ARTICLE TEMPLATE

A Class of Distributed Event-Triggered Average Consensus Algorithms for Multi-Agent Systems

Ping Xu, Cameron Nowzari, Zhi Tian

Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA, 22030, USA

ARTICLE HISTORY
Compiled November 26, 2019

ABSTRACT
This paper proposes a class of distributed event-triggered algorithms that solve the average consensus problem in multi-agent systems. By designing events such that a specifically chosen Lyapunov function is monotonically decreasing, event-triggered algorithms succeed in reducing communications among agents while still ensuring that the entire system converges to the desired state. However, depending on the chosen Lyapunov function the transient behaviors can be very different. Moreover, performance requirements also vary from application to application. Consequently, we are instead interested in considering a class of Lyapunov functions such that each Lyapunov function produces a different event-triggered coordination algorithm to solve the multi-agent average consensus problem. The proposed class of algorithms all guarantee exponential convergence of the resulting system and exclusion of Zeno behaviors. This allows us to easily implement different algorithms that all guarantee correctness to meet varying performance needs. We show that our findings can be applied to the practical clock synchronization problem in wireless sensor networks (WSNs) and further corroborate their effectiveness with simulation results.

KEYWORDS
Event-triggered control, distributed coordination, multi-agent consensus, varying performance needs, clock synchronization.

1. Introduction

The consensus problem of multi-agent systems where a group of agents are required to agree upon certain quantities of interest finds broad applications in areas such as unmanned vehicles, mobile robots, and wireless sensor networks (WSNs) (Liang, Wang, Shen, & Liu, 2012; Olfati-Saber & Jalalkamali, 2012; Peng, Wen, Rahmani, & Yu, 2015). Toward this problem, one effective and efficient method is the distributed event-triggered coordination approach, which was first proposed in (Dimarogonas & Johansson, 2009), and have been studied extensively over the last decades (Liu, Zhang, Yu, & Sun, 2018; Nowzari & Cortés, 2016; Xie, Xu, Li, & Zou, 2015; Yi, Lu, & Chen, 2016), see references in (Ding, Han, Ge, & Zhang, 2017; Nowzari, Garcia, & Cortés, 2019) for recent advances and more details.

The main idea behind distributed event-triggered algorithms is that the iterative communication between agents and their one-hop neighbors only happens when certain conditions/events are triggered. Through skipping unnecessary communica-

tions, the communication efficiency is increased, and at the same time the desired properties of the system are maintained. The triggering conditions of the event-triggered algorithms can be time-dependent (Seyboth, Dimarogonas, & Johansson, 2013), state-dependent (Liu et al., 2018; Nowzari & Cortés, 2014, 2016), or a combination of both (Girard, 2015; Sun, Huang, Anderson, & Duan, 2016; Yi, Liu, Dimarogonas, & Johansson, 2017). In general, the time-dependent thresholds are easy to design to exclude deadlocks (or Zeno behavior, meaning an infinite number of events triggered in a finite number of time period (Johansson, Egerstedt, Lygeros, & Sastry, 1999)), but require global information to guarantee convergence to exactly a consensus state. While state-dependent thresholds are easier to design, these triggers might be risky to implement as Zeno behavior is harder to exclude. As the occurrence of Zeno behavior is impossible in a given physical implementation, the exclusion of it is therefore necessary and essential to guarantee the correctness of an event-triggered algorithm.

In this paper, we focus on developing event-triggered algorithms with state-dependent triggering thresholds that exclude the Zeno behavior. To be specific, an event-triggered controller with state-dependent triggering thresholds can generally be developed from a given Lyapunov function to maintain stability of a certain system while reducing sampling or communication, using the given Lyapunov function as a certificate of correctness. In other words, all events are triggered based on how we want the given Lyapunov function to evolve in time. However, there are no formal guarantees on the gained efficiency. Moreover, it is known that a Lyapunov function is not unique for a given system, and each individual function may result in a totally different, but equally valid/correct triggering law. Consequently, there are many works that propose one such algorithm based on one function that all have the same guarantee: asymptotic convergence to a consensus state. That means there is no established way to compare the performance of two different event-triggered algorithms that solve the same problem. In particular, given two different event-triggered algorithms that both guarantee convergence, their trajectories and communication schedules may be wildly different before ultimately converging to the desired set of states. There are some new works that are addressing exactly this topic (Borgers, Geiselhart, & Heemels, 2017; Heijmans, Borgers, & Heemels, 2017; Khashooei, Antunes, & Heemels, 2017; Ramesh, Sandberg, & Johansson, 2016), which set the basis for this paper. More specifically, once established methods of comparing the performance of event-triggered algorithms against one another are developed, current available algorithms will likely be revisited to optimize different types of performance metrics. In particular, we notice that different algorithms are better than others in different scenarios when considering metrics such as convergence speed or total energy consumption. Therefore, instead of trying to design only one event-triggered algorithm that simply guarantees convergence, we design an entire class of event-triggered algorithms that can be easily tuned to meet varying performance needs.

Our work is motivated by (Nowzari & Cortés, 2016) that solves the exact problem we consider, i.e., design a distributed event-triggered algorithm with state-dependent triggers for multi-agent systems over weight-balanced directed graphs. We first develop a distributed event-triggered algorithm based on an alternative Lyapunov candidate function, which we name it as **Algorithm 2**. For the algorithm proposed by Nowzari and Cortés (2016), we name it as **Algorithm 1**. Observing that the two algorithms result in different performance for different network topologies, we then parameterize an entire class of Lyapunov functions from the two algorithms and show how each individual function can be used to develop a **Combined Algo-**

rithm. More specifically, choosing any parameter $\lambda \in [0, 1]$ yields an event-triggered algorithm that guarantees convergence. Changing λ can then help achieve varying performance goals while always guaranteeing stability. With the asymptotic convergence and exclusion of Zeno behavior for both **Algorithm 1** and **Algorithm 2**, we establish that the entire class of **Combined Algorithms** also exclude Zeno behavior and guarantee convergence of the system. In addition to the theoretic analysis, we also study the practical clock synchronization problem that exists in WSNs (Dimarogonas & Johansson, 2009), which is crucial especially when operations such as data fusion, power management and transmission scheduling are performed (Kadowaki & Ishii, 2015; Wu, Chaudhari, & Serpedin, 2011). We use various simulations to illustrate the correctness and performance of our proposed algorithms.

The rest of this paper is organized as follows. Section 2 introduces the preliminaries and Section 3 formulates the problem of interest. Section 4 first summarizes the related work (Nowzari & Cortés, 2016) and then proposes a novel strategy based on an alternative Lyapunov function. Section 5 analyzes the non-Zeno behavior and convergence property of the proposed strategy. The combined algorithms that are developed based on the combined Lyapunov functions are proposed in Section 6, followed by a case study of clock synchronization in Section 7. Section 8 presents the simulation results and Section 9 concludes this work.

Notations: $\mathbb{R}$, $\mathbb{R}_{>0}$, $\mathbb{R}_{\geq 0}$ denote the set of real, positive real, and nonnegative real numbers, respectively. $\mathbf{1}_N \in \mathbb{R}^N$ and $\mathbf{0}_N \in \mathbb{R}^N$ denote the $N \times 1$ column vectors with entries all equal to one and zero, respectively. $\|\cdot\|$ denotes the Euclidean norm for vectors or induced 2-norm for matrices. For a finite set S, $|S|$ denotes its cardinality.

2. Preliminaries

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, W\}$ denote a weighted directed graph (or weighted digraph) that is comprised of a set of vertices $\mathcal{V} = \{1, \ldots, N\}$, directed edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, and weighted adjacency matrix $W \in \mathbb{R}_{\geq 0}^{N \times N}$. Given an edge $(i, j) \in \mathcal{E}$, we refer to j as an out-neighbor of i and i as an in-neighbor of j. The sets of out- and in-neighbors of a given agent i are $\mathcal{N}_i^{out}$ and $\mathcal{N}_i^{in}$, respectively. The weighted adjacency matrix W satisfies $w_{ij} > 0$ if $(i, j) \in \mathcal{E}$ and $w_{ij} = 0$ otherwise. A path from vertex i to j is an ordered sequence of vertices such that each intermediate pair of vertices is an edge. A digraph $\mathcal{G}$ is strongly connected if there exists a path from all $i \in \mathcal{V}$ to all $j \in \mathcal{V}$. The out- and in-degree matrices D^{out} and D^{in} are diagonal matrices whose diagonal elements are

$$d_i^{out} = \textstyle\sum_{j \in \mathcal{N}_i^{out}} w_{ij}, \quad d_i^{in} = \textstyle\sum_{j \in \mathcal{N}_i^{in}} w_{ji},$$

respectively. A digraph is weight-balanced if $D^{out} = D^{in}$, and the weighted Laplacian matrix is given by $L = D^{out} - W$.

For a strongly connected and weight-balanced digraph, zero is a simple eigenvalue of L. In this case, we order its eigenvalues as $\lambda_1 = 0 < \lambda_2 \leq \cdots \leq \lambda_N$. Note the following property will be of use later:

$$\lambda_2(L) x^T L^T x \leq x^T L^T L x \leq \lambda_N(L) x^T L^T x. \tag{1}$$

Another property we need is the Young's inequality (Hardy, Littlewood, & Pólya,

1952), which states that given $x, y \in \mathbb{R}$, for any $\varepsilon \in \mathbb{R}_{>0}$,

$$xy \leq \frac{x^2}{2\varepsilon} + \frac{\varepsilon y^2}{2}. \tag{2}$$

3. Problem Statement

Consider the average consensus problem for an N-agent network described by a weight-balanced and strongly connected digraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, W\}$. Without loss of generality, we say that an agent i is able to receive information from neighbors in $\mathcal{N}_i^{out}$ and send information to neighbors in $\mathcal{N}_i^{in}$. Assume that all inter-agent communications are instantaneous and of infinite precision. Let x_i denote the state of agent $i \in \mathcal{V}$ and consider the single-integrator dynamics

$$\dot{x}_i(t) = u_i(t). \tag{3}$$

The well-known distributed continuous control law

$$u_i(t) = -\sum_{j \in \mathcal{N}_i^{out}} w_{ij}(x_i(t) - x_j(t)) \tag{4}$$

drives the states of all agents in the system to asymptotically converge to the average of their initial states (Olfati-Saber & Murray, 2004). However, its implementation requires all agents to continuously access their neighbors' state information and keep updating their own control signals, which is practically unrealistic in terms of both communication and control. To relax both of these requirements, we adopt the modified distributed event-triggered control law (Dimarogonas, Frazzoli, & Johansson, 2012)

$$u_i(t) = -\sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t) - \hat{x}_j(t)), \tag{5}$$

where $\hat{x}_i(t)$ to denote the last broadcast state of agent i and it remains constant between two broadcasts. That is, if we let t_{last} be the last time at which agent i broadcasts its state information and t_{next} be the next time it is going to broadcast, then $\hat{x}_i(t) = x_i(t_{last})$ for $t \in [t_{last}, t_{next})$. With this framework, neighbors of a given agent are able to receive state information from it only when this agent decides to broadcast its state information to them. After receiving the information from their neighbors, agents then update their own control signals.

Along with the above controller (5), each agent i is equipped with a triggering function $f_i(\cdot)$ that takes values in $\mathbb{R}$. Our first objective is to identify triggers that depend on local information only, i.e., on the true state $x_i(t)$, its last broadcast state $\hat{x}_i(t)$, and its neighbors last broadcast state $\hat{x}_j(t)$ for $j \in \mathcal{N}_i$. Specifically, we need to design triggering functions for each agent $i \in \mathcal{V}$ such that an event is triggered as soon as the triggering condition

$$f_i(t, x_i(t), \hat{x}_i(t), \hat{x}_j(t)) > 0 \tag{6}$$

is fulfilled. The triggered event then drives agent i to broadcast its state so that its neighbors can update their states. To do so, the general steps are to identify a Lyapunov function for the system, and then derive triggering rules from the Lyapunov function while maintaining the stability of the system and ensures asymptotic convergence to a consensus state.

Notice that a Lyapunov function is not unique for a given system, and each individual function may result in a totally different, but equally valid/correct triggering law. Moreover, when considering metrics such as convergence speed or total energy consumption, different algorithms are better than others in different scenarios. Since there is no established way to compare the performance of two different event-triggered algorithms that solve the same problem and performance requirements may vary from application to application, therefore, our second objective is to design an entire class of event-triggered algorithms that can be easily tuned to meet varying performance needs. Before presenting our work, we first introduce the algorithm that motivates our work (Nowzari & Cortés, 2016).

4. Distributed Trigger Design

4.1. Related work

The exact same problem of distributed event-triggered coordination for multi-agent systems over weight-balanced digraphs has been studied by Nowzari and Cortés (2016). As their findings are essential in developing our algorithms, we first summarize their algorithm and name it **Algorithm 1**.

The event-triggered law proposed in (Nowzari & Cortés, 2016) is Lyapunov-based, with the Lyapunov candidate function be

$$V_1(x(t)) = \tfrac{1}{2}(x(t)-\bar{x})^T(x(t)-\bar{x}), \tag{7}$$

where $x(t) = (x_1(t), ..., x_N(t))^T \in \mathbb{R}^N$ is the column vector of all agents' states and $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i(0)\mathbf{1}_N$ is the average of all initial conditions.

The derivative of $V_1(x(t))$ takes the form

$$\dot{V}_1(x(t)) = x^T(t)\dot{x}(t) - \bar{x}^T\dot{x}(t) = -x^T(t)L\hat{x}(t) + \bar{x}^TL\hat{x}(t) = -x^T(t)L\hat{x}(t), \tag{8}$$

where $\dot{x}(t) = u(t) = -L\hat{x}(t)$ is the compact vector-matrix form of equation (3) and (5), with $\hat{x}(t) = (\hat{x}_1(t), ..., \hat{x}_N(t))^T \in \mathbb{R}^N$ the vector of last broadcast states of all agents. The second term $\bar{x}^TL\hat{x}(t) = 0$ comes from the fact that the digraph $\mathcal{G}$ is weight-balanced, meaning $\mathbf{1}_N^TL = \mathbf{0}^T$, therefore $\bar{x}^TL\hat{x}(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(0)\mathbf{1}_N^TL\hat{x}(t) = 0$.

Expand (8) and apply Young's inequality (2), $\dot{V}_1(x(t))$ is upper bounded by

$$\dot{V}_1(x(t)) \leq -\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i^{out}} w_{ij}\Big[(1-a_i)(\hat{x}_i(t)-\hat{x}_j(t))^2 - \tfrac{e_i^2(t)}{a_i}\Big], \tag{9}$$

where $a_i \in (0,1)$ and $e_i(t) = \hat{x}_i(t) - x_i(t)$ is the difference between agent i's last broadcast state and its current state at time t.

To make sure that the Lyapunov function $V_1(x(t))$ is monotonically decreasing requires

$$\sum_{j\in\mathcal{N}_i^{out}} w_{ij}\Big[(1-a_i)(\hat{x}_i(t)-\hat{x}_j(t))^2 - \tfrac{e_i^2(t)}{a_i}\Big] \geq 0,$$

for all agents $i \in \mathcal{V}$ at all times, which can be accomplished by enforcing

$$e_i^2(t) \leq \tfrac{a_i(1-a_i)}{d_i^{out}}\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t)-\hat{x}_j(t))^2. \tag{10}$$

It is found in (Nowzari & Cortés, 2016) that by setting $a_i = 0.5$ for all agents, the trigger design will be optimal. Therefore, the triggering function in (Nowzari & Cortés, 2016) is defined as

$$f_i(e_i(t)) = e_i^2(t) - \frac{\sigma_i}{4d_i^{out}} \sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t) - \hat{x}_j(t))^2, \tag{11}$$

where $\sigma_i \in (0,1)$ is a design parameter that affects the flexibility of the triggers. According to the triggering function (11), an event is triggered when $f_i(e_i(t)) > 0$ or when $f_i(e_i(t)) = 0$ and $\phi_i = \sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t) - \hat{x}_j(t))^2 \neq 0$.

Basically, the trigger above makes sure that $\dot{V}_1(x(t))$ is always negative as long as the system has not converged, therefore, **Algorithm 1** guarantees all agents to converge to the average of their initial states, i.e., $\lim_{t\to\infty} x(t) = \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i(0)\mathbf{1}_N$, interested readers are referred to (Nowzari & Cortés, 2016, Theorem 5.3) for more details.

4.2. Proposed new algorithm

As we know, the Lyapunov function is not unique for the stability studying of the same system, and each individual function may result a totally different triggering law. Therefore, we propose a novel triggering strategy named **Algorithm 2** based on an alternative Lyapunov candidate function

$$V_2(x(t)) = \tfrac{1}{2}x(t)^T L^T x(t). \tag{12}$$

The following result characterizes a local condition for all agents in the network such that the Lyapunov candidate function $V_2(x(t))$ is monotonically nonincreasing.

Lemma 4.1. *For $i \in \mathcal{V}$, with $b_i, c_j < \frac{1}{d_i^{out}}$ $\forall i, j \in \mathcal{V}$, define $e_i(t) = \hat{x}_i(t) - x_i(t)$ as in Section 4.1, with $u_i(t)$ given in* (5)*, then*

$$\dot{V}_2(x(t)) \leq -\sum_{i=1}^{N}\left[\delta_i u_i^2(t) - \left(\frac{d_i^{out}}{2b_i} + \frac{d_i^{out}}{2c_i}\right)e_i^2(t)\right), \tag{13}$$

where

$$\delta_i \triangleq 1 - \frac{d_i^{out}b_i}{2} - \sum_{j\in\mathcal{N}_i^{out}} \frac{w_{ij}c_j}{2}. \tag{14}$$

Proof. See Appendix A. □

From Lemma 4.1, a sufficient condition to guarantee the proposed Lyapunov candidate function $V_2(x(t))$ is monotonically decreasing is to ensure that

$$\delta_i u_i^2(t) - \left(\frac{d_i^{out}}{2b_i} + \frac{d_i^{out}}{2c_i}\right)e_i^2(t) \geq 0$$

for all agents $i \in \mathcal{V}$ at all times, or

$$e_i^2(t) \leq \frac{2\delta_i b_i c_i}{(b_i+c_i)d_i^{out}}\left(\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t) - \hat{x}_j(t))\right)^2. \tag{15}$$

The triggering function developed from **Algorithm 2** is therefore derived as

$$f_i(e_i(t)) = e_i^2(t) - \frac{2\sigma_i\delta_i b_i c_i}{(b_i+c_i)d_i^{out}}\Big(\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t)-\hat{x}_j(t))\Big)^2, \tag{16}$$

where $\sigma_i \in (0,1)$ is a design parameter that affects how flexible the trigger is and controls the trade-off between communication and performance. Setting σ_i close to 0 is generally greedy, meaning that the trigger is enabled more frequently and more communications are required, therefore makes agent i contribute more to the decrease of the Lyapunov function $V_2(x(t))$, leading to a faster convergence of the network while setting the value of σ_i close to 1 achieves the opposite results. Note that the roles of b_i, c_i, c_j are beyond system stabilization, they are also important to the trigger's performance. The larger value of $\frac{2\delta_i b_i c_i}{(b_i+c_i)d_i^{out}}$, the less communication shall be needed since it means that the system is more error-tolerant.

Corollary 4.2. *For agent $i \in \mathcal{V}$ with the triggering function defined in* (16)*, if the condition $f_i(e_i) \leq 0$ is enforced at all times, then*

$$\dot{V}_2(x(t)) \leq -\sum_{i=1}^{N}(1-\sigma_i)\delta_i\big(\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t)-\hat{x}_j(t))\big)^2.$$

Similar as the work done in (Nowzari & Cortés, 2016), to avoid the possibility that agent i may miss any triggers, we define an event either by

$$f_i(e_i(t)) > 0 \quad or \tag{17}$$
$$f_i(e_i(t)) = 0 \quad and \quad \phi_i \neq 0 \tag{18}$$

where $\phi_i = \big(\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t)-\hat{x}_j(t))\big)^2$.

We also prescribe the following additional trigger as in (Nowzari & Cortés, 2016) to address the non-Zeno behavior. Let t^i_{last} be the last time at which agent i broadcasts its information to its neighbors. If at some time $t \geq t^i_{last}$, agent i receives information from a neighbor $j \in \mathcal{N}_i^{out}$, then agent i immediately broadcasts its state if

$$t \in (t^i_{last}, t^i_{last} + \varepsilon_i), \tag{19}$$

where

$$\varepsilon_i < \sqrt{\frac{2\sigma_i\delta_i b_i c_i}{(b_i+c_i)d_i^{out}}} \tag{20}$$

is a parameter selected to ensure the exclusion of Zeno behavior, and we will demonstrate how it is designed in the following section.

We summarize the differences between **Algorithm 1** proposed in (Nowzari & Cortés, 2016) and **Algorithm 2** proposed here in Table 1. Once the triggering function and parameters ε_i are chosen for each agent, either algorithm can be implemented using the coordination algorithm provided in Table 2.

Note that both algorithms guarantee exponential convergence and the exclusion of Zeno behavior, as analyzed in Section 5 and in (Nowzari & Cortés, 2016, Section 5). However, except for these similarities, we have no idea which algorithm works better for under varying performance need and initial conditions, which motivates our work in Section 6.

Table 1. Difference between **Algorithm 1** and **Algorithm 2**.

	Triggering function	Parameter design
Algorithm 1	$f_i(e_i) \triangleq e_i^2(t) - \frac{\sigma_i}{4d_i^{out}} \sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t) - \hat{x}_j(t))^2$	$\varepsilon_i < \sqrt{\frac{\sigma_i}{4d_i^{out} w_i^{\max} \lvert\mathcal{N}_i^{out}\rvert}}$
Algorithm 2	$f_i(e_i) \triangleq e_i^2 - \frac{2\sigma_i\delta_i b_i c_i}{(b_i+c_i)d_i^{out}} \Big(\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i - \hat{x}_j \Big)^2$	$\varepsilon_i < \sqrt{\frac{2\sigma_i\delta_i b_i c_i}{(b_i+c_i)d_i^{out}}}$

Table 2. Distributed Event-Triggered Coordination Algorithm.

At all times t, agent $i \in \{1, \ldots, N\}$ performs:

if $f_i(e_i(t)) > 0$ or $(f_i(e_i(t)) = 0$ and $\phi_i \neq 0)$ **then**
  broadcast state information $x_i(t)$ and update control signal $u_i(t)$
end if
if new information $x_j(t)$ is received from some neighbor(s) $j \in \mathcal{N}_i^{out}$ **then**
  if agent i has broadcast its state at any time $t' \in [t - \varepsilon_i, t)$ **then**
    broadcast state information $x_i(t)$
  end if
  update control signal $u_i(t)$
end if

5. Stability Analysis of Algorithm 2

In this section, we show that **Algorithm 2** guarantees that no Zeno behavior exists in the network executions. In addition, we show that when executing **Algorithm 2**, all agents converge exponentially to the average of their initial states.

Proposition 5.1. *(Non-Zeno Behavior) Consider the system* (3) *executing control law* (5)*. The triggering function is given by* (16)*. If the underlying digraph of the system is weight-balanced and strongly connected, then when executing the algorithm described in Table 2, the system with any initial conditions will not exhibit Zeno behavior.*

Proof. To prove that the system does not exhibit Zeno behavior, we need to show that no agent broadcasts its state an infinite number of times in any finite time period. We divide the proof into two steps, the first step shows the existence of that finite time period and gives its value; while in the second step, we show that no information can be transmitted an infinite number of times in that finite time period.

Step 1: This step shows that if an agent does not receive new information from its out-neighbors, its inter-events time is bounded by a positive constant.

Assume that agent $i \in \mathcal{V}$ has just broadcast its state at time t_0, then $e_i(t_0) = 0$. For $t > t_0$, while no new information is received, $\hat{x}_i(t)$ and $\hat{x}_j(t)$ remain unchanged. Given that $\dot{e}_i = -\dot{x}_i$, the evolution of the error is simply

$$e_i(t) = -(t - t_0)\hat{z}_i, \tag{21}$$

where $\hat{z}_i = \sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_j - \hat{x}_i)$. Since we are considering the case that no neighbors of agent i broadcast their states, therefore trigger (19) is irrelevant. We then need to find out the next time point t^* when $f_i(e_i(t^*)) = 0$ and agent i is triggered to broadcast. This can be done following trigger (18). If $\hat{z}_i = 0$, no broadcasts will ever happen because $e_i(t) = 0$ for all $t \geq t_0$. Consider the case when $\hat{z}_i \neq 0$, using (21),

trigger (18) prescribes a broadcast at time $t^* \geq t_0$ that satisfies

$$(t^* - t_0)^2 \hat{z}_i^2 - \frac{2\sigma_i \delta_i b_i c_i}{(b_i + c_i) d_i^{out}} \hat{z}_i^2 = 0,$$

or equivalently

$$(t^* - t_0)^2 = \frac{2\sigma_i \delta_i b_i c_i}{(b_i + c_i) d_i^{out}}.$$

Therefore, we can lower bound the inter-events time by

$$\tau_i = t^* - t_0 = \sqrt{\frac{2\sigma_i \delta_i b_i c_i}{(b_i + c_i) d_i^{out}}},$$

which explains our choice in (20). By this step, if none of agent i's neighbors broadcast, agent i will not be triggered infinitely fast. Next, we show that messages can not be sent infinitely over a finite time period when one or more neighbors of agent i trigger(s).

Step 2: Same as **Step 1**, assume agent i has just broadcast its state at time t_0, thus $e_i(t_0) = 0$. Our reasoning is as follows:

1) If no information is received by time $t_0 + \varepsilon_i < t_0 + \tau_i$, then no trigger happens for agent i.

2) Let us then consider the situation that at least one neighbor of agent i broadcasts its information at some time $t_1 \in (t_0, t_0 + \varepsilon_i)$, which means that agent i would also re-broadcast its information at time t_1 due to trigger (19). Define I as the set in which all agents have broadcast information at time t_1, then as long as no agent $k \in I$ sends new information to any agent in I, agents in I will not broadcast new information for at least $\min_{j \in I} \tau_j$ seconds, which includes the original agent i. As no new information is received by any agent in I by time $t_1 + \min_{j \in I} \varepsilon_j$, there is no problem.

3) Again consider the case that at least one agent k sends new information to some agent $j \in I$ at time $t_2 \in (t_1, t_1 + \min_{j \in I} \varepsilon_j)$, then by trigger (19), all agents in I would also broadcast their state information at time t_2 and agent k will now be added to I. The remaining reasoning is just to repeat what has been reasoned, thus, the only situation for infinite communications to occur in a finite time period is to have a network of infinite agents, which is impossible for the N-agent network we consider.

Therefore, **Step 1** and **Step 2** conclude that **Algorithm 2** excludes Zeno behavior for the network. □

Next we establish the global exponential convergence.

Theorem 5.2. *(Exponential Convergence to Average Consensus). Given the system* (3) *executing Table 2 over a weight-balanced, strongly connected digraph, all agents exponentially converge to the average of their initial states, i.e.* $\lim_{t \to \infty} x(t) = \bar{x}$*, where* $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i(0) \mathbf{1}_N$.

Proof. The triggering events (17) and (18) ensure that

$$\dot{V}_2(x(t)) \leq \sum_{i=1}^{N} (\sigma_i - 1) \delta_i \Big(\sum_{j \in \mathcal{N}_i^{out}} w_{ij} (\hat{x}_i(t) - \hat{x}_j(t)) \Big)^2. \tag{22}$$

To show that the convergence is exponential, we show that the evolution of $V_2(x(t))$ towards 0 is exponential. Omit the time stamp t for simplicity, and define $\sigma_{\max} =$

$\max_{i\in\mathcal{V}}\sigma_i$, $\delta_{\max}=\max_{i\in\mathcal{V}}\delta_i$ to further bound (22):

$$\begin{aligned}\dot{V}_2(x) &\le (\sigma_{\max}-1)\delta_{\max}\sum_{i=1}^{N}\Big(\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i-\hat{x}_j)\Big)^2\\ &= (\sigma_{\max}-1)\delta_{\max}\hat{x}^T L^T L\hat{x}\\ &\le (\sigma_{\max}-1)\delta_{\max}\lambda_2(L)\hat{x}^T L^T\hat{x},\end{aligned}$$

where we use (1) to come up with the last inequality. Note that

$$\begin{aligned}V_2(x)=\frac{1}{2}x^T L^T x &= \frac{1}{2}(\hat{x}-e)^T L^T(\hat{x}-e)\\ &= \frac{1}{2}(\hat{x}^T L^T\hat{x}-\hat{x}^T L^T e-e^T L^T\hat{x}+e^T L^T e)\\ &\le \frac{1}{2}(2\hat{x}^T L^T\hat{x}+2e^T L^T e)\\ &\le \hat{x}^T L^T\hat{x}+\|L\|\|e\|^2.\end{aligned}\tag{23}$$

Substitute (15) into (23), define $d_{\min}^{out}=\min_{i\in\mathcal{V}}d_i^{out}$, $b_{\max}=\max_{i\in\mathcal{V}}b_i$, $c_{\max}=\max_{i\in\mathcal{V}}c_i$, $b_{\min}=\min_{i\in\mathcal{V}}b_i$, and $c_{\min}=\min_{i\in\mathcal{V}}c_i$, using (1), we have

$$\begin{aligned}\hat{x}^T L^T\hat{x}+\|L\|\|e\|^2 &\le \hat{x}^T L^T\hat{x}+\|L\|\frac{2\sigma_{\max}\delta_{\max}b_{\max}c_{\max}}{(b_{\min}+c_{\min})d_{\min}^{out}}\hat{x}^T L^T L\hat{x}\\ &\le \hat{x}^T L^T\hat{x}+\|L\|\frac{2\sigma_{\max}\delta_{\max}b_{\max}c_{\max}}{(b_{\min}+c_{\min})d_{\min}^{out}}\lambda_N(L)\hat{x}^T L^T\hat{x}\\ &= (1+\frac{2\|L\|\sigma_{\max}\delta_{\max}b_{\max}c_{\max}\lambda_N(L)}{(b_{\min}+c_{\min})d_{\min}^{out}})\hat{x}^T L^T\hat{x}.\end{aligned}\tag{24}$$

Relate (23) with (24) gives

$$\begin{aligned}\dot{V}_2(x) &\le (\sigma_{\max}-1)\delta_{\max}\lambda_2(L)\hat{x}^T L^T\hat{x}\\ &\le \frac{(\sigma_{\max}-1)\delta_{\max}\lambda_2(L)}{2(1+\frac{2\|L\|\sigma_{\max}\delta_{\max}b_{\max}c_{\max}\lambda_N(L)}{(b_{\min}+c_{\min})d_{\min}^{out}})}x^T L^T x\\ &= \frac{(\sigma_{\max}-1)(b_{\min}+c_{\min})\delta_{\max}\lambda_2(L)d_{\min}^{out}}{(b_{\min}+c_{\min})d_{\min}^{out}+2\|L\|\sigma_{\max}\delta_{\max}b_{\max}c_{\max}\lambda_N(L)}V_2(x).\end{aligned}\tag{25}$$

Substitute $A=\frac{(\sigma_{\max}-1)(b_{\min}+c_{\min})\delta_{\max}\lambda_2(L)d_{\min}^{out}}{(b_{\min}+c_{\min})d_{\min}^{out}+2\|L\|\sigma_{\max}\delta_{\max}b_{\max}c_{\max}\lambda_N(L)}$ into (25), we have $\dot{V}_2(x(t))\le AV_2(x(t))$, therefore we conclude that $V_2(x(t))\le V_2(x(0))\exp(At)$ and the network converges exponentially to the average of its initial state. □

With the theoretical foundation of **Algorithm 2**, we are now ready to propose a class of event-triggered algorithms that can be tuned to meet varying performance needs under different scenarios.

6. A Class of Event-Triggered Algorithms

As stated in Section 1, for a given system, there are many works studying event-triggered control using Lyapunov functions to reach the goal of maintaining the stability of the system, while increasing the efficiency of the system. However, there is very little work currently available that mathematically quantifies these benefits. Recently, some works began establishing results along this line (Antunes & Heemels, 2014; Khashooei et al., 2017; Ramesh et al., 2016), still this area is in its infancy. In particular, there are not yet established ways to compare the performance of an event-triggered algorithm with another. Consequently, many different algorithms can be proposed to ultimately solve the same problem, while each algorithm is slightly different and produces different trajectories. Specifically in our case, **Algorithm 1** and **Algorithm 2** solve the exact same problem, and offer the exact same guarantees, i.e., they both exclude Zeno behavior and ensure asymptotic convergence of the network. So, which algorithm should we use? Moreover, we have found that depending on the initial conditions and network topology, each algorithm may out-perform the other in terms of different evaluation metrics. In any case, once these performance metrics become better researched, there will likely be more standard ways to mathematically compare the two different algorithms. Therefore, for now, instead of designing only one event-triggered algorithm for the system that only works better in one situation, we aim to design an entire class of algorithms that can easily be tuned to meet varying performance needs.

We do this by parameterizing a set of Lyapunov functions rather than studying only a specific one. To the best of our knowledge, this paper is then a first study of how to design an entire class of algorithms that use different Lyapunov functions to guarantee correctness, with the intention of being able to use the best one at all times. In this paper, we utilize only two Lyapunov functions, however, we can also use as many Lyapunov functions as we want and combine them all to develop the entire class of algorithms.

Specifically, given any $\lambda \in [0,1]$, we define a combined Lyapunov function as

$$V_\lambda(x(t)) = \lambda V_1(x(t)) + (1-\lambda)V_2(x(t)). \tag{26}$$

Accordingly, the derivative of $V_\lambda(x(t))$ takes the form

$$\dot{V}_\lambda(x(t)) = \lambda \dot{V}_1(x(t)) + (1-\lambda)\dot{V}_2(x(t)). \tag{27}$$

Following the steps of deriving the triggering functions in Section 4, the triggering function developed based on the combined Lyapunov function (26) is given by

$$\begin{aligned} f_i(e_i(t)) = e_i^2(t) - \sigma_i \Big[&\frac{\lambda}{4d_i^{out}} \sum_{j \in \mathcal{N}_i^{out}} w_{ij} \Big(\hat{x}_i(t) - \hat{x}_j(t) \Big)^2 + \\ &\frac{(1-\lambda) 2\delta_i b_i c_i}{(b_i + c_i) d_i^{out}} \Big(\sum_{j \in \mathcal{N}_i^{out}} w_{ij} (\hat{x}_i(t) - \hat{x}_j(t)) \Big)^2 \Big]. \end{aligned} \tag{28}$$

We refer to the algorithm developed from the combined Lyapunov function as the **Combined Algorithm** parameterized by λ, with $\lambda \in [0,1]$. Note that $\lambda = 0$ recovers **Algorithm 2** and $\lambda = 1$ recovers **Algorithm 1**.

Similarly, for the **Combined Algorithm**, we use the following events to avoid missing any triggers:

$$f_i(e_i(t)) > 0, \tag{29}$$
$$f_i(e_i(t)) = 0 \quad and \quad \phi_i \neq 0, \tag{30}$$

where, with a slight abuse of notation, $\phi_i = \frac{\lambda}{4d_i^{out}} \sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t) - \hat{x}_j(t))^2 + \frac{(1-\lambda)2\delta_i b_i c_i}{(b_i+c_i)d_i^{out}}(\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i(t) - \hat{x}_j(t)))^2$.

The parameter that bounds the inter-events time and excludes Zeno behavior is also designed:

$$\varepsilon_i < \sqrt{\frac{\lambda\sigma_i}{4d_i^{out}w_i^{\max}|\mathcal{N}_i^{out}|} + \frac{2(1-\lambda)\sigma_i\delta_i b_i c_i}{(b_i+c_i)d_i^{out}}}.$$

Then, with the triggering function (28) and ε_i defined above, the **Combined Algorithm** can also be implemented using Table 2.

Corollary 6.1. *Both **Algorithm 1** and **Algorithm 2** ensure all agents to exponentially converge to the average of their initial states with the proof that their Lyapunov functions converge exponentially. Therefore, as a linear combination of $V_1(x(t))$ and $V_2(x(t))$, $V_\lambda(x(t))$ also converges exponentially, which means that a network executing the **Combined Algorithm** shall converge exponentially to the average of its initial states.*

To illustrate the correctness and effectiveness of **Algorithm 2** and the **Combined Algorithm**, we introduce the fundamental clock synchronization problem that exists in wireless sensor networks (WSNs) as a case study.

7. Case Study: Clock Synchronization

7.1. Background

WSNs are broadly applied in areas such as disaster management, border protection, and security surveillance, to name a few, thanks to their low-cost and collaborative nature (Abbasi & Younis, 2007; Gungor, Lu, & Hancke, 2010). However, the underlying local clocks of these sensors are often in disagreement due to the imperfections of clock oscillators. To guarantee consistency in the collected data, it is crucial to synchronize these clocks with high precision. In addition, as the small micro-processors embedded in each sensor node are usually resource-limited (Gungor et al., 2010), energy-efficient communication protocols for clock synchronization are therefore desired.

Quite a lot approaches have been proposed to solve this problem, ranging from centralized to distributed, time-triggered to event-triggered, see (Carli & Zampieri, 2014; Chen, Li, Huang, & Tang, 2015; Choi & Shen, 2010; Garcia, Mou, Cao, & Casbeer, 2017; Kadowaki & Ishii, 2015; Maróti, Kusy, Simon, & Lédeczi, 2004; Simeone & Spagnolini, 2007; Solis, Borkar, & Kumar, 2006) and references therein. To solve this fundamental problem, we propose to apply our event-triggered algorithms, i.e., **Algorithm 2** and the **Combined Algorithm** in this practical case. One of the most related works is done by Chen et al. (2015), where an event-triggered algorithm with state-dependent triggers is proposed. However, the virtual clocks they synchronize are formed in a discrete manner, which may encounter abrupt changes.

The ability of avoiding abrupt changes is essential in clock synchronization since time discontinuity due to these changes can cause serious faults such as missing important events (Sundararaman, Buy, & Kshemkalyani, 2005). While another event-triggered algorithm proposed by Garcia et al. (2017) does synchronize continuous-time virtual clocks, however, their time-dependent trigger design requires global information. Motivated by these two works, we introduce our state-dependent event-triggered algorithms that synchronize continuous-time virtual clocks.

7.2. Clock synchronization problem formulation

Consider an N-sensor WSN whose topology is described by a strongly-connected weight-balanced underlying digraph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, W\}$, with $\mathcal{V}$, $\mathcal{E}$, W defined as in Section 2. Without loss of generality, we say that a sensor i is able to receive information from its neighbors in $\mathcal{N}_i^{out}$ and send information to neighbors in $\mathcal{N}_i^{in}$. Each sensor in the network is equipped with a microprocessor with an underlying local clock $l_i(t)$, which is a function of the absolute time $t \in \mathbb{R}_{\geq 0}$. Ideally, the local clocks should be configured as $l_i(t) = t$ so that the notion of time is consistent throughout the system. In reality (Kadowaki & Ishii, 2015), however, they are in the form of

$$l_i(t) = \gamma_i t + o_i,\ i = 1, \ldots, N, \tag{31}$$

where the unknown constants $\gamma_i \in \mathbb{R}_{>0}$ and $o_i \in \mathbb{R}$ represent the clock drift and offset of i-th clock, respectively.

As the absolute time t is not available, the clock drift γ_i and offset o_i can not be computed directly. To synchronize the system, here we mean to synchronize the virtual clocks $T_i(t)$ of all sensors defined by (Kadowaki & Ishii, 2015)

$$T_i(t) = \alpha_i(l_i(t))l_i(t),\ i = 1, \ldots, N, \tag{32}$$

where $\alpha_i(l_i(t))$ is the controlled drift and is a function of node i's local time $l_i(t)$.

The clock synchronization is said to be achieved if

$$\lim_{t\to\infty} |T_i(t) - T_j(t)| = 0,\ \forall\, i, j \in \{1, \ldots, N\}. \tag{33}$$

For simple implementation, in this paper we consider the particular case where only clock drift is present, i.e., the clock offset $o_i = 0$ for $i = 1, \ldots, N$. We also assume $\gamma_i \in [1 - \epsilon_\gamma, 1 + \epsilon_\gamma]$, where ϵ_γ is known. The local clocks are then given by

$$l_i(t) = \gamma_i t,\ i = 1, \ldots, N. \tag{34}$$

Substitute (34) into (32) gives the expressions of virtual clocks

$$T_i(t) = \gamma_i \alpha_i(l_i(t))t,\ i = 1, \ldots, N. \tag{35}$$

Note that the virtual clocks are continuous by definition, therefore the abrupt changes on the clocks are avoided.

The dynamics of $\alpha_i(l_i(t))$ is specified by

$$\frac{\mathrm{d}\alpha_i(l_i(t))}{\mathrm{d}l_i(t)} = -\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{\alpha}_i(l_i(t)) - \tfrac{\gamma_j}{\gamma_i}\hat{\alpha}_j(l_j(t))), \tag{36}$$

where $\hat{\alpha}_i(l_i(t))$, $\hat{\alpha}_j(l_j(t))$ represent the last broadcast state values of sensor i and j at their local time l_i and l_j, respectively. Though γ_i and γ_j can not be computed directly, the value of $\frac{\gamma_j}{\gamma_i}$ can be obtained as follows (Garcia et al., 2017): record the local time of node i and node j when node i receives information from node j at two time points, say t_m and t_n, then $\frac{a_j}{a_i}$ can be computed using $\frac{\gamma_j}{\gamma_i} = \frac{l_j(t_m)-l_j(t_n)}{l_i(t_m)-l_i(t_n)}$. Note we only need the local clock time, not the exact values of t_m and t_n.

Define $e_i(l_i(t)) = \hat{\alpha}_i(l_i(t)) - \alpha_i(l_i(t))$ as sensor i's state error, where $\alpha_i(l_i(t))$ is its current controlled drift. An event for sensor i is triggered as soon as the triggering function

$$f_i(l_i(t), \alpha_i(l_i(t)), \hat{\alpha}_i(l_i(t)), \hat{\alpha}_j(l_j(t))) > 0 \tag{37}$$

is fulfilled. The triggered event then drives sensor i to broadcast its current state $\alpha_i(l_i(t))$ to its neighbors so that they can update their states accordingly. Our objective is to apply **Algorithm 2** and the **Combined Algorithm** so as to design triggering functions (37) for each sensor with its locally available information so that the virtual clocks are synchronized, i.e., (33) is satisfied.

7.3. Distributed event-triggered clock synchronization algorithms

The event-triggered algorithms for clock synchronization are developed based on Lyapunov functions. To begin, let us first rewrite (35) as

$$T_i(t) = \gamma_i \alpha_i(l_i(t))t = y_i(t)t, \tag{38}$$

where $y_i(t) = \gamma_i \alpha_i(l_i(t))$ is called the modified drift. It is clear that once consensus is achieved on the variables $y_i(t)$, the clock synchronization will be realized regardless of the individual values of γ_i and $\alpha_i(l_i(t))$.

We then adopt the Lyapunov candidate functions proposed in Section 4, with the modified drifts as variables, i.e., $V_1(y(t)) = \frac{1}{2}(y(t) - \bar{y})^T(y(t) - \bar{y})$, $V_2(y(t)) = \frac{1}{2}y(t)^T L^T y(t)$, and $V_\lambda(y(t)) = \lambda V_1(y(t)) + (1-\lambda)V_2(y(t))$. As the algorithm development with different Lyapunov functions are similar, we only use $V_2(y(t)) = \frac{1}{2}y(t)^T L^T y(t)$ as an example to illustrate the derivation process.

The dynamics of the modified drift $y_i(t)$ is derived as follows:

$$\begin{aligned} \dot{y}_i(t) = \frac{\mathrm{d}\alpha_i(l_i(t))}{\mathrm{d}l_i(t)} \cdot \frac{\mathrm{d}l_i(t)}{\mathrm{d}t} &= \gamma_i^2 \Big(- \sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\hat{\alpha}_i(l_i(t)) - \frac{\gamma_j}{\gamma_i}\hat{\alpha}_j(l_j(t)))\Big) \\ &= -\gamma_i \Big(\sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\gamma_i \hat{\alpha}_i(l_i(t)) - \gamma_j \hat{\alpha}_j(l_j(t))\Big) \\ &= -\gamma_i \sum_{j \in \mathcal{N}_i^{out}} w_{ij}\Big(\hat{y}_i(t) - \hat{y}_j(t)\Big). \end{aligned} \tag{39}$$

We then specify the following Lemma to upper bound the derivatives of $V_2(y(t))$.

Lemma 7.1. *In clock synchronization, for $i \in \mathcal{V}$, let $b_i, c_j > 0$ for all $i, j \in \mathcal{V}$ (the same b_i, c_j as in Lemma 4.1, define $\nu_i(t) = \sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\hat{y}_i(t) - \hat{y}_j(t))$, and $e_{yi}(t) =$*

$\hat{y}_i(t) - y_i(t)$, *then the derivative of* $V_2(y(t)) = \frac{1}{2}y(t)^T L^T y(t)$ *is upper bounded by*

$$\dot{V}_2(y(t)) \leq -\sum_{i=1}^{N} \gamma_i \left[\delta_i \Big(\sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\hat{y}_i(t) - \hat{y}_j(t)) \Big)^2 - \Big(\frac{d_i^{out}}{2b_i} + \frac{d_i^{out}}{2c_i} \Big) e_{yi}^2(t) \right], \quad (40)$$

where δ_i *is what defined in* (14).

The proof is similar to the proof for Lemma 4.1 and is omitted due to space limit.

From Lemma 7.1, we can see that as long as $\dot{V}_2(y(t)) < 0$ and $\|Ly(t)\| \neq 0$ hold, $y_i(t)$ achieves consensus, meaning $\lim_{t\to\infty} |y_i(t) - y_j(t)| = 0$. Recall that $T_i(t) = y_i(t)t$, therefore, $\lim_{t\to\infty} |T_i(t) - T_j(t)| = 0$, proving that the synchronization on virtual clocks can be achieved.

A sufficient condition to ensure that $V_2(y(t))$ is monotonically decreasing is

$$\begin{aligned} e_{yi}^2(t) &\leq \frac{\delta_i}{(\frac{d_i^{out}}{2b_i} + \frac{d_i^{out}}{2c_i})} \Big(\sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\hat{y}_i(t) - \hat{y}_j(t)) \Big)^2 \\ &= \frac{2b_i c_i \delta_i}{(b_i + c_i) d_i^{out}} \Big(\sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\gamma_i \hat{\alpha}_i(l_i(t)) - \gamma_j \hat{\alpha}_j(l_i(t))) \Big)^2 \\ &= \frac{2\gamma_i^2 b_i c_i \delta_i}{(b_i + c_i) d_i^{out}} \Big(\sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\hat{\alpha}_i(l_i(t)) - \frac{\gamma_j}{\gamma_i} \hat{\alpha}_j(l_j(t))) \Big)^2. \end{aligned} \quad (41)$$

With $e_{yi}(t) = \gamma_i e_i(l_i(t))$, we define the triggering function developed from **Algorithm 2** as

$$f_i(e_i(l_i(t))) = e_i^2(l_i(t)) - \frac{2b_i c_i \delta_i}{(b_i + c_i) d_i^{out}} \Big(\sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\hat{\alpha}_i(l_i(t)) - \frac{\gamma_j}{\gamma_i} \hat{\alpha}_j(l_j(t))) \Big)^2. \quad (42)$$

To ensure no triggers are missed by sensor i, we define an event either by

$$f_i(e_i(l_i(t))) > 0 \quad or \quad (43)$$

$$f_i(e_i(l_i(t))) = 0 \quad and \quad \sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\hat{\alpha}_i(l_i(t)) - \frac{\gamma_j}{\gamma_i} \hat{\alpha}_j(l_j(t))) \neq 0. \quad (44)$$

Similarly, an additional trigger is prescribed to address the non-Zeno behavior. Let l_i^{last} be the last time at which sensor i broadcasts its information to its neighbors. If at some time $l_i(t) \geq l_i^{last}$, sensor i receives information from a neighbor $j \in \mathcal{N}_i^{out}$, then it immediately broadcasts its state if

$$l_i(t) \in (l_i^{last}, l_i^{last} + \varepsilon_i'), \quad (45)$$

where

$$\varepsilon_i' < \sqrt{\frac{2\sigma_i b_i c_i \delta_i}{(b_i + c_i) d_i^{out}}} \quad (46)$$

whose design is as given in Proposition 5.1.

The following result presents **Algorithm 2** in the clock synchronization application.

Theorem 7.2. *For an N-sensor network over a weight-balanced digraph, assume only clock drift exists, i.e., $o_i = 0,\ \forall i \in \mathcal{V}$. With the virtual clocks* (32)*, dynamics given in* (36)*, the distributed event-triggered consensus algorithm* (42)-(46) *(**Algorithm 2**) achieves asymptotic synchronization for the virtual clocks, i.e.,* (33) *is satisfied.*

We haven shown that **Algorithm 2** can be applied to the practical clock synchronization problem. Next, we show that the **Combined Algorithm** can also be applied to solve the clock synchronization problem. To do so, we first derive the triggering law for the clock synchronization problem from **Algorithm 1** as

$$f_i(e_i(l_i(t))) = e_i^2(l_i(t)) - \frac{\sigma_i}{4d_i^{out}} \sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{\alpha}_i(l_i(t)) - \frac{\gamma_j}{\gamma_i}\hat{\alpha}_j(l_j(t)))^2, \tag{47}$$

with an inter-event period bounded by $\varepsilon_i' < \sqrt{\frac{\sigma_i}{4d_i^{out}w_i^{\max}|\mathcal{N}_i^{out}|}}$.

Then, with the triggering rules (42) - (47) and the analysis in Section 6, designing the triggering function for the clock synchronization problem from the **Combined Algorithm** is straightforward. That is,

$$\begin{aligned} f_i(e_i(l_i(t))) = e_i^2(l_i(t)) &- \frac{\lambda\sigma_i}{4d_i^{out}} \sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{\alpha}_i(l_i(t)) - \frac{\gamma_j}{\gamma_i}\hat{\alpha}_j(l_j(t)))^2 \\ &- \frac{2(1-\lambda)\sigma_i\delta_i b_i c_i}{(b_i+c_i)d_i^{out}}\Big(\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{\alpha}_i(l_i) - \frac{\gamma_j}{\gamma_i}\hat{\alpha}_j(l_j))\Big)^2, \end{aligned} \tag{48}$$

with an inter-event period bounded by $\varepsilon_i' < \sqrt{\frac{\lambda\sigma_i}{4d_i^{out}w_i^{\max}|\mathcal{N}_i^{out}|} + \frac{2(1-\lambda)\sigma_i\delta_i b_i c_i}{(b_i+c_i)d_i^{out}}}$.

Theorem 7.3. *For an N-sensor network over a weight-balanced digraph, assume only clock drift exists, i.e., $o_i = 0,\ \forall i \in \mathcal{V}$. With the virtual clocks* (32)*, dynamics given in* (36)*, the distributed event trigging rule defined in* (48)*, then the **Combined Algorithm** achieves asymptotic synchronization for the virtual clocks when the triggering condition $f_i(e_i(l_i(t))) > 0$ or $f_i(e_i(l_i(t))) = 0$ with $e_i^2(l_i(t)) > 0$ is met.*

The proof of the theorem and the stability analysis, non-Zeno behavior exclusion are as given in Section 5, therefore are omitted.

8. Simulation Results

In this section, we apply **Algorithm 1** and **Algorithm 2** to the event-triggered clock synchronization problem, to show the effectiveness of both algorithms. We then demonstrate the performance of the proposed algorithms through several simulations and show how either **Algorithm 1** or **Algorithm 2** could be argued to be 'better' given different network topology, which has set the basis for our introduction of the **Combined Algorithm** to easily go between the two.

We first show that both **Algorithm 1** and **Algorithm 2** are able to synchronize the virtual clocks in WSNs. We consider four different network topologies, with their corresponding weighted adjacency matrices listed in Table 3.

The clock offset is 0 for all nodes, and the unknown clock drifts are $\gamma = [0.65\ 0.79\ 0.91\ 1.25\ 1.4]^T$. The evolution of the local clocks with respect to the absolute

Table 3. Four different networks.

Network 1: Random network

$$W_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 5/6 & 0 & 1/6 & 0 & 0 \\ 1/6 & 0 & 1/6 & 1/2 & 1/6 \\ 0 & 0 & 1/6 & 0 & 5/6 \end{bmatrix}$$

Network 2: Ring network

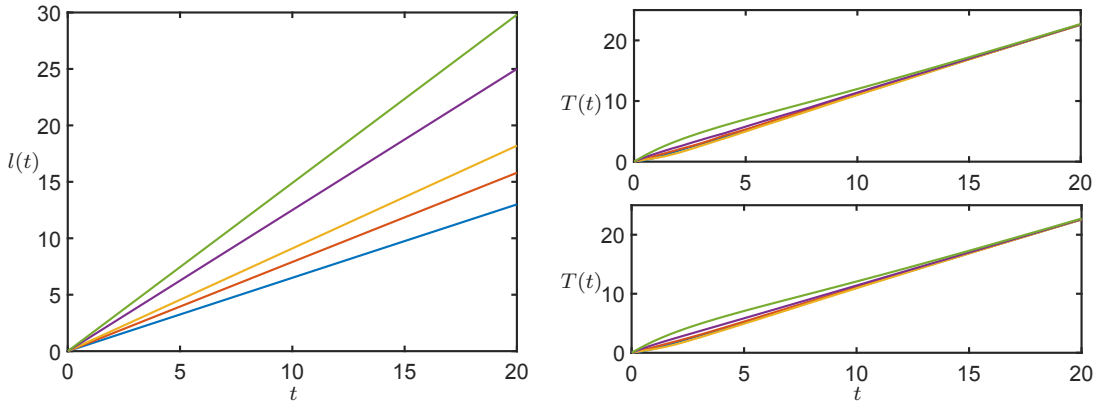
$$W_2 = \begin{bmatrix} 0 & 1/2 & 0 & 0 & 1/2 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 0 & 1/2 \\ 1/2 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

Network 3: Complete network

$$W_3 = \begin{bmatrix} 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 0 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 0 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 0 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 \end{bmatrix}$$

Network 4: Star network

$$W_4 = \begin{bmatrix} 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 0 & 0 & 0 \end{bmatrix}$$



(a) Local clocks (b) Virtual clocks

Figure 1. Plots of the simulation results of the clock synchronization on Network 1. (a) The local clocks are the same for both algorithms. (b) Virtual clocks with the implementation of event-triggered control. Both **Algorithm 1** (top) and **Algorithm 2** (bottom) are able to synchronize the virtual clocks.

time t is shown in Figure 1a. We can see that without any control, the local clocks will diverge.

Then, we implement **Algorithm 1** and **Algorithm 2** with the control law (36), triggering functions (47) and (42) developed from **Algorithm 1** and **Algorithm 2**, respectively, to achieve clock synchronization. The involved parameters are set to be $\sigma_i = 0.5$, and $b_i = c_i = 0.5/d_i^{out}$ for all $i \in \{1, \ldots, N\}$. Both algorithms are able to synchronize the virtual clocks on all four networks and we take the result on **Network 1** as an example and show the virtual clock evolution in Figure 1b. However, except for the synchronization, we have no idea which algorithm performs better on other evaluation metrics, for example, the convergence speed and total energy consumption. Also, the performance evaluation result may differ for different network topologies. Therefore, in the following simulations, we show the difference of the two different algorithms on four network topologies with different evaluation metrics.

We plot the triggering instances of all nodes in the network when implementing the event-triggered algorithms in Figure 2a, 2c, 2e, 2g. We notice that in general, the number of events triggered when implementing **Algorithm 1** is less than that when implementing **Algorithm 2**. We also plot the evolution of Lyapunov functions, i.e., $V_1(y(t)) = \frac{1}{2}(y(t)-\bar{y})^T(y(t)-\bar{y})$, $V_2(y(t)) = \frac{1}{2}y(t)^T L^T y(t)$, and $V_\lambda(y(t)) = \lambda V_1(y(t)) +$

$(1-\lambda)V_2(y(t))$ for all networks with $\sigma_i = 0.5$ for all agents in Figure 2b, 2d, 2f, 2h, which again corroborates our analysis that both algorithms ensure convergence, or in this case, synchronization for the resulting systems. We can see that except for **Network 3**, the Lyapunov function of **Algorithm 2** in the other three networks converges faster than that of **Algorithm 1**. It is also noted that when the number of events triggered when implementing **Algorithm 2** is noticeably larger than that in implementing **Algorithm 1**, the convergence speed of the Lyapunov function in **Algorithm 2** is also noticeably faster than that in **Algorithm 1**. This is reasonable, since the more events are triggered, the more information is communicated in the network, and the faster the consensus will be reached.

The above simulations corroborate our argument that depending on the chosen evaluation metric, either algorithm can be argued to be 'better' than the other. To better quantize/visualize the performance difference of the two algorithms and demonstrate our motivations for proposing the **Combined Algorithm**, we then executing all algorithms with respect to varying σ. We evaluate these algorithms with four performance metrics, 1) the total number of events triggered, denoted by N_e, 2) the time needed for each network to reach a 99% convergence of the Lyapunov function, denoted by T_{con}, 3) the total communication energy required to achieve a 99% convergence, denoted by E, and 4) the square of the H_2-norm of the system, denoted by $\mathcal{C}$ (Dezfulian, Ghaedsharaf, & Motee, 2018). The total communication energy needed is calculated by multiplying the power in units of milliwatt (mW) with T_{con}, where we adopt the following power calculation model in units of mW (Martins et al., 2008):

$$\mathcal{P} = \sum_{i=1}^{N} \left[\sum_{j\in\{1,\dots,N\}, j\neq i} \eta 10^{0.1 P_{i\to j} + \zeta \|\alpha_i(l_i(t)) - \alpha_j(l_j(t))\|} \right],$$

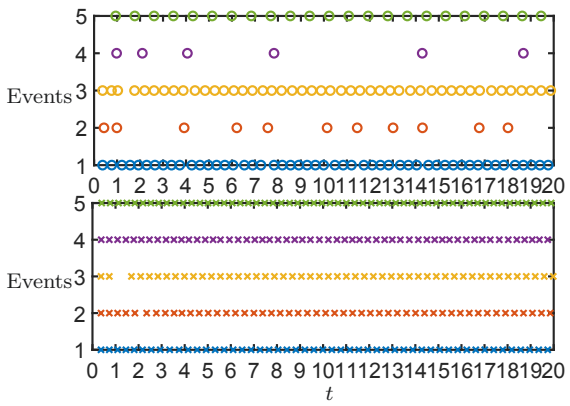
where $\zeta > 0$ and $\eta > 0$ depend on the characteristics of the wireless medium and $P_{i\to j}$ is the power of the signal transmitted from agent i to agent j in units of dBmW. Similar as (Nowzari & Cortés, 2012), we set η, ζ and $P_{i\to j}$ to be 1. The square of the H_2-norm, $\mathcal{C}$ is defined by

$$\mathcal{C} := \int_{t=0}^{\infty} \sum_{i=1}^{N} (y_i(t) - \bar{y})^2 dt,$$

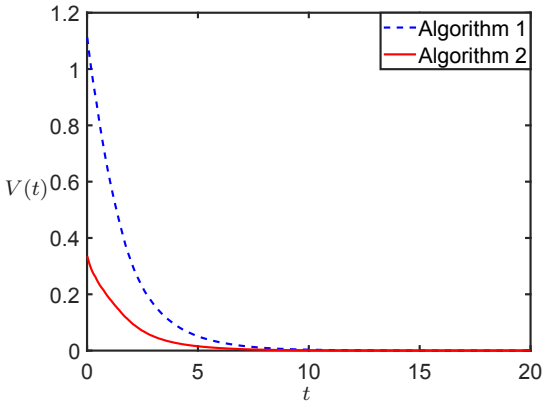
where $y_i(t)$ is the modified drift of each local clock and $\bar{y}$ is the average of the modified drift of the system.

The involved parameters are set to be $b_i = c_i = 0.5/d_i^{out}$ for all $i \in \{1, \dots, N\}$. The same control law (36) is applied. For **Algorithm 1** and **Algorithm 2** that achieve clock synchronization, their triggering functions are given by (47) and (42), respectively. For the **Combined Algorithm**, its triggering function is given by (48), with $\lambda = 0.5$. For each σ, we run 10 simulations with random clock drift that satisfies to $\gamma_i \in (0.7, 1.3)$ and obtain the average of N_e, T_{con}, E, and C in each simulation.

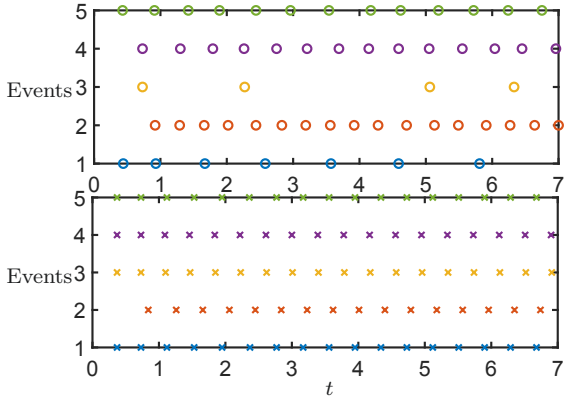
From the top figures in Figure 3a, 3c, 3e, and 3g, we can see that for different σ, the total number of events triggered in the system when executing **Algorithm 2** is larger than that when executing **Algorithm 1**. On the other hand, from the bottom figures in Figure 3a, 3c, 3e, and 3g, we can see that the time needed to reach a 99% convergence of the system is usually much less when executing **Algorithm 2**, with the only exception for the complete network, where both algorithms have similar



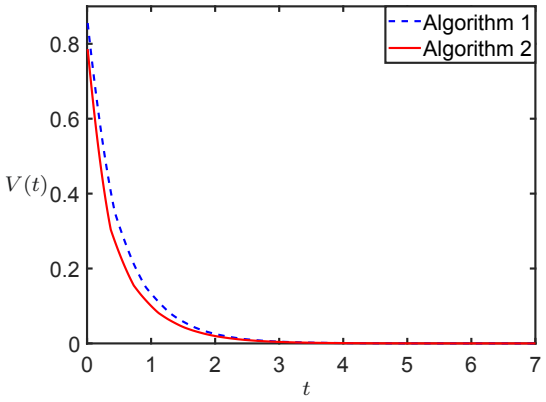
(a) Network 1, triggering instances



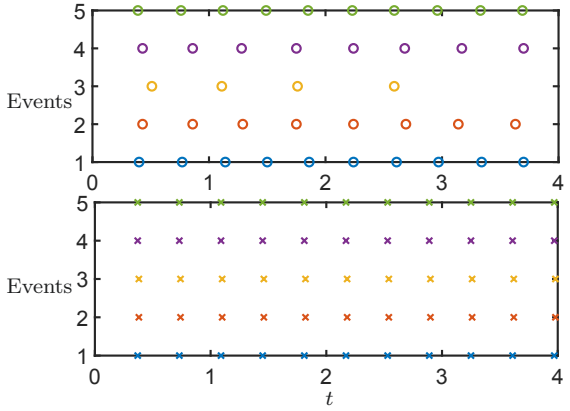
(b) Network 1, Lyapunov function



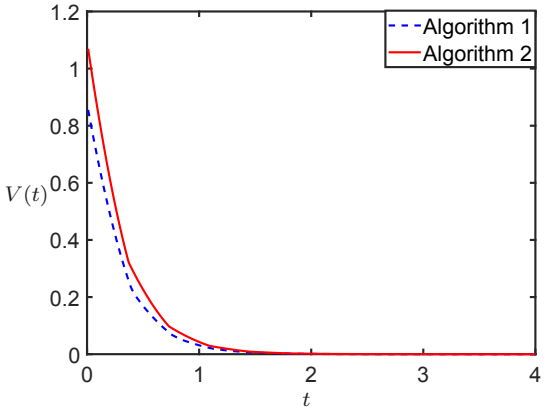
(c) Network 2, triggering instances



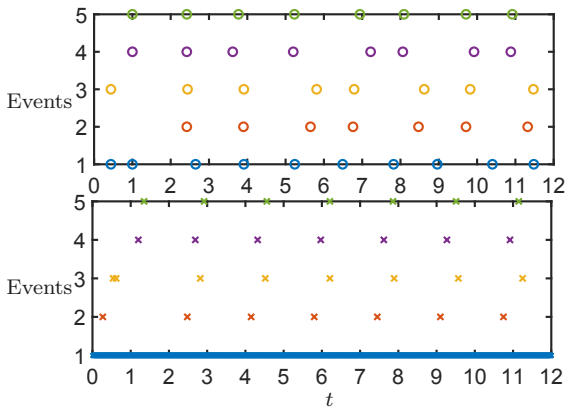
(d) Network 2, Lyapunov function



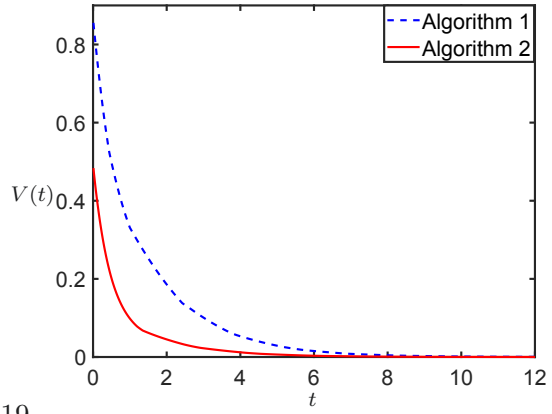
(e) Network 3, triggering instances



(f) Network 3, Lyapunov function

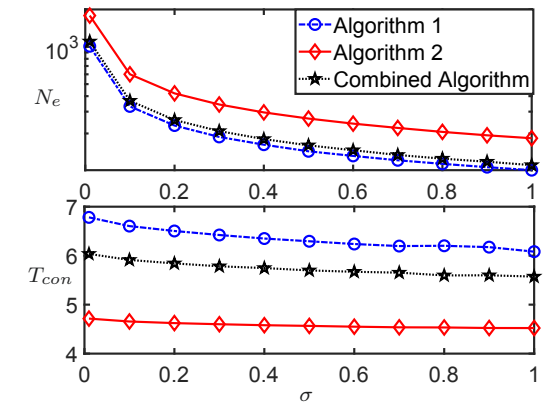


(g) Network 4, triggering instances

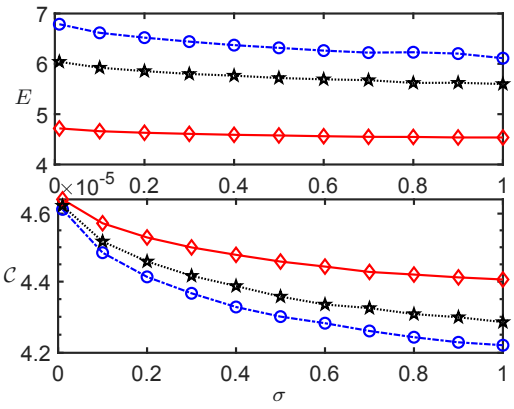


(h) Network 4, Lyapunov function

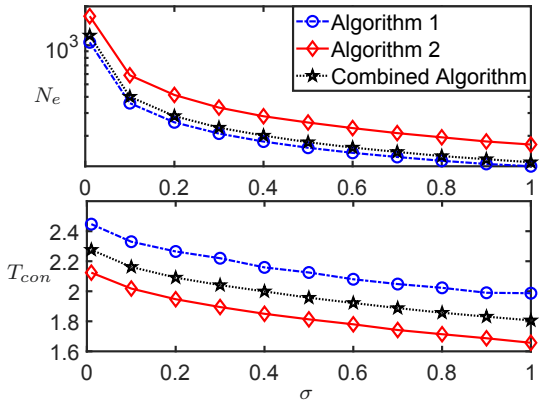
Figure 2. Plots of the triggering instances and the evolution of Lyapunov candidate functions on four networks when implementing both Algorithms. For figure (a), (c), (e), (g), **Algorithm 1** is on the top and **Algorithm 2** is on the bottom.



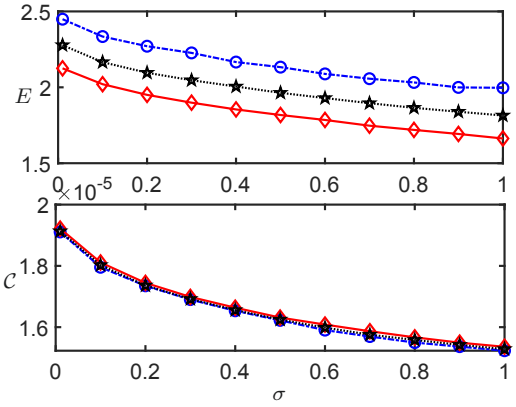
(a) Network 1



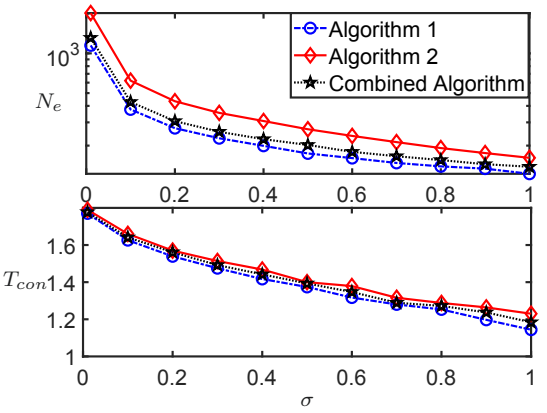
(b) Network 1



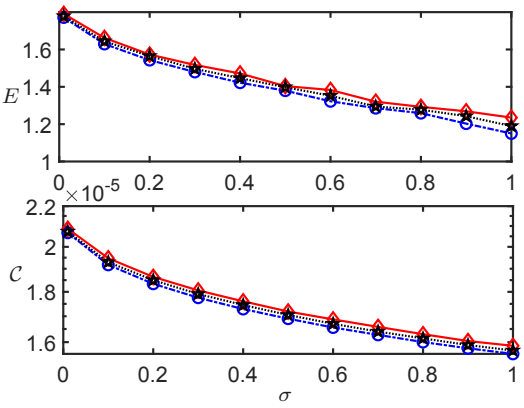
(c) Network 2



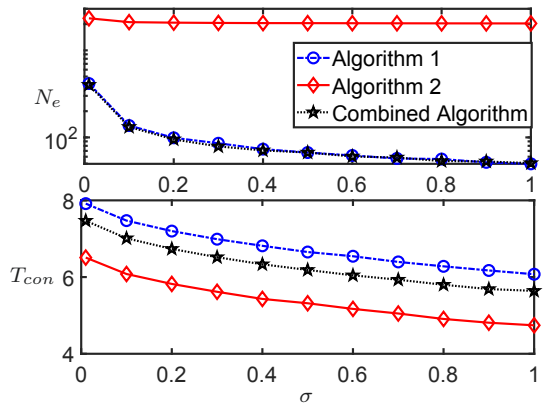
(d) Network 2



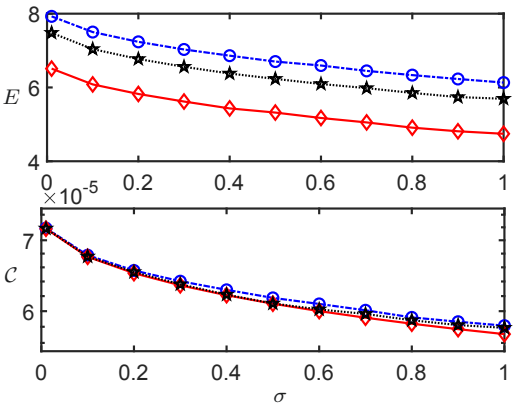
(e) Network 3



(f) Network 3



(g) Network 4



(h) Network 4

Figure 3. Plots of different evaluation metrics. For figure (a), (c), (e), (g), top: total events triggered, bottom: convergence time (bottom); for figure (b), (d), (f), (h), top: energy consumption, bottom: H_2-norm squared.

convergence speed. As the total communication energy consumption is related with both the total number of events triggered and the time required to reach convergence, we can see from the top figures in Figure 3b, 3d, 3f, and 3h that either algorithm can outperform the other in terms of the total energy consumption for different network topologies. The H_2-norm squared evaluates the distance of each local modified drift with the average modified drift of the system, whose value therefore also indicates the convergence speed of the system to some extent, see the bottom figures in Figure 3b, 3d, 3f, and 3h. Therefore, depending on different network topologies and depending on what performance metrics are most important for the application at hand, it may be desirable to implement different types of event-triggered algorithms. Note that the **Combined Algorithm** can easily be tuned to approach either **Algorithm 1** or **Algorithm 2** or anything in between to meet varying system needs by setting values for λ. This also motivates our future work of adapting λ online to further improve performance.

9. Conclusion

This paper proposes a class of distributed event-triggered communication and control law for multi-agent systems whose underlying directed graphs are weight-balanced. The class of algorithms are developed from a class of Lyapunov functions, each of which is a linear combination (parameterized by $\lambda \in [0, 1]$) of two Lyapunov functions. Each λ defines a new Lyapunov function coupled with a new event-triggered coordination algorithm which uses that particular function to guarantee correctness and is able to exclude the possibility of Zeno behavior. We show that the proposed entire class of event-triggered algorithms can be tuned to meet varying performance needs by adjusting λ. We also apply the proposed distributed event-triggered algorithms to solve the practical clock synchronization problem in WSNs. For the future research, we will focus on developing a unified evaluation metric (which is a function of different performance needs) that can be used to evaluate the performance of different algorithms. In that way, the class of distributed algorithms will be developed from a tunable algorithm to an adaptive algorithm.

Funding

This work was supported by the NSF under Grant #204294.

References

Abbasi, A. A., & Younis, M. (2007). A survey on clustering algorithms for wireless sensor networks. *Computer communications*, *30*(14-15), 2826–2841.

Antunes, D., & Heemels, W. (2014). Rollout event-triggered control: Beyond periodic control performance. *IEEE Transactions on Automatic Control*, *59*(12), 3296–3311.

Borgers, D., Geiselhart, R., & Heemels, W. (2017). Tradeoffs between quality-of-control and quality-of-service in large-scale nonlinear networked control systems. *Nonlinear Analysis: Hybrid Systems*, *23*, 142–165.

Carli, R., & Zampieri, S. (2014). Network clock synchronization based on the second-order linear consensus algorithm. *IEEE Transactions on Automatic Control*, *59*(2), 409–422.

Chen, Z., Li, D., Huang, Y., & Tang, C. (2015). Event-triggered communication for distributed time synchronization in wsns. In *Control conference (ccc), 2015 34th chinese* (pp. 7789–7794).

Choi, B. J., & Shen, X. (2010). Distributed clock synchronization in delay tolerant networks. In *Communications (icc), 2010 ieee international conference on* (pp. 1–6).

Dezfulian, S., Ghaedsharaf, Y., & Motee, N. (2018). On performance of time-delay linear consensus networks with directed interconnection topologies. In *2018 annual american control conference (acc)* (pp. 4177–4182).

Dimarogonas, D. V., Frazzoli, E., & Johansson, K. H. (2012). Distributed event-triggered control for multi-agent systems. *IEEE Transactions on Automatic Control*, *57*(5), 1291–1297.

Dimarogonas, D. V., & Johansson, K. H. (2009). Event-triggered control for multi-agent systems. In *Decision and control, 2009 held jointly with the 2009 28th chinese control conference. cdc/ccc 2009. proceedings of the 48th ieee conference on* (pp. 7131–7136).

Ding, L., Han, Q.-L., Ge, X., & Zhang, X.-M. (2017). An overview of recent advances in event-triggered consensus of multiagent systems. *IEEE transactions on cybernetics*, *48*(4), 1110–1123.

Garcia, E., Mou, S., Cao, Y., & Casbeer, D. W. (2017). An event-triggered consensus approach for distributed clock synchronization. In *American control conference (acc), 2017* (pp. 279–284).

Girard, A. (2015). Dynamic triggering mechanisms for event-triggered control. *IEEE Transactions on Automatic Control*, *60*(7), 1992–1997.

Gungor, V. C., Lu, B., & Hancke, G. P. (2010). Opportunities and challenges of wireless sensor networks in smart grid. *IEEE transactions on industrial electronics*, *57*(10), 3557–3564.

Hardy, G. H., Littlewood, J. E., & Pólya, G. (1952). *Inequalities*. Cambridge university press.

Heijmans, S. H., Borgers, D. P., & Heemels, W. (2017). Stability and performance analysis of spatially invariant systems with networked communication. *IEEE Transactions on Automatic Control*, *62*(10), 4994–5009.

Johansson, K. H., Egerstedt, M., Lygeros, J., & Sastry, S. (1999). On the regularization of zeno hybrid automata. *Systems & Control Letters*, *38*(3), 141–150.

Kadowaki, Y., & Ishii, H. (2015). Event-based distributed clock synchronization for wireless sensor networks. *IEEE Transactions on Automatic Control*, *60*(8), 2266–2271.

Khashooei, B. A., Antunes, D., & Heemels, W. (2017). Output-based event-triggered control with performance guarantees. *IEEE Transactions on Automatic Control*.

Liang, J., Wang, Z., Shen, B., & Liu, X. (2012). Distributed state estimation in sensor networks with randomly occurring nonlinearities subject to time delays. *ACM Transactions on Sensor Networks (TOSN)*, *9*(1), 4.

Liu, J., Zhang, Y., Yu, Y., & Sun, C. (2018). Fixed-time event-triggered consensus for nonlinear multiagent systems without continuous communications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Maróti, M., Kusy, B., Simon, G., & Lédeczi, Á. (2004). The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on embedded networked sensor systems* (pp. 39–49).

Martins, N. C., et al. (2008). *Jointly optimal placement and power allocation of wireless networks* (Unpublished doctoral dissertation).

Nowzari, C., & Cortés, J. (2012). Self-triggered coordination of robotic networks for optimal deployment. *Automatica*, *48*(6), 1077–1087.

Nowzari, C., & Cortés, J. (2014). Zeno-free, distributed event-triggered communication and control for multi-agent average consensus. In *American control conference (acc), 2014* (pp. 2148–2153).

Nowzari, C., & Cortés, J. (2016). Distributed event-triggered coordination for average consensus on weight-balanced digraphs. *Automatica*, *68*, 237–244.

Nowzari, C., Garcia, E., & Cortés, J. (2019). Event-triggered communication and control of networked systems for multi-agent consensus. *Automatica*, *105*, 1–27.

Olfati-Saber, R., & Jalalkamali, P. (2012). Coupled distributed estimation and control for mobile sensor networks. *IEEE Transactions on Automatic Control*, *57*(10), 2609–2614.

Olfati-Saber, R., & Murray, R. M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on automatic control*, *49*(9), 1520–1533.

Peng, Z., Wen, G., Rahmani, A., & Yu, Y. (2015). Distributed consensus-based formation control for multiple nonholonomic mobile robots with a specified reference trajectory. *International Journal of Systems Science*, *46*(8), 1447–1457.

Ramesh, C., Sandberg, H., & Johansson, K. H. (2016). Performance analysis of a network of event-based systems. *IEEE Transactions on Automatic Control*, *61*(11), 3568–3573.

Seyboth, G. S., Dimarogonas, D. V., & Johansson, K. H. (2013). Event-based broadcasting for multi-agent average consensus. *Automatica*, *49*(1), 245–252.

Simeone, O., & Spagnolini, U. (2007). Distributed time synchronization in wireless sensor networks with coupled discrete-time oscillators. *EURASIP Journal on Wireless Communications and Networking*, *2007*(1), 057054.

Solis, R., Borkar, V. S., & Kumar, P. (2006). A new distributed time synchronization protocol for multihop wireless networks. In *Decision and control, 2006 45th ieee conference on* (pp. 2734–2739).

Sun, Z., Huang, N., Anderson, B. D., & Duan, Z. (2016). A new distributed zeno-free event-triggered algorithm for multi-agent consensus. In *Decision and control (cdc), 2016 ieee 55th conference on* (pp. 3444–3449).

Sundararaman, B., Buy, U., & Kshemkalyani, A. D. (2005). Clock synchronization for wireless sensor networks: a survey. *Ad hoc networks*, *3*(3), 281–323.

Wu, Y.-C., Chaudhari, Q., & Serpedin, E. (2011). Clock synchronization of wireless sensor networks. *IEEE Signal Processing Magazine*, *28*(1), 124–138.

Xie, D., Xu, S., Li, Z., & Zou, Y. (2015). Event-triggered consensus control for second-order multi-agent systems. *IET Control Theory & Applications*, *9*(5), 667–680.

Yi, X., Liu, K., Dimarogonas, D. V., & Johansson, K. H. (2017). Distributed dynamic event-triggered control for multi-agent systems. In *Decision and control (cdc), 2017 ieee 56th annual conference on* (pp. 6683–6698).

Yi, X., Lu, W., & Chen, T. (2016). Distributed event-triggered consensus for multi-agent systems with directed topologies. In *2016 chinese control and decision conference (ccdc)* (pp. 807–813).

Appendix A. Proof of Lemma 4.1

Proof. Omit the time stamp t for simplicity. The derivative of $V_2(x)$ takes the form

$$\dot{V}_2(x) = x^T L^T \dot{x}. \tag{A1}$$

Substitute the vector form $x = \hat{x} - e$ into (A1), and expand it with (3), we have

$$\begin{aligned}\dot{V}_2(x) &= \hat{x}^T L^T \dot{x} - e^T L^T \dot{x} \\ &= \sum_{i=1}^{N}\Big(\sum_{j\in\mathcal{N}_i^{out}} w_{ij}(\hat{x}_i - \hat{x}_j)u_i - \sum_{j\in\mathcal{N}_i^{out}} w_{ij}(e_i - e_j)u_i\Big) \\ &= \sum_{i=1}^{N}\Big(-u_i^2 - \sum_{j\in\mathcal{N}_i^{out}} w_{ij}e_i u_i + \sum_{j\in\mathcal{N}_i^{out}} w_{ij}e_j u_i\Big) \\ &= \sum_{i=1}^{N}\Big(-u_i^2 - d_i^{out}e_i u_i + \sum_{j\in\mathcal{N}_i^{out}} w_{ij}e_j u_i\Big).\end{aligned} \tag{A2}$$

For $b_i,\ c_j > 0$, applyYoung's inequality (2) to the cross terms at the right hand side of (A2) gives

$$\begin{aligned}-d_i^{out}e_i u_i &\le \frac{d_i^{out}}{2b_i}e_i^2 + \frac{d_i^{out}b_i}{2}u_i^2, \\ \sum_{j\in\mathcal{N}_i^{out}} w_{ij}e_j u_i &\le \sum_{j\in\mathcal{N}_i^{out}} \frac{w_{ij}}{2c_j}e_j^2 + \sum_{j\in\mathcal{N}_i^{out}} \frac{w_{ij}c_j}{2}u_i^2.\end{aligned}$$

Since the digraph is weight-balanced, the following equality holds:

$$\sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i^{out}} \frac{w_{ij}}{2c_j}e_j^2 = \sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i^{in}} \frac{w_{ji}}{2c_i}e_i^2 = \sum_{i=1}^{N}\frac{d_i^{in}}{2c_i}e_i^2 = \sum_{i=1}^{N}\frac{d_i^{out}}{2c_i}e_i^2.$$

Combine the above inequalities and equality, we obtain an upper bound for $\dot{V}_2(x)$:

$$\begin{aligned}\dot{V}_2(x) &\le \sum_{i=1}^{N}\Big(-u_i^2 + \frac{d_i^{out}e_i^2}{2b_i} + \frac{d_i^{out}b_i u_i^2}{2} + \frac{d_i^{out}e_i^2}{2c_i} + \sum_{j\in\mathcal{N}_i^{out}}\frac{w_{ij}c_j}{2}u_i^2\Big) \\ &= -\sum_{i=1}^{N}\left[\Big(1 - \frac{d_i^{out}b_i}{2} - \sum_{j\in\mathcal{N}_i^{out}}\frac{w_{ij}c_j}{2}\Big)u_i^2 - \Big(\frac{d_i^{out}}{2b_i} + \frac{d_i^{out}}{2c_i}\Big)e_i^2\right] \\ &= -\sum_{i=1}^{N}\left[\delta_i u_i^2 - \Big(\frac{d_i^{out}}{2b_i} + \frac{d_i^{out}}{2c_i}\Big)e_i^2\right],\end{aligned} \tag{A3}$$

with δ_i defined in (14). To ensure $\delta_i > 0$, we require $b_i, c_j < \frac{1}{d_i^{out}}$. □