Coordination of multi-agent systems via asynchronous cloud communication

Sean L. Bowman Cameron Nowzari George J. Pappas

***Abstract*— In this work we study a multi-agent coordination problem in which agents are only able to communicate with each other intermittently through a cloud server. To reduce the amount of required communication, we develop a self-triggered algorithm that allows agents to communicate with the cloud only when necessary rather than at some fixed period. Unlike the vast 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 themselves. Using a notion of "promises" about future control inputs, agents are able to keep track of higher quality estimates about their neighbors allowing them to stay disconnected from the cloud for longer periods of time while still guaranteeing a positive contribution to the task. We show that our self-triggered coordination algorithm guarantees that the system asymptotically reaches the desired state. 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." Specifically, we consider the problem of coordinating a number of submarines that only can communicate with a base station while at the surface of the water. While a majority of related works allow for an agent to push information to its neighbors at any desired time, communicating with the outside world when underwater is extremely expensive, if not impossible [1], [2], and so a submarine must perform all communication while surfaced.

Each time a submarine surfaces, it must determine the next time to surface and the control law to use while underwater in order to achieve some desired global task based only on information available on the server at that moment. In this paper we are interested in designing a self-triggered coordination algorithm in which agents autonomously schedule the next time to communicate with the cloud based on currently available information. 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 the context of the multi-agent coordination problem in general, the literature is extensive [3], [4], [5]. In our specific problem of multi-agent consensus, Olfati-Saber and Murray [6] introduce a continuous-time law that guarantees consensus convergence on undirected as well as weight-balanced digraphs. However, the majority of these works assume agents can continuously, or at least periodically, obtain information about their neighbors.

A useful tool for determining discrete communication times in this manner is event-triggered control, where an algorithm tunes controller executions to the state evolution of a given system, see e.g., [7], [8]. In particular, event-triggered control has been successfully applied to multi-agent systems to reduce overall communication, sensing, and/or actuation effort of the agents. In [9], the authors formulate a threshold on system error to determine when control signals need to be updated. Event-triggered ideas have also been applied to the acquisition of information rather than control. Several approaches [10], [11], [12] utilize periodically sampled data to reevaluate the controller trigger.

Event-triggered approaches generally require the persistent monitoring of some triggering function as new information is being obtained. Unfortunately, this is not directly applicable to our setup because the submarines only get new information when they surface. Instead, self-triggered control [13], [14], [15] removes the need to continuously monitor the triggering function, instead requiring each agent to compute its next trigger time based solely on the information available at the previously triggered sample time.

The first to apply these ideas to consensus, Dimarogonas et al. [16], remove the need for continuous control by introducing an event-triggered rule to determine when an agent should update its control signal, however still requiring continuous information about their neighbors. In [17], the authors further remove the need for continuous neighbor state information, creating a time-dependent triggering function to determine when to broadcast information. The authors in [18] similarly broadcast based on a state-dependent triggering function. Recently, these ideas have been extended from undirected graphs to arbitrary directed ones [12], [19], [20].

A major drawback of all aforementioned works is that they require all agents to be "listening," or available to receive information, at all times. Specifically, when any agent decides to broadcast information to its neighbors, it is assumed that all neighboring agents in the communication graph are able to instantaneously receive that information. Instead, we are interested in a situation where in between surfacings when disconnected from the cloud an agent has zero ability to communicate with other agents.

In [21], the authors study a very similar problem to the one we consider here but develop an event-triggered solution in which all Autonomous Underwater Vehicles (AUVs) must surface together at the same time. This problem has very recently been looked at in [22], [23] where the authors utilize event- and self-triggered coordination strategies to determine when the AUVs should resurface. In [22], a time-dependent triggering rule $\beta(\sigma_0, \sigma_1, \lambda_0, t)$ is developed that

Sean L. Bowman is with the Computer and Information Science department, and Cameron Nowzari and George J. Pappas are with the Electrical and Systems Engineering Department, University of Pennsylvania, Philadelphia, {seanbow,cnowzari,pappasg}@seas.upenn.edu.

ensures practical convergence (in the presence of noise) of the whole system to the desired configuration. Instead, the authors in [23] develop a state-dependent triggering rule with no explicit dependence on time; however, the self-triggered algorithm developed there is not guaranteed to avoid Zeno behaviors which makes it an incomplete solution to the problem. In this work we incorporate ideas of promises from team-triggered control [24], [25] to develop a state-dependent triggering rule that guarantees asymptotic convergence to consensus while ensuring that Zeno behavior is avoided.

Statement of contributions: Our main contribution is the development of a novel distributed team-triggered algorithm that combines ideas from self-triggered control with a notion of "promises." These promises allow agents to make better decisions since they have higher quality information about their neighbors in general. Our algorithm incorporates these promises into the state-dependent trigger to determine when they should communicate with the cloud. In contrast to [22], [23], our algorithm uses a state-dependent triggering rule with no explicit dependence on time, no global parameters, and no possibility of Zeno behavior.

In general, distributed event- and self-triggered algorithms are designed so that agents are *never* contributing negatively to the global task, generally defined by the evolution of a Lyapunov function V. Instead, we actually allow an agent to be contributing negatively to the global task temporarily as long as it is accounted for by its net contribution over time. Our algorithm guarantees the system converges asymptotically to consensus while ensuring that Zeno executions cannot occur. Finally, we illustrate our results through simulations.

II. Problem Statement

We consider an N agent system with single-integrator dynamics $\dot{x}_i(t) = u_i(t)$ for all $i \in \{1, \dots, N\}$, where we are interested in reaching a consensus configuration, i.e. where $\|x_i(t) - x_j(t)\| \to 0$ as $t \to \infty$ for all $i, j \in \{1, \dots, N\}$. For simplicity, we consider scalar states $x_i \in \mathbb{R}$, but all results are extendable to arbitrary dimensions.

Given a connected communication graph $\mathcal{G}$, it is well known [6] that the distributed continuous control law

$$u_i(t) = -\sum_{j \in \mathcal{N}_i} (x_i(t) - x_j(t)) \tag{1}$$

drives each agent of the system to asymptotically converge to the average of the agents' initial conditions. 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.

Several recent works have been aimed at relaxing these requirements [12], [19], [20], [17]. However, they all require agents to be "listening" continuously to their neighbors, i.e. when an event is triggered by one agent, its neighbors are immediately aware and can take action accordingly.

Unfortunately, as we assume here that agents are unable to perform any communication while submerged, we cannot continuously detect neighboring events. 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).

t_i^{last}	Last time agent i surfaced
t_i^{expire}	Control expiration time of agent i
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
$M_i(t_i^{\text{last}})$	Most recent control bound promise from agent i

TABLE I

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

Let $\{t_i^\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$ be the sequence of times at which agent i surfaces. Then, our algorithm is based on a piecewise constant implementation of the controller (1) given by

$$u_i^\star(t) = -\sum_{j \in \mathcal{N}_i} (x_i(t_i^\ell) - x_j(t_i^\ell)), \quad t \in [t_i^\ell, t_i^{\ell+1}). \tag{2}$$

Remark II.1 Later we will allow the control input $u_i(t)$ to change in a limited way while agent i is submerged, but for now we assume the control is piecewise constant on the intervals $[t_i^\ell, t_i^{\ell+1})$. Motivation for and details behind changing the control while submerged are discussed in section III-B.•

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

A. Cloud communication model

We assume that there exists a base station or "cloud" that agents are able to upload data to and download data from when they surface. At any 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. The server also contains a control expiration time $t_i^{\text{expire}} \leq t_i^{\text{next}}$ and a *promise* M_i for each agent i which will be explained in Section III where we develop the self-triggered coordination algorithm. This information is summarized in Table I. For simplicity, we assume that agents can download/upload information to/from the cloud instantaneously.

While the link is open, agent i downloads all the information in Table I for each neighbor $j \in \mathcal{N}_i$. Agent i then (instantaneously) computes its control signal $u_i(t_i^\ell)$ and next surfacing time $t_i^{\ell+1}$ such that it knows it will make a net positive contribution to the consensus over the interval $[t_i^\ell, t_i^{\ell+1})$. Finally, before closing the communication link and diving, agent i calculates a promise M_i bounding its future control inputs and uploads all data to the server. The goal of this paper is to design a self-triggered algorithm picking

these control inputs and times such that as $t \to \infty$,

$$|x_j(t) - x_i(t)| \to 0 \tag{3}$$

for all agents $i, j \in \{1, \dots, N\}$. In the next section we describe this algorithm in detail.

III. Distributed Trigger Design

Consider the objective function

$$V(x(t)) = \frac{1}{2} x^T(t) L x(t), \tag{4}$$

where L is the Laplacian of the connected communication graph $\mathcal{G}$. Note that $V(x) \geq 0$ for all $x \in \mathbb{R}^N$ and $V(x) = 0$ if and only if $x_i = x_j$ for all $i, j \in \{1, \dots, N\}$. Thus, the function $V(x)$ encodes the objective of the problem and we are interested in driving $V(x) \to 0$. For simplicity, we drop the explicit dependence on time when referring to time t.

Taking the derivative of V with respect to time, we have

$$\dot{V} = \dot{x}^T L x = -\sum_{i=1}^{N} \dot{x}_i \sum_{j \in \mathcal{N}_i} (x_j - x_i).$$

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

$$\dot{V}_i \triangleq -\dot{x}_i \sum_{j \in \mathcal{N}_i} (x_j - x_i). \tag{5}$$

Note that we have essentially distributed $\dot{V}$ in a way that clearly shows how each agent's motion contributes to the global objective, allowing us to write

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

Ideally, we now wish to design a self triggered algorithm such that $\dot{V}_i(x(t)) \leq 0$ for all agents i at all times t. Thus at surfacing time t_i^ℓ, agent i must determine $t_i^{\ell+1}$ and $u_i(t)$ such that $\dot{V}_i(t) \leq 0$ for all $t \in [t_i^\ell, t_i^{\ell+1})$.

While in the full algorithm, we allow an agent to modify its control by setting it to 0 while still submerged, for clarity in this subsection we assume the control is constant on the entire submerged interval and ignore the control "expiration time" t_i^{expire}; its motivation and (minor) modifications to the algorithm will be described in section III-B.

Note that given the information agent i downloaded from the server at time t_i^ℓ, it is able to exactly compute the state of a neighboring agent $j \in \mathcal{N}_i$ up to the time it resurfaces t_j^{next}. For any $t_i^\ell \leq t < t_j^{\text{next}}$,

$$x_j(t) = x_j(t_j^{\text{last}}) + u_j(t_j^{\text{last}})(t - t_j^{\text{last}}). \tag{6}$$

Let $T_i = \min_{j \in \mathcal{N}_i} t_j^{\text{next}}$ be the first time at which any neighboring agent j is scheduled to surface. Thus, agent i can calculate the position of its neighbors exactly until time T_i. For times t with $t_i^\ell \leq t < T_i$, we can then write the local objective function contribution as

$$\begin{aligned} \dot{V}_i = -u(t_i^\ell) \sum_{j \in \mathcal{N}_i} \big[&(x_j(t_i^\ell) - x_i(t_i^\ell)) + \\ &(t - t_i^\ell)(u_j(t_j^{\text{last}}) - u_i(t_i^\ell)) \big]. \end{aligned} \tag{7}$$

Let $t^\star$ be the smallest time $t^\star \geq t_i^\ell$ such that $\dot{V}_i \leq 0$, assuming all neighbor agents continue to use a constant control, is not satisfied (this time is easily computable with known information from (7)). If $t^\star \leq T_i$, then agent i simply sets $t_i^{\text{next}} = t_i^{\ell+1} = t^\star$, and by continuity of $\dot{V}_i$ it is guaranteed that $\dot{V}_i(t) \leq 0$ for all $t \in [t_i^\ell, t_i^{\ell+1})$.

If $t^\star > T_i$, however, it can no longer be guaranteed that $\dot{V}_i(t) \leq 0$ for $t > T_i$. For any $t > T_i$, we can write

$$x_j(t) = x_j(T_i) + \int_{T_i}^{t} \dot{x}_j(\tau) d\tau, \tag{8}$$

and also, for $t^\star \leq t \leq t_i^{\ell+1}$,

$$x_i(t) = x_i(T_i) + u_i(t_i^{\text{last}})(t - T_i). \tag{9}$$

We can then write $\dot{V}_i(t)$ as

$$\begin{aligned} \dot{V}_i(t) = -u_i(t_i^\ell) \Bigg[&\sum_{j \in \mathcal{N}_i} (x_j(T_i) - x_i(T_i)) + \\ &\sum_{j \in \mathcal{N}_i} \int_{T_i}^{t} \dot{x}_j(\tau) d\tau - u_i(t_i^\ell)(t - T_i) \Bigg]. \end{aligned} \tag{10}$$

Let us first consider the case where $u_i(t_i^\ell) < 0$. We see that $\dot{V}_i \leq 0$ iff the bracketed quantity in (10) is ≤ 0. While agent i is submerged, however, agent j is autonomously updating its control signal; thus, agent i does not have access to $\dot{x}_j(t)$, making it hard to determine how to move without surfacing. To remedy this, we borrow an idea of *promises* from team-triggered control [25]. Suppose that although we don't know $\dot{x}_j(t)$ exactly for $t > T_i$, we know some bound $M_j(t) \geq 0$ such that $|\dot{x}_j(t)| \leq M_j(t)$. Then,

$$\int_{T_i}^{t} \dot{x}_j(\tau) d\tau \leq \int_{T_i}^{t} M_j(t) d\tau = M_j(t)(t - T_i). \tag{11}$$

Using this bound, we get the following sufficient condition for $\dot{V}_i \leq 0$:

$$(t - T_i) \sum_{j \in \mathcal{N}_i} \big(u_i(t_i^\ell) - M_j(t) \big) \geq \sum_{j \in \mathcal{N}_i} (x_j(T_i) - x_i(T_i)). \tag{12}$$

If $u_i(t_i^\ell) > 0$, we similarly get

$$(t - T_i) \sum_{j \in \mathcal{N}_i} \big(u_i(t_i^\ell) + M_j(t) \big) \leq \sum_{j \in \mathcal{N}_i} (x_j(T_i) - x_i(T_i)). \tag{13}$$

Letting $T_i^\star$ be the smallest $T_i^\star \geq T_i$ such that (12) (or (13)) is no longer satisfied, i.e. the smallest time such that we can no longer guarantee that $\dot{V}_i \leq 0$, we know that $\dot{V}_i \leq 0$ on all of $[t_i^\ell, T_i^\star]$ by continuity of $\dot{V}_i$.

It is additionally possible to allow agent i to remain submerged for longer by allowing $\dot{V}_i$ to temporarily become positive as long as we select $t_i^{\ell+1}$ such that the total contribution to the objective function on the interval $[t_i^\ell, t_i^{\ell+1})$,

$$\Delta V_i^\ell \triangleq \int_{t_i^\ell}^{t_i^{\ell+1}} \dot{V}_i(\tau) d\tau, \tag{14}$$

is nonpositive.

Let B_i be the contribution from agent i to the objective

function between t_i^ℓ and T_i, i.e.

$$B_i = \int_{t_i^\ell}^{T_i} \dot{V}_i(\tau) d\tau. \tag{15}$$

Note here we are specifically interested in the case when $t^\star > T_i$, and so B_i is guaranteed to be negative. Similarly, let $C_i(t)$ be the contribution from T_i to t, i.e.

$$C_i(t) = \int_{T_i}^{t} \dot{V}_i(\tau) d\tau. \tag{16}$$

Using the bounds $M_j(t)$, it is possible to bound $C_i(t)$:

$$C_i(t) \leq \overline{C}_i(t) = \frac{\beta_i}{2}(t - T_i)^2 + \gamma_i(t - T_i) \tag{17}$$

where

$$\beta_i = \sum_{j \in \mathcal{N}_i} |u_i(t_i^{\text{last}})| M_j(t) + u_i(t_i^{\text{last}})^2 \tag{18}$$

and

$$\gamma_i = u_i(t_i^{\text{last}}) \sum_{j \in \mathcal{N}_i} \left(x_j(T_i) - x_i(T_i) \right). \tag{19}$$

Let T_i^{total} be the smallest time $T_i^{\text{total}} \geq T_i$ such that

$$B_i + \overline{C}_i(T_i^{\text{total}}) < 0 \tag{20}$$

is no longer satisfied, i.e., the first time at which agent i can no longer guarantee it is still making a positive contribution to the global objective since the last time it surfaced. Setting $t_i^{\ell+1} = T_i^{\text{total}}$ thus ensures that the total contribution from agent i to the global objective function, $\Delta V_i^\ell = B_i + C_i(t_i^{\ell+1})$, is nonpositive.

Selecting $t^{\ell+1} = T_i^\star$ ensures that $\dot{V}_i < 0$ over the submerged interval, ensuring that agent i is making progress towards the global objective at all t. Selecting $t^{\ell+1} = T_i^{\text{total}}$ is a trade-off; while this time allows the agent to remain submerged for longer, as it allows some positive contribution to the objective function, overall progress is slower. Thus, we propose a tuning parameter $\sigma_i \in [0, 1]$, selecting a time $t_i^{\ell+1}$ such that $T_i^\star \leq t^{\ell+1} \leq T_i^{\text{total}}$:

$$t_i^{\ell+1} = (1 - \sigma_i) T_i^\star + \sigma_i T_i^{\text{total}}. \tag{21}$$

By continuity of $\dot{V}_i$ and the definitions of $T_i^\star, T_i^{\text{total}}$, it is guaranteed that $T_i^\star \leq t_i^{\ell+1} \leq T_i^{\text{total}}$ and that we still have $B_i + C_i(t_i^{\ell+1}) \leq 0$. Setting all σ_i near 0 allows faster convergence with more frequent surfacing, while σ_i near 1 results in slower convergence but less frequent surfacing.

A. Selecting promises M_j

As it isn't possible in general for agent i to bound a neighbor agent j's future control inputs from past state and control information, instead each agent makes a *promise* M_i about its future control inputs each time it connects to the server. Let M_i^ℓ be the promise made by agent i at time t_i^ℓ. From equations (12), (13), (20), it is clear that the smaller M_j is for any $j \in \mathcal{N}_i$, the longer agent i is able to stay submerged. However, limiting the control too much below the ideal control (2) will slow convergence.

As a balance, at time t_i^ℓ agent i sets its promise to be exactly the magnitude of its ideal control input: $M_i^\ell = \left| u_i^\star(t_i^\ell) \right|$. Note however that this does not mean agent i can use its ideal control law at all times and still abide by previous promises it has made; if the new desired input and promise is greater in magnitude than its previous promise, to remain truthful to previous promises agent i must wait until the new promise has been received by all of its neighbors when they surface before it can use its desired control input.

Let τ_{ij}^ℓ be the time that agent j sees agent i's ℓth promise, i.e. $\tau_{ij}^\ell = t_j^{\text{next}}(t_i^\ell)$. When submerging for an interval $[t_i^\ell, t_i^{\ell+1})$, agent i needs to guarantee that all promises M_i currently believed by $j \in \mathcal{N}_i$ are abided by.

Let $p_{ij}^{\text{last}}(t)$ be the index of the most recent promise by agent i that agent j is aware of at time t, i.e.

$$p_{ij}^{\text{last}}(t) = \underset{\ell \,:\, \tau_{ij}^\ell \leq t}{\arg\max}\ \tau_{ij}^\ell, \tag{22}$$

and let $\mathcal{P}_i^\ell$ be the set of promise indices that agent i must abide by when submerging on $[t_i^\ell, t_i^{\ell+1})$, i.e.

$$\mathcal{P}_i^\ell = \left\{\ p_{ij}^{\text{last}}(t) \mid j \in \mathcal{N}_i, t \in [t_i^\ell, t_i^{\ell+1})\ \right\}. \tag{23}$$

To abide by all promises that agent i's neighbors believe about its controls, then, it simply needs to bound its control input magnitude by

$$u_i^{\max}(t_i^\ell) = \min_{k \in \mathcal{P}_i^\ell} M_i^k. \tag{24}$$

With this maximum, the actual control law used and uploaded by agent i on the interval $[t_i^\ell, t_i^{\ell+1})$ is given by bounding the ideal control magnitude by $u_i^{\max}(t_i^\ell)$, or

$$u_i(t_i^\ell) = \begin{cases} u_i^\star(t_i^\ell) & \left| u_i^\star(t_i^\ell) \right| \leq u_i^{\max}(t_i^\ell), \\ u_i^{\max}(t_i^\ell) \frac{u_i^\star(t_i^\ell)}{\left| u_i^\star(t_i^\ell) \right|} & \text{otherwise.} \end{cases} \tag{25}$$

B. Avoiding Zeno behavior

While the presented method of selecting surfacing times guarantees convergence, it is susceptible to Zeno behavior, i.e. requiring some agent i to surface an infinite number of times in a finite time period. To avoid this behavior, we introduce a fixed dwell time $T_i^{\text{dwell}} > 0$, and force each agent to remain submerged for at least a duration of T_i^{dwell}. Unfortunately, this means that in general, there may be times at which an agent i is forced to remain submerged even when it does not know how to move to contribute positively to the global task (or it may not even be possible if it is at a local minimum). Remarkably, from the way have have distributed $\dot{V}$ using (5), if agent i sets $u_i(t) = 0$ its instantaneous contribution to the global objective is exactly 0.

Thus, we allow an agent's control to change while it is submerged and modify the control law described in the previous section as follows. If the chosen ideal surfacing time $t_i^{\text{ideal}} = (1 - \sigma_i) T_i^\star + \sigma_i T_i^{\text{total}}$ is greater than or equal to $t_i^\ell + T_i^{\text{dwell}}$, then nothing changes; agent i sets its next surfacing time $t_i^{\ell+1} = t_i^{\text{ideal}}$ and uses the control law (25) on the entire submerged interval.

If, on the other hand, $t_i^{\text{ideal}} < t_i^\ell + T_i^{\text{dwell}}$, we let agent i use the usual control law until t_i^{ideal}, until which it knows it can make a positive contribution. After t_i^{ideal}, we force agent i to remain still until it has been submerged for a dwell time

Algorithm 1: `self-triggered coordination`

At surfacing time t_i^ℓ, agent $i \in \{1,\dots,N\}$ performs:

download $t_j^{\text{last}}, t_j^{\text{expire}}, t_j^{\text{next}}, x_j(t_j^{\text{last}}), u_i(t_j^{\text{last}}), M_j\ \forall j \in \mathcal{N}_i$ from cloud
compute neighbor positions $x_j(t_i^\ell)$ using (27)
compute ideal control $u_i^\star(t_i^\ell) = -\sum_{j\in\mathcal{N}_i}(x_i(t_i^\ell) - x_j(t_i^\ell))$
compute $u_i^{\max}(t_i^\ell)$ using (24) and saved τ_{ij} data
compute control $u_i(t_i^\ell)$ with (25)
compute $T_i = \min_{j\in\mathcal{N}_i} t_j^{\text{next}}$
compute $t^\star$ using (7) as first time when $\dot{V}_i \le 0$ is no longer satisfied
if $t^\star < T_i$ **then**
    set $t_i^{\text{ideal}} = t^\star$
else
    if $u_i(t_i^\ell) = 0$ **then**
        set $t_i^{\text{next}} = T_i + T^{\text{dwell}}$
    else
        if $u_i(t_i^\ell) < 0$ **then**
            compute $T_i^\star$ as first time when (12) is no longer satisfied
        else
            compute $T_i^\star$ as first time when (13) is no longer satisfied
        end if
        compute B_i using (15)
        compute T_i^{total} as first time when (20) is no longer satisfied
        set $t_i^{\text{ideal}} = (1-\sigma_i)T_i^\star + \sigma_i T_i^{\text{total}}$
    end if
end if
if $t_i^{\text{ideal}} < t_i^\ell + T_i^{\text{dwell}}$ **then**
    set $t_i^{\text{expire}} = t_i^{\text{ideal}}$
    set $t_i^{\ell+1} = t_i^\ell + T_i^{\text{dwell}}$
else
    set $t_i^{\text{expire}} = t_i^{\ell+1} = t_i^{\text{ideal}}$
end if
upload promise $M_i = |u_i^\star(t_i^\ell)|$ to cloud
upload $t_i^{\text{last}} = t_i^\ell$, $t_i^{\text{next}} = t_i^{\ell+1}$, t_i^{expire}, $u_i(t_i^\ell)$, $x_i(t_i^\ell)$ to cloud
dive and set $u_i(t) = u_i(t_i^\ell)$ for $t \in [t_i^\ell, t_i^{\text{expire}})$, $u_i(t) = 0$ for $t \in [t_i^{\text{expire}}, t_i^{\ell+1})$

duration. In other words, we set $t^{\text{next}} = t_i^\ell + T_i^{\text{dwell}}$, $t^{\text{expire}} = t_i^{\text{ideal}}$ and use control law (25) on the interval $[t_i^\ell, t_i^{\text{expire}})$.

For $t \in [t_i^{\text{expire}}, t_i^{\ell+1})$, we then set $u_i(t) = 0$, and note that because $\dot{V}_i(t) = 0$ on this interval we still have the desired contribution to the global objective

$$\int_{t_i^\ell}^{t_i^{\ell+1}} \dot{V}_i(\tau)d\tau = \int_{t_i^\ell}^{t_i^{\text{expire}}} \dot{V}_i(\tau)d\tau < 0. \tag{26}$$

Agent i is then still able to calculate the position of any neighbor j exactly for any $t < t_j^{\text{next}}$ using information available on the cloud by

$$x_j(t) = \begin{cases} x_j(t_j^{\text{last}}) + u_j(t_j^{\text{last}})(t - t_j^{\text{last}}) & t < t_j^{\text{expire}}, \\ x_j(t_j^{\text{last}}) + u_j(t_j^{\text{last}})(t_j^{\text{expire}} - t_j^{\text{last}}) & \text{otherwise.} \end{cases} \tag{27}$$

An overview of the fully synthesized self-triggered coordination algorithm is presented in Algorithm 1. Next, we present the main convergence result of this algorithm. The proof is omitted for space considerations.

Theorem III.1 *Given the dynamics $\dot{x}_i(t) = u_i(t)$ and $\mathcal{G}$ connected, if the sequence of update times $\{t_i^\ell\}$ and control laws $u_i(t_i^\ell)$ are determined by Algorithm 1 for all $i \in \{1,\dots,N\}$, then*

$$|x_i(t) - x_j(t)| \to 0 \tag{28}$$

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

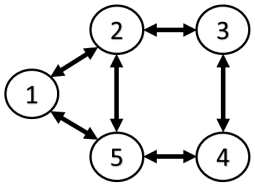

Fig. 1. Simulated communication network

IV. Simulation

In this section we simulate a system of 5 agents with initial condition $x = [9\ -2\ 0.5\ 8.5\ 4]^T$ for a total time of 6 seconds, and with all $\sigma_i = \sigma$ for three different values of σ: 0.01, 0.5, and 0.99. The topology of the communication network is shown in figure 1. For comparison, we ran the simulation from the same configuration with a simple periodic triggering rule where each agent surfaces every T seconds and uses the constant control law $u_i^\star(t_i^\ell)$ on each submerged interval. We show results here for $T = 0.2$, 0.35, and 0.4 seconds. Note that for the undirected graph in figure 1 the system will converge as long as $T < T^* = 2/\lambda_{\max}(L) = 0.4331$.

The cumulative number of surfacings by all agents up to time t, denoted $N_S(t)$, is shown in figure 2 for our algorithm and the periodic trigger. As expected, as a higher value of σ allows an agent to stay submerged for longer, $N_S(t)$ is decreasing in σ.

The evolution of the objective function $V(x(t))$ for the same six configurations described above is displayed in figure 3. Again, as expected, the lower the value of σ, the more quickly the objective $V(x(t))$ decreases, and values of σ farther from 0 actually allow the objective to momentarily increase. It is interesting to note that when comparing against our self-triggering algorithm against the periodic algorithm with a period that results in a similar number of surfacings (e.g. $\sigma = 0.5$ and $T = 0.35$ or $\sigma = 0.01$ and $T = 2$), our algorithm results in the objective $V(x(t))$ decreasing more rapidly. This shows that we are able to yield better performance with a similar amount of communication. Additionally, determining the minimum period $T^* = 0.4331$ requires global information making it even more undesirable. These benefits are in addition to the other clear advantage of our algorithm being naturally asynchronous and not requiring agents to be continuously listening.

A single agent's control law (agent 5) from a run of our algorithm with $\sigma = 0.5$ is shown in figure 4, along with its "promise" currently on the cloud server. As can be seen, there exists a lag between when the promised control max $M_5(t)$ increases and when the actual control increases likewise. While $M_5(t)$ represents the ideal control that the agent would use, it is still bound to a previous promise until the newer one propagates to all neighbor agents.

V. Conclusion

We have presented a novel self-triggering algorithm that, given only the ability to communicate asynchronously at discrete intervals through a cloud server, provably drives a set of agents to consensus without Zeno behavior. Unlike most previous work, we do not require an agent to be able to listen continuously, instead only being able to receive information

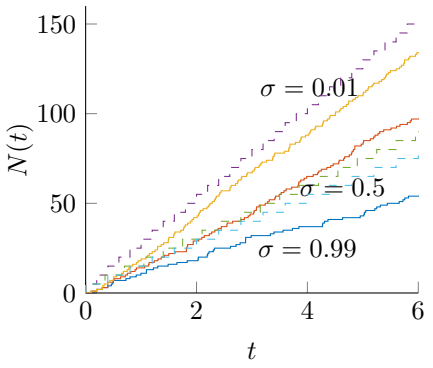

Fig. 2. Cumulative surfacings up t for $\sigma \in \{0.01, 0.5, 0.99\}$, and for a periodic triggering rule with $T \in \{0.2, 0.35, 0.4\}$. Our surfacings are solid lines, periodic as dashed. Periodic plots are stacked vertically with $T = 0.2, 0.35, 0.4$ from top to bottom (or in purple, green, and blue, respectively, with color).

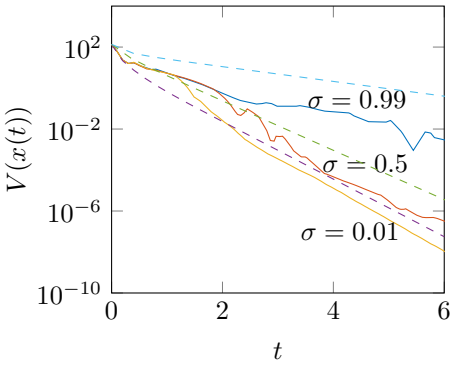

Fig. 3. Objective function $V(x(t))$ with $\sigma \in \{0.01, 0.5, 0.99\}$, as well as for a periodic triggering rule with $T \in \{0.2, 0.35, 0.4\}$. Our runs are solid lines, periodic are dashed. Periodic results are stacked vertically with $T = 0.4, 0.35, 0.2$ from top to bottom (or in blue, green, and purple, respectively, with color).

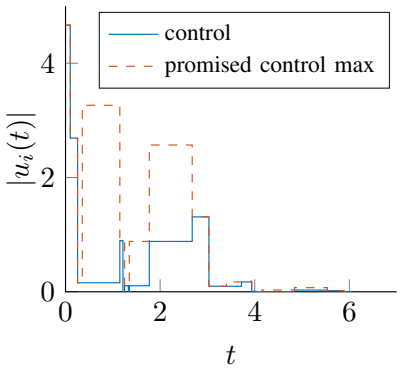

Fig. 4. Magnitude of control law in use by agent $i = 5$, $|u_5(t)|$, as well as its current promise on the cloud server $M_5(t)$.

at its discrete surfacing times. Through the use of control promises, we are able to bound the states of neighboring agents, allowing an agent to remain submerged until its total contribution to the consensus would become detrimental. The given algorithm requires no global parameters, and is fully distributed, requiring no computation to be done off of each local platform. Simulation results show the effectiveness of the proposed algorithm.

In the future, we are interested in investigating control laws different from (25) that may be able to provide more infrequent surfacings or faster convergence. We are additionally interested in methods to reach approximate consensus rather than true asymptotic consensus, and guaranteeing no Zeno behavior without the introduction of a dwell time.

Acknowledgments

This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) 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] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, pp. 215–233, Jan 2007.

[4] W. Ren and R. W. Beard, *Distributed consensus in multi-vehicle cooperative control : theory and applications*. Communications and control engineering, London: Springer, 2008.

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

[6] 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, pp. 1520–1533, Sept 2004.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] C. Nowzari and G. J. Pappas, "Multi-agent coordination with asynchronous cloud access," in *American Control Conference*, (Boston, MA), June 2016. Submitted.

[24] C. Nowzari and J. Cortés, "Self-triggered and team-triggered control of networked cyber-physical systems," in *Event-Based Control and Signal Processing* (M. Miskowicz, ed.), Embedded Systems, Boca Raton, FL: CRC Press, 2015.

[25] C. Nowzari and J. Cortés, "Team-triggered coordination for real-time control of networked cyberphysical systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 1, pp. 34–47, 2016.