2016 American Control Conference (ACC)
Boston Marriott Copley Place
July 6-8, 2016. Boston, MA, USA

Multi-agent coordination with asynchronous cloud access

Cameron Nowzari George J. Pappas

Abstract— **In this work we study a multi-agent coordination problem in which agents are only able to communicate intermittently with the cloud. To minimize the amount of required communication, we are interested in developing a self-triggered algorithm that determines when communication with the cloud is necessary. Unlike the overwhelming majority of similar works that propose distributed event- and/or self-triggered control laws, this work doesn't assume agents can be 'listening' continuously. In other words, when an event is triggered by one agent, neighboring agents will not be aware of this until the next time they establish communication with the cloud. Each time an agent communicates with the cloud, it must determine the next time to reestablish communication while guaranteeing the completion of some global task. We show that our self-triggered coordination algorithm guarantees that the system asymptotically reaches the set of desired states. Simulations illustrate our results.**

I. Introduction

This paper considers a multi-agent coordination problem where agents can only communicate with one another indirectly through the use of a central base station or 'cloud'. More specifically, we consider the problem of coordinating a number of submarines that can only communicate with a base station when at the surface of the water. While submerged, communication with the outside world is impossible. Thus, each time a submarine surfaces, we are interested in an algorithm that determines the next time it should surface in order to adequately achieve some desired global task. While we motivate our problem via an underwater coordination problem in which communication while submerged is impossible, it is also directly applicable to scenarios where wireless-capable agents cannot be 'listening' to any communication channels continuously.

Literature review: In this work we are interested in employing a self-triggered strategy that determines when it is necessary for an agent to communicate with the cloud in order to guarantee the completion of some global task. Our work is motivated by the problem of coordinating submarines, or Autonomous Underwater Vehicles (AUVs), that are only able to communicate with the cloud when they are at the surface of the water. This is because communicating with the outside world when underwater is extremely expensive, if not impossible [1], [2]. Thus, we are interested in an algorithm that agents can use to determine when communication with the cloud should be established in order to achieve some desired formation.

Cameron Nowzari and George J. Pappas are with the Electrical and Systems Engineering Department, University of Pennsylvania, Philadelphia, {cnowzari,pappasg}@seas.upenn.edu.

A useful tool then is event-triggered control, where an algorithm is designed to tune controller executions to the state evolution of a given system, see e.g., [3], [4]. These ideas have also been applied to multi-agent systems where the goal is to trade in computation and decision making to reduce the overall communication, sensing, and/or actuator efforts of the agents. In [5], the authors find criteria for agents that determine when their control signals should be updated. In [6], the authors consider networked systems with disturbances, communication delays, and packet drops. In addition to deciding when control signals should be updated, several works have also explored the application of event-triggered ideas to the acquisition of information, be it through either communication or sensing. To this end, [7], [8], [9] combine event-triggered controller updates with sampled data that allows for the periodic evaluation of the triggers. Other works like [10] even drop the need for periodic access to information by considering event-based broadcasts, where agents decide with local information only when to share information with neighbors. Self-triggered control [11], [12], [13] relaxes the need for continuously monitoring some 'trigger' by deciding when a future sample should be taken based on the available information from the last sample.

Regarding multi-agent coordination, there is a considerable amount of available literature, see e.g., [14], [15], [16]. In particular, we formulate our problem as a multi-agent consensus problem for which there has been a lot of work done already. A continuous-time algorithm that achieves asymptotic convergence to average consensus for both undirected and weight-balanced directed graphs is introduced in [17]. The work [18] builds on this algorithm to propose a Lyapunov-based event-triggered strategy that dictates when agents should update their control signals. However, this strategy requires each agent to have perfect information about their neighbors at all times. The work [19] uses event-triggered broadcasting with time-dependent triggering functions to provide an algorithm where each agent only requires exact information about itself, rather than its neighbors. In [20], the authors propose an event-triggered broadcasting law with state-dependent triggering functions where agents again do not rely on the availability of continuous information about their neighbors. More recently, these works have been extended to arbitrary directed graphs, rather than only undirected ones [9], [21], [22]. One large drawback of all the above works is that all agents are required to be 'listening' at all times. More specifically, when an event is triggered by some agent, neighboring agents are immediately made aware of this. For example, in the case of event-triggered broadcasting, it is assumed that when a message is broadcast it is immediately (or with some delay) received by neighboring agents. Instead, we are interested in a setup where an agent has zero access to the outside world when disconnected from the cloud (i.e., when underwater).

In [23], the authors consider an extremely similar problem as ours but develop an event-triggered solution in which all AUVs must surface at the same time. Our work is very closely related to [24], where the authors consider a very similar problem, motivation, and solution to the ones we propose. However, since they consider noise and propose a time-dependent triggering rule, they are only concerned with reaching practical consensus (i.e., a ball around the desired consensus state). Instead, we are interested in developing a state-dependent triggering rule that better aligns the events with the global objective and guarantees the agents asymptotically reach an exact agreement state.

Statement of contributions: In this work we utilize a cloud communication model for a multi-agent system motivated by a multi-agent AUV coordination problem. In particular, agents are only able to communicate with the cloud when at the surface of the water, and must autonomously decide when to resurface. Our main contribution is the development of a novel distributed self-triggered algorithm that agents can use each time they surface to determine the next time they need to surface. In general, event- and self-triggered algorithms are designed such that agents locally trigger events such that they can each guarantee to be contributing positively to some global task at all times. More specifically, they are often designed with the use of some Lyapunov function V for which each agent can guarantee they are monotonically optimizing V. However, because we assume agents are unaware of what is happening while they are underwater, our algorithm does not rely on this guarantee. More specifically, we actually allow an agent to be contributing negatively to the global task as long as it is accounted for by its net contribution. After developing the algorithm, we are able to guarantee its convergence to the desired set of states. Finally, we illustrate our results through simulations.

Preliminaries: We denote by $\mathbb{R}$ the set of real numbers. The Euclidean norm on $\mathbb{R}^N$ is denoted by $\|\cdot\|$. A graph $\mathcal{G} = (V, E)$ is comprised of a set of vertices $V = \{1, \dots, N\}$ and edges $E \subset V \times V$. The graph $\mathcal{G}$ is undirected if for any edge $(i, j) \in E$, the edge $(j, i) \in E$ also. An edge $(i, j) \in E$ means that vertex j is a neighbor of i. The set of neighbors of a given node i is given by $\mathcal{N}_i$. The adjacency matrix $A \in \mathbb{R}^{N \times N}$ is defined by $a_{ij} = 1$ if $(i, j) \in E$ and $a_{ij} = 0$ otherwise. A path from vertex i to j is an ordered sequence of vertices such that each intermediate pair of vertices is an edge. An undirected graph $\mathcal{G}$ is connected if there exists a path from all $i \in V$ to all $j \in V$. The degree matrix D is a diagonal matrix where $d_{ii} = |\mathcal{N}_i|$. The Laplacian matrix is defined as $L = D - A$. For undirected graphs the Laplacian is symmetric $L = L^T$ and positive semidefinite. If the graph $\mathcal{G}$ is connected, the Laplacian has exactly one eigenvalue at 0 (with associated eigenvector $\mathbf{1}_N$) with the rest strictly positive, $0 = \lambda_1(L) < \lambda_2(L) \leq \dots \leq \lambda_N(L)$.

II. Problem statement

Consider the N agent coordination problem with single-integrator dynamics

$$\dot{x}_i(t) = u_i(t), \tag{1}$$

for all $i \in \{1, \dots, N\}$, where we are interested in reaching a configuration such that $\|x_i(t) - x_j(t)\| \to 0$ as $t \to 0$ for all $i, j \in \{1, \dots, N\}$. For simplicity, we consider scalar states $x_i \in \mathbb{R}$ for all agents but note that all results are readily extendable to arbitrary dimensions.

Given a connected communication graph $\mathcal{G}$, it is well known [17] that the distributed continuous control law

$$u_i(t) = -\sum_{j \in \mathcal{N}_i} (x_i(t) - x_j(t)) \tag{2}$$

drives each agent of the system to asymptotically converge to the average of the agents' initial conditions. In compact form, this can be expressed by

$$\dot{x} = -Lx,$$

where $x = (x_1, \dots, x_N)^T$ is the column vector of all agent states and L is the Laplacian of $\mathcal{G}$. However, in order to be implemented, this control law requires each agent to continuously have information about its neighbors and continuously update its control law.

There have been many recent works aimed at relaxing one or both of these requirements [9], [25], [22], [19]. However, they all require events triggered by some agent to be immediately acknowledged by neighboring agents. In other words, when an event is triggered by one agent, its neighbors are immediately aware and can take action accordingly.

In all of these works, they utilize a piecewise constant implementation of the controller (2) given by

$$u_i(t) = -\sum_{j \in \mathcal{N}_i} (\widehat{x}_i(t) - \widehat{x}_j(t)), \tag{3}$$

where $\widehat{x}_j(t)$ is the last broadcast state of agent j. Note that although agent i has access to its own state $x_i(t)$, the controller (5) uses the last broadcast state $\widehat{x}_i(t)$. This is to ensure that the average of the agents' initial states is preserved throughout the evolution of the system. More specifically, utilizing this controller, one has

$$\frac{d}{dt}(\mathbf{1}_N^T x(t)) = \mathbf{1}_N^T \dot{x}(t) = \mathbf{1}_N^T L \widehat{x}(t) = 0, \tag{4}$$

where $\widehat{x} = (\widehat{x}_1, \dots, \widehat{x}_N)^T$ and we have used the fact that L is symmetric and $L\mathbf{1}_N = 0$. This means that when an agent i broadcasts its current state $x_i(t^*)$ at some time $t*$, all its neighbor $j \in \mathcal{N}_i$ immediately update $\widehat{x}_i = x_i(t^*)$ so that the average can be preserved.

Unfortunately, we do not have the luxury of being able to continuously detect events that occur. Instead, we assume that agents are only able to update their control signals when their own events are triggered (i.e., when they are surfaced). Let $\{t_i^\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$ be the sequence of times at which agent i surfaces. Then, we utilize a piecewise constant implementation of the controller (2) given by

$$u_i(t) = -\sum_{j \in \mathcal{N}_i} (x_i(t_i^\ell) - x_j(t_i^\ell)), \quad t \in [t_i^\ell, t_i^{\ell+1}). \tag{5}$$

The purpose of this paper is to develop a self-triggered algorithm that determines how the sequence of times $\{t_i^\ell\}$ can be chosen by the agents such that the system converges to the desired agreement statement. More specifically, each agent i at each surfacing time t_i^ℓ must determine the next surfacing time $t_i^{\ell+1}$ only using information available to them on the cloud. The closed loop system should then have trajectories such that $|x_i(t) - x_j(t)| \to 0$ as $t \to 0$ for all $i, j \in \{1, \ldots, N\}$. We describe the cloud communication model next.

Cloud communication model

We assume there exists a base station or 'cloud' that agents are able to upload and download data to when the communication link between them is open. The cloud can essentially store as much data as it needs to, but no computations can be done on the cloud. At any given time $t \in [t_i^\ell, t_i^{\ell+1})$, the cloud stores the following information about agent i: the last time $t_i^{\text{last}}(t) = t_i^\ell$ that agent i surfaced, the next time $t_i^{\text{next}}(t) = t_i^{\ell+1}$ that agent i is scheduled to surface, the state $x_i(t_i^{\text{last}})$ of agent i when it last surfaced, and the last control signal $u_i(t_i^{\text{last}})$ used by agent i. This information is summarized in Table I.

For simplicity, we assume that agents can download/upload information to/from the cloud instantaneously and that only one agent can be connected to the cloud at a time. Let t_i^ℓ be a time at which agent i surfaces to communicate with the cloud. The communication link with the cloud is established at time t_i^ℓ, and we immediately update $t_i^{\text{last}} = t_i^\ell$ and $x_i(t_i^\ell)$ based on agent i's current position. While the link is open, agent i has access to all the information in Table I for each neighbor $j \in \mathcal{N}_i$. Note that, using the information available in the cloud, agent i is able to perfectly reconstruct the states of its neighbors

$$x_j(t_i^\ell) = x_j(t_j^{\text{last}}) + u_j(t_j^{\text{last}})(t_i^\ell - t_j^{\text{last}}),$$

for all $j \in \mathcal{N}_i$. Using this information, agent i computes and uploads its control signal,

$$u_i(t_i^\ell) = -\sum_{j \in \mathcal{N}_i} (x_i(t_i^\ell) - x_j(t_i^\ell)),$$

to the cloud. Finally, before closing the communication link to the cloud, agent i must determine the next time $t_i^{\text{next}} = t_i^{\ell+1} > t_i^\ell$ at which it will resurface. The goal of this paper is then to find a way to determine t_i^{next} such that the system converges to an agreement state, i.e., $|x_i(t) - x_j(t)| \to 0$ as $t \to \infty$ for all $i, j \in \{1, \ldots, N\}$.

t_i^{last}	Last time agent i surfaced
t_i^{next}	Next time agent i will surface
$x_i(t_i^{\text{last}})$	Last updated position of agent i
$u_i(t_i^{\text{last}})$	Last trajectory of agent i

TABLE I

DATA STORED ON THE CLOUD FOR ALL AGENTS i AT ANY TIME t.

Remark II.1 (Multi-agent formation control) We note here that for simplicity, we formulate the multi-agent coordination problem as a consensus problem. The formal treatment can easily be modified to handle a formation control problem by letting $x_i(t) = p_i(t) - b_i$, where $p_i(t)$ is the actual position of agent i and b_i is the desired displacement from the average position of the fleet.

III. Distributed trigger design

Consider the function

$$V(x(t)) = \frac{1}{2} x^T(t) L x(t).$$

Note that $V(x) \geq 0$ and $V(x) = 0$ if and only if $x_j = x_i$ for all $i, j \in \{1, \ldots, N\}$. Thus, the function $V(x)$ encodes the objective of the problem and we are interested in driving $V(x(t)) \to 0$. For simplicity, when we drop the explicit dependence on time we are referring to time t. Then, if all agents use the control law (5), we have

$$\begin{aligned}
\dot{V} &= x^T L \dot{x} \\
&= -\sum_{i=1}^{N} \left(\sum_{j \in \mathcal{N}_i} x_j - x_i \right) \left(\sum_{j \in \mathcal{N}_i} x_j(t_i^{\text{last}}) - x_i(t_i^{\text{last}}) \right) \\
&= -\sum_{i=1}^{N} \left(\sum_{j \in \mathcal{N}_i} x_j - x_i \right)^2 + \\
&\left(\sum_{j \in \mathcal{N}_i} x_j - x_i \right) \left(\sum_{j \in \mathcal{N}_i} x_j - x_i - (x_j(t_i^{\text{last}}) - x_i(t_i^{\text{last}})) \right).
\end{aligned}$$

Let us split up $\dot{V} = \sum_{i=1}^{N} \dot{V}_i$, where

$$\dot{V}_i = -\left(\sum_{j \in \mathcal{N}_i} x_j - x_i \right) \left(\sum_{j \in \mathcal{N}_i} x_j(t_i^{\text{last}}) - x_i(t_i^{\text{last}}) \right). \tag{6}$$

Then,

$$\begin{aligned}
V(x(t)) &= V(x(0)) + \int_0^t \dot{V}(x(\tau)) d\tau \\
&= V(x(0)) + \sum_{i=1}^{N} \int_0^t \dot{V}_i(x(\tau)) d\tau.
\end{aligned}$$

Ideally, at this point we would be able to design a self-triggered algorithm such that $\dot{V}_i \leq 0$ for all agents i at all times. This means that at time t_i^ℓ, agent i must determine $t_i^{\ell+1}$ such that $\dot{V}_i(t) \leq 0$ for all $t \in [t_i^\ell, t_i^{\ell+1})$. To do this we need to be able to enforce

$$\Big|\sum_{j\in\mathcal{N}_i} x_j - x_i\Big| \geq \Big|\sum_{j\in\mathcal{N}_i} (x_j - x_j(t_i^{\text{last}})) - (x_i - x_i(t_i^{\text{last}}))\Big| \tag{7}$$

at all times $t \in [t_i^\ell, t_i^{\ell+1})$, which requires exact information about agent i's neighbors for all time $t \geq t_i^\ell$. Fortunately, agent i can compute a neighboring agent j's state exactly for $t \in [t_i^\ell, t_j^{\text{next}}]$ by computing

$$x_j(t) = x_j(t_j^{\text{last}}) + u_j(t_j^{\text{last}})(t - t_j^{\text{last}}) \tag{8}$$

using information available on the cloud. Unfortunately, this is only valid until time t_j^{next} when agent j resurfaces and changes its trajectory. Let $T_i = \min_{j\in\mathcal{N}_i} t_j^{\text{next}}$ be the next time at which a neighboring agent of i is scheduled to surface. This means agent i can exactly compute the trajectories of its neighbors using (8) until time T_i. Next, we show how this is used to compute the next triggering time t_i^{next}.

Self-triggered communication

At time t_i^ℓ, agent i uses the information on the cloud about its neighbors to compute the trajectories of its neighbors assuming they never change their control signals using (8). For now, we assume that

$$\Big|\sum_{j\in\mathcal{N}_i} x_j(t_i^\ell) - x_i(t_i^\ell)\Big| \neq 0. \tag{9}$$

Thus, at time t_i^ℓ, the LHS of (7) is strictly positive while the RHS is exactly 0. Agent i then computes t^* as the smallest time $t^* \geq t_i^\ell$ such that (7) is not satisfied. It is easy to show that $t^* > t_i^\ell$ if (9) is satisfied, but we note that this is not enough to rule out the possibility of Zeno behaviors; we will comment on this later. If $t^* \leq T_i$, then agent i simply sets $t_i^{\text{next}} = t_i^{\ell+1} = t^*$ and by continuity of $\dot{V}_i$, it is guaranteed that $\dot{V}_i(t) \leq 0$ for all $t \in [t_i^\ell, t_i^{\ell+1})$. On the other hand, if $t^* > T_i$ it can no longer be guaranteed that $\dot{V}_i(t) \leq 0$ for $t > T_i$. However, we do know that $\dot{V}_i(t) \leq 0$ for $t \in [t_i^\ell, T_i]$. Using this, we define

$$B_i = \int_{t_i^\ell}^{T_i} \dot{V}_i(\tau)d\tau \tag{10}$$

as the 'benefit' or 'contribution' that agent i has made to the global objective between t_i^{last} and T_i, from which point on it is no longer guaranteed that it is making a positive contribution.

However, in order to reduce the frequency of resurfacing, and more importantly to avoid multiple agents surfacing together, we allow agent i to stay underwater so long as the net contribution is still positive. Thus, we define the next surfacing time as

$$t_i^{\ell+1} = T_i + \sigma h(B_i),$$

where $h(\cdot)$ is to be determined and $\sigma \in [0, 1)$. Note that $h(\cdot)$ will depend on more than B_i, but we dot not show the explicit dependence on other parameters for simplicity. That is, the bigger the benefit that agent i can guarantee between time $t \in [t_i^\ell, T_i]$, the longer agent i can stay submerged after time T_i. Ideally, $h(B_i)$ is implicitly defined by

$$-\int_{T_i}^{T_i+h(B_i)} \dot{V}_i(\tau)d\tau = B_i,$$

i.e., the time at which no net contribution can no longer be made. Unfortunately, computing this time requires information that is not available to agent i while underwater. Instead, we are interested in developing lower-bounds on $h(B_i)$ that can be computed with information available to agents while underwater. For reasons of space, we refer this discussion to future work and consider only $\sigma = 0$ in the remainder of this paper.

In the case that (9) is not satisfied, that is,

$$\Big|\sum_{j\in\mathcal{N}_i} x_j(t_i^\ell) - x_i(t_i^\ell)\Big| = 0,$$

for some $\ell \in \mathbb{Z}_{\geq 0}$ and $i \in \{1, \dots, N\}$, it is possible that $t^* = \infty$. In this case we automatically have $t^* > T_i$, and furthermore we have $\dot{V}_i(t_i^\ell) = 0$. This essentially means agent i has reached a local minimum and needs to wait for a neighbor to resurface before it can move again. Theoretically, it would ideally resurface at exactly the same time as its first neighbor to do so, but we have assumed this to not be possible. For simplicity, in this case we set $t_i^{\ell+1} = T_i + T^{\text{dwell}}$ where $T^{\text{dwell}} > 0$ is an a priori chosen dwell time that agent i must wait before surfacing.

We note that this is essentially done, for the sake of simplicity, so there is no question of 'who gets to upload/download first?' when multiple agents surface simultaneously. One could alternatively imagine an ordering among the agents such that if multiple agents surface simultaneously, the order of communications is determined, but we do not enter into the details of this here. Our subsequent analysis then simply assumes that T^{dwell} is a negligible time constructed to simplify the analysis.

The formal `self-triggered coordination algorithm` is presented in Algorithm I and the main convergence result is stated next.

Theorem III.1 *Given the dynamics* (1) *with control law* (5) *and $\mathcal{G}$ connected, if the sequence of update times $\{t_i^\ell\}$ is determined by Algorithm 1 for all $i \in \{1, \dots, N\}$, then*

$$|x_i(t) - x_j(t)| \to 0$$

for all $i, j \in \{1, \dots, N\}$ as $t \to \infty$.

Algorithm 1 : `self-triggered coordination algorithm`

```
At surfacing time $t_i^\ell$, agent $i \in \{1,\dots,N\}$ performs:
 1: download $t_j^{\text{last}}, t_j^{\text{next}}, x_j(t_j^{\text{last}}), u_i(t_j^{\text{last}})$ for all $j \in \mathcal{N}_i$ from cloud
 2: compute neighbor positions $x_j(t_i^\ell) = x_j(t_j^{\text{last}}) + u_j(t_j^{\text{last}})(t_i^\ell - t_j^{\text{last}})$
 3: compute control $u_i(t_i^\ell) = -\sum_{j\in\mathcal{N}_i}(x_i(t_i^\ell) - x_j(t_i^\ell))$
 4: compute $T_i = \min_{j\in\mathcal{N}_i} t_j^{\text{next}}$
 5: compute $t^*$ as first time when (7) is no longer satisfied using (8)
 6: if $t^* < T_i$ then
 7:     set $t_i^{\text{next}} = t^*$
 8: else
 9:     if $u_i(t_i^\ell) = 0$ then
10:         set $t_i^{\text{next}} = T_i + T^{\text{dwell}}$
11:     else
12:         compute $B_i$ using (10)
13:         set $t_i^{\text{next}} = T_i + T^{\text{dwell}} + \sigma h(B_i)$
14:     end if
15: end if
16: upload $t_i^{\text{last}} = t_i^\ell$, $t_i^{\text{next}} = t_i^{\ell+1}$, $u_i(t_i^\ell)$, $x_i(t_i^\ell)$ to cloud
17: dive and set $u_i(t) = u_i(t_i^\ell)$ for $t \in [t_i^\ell, t_i^{\ell+1})$
```

Proof: Consider $V = \frac{1}{2}x^T Lx$, then

$$V(x(t)) = V(x(0)) + \int_0^t \dot{V}(\tau)d\tau$$
$$= V(x(0)) + \sum_{i=1}^N \int_0^t \dot{V}_i(\tau)d\tau.$$

Letting $\ell_i^{\max}(t) = \text{argmax}_{\ell\in\mathbb{Z}_{\geq 0}}\, t_i^\ell \leq t$ be the index ℓ^* such that $t_i^{\text{last}}(t) = t_i^{\ell^*}$, we can expand this as

$$V(x(t)) = V(x(0)) + \sum_{i=1}^N \sum_{\ell=0}^{\ell_i^{\max}} \int_{t_i^\ell}^{\min\{t_i^{\ell+1},t\}} \dot{V}_i(\tau)d\tau.$$

By definition of the self-triggering times $t_i^{\ell+1}$, we know that

$$\Delta V_i^\ell \triangleq \int_{t_i^\ell}^{t_i^{\ell+1}} \dot{V}_i(\tau)d\tau \leq 0$$

for all $\ell \in \{1,\dots,\ell_i^{\max}-1\}$. Thus, we have

$$V(x(t)) \leq V(x(0)) + \sum_{i=1}^N \sum_{\ell=0}^{\ell_i^{\max}-1} \Delta V_i^\ell.$$

It is then clear that $V(x(t))$ is a nonincreasing function along the trajectories of the closed-loop system and bounded from below (by 0). This means that $\lim_{t\to\infty} V(x(t)) = C \geq 0$ exists. Furthermore, since $\Delta V_i^\ell \leq 0$ for all ℓ, it is guaranteed that $\Delta V_i^\ell \to 0$ as $\ell \to \infty$ for all $i \in \{1,\dots,N\}$. Thus, by LaSalle's Invariance Principle [26], the trajectories of the system converge to the largest invariant set contained in

$$\{x \in \mathbb{R}^N | \dot{V}_i(x) = 0 \text{ for all } i \in \{1,\dots,N\}\}.$$

Recalling (6), it is easy to see that this is equivalent to the set

$$\{x \in \mathbb{R}^N | \sum_{j\in\mathcal{N}_i} x_j - x_i = 0 \text{ for all } i \in \{1,\dots,N\}\},$$

which means $(Lx)_i = 0$ for all i, which concludes the proof. ■

Remark III.2 (Zeno behavior) It is important to note here that we have not yet been able to rule out the possibility of Zeno behavior between triggers. More specifically, we have not been able to rule out the possibility of a single agent i requiring an infinite number of resurfaces before another agent can resurface. The proof of Theorem III.1 implictly assumes that this does not occur, i.e., $t_i^\ell \to \infty$ as $\ell \to \infty$ and $\ell \to \infty$ for all $i \in \{1,\dots,N\}$. •

IV. Simulations

Here we demonstrate the effectiveness of our proposed algorithm through a typical simulation with $N = 5$ agents and initial condition $x(0) = [-1, 0, 2.1, 2, 1]^T$. We run our simulation for 10 seconds using a dwell time $T^{\text{dwell}} = 0.0001$ seconds with an undirected topology given by $E = [(1,4),(1,5),(2,3),(3,4),(4,5)]$.

Figure 1 shows the trajectories of the closed-loop system. Figure 2 shows the evolution of $V(x(t)) = \frac{1}{2}x^T(t)Lx(t)$ along the trajectories of the system and Figure 3 shows the surfacing times of the five agents with a total of 422 events occurring in 10 seconds. Note that this seems to corroborate the assumption that Zeno behaviors are not an issue because the dwell time of 0.0001 seconds hardly comes into play. Note that we have not compared this against any other event-triggered consensus algorithms as they would not be fair. We are not aware of any other event-triggered consensus algorithms in which an event triggered by one agent does not immediately affect any other agent. The work [24] that inspired ours is closest, but their consideration of noise and only achieving practical consensus makes it difficult to compare with ours.

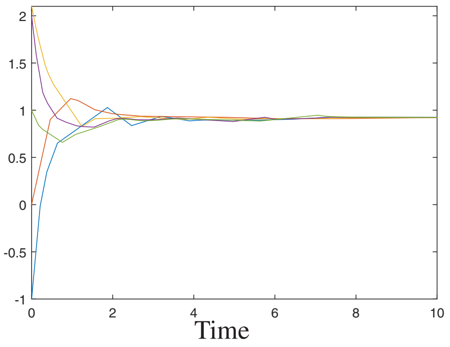

Fig. 1. Trajectories of the system.

V. Conclusions

We have considered a multi-agent coordination problem in which agents can only communicate with one another indirectly, via communication with a base station or 'cloud'. To minimize the amount of communications with the cloud, we have developed a self-triggered algorithm that autonomously determines the next time communication with the cloud should be established in order to achieve some desired

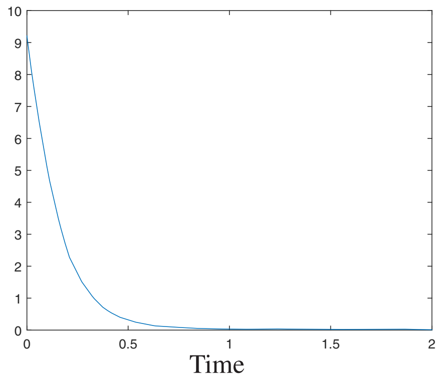

Fig. 2. Evolution of the Lyapunov function V.

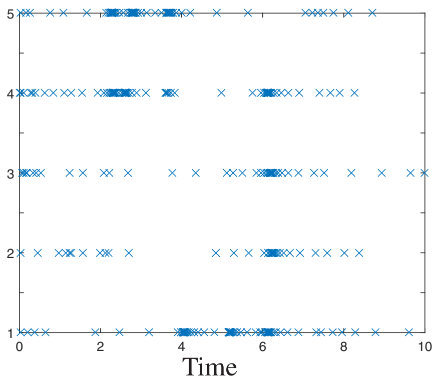

Fig. 3. Events triggered (surfacing times) by each agent along the evolution of the algorithm.

global task. Unlike the overwhelming majority of distributed event/self-triggered algorithms in the literature, no agent is required to be 'listening' at all times. For instance, many problems propose event-triggered broadcasting solutions; however, they implicitly assume that when an agent broadcasts a message, other agents are able to instantaneously (or with some delay) receive this information. Instead, when an agent is disconnected from the cloud, we assume it has absolutely no connection with the outside world and thus will not know about neighboring events being triggered until communication with the cloud is reestablished. We are very interested in developing a more pragmatic algorithm that guarantees that Zeno behavior cannot occur.

Acknowledgments

This work was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

[1] N. A. Cruz, B. M. Ferreira, O. Kebkal, A. C. Matos, C. Petrioli, R. Petroccia, and D. Spaccini, "Investigation of underwater nerworking enabling the cooperative operation of multiple heterogeneous vehicles," *Marine Technology Science Journal*, vol. 47, pp. 43–58, 2013.

[2] E. Fiorelli, N. E. Leonard, P. Bhatta, D. A. Paley, R. Bachmayer, and D. M. Fratantoni, "Multi-AUV control and adaptive sampling in Monterey Bay," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 935–948, 2006.

[3] K. J. Åström and B. M. Bernhardsson., "Comparison of Riemann and Lebesgue sampling for first order stochastic systems," in *IEEE Conf. on Decision and Control*, (Las Vegas, NV), pp. 2011–2016, Dec. 2002.

[4] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *IEEE Conf. on Decision and Control*, (Maui, HI), pp. 3270–3285, 2012.

[5] M. Mazo Jr. and P. Tabuada, "Decentralized event-triggered control over wireless sensor/actuator networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2456–2461, 2011.

[6] X. Wang and M. D. Lemmon, "Event-triggering in distributed networked control systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 586–601, 2011.

[7] G. Xie, H. Liu, L. Wang, and Y. Jia, "Consensus in networked multi-agent systems via sampled control: fixed topology case," in *American Control Conference*, (St. Louis, MO), pp. 3902–3907, 2009.

[8] W. P. M. H. Heemels and M. C. F. Donkers, "Model-based periodic event-triggered control for linear systems," *Automatica*, vol. 49, no. 3, pp. 698–711, 2013.

[9] X. Meng and T. Chen, "Event based agreement protocols for multi-agent networks," *Automatica*, vol. 49, no. 7, pp. 2125–2132, 2013.

[10] M. Zhong and C. G. Cassandras, "Asynchronous distributed optimization with event-driven communication," *IEEE Transactions on Automatic Control*, vol. 55, no. 12, pp. 2735–2750, 2010.

[11] A. Anta and P. Tabuada, "To sample or not to sample: self-triggered control for nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 55, no. 9, pp. 2030–2042, 2010.

[12] X. Wang and M. D. Lemmon, "Self-triggered feedback control systems with finite-gain L_2 stability," *IEEE Transactions on Automatic Control*, vol. 54, no. 3, pp. 452–467, 2009.

[13] C. Nowzari and J. Cortés, "Self-triggered coordination of robotic networks for optimal deployment," *Automatica*, vol. 48, no. 6, pp. 1077–1087, 2012.

[14] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[15] W. Ren and R. W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control*. Communications and Control Engineering, Springer, 2008.

[16] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Applied Mathematics Series, Princeton University Press, 2010.

[17] R. Olfati-Saber and R. M. 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.

[18] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1291–1297, 2012.

[19] G. S. Seybotha, D. V. Dimarogonas, and K. H. Johansson, "Event-based broadcasting for multi-agent average consensus," *Automatica*, vol. 49, no. 1, pp. 245–252, 2013.

[20] E. Garcia, Y. Cao, H. Yu, P. Antsaklis, and D. Casbeer, "Decentralised event-triggered cooperative control with limited communication," *International Journal of Control*, vol. 86, no. 9, pp. 1479–1488, 2013.

[21] C. Nowzari and J. Cortés, "Distributed event-triggered coordination for average consensus on weight-balanced digraphs," *Automatica*, vol. 68, pp. 237–244, 2016.

[22] X. Meng, L. Xie, Y. C. Soh, C. Nowzari, and G. J. Pappas, "Periodic event-triggered average consensus over directed graphs," in *IEEE Conf. on Decision and Control*, (Osaka, Japan), pp. 4151–4156, Dec. 2015.

[23] P. V. Teixeira, D. V. Dimarogonas, K. H. Johansson, and J. Sousa, "Event-based motion coordination of multiple underwater vehicles under disturbances," in *IEEE OCEANS*, (Sydney, Australia), pp. 1–6, 2010.

[24] A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson, "Control of multi-agent systems with event-triggered cloud access," in *European Control Conference*, (Linz, Austria), pp. 954–961, 2015.

[25] C. Nowzari and J. Cortés, "Zeno-free, distributed event-triggered communication and control for multi-agent average consensus," in *American Control Conference*, (Portland, OR), pp. 2148–2153, 2014.

[26] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 3 ed., 2002.