Asynchronous Consensus-Based Distributed Target Tracking

Silvia Giannini, Antonio Petitti, Donato Di Paola, Alessandro Rizzo

Abstract—This paper addresses the problem of distributed target tracking, performed by a network of agents which update their local estimates asynchronously. The proposed solution extends and improves an existing consensus-based distributed target tracking framework to cope with real-world settings in which each agent is driven by a different clock. In the consensus-based target tracking framework, it is assumed that only a few agents can actually measure the target state at a given time, whereas the remainder is able to perform a model-based prediction. Subsequently, an algorithm based on max-consensus makes all the agents agree, in finite time, on the best available estimate in the network. The limitations imposed by the assumption of synchronous updates of the network nodes are here overcome by the introduction of the concept of asynchronous iteration. Moreover, an event-based approach makes for the lack of a common time scale at the network level. Furthermore, the asynchronous scenario can be derived as a special case of the asynchronous setting. Finally, numerical simulations confirm the validity of the approach.

I. INTRODUCTION

In recent years, target tracking problems have attracted a growing interest in the field of distributed cooperation of autonomous agents, for their innovative applications in defense, surveillance, security, search and rescue, and environmental monitoring, to name a few [1]–[5]. The aim of distributed target tracking is to produce a unique estimate of the state of a moving object of interest, defined as target, starting from local estimates produced by a pool of networked agents, according to some optimality criterion. One of the most common approaches to achieve this goal is sensor fusion, which can be performed through centralized, hierarchical, or distributed approaches [6].

In this paper, we deal with a particular class of distributed target tracking approaches, in which the problem is formulated as a network consensus one, computation is fully distributed, and dedicated fusion centers are not present [7]–[9]. All network nodes are functionally identical, thus enhancing robustness, redundancy, and interchangeability. Several distributed consensus-based algorithms for target tracking assume that agents are able to synchronously estimate and transmit their local information to neighboring agents [10], [11]. On the other hand, considering asynchronous updates would lead to the definition of more realistic strategies, in which issues like heterogeneity, time-varying communication delays and packet dropouts must be properly described and tackled. To this aim, results from the theory of asynchronous systems are useful to define effective models for multi-agent systems in real-world applications, where agents do not have access to a global clock, and messages used for synchronization could be subjected to unknown delays [12].

The proposed asynchronous Consensus-based Distributed Target Tracking (aCDTT) algorithm is the asynchronous implementation of the CDTT algorithm [8], previously presented by some of the authors as a valid distributed solution to the synchronous target tracking problem in heterogeneous sensor networks. CDTT is a discrete time iterative strategy, where each iteration consists of two phases: an estimation phase - in which each agent individually measures or predicts the target position at the current time step; followed by a consensus phase - where all the networked agents agree on the best estimated target position available among the network nodes.

While in [8] it is assumed that all the operations are executed in a synchronous fashion, in this work we assume that agents’ clocks are not synchronized. We only make a very loose, yet realistic assumption: that is, that the time interval between two consecutive updates of any agent’s state is not arbitrarily long (this is technically known as partial asynchronism assumption [12]). Under this assumption, the main challenge is to make all agents still able to reach a common outcome on the best estimate of the target state, starting from local estimates produced at different time instants, yet without using a time synchronization protocol. To this aim, an event-based framework is adopted.

The paper is structured as follows. The target tracking problem with asynchronous measurements is described in Section II. In Section III, an illustrative example of the ineffectiveness of CDTT in asynchronous frameworks is given. Section IV describes the aCDTT algorithmic solution. Numerical results in Section V show that aCDTT exhibits a tracking error performance which is comparable to that of the synchronous version of the algorithm. Moreover, the synchronous scenario can be obtained as a special case of the general asynchronous setting. Finally, we show that the adoption of the event-based framework is advantageous even for the synchronous settings, as it yields a reduction of the communication overhead.

978-1-4673-5716-6/13/$31.00 ©2013 IEEE 2006
II. Problem Formulation

We consider a multi-agent system composed by a network of $n$ agents, endowed with sensing, computation and communication capability. The network topology is determined by the communication range $r_c \in \mathbb{R}$, that we assume equal for every agent. We also assume that the agent’s positions do not change in time, so that the communication structure of the multi-agent system is modeled by a static undirected graph $G = (\mathcal{I}, \mathcal{E})$, where $\mathcal{I} \triangleq \{1, \ldots, n\}$ is a finite nonempty set of nodes representing agents, and $\mathcal{E} \subseteq \mathcal{I} \times \mathcal{I}$ is a set of node pairs, called edges (or links). The pair $(i, j) \in \mathcal{E}$ if and only if node $i$ can communicate with node $j$, i.e., $\|p_i^{[i]} - p_j^{[j]}\| \leq r_c$, where $p_i^{[i]}$ and $p_j^{[j]}$ respectively denote the positions of agents $i$ and $j$ in a planar environment $E$, with $E$ a compact, convex subset of $\mathbb{R}^2$. Note that the graph $G$ is undirected, due to the constant value of the communication range $r_c$. We adopt a one-hop communication scheme among agents, that is, each agent can send and receive messages only to/from its direct neighbors. We indicate with $\mathcal{N}^{[i]} = \{j \in \mathcal{I} | (j, i) \in \mathcal{E}\}$ the set of the neighboring agents of node $i$. Moreover, we assume that the graph $G$ is connected, i.e., there exists a multi-hop undirected path between any two nodes in $\mathcal{I}$.

The target is modeled as a point moving in the environment $E$. The target position and velocity over time are $\xi(t) \in E$ and $v(t) \in \mathbb{R}^2$, respectively, and the target state vector is defined as

$$x(t) = [(\xi(t))^T, v(t)^T]^T.$$  \hspace{1cm} (1)

The task of the multi-agent system is to estimate the target state over time, and to reach an agreement over it, in a totally distributed and asynchronous fashion, in order to globally track the target motion. Each node of the network can measure the position of the target only if $\|p_i^{[i]} - \xi(t)\| \leq r_s^{[i]}$, where $r_s^{[i]}$ is the range of the $i$-th agent sensing device. We define with $T^{[i]} = \{t_0^{[i]}, t_1^{[i]}, \ldots\}$, $i \in \mathcal{I}$, the set of local discrete event times for agent $i$. Event times correspond to agent $i$’s operating instants (state estimations or state updates). The sets $T^{[i]}$ need not to be known by any other agent, except the $i$-th. Any particular realization of these sets will be called a scenario [12].

In order to let agents agree on the best estimate of the target state produced over the network, we adopt an asynchronous max-consensus protocol, which is defined below.

**Definition 2.1: (Asynchronous max-consensus protocol)** Let $\eta^{[i]}(t_0^{[i]})$ be the initial value of a variable possessed by each node $i \in \mathcal{I}$. Through a max-consensus protocol, the network nodes must agree on the maximum value for $\eta^{[i]}$. The asynchronous max-consensus update rule for agent $i$ at time $t_{m+1}^{[i]} \in T^{[i]}$ is given by:

$$\eta^{[i]}(t_{m+1}^{[i]}) = \max_{j \in \mathcal{N}^{[i]}} \{\eta^{[i]}(t_{m}^{[i]}), \eta^{[j]}(t_{m}^{[j]})\}, \quad m \in \mathbb{N},$$  \hspace{1cm} (2)

where the quantity $t_{m}^{[j]} \leq t_{m+1}^{[j]}$ corresponds to the time instant when information $\eta^{[j]}$ from a neighboring agent $j$ becomes available to agent $i$. This means that agent $i$, at each discrete event time $t_{m+1}^{[i]}$, $m \geq 0$, updates its variable to the maximum of the values of the neighboring agent’s variables, which are made available to it in the interval between two consecutive event times, $[t_{m}^{[i]}, t_{m+1}^{[i]}]$. Equation (2) is able to describe any kind of asynchronous update, including the case of agents which do not own a constant clock period. The only assumption to be made is that agents’ updates cannot be arbitrarily slow. This is formalized in the following partial asynchronism assumption.

**Assumption 2.1: (Partial asynchronism [12])** There exists a positive constant $B \in \mathbb{N}$ (called asynchronism measure) such that: (i) each agent performs an update at least once during any time interval of length $B$; (ii) the information used by any agent in the update rule (2) is outdated by at most $B$ time units.

Under this assumption, it can be proved that the dynamics described in Eq. (2) converges to a fixed point in finite time for all agents $i \in \mathcal{I}$, and that the point is the maximum among the initial values of the consensus variable of all agents, i.e., $\max_{i \in \mathcal{I}} \eta^{[i]}(t_0^{[i]})$. Thus, Eq. (2) implements an effective asynchronous max-consensus protocol. The proof of convergence, based on the equivalence between the asynchronous dynamics on a fixed topology and a synchronous dynamics on a switching topology, is given in [13].

III. An Illustrative Example

In this section, we provide an example of a network of agents which performs the synchronous CDTT [8] in an asynchronous setting. The CDTT is a distributed iterative algorithm where each iteration consists of two phases: estimation and consensus. In the estimation phase, each agent measures or predicts the target position. Subsequently, in the consensus phase each agent communicates with its neighbors to agree, via a max-consensus protocol, on the best estimate of the target position, computed and stored in some node of the network during the estimation phase. The optimality criterion is computed over a quantity called perception confidence value, $\gamma^{[i]} \in \mathbb{R}$, $i \in \mathcal{I}$, which quantifies the accuracy of the individual agents measurements. In an asynchronous setting, the estimation phase and each run of the consensus phase are executed asynchronously by each agent $i$ at times marked by the local event times $T^{[i]}$. This implies that each agent firstly produces a value for the estimation phase, then takes part in the consensus phase, without waiting for all the other agents to end their estimation phase. In the following, we show that the application of the synchronous CDTT in an asynchronous setting may lead to incorrect agreement. Consider the three agents network illustrated in Fig. 1. It is well known that the synchronous max-consensus protocol converges in a number of iterations less or equal to the network diameter [14]. The diameter of the considered network is $D = 2$, so that, in a synchronous setting, two iterations of the max-consensus protocol would be in principle enough for the nodes to converge to the maximum of the initial values of the network variables $\gamma^{[i]}$. The perception confidence values locally computed are also indicated. We assume that at the event times $t_0^{[i]}$,
Fig. 1. An illustrative example: communication topology of a network of three agents, with diameter $D = 2$, and perception confidence values taken at given local time instants.

TABLE I

<table>
<thead>
<tr>
<th>ID</th>
<th>Consensus Variable $\gamma_i$</th>
<th>Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\gamma_1(t_1^{[1]}) = 4$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$\gamma_1(t_1^{[2]}) = \max(\gamma_1(t_0^{[1]})) + 4$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\gamma_1(t_1^{[3]}) = \max(\gamma_1(t_0^{[2]})) + 4$</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>$\gamma_2(t_2^{[1]}) = 7$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$\gamma_2(t_2^{[2]}) = \max(\gamma_2(t_1^{[1]})) + 7$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\gamma_2(t_2^{[3]}) = \max(\gamma_2(t_1^{[2]})) + 7$</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>$\gamma_3(t_3^{[1]}) = 7$</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$\gamma_3(t_3^{[2]}) = \max(\gamma_3(t_2^{[1]})) + 7$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\gamma_3(t_3^{[3]}) = \max(\gamma_3(t_2^{[2]})) + 7$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$\gamma_3(t_3^{[4]}) = 5$</td>
<td>1</td>
</tr>
</tbody>
</table>

Each agent $i \in I$ performs its first estimate, computing for example the values: $\gamma_1(t_0^{[1]}) = 4$, $\gamma_2(t_0^{[2]}) = 1$, and $\gamma_3(t_0^{[3]}) = 6$. Following the synchronous max-consensus protocol, each agent assumes that two iterations are needed at most to converge to the maximum, so that at the third event time it can proceed with a new estimation phase. That is, new estimates will be produced at event times $t_3^{[4]}$. Let us assume that the subsequent perception confidence values are $\gamma_1(t_1^{[3]}) = 7$, $\gamma_2(t_2^{[3]}) = 4$, and $\gamma_3(t_3^{[3]}) = 5$. Note that the maximum values of $\gamma_i$ across the network are, after the first measurement, $\gamma_3(t_0^{[3]}) = 6$, and, after the second measurement, $\gamma_1(t_1^{[3]}) = 7$. As can be noted from Fig. 2, where the time lines of each agent are illustrated, the clocks are not synchronized. Thus, CDTT iterations, along with their estimation and consensus phases, will be misaligned.

The evolution of the first iteration of the CDTT algorithm is reported in Table I, and described in the following with reference to Fig. 2. Each CDTT iteration, reported for each agent (ID), takes three event times: one for the estimation phase (phase 1), and two for the consensus phase (phase 2), since the diameter of the communication network is $D = 2$. At time $t_i^q$, $i \in \{1, 2, 3\}$, all the agents perform an estimate. At the first consensus iteration for agent 1, time $t_1^1$, no other agent has entered the consensus phase, (see Fig. 2) so the updated value of $\gamma_1(t_1^{[1]})$ is based only on its own previous value. At the end of its consensus phase, that is at time $t_2^1$, agent 1 converges to the wrong value $\gamma_1(t_1^{[1]}) = 4$, which is obtained without taking into account the values $\gamma_2(t_1^{[2]})$ and $\gamma_3(t_0^{[3]})$, since the first consensus iteration for agent 2 has not yet occurred. Note that at time $t_1^1$ the first consensus iteration for agent 3 has already occurred, that is, at time $t_3^3$, but the value $\gamma_3(t_0^{[3]})$ is not available to agent 1, due to the absence of a direct communication link between agent 1 and agent 3. This value will be made available through agent 2, as soon as it executes its first consensus iteration. Differently, agent 3 ends its consensus phase at time $t_2^3$, knowing both values $\gamma_1(t_0^{[1]})$ and $\gamma_2(t_1^{[2]})$. Therefore, it converges to the correct value $\gamma_3(t_1^{[3]}) = 6$. Finally, agent 2 terminates the consensus phase at time $t_2^2$, with the value $\gamma_2(t_2^{[3]}) = 7$. This wrong result is obtained because agent 2, at the second iteration of the consensus phase, considers the value $\gamma_1(t_1^{[3]})$, received by agent 1 as the output of its first consensus iteration of the second CDTT iteration (that is, $\gamma_1(t_1^{[3]}) = \max(\gamma_2(t_2^{[2]}), \gamma_3(t_1^{[3]}))$).

In conclusion, due to the network topology and the time lines misalignment, agent 3 is the only one that converges to the correct maximum value of $\gamma$ at the first iteration of CDTT. Therefore, it is evident that a mechanism preventing missing and mixing of information between CDTT iterations is required when asynchronous settings must be taken into account.

IV. THE aCDTT ALGORITHM

A. Asynchronous Iterations

The aCDTT algorithm reflects the iterative structure of CDTT. To deal with its asynchronous formulation, we define the concept of asynchronous iteration. To this aim, it is now necessary to describe the time behavior of the system from an external point of view. We remark that this is only a description method, whereas in reality each agent does not have knowledge of the update times of the rest of the network nodes, as explained in Section II. With this in mind, the network behavior can be described as the interlacing of two time windows: the estimation time-window, and the consensus time-window.

Definition 4.1 (Estimation time-window): The $q$-th estimation time-window $W_e(q)$, $q \in \mathbb{N}$, is the time elapsed between the first and the last occurrence of the $q$-th estimate in the network, whichever nodes performed them. Obviously, the temporal length of $W_e(q)$ is variable over $q$.

Definition 4.2 (Consensus time-window): The $q$-th consensus time-window, $W_c(q)$, is the global time interval between the event in which any agent firstly starts the
consensus phase on the estimates produced in $W_e(q)$, and the event in which the last agent reaches consensus on the same values produced in $W_e(q)$.

**Definition 4.3 (aCDTT global iteration):** An aCDTT global iteration, indexed with $q \in \mathbb{N}$, consists of the union of the $q$-th estimation time-window, $W_e(q)$, and the $q$-th consensus time-window, $W_c(q)$, possibly overlapping in time.

**Definition 4.4 (aCDTT local iteration):** An aCDTT local iteration performed by agent $i \in \mathcal{I}$ consists of the sequence of the local execution of the estimation phase, and the subsequent execution of the consensus phase.

The $q$-th aCDTT global iteration starts when $W_e(q)$ starts, and ends as soon as $W_e(q)$ ends. This means that every aCDTT local iteration performed by each agent is always contained in an aCDTT global iteration. In general, an overlap between the two time-windows $W_e(q)$ and $W_c(q)$ of the same $q$-th aCDTT global iteration may occur, since each agent enters its $q$-th consensus phase as soon as it finishes its own estimation phase (that is, $W_e(q)$ might start before $W_e(q)$ ends). Moreover, since the event-based approach, adopted to report the local achievement of the consensus, triggers an agent to start the estimation phase for the subsequent aCDTT global iteration (see Sec.IV-B), also two successive aCDTT global iterations can overlap in time. To prevent the mix of information between consecutive tracking iterations (as it is shown for agent 2 in the illustrative example of Section III), it is necessary to distinguish information-packets belonging to different subsequent, but interleaving, aCDTT global iterations. To cope with this issue, we adopt a mechanism, better illustrated in Sec. IV-C, which relies on local labeling of every computation variable belonging to the same aCDTT global iteration.

**B. Algorithm Description**

Let us now describe in detail the two phases of an aCDTT local iteration. We assume that each agent $i \in \mathcal{I}$ has all the necessary information available up to the iteration $q-1$. The generic $q$-th local aCDTT iteration starts with the initialization of a binary vector of tokens, $\text{tok}^{[i]} \in \{0,1\}^n$, with all zeros. The token vector is a collection of flags through which agent $i$ keeps track of the agents by which the estimated target state has been already received (its working principle is inspired to [15] and will be further explained in the remainder of the section). It is updated at each local event time $t^{[i]}_k \in T^{[i]}$, and allows every agent to be aware of the progress of the consensus process at each aCDTT iteration.

1) **Phase 1 - Estimation Phase:** Each agent $i \in \mathcal{I}$ estimates the target position exactly once in $W_e(q)$, $q \in \mathbb{N}$. In particular, at the first event time $t^{[i]}_0 \in T^{[i]}$ occurring for agent $i$ in the new estimation window $W_e(q)$, the relation $\text{tok}^{[i]}(i) = 0$ is certainly verified, and the estimation phase is carried out for the $q$-th iteration producing the target position estimate, i.e., $\hat{x}^{[i]}(q) = \hat{x}^{[i]}(t^{[i]}_0)$. Then, the element $\text{tok}^{[i]}(i)$ is set to 1, while the other elements $\text{tok}^{[i]}(j), j \in \mathcal{I}, j \neq i$, are still set to 0. The way in which the estimate $\hat{x}^{[i]}(t^{[i]}_0)$ is produced depends on the ability of the $i$-th agent of directly measure the target position, discriminating the role of agent $i$ at time $t^{[i]}_0$ (that is, a sensing or a predicting agent [8]). The $i$-th agent is called a sensing agent, if it can directly measure the target position at time $t^{[i]}_0$, i.e., if the distance $d^{[i]}(t^{[i]}_0) = ||p^{[i]}(q) - \xi(t^{[i]}_0)|| \leq r^{[i]}$, where $\xi(t^{[i]}_0)$ represents the effective position of the target at time $t = t^{[i]}_0$. Thus, the measurement of the target position performed by the $i$-th sensing agent at time $t^{[i]}_0$ is stored in the $z^{[i]}(t^{[i]}_0)$ vector.

**Measurement** is obtained assuming a range-bearing model, in which the higher the relative distance between agent $i$ and the target, the higher the uncertainty on measurement. Moreover, each agent $i \in \mathcal{I}$ carries a mobility model for the target, given by a linear sampled-time dynamical process. Thus, to obtain an optimized estimate of the target state, dubbed with $\hat{x}^{[i]}(q)$, we employ a Kalman filter [1] able to minimize the error variance of the state estimate. This is equivalent to minimizing the trace of the a posteriori estimate covariance matrix $P^{[i]}(q)$. Moreover, the Kalman filter estimates the speed of the target, since it cannot be measured directly. On the other hand, if the target is not within the sensing range of the $i$-th agent, this is called a predicting agent, and the target state is estimated by using only the state prediction equation of the Kalman filter, i.e. $\hat{x}^{[i]}(q) = \hat{x}^{[i]}(q-1)$. In this case, the a posteriori state estimate $\hat{x}^{[i]}(q-1)$ is substituted by $z(q-1)$, that is the output, equal for each agent, of the consensus phase described in the following.

In conclusion, at the end of the estimation phase for the $q$-th aCDTT local iteration, agent $i$ has computed an individual estimate of the target state $\hat{x}^{[i]}(q)$, and possesses the corresponding error covariance matrix $P^{[i]}(q)$, computed by the local Kalman filter. Consequently, it sets its perception confidence value $\gamma^{[i]}(q) = (\text{Tr}(P^{[i]}(q)))^{-1}$, where $\text{Tr}(\cdot)$ is the matrix trace operator. This is related to the concept of Fisher information [16], and it grows with the accuracy of the measurement performed by agent $i$ for the $q$-th iteration. Therefore, it quantifies the influence of the single agent estimate on the final outcome of the consensus phase. It is worth to note that this formulation of the perception confidence value takes also into account the case where an agent, switching from sensing to predicting status over two subsequent aCDTT iterations, might produce a more accurate estimate than an agent which keeps to be in the sensing status, yet owning a larger sensing range, because of the dependence of the uncertainty of the measurement on the target distance.

The estimation phase is then forbidden to agent $i$ in subsequent events, until the condition $\bigwedge_{j=1}^n \text{tok}^{[j]}(j) = 1, j \in \mathcal{I}$, is verified (where $\wedge$ denotes the logical and operation). On the other hand, between two consecutive estimation phases, agent $i$ is involved in the consensus phase.

2) **Phase 2 - Consensus Phase:** In the consensus phase, a max-consensus based algorithm on the perception confidence value is run in order to select, in a totally distributed and asynchronous way, the agent that produced the most accurate estimate of the target position in the window $W_e(q)$, and to propagate its corresponding estimate through the whole network. This Phase is schematized in Algorithm 1. Agent
Algorithm 1 Consensus phase for agent $i$ at iteration $q$

1: Input: $\gamma^{i}(q)$, $\hat{x}^{i}(q)$, $P^{i}(q)$, $\text{tok}^{i}$
2: Output: $\chi^{i}(q)$, $\Pi^{i}(q)$
3: $\psi^{i}(q) = \gamma^{i}(q)$
4: $\chi^{i}(q) = \hat{x}^{i}(q)$
5: $\Pi^{i}(q) = P^{i}(q)$
6: while $(\bigwedge_{j=1}^{n}\text{tok}^{j}(i)) = 0$ do
7: \hspace{1em} $\text{RECEIVE}(\text{tok}^{j}, \psi^{j}(q), \chi^{j}(q), \forall j \in \mathcal{N}^{i})$
8: \hspace{1em} $\psi^{i} := \bigvee_{j \in \mathcal{N}^{i}} \psi^{j}$
9: \hspace{1em} $\chi^{i}(q) = \max_{j \in \mathcal{N}^{i}} \{\chi^{j}(q)\}$
10: \hspace{1em} $\mu^{i}(q) = \text{thr}(\arg\max_{j \in \mathcal{N}^{i}} \{\chi^{j}(q)\})$
11: \hspace{1em} $\chi^{i}(q) = \chi^{[\mu^{i}]}(q)$
12: \hspace{1em} $\Pi^{i}(q) = \Pi^{[\mu^{i}]}(q)$
13: \hspace{1em} $\text{SEND}(\text{tok}^{i}, \chi^{i}(q), \mu^{i}(q), \chi^{i}(q), \Pi^{i}(q))$
14: end while
15: $\chi^{i}(q) = \chi^{[\mu^{i}]}(q)$
16: $\Pi^{i}(q)$

$i$ sets its internal variables $\psi^{i}(q) \in \mathbb{R}$, $\chi^{i}(q) \in \mathbb{R}^{4}$, and $\Pi^{i}(q) \in \mathbb{R}^{4}$, respectively, to the values $\gamma^{i}(q)$, $\hat{x}^{i}(q)$, $P^{i}(q)$, obtained from Phase 1 (lines 3-5). Then, an asynchronous max-consensus protocol on the variable $\chi^{i}(q)$ is run. Let us now detail the steps of a single asynchronous max-consensus iteration (lines 7-13). First, the value of the vector $\text{tok}^{i}(q)$ is updated (line 8) with the token information received by the neighbors in $\mathcal{N}^{i}$ (line 7). Then, starting from the values $\chi^{i}(q), j \in \mathcal{N}^{i}$, agent $i$ computes a new max-value $\chi^{i}(q)$ for iteration $q$, the corresponding vectors $\psi^{i}(q)$, and the matrix $\Pi^{i}(q)$ (lines 9-11). At step 9, a tie-break function $\text{thr}$ returns a single value when the maximum value for $\chi^{j}(q)$ is allocated in more than one neighboring agent. The tie-break rule could be, for example, to choose $\chi^{j}(q)$ with the lower index among those owning the maximum $\chi^{j}(q)$. At the end (line 13), agent $i$ propagates $\text{tok}^{i}(q)$, the value $\chi^{i}(q)$, the vector $\psi^{i}(q)$, and the covariance matrix $\Pi^{i}(q)$ to the neighboring agents $\mathcal{N}^{i}$. The associated agent index is also sent, due to the tie-break rule implemented. Every iteration of the consensus loop of Algorithm 1 runs in correspondence with the event times $t_{q}^{[i]} \in T^{[i]}$, until the convergence condition of the consensus algorithm for the $q$-th iteration is verified by agent $i$ ($\bigwedge_{j=1}^{n}\text{tok}^{j}(i) = 1$). Under this condition, agent $i$ can mark its consensus phase as terminated (see remark 4.1). An agreement both on the best estimate of the target state and on the related a posteriori covariance matrix is achieved asynchronously by every agent for each aCDTT iteration $q$. Each agent will store these values in its variables $\chi(q)$ and $\Pi(q)$, respectively (lines 15-16). These two variables are fed back to the next local estimation phase, in order to let the local Kalman filters predict the a priori state estimate starting from the best available estimate produced by the network in the previous aCDTT iteration and, therefore, improve the individual prediction performance.

Remark 4.1: The convergence of the max-consensus protocol implemented in Algorithm 1 is guaranteed, at each iteration $q$, by the token-based mechanism and the associative property of the max operator. In fact, an element $\text{tok}^{[i]}(s)$ set to 1 means that agent $i$ has received, through a neighboring agent $j \in \mathcal{N}^{i}$, the state information produced by agent $s \in \mathcal{I}$ for the ongoing iteration, i.e., a path from the source agent $s$ to agent $i$ has been established in the network. This is always possible, if we assume the graph $G$ connected. Moreover, Assumption 2.1 guarantees that each agent updates and spreads its information state in a time interval not indefinitely long, i.e., the condition $\bigwedge_{j=1}^{n}\text{tok}^{j}(i) = 1$ will be eventually verified over time. We prove in [13] that the upper bound for the convergence time of Algorithm 1 depends on the asynchronism measure $B$, and on the network diameter $D$. In particular, the condition $\bigwedge_{j=1}^{n}\text{tok}^{j}(i) = 1$ will be verified in at most $B \cdot D$ event times.

C. Labels management

In order to prevent the mixing of information between consecutive tracking iterations, we adopt a labeling mechanism, inspired to [15]. We assume that every computation variable in the same aCDTT iteration is labelled with the same element in the set $\mathcal{L} = \{A, B, C\}$. It can be easily verified that subsequent iterations for each agent $i \in \mathcal{I}$ will always exhibit labels in a sequence of the form $\{A, B, C, A, B, C, \ldots\}$ [15]. Let us consider the generic agent $i$, suppose that it is in the $q$-th iteration, and that this iteration has been labeled with $A$, i.e., $\text{q}^{(A)}$. When the condition $\bigwedge_{j=1}^{n}\text{tok}^{j}(A) = 1$, $j \in \mathcal{I}$, is satisfied, agent $i$ will pass to the following iteration $q+1$, and will label all the corresponding variables with $B$. That is, iteration $q+1$ is referred as $q+1(B)$. It can be verified that, for agent $i$ to transition from iteration $q(A)$ to $q+1(B)$, it is necessary that all other agents have also executed the estimation phase for $q(A)$, since otherwise agent $i$ will be missing tokens from the agents that are currently in iteration $q-1(C)$, and Algorithm 1 could not converge. Once agent $i$ transitions to iteration $q+1(B)$, it initializes all variables for this iteration with label $B$, while it keeps and propagates the variables of iteration $q(A)$ for agents that are still in $q(A)$. Furthermore, all the variables of the iteration $q-1(C)$ are deallocated, since no agent is involved in it any more.

V. Numerical Results

To evaluate the performance of the aCDTT algorithm, we define an accuracy index of the target tracking performance as $\alpha = \frac{1}{T_{q}} \sum_{t_{q}}^{T_{q}-1} \|\xi(t_{q}) - \xi(t_{q})\|^{2}$, where $q_{j}$ is the number of completed global aCDTT iterations and $\xi(t_{q})$ is the real position of the target at the discrete time $t_{q}$ at which the best target estimation $\hat{\xi}(q)$ is performed by an agent in the $q$-th iteration. We consider a maneuvering target moving on a square field $E$. We run Monte Carlo simulations on 100 random target trajectories, considering networks with different numbers of agents $n$. The network nodes are made heterogeneous, i.e., agents are equipped with range-bearing sensors with different sensing radii. Moreover, we consider different coverage levels $\rho$, defined as the ratio between the area covered by the sensing areas of the agents, and the total area $L^{2}$. For more details about the definition of the performance parameters the reader is referred to [8]. Figure 3 shows the trend of the performance parameter $\alpha$ versus the
coverage ratio $\rho$, for different values of the number of sensors $n$ and different values of the asynchronism measure $B$. We set different clock periods $c^{[i]} = b \cdot c$ for each agent, where $b$ is an integer number picked at random in the set $\{1, \ldots, B\}$ and $c = 1 \cdot 10^{-4}$. Clock periods are then kept fixed in time. The initial guess of the target state (corresponding to iteration $q = 0$) is set to $x^{[i]}(0) = 0$, $\forall i \in I$, that is, the initial estimate of the target position coincides with the center of the field $E$, and the initial target velocity is considered null.

Similar to the synchronous framework [8], simulation results in the asynchronous scenario show that the tracking performance is improved (i.e., $\alpha$ decreases) as long as the coverage ratio $\rho$ and the number of nodes $n$ increase, confirming the validity of our approach. Moreover, the asynchronous framework can be obtained as a particular case of the asynchronous one ($B = 1$). Due to the ability of the token mechanism in determining asynchronously the convergence of the max-consensus protocol, an improvement in the communication overhead with respect to the synchronous implementation of the CDTT algorithm (see Tab. II), is achieved. In fact, the max-consensus phase of the aCDTT algorithm with $B = 1$ converges after $D$ time units, even if the diameter $D$ of the network is unknown. On the contrary, in the absence of such information, the max-consensus phase of CDTT has to be run for $n - 1$ iterations, which represents the worst case diameter of a network with $n$ agents [8]. Moreover, the labeling mechanism enables the presence of two interleaving iterations even in a synchronous setting, allowing the user to increase the sampling frequency of the tracking algorithm.

VI. CONCLUSIONS

In this paper, we present a distributed target tracking framework for networks of agents with asynchronous and possibly time-variant clocks. The proposed algorithm makes use of the properties of the asynchronous max-consensus protocol to let all the agents agree, in finite time, on the most accurate estimate across the network. Furthermore, introducing the concept of asynchronous iteration, we propose an event-based approach which makes for the lack of a common time scale at the network level. Numerical results confirm the validity of our approach, showing that the aCDTT algorithm maintains the same qualitatively trend of the tracking performance of its synchronous counterpart. Moreover, we prove that the synchronous application of the algorithm, derived from the asynchronous version by setting the asynchronism measure equal to one, yields a lower communication load, if compared with the previous synchronous implementation of the algorithm.

**REFERENCES**