.e. each agent plans its motion based on its own state, and on logical conditions on the state of only other agents in a suitable neighborhood. We have shown that this class of systems has a hybrid dynamics [6], [10]. Our aim is to provide a general technique that allows an Intrusion Detection System (IDS) to be automatically designed and built so as to discover motion misbehavior of any agent. We require the technique to be applicable for any system in the considered class, and to be distributed due to the absence of a centralized infrastructure.

In previous work, we have already investigated the problem and partially solved it. First, we proposed a distributed solution where each agent monitors all the other agents that lie within a safety region from it and tries to classify their behavior by using only its own sensors [6]. In particular, the hybrid model of a given target agent can be "inverted" by associating the above mentioned logical conditions to the set of configurations where these conditions are satisfied. Then, by measuring two consecutive states of the target agent's motion, estimates of its neighbors can be statically computed, consisting of continuous sets. In [11], we proposed an implementation of such a local monitor that can be run irrespectivly of the current neighborhood of the target agent, and of the visibility condition of the monitor itself. In particular, we assumed ideal sensors that can precisely read the state of every agent laying within a maximum distance and that are not hidden by other agents.

The detection capability of this base local monitor and hence of the underlying IDS can be improved by proceeding toward two different directions. On the one hand, we investigated whether and how monitor communication can be used, and we have developed a *set–valued consensus protocol* by running which agents can reach an agreement on the cooperation or uncooperation of a common neighbor [10]. Such a monitor agreement is a mandatory step before starting any emergency and escape maneuver. Furthermore, in such malicious scenarios, it is reasonable to assume that some monitors may send false information either to justify their incorrect behaviors or blame other cooperative agents. In [12], we showed that each local monitor can be supported by a distributed message validation mechanism that allows any false data to be discarded based on measurement redundancy.

On the other hand, another improving direction aims at providing each local monitor with the ability to dynamically estimate the state of all neighbors of the target agent, which is the subject of the present work. In this vein, we will show that the dynamic version of the intrusion detection problem can be solved as soon as tools are available for

A. Fagiolini, F. Babboni, and A. Bicchi are with the Interdepartmental Research Center "E. Piaggio" of the Università di Pisa, Italy, a.fagiolini@ing.unipi.it, francesco.babboni@gmail.com, bicchi@ing.unipi.it.

the observability computation of any system in the considered class. To this aim, we focused on the flourishing literature on hybrid systems. During the last 15 years, this research community has intensively studied may aspects of such systems. Reachability computation is at present one of the most investigated properties for these systems, and indeed various significant results are available for linear systems [13], affine systems [14], and some other classes of systems [15]–[18]. More recently developed is the literature on the dual problem of *observability computation* for hybrid systems. Very useful works are already available such as [19]–[22]. However, it is true that the literature is missing of a general result that remains valid for any system in a wide class. In this vein, we would like to adopt an approach similar to e.g. the one in [23], where a general framework, ARIADNE, allows reachability computation for a wide class of systems. More precisely, we aim at presenting a general procedure to state estimation for any system in the considered class. In our case, the fact that we have set–valued measurements makes the problem more difficult and yet very challenging [24]. The main contributions of the present work are indeed the following two. We first show that the dynamic intrusion detection problem is reducible to observability computation for hybrid systems, and then we provide the general framework of an algorithm for doing this.

II. Problem Statement

We consider robotic applications requiring *motion coordination* in a set of n agents, $\mathcal{A}_1, \ldots, \mathcal{A}_n$. More precisely, considered agents are robots running pre–assigned tasks that require them to move within a common physical environment, or world $\mathcal{W}$. Every robot is described by a vector q_i belonging to a configuration space $\mathcal{Q}$. Then, given a desirable system property, as e.g. the ability to avoid agent collisions, or dead– and live–locks, we assume that a suitable coordination strategy has been designed and encoded into a set $\mathcal{R}$ of decentralized logical *rules* to which agents are supposed to adhere during their motion. According to the set of rules $\mathcal{R}$, agents can perform at any instant t one of κ actions, or *motion maneuvers*, $\Sigma = \{\sigma^1, \sigma^2, \ldots, \sigma^\kappa\}$, and has to change from a current maneuver to another one whenever one of a set of ν logical conditions, or *events*, $E = \{e^1, e^2, \ldots, e^\nu\}$ depending on a suitable agent's neighborhood, called *influence set*, $\mathcal{I}_i(t)$ occurs. Then, from a logical point of view, each agent $\mathcal{A}_i$ is also assigned with a discrete variable $\sigma_i(t) \in \Sigma$ that represents its current maneuver. In [6], [10], we have shown that these types of systems, where agents have a physical dynamics, but interact according to *event–based* cooperation rule sets $\mathcal{R}$, can be modeled as hybrid systems :

$$\dot{q}_i(t) = \mathcal{H}(q_i(t), I_i(t)) \,, \tag{1}$$

where $\mathcal{H} : \mathcal{Q} \times \mathcal{Q}^p \to T_\mathcal{Q}$, $T_\mathcal{Q}$ is the space tangent to $\mathcal{Q}$, $I_i(t) = \{q_{i_1}(t), \ldots, q_{i_p}(t)\}$, and $i_1, \ldots, i_p$ are the indices of agents in $\mathcal{I}_i(t)$. Under this view, we will consider $q_{i_1}(t), \ldots, q_{i_p}(t)$ as *inputs* of model $\mathcal{H}$ and $q_i(t)$ as its

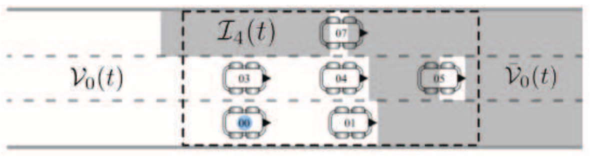


Fig. 1. Partition of the influence set $\mathcal{I}_4(t)$ of agent $\mathcal{A}_4$ w.r.t. agent $\mathcal{A}_0$'s visibility $\mathcal{V}(q_0(t), I_4(t))$.

output. An agent that follows the cooperation rules $\mathcal{R}$ is said to be *$\mathcal{R}$–compliant*.

As stated above, to achieve the system's goal and guarantee the desired system property all agents must adhere to the cooperation rule set $\mathcal{R}$. Therefore, it is essential to be able to *detect and isolate* any uncooperative agent. Our approach to solve the problem requires that each agent participate in the *intrusion detection function*. However, this poses some difficulty since agents know only partially a target agent's input, as they have *limited line–of–sight visibility*. Indeed, the challenge of a robot acting as a decentralized monitor is to distinguish a faulty or malicious robot in its neighborhood from a correctly cooperating robot whose actions may be influenced by other robots out of the monitor's range. We conveniently can introduce a *visibility map* $\mathcal{V}(q_h, I_i)$ as a nonlinear function that, given the configuration of the monitoring agent $\mathcal{A}_h$, returns the configurations in I_i that can be "seen" from the agent itself. Then, the influence set $\mathcal{I}_i(t)$ of agent $\mathcal{A}_i$ can be partitioned w.r.t. $\mathcal{A}_h$ into a known region $\mathcal{I}_i^{obs}(t)$ and an unknown one $\mathcal{I}_i^{unobs}(t)$ (see e.g. Fig 1). Indeed we have:

$$\mathcal{I}_i(t) = \mathcal{I}_i^{obs}(t) \cup \mathcal{I}_i^{unobs}(t) \,. \tag{2}$$

In this vein, previous work has addressed the following:

Problem 1: Given agent $\mathcal{A}_i$'s (hybrid) motion model $\mathcal{H}$, the partition of its influence set $\mathcal{I}_i(t)$ in Eq. 2 w.r.t. agent $\mathcal{A}_h$'s visibility $\mathcal{V}_h$, and n_o configurations $q_{i_1}(t)_h, \ldots, q_{i_{n_0}}(t)_h \in \mathcal{I}_i^{obs}(t)$ of known neighbors of agent $\mathcal{A}_i$, determine, if it exists, a choice of $p - n_o$ configurations $\hat{q}_{i_{n_o+1}}(t), \ldots, \hat{q}_{i_p}(t) \in \mathcal{I}_i^{unobs}(t)$ such that the expected motion

$$\begin{aligned}\tilde{q}_i(t) \quad &= \quad q_i(t_k) + \textstyle\int_{t_k}^{t} \mathcal{H}(q_i(\tau), q_{i_1}(\tau), \ldots, \\ &\qquad q_{i_{n_0}}(\tau), \hat{q}_{i_{n_0+1}}(\tau), \ldots, \hat{q}_{i_p}(\tau))\, d\tau \,,\end{aligned}$$

equals the measure one, i.e. $\tilde{q}_i(t) = q_i(t)$ for all $t \in T_k$.

Solving this problem is in general a hard task due to the nonlinear and differential nature of the motion model $\mathcal{H}$. It basically requires that an *unknown input observer* (UIO) $\mathcal{H}^\dagger$ of the hybrid model is built. Furthermore, a direct approach for the computation of such a UIO leads to find ad–hoc solutions for specific cases. However, we showed in [6], [10], [11] how this can be avoided for the considered class of robotic multi–agent systems. The reader may assume the existence of a systematic procedure to build a UIO, $\mathcal{H}^\dagger$, s.t.

$$(\hat{q}_{i_{n_o+1}}(t), ..., \hat{q}_{i_p}(t)) = \mathcal{H}^\dagger(q_i(t), q_{i_1}(t), .., q_{i_{n_0}}(t)) \,, \tag{3}$$

where $\hat{q}_{i_m}(t)$ for $m = n_o + 1, \ldots, p$ are continuous sets estimating configurations of agents in $\mathcal{I}_i^{unobs}(t)$ that can explain the validated motion $q_i(t)$ of agent $\mathcal{A}_i$.

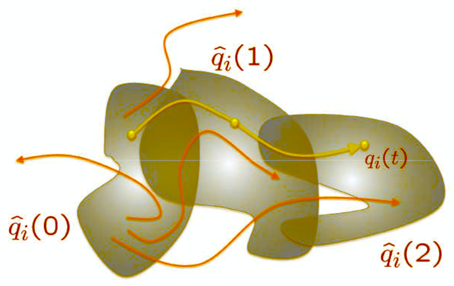


Fig. 2. State observability with set–valued measurement.

By using this mechanism, agent $\mathcal{A}_h$ may decide on the cooperativeness b_{hi} of agent $\mathcal{A}_i$: as long as a choice for $\hat{q}_l$ exists, agent $\mathcal{A}_i$ can be considered as possibly cooperative or $\mathrm{uncertain}$ (as a matter of fact, $\mathcal{A}_i$ can not verify the correctness of these estimates), but if no values for these estimates exist, agent $\mathcal{A}_i$ is considered as $\mathrm{uncooperative}$.

In this work, we want to extend the capability of this *static local monitor*. To this aim, we consider the following:

Problem 2: Given a set of n agents with continuous dynamics f, and a set $\mathcal{R}$ of cooperation rules, find an automatic synthesis procedure to build a *local monitor* that is able to *dynamically* detect motion misbehavior of some neighboring agents, by using measures taken at different times.

III. Prediction/Correction Scheme for Dynamic State Estimation with Set–Valued Measures

In this section we describe the mechanism that allows a generic agent $\mathcal{A}_h$ to dynamically estimate the current state ξ_i of the neighborhood I_i of agent $\mathcal{A}_i$. At a generic observation instant t_k, the instantaneous estimate of ξ_i is:

$$\xi_i = \{q_i, q_{i_1}, \ldots, q_{i_{n_o}}, \hat{q}_{i_{n_o+1}}, \ldots, \hat{q}_{i_p}\} \qquad (4)$$

where $q_{i_1}, \ldots, q_{i_{n_o}}$ are the configurations of visible agents, and $\hat{q}_{i_{n_o+1}}, \ldots, \hat{q}_{i_p}$ are configurations of other agents reconstructed by the UIO of Eq. 3. Note that its components are *polyhedral sets* representing possible configuration of $\mathcal{A}_i$'s neighbors. Then, Problem 2 involves finding a solution to the following one:

Problem 3 (Set–Valued State Estimation): Given a sequence of κ estimates, $\xi_i(t_0), \ldots, \xi_i(t_\kappa)$, taken at successive times, find the smallest set $\xi^*(t_k)$ of all points that are $\mathcal{R}$–compliant, i.e. the set of points $\bar{q}_i \in \xi_i(t_\kappa)$ for which there exists a *chain of points*, $\bar{q}_i(t_0), \ldots, \bar{q}_i(t_{\kappa-1})$ s.t.

$$\begin{cases} \bar{q}_i(t_1) = \phi_{\mathcal{H}}(\bar{q}_i(t_0))\,, \\ \bar{q}_i(t_2) = \phi_{\mathcal{H}}(\bar{q}_i(t_1))\,, \\ \vdots \\ \bar{q}_i(t_\kappa) = \phi_{\mathcal{H}}(\bar{q}_i(t_{\kappa-1}))\,, \end{cases}$$

where $\phi_{\mathcal{H}}(\cdot)$ is the solution of the ordinary differential equation in Eq. 1. (see Fig. 2 and recall chain–rechability from [17]).

Let us first present a general procedure by which an estimate of the current neighborhood state can be iteratively computed (see Algorithm 1). This estimate is updated as soon as new measurements of agent $\mathcal{A}_i$'s neighborhood are available. The essence of the algorithm consists of a prediction/correction step: at line 13, a forward projection Proj$(\xi_i(k-1))$ of the latest estimate is combined with the new measurement $U_i(k)$ by set–intersection.

Algorithm 1 is based on the availability of operator Proj that computes a forward–projection of the continuous input set, according to the hybrid model $\mathcal{H}$. An implementation of the *set projector* Proj would solve Problem 3, but its implementation can be easily found only for linear systems. This is not shown here for space limitation and leverages on the fact that, in the linear setting, exact system solution is known, and polyhedral sets map to other polyhedral sets. The fact that, for a generic nonlinear system, a closed form of the system evolution is unknown can be overcome by numerical integration. However, the problem with nonlinear ordinary differential equations (ODE) is that convex sets map to complicated geometric shapes. Furthermore, starting by a *connected set*, the nonlinear system flow may produce "holes" inside the set [25]. It is worth noting that this problem does not arise in reachability computation, where it is sufficient to determine if a point is eventually reached. Then, in our case, the evolutions of all points in the original set are to be "tracked". Due to the difficulty in processing an infinite number of points, we reduces to compute conservative approximations of the state estimate. We first give the following:

Definition 1: Given a set ξ, an *ε–overapproximation* of ξ is a set ξ_ε of points such that:

- $\xi \subseteq \xi_\varepsilon$, and
- for all $q_\varepsilon \in \xi_\varepsilon$, there exists a point $q \in \xi$ s.t.

$$\|q - q_\varepsilon\| \leq \varepsilon\,,$$

where $||\cdot||$ is the Haussdorf norm.

Then, we want to solve the following:

Problem 4 (ε–observability): Under the hypotheses of Problem 3, find an ε–overapproximation of the $\xi^*(t_k)$.

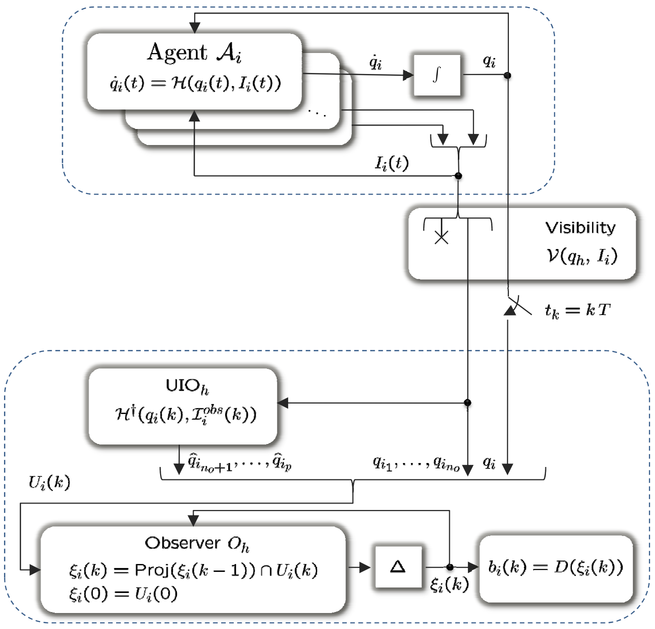
Let us first recall the following:

Definition 2 (Lipschitz function): A real–valued function f defined on a subset $\mathcal{K}$ of the real numbers $f : \mathcal{K} \subseteq \mathbb{R} \to \mathbb{R}$ is called *Lipschitz continuous*, or is said to satisfy a *Lipschitz condition* if there exists a constant $L \geq 0$ such that for all $x_1, x_2 \in \mathcal{D}$

$$\|f(x_1) - f(x_2)\| \leq L\|x_1 - x_2\|.$$

A general implementation of operator Proj can then be obtained as follows (see Algorithm 2). The algorithm receives as input the current state ξ and produces the predicted state ξ^+. It proceeds by considering a grid of points, ξ_{grid}, with a suitable mesh size $\mu \in \mathbb{R}$. By the Fundamental Inequality Theorem [26], a Lipschitz system always admits a conservative mesh size μ for which an ε–overapproximation of its trajectories can be computed. Indeed, given a generic initial system configuration $q(0)$ and another initial configuration $\bar{q}(0)$, such that

$$\|q(0) - \bar{q}(0)\| \leq \mu\,,$$



Agent $\mathcal{A}_i$
$\dot{q}_i(t) = \mathcal{H}(q_i(t), I_i(t))$
$\dot{q}_i$
$\int$
q_i
$I_i(t)$
Visibility
$\mathcal{V}(q_h, I_i)$
$t_k = kT$
UIO$_h$
$\mathcal{H}^\dagger(q_i(k), \mathcal{I}_i^{obs}(k))$
$\hat{q}_{i_{no}+1}, \dots, \hat{q}_{i_p}$
$q_{i_1}, \dots, q_{i_{no}}$
q_i
$U_i(k)$
Observer O_h
$\xi_i(k) = \text{Proj}(\xi_i(k-1)) \cap U_i(k)$
$\xi_i(0) = U_i(0)$
Δ
$\xi_i(k)$
$b_i(k) = D(\xi_i(k))$

Fig. 3. Prediction/correction scheme of a dynamic state estimator.

then, for any sampling time T, it holds:

$$\|q(T) - \bar{q}(T)\| \leq \mu\, e^{L\,T} \,,$$

Thus, the choice $\mu = \varepsilon / e^{L\,T}$ guarantees that a ball centered at $q(T)$ with radius ε includes all system trajectories starting from points lying within a maximum distance μ from $q_i(0)$. Indeed we trivially have $\|q(T) - \bar{q}(T)\| \leq \varepsilon$. Therefore, to realize the Proj operator, it is sufficient to compute a forward projection of all points in the original grid (see line 4), and then to enlarge these points by a sufficiently large ball (lines 5–8). The radius of the ball can be computed based on the local value of the Lipschitz constant. This reminds the idea of lifting faces of a polyhedra by local behavior of the system [15]. These balls are iteratively added to the predicted set ξ^+.

The algorithm is complete, i.e. its finite–time termination can be guaranteed due to the Lipschitz condition. Finally, the complete scheme of the dynamic monitor is depicted in Fig. 3. It is worth noting that this approach requires some computational effort, but its applicability to set–valued observation is quite wide. The only requirement is on the Lipschitz condition of the system. Hence, we believe that this work goes into the same direction of other tools as e.g. ARIADNE, and it can be used to realize an intrusion detection system for a large class of dynamical systems.

IV. Application

A. An Automated Highway

Consider n mobile agents that are traveling along a highway with different maximum speed and different final positions. Agents are supposed to cooperate according to the common driving rules in order to avoid collisions. Informally, the rule set is $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$, where:

Algorithm 1 Dynamic Monitor

```
Inputs: $\mathcal{H}$, $\varepsilon$.
Outputs: Cooperativeness $b_i$, Estimated neighborhood $\xi_i$.
 1: for all times $k = 1, 2, \dots$ do
 2:     Compute $\mathcal{N}_h = \{q_1, q_2, \dots\}$ = getEnvironment()  ◁ get all visible agents
 3:     for all new agents $q_i \in \mathcal{N}_h$ do
 4:         $\xi_i(k-1) = \mathcal{Q}^p$  ◁ initialize neighborhood's state
 5:     end for
 6:     for all agents $q_i \in \mathcal{N}_h$ do
 7:         $b_i = D(\xi_i(k))$  ◁ determine agents' cooperativeness
 8:         Compute $\{q_{i_1}, \dots, q_{i_{n_o}}\}$ = neighbors($q_i, \mathcal{N}_h$)
 9:         Set $I_i^{obs} = \{q_{i_1}, \dots, q_{i_{n_o}}\}$
10:         Compute $\{\hat{q}_{i_{n_o+1}}, \dots, \hat{q}_{i_p}\}$ = uio($q_i, I_i^{obs}$)
11:         Set $I_i^{unobs} = \{\hat{q}_{i_{n_o+1}}, \dots, \hat{q}_{i_p}\}$
12:         Set $U_i(k) = \{q_i, I_i^{obs}, I_i^{unobs}\}$
13:         Set $\xi_i(k)$ = Proj($\xi_i(k-1), \mathcal{H}, \varepsilon, T) \cap U_i(k)$
14:     end for
15: end for
16: Simplify $\xi^+$
```

Algorithm 2 ε–overapproximation of set forward projection

```
Inputs: $\xi$, $\mathcal{H}$, $\varepsilon$, $T$.
Outputs: Projected set $\xi^+$.
 1: $\xi_{grid}$ = grid($\xi, \varepsilon$)
 2: $\xi^+ = \emptyset$  ◁ generate grid points
 3: for all points $q_i \in \xi_{grid}$ do
 4:     $q_i^+$ = integrate($\mathcal{H}, q_i, T$)  ◁ compute forward projection of $q_i$ by numerical integration
 5:     $\sigma_i = \mathcal{D}(q_i)$  ◁ compute current maneuver
 6:     $L$ = lipschitz($q_i, \sigma_i$)  ◁ compute local estimate of system's Lipschitz constant
 7:     $\mu = \varepsilon / e^{L\,T}$
 8:     $\hat{q}_i^+$ = ball($q_i^+, \mu$)  ◁ compute point overapproximation
 9:     Add $\hat{q}_i^+$ to $\xi^+$.
10: end for
```

- $r_1 \overset{\text{def}}{=}$ “proceed at the maximum speed along the right-most free lane when possible (fast maneuver)”;
- $r_2 \overset{\text{def}}{=}$ “if a slower vehicle proceeds in front on the same lane, then overtake the vehicle if the next lane on the left is free (left maneuver), or reduce the speed (slow maneuver) otherwise”;
- $r_3 \overset{\text{def}}{=}$ “as soon as the next lane on the right becomes free, change to that lane (right maneuver)”;
- $r_4 \overset{\text{def}}{=}$ “overtaking any vehicle on the right is forbidden”.

The generic agent chooses one of these maneuvers based on events on its neighborhood. Agent $\mathcal{A}_i$'s configuration is $q_i(t) = (x_i(t), y_i(t), \theta_i(t), v_i(t))$ and has the continuous–time unicycle–like dynamics f:

$$\begin{cases} \dot{x}_i(t) = v_i(t)\, \cos(\theta_i(t))\,, \\ \dot{y}_i(t) = v_i(t)\, \sin(\theta_i(t))\,, \\ \dot{\theta}_i(t) = \omega_i(t)\,, \\ \dot{v}_i(t) = a_i(t)\,, \end{cases}$$

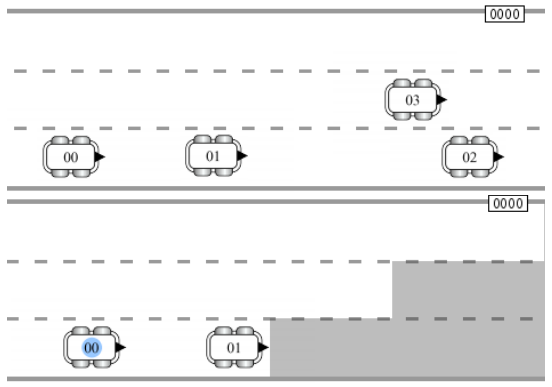


Fig. 4. Initial configuration of the simulation run, and initial visibility condition of monitoring robot 0 (robots 2 and 3 are hidden).

where $a_i(t)$ and $\omega_i(t)$ are linear acceleration and angular velocities, respectively. According to the set $\mathcal{R}$, the maneuver $\sigma_i(t)$ of the i–th robot may take value on the set $\Sigma = \{\text{fast}, \text{left}, \text{right}, \text{slow}\}$. The dynamics of the agent maneuver is omitted here for the sake of space, but can be found in [10].

B. Detection of misbehaving vehicles

In this section, we first want to show that the dynamic monitor is indeed able to provide better state estimation of the neighborhood of a given agent, and then we show that it is able to discover motion misbehaviors that were undetectable with the static monitor proposed in [6], [10]. The reader may refer to the site $\text{http://www.piaggio.ccii.unipi.it/\~{}fagiolini/icra2009}$ for the complete simulation run.

To this aim, consider a simulation run with 4 vehicles, where robot 0 is a vehicle that is monitoring robot 1 (see Fig. 4). In the first part of the simulation, robot 1 is following the cooperation rules $\mathcal{R}$. Thus, it will remain in the right–most lane and perform a fast maneuver as long as robot 2 is sufficiently far. This happens until simulation step $t = 69$. During this time, monitoring robot 0 will estimate the absence of a vehicle in front of robot 1 (green region in Fig. 5–a). At simulation step $t = 70$, robot 1 will correctly start overtaking robot 2 (its maneuver changes, from fast to left). The static monitor presented in [10] will estimate the presence of a vehicle in front of vehicle 1 (red region in Fig. 5–b), but it will "forget" the information collected at the previous instant. On the contrary, the dynamic monitor will be able to compute a projection of the green region estimated at $t = 69$ and then will intersecting with the red region estimated at $t = 70$ (see Algorithm 1 and 2). A better estimation (narrower red region) of the neighborhood of robot 1 can be found in this way (see Fig. 6).

As the simulation continues, robot 1 approches the second lane and then robot 3 becomes hidden. In the meanwhile, robot 2 becomes visible since it is not anymore hidden by robot 1. At simulation time $t = 89$, monitoring robot 0 has estimated two projected red regions (Fig. 7): the lower one is verified by robot 2, whereas the upper one is a projection of

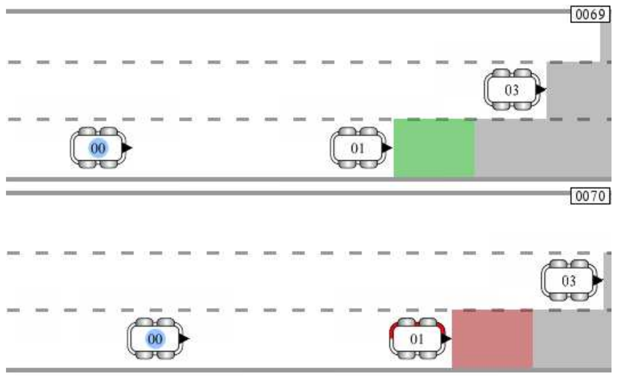


Fig. 5. Estimation of robot 1's neighborhood made by a static monitor embedded on robot 0.

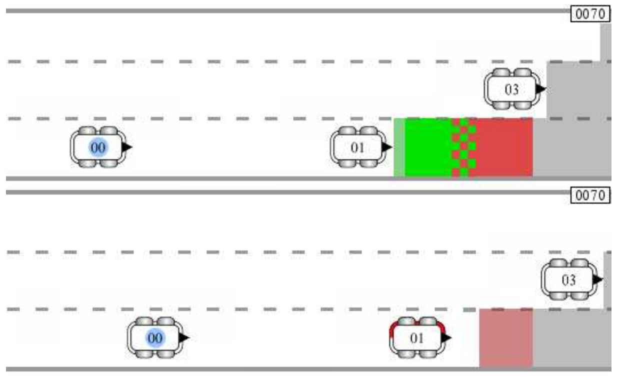


Fig. 6. Estimation of robot 1's neighborhood made by a dynamic monitor embedded on robot 0. The projection of a green region detected at time $t = 69$ is intersected with a red region detected at $t = 70$.

the last measured configuration of robot 3 that has become hidden. Until this time, robot 1 has correctly following the cooperation rules $\mathcal{R}$, and the monitoring process has confirmed this.

Suppose that robot 1 starts from now to misbehave by not respecting the safety distance from the preceding robot 3. As it can be seen in Fig. 8–a, monitoring robot 0 detects a green region representing a free portion of the second lane, due to robot 1's behavior. At the same time, it will continue to estimate the configuration of robot 3. Due to the fact that robot 3 has a lower maximum speed, the red region representing its possible configuration will be completely included into a green region at $t = 153$ (8–b). At this time the hypothesis of a green region in front of robot 1 becomes inconsistent with the presence of robot 3, and robot 1 can be deemed as uncooperative. Again, this misbehavior would have not been discovered by a static monitor, since it was not able to compute an estimation of robot 3's configuration.

V. Conclusion

The definition of a general technique to build a dynamic and distributed intrusion detector for a class of multi–agent systems was proposed in this work. A completely distributed algorithm was proposed in previous work, whereas in this paper the system detection capability is improved by allowing

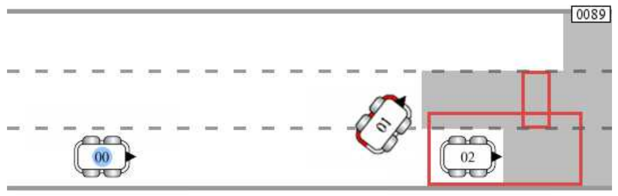


Fig. 7. Monitoring robot 0 has estimated two projected red regions: the lower is verified by robot 2, whereas the upper one is a projection of the last measured configuration of robot 3 that has become hidden.

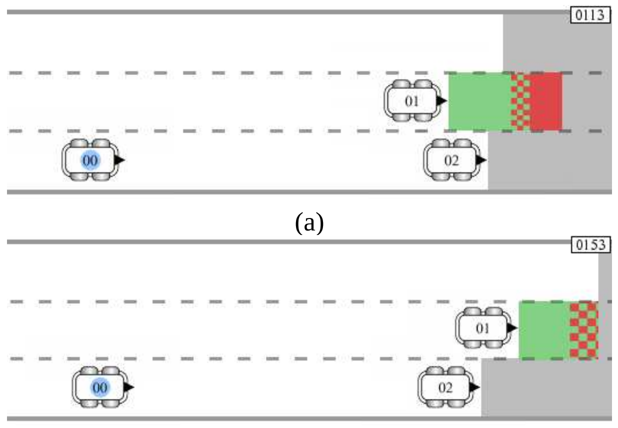


Fig. 8. Monitoring robot 0 detects a green region representing a free portion of the second lane, due to robot 1's behavior, but also a red region due to robot 3. When the two regions become inconsistent robot 1 can be deemed as uncooperative.

monitors to use information collected at different instants and thus realizing a dynamic state observer that is valid for any system in the considered class. Future work will investigate issues related to communication and synchronization of the monitors in a real experimental implementation.

VI. Acknowledgment

Authors wish to thank Claire Tomlin, Richard M. Murray, and Domitilla Del Vecchio for useful discussion on state estimation for hybrid systems, Joao P. Espanha and Magnus Egerstedt for their useful comments on the problem formulation as well as state estimation problem. This work has been partially supported by the European Commission with contract FP7-IST-2008-224428 "CHAT - Control of Heterogeneous Automation Systems: Technologies for scalability, reconfigurability and security", and with contract number FP7-2007-2-224053 CONET, the "Cooperating Objects Network of Excellence". This work has been also partially supported by Research Project 2007 funded by Cassa di Risparmio di Livorno, Lucca e Pisa.

References

[1] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *Automatic Control, IEEE Transactions on*, vol. 31, no. 9, pp. 803–812, 1986.

[2] D. Bertsekas and J. Tsitsiklis, "A survey of some aspects of parallel and distributed iterative algorithms," 1989.

[3] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*. Prentice Hall, 2004.

[4] S. Bosworth, M. Kabay, and I. NetLibrary, *Computer Security Handbook*. John Wiley & Sons, 2002.

[5] J. Blum and A. Eskandarian, "The threat of intelligent collisions," *IT Professional*, vol. 6, no. 1, pp. 24–29, Jan.-Feb. 2004.

[6] A. Fagiolini, G. Valenti, L. Pallottino, G. Dini, and A. Bicchi, "Decentralized Intrusion Detection For Secure Cooperative Multi-Agent Systems," in *Proc. 46th IEEE Conf. on Decision and Control*, 2007, pp. 1553–1558.

[7] F. Pasqualetti, A. Bicchi, and F. Bullo, "Distributed intrusion detection for secure consensus computations," in *Proc. 46th IEEE Conf. on Decision and Control*, New Orleans, LA, USA, 12–14 December 2007, pp. 5594–5599.

[8] M. Franceschelli, M. Egerstedt, and A. Giua, "Motion Probes for Fault Detection and Recovery in Networked Control Systems," *American Control Conference, Seattle, WA, June*, 2008.

[9] M. Ji and M. Egerstedt, "Observability and estimation in distributed sensor networks," *Proc. 46th IEEE Conf. on Decision and Control*, pp. 4221–4226, 2007.

[10] A. Fagiolini, M. Pellinacci, G. Valenti, G. Dini, and A. Bicchi, "Consensus based Distributed Intrusion Detection for Multi Robot Systems," in *Proc. IEEE International Conf. on Robotics and Automation*, 2008, pp. 120–127.

[11] A. Fagiolini, G. Valenti, L. Pallottino, G. Dini, and A. Bicchi, "Local Monitor Implementation for Decentralized Intrusion Detection in Secure Multi Agent Systems," in *Proc. 3rd Annual IEEE Conf. on Automation Science and Engineering*, 2007, pp. 454–459.

[12] A. Fagiolini, A. Bicchi, G. Dini, and I. Savino, "Tolerating malicious monitors in detecting misbehaving robots," in *IEEE International Workshop on Safety, Security, and Rescue Robotics*, Tohoku University Aobayama Campus, Sendai, Japan, 2008.

[13] A. Girard and C. L. Guernic, "Zonotope/Hyperplane Intersection for Hybrid Systems Reachability Analysis," in *Hybrid Systems: Computation and Control, Lecture Notes in Computer Science*, 2008, pp. 215–228.

[14] A. Girard, "Reachability of Uncertain Linear Systems Using Zonotopes," in *Hybrid Systems: Computation and Control, Lecture Notes in Computer Science*, 2005, pp. 291–305.

[15] T. Dang and O. Maler, "Reachability analysis via face lifting," in *Hybrid Systems: Computation and Control, Lecture Notes in Computer Science*. Springer-Verlag, 1998, pp. 96–109.

[16] E. Asarin, T. Dang, and A. Girard, "Reachability Analysis of Nonlinear Systems Using Conservative Approximation," in *Hybrid Systems: Computation and Control, Lecture Notes in Computer Science*, 2003, pp. 20–35.

[17] P. Collins, "Continuity and Computability of Reachable Sets," in *Theoretical Computer Science*, vol. 341, 2005, pp. 162–195.

[18] D. D. Vecchio and R. Murray, *Complexity Management in the State Estimation of Multi–Agent Systems*. John Wiley & Sons, Ltd, 2001.

[19] R. Vidal, R. Chiuso, S. Soatto, and S. Sastry, "Observability of linear hybrid systems," in *In Hybrid Systems: Computation and Control, LNCS*. Springer Verlag, 2003, pp. 526–539.

[20] M. Babaali and M. Egerstedt, "Observability of switched linear systems," in *Hybrid Systems: Computation and Control, Lecture Notes in Computer Science*. Springer Verlag, 2004, pp. 48–63.

[21] M. Babaali and G. J. Pappas, "Observability of switched linear systems in continuous time," in *HSCC*, 2005, pp. 103–117.

[22] A. D'Innocenzo, "Observability and Temporal Properties of Hybrid Systems: Analysis and Verification," Ph.D Thesis, University of L'Aquila, Department of Electrical Engineering and Computer Science, 2006.

[23] L. Benvenuti, D. Bresolin, A. Casagrande, P. Collins, A. Ferrari, E. Mazzi, T. Villa, and A. Sangiovanni-Vincentelli, "Reachability Computation for Hybrid Systems with Ariadne," in *Proc. of the 17th IFAC World Congress*, 2008.

[24] J. Aubin and H. Frankowska, *Set-Valued Analysis*. Birkhäuser, 1990.

[25] R. Alur, S. Kannan, and S. L. Torre, "Polyhedral flows in hybrid automata," *Form. Methods Syst. Des.*, vol. 24, no. 3, pp. 261–280, 2004.

[26] A. Bressan and B. Piccoli, *Introduction to the Mathematical Theory of Control*. American Institute of Mathematical Sciences, Applied Math Series Vol. 2, 2007.