Event-Triggered Communication and Control of Networked Systems for Multi-Agent Consensus

Cameron Nowzari [a]   Eloy Garcia [b]   Jorge Cortés [c]

[a] *Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA, 22030, USA*

[b] *Control Science Center of Excellence, Air Force Research Laboratory, Wright-Patterson AFB, OH, 45433, USA*

[c] *Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, 92093, USA*

Abstract

This article provides an introduction to event-triggered coordination for multi-agent average consensus. We provide a comprehensive account of the motivations behind the use of event-triggered strategies for consensus, the methods for algorithm synthesis, the technical challenges involved in establishing desirable properties of the resulting implementations, and their applications in distributed control. We pay special attention to the assumptions on the capabilities of the network agents and the resulting features of the algorithm execution, including the interconnection topology, the evaluation of triggers, and the role of imperfect information. The issues raised in our discussion transcend the specific consensus problem and are indeed characteristic of cooperative algorithms for networked systems that solve other coordination tasks. As our discussion progresses, we make these connections clear, highlighting general challenges and tools to address them widespread in the event-triggered control of networked systems.

Key words: networked systems, event-triggered control, distributed coordination, multi-agent consensus

1 Introduction

This article provides an introduction to the topic of event-triggered coordination of networked systems, with a particular emphasis on the multi-agent consensus problem. In many applications involving multiple agents such as vehicles, sensors, computers, etc., a group of agents must agree upon various physical and virtual quantities of interest, at which point it is said that a consensus has been achieved. Consensus is a long-standing area of research, particularly in computer science, and arguably forms the foundation for distributed computing in general (Olfati-Saber and Murray, 2004; Ren and Beard, 2008; Lynch, 1997; Ren et al., 2007). Consequently, there is a vast amount of literature available on consensus problems. Olfati-Saber et al. (2007) provided a brief history of the field and its success 10 years ago which, in a nutshell, has to do with the extremely wide applicability of such problems across many different disciplines. As a result, consensus problems in general are indeed still an extremely active area of research.

Email addresses: `cnowzari@gmu.edu` (Cameron Nowzari), `eloy.garcia.2@us.af.mil` (Eloy Garcia), `cortes@ucsd.edu` (Jorge Cortés).

On the other hand, the idea of event-triggered control has an interesting history that only recently seems to be gaining popularity throughout the controls community. The basic idea of event-triggered control is to abandon the paradigm of periodic (or continuous) sampling/control in exchange for deliberate, opportunistic aperiodic sampling/control to improve efficiency. For instance, it may not make sense to constantly monitor the state of an already-stable system just in case something goes wrong. Instead it is more efficient to only sporadically check on the system to make sure things are behaving well. The general topic of research on these types of problems is then to determine precisely when control signals should be updated to improve efficiency while still guaranteeing a desired quality of service. Many different researchers have explored these ideas over the past five decades under many different names including 'event-based sampling', 'adaptive sampling', 'event-triggered control', 'Lebesgue sampling', 'send-on-delta concept', or 'minimum attention control' (Åström and Bernhardsson, 1999, 2002; Årzén, 1999; Sandee et al., 2005; Brockett, 1997; Miskowicz, 2006; Heemels et al., 2008; Tabuada, 2007; Otanez et al., 2002; Mitchell and McDaniel, 1967; Seuret et al., 2014). While these ideas have been tossed around since the 1960's, it is only in the last 10 years that

the field has started maturing to soon stand alone in the area of systems and control (Heemels et al., 2012; Hetel et al., 2017), with a specific set of interesting challenges in the context of network systems.

Why Event-Triggering?

The idea of event-triggered sampling and control stems from an implementation issue; how can we control continuous-time systems using digital controllers? The current standard method is to simply have the digital controller take actions periodically; and while ideas of aperiodic sampling and control have been proposed long ago, some modern textbooks seem to suggest that periodic sampling and control is the only way to implement feedback control laws on digital systems (Åström and Wittenmark, 1997; Franklin et al., 2010; Heemels et al., 2012). However, real systems in general are not able to acquire samples at an exact operating frequency. Consequently, the stability of sampled-data systems with aperiodic sampling has been a longstanding research area within the systems and controls community. As a result, there are already many different standard methods and ways of thinking about such problems and analyzing stability. For example, aperiodic sampling can actually just be modeled as a specific time-delay system. The same system might be modeled as a hybrid system with impulsive dynamics. More specifically, a Linear Time Invariant (LTI) system with aperiodic sampled-data might be reformulated as a Linear Time Varying (LTV) or Linear Parameter Varying (LPV) system. Another option is to derive the input/output relationships to study the effect of aperiodic sampling on an output as is often done in robust control. In any case, the main question of interest in a nutshell is: how quickly does the system need to be sampled to guarantee stability? The recent survey (Hetel et al., 2017) presents various methods to address this question.

Aperiodic Sampling as an Opportunity

Whereas the above paradigm views aperiodic sampling as a type of disturbance with respect to the ideal case of exactly periodic sampling, new advances in event-triggering methods suggest treating aperiodic sampling as an opportunity rather than an inconvenience or disturbance. As mentioned above, these types of problems generally arise as we try to control continuous-time systems with digital controllers using an idealized controller that assumes exact state information and continuous feedback is always possible. The natural question to ask at this point is then exactly how fast does the controller need to sample the system and feed back the control input to ensure closed-loop stability? The answers to these questions then often come in terms of robustness guarantees to the tune of "as long as the sampling/control frequency is greater than some threshold, then the steady-state error is guaranteed to remain less than some quantity." In other words, as long as the feedback loop for a given system is 'fast enough', the system behaves similarly to the ideal system. In particular, this paradigm hinges largely on the intuitive fact that, as long as the sampling rate is sufficiently fast, the system behaves well. However, early studies into this question revealed that this is not always the case, e.g., it is actually possible to speed up the sampling rate and have a closed-loop feedback control system degrade in performance (Gupta, 1963; Bekey and Tomovic, 1966; Liff and Wolf, 1966; Tomovic and Bekey, 1966; Mitchell and McDaniel, 1967; Ciscato and Martiani, 1967). However, this was a non-issue in general due to the fact that while it is not guaranteed that speeding up the sampling rate improved performance at all times, it is true that continuing to increase the sampling speed will eventually yield better performance at some point. While this paradigm has been mostly sufficient for controlling many different autonomous systems in the past, it seems quite limiting in many application areas today. Instead, event-triggered ideas essentially recast the question of 'how fast should a control system respond' as 'exactly when should a control system respond?' to improve efficiency.

Why Now?

When considering a dedicated sensor and actuator that are not connected to any wireless network, it may be reasonable to ask the sensor to take samples as fast as possible at all times and have the actuator act accordingly. In this setting, it is probably not even worth developing a more efficient or intelligent sensor as the dedicated sensor periodically taking measurements is not affecting anything else. However, this may not be practical if we only have a remote sensor and the sensor data must be transmitted back to the actuator over a wireless channel, especially if this wireless channel must be shared among other devices. In this case the action of acquiring a sample is now literally a resource that must be managed efficiently. More generally, a strong motivation for the resurgence of these topics is likely due to the increasing popularity of networked cyber-physical systems across all disciplines. In particular, the inherently tight couplings required between physical processes (e.g., sampling, actuation, motion) and cyber processes (e.g., communication, computation, storage) in networked systems reveals the need for more efficient deployment of such systems by treating things like wireless communication or computation as resources rather than taking them for granted. This suggests the application of event-triggered ideas not only to determine when control signals should be updated, but to a wider array of capabilities including data acquisition, actuation, and communication. It is in this sense that, hereafter, we employ the term event-triggered "communication" to refer to a communication event and the term event-triggered "control" to refer to a controller update event. When both appear in conjunction, we refer to the combination of event-triggered communication and control as event-triggered "coordination".

In particular, we focus on the Internet of Things (IoT) and other large-scale networks as a strong motivator for why we should think about event-triggered coordination

schemes rather than periodic ones. IoT devices need to support a large variety of sensors and actuators that interact with the physical world, in addition to standard cyber capabilities such as processing, storage, or communication. However, as IoT devices aim to support services and applications that interact with the physical world, large numbers of these devices need to be deployed and work reliably with minimal human intervention (Perera et al., 2014). This requirement places a lot of crucial constraints on what we expect of our IoT devices (Kolios et al., 2016). First of all, these devices in general will be battery-operated and have small form factors, making energy efficiency a critical design consideration. Second, these devices will need to have a wide range of capabilities to integrate seamlessly within a larger IoT network, translating to high computational complexities. Third, the majority of communications within IoT networks are wireless, meaning wireless congestion is another important consideration. Consequently, the cyber operations (processing, storage, and communication) can no longer be taken for granted and must instead be viewed as a scarce, globally shared resource.

Given these new-age design considerations, it is not surprising that event-triggering has recently been gaining a lot of traction as a promising paradigm for addressing the issues above (Perera et al., 2014; Kolios et al., 2016). Event-triggered methods are useful here in that they address precisely *when* different actions (e.g., sensor sample, wireless communication) should occur to efficiently maintain some desired property. The resurgence we see now may be credited to the seminal works (Åström and Bernhardsson, 1999, 2002; Årzén, 1999), where the advantages of event-triggered control over periodic implementations were highlighted. Interestingly, Albert and Bosch (2004) compared the differences between event-triggered and time-triggered distributed control systems and concluded that one of the main deficiencies of periodic control is in its lack of flexibility and scalability. Given the current vision of the IoT being extremely massive and interactive, it is clear we need methods to help enhance flexibility and scalability for these systems of the future. These ideas have thus been gaining more momentum which we also credit partly to the rise of networked control systems in general.

Technical Challenges Specific to Networked Systems

Early works on the subject assume a single decision-maker is responsible for when different actions should be taken by a system; however, we now need ways of implementing these ideas in fully distributed settings to be applicable to the IoT. Heemels et al. (2012) provide a survey of event-triggered and self-triggered control but focus on scenarios where the events are being dictated by a single decision-maker or controller. Instead, the focus of this article is on how to extend these ideas to distributed settings and the new technical challenges that must be addressed in doing so. In particular, we must emphasize the fact that applying the ideas of event-triggered control to

networked systems in general poses many new challenges that do not exist in either area alone. For instance, event-triggered coordination algorithms automatically introduce asynchronism into a system which makes their analysis more difficult. Furthermore, it often becomes difficult to find local triggering rules that agents with distributed information can apply to ensure some system level properties are satisfied; whereas in a centralized setup it is generally easier to find a triggering rule that can directly control some quantity of interest. For example, it is easy to constrain a centralized decision maker to allocate at most a certain number of actions per time period; however, it is more difficult to distribute these decisions to both be efficient and still be sure that the total number of actions per time period constraint is respected.

Why Consensus?

Given the wide variety of opportunistic state-triggered control ideas in networked systems, we have made a conscious decision to focus specifically on consensus problems, as a canonical example of distributed algorithms in general. Nevertheless, our discussion illustrates many of the challenges that arise beyond the specific problem of consensus. For example, it is already known that the separation principle does not hold for event-triggered control systems in general (Ramesh et al., 2011). Since the idea of event-triggered coordination is to take various actions when only deemed necessary, the specific task at hand is tightly coupled with when events should be triggered. However, event-triggered algorithms are certainly not unique solutions to any given problem either. Given a specific problem instance, there are many different event-triggered algorithms that can solve the problem. By choosing a simple, but concrete set of problems, this article discusses many different event-triggered algorithms that have been recently proposed and what exactly the seemingly subtle differences are. At the same time, the set of consensus problems is still general enough that the methods/reasoning behind the event-triggered algorithms we discuss throughout this article are applicable to a number of different application areas related to networked systems. The problem set is also rich enough to capture the same technical difficulties that arise in many other networked event-triggered scenarios such as how to deal with the natural asynchronism introduced into the systems and how to guarantee Zeno behaviors are excluded (e.g., algorithm certificates including deadlocks being avoided).

Organization

We begin in Section 2 by formulating the basic multi-agent consensus problem and provide a short background on the first time event-triggered ideas were applied to it. We then close the section by identifying five different categories of properties to help classify different pairs of consensus problems and event-triggered coordination solutions. In Section 3, we provide details behind the five different categories including the motivations behind

them, and provide numerous examples of different algorithms that fall under the different classifications. Proofs of most results discussed in the article are presented in the Appendix. In Section 4, we take a step back from consensus by providing many different general networking areas that can both directly and indirectly benefit from the ideas discussed in this article. In Section 5, we provide an outlook on the role of event-triggered coordination in networked systems beyond multi-agent consensus and discuss interesting future lines of research. Finally, we gather some concluding thoughts in Section 6.

Preliminaries

We introduce some notational conventions used throughout the article. Let $\mathbb{R}, \mathbb{R}_{>0}, \mathbb{R}_{\geq 0}$, and $\mathbb{Z}_{>0}$ denote the set of real, positive real, nonnegative real, and positive integer numbers, respectively. We denote by $\mathbf{1}_N$ and $\mathbf{0}_N \in \mathbb{R}^N$ the column vectors with entries all equal to one and zero, respectively. The N-dimensional identity matrix is denoted by I_N. Given two matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$, we denote by $A \otimes B \in \mathbb{R}^{mp \times nq}$ as their Kronecker product. We let $\| \cdot \|$ denote the Euclidean norm on $\mathbb{R}^N$. We let $\mathrm{diag}\left(\mathbb{R}^N\right) = \{x \in \mathbb{R}^N \mid x_1 = \cdots = x_N\} \subset \mathbb{R}^N$ be the agreement subspace in $\mathbb{R}^N$. For a finite set S, we let $|S|$ denote its cardinality. Given $x, y \in \mathbb{R}$, Young's inequality states that, for any $\varepsilon \in \mathbb{R}_{>0}$,

$$ xy \leq \frac{x^2}{2\varepsilon} + \frac{\varepsilon y^2}{2}. \tag{1} $$

A weighted directed graph (or weighted digraph) $\mathcal{G}_{\mathrm{comm}} = (V, E, W)$ is comprised of a set of vertices $V = \{1, \ldots, N\}$, directed edges $E \subset V \times V$ and weighted adjacency matrix $W \in \mathbb{R}_{\geq 0}^{N \times N}$. Given an edge $(i, j) \in E$, we refer to j as an out-neighbor of i and i as an in-neighbor of j. The weighted adjacency matrix $W \in \mathbb{R}^{N \times N}$ satisfies $w_{ij} > 0$ if $(i, j) \in E$ and $w_{ij} = 0$ otherwise. The sets of out- and in-neighbors of a given node i are $\mathcal{N}_i^{\mathrm{out}}$ and $\mathcal{N}_i^{\mathrm{in}}$, respectively. The graph $\mathcal{G}_{\mathrm{comm}}$ is *undirected* if and only if $w_{ij} = w_{ji}$ for all $i, j \in V$. For convenience, we denote the set of neighbors of a given node i in an undirected graph as simply $\mathcal{N}_i$. 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 $\mathcal{G}_{\mathrm{comm}}$ is strongly connected if there exists a path between any two vertices. The out- and in-degree matrices D^{out} and D^{in} are diagonal matrices where

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

respectively. A digraph is weight-balanced if $D^{\mathrm{out}} = D^{\mathrm{in}}$. The (weighted) Laplacian matrix is $L = D^{\mathrm{out}} - W$. Based on the structure of L, at least one of its eigenvalues is zero and the rest of them have nonnegative real parts. If the digraph $\mathcal{G}_{\mathrm{comm}}$ is strongly connected, 0 is a simple eigenvalue with associated eigenvector $\mathbf{1}_N$. The digraph $\mathcal{G}_{\mathrm{comm}}$

is weight-balanced if and only if $\mathbf{1}_N^T L = \mathbf{0}_N$, which is also equivalent to $L_s = \frac{1}{2}(L + L^T)$ being positive semidefinite. For a strongly connected and weight-balanced digraph, zero is a simple eigenvalue of L_s. In this case, we order its eigenvalues as $\lambda_1 = 0 < \lambda_2 \leq \cdots \leq \lambda_N$, and note the inequality

$$ x^T L x \geq \lambda_2(L_s) \| x - \frac{1}{N}(\mathbf{1}_N^T x)\mathbf{1}_N \|^2, \tag{2} $$

for all $x \in \mathbb{R}^N$. The following property will also be of use later,

$$ \lambda_2(L_s) x^T L x \leq x^T L_s^2 x \leq \lambda_N(L_s) x^T L x. \tag{3} $$

This can be seen by noting that L_s is diagonalizable and rewriting $L_s = S^{-1} D S$, where D is a diagonal matrix containing the eigenvalues of L_s.

2 What is Event-Triggered Consensus?

In this section we formally state the problem of event-triggered consensus, which results from the application of event-triggered control to the multi-agent consensus problem. We first describe the basic approach to event-triggered control design and then particularize our discussion to event-triggered consensus.

2.1 A primer on event-triggered control

We start by informally describing the event-triggered design approach to stabilization along the lines proposed in (Tabuada, 2007). Given a system on $\mathbb{R}^n$ of the form

$$ \dot{x} = F(x, u) $$

with an unforced equilibrium at x^* (i.e., $F(x^*, 0) = 0$), the starting point is the availability of

(i) a continuous-time controller $k : \mathbb{R}^n \to \mathbb{R}^m$, along with

(ii) a certificate of its correctness in the form of a Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$.

In other words, the closed-loop system $\dot{x} = F(x, k(x))$ makes x^* asymptotically stable, and this fact can be guaranteed through V as Lyapunov function. The idea of event-triggered control is the following: rather than continuously updating the input u as $k(x)$, use instead a sampled version $\widehat{x}$ of the state to do it as $k(\widehat{x})$. This sample gets updated in an opportunistic fashion in a way that still ensures that V acts as a certificate of the resulting sampled implementation. If done properly, this has the advantage of not requiring continuous updates of the input while still guaranteeing the original stabilization of the equilibrium point. The question is then how to determine when the sampled state needs updating. Formally, the closed-loop dynamics looks like

$$ \dot{x} = F(x, k(\widehat{x})) \tag{4} $$

4

and hence one has $\dot{V} = \nabla V(x) \cdot F(x, k(\widehat{x}))$. More specifically, letting $\{t_\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$ denote the sequence of event times at which the control input is updated, the control input is given by

$$u(t) = k(\widehat{x}(t)), \qquad (5)$$

where the sampled state is given by

$$\widehat{x}(t) = x(t_\ell) \quad \text{for} \quad t \in [t_\ell, t_{\ell+1}), \qquad (6)$$

for some sequence of time $\{t_\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$. In other words, the control signal $u(t)$ is only updated at the discrete times t_ℓ and the input is held constant in between events. The goal is to determine a specific event-condition such that the closed-loop system still converges to the desired state.

Under mild conditions on $F, k,$ and V (formally, F uniformly -in x- Lispchitz in its second argument, k Lispchitz, and ∇V bounded), some manipulations of $\dot{V}$ leads to expressions of the form

$$\dot{V} \leq \nabla V(x) \cdot F(x, k(x)) + G(x)\|e\|, \qquad (7)$$

for some function G taking nonnegative values, and where $e = \widehat{x} - x$ is the error between the sampled and the actual state. The first term of the derivative in (7) is negative, while the second vanishes when the sampled state coincides with the actual one, i.e., $e = 0$. Therefore, to ensure that $\dot{V} < 0$, one can simply define a trigger that prescribes that the sampled state should be updated whenever the magnitudes of the first and second term are equal.

This is encoded through what is called a *triggering function* or *event-trigger* $f(\cdot)$, which evaluates whether a given state x and error e combination should trigger an event or not. With a slight abuse of notation, we define this condition as

$$f(e, w) \triangleq g(e) - h(w) = 0, \qquad (8)$$

where $g : \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ is a nonnegative function of the error with $g(0) = 0$ and $h \in \mathbb{R}_{\geq 0}$ is a threshold function that may depend on variables like the state x, the sampled state $\widehat{x}$, or time t, and even additional variables or parameters. For now, we lump them all together in the variable w, and as we make progress in our exposition, we detail what w is in each case. The point of this triggering function (8) is that it guarantees some function of the error $g(e)$ is always smaller than some threshold $h(\cdot)$. This happens because when the condition (8) is satisfied, an event is triggered, which resets the error $e = 0$ and thus $g(e) = 0$ is also reset. Specifically, given a triggering function, event times are implicitly defined by

$$t_{\ell+1} = \min\{t' \geq t_\ell \mid f(e(t'), w(t')) = 0\}. \qquad (9)$$

We are then interested in designing these functions g and h in such a way that the closed-loop dynamics (4) with control inputs (5) driven by (9) ensures $x \to x^*$. Using (7), it is easy to see that the triggering function with $w = x$ defined by

$$g(e) = \|e\|,$$
$$h(x) = \frac{|\nabla V(x) \cdot F(x, k(x))|}{|G(x)|},$$

ensures that

$$\dot{V} \leq \nabla V(x) \cdot F(x, k(x)) + G(x)\|e\| < 0$$

at all times. This fact can ultimately be used to show that x asymptotically approaches x^* as long as there are no deadlocks in the execution. We discuss this point in detail next.

2.1.1 Deadlocks, Zeno behavior, and Minimum Inter-Event Time

With the trigger design in place, one can analyze the behavior of the resulting implementation, such as guaranteeing liveness and the absence of deadlocks. We formalize this in the following definition.

Definition 2.1 (Zeno behavior) *Given the closed-loop dynamics (4) with control inputs (5) driven by (9) a solution with initial condition $x(0) = x_0$ exhibits* Zeno behavior *if there exists $T > 0$ such that $t_\ell \leq T$ for all $\ell \in \mathbb{Z}_{\geq 0}$.*

In other words, if the event-triggered controller defined by the triggering function (8) demands that an infinite number of events (e.g., controller updates) occur in a finite time period, the solution exhibits Zeno behavior. Note that it is possible that depending on the initial condition $x(0) \in \mathbb{R}^n$, different solutions may or may not exhibit Zeno behavior. Only in the case when it is guaranteed that Zeno behavior does not occur along *any* trajectory, we say that the system (as a whole) does not exhibit Zeno behavior.

Being able to rule out Zeno behavior is extremely important in validating the correctness of a given event-triggered controller. In general, the event-triggered algorithms we discuss are comprised of some kind of control law and triggering rule, with the latter driving what information is being used by the control law in real time. The existence of Zeno behavior means there exists an accumulation time $T > 0$ by which an infinite number of events will be triggered. This is problematic for any physical implementation on a real-time system, as it is asking the controller to be updated with new information an infinite number of times in a finite time period.

Another point worth highlighting is the difference between ruling out Zeno behavior versus ensuring a uniform minimum time between any two consecutive events. In fact, the guarantee on lack of Zeno behavior is weaker

than ensuring that there exists a quantity $\tau^{\min}$ that uniformly lower bounds the time in between consecutive events, i.e.,

$$t_{\ell+1} - t_\ell \geq \tau^{\min} > 0$$

for all $\ell \in \mathbb{Z}_{\geq 0}$, which is a more pragmatic property when considering physical hardware. We refer to $\tau^{\min}$ as the minimum inter-event time (MIET) (Borgers and Heemels, 2014). Since dedicated hardware can only operate at some maximum frequency (e.g., a physical device can only broadcast a message or evaluate a function a finite number of times in any finite period of time), ensuring the existence of a positive MIET is more appropriate for physical implementation that simply ruling out Zeno behavior.

Next, we provide examples describing the seemingly subtle differences between these concepts; and more importantly, their implications on correctness and implementation. Consider the dynamic system (4) for which a triggering function f has been defined as in (8) that leads to three different sequences of event times $\{t_\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$ described by (9):

(Zeno behavior): Consider

$$t_{\ell+1} - t_\ell = \frac{1}{(\ell+1)^2},$$

for $\ell \in \mathbb{Z}_{\geq 0}$. Given $t_0 = 0$, this defines the sequence of times as

$$t_\ell = \sum_{n=1}^{\ell} \frac{1}{n^2}.$$

As the number of events $\ell \to \infty$, we have that $t_\ell \leq \frac{\pi^2}{6}$ for all $\ell \in \mathbb{Z}_{\geq 0}$. This means that even if there existed a physical device that can perform actions this quickly, the theoretical analysis of the closed-loop dynamic system (4) is not valid beyond $T = \frac{\pi^2}{6}$.

(Non-Zeno behavior without a positive MIET): Consider

$$t_{\ell+1} - t_\ell = \frac{1}{\ell+1},$$

for $\ell \in \mathbb{Z}_{\geq 0}$. Given $t_0 = 0$, this defines the sequence of times as

$$t_\ell = \sum_{n=1}^{\ell} \frac{1}{n}.$$

In this case, as $\ell \to \infty$ we also have that $t_\ell \to \infty$, which means Zeno behavior can be excluded. However, since the inter-event times $t_{\ell+1} - t_\ell$ go to 0 as $\ell \to \infty$, there does not exist a positive MIET $\tau^{\min}$. This means that

although the theoretical analysis might guarantee stability of the closed-loop dynamic system (4), it would require hardware that can perform actions infinitely fast.

(Positive MIET): Consider

$$t_{\ell+1} - t_\ell = c + \frac{1}{\ell+1},$$

for some $c > 0$ and all $\ell \in \mathbb{Z}_{\geq 0}$. Given $t_0 = 0$, this defines the sequence of times as

$$t_\ell = \sum_{n=1}^{\ell} cn + \frac{1}{n}.$$

We can now guarantee not only the absence of Zeno behavior, but that there exists a positive MIET such that all inter-event times are lower-bounded $t_{\ell+1} - t_\ell \geq \tau^{\min} = c > 0$. This not only guarantees stability of the closed-loop dynamic system (4), but also that the solution can actually be implemented using a device that can take actions at a frequency faster than $\frac{1}{\tau^{\min}}$.

Based on the above discussion, it is important to realize that a complete, fully implementable event-triggered control solution to a problem should also include the existence of a positive MIET.

2.2 Multi-agent average consensus

Here, taking as reference our discussion above, we proceed to describe the multi-agent average consensus problem, identifying as we go the key elements (continuous-time controller and certificate) necessary to tackle the design of event-triggered coordination mechanisms. We start with a simple, yet illustrative, scenario to introduce the main ideas. Towards the end of the section, we discuss various directions along which the problem and its treatment gains in complexity and realism.

We let $\mathcal{G}_{\text{comm}}$ denote the connected, undirected graph that describes the communication topology in a network of N agents. In other words, agent j can communicate with agent i if j is a neighbor of i in $\mathcal{G}_{\text{comm}}$. We denote by $x_i \in \mathbb{R}$ the state of agent $i \in \{1, \ldots, N\}$ and consider single-integrator dynamics

$$\dot{x}_i(t) = u_i(t). \tag{10}$$

It is well-known that the distributed controller

$$u_i^*(t) = -\sum_{j \in \mathcal{N}_i} (x_i(t) - x_j(t)) \tag{11}$$

drives the states of all agents to the average of the initial conditions (Olfati-Saber and Murray, 2004; Olfati-Saber et al., 2007). This is formalized in Theorem 2.2.

Theorem 2.2 (Continuous controller (Olfati-Saber and Murray, 2004)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the dynamics (10), if all agents*

implement the control law (11), then the system asymptotically achieves multi-agent average consensus; i.e.,

$$\lim_{t \to \infty} x_i(t) = \frac{1}{N} \sum_{j=1}^{N} x_j(0) \tag{12}$$

for all $i \in \{1, \ldots, N\}$.

Implementing (11) in a digital setting is not possible since it requires all agents to have continuous access to the state of their neighbors and the control inputs $u_i(t)$ must also be updated continuously. This is especially troublesome in the context of wireless networked systems since this means agents must communicate with each other continuously as well. Instead, researchers have been interested in applying event-triggered strategies to relax these requirements.

2.3 Centralized event-triggered control

Consider the dynamics (10) and the ideal control law (11). Letting $x = (x_1, \ldots, x_N)^T$ and $u = (u_1, \ldots, u_N)^T$, the closed-loop dynamics of the ideal system is given by

$$\dot{x}(t) = -Lx(t), \tag{13}$$

where L is the Laplacian of $\mathcal{G}_{\text{comm}}$. As stated before, implementing this requires all agents to continuously update their control signals which is not realistic for digital controllers. Instead, let us consider a digital implementation of this ideal controller

$$u(t) = -Lx(t_\ell), \quad t \in [t_\ell, t_{\ell+1}), \tag{14}$$

where the event times $\{t_\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$ are to be determined such that the system still converges to the desired state. It is worth mentioning here that the control law (14) is chosen such that the average of all agent states is an invariant quantity regardless of how the event times $\{t_\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$ are chosen, thus preserving the average of the initial conditions throughout the evolution of the system. More specifically, utilizing this controller,

$$\frac{d}{dt}(\mathbf{1}_N^T x(t)) = \mathbf{1}_N^T \dot{x}(t) = \mathbf{1}_N^T Lx(t_\ell) = 0, \tag{15}$$

where we have used the fact that L is symmetric and $L\mathbf{1}_N = 0$.

Let $e(t) = x(t_\ell) - x(t)$ for $t \in [t_\ell, t_{\ell+1})$ be the state measurement error. For simplicity, we denote by $\hat{x}(t) = x(t_\ell)$ for $t \in [t_\ell, t_{\ell+1})$ as the state that was used in the last update of the control signal. The closed-loop dynamics of the controller (14) is then given by

$$\dot{x}(t) = -L\hat{x}(t) = -L(x(t) + e(t)). \tag{16}$$

We are then interested in designing a triggering condition of the form (8) in such a way that the closed-loop

dynamics (16) driven by (9) ensures multi-agent average consensus is achieved. The problem can now be formalized as follows.

Problem 2.3 (Centralized event-triggered consensus) *Given the closed-loop dynamics* (16), *find an event-trigger* $f(\cdot)$ *such that the sequence of times* $\{t_\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$ *ensures multi-agent average consensus* (12) *is achieved.*

Some of the first works to consider this problem were Dimarogonas and Frazzoli (2009); Dimarogonas and Johansson (2009); Kharisov et al. (2010). Following (Dimarogonas et al., 2012), to solve this problem we consider the Lyapunov function

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

Given the closed-loop dynamics (16), we have

$$\dot{V} = x^T L\dot{x} = -x^T LL(x + e) = -\underbrace{\|Lx\|^2}_{\text{"good"}} - \underbrace{x^T LLe}_{\text{"bad"}}. \tag{17}$$

The main idea of (Lyapunov-based) event-triggered control is then to determine when the controller should be updated (i.e., when the error e should be reset to 0) by balancing the "good" term against the potentially "bad" term. More specifically, we are interested in finding conditions on the error e such that $\dot{V} < 0$ at all times. Using norms, we can bound

$$\dot{V} \leq -\|Lx\|^2 + \|Lx\|\|L\|\|e\|.$$

Then, if we can somehow enforce the error e to satisfy

$$\|e\| \leq \sigma \frac{\|Lx\|}{\|L\|},$$

with $\sigma \in (0, 1)$ for all times, we have

$$\dot{V} \leq (\sigma - 1)\|Lx\|^2,$$

which is strictly negative for all $Lx \neq 0$. It is then easy to see that the following centralized event-trigger using

$$g(e) = \|e\|,$$
$$h(x) = \sigma \frac{\|Lx\|}{\|L\|},$$

ensures this is satisfied at all times. Note that in this case we have a state-dependent threshold $h(x)$, but other types of thresholds will be discussed later.

Theorem 2.4 (Centralized event-triggered control (Dimarogonas et al., 2012)) *Given a connected,*

undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (16), *if the event times are determined as the times when*

$$f(e, x) \triangleq \|e\| - \sigma \frac{\|Lx\|}{\|L\|} = 0, \qquad (18)$$

then the system achieves multi-agent average consensus.

In other words, given a control update at time t_ℓ, the next time $t_{\ell+1}$ the controller is updated is given by (9),

$$t_{\ell+1} = \min\{t' \geq t_\ell \mid \|e(t')\| = \sigma \frac{\|Lx(t')\|}{\|L\|}\}.$$

The proof of convergence to the desired state then follows almost directly from the proof of Theorem 2.2 and the fact that the sum of all states is still an invariant quantity. Furthermore, Dimarogonas et al. (2012) are able to rule out the existence of Zeno behavior, cf. Definition 2.1, by showing there exists a positive MIET

$$\tau^{\min} = \frac{\sigma}{\|L\|(1 + \sigma)}$$

uniformly bounding the inter-event times, i.e.,

$$t_{\ell+1} - t_\ell \geq \tau^{\min} > 0 \quad \text{for all} \quad \ell \in \mathbb{Z}_{\geq 0}. \qquad (19)$$

As discussed in Section 2.1.1, the existence of the positive MIET guarantees that the design is implementable over physical platforms.

The centralized event-triggered controller (14) with triggering law (18) relaxes the requirement that agents need to continuously update their control signals; however, it still requires the controller to have perfect state information at all times to be able to evaluate the triggering condition $f(\cdot)$. Next, we provide a distributed solution instead of a centralized one.

2.4 Decentralized event-triggered control

In the previous section we presented a centralized event-triggered control law to solve the multi-agent average consensus problem. Unfortunately, implementing this requires a centralized decision maker and requires all agents in the network to update their control signals simultaneously. Given the nature of and motivation behind consensus problems, this is the first requirement we want to get rid of. Here we present in detail the first real problem of interest concerning this article.

Following (Dimarogonas et al., 2012), consider a distributed digital implementation of the ideal controller (11). In this case we assume each agent i has its own sequence of event times $\{t_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}}$. At any given time t, let

$$\widehat{x}_i(t) = x_i(t_\ell^i) \text{ for } t \in [t_\ell^i, t_{\ell+1}^i) \qquad (20)$$

be the state of agent i at its last update time. The distributed event-triggered controller is then given by

$$u_i(t) = - \sum_{j \in \mathcal{N}_i} (\widehat{x}_i(t) - \widehat{x}_j(t)). \qquad (21)$$

It is important to note here that the latest updated state $\widehat{x}_j(t)$ of agent $j \in \mathcal{N}_i$ appears in the control signal for agent i. This means that when an event is triggered by a neighboring agent j, agent i also updates its control signal accordingly. As in the centralized case, let $e_i(t) = x_i(t_\ell^i) - x_i(t)$ be the state measurement error for agent i. Then, letting $\widehat{x} = (\widehat{x}_1, \ldots, \widehat{x}_N)^T$ and $e = (e_1, \ldots, e_N)^T$, the closed-loop dynamics of the controller (21) is given by

$$\dot{x}(t) = -L\widehat{x}(t) = -L(x(t) + e(t)). \qquad (22)$$

Parallel to the general case in (9), an event-trigger $f_i(\cdot)$ for agent i is a function that determines its sequence of event times $\{t_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}}$ via

$$t_{\ell+1}^i = \min\{t' \geq t_\ell^i \mid f_i(e(t'), w(t')) = 0\}. \qquad (23)$$

The problem we seek to solve can now be formalized as follows.

Problem 2.5 (Decentralized event-triggered consensus) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (22), find an event-trigger $f_i(\cdot)$ for each agent i that is locally computable and such that the sequences of times $\{t_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}}$ ensures multi-agent average consensus (12) is achieved.*

By *locally computable* function f_i, we mean that its value only depends on variables that correspond to agent i and its neighbors. Formally, this means that one can write

$$f_i(e, w) = f_i(e_i, w_i) \triangleq g_i(e_i) - h_i(w_i),$$

where w_i represents information that is locally available to agent i. Unlike in Problem 2.3, where we seek a single event-triggering function $f(\cdot)$ that depends on the global state x to determine a global schedule, here we are interested in having each agent i determine in a distributed way when its local error e_i should be reset to 0.

Following Dimarogonas et al. (2012), to solve this problem we again consider the Lyapunov function

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

Given the closed-loop dynamics (22), we have

$$\dot{V} = -\|Lx\|^2 - x^T L L e. \qquad (24)$$

As before, we are interested in finding conditions on the error e such that $\dot{V} < 0$ at all times; however, we must

now do this in a distributed way. For simplicity, let $Lx \triangleq z = (z_1, \ldots, z_N)^T$. Then, expanding out $\dot{V}$ yields

$$\dot{V} = -\left[\sum_{i=1}^{N} z_i^2 - \sum_{j \in \mathcal{N}_i} z_i(e_i - e_j) \right]$$
$$= -\left[\sum_{i=1}^{N} z_i^2 - |\mathcal{N}_i| z_i e_i + \sum_{j \in \mathcal{N}_i} z_i e_j \right].$$

Using Young's inequality (1) and the fact that $\mathcal{G}_{\text{comm}}$ is symmetric, we can bound this by

$$\dot{V} \leq -\left[\sum_{i=1}^{N} (1 - a|\mathcal{N}_i|) z_i^2 + \frac{1}{a} |\mathcal{N}_i| e_i^2 \right] \qquad (25)$$

for all $a > 0$. Letting $a \in (0, 1/|\mathcal{N}_i|)$ for all i, if we can enforce the error of all agents to satisfy

$$e_i^2 \leq \frac{\sigma_i a (1 - a|\mathcal{N}_i|)}{|\mathcal{N}_i|} z_i^2$$

with $\sigma_i \in (0, 1)$ for all times, we have

$$\dot{V} \leq \sum_{i=1}^{N} (\sigma_i - 1)(1 - a|\mathcal{N}_i|) z_i^2, \qquad (26)$$

which is strictly negative for all $Lx \neq 0$. In order to compute z_i, agent i needs access to its own state and its neighbors states,

$$w_i = x_{\mathcal{N}_i} \triangleq (x_i, \{x_j\}_{j \in \mathcal{N}_i}).$$

The following decentralized event-trigger then ensures that $\dot{V}$ is strictly negative until consensus has been achieved.

Theorem 2.6 (Decentralized event-triggered control (Dimarogonas et al., 2012)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (22), if the event times of each agent i are determined as the times when*

$$f_i(e_i, x_{\mathcal{N}_i}) \triangleq e_i^2 - \frac{\sigma_i a (1 - a|\mathcal{N}_i|)}{|\mathcal{N}_i|} z_i^2 = 0, \qquad (27)$$

with $0 < a < 1/|\mathcal{N}_i|$ for all $i \in \{1, \ldots, N\}$, then all non-Zeno trajectories of the system asymptotically achieve multi-agent average consensus.

Note that the trigger (27) can be evaluated by agent i using only information about its own and neighbors' states. However, it should also be noted that implementing this

algorithm requires each agent i to have exact, continuous state information about its neighbors $\{x_j(t)\}_{j \in \mathcal{N}_i}$. We address this in Section 3.1 below.

The proof of convergence to the desired state then directly follows from the proof of Theorem 2.2 and the fact that the sum of all states is still an invariant quantity. However, it is important to note that this argument is only valid along non-Zeno trajectories, as discussed in Section 2.1.1. Recall that the result of Theorem 2.4 claimed all trajectories of the system achieves multi-agent average consensus, but this was only possible since it was already established in (19) that Zeno behavior is impossible using the trigger (18) proposed in Theorem 2.4 due to the existence of the positive MIET $\tau^{\min}$.

Instead, in the derivation of the result of Theorem 2.6, Dimarogonas et al. (2012) only show that at all times there exists one agent i for which the inter-event times are strictly positive. Unfortunately, this is not enough to rule out Zeno behavior, which is quite problematic, both from a pragmatic and theoretical viewpoint, as the trajectories of the system are no longer well-defined beyond the accumulation point in time. Consequently, the main convergence result can only be concluded for trajectories that do not exhibit Zeno behavior. Since Zeno behavior has in fact not yet been ruled out for all trajectories using the trigger (27), the milder result of Theorem 2.6 is all one can state.

The intuitive reason for this is actually quite simple but it leads to troubling implications: The main idea behind event-triggered control is to only take certain actions when necessary. Since we are interested in decentralized control protocols to achieve consensus for a large system, it is easy to imagine some rare cases where some agent i^* is already in agreement with its neighbors $j \in \mathcal{N}_{i^*}$, but the rest of the system has not yet finished evolving. In this case, once agent i reaches local consensus with its neighbors, it wants to remain there. Unfortunately, this means that the instant any of its neighbors begins to change its state (because the rest of the network has not yet stabilized), the trigger prescribes that agent i^* acts in response.

More specifically, looking at the trigger (27) reveals that when $z_{i^*} = (Lx)_{i^*} = 0$ for some agent i^*, the algorithm presented in Theorem 2.6 is demanding that events be triggered continuously, i.e., that the control signal be updated continuously. This happens because the instance one of agent i^*'s neighbors begins moving, agent i^* should also be moving immediately, but the only way to ensure this is to update the control signal continuously. Since this is not physically possible, the result of Theorem 2.6 is incomplete until we can rule out the possibility of Zeno behavior.

Remark 2.7 (Zeno behavior and general networked systems) The issue pointed out above is not specific to consensus problems, and in fact is characteristic of distributed event-triggered algorithms operating

on networks. More specifically, when a centralized controller is determining when events are triggered, this results in a single time schedule for which it must be guaranteed that an infinite number of events are not triggered in a finite time period. However, when developing a distributed event-triggered strategy, individual agents make independent decisions regarding when events occur based on partial information. This may not only result in many more triggers occurring than in the centralized case, but also considerably complicates obtaining guarantees about avoiding deadlocks in the network. Such analysis usually requires the characterization of additional properties of the original algorithm regarding robustness to error and the impact of the inter-agent interconnections on the evolution of their states.   ●

2.5 Classification of Event-Triggered Consensus Algorithms

We have presented above in detail the distributed event-triggered control problem (Problem 2.5) and solution (Theorem 2.6). Since the conception of this problem and solution, the literature has grown significantly both in numbers and complexity of the problems and solutions considered. To help navigate it, our goal here is to identify a number of categories to systematically classify different problem-solution pairs by their properties. For example, we define this particular problem-solution pair (Problem 2.5 and Theorem 2.6) to have single-integrator dynamics, an undirected interaction graph, events that trigger control updates, triggers that are evaluated continuously, and trigger thresholds that are state-dependent.

In particular, we focus on five main categories to help distinguish different problem instances and their solutions: **Dynamics, Topology, Trigger Response, Event Detection,** and **Trigger Dependence.** The first two categories are related to the physical problem setup, where Dynamics describes the specific type of agent dynamics and Topology captures the type of interactions across network agents. The last three categories are related to the capabilities and/or assumptions placed on the agents communication/computation abilities. Trigger Response refers to the actions taken by agents in response to an event being triggered, Event Detection refers to how events described by triggering functions are monitored, and finally, Trigger Dependence refers to the arguments and variables that triggering functions depend on. Table 1 summarizes the main distinctions which are covered in this section in further detail.

We begin by discussing the shortcomings of the problem-solution pair presented in Section 2.4, and how they can be addressed. The remainder of this article is then devoted to describing in detail exactly what the different categories of Table 1 mean, showing exactly how the different properties change the canonical problem-solution pair described in Section 2.4 (Problem 2.5 and Theorem 2.6), and surveying the vast field in terms of these newly proposed categories and properties.

3 Event-Triggered Consensus Algorithms

In this section we carefully discuss the different types of event-triggered consensus algorithms outlined in Section 2.5. We begin by exploring the different roles a triggering function has on the system. More specifically, we look at what types of actions agents take in response to a trigger, how often the triggering functions are evaluated, and what exactly the triggering functions depend on. We partition this discussion on triggers into three categories: Trigger Response, Event Detection, and Trigger Dependence.

Let us discuss what capabilities agents physically need to realize different solutions to Problem 2.5. We note that, barring the distributed computation aspect, the centralized event-triggered controller presented in Theorem 2.4 is a solution to Problem 2.5, where all agent triggers $f_i(\cdot)$ are defined as in (18). However, implementing this solution requires all agents to have exact global state information x at all times to properly monitor the function (18). Instead, Theorem 2.6 relaxes this requirement by providing a local event-triggering function $f_i(e_i, x_{\mathcal{N}_i})$ that each agent i can monitor with only its neighbors' state information $x_{\mathcal{N}_i}$. However, this solution still requires each agent i to have exact state information about their neighbors at all times. If state information is communicated wirelessly, this means continuous wireless communication to implement the solution. In the following, we propose various solutions to Problem 2.5 that require less stringent assumptions.

3.1 Trigger Response

We begin by discussing the different actions that agents might take in response to an event being triggered. In the previous section, we presented event-triggered *control* laws to determine when control signals should be updated; however, this relies on the continuous availability of some state information. In particular, each agent i requires exact state information about their neighbors $j \in \mathcal{N}_i$ to evaluate the trigger (27) and determine when its control signal u_i should be updated. Instead, here we are interested in applying the event-triggered paradignm to also drive when communication among agents should occur in addition to control updates. We refer to the combination of event-triggering for communication and control as 'event-triggered coordination.'

As in Section 2.4, we assume each agent i has its own sequence of event times $\{t^i_\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$. However, these event times now correspond to when messages are broadcast by agent i; not just when control signals are updated. At any given time t, let

$$\widehat{x}_i(t) = x_i(t^i_\ell) \text{ for } t \in [t^i_\ell, t^i_{\ell+1}) \qquad (28)$$

be the last *broadcast* state of agent i. Then, at any given time t, agent i only has access to the last broadcast state $\widehat{x}_j(t)$ of its neighbors $j \in \mathcal{N}_i$ rather than exact states $x_j(t)$.

Category	Properties	Technical Meaning
Trigger Response (Section 3.1)	control update only	u_i updated at event times $\{t_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}}$
	control update and information push	$u_i, \widehat{x}_i$ updated at event times
	control update and information pull	$u_i, \{\widehat{x}_j\}_{j \in \mathcal{N}_i}$ updated at event times
	control update and information exchange	$u_i, \widehat{x}_i, \widehat{x}_j$ (for some $j \in \mathcal{N}_i$) updated at event times
Event Detection (Section 3.2)	continuous	trigger evaluated at all times $t \in \mathbb{R}_{\geq 0}$
	periodic	trigger evaluated periodically $t \in \{0, h_i, 2h_i, \dots\}$
	aperiodic	trigger evaluated aperiodically $t \in \{t_0, t_1, \dots\}$
Trigger Dependence (Section 3.3)	static: state	$f_i(\cdot) = f_i(e_i, w_i)$
	static: time	$f_i(\cdot) = f_i(e_i, t)$
	dynamic	$f_i(\cdot) = f_i(e_i, w_i, \chi_i), \quad \dot{\chi}_i = \eta_i(e_i, w_i, \chi_i)$
Topology (Section 3.4)	static	communication graph $\mathcal{G}_{\mathrm{comm}}$ is fixed/constant
	dynamic	communication graph $\mathcal{G}_{\mathrm{comm}}$ is changing over time
Dynamics (Section 3.5)	single-integrator	$\dot{x}_i(t) = u_i(t)$
	double-integrator	$\ddot{x}_i(t) = u_i(t)$
	linear	$\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t)$
	nonlinear	$\dot{x}_i(t) = F_i(x_i(t), u_i(t))$

Table 1
Description of the technical differences between different category classifications.

The distributed event-triggered controller is then still given by

$$u_i(t) = -\sum_{j \in \mathcal{N}_i} (\widehat{x}_i(t) - \widehat{x}_j(t)). \qquad (29)$$

It is important to note here that the latest broadcast state $\widehat{x}_j(t)$ of agent $j \in \mathcal{N}_i$ appears in the control signal for agent i at any time t. This means that when an event is triggered by a neighboring agent j, agent i also updates its control signal accordingly. As before, let $e_i(t) = \widehat{x}_i(t) - x_i(t)$ be the state measurement error for agent i. Then, letting $\widehat{x} = (\widehat{x}_1, \dots, \widehat{x}_N)^T$ and $e = (e_1, \dots, e_N)^T$, the closed-loop dynamics of the controller (29) is again given by

$$\dot{x}(t) = -L\widehat{x}(t) = -L(x(t) + e(t)). \qquad (30)$$

However, it should be noted that we are now looking for an event-trigger for each agent i that does not require exact information about its neighbors. More specifically, we recall the result of Theorem 2.6 and notice that the event-trigger for agent i depends on the exact state $x_j(t)$ of all its neighbors $j \in \mathcal{N}_i$. It was first identified by Kharisov et al. (2010) that this solution may not be practical in many cases, particularly in wireless network settings, as this means agents must be in constant communication with each other. Instead, we are interested in finding a solution that only depends on the last broadcast information $\widehat{x}_j(t)$.

Specifically, we are now looking for a triggering function f_i that only depends on its own state x_i and the last broadcast state of its neighbors $\{\widehat{x}_j\}_{j \in \mathcal{N}_i}$, rather than their true states $\{x_j\}_{j \in \mathcal{N}_i}$. Following (Nowzari and Cortés, 2014; Garcia et al., 2013), to solve this problem we again consider the Lyapunov function

$$V(x) = \frac{1}{2}x^T L x. \qquad (31)$$

Given the closed-loop dynamics (30), we have

$$\dot{V} = -\|Lx\|^2 - x^T L L e,$$

just as we did in (17) and (24). However, since we are interested in identifying conditions for $\dot{V}$ to be negative in terms of the most recently broadcast information $\widehat{x}$ instead of actual state information, we can rewrite this using $e = \widehat{x} - x$ as

$$\dot{V} = -\|L\widehat{x}\|^2 + \widehat{x}^T L L e.$$

Letting $\widehat{z} = L\widehat{x} = (\widehat{z}_1, \dots, \widehat{z}_N)$, it is easy to see that we can bound

$$\dot{V} \leq -\left[\sum_{i=1}^N (1 - a|\mathcal{N}_i|)\widehat{z}_i^2 + \frac{1}{a}|\mathcal{N}_i|e_i^2\right]$$

for all $a > 0$ following essentially the same steps to arrive at (25). Letting $a \in (0, 1/|\mathcal{N}_i|)$ for all i, if we can enforce

the error of all agents to satisfy

$$e_i^2 \leq \frac{\sigma_i a(1 - a|\mathcal{N}_i|)}{|\mathcal{N}_i|} \widehat{z}_i^2 \qquad (32)$$

with $\sigma_i \in (0, 1)$ for all times, we have

$$\dot{V} \leq \sum_{i=1}^{N} (\sigma_i - 1)(1 - a|\mathcal{N}_i|)\widehat{z}_i^2$$

which is strictly negative for all $L\widehat{x} \neq 0$. In order to compute $\widehat{z}_i$, agent i only needs access to its own and neighbors' broadcast states rather than true states,

$$w_i = \widehat{x}_{\mathcal{N}_i} \triangleq (\widehat{x}_i, \{\widehat{x}_j\}_{j \in \mathcal{N}_i}).$$

The following decentralized event-trigger then ensures that $\dot{V}$ is strictly negative until consensus is achieved.

Theorem 3.1 (Decentralized event-triggered coordination (Garcia et al., 2013)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (30), if event times of each agent i are determined as the times when*

$$f_i(e_i, \widehat{x}_{\mathcal{N}_i}) \triangleq e_i^2 - \frac{\sigma_i a(1 - a|\mathcal{N}_i|)}{|\mathcal{N}_i|} \widehat{z}_i^2 \geq 0, \qquad (33)$$

with $0 < a < 1/|\mathcal{N}_i|$ for all $i \in \{1, \ldots, N\}$, then all non-Zeno trajectories of the system asymptotically achieve multi-agent average consensus.

It is important to note here that unlike the previous triggers (18) and (27), agent i's event is triggered whenever the inequality (33) is satisfied rather than an equality. This is because unlike the previous two triggering functions, this one is discontinuous because it depends on the last broadcast state $\widehat{x}$ rather than the exact state x, which can abruptly change anytime an agent triggers an event. In any case, the point of the trigger is to ensure (32) is satisfied at all times, which this trigger does (because the instant it is violated for some agent i, agent i can immediately set $e_i = 0$).

Note that Theorem 3.1) and Theorem 2.6 both solve the exact same problem: Problem 2.5; except we have now considered events that not only determine when control signals should be updated, but also when agents should broadcast information to their neighbors. In more general terms, we refer to this as a **Trigger Response** of control signal updates *and* information pushing. Information pushing, or broadcasting, refers to the action of an agent i pushing unsolicited information onto its neighbors $j \in \mathcal{N}_i$. Table 1 describes the different ways in which we classify both the problems and the solutions.

Following Table 1, we can say this problem-solution pair has single-integrator dynamics, an undirected interaction graph, events that trigger broadcasts and control updates, triggers that are evaluated continuously, and trigger thresholds that are state-dependent. Note, however, that these properties alone are not enough to uniquely identify a solution to Problem 2.5. Next, we present an alternate solution to Problem 2.5 that is described by the exact same properties listed above.

Following Nowzari and Cortés (2016), to solve this problem in a different way we consider a different Lyapunov function,

$$V(x) = \frac{1}{2}(x - \bar{x}\mathbf{1})^T (x - \bar{x}\mathbf{1}), \qquad (34)$$

where $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i(0)$ is the average of all initial conditions. Then, given the closed-loop dynamics (30), we have

$$\dot{V} = x^T \dot{x} - \bar{x}\mathbf{1}^T \dot{x} = -x^T L\widehat{x} - \bar{x}\mathbf{1}^T L\widehat{x} = -x^T L\widehat{x},$$

where we have used the fact that the graph is symmetric in the last equality. As always, we are interested in finding conditions on the error e such that $\dot{V} < 0$ at all times; however, we must now do it without access to neighboring state information.

As in the previous solution methods, our first step is to upper-bound $\dot{V}$ to find conditions to ensure it is never positive. The following result from (Nowzari and Cortés, 2016) can then be used to find conditions that ensure $\dot{V} < 0$ at all times.

Lemma 3.2 ((Nowzari and Cortés, 2016)) *Given $V(x) = \frac{1}{2}(x - \bar{x}\mathbf{1})^T (x - \bar{x}\mathbf{1})$ and the closed-loop dynamics (30),*

$$\dot{V} \leq \sum_{i=1}^{N} e_i^2 |\mathcal{N}_i| - \sum_{j \in \mathcal{N}_i} \left(\frac{1}{4}(\widehat{x}_i - \widehat{x}_j)^2 \right).$$

Leveraging Lemma 3.2, it is easy to see that if we can enforce the error of all agents to satisfy

$$e_i^2 \leq \sigma_i \frac{1}{4|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} (\widehat{x}_i - \widehat{x}_j)^2 \qquad (35)$$

with $\sigma_i \in (0, 1)$ for all times, we have

$$\dot{V} \leq \sum_{i=1}^{N} \frac{\sigma_i - 1}{4} \sum_{j \in \mathcal{N}_i} (\widehat{x}_i - \widehat{x}_j)^2, \qquad (36)$$

which is strictly negative for all $L\widehat{x} \neq 0$. For simplicity, we use the shorthand notation

$$\widehat{\phi}_i = \sum_{j \in \mathcal{N}_i} (\widehat{x}_i - \widehat{x}_j)^2$$

in the definition of the following decentralized event-trigger, which ensures that this is satisfied at all times.

Theorem 3.3 (Decentralized event-triggered coordination (Nowzari and Cortés, 2016)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics* (30), *if the event times of each agent i are determined as the times when*

$$f_i(e_i, \widehat{x}_{\mathcal{N}_i}) \triangleq e_i^2 - \sigma_i \frac{1}{4|\mathcal{N}_i|}\widehat{\phi}_i \geq 0, \qquad (37)$$

for all $i \in \{1, \ldots, N\}$, then all non-Zeno trajectories of the system achieve multi-agent average consensus.

Note that both solutions presented in Theorem 3.1 and Theorem 3.3 are classified the same way according to the categories in Table 1, because they can be implemented under the same assumptions on the agent capabilities. In this case agents need to be able to receive broadcasted information from their neighbors on some connected, undirected communication graph. Note that both of these solutions are still technically incomplete as they do not rule out the possibility of Zeno behavior. In fact, the original design in Nowzari and Cortés (2016) includes an additional trigger to ensure that Zeno behavior does not occur.

We have now discussed two different actions an agent might take in response to an event being triggered: control updates and broadcasting a message. Instead, one could imagine other types of actions resulting from an event being triggered as well. In particular, in Table 1 we highlight the possibility of events triggering *information pulls* or *exchanges*, rather than broadcasts (or information pushes). More specifically, we refer to an information push by an agent as a broadcast message that can be received by all its neighbors. Instead, we can think of an information pull by an agent as a request for updated information to its neighbors. That is, when an event is triggered by an agent, rather than telling its neighbors its current state, it instead requests state information from its neighbors. This idea is related to self-triggered control design (Anta and Tabuada, 2010; Heemels et al., 2012), where the decision maker, at each event time, immediately schedules its next event time with the information available at the current event time (rather than continuously monitoring a triggering condition as one normally does in event-triggered control). In self-triggered coordination with wireless communication, agents use their current information to determine when in the future they need to acquire new information from others (Nowzari and Cortés, 2012; De Persis and Frasca, 2013; Fan et al., 2015; Henriksson et al., 2015).

In addition to information pushes or pulls, one could also imagine scenarios in which an information exchange or swap may be more practical. Applying event-triggered ideas to gossiping protocols has recently been called edge-based event-triggered coordination (De Persis and

Frasca, 2013; De Persis and Postoyan, 2017; Cao et al., 2015; Xu et al., 2016; Wei et al., 2018; Duan et al., 2017). In this setting events are triggered along specific edges of a graph rather than its nodes. More specifically, rather than a single agent sending information to or requesting information from *all* its neighbors at once, events are instead triggered at the link level which drives a direct agent-to-agent information exchange.

3.2 Event Detection

Having discussed the different types of actions that an event can drive, until now we have assumed that the triggering functions can be monitored continuously. This is troublesome since this is technically not possible for cyber-physical systems. Unfortunately, even if continuous event detection were possible, most of the algorithms presented in the article thus far are not guaranteed to avoid Zeno behaviors making them risky to implement on real systems. In fact, as mentioned above, until Zeno behavior is guaranteed not to occur in the system, the convergence results of Theorems 2.6, 3.1, and 3.3 are not even valid.

Consequently, being able to properly rule out the existence of Zeno behavior in an event-triggered consensus problem is both subtle and critical for its correctness. Recalling Definition 2.1, Zeno behavior is defined as having an infinite number of events triggered in any finite time period. Unfortunately, it turns out that in all the algorithms presented so far, it is not guaranteed that Zeno behavior will not occur.

Note that in some cases the algorithms can be slightly modified to theoretically avoid Zeno behavior, but even in these cases it turns out that the time between two events generated by a single agent may be arbitrarily small, see e.g., Nowzari and Cortés (2016). More specifically, even if it can be guaranteed that an infinite number of events are not triggered in any finite time period, the time between two events might not have a uniform lower bound. This means that even with a non-Zeno guarantee, this is still troublesome from an implementation viewpoint because an agent's hardware/software physically cannot keep up with how quickly events are being generated, cf. Section 2.1.1.

Motivated by this discussion, researchers have considered enforcing a minimum time between events as a more practical constraint for event-triggered solutions. Until now we have assumed that all event-triggers can be evaluated continuously. Or more specifically we say that the **Event Detection** occurs continuously. That is, the exact moment at which a triggering condition is met, an action (e.g., state broadcast and control signal update) is carried out. However, as mentioned above this may be an unrealistic assumption when considering actual digital implementations. More specifically, a real device cannot continuously evaluate whether a triggering condition has occurred or not.

This observation motivates the need for relaxing the con-

tinuous event detection requirement and instead determine a discrete set of times at which triggering functions should be evaluated. The most natural way to approach this is to study sampled-data (or periodically checked) event-triggered coordination strategies.

Specifically, given a sampling period $h \in \mathbb{R}_{>0}$, we let $\{t_{\ell'}\}_{\ell' \in \mathbb{Z}_{\geq 0}}$, where $t_{\ell'+1} = t'_\ell + h$, denote the sequence of times at which agents evaluate decisions about whether to broadcast their state to their neighbors or not. This type of design is more in line with the constraints imposed by real-time implementations, where individual components work at some fixed frequency, rather than continuously. An inherent and convenient feature of this strategy is the automatic lack of Zeno behavior (since inter-event times are naturally lower bounded by h).

Under the new framework we still have familiar equations. In particular, the control law of each agent is still given by

$$u_i(t) = -\sum_{j \in \mathcal{N}_i} (\widehat{x}_i - \widehat{x}_j), \qquad (38)$$

which means we still have the same closed-loop dynamics,

$$\dot{x}(t) = -L\widehat{x}(t) = -L(x(t) + e(t)). \qquad (39)$$

The difference only shows up when considering when broadcasts occur. That is, we now have

$$\widehat{x}_i(t) = x_i(t^i_\ell) \text{ for } t \in [t^i_\ell, t^i_{\ell+1}), \qquad (40)$$

just as we did in (28) when considering continuous event detection, except now the event times $\{t^i_\ell\} \subset \{0, h, 2h, \dots\}$ can only occur at discrete time-instances.

To again solve Problem 2.5 by now relaxing the continuous monitoring requirement, let us begin by revisiting the result of Theorem 3.3, where we used the Lyapunov function (34),

$$V(x) = \frac{1}{2}(x - \bar{x}\mathbf{1})^T(x - \bar{x}\mathbf{1}), \qquad (41)$$

and Lemma 3.2 to find conditions on the error e_i such that $\dot{V}$ was always negative. Since the Lyapunov function here and closed-loop dynamics (39) has the exact same functional form as before, Lemma 3.2 still holds as well. That is, given the Lyapunov function (41) and closed-loop dynamics (39), we have the upper-bound

$$\dot{V} \leq \sum_{i=1}^N e_i^2 |\mathcal{N}_i| - \sum_{j \in \mathcal{N}_i} \left(\frac{1}{4}(\widehat{x}_i - \widehat{x}_j)^2 \right). \qquad (42)$$

Then, just as before, we want to find conditions on e_i such that this is always negative. However, the issue now is that we can only generate events at the discrete

times $\{0, h, 2h, \dots\}$, meaning we can only reset the error $e_i(t) = 0$ at these specific times rather than any time $t \in \mathbb{R}_{\geq 0}$. Thus, we must now find a triggering condition that is only evaluated at these sampling times, but still guarantees that the righthand side of (42) is negative *for all $t \in \mathbb{R}_{\geq 0}$*.

Intuitively, as long as the sampling period h is small enough, the closed-loop system with a periodically checked event-triggering condition will behave similarly to the system with triggers being evaluated continuously. Interestingly, two different groups have developed two different, albeit similar, algorithms based on the same Lyapunov function (41) using two different ways of upper-bounding its time-derivative. We omit the details here but present the two solutions to this problem based on the works (Meng and Chen, 2013; Nowzari and Cortés, 2016) next.

Theorem 3.4 (Periodic event-triggered coordination (Meng and Chen, 2013)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (39), if the event times of each agent i are determined as the times $t' \in \{0, h, 2h, \dots\}$ when*

$$f_i(e_i, \widehat{x}_{\mathcal{N}_i}) \triangleq e_i^2 - \sigma_i \widehat{z}_i^2 \geq 0,$$

and $h \in \mathbb{R}_{>0}$ and σ_{max} satisfy

$$h \leq \frac{1}{2\lambda_N} \text{ and } \sigma_{max} < \frac{1}{\lambda_N^2}, \qquad (43)$$

where $\sigma_{max} = \max_{i \in \{1, \dots, N\}} \sigma_i$, then the system achieves multi-agent average consensus.

Theorem 3.5 (Periodic event-triggered coordination (Nowzari and Cortés, 2016)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (39), if the event times of each agent i are determined as the times $t' \in \{0, h, 2h, \dots\}$ when*

$$f_i(e_i, \widehat{x}_{\mathcal{N}_i}) \triangleq e_i^2 - \sigma_i \frac{1}{4|\mathcal{N}_i|} \widehat{\phi}_i \geq 0, \qquad (44)$$

and $h \in \mathbb{R}_{>0}$ and σ_{max} satisfy

$$\sigma_{\max} + 4h|\mathcal{N}_{max}|^2 < 1, \qquad (45)$$

where $|\mathcal{N}_{max}| = \max_{i \in \{1, \dots, N\}} |\mathcal{N}_i|$, then the system achieves multi-agent average consensus.

Note that the results of Theorems 3.4 and 3.5 guarantee that *all trajectories* of the systems can achieve multi-agent average consensus under their respective conditions since Zeno executions are trivially ruled out because agents can only trigger an event at most every $h > 0$ seconds. This trivially gives us a positive MIET $\tau^{\min} = h$. We also note here the small difference in the triggering

14

functions and conditions on h and $\sigma_{\max}$ for convergence of these results are a result of different ways of upper-bounding $\dot{V}$ and ultimately being able to guarantee that $\dot{V} < 0$ at all times. In particular, we note that the conditions for guaranteeing convergence in Theorem 3.4 are less conservative but requires the algebraic connectivity of the communication graph; whereas the conditions for guaranteeing convergence in Theorem 3.5 may be more strict but easier to compute by the agents themselves.

A drawback of these and similar solutions (Meng et al., 2015; Fuan et al., 2016) is that the period h must be the same for all agents, requiring synchronous action. This assumption may be restrictive in practical scenarios where data cannot be consistently acquired. Instead, it seems desirable to develop asynchronous versions of these solutions or, more generally, solutions where the **Event Detection** occurs aperiodically rather than continuously or periodically. We are only aware of a few recent works that have begun investigating the asynchronism issue (Meng et al., 2017; Liu et al., 2017; Duan et al., 2017; Liu et al., 2017).

More specifically, new algorithms may be required to consider the case of aperiodic sampled-data event detection, or even self-triggered event detection. In the former case agents would obtain samples at different instances of time, and then take appropriate actions in response. In the latter case, one could imagine a scenario where agents are not only responsible for determining when communication should occur, but also when local samples should be taken. In this case it may be useful to consider self-triggered sampling combined with event-triggered communication and control. More specifically, the agents would determine by themselves when future samples should be taken, and then event-decisions should be made based on the taken samples.

3.3 Trigger Dependence

We have now discussed what agents should do in response to a trigger and how carefully these triggers need to be monitored. We are now interested in studying what these triggering functions should actually depend on and why. In particular, we have only considered triggering functions so far that depend on locally available information and no exogenous signals. In this section we present the difference between static and dynamic triggering functions. A static triggering function means that the trigger only depends on currently available information (i.e., memoryless), whereas a dynamic triggering function may depend on additional internal dynamic variables.

We begin with static time-dependent triggering functions, rather than the state-dependent ones we have used until now. Let us now revisit Problem 2.5 again, except this time we are interested in designing an event-trigger threshold that is time-dependent rather than state-dependent. The time-dependent event-trigger to solve this problem was first developed by Seyboth et al. (2013) and is presented next.

Theorem 3.6 (Decentralized event-triggered coordination (time-dependent) (Seyboth et al., 2013)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (30), if the event times of each agent i are determined as the times when*

$$f_i(e_i, t) \triangleq \|e_i\| - (c_0 + c_1 e^{-\alpha t}) = 0, \qquad (46)$$

with constants $c_0, c_1 \geq 0$ and $c_0 + c_1 > 0$, then all non-Zeno trajectories of the system reach a neighborhood of multi-agent average consensus upper-bounded by

$$r = \|L\|\sqrt{N}c_0/\lambda_2(L).$$

Moreover, if $c_0 > 0$ or $0 < \alpha < \lambda_2(L)$, then the closed-loop system does not exhibit Zeno behavior.

This solution uses a triggering function whose threshold depends on time rather than state. Thus, we can say this problem-solution pair has single-integrator dynamics, an undirected interaction graph, events that trigger broadcasts and control updates, triggers that are evaluated continuously, and trigger thresholds that are time-dependent rather than state-dependent.

We point out here the closely related notion of "event-triggered mechanism" (ETM), as presented in (Borgers and Heemels, 2014), where three classes are presented: relative, absolute, and mixed. Here, we have proposed slightly more general classes of trigger dependencies such that the relative ETM is a special case of our state-dependent triggers. Similarly, the absolute ETM (or constant triggering threshold) is a special case of the time-dependent trigger of Theorem 3.6 with $c_1 = 0$, which were among the first types of event-trigger thresholds considered in network settings unrelated to consensus (Miskowicz, 2006; Zhong and Cassandras, 2010). The mixed ETM is a combination of these two triggers, but we do not discuss the distinction in this article.

Constant thresholds (or absolute ETMs) give two main advantages. The first is their simplicity to implement, and the second is that it is generally easy to rule out the possibility of Zeno behavior for them. Since the threshold is a constant, it usually takes some nonzero minimum amount of time for the error to be able to reach this threshold from zero. Note that this is evident in the result of Theorem 3.6 which guarantees for $c_0 > 0$ that Zeno behaviors do not occur. The drawback is that the constant thresholds generally do not generate events at times that align well with the evolution of the task at hand, and hence, the price we pay is that one is not able to guarantee exact convergence all the way to the desired states. This is discussed in detail next as a particular case of the time-dependent algorithm presented above.

Both the advantages and disadvantages of the event-triggered coordination law with the time-dependent triggers proposed in Theorem 3.6 come from the tunable design parameters c_0, c_1, and α, which play important roles

in the performance of the algorithm (e.g., convergence speed and amount of events triggered). For example, setting c_1 small and α large increases the convergence rate at the cost of more events being triggered, whereas setting a large c_0 reduces the number of events being triggered at the cost of not being able to converge exactly to the initial average. These parameters can then be tuned to give a desired balance between performance and efficiency. Another advantage of the time-dependent triggers are their simplicity to design and implement.

Unfortunately, there are also some physical limits to how these parameters can be tuned to guarantee Zeno behaviors do not occur. For example, if α is set too high we may be asking the system to converge faster than is physically possible, leading to an infinite number of events being generated in a finite time. In particular we focus our discussion here on the parameters c_0 and α and their effects on convergence and possible Zeno behaviors. We begin with the more desirable $c_0 = 0$ case, as in this case the result of Theorem 3.6 states that the system will asymptotically achieve exact multi-agent average consensus as defined in (12). However, in this case we require $\alpha < \lambda_2(L)$ to guarantee Zeno behaviors can be avoided and, unfortunately, $\lambda_2(L)$ is a global quantity that requires knowledge about the entire communication topology to compute. There are indeed methods for estimating this quantity in a distributed way (see e.g., Aragues et al. (2012); Yang et al. (2010)) but we do not discuss this here. On the other hand, when $c_0 > 0$ we can guarantee that Zeno behaviors are avoided regardless of our choice of α; however, we lose the exact asymptotic convergence guarantee. Note that in the case of constant triggers, i.e., $c_1 = 0$, we must have $c_0 > 0$. That means in these cases we can only guarantee convergence to a neighborhood of the desired average consensus state rather than exact average consensus.

As a result of the above discussion, we see that it is difficult for the agents to choose the parameters c_0, c_1, and α without global knowledge to ensure asymptotic convergence to the average consensus state while also guaranteeing Zeno executions are avoided. On the other hand, using state-dependent triggers might be more risky to implement as it is generally harder to rule out the possibility of Zeno behaviors.

Referring back to Table 1, we have now discussed both types of static triggering functions. Some works have also considered hybrid or mixed event-time driven coordination, where events may be generated by both state and time events (Xiao and Chen, 2016; Xiao et al., 2016; Borgers and Heemels, 2014; Sun et al., 2016). Beyond events generated as functions of time or state, other works have also considered input-based events that depend on the control signal being used (Wu et al., 2016; Adaldo et al., 2016). While the analysis is slightly different, the intuitive idea is similar. When considering state-based events, we generally trigger an event when the error between the true state and state currently used by the controller ex-

ceeds some threshold. However, ultimately what matters in a control system is the signal being used. Thus, these input-based triggering functions instead define an input error between the actual input being used and the desired input if exact state information was available. Thus, even if the state error is large, these algorithms do not trigger a controller update until the input error exceeds some threshold.

Beyond static triggers, the idea of a dynamic event-triggering function has recently been applied as a promising method to rule out Zeno behavior (Dolk and Heemels, 2015; Girard, 2017; Yi et al., 2017; Dolk et al., 2017). In this case the triggering function $f(\cdot)$ depends on an additional, internal dynamic variable with its own dynamics that can be designed separately.

Let us revisit Problem 2.5, except that now we aim to design a dynamic event-triggered coordination strategy that can guarantee average consensus from all initial conditions with no global information and including non-Zeno guarantees. This result is presented next.

Theorem 3.7 (Dynamic event-triggered control (Yi et al., 2017)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (22), if the event times of each agent i are determined by*

$$f_i(e_i, \widehat{x}_{\mathcal{N}_i}, \chi_i) \triangleq |\mathcal{N}_i| e_i^2 - \frac{\sigma_i}{4}\widehat{\phi}_i - \chi_i \geq 0, \qquad (47a)$$

$$\dot{\chi}_i(e_i, \widehat{x}_{\mathcal{N}_i}, \chi_i) = -\chi_i + \frac{\sigma_i}{4}\widehat{\phi}_i - e_i^2, \qquad (47b)$$

with $\chi_i(0) > 0$ and $\sigma_i \in [0, 1)$ for all $i \in \{1, \ldots, N\}$, then the system asymptotically achieves multi-agent average consensus.

Theorem 3.7 fully solves Problem 2.5 in a distributed way. Notably, this solution does not require agents to have any global information to implement the algorithm, and guarantees convergence to the desired consensus state by also guaranteeing Zeno behavior does not occur along any trajectory. However, it should be noted that while this solution theoretically solves Problem 2.5, it does not guarantee the existence of a positive MIET $\tau^{\min}$, which poses problems for practical implementation, as discussed in Section 2.1.1. More recently, Berneburg and Nowzari (2019) have developed a new dynamic triggering strategy that guarantees a positive MIET for each agent, providing a complete and implementable solution to Problem 2.5. This result is formalized next.

Theorem 3.8 (Dynamic event-triggered control with a positive MIET (Berneburg and Nowzari, 2019)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (22), if the event times of each*

Solution classification	Triggering mechanism	Properties
Theorem 2.6 (Dimarogonas et al., 2012) **Trigger Response**: control updates **Event Detection**: continuous **Trigger Dependence**: static: state	$f_i(e_i, x_{\mathcal{N}_i}) \triangleq e_i^2 - \frac{\sigma_i a(1-a|\mathcal{N}_i|)}{|\mathcal{N}_i|} z_i^2 = 0$	requires continuous monitoring of neighbors $x_{\mathcal{N}_i}$; no non-Zeno guarantee
Theorem 3.1 (Garcia et al., 2013) Theorem 3.3 (Nowzari and Cortés, 2016) **TR**: control updates, info push **ED**: continuous **TD**: static: state	$f_i(e_i, \widehat{x}_{\mathcal{N}_i}) \triangleq e_i^2 - \frac{\sigma_i a(1-a|\mathcal{N}_i|)}{|\mathcal{N}_i|} \widehat{z}_i^2 \geq 0$ or $f_i(e_i, \widehat{x}_{\mathcal{N}_i}) \triangleq e_i^2 - \sigma_i \frac{1}{4|\mathcal{N}_i|} \widehat{\phi}_i \geq 0$	no non-Zeno guarantee
Theorem 3.4 (Meng and Chen, 2013) Theorem 3.5 (Nowzari and Cortés, 2016) **TR**: control updates, info push **ED**: periodic **TD**: static: state	(Only at times $t \in \{0, h, 2h, \dots\}$) $f_i(e_i, \widehat{x}_{\mathcal{N}_i}) \triangleq e_i^2 - \sigma_i \widehat{z}_i^2 \geq 0$ or $f_i(e_i, \widehat{x}_{\mathcal{N}_i}) \triangleq e_i^2 - \sigma_i \frac{1}{4|\mathcal{N}_i|} \widehat{\phi}_i \geq 0$	positive MIET $\tau^{\min} = h$; requires synchronous period $h > 0$ to guarantee convergence
Theorem 3.6 (Seyboth et al., 2013) **TR**: control updates, info push **ED**: continuous **TD**: static: time	$f_i(e_i, t) \triangleq \|e_i\| - (c_0 + c_1 e^{-\alpha t}) = 0$	requires algebraic connectivity λ_2 to guarantee non-Zeno ; no positive MIET
Theorem 3.7 (Yi et al., 2017) **TR**: control updates, info push **ED**: continuous **TD**: dynamic	$f_i(e_i, \widehat{x}_{\mathcal{N}_i}, \chi_i) \triangleq |\mathcal{N}_i| e_i^2 - \frac{\sigma_i}{4} \widehat{\phi}_i - \chi_i \geq 0$ $\dot{\chi}_i = -\chi_i + \frac{\sigma_i}{4} \widehat{\phi}_i - e_i^2$	guarantees non-Zeno; no positive MIET
Theorem 3.8 (Berneburg and Nowzari, 2019) **TR**: control updates, info push **ED**: continuous **TD**: dynamic	$f_i(\chi_i) \triangleq -\chi_i \geq 0$ $\dot{\chi}_i = \min\left\{-1, \frac{\widehat{\phi}_i}{e_i^2} + 2(\chi_i+1)\frac{\widehat{z}_i}{e_i} - 1\right\}$	positive MIET $\tau_i^{\min} = \frac{1}{\sqrt{|\mathcal{N}_i|}}\left[\operatorname{atan}(2\sqrt{|\mathcal{N}_i|}) - \operatorname{atan}(\sqrt{|\mathcal{N}_i|})\right]$

Table 2
Summary of solutions to the decentralized event-triggered consensus problem, cf. Problem 2.5, discused in this article. Note that all these solutions assume that the communication **Topology** is undirected and connected and the **Dynamics** of each agent are single-integrators. Table 3 recalls all the relevant terms.

agent i are determined by

$$f_i(\chi_i) \triangleq -\chi_i \geq 0, \tag{48}$$

$$\dot{\chi}_i(e_i, \widehat{x}_{\mathcal{N}_i}, \chi_i) = \min\left\{-1, \frac{\widehat{\phi}_i}{e_i^2} + 2(\chi_i+1)\frac{\widehat{z}_i}{e_i} - 1\right\}, \tag{49}$$

with $\chi_i(t_\ell^i) \triangleq 1$ for all $\ell \in \mathbb{Z}_{\geq 0}$ and $i \in \{1, \dots, N\}$, then the system asymptotically achieves multi-agent average consensus. Moreover, there exists a positive MIET for

each agent i given by

$$\tau_i^{min} = \sqrt{\frac{1}{|\mathcal{N}_i|}}\left[\operatorname{atan}(2\sqrt{|\mathcal{N}_i|}) - \operatorname{atan}(\sqrt{|\mathcal{N}_i|})\right]. \tag{50}$$

Note that the trigger in Theorem 3.8 is slightly different from the rules presented above in that, in addition to the local error e_i being reset to 0 at each event triggered by agent i, the internal dynamic variable χ_i is reset to 1 at these times as well. The existence of a positive MIET (50) makes the solution presented in Theorem 3.8 truly implementable on physical platforms. Although the solutions presented in Theorems 3.4 and 3.5 also provide a trivial MIET $\tau^{\min} = h$ guarantee based on the sampling

period h, they require perfectly synchronized executions among the network agents.

Table 2 summarizes all proposed solutions to Problem 2.5 discussed up to this point in the article while emphasizing their limitations. Table 3 recalls the relevant terms.

x_i	state of agent i
u_i	control input of agent i
$\widehat{x}_i$	last broadcast state of agent i
$f_i(\cdot)$	triggering functions
L	Laplacian matrix
$z_i = (Lx)_i$	$\sum_{j \in \mathcal{N}_i}(x_i - x_j)$
$\widehat{z}_i = (L\widehat{x})_i$	$\sum_{j \in \mathcal{N}_i}(\widehat{x}_i - \widehat{x}_j)$
$\widehat{\phi}_i$	$\sum_{j \in \mathcal{N}_i}(\widehat{x}_i - \widehat{x}_j)^2$
$x_{\mathcal{N}_i} = (x_i, \{x_j\}_{j \in \mathcal{N}_i})$	state of agent i and neighbors
$\widehat{x}_{\mathcal{N}_i} = (\widehat{x}_i, \{\widehat{x}_j\}_{j \in \mathcal{N}_i})$	last broadcast state of
	agent i and neighbors

Table 3
List of terms related to Problem 2.5 and its solutions.

3.4 Topology

We have focused all our solutions so far on solving the same Problem 2.5. Our discussion describes the different types of capabilities on the agents assumed by the different solutions along with their benefits and drawbacks. A commonality between all of them is the requirement of undirected communication topologies and single-integrator dynamics. From here on, we discuss what happens in the case of more complicated topologies and dynamics.

Beyond the scenarios with undirected communication graphs considered so far, here we extend the ideas of the article to the case where communication topologies are directed. The earliest works we are aware of to address this problem are presented in (Liu and Chen, 2011; Liu et al., 2012; Chen et al., 2014; Xie and Xie, 2014), where the authors present either centralized or event-triggered *control* solutions only. In other words, similar to Theorem 2.6, the algorithms assume that agents have continuous access to neighboring state information at all times. Here, we are instead interested in event-triggered coordination strategies similar to Theorems 3.1 and 3.3 that not only determine when control signals should be updated but also when communication should occur.

Here we consider communication topologies that are described by weight-balanced directed graphs. More specifically, we say that agent i can only send messages to its in-neighbors $j \in \mathcal{N}_i^{\text{in}}$ and it can only receive messages from its out-neighbors $j \in \mathcal{N}_i^{\text{out}}$, where the neighboring sets are not necessarily the same. Then, consider the same type of control law as before given in (21) and (29), except each agent can now only use information about their out-neighboring states,

$$u_i(t) = -\sum_{j \in \mathcal{N}_i^{\text{out}}} w_{ij}(\widehat{x}_i - \widehat{x}_j). \tag{51}$$

Conveniently, the closed-loop system dynamics is still given by (30) where the only difference now is that the Laplacian L is no longer symmetric,

$$\dot{x}(t) = -L\widehat{x}(t) = -L(x(t) + e(t)). \tag{52}$$

The problem can now be formalized as follows.

Problem 3.9 (Decentralized event-triggered coordination on directed graphs) *Given a weight-balanced communication graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (52), find an event-trigger $f_i(\cdot)$ for each agent i that is locally computable such that the sequences of times $\{t_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}}$ ensures multi-agent average consensus (12) is achieved.*

Note that Problem 3.9 is identical to Problem 2.5 except we now consider a directed (balanced) graph rather than an undirected one. Just as we did in solving Problems 2.3 and 2.5, let us again consider the Lyapunov function

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

Given the closed-loop dynamics (52), we have

$$\dot{V} = x^T L \dot{x} = -x^T L L(x + e).$$

Unfortunately L no longer being symmetric causes a serious problem because we cannot expand out this equation in a way that does not include the in-neighbors of a given agent i. As a result, using this Lyapunov function, we are not able to find a local triggering function f_i for agent i that only depends on the information actually available to it (its own state and the last broadcast state of its out-neighbors.)

Instead, let us consider the other Lyapunov function (34)

$$V(x) = \frac{1}{2}(x - \bar{x}\mathbf{1})^T(x - \bar{x}\mathbf{1}), \tag{53}$$

where $\bar{x} = \frac{1}{N}\sum_{i=1}^N x_i(0)$ is the average of all initial conditions. Then, given the closed-loop dynamics (52), we have

$$\dot{V} = x^T\dot{x} - \bar{x}\mathbf{1}^T\dot{x} = -x^T L\widehat{x} - \bar{x}\mathbf{1}^T L\widehat{x} = -x^T L\widehat{x},$$

where we have used the fact the graph is weight-balanced in the last equality.

Remarkably, a similar analysis to that used in the proof of Lemma 3.2 holds (after replacing $\mathcal{N}_i$ with $\mathcal{N}_i^{\text{out}}$ and explicitly considering weights) to yield the following bound.

Lemma 3.10 ((Nowzari and Cortés, 2016)) *Given* $V(x) = \frac{1}{2}(x - \bar{x}\mathbf{1})^T(x - \bar{x}\mathbf{1})$ *and the closed-loop dynamics* (52),

$$\dot{V} \leq \sum_{i=1}^N e_i^2 |\mathcal{N}_i^{out}| - \frac{1}{4} \sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\widehat{x}_i - \widehat{x}_j)^2.$$

Leveraging Lemma 3.10, it is easy to see that if we can enforce the error of all agents to satisfy

$$e_i^2 \leq \sigma_i \frac{1}{4|\mathcal{N}_i^{\text{out}}|} \sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\widehat{x}_i - \widehat{x}_j)^2, \qquad (54)$$

with $\sigma_i \in (0, 1)$ for all times, we have

$$\dot{V} \leq \sum_{i=1}^N \frac{\sigma_i - 1}{4} \sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\widehat{x}_i - \widehat{x}_j)^2, \qquad (55)$$

which is strictly negative for all $L\widehat{x} \neq 0$. Letting $\widehat{x}_{\mathcal{N}_i^{out}} = (\widehat{x}_i, \widehat{x}_{\mathcal{N}_i^{out}}))$, the following decentralized event-trigger ensures this is satisfied at all times.

Theorem 3.11 (Decentralized event-triggered coordination on directed graphs (Nowzari and Cortés, 2016)) *Given a weight-balanced communication graph* $\mathcal{G}_{comm}$ *and the closed-loop dynamics* (52), *if the event times of each agent* i *are determined as the times when*

$$f_i(e_i, \widehat{x}_{\mathcal{N}_i^{out}}) \triangleq \qquad (56)$$
$$e_i^2 - \sigma_i \frac{1}{4|\mathcal{N}_i^{out}|} \sum_{j \in \mathcal{N}_i^{out}} w_{ij}(\widehat{x}_i - \widehat{x}_j)^2 \geq 0,$$

then all non-Zeno trajectories of the system achieve multi-agent average consensus.

Remark 3.12 (Weight-balanced assumption) For implementations where the weights of the directed graph are design parameters, one can think of choosing them in a way that makes the given directed interaction topology weight-balanced. For cases where such choices can be made before the event-triggered consensus algorithm is implemented, the works (Gharesifard and Cortés, 2012; Rikos et al., 2014) present provably correct distributed strategies that, given a directed communication topology, allow a network of agents to find such weight edge assignments. •

In order to guarantee that the agents can converge exactly to the average of the initial agent states, Theorem 3.11 relies on L being weight-balanced. Consequently, it is unknown if and where the system will converge for directed graphs in general. If the directed graph contains a rooted spanning tree, agreement can be reached but it is not clear to where. In other cases it may not even be possible to reach agreement.

We have now discussed different types of static or fixed communication topologies, but one could easily imagine scenarios with both time-varying (Zhu et al., 2016) or state-dependent interaction graphs for which some modified algorithms may need to be developed. In the case of state-dependent interaction graphs, an additional challenge that must be addressed is how to preserve connectivity of the network while performing the primary consensus task (Yu and Antsaklis, 2012; Fan and Hu, 2015; Yi et al., 2017; Yu et al., 2016).

3.5 *Dynamics*

So far, we have only considered the simple single-integrator dynamics (10). While these simple dynamics are certainly useful for virtual states (e.g., a temperature estimate) or in demonstrating the ideas of event-triggered consensus in general, this might be too limited in cases where states correspond to physical quantities. In this section we begin by discussing double-integrator dynamics before moving onto general linear dynamics. We note here that as we generalize the dynamics we also consider goals beyond average consensus. More specifically, depending on the dynamics, static average consensus may not be possible, in which case we will instead aim for synchronization.

3.5.1 *Double-integrator systems*

Let us consider the case where the state of agent i is denoted by $x_i = (r_i, v_i) \in \mathbb{R} \times \mathbb{R}$ with double-integrator dynamics,

$$\dot{r}_i(t) = v_i(t), \qquad (57)$$
$$\dot{v}_i(t) = u_i(t).$$

Then, it is known that the distributed controller

$$u_i^*(x) = -\sum_{j \in \mathcal{N}_i} (r_i - r_j + \gamma(v_i - v_j)) \qquad (58)$$

with $\gamma > 0$ drives the states of all agents to a consensus trajectory (Ren and Atkins, 2007; Ren, 2008). Different from consensus with single-integrator dynamics, where the agents are to reach a steady state, the consensus problem of double-integrator systems is rather to synchronize the outputs. More specifically, we say that a system asymptotically achieves synchronization if

$$\lim_{t \to \infty} (x_i(t) - x_j(t)) = 0. \qquad (59)$$

Note in particular that different from the definition of consensus we used for single-integrator dynamics (12), we only require $x_i(t)$ and $x_j(t)$ to tend together as time goes to infinity rather than both going to the same stationary point. This is formalized in Theorem 3.13.

Theorem 3.13 (Continuous controller (double-integrators) (Ren, 2008)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the dynamics (57), if all agents implement the control law (58), then the system asymptotically achieves synchronization (59).*

Implementing (58) requires agents to have continuous information about one another. Thus, we are interested in an event-triggered implementation of the ideal control law (58) to relax this requirement. Letting $\{t_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}}$ be the sequence of event-times for agent i, we let

$$\widehat{x}_i(t) = x_i(t_\ell^i) = (r_i(t_\ell^i), v_i(t_\ell^i)) \text{ for } t \in [t_\ell^i, t_{\ell+1}^i) \quad (60)$$

be the last broadcast state of agent i. Then, at any given time t, agent i only has access to the last broadcast state $\widehat{x}_j(t)$ of its neighbors $j \in \mathcal{N}_i$ rather than exact states $x_j(t)$.

The distributed event-triggered controller we consider is then given by

$$u_i(t) = \quad (61)$$
$$-\sum_{j \in \mathcal{N}_i} (\widehat{r}_i + (t - t_\ell^i)\widehat{v}_i - \widehat{r}_j - (t - t_\ell^i)\widehat{v}_j + \gamma(\widehat{v}_i - \widehat{v}_j)),$$

for $t \in [t_\ell^i, t_{\ell+1}^i)$. It should be noted that this controller utilizes a first-order-hold (FOH) instead of a ZOH for the r_j components of the state. We now define two different errors

$$e_{r,i}(t) = \widehat{r}_i(t) + (t - t_\ell^i)\widehat{v}_i(t) - r_i(t),$$
$$e_{v,i}(t) = \widehat{v}_i(t) - v_i(t).$$

Then, defining the stack vector of the error $e = \begin{bmatrix} e_r^T & \gamma e_v^T \end{bmatrix}^T$, the closed-loop dynamics of the controller (61) is given by

$$\begin{bmatrix} \dot{r}(t) \\ \dot{v}(t) \end{bmatrix} = \Gamma \begin{bmatrix} r(t) \\ v(t) \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ L & L \end{bmatrix} e(t), \quad (62)$$

where

$$\Gamma = \begin{bmatrix} \mathbf{0}_{N \times N} & I_N \\ -L & -\gamma L \end{bmatrix}. \quad (63)$$

The problem can now be formalized as follows.

Problem 3.14 (Decentralized event-triggered coordination with double-integrator dynamics) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (62), find an event-trigger $f_i(\cdot)$ for each agent that is locally computable such that the sequences of times $\{t_\ell^i\}_{\ell \in \mathbb{Z}_{\geq 0}}$ ensures synchronization (59) is achieved.*

This problem is first addressed in (Xie et al., 2015; Cao et al., 2014) where state-dependent triggering rules are developed; however, these algorithms rely on continuous state information about neighbors. The first work we are aware of that solves Problem 3.14 where information is only shared at event times is (Seyboth et al., 2013), which proposes the time-dependent triggering threshold presented next.

Theorem 3.15 (Decentralized event-triggered coordination with double-integrator dynamics (Seyboth et al., 2013)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (62), if the event times of each agent i are determined as the times when*

$$f_i(e_i, t) \triangleq \left\| \begin{bmatrix} e_{r,i} \\ \gamma e_{v,i} \end{bmatrix} \right\| - (c_0 + c_1 e^{-\alpha t}) = 0, \quad (64)$$

with constants $c_0, c_1 \geq 0$ and $c_0 + c_1 > 0$, then all non-Zeno trajectories of the system reaches a neighborhood of consensus upper-bounded by

$$r = \|L\|\sqrt{2N} c_0 c_V / \lambda_3(\Gamma),$$

where $c_V > 0$ is related to the graph L. Moreover, if $c_0 > 0$ or $0 < \alpha < \lambda_3(\Gamma)$, then the closed-loop system does not exhibit Zeno behavior.

Similar to the result of Theorem 3.6, this solution uses a triggering function whose threshold depends on time rather than state. However, here we have also considered the use of a first-order holder controller between events. More specifically, this problem-solution pair has double-integrator dynamics, an undirected interaction graph, events that trigger broadcasts and control updates, triggers that are evaluated continuously, and trigger thresholds that are time-dependent.

Also similar to the algorithm presented in Theorem 3.6, this solution has design parameters c_0, c_1, and α that can be tuned to balance performance and efficiency. However, there are also limits to how these parameters can be tuned. In particular, we recall that there are only two ways to guarantee Zeno behavior. The first is to set $c_0 > 0$, but this sacrifices being able to guarantee convergence all the way to the exact consensus state. The second is to set $\alpha < \lambda_3(\Gamma)$, but this requires global knowledge of the entire network structure to be able to compute.

We have only considered undirected topologies here there are indeed works that have considered directed topologies as well. For directed topologies, even when the graph contains a spanning tree, the ideal controller (58) (with continuous communication and actuation) is only guaranteed to drive the system to a consensus state if γ is sufficiently large. We omit the details and instead refer the interested reader to (Seyboth et al., 2013).

Other works addressing event-triggered consensus of double-integrator systems include (Yan et al., 2014; Xue and Hirche, 2013; Yin and Yue, 2013; Mu et al., 2015) with various modifications. The work by Xue and Hirche (2013) consider the case of heterogeneous communication networks. In this case the positions and velocities are shared using different communication graphs. The work Mu et al. (2015) discusses the leader-follower consensus problem with double-integrator systems and (Garcia et al., 2016) addresses decentralized event-triggered consensus of second-order systems in the presence of communication imperfections. In particular, this reference considered the presence of communication delays and packet dropouts using a broadcasting style of communication. Discrete-time systems have also been explored in this context and we refer the interested reader to (Chen and Hao, 2012; Zhu et al., 2017; Yin and Yue, 2013; Yin et al., 2013).

3.5.2 Linear systems

The event-triggered approach has also been extended to consider more general dynamics than double-integrator models. Here we discuss in detail the synchronization problem for a homogeneous group of N agents or subsystems with linear dynamics.

Letting $x_i \in \mathbb{R}^n$ denote the state of agent i, we consider homogeneous linear dynamics

$$\dot{x}_i(t) = Ax_i(t) + Bu_i(t), \qquad (65)$$

where the pair (A, B) is controllable. The objective of the consensus or synchronization problem of (65) is to drive the state of each system x_i to a common response or trajectory, that is, the corresponding elements of each agent's state need to converge to a single trajectory.

The synchronization of multi-agent systems with linear dynamics and assuming continuous communication has been extensively studied, e.g., Li et al. (2010, 2011); Ma and Zhang (2010); Ren (2008); Scardovi and Sepulchre (2009); Su and Huang (2012); Tuna (2008, 2009); Seo et al. (2009). It is known (Ma and Zhang, 2010; Garcia et al., 2014) that the distributed controller

$$u_i(t) = cF \sum_{j \in \mathcal{N}_i} (x_i(t) - x_j(t)) \qquad (66)$$

with $c > 0$ ensures that the system achieves synchronization under some suitable conditions on the matrix F. This is formalized next.

Lemma 3.16 ((Garcia et al., 2014)) *Given the dynamics* (65), *if all agents implement the control law* (66) *and the matrices* $(A + c\lambda_j(L)BF)$ *are Hurwitz for* $j = 2, 3, \ldots, N$, *then the system asymptotically achieves synchronization.*

Unfortunately, the condition of Lemma 3.16 requires checking several matrices that are functions of every

nonzero eigenvalue of the Laplacian L of the communication graph. This is a difficult condition to check in general as it requires knowledge of all eigenvalues of L in order to compute the eigenvalues of the matrices $(A + c\lambda_j(L)BF)$, for $j = 2, 3, \ldots, N$. Less restrictive conditions involve the design of the consensus protocol parameter using only the smallest non-zero eigenvalue of the Laplacian matrix, λ_2, or an *estimate* of such eigenvalue. The following result gives one way of designing the controller (66) to satisfy the conditions of Lemma 3.16.

Theorem 3.17 (Continuous controller (linear dynamics)) *Given a connected, undirected graph* $\mathcal{G}_{comm}$ *and* (A, B) *controllable, if all agents implement the protocol* (66), *with*

$$F = -B^T P, \qquad (67)$$
$$c \geq 1/\lambda_2, \qquad (68)$$

where P *is the unique solution to*

$$PA + A^T P - 2PBB^T P + 2\alpha P < 0, \qquad (69)$$

then the system ensures synchronization (59) *is achieved.*

Next, we turn our attention to seek event-triggered implementations of the ideal control law (66). One of the earliest works to address this problem is (Zhu et al., 2014), which considers a digital implementation of the ideal controller (66)

$$u_i(t) = cF \sum_{j \in \mathcal{N}_i} (\widehat{x}_i(t) - \widehat{x}_j(t)), \qquad (70)$$

where

$$\widehat{x}_i(t) = x_i(t_\ell^i) \text{ for } t \in [t_\ell^i, t_{\ell+1}^i) \qquad (71)$$

denotes the state of agent i at its last event time. Zhu et al. (2014) then propose a simple constant threshold event-triggering algorithm where agent i broadcasts its state to its neighbors whenever its error $e_i(t) = \widehat{x}_i(t) - x_i(t)$ exceeds a fixed threshold.

Unfortunately, this algorithm faces some issues resulting from the fact that the closed-loop system may actually be unstable. More specifically, the algorithm proposed in (Zhu et al., 2014) generally provides poor performance in terms of reducing control updates and may lead to Zeno behavior when the agents have unstable dynamics, i.e., when one or more eigenvalues of the matrix A have positive real parts. Under this scenario the exponential divergence of the states of the agents causes the error e_i to grow greater than the fixed threshold used in Zhu et al. (2014) faster and faster ultimately leading to the undesired Zeno behavior.

The event-triggered consensus of linear systems using ZOH implementations was also addressed in Guo et al.

(2014). In this reference the agents measure their state and evaluate their event-triggered conditions periodically, at every h time units. However, the convergence conditions expressed in Guo et al. (2014) require explicit knowledge of the Laplacian matrix, which is an impediment for decentralized implementation.

The early work by Liu et al. (2012) provides an event-triggered strategy to reduce communication of a class of linear systems without explicitly addressing Zeno behavior. The works Demir and Lunze (2012) and Demir and Lunze (2012) study event-triggered synchronization of linear systems using a ZOH implementation with constant thresholds. The work Zhang et al. (2014) uses a ZOH method but restricts only actuation updates, while continuous communication is still required for the agents to determine the triggering instants. The work Zhou et al. (2015) addresses leader-follower consensus problem of linear systems but, similar to the previous reference, the event conditions require continuous state information from neighbors and from the leader, which limits the application of this approach for reducing communication frequency.

Fortunately, some new frameworks and algorithms have recently been developed to overcome this problem (Mu et al., 2015; Ding et al., 2015; Yang et al., 2014, 2015, 2016; Garcia et al., 2014; Liu et al., 2015; Garcia et al., 2017; Zhang et al., 2014; De Persis, 2013). Recent contributions have relied on model-based or estimation approaches in order to address event-triggered coordination, and to improve the performance of the multi-agent coordination system in terms of asymptotic convergence and reduction of generated events. Some of these approaches rely on both, sensing the states of neighbors and transmitting the control inputs, while other approaches only assume broadcasting capabilities.

More specifically, we now make one big change to the definition of $\widehat{x}_i(t)$. Until now we have always treated $\widehat{x}_i(t)$ as a piece-wise constant value that only changed when agent i triggered an event. Instead, with a slight abuse of notation, we now consider it as a time-varying estimate of the state of agent i with dynamics

$$\dot{\widehat{x}}_i(t) = A\widehat{x}_i(t) + B\widehat{u}_i(t), \tag{72}$$

where

$$\widehat{u}_i(t) = u_i(t_\ell^i) \text{ for } t \in [t_\ell^i, t_{\ell+1}^i) \tag{73}$$

is the control input used by agent i at its last event time. Then, when an event is triggered by agent i at time t_ℓ^i, it sends its control input $u_i(t_\ell^i)$ in addition to its current state $x_i(t_\ell^i)$. With this information, any neighbor $j \in \mathcal{N}_i$ can then propagate the estimate (72) forward in time.

Given these new model-predictive estimates, we redefine the control law in the same way as (70) but with $\widehat{x}_j(t)$ now given by (72), rather than being piece-wise constant. The closed-loop system dynamics of (65) with these control inputs is then given by

$$\dot{x}(t) = I_N \otimes Ax(t) + cL \otimes BFx(t) + cL \otimes BFe(t). \tag{74}$$

The problem can now be formalized as follows.

Problem 3.18 (Decentralized event-triggered coordination with linear dynamics) *Given the closed-loop dynamics (74), find an event-trigger $f_i(\cdot)$ for each agent that is locally computable such that the sequences of times $\{t_\ell^i\}_{\ell\in\mathbb{Z}_{\geq 0}}$ ensures synchronization (59) is achieved.*

Leveraging the result of Theorem 3.17, a state-dependent triggering rule to solve Problem 3.18 is proposed in (Garcia et al., 2014) based on the Lyapunov function $V = x^T\mathcal{L}x$, where $\mathcal{L} = L \otimes P$ and P is defined by (69). This result is formalized next.

Theorem 3.19 (Decentralized (state-dependent) event-triggered coordination with linear dynamics (Garcia et al., 2014)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (74), if the event times of each agent i are determined as the times when*

$$f_i(e_i, \widehat{x}_{\mathcal{N}_i}) \triangleq \delta_i - \sigma_i\widehat{z}_i^T\Theta_i\widehat{z}_i \geq 0, \tag{75}$$

where

$$\widehat{z}_i = \sum_{j\in\mathcal{N}_i} (\widehat{x}_i - \widehat{x}_j),$$
$$\Theta_i = (2c_2 - b_i|\mathcal{N}_i|(c_2 - c_1))PBB^TP,$$
$$\delta_i = 2(c_2 - c_1)|\mathcal{N}_i|\widehat{z}_i^TPBB^TPe_i + |\mathcal{N}_i|e_i^TPBB^TPe_i$$
$$\times \left[2c_1|\mathcal{N}_i|(1 + b_i) + \frac{c_2 - c_1}{b_i} + c_1(N - 1)\left(b_i + \frac{3}{b_i}\right)\right],$$

with constants $c_1 \geq 1/\lambda_2$, $c_2 > 0$, and $0 < b_i < \frac{2c_2}{|\mathcal{N}_i|(c_2 - c_1)}$ for $c_2 > c_1$, or $b_i > 0$ otherwise, then all non-Zeno trajectories of the system asymptotically achieve synchronization.

Theorem 3.19 guarantees asymptotic synchronization of the agents with linear dynamics along all its non-Zeno trajectories. Unfortunately, this result does not guarantee the exclusion of Zeno behavior. Following our discussion in Section 2.1.1, there are now several methods that can be used to address this issue. For instance, in (Garcia et al., 2014) a small fixed parameter is included in the trigger function (75) to avoid Zeno behaviors. While the modified algorithm is able to ensure Zeno behavior do not occur, the price to pay is that it can no longer guarantee convergence exactly to a synchronized state, but rather to a neighborhood around it. We omit the details but refer the interested reader to (Garcia et al., 2014).

For completeness, the next result from (Garcia et al., 2017) also solves Problem 3.18 using a time-dependent

triggering function rather than a state-dependent one as in Theorem 3.19.

Theorem 3.20 (Decentralized (time-dependent) event-triggered coordination with linear dynamics (Garcia et al., 2017)) *Given a connected, undirected graph $\mathcal{G}_{comm}$ and the closed-loop dynamics (74), if the event times of each agent i are determined as the times when*

$$f_i(e_i, t) \triangleq \|e_i\| - c_1 e^{-\alpha t} = 0,$$

with constants $c_1, \alpha > 0$, then all non-Zeno trajectories of the system asymptotically achieve synchronization. Moreover, there exists λ^* such that for $\alpha < \lambda^*$, the closed-loop system does not exhibit Zeno behavior.

Similar to the result of Theorem 3.6, Zeno behavior can be avoided if α is chosen small enough, where the critical value λ^* again depends on global information. The interested reader is referred to (Garcia et al., 2017). Similar problems in the presence of input saturation are considered in (Liu et al., 2016; Zhou et al., 2016).

We stop our discussion here at homogeneous agents with linear dynamics. Indeed there are also some relevant works that address the case of heterogeneous agents with linear dynamics (Zhou et al., 2017) as well. Additionally, recent works have also considered nonlinear dynamics but we omit the details and instead refer the interested reader to the works (Xiuxia and Dong, 2013; Zhang et al., 2015; Li et al., 2016,; Hu and Cao, 2017; Luzza et al., 2016). Instead, some works consider nonlinear control inputs with simpler dynamics to achieve finite-time or fast consensus (Guo and Dimarogonas, 2013; Zhang et al., 2015).

3.6 Uncertainty

Throughout this article we have assumed there are no disturbances or uncertainties of any kind, which clearly idealizes many instances of the problems we are interested in. Consequently, it is clearly important to determine how robust the algorithms we have discussed so far are in the presence of different sources of uncertainty, and how the algorithms might need to be modified to accommodate them. While we do not go into the same level of detail for these algorithms as we have done in the rest of the article, we provide a brief discussion of the technical issues and challenges on this front.

As discussed in Section 2.1.1, guaranteeing the existence of a positive MIET is crucial in ensuring that a proposed solution can actually be implemented. Interestingly, even if an event-triggered controller guarantees a positive MIET when disturbances do not exist, it is possible that arbitrarily small disturbances are enough to void this guarantee. The robustness of the existence of a positive MIET is therefore a critical property. We refer to Borgers and Heemels (2014) for a detailed discussion on the notion of robustness of event-triggered con-

trollers in the presence of disturbances and, in particular, the notion of *local event-separation* (guaranteeing non-Zeno executions when disturbances are not present) and the stronger notions of *semi-global* and *global event-separation* (with event-triggered controllers being robust against disturbances). The implementation details when designing event-triggered communication and control algorithms highlight the importance of this topic. While this article has stopped just short of this technical discussion, future works on these topics should be mindful of these implementation details to ensure the solutions can be practically implemented.

3.6.1 Quantization

Beyond state disturbances, imperfect wireless communication mechanisms present an additional set of challenges. As a majority of this article assumes messages are shared wirelessly, quantization of transmitted information is a natural issue that must be addressed. This problem was first studied by Garcia et al. (2013) using uniform quantizers. In general, an algorithm which would guarantee asymptotic convergence to the global initial average when non-quantized information is transmitted by each agent, is instead only able to converge to a bounded region around the initial average in the case where uniform quantizers are implemented at each node. This type of result is also known as *practical consensus*. The states of the agents satisfy $\lim_{t \to \infty} \|x_i(t) - \bar{x}\| \leq c$, where c is a constant which depends on the size of the quantization step and $\bar{x} = \frac{1}{N} \sum_{i=0}^{N} x_i(0)$.

The more recent work (Zhang et al., 2015) considers both uniform and logarithm quantizers. In the case of logarithmic quantizers, asymptotic convergence to the initial average is still not guaranteed in general. The difference between any two states is still bounded, but now this bound also depends on the value of the initial average. For instance, if the initial average happens to be equal to zero, then asymptotic convergence to the initial average is achieved. Compared to uniform quantizers, the use of logarithmic quantizers has been of significant advantage in stabilization problems since the quantization error diminishes as the signal to be quantized tends to zero. However, in consensus problems of single-integrator systems, the steady state value of the overall system is in general not equal to zero. Hence, it is expected that the results shown by Zhang et al. (2015) do not guarantee asymptotic convergence and the bounds on the state disagreement depend on the value of the initial average. Some more recent works that study further extensions such as dynamic quantization or self-triggered mechanisms are proposed in (De Persis and Frasca, 2013; Li et al., 2016; Yi et al., 2016; Senejohnny et al., 2018).

3.6.2 Communication delays and packet dropouts

In addition to quantization, specifically considering wirelessly networked systems introduces many new sources of uncertainty in the form of delays and packet drops. Consequently, some of these issues have already been studied

when the algorithms were first developed. Early in the development of event-triggered consensus algorithms, (Seyboth et al., 2013) analyzes the presence of communication delays in the consensus of single-integrator systems. It was shown in that reference that the closed-loop overall system using event-triggered controllers is Input-to-State Stable (ISS) with respect to the state errors introduced by the event-triggered controllers and practical consensus was demonstrated in the presence of delays bounded by a function of the largest eigenvalue of the graph Laplacian. Consensus of discrete-time single-integrator systems with communication delays was studied in (Li et al., 2014). The work (Garcia et al., 2016) provides an approach for consensus of double-integrator systems using a time-dependent threshold for systems with non-consistent packet dropouts and delays. Non-consistent packet dropouts means that a packet of information broadcasted by a given agent i may be received by all, some, or none of the intended recipients j such that $i \in \mathcal{N}_j$. Similarly, the communication delay associated to a given broadcast message can be different in general to every receiving agent, given that the message is successfully received. This is studied in Dolk et al. (2017) for general linear systems where a dynamic event-triggered coordination strategy is developed that guarantees average consensus but requires global information to determine whether the system will converge or not.

4 Applications of Event-Triggered Consensus

Here we provide examples of both direct and indirect applications of the various algorithms discussed in the article. Our presentation is not meant to be exhaustive of every area, but rather serve as an initial point of reference for interested readers that seek to employ the results presented above. Our focus is not on multi-agent consensus per se, but in works that have addressed the need for event-triggered coordination in problems that involve consensus.

Formation control. The connections between consensus and some practical problems such as formation control of groups of vehicles have already been long established before event-triggered ideas became popular (Ren and Beard, 2008; Ren and Atkins, 2007). This problem is relatively straightforward in the centralized case where all agents know the desired shape and location of the final formation. However, in the decentralized version of the problem, each vehicle may know the desired formation shape but the location of the formation needs to be negotiated and agreed upon by the distributed agents (Ren et al., 2007).

Letting $p_i \in \mathbb{R}^n$ represent the position of agent i in some space, the goal is to drive all agents $i \in \{1, \ldots, N\}$ such that $p_i(t) \to \bar{p} + b_i$, where $\bar{p}$ is the average position of all agents and b_i represents the desired relative displacement of agent i defining its position in the formation with respect to the center. Since the vectors b_i are constant, the

agents then simply need to perform average consensus on the virtual state $x_i = p_i - b_i$ to agree upon the center of the formation. Early works generally assumed that agents have continuous, or at least periodic, access to information about their neighbors. Consequently, several groups of researchers have considered applying the event-triggered coordination ideas to consensus-based formation control algorithms to relax this requirement (Chu et al., 2018; Nowzari and Pappas, 2016; Adaldo et al., 2017).

Leader-tracking. Similar to formation control, another popular application is to actually try to directly control the entire group of agents using a small subset (or even just a singular) of agents to drive the rest of the group. In the leader-tracking (or leader-follower) problem there exists (at least) one particular agent, called the leader (or pinned agent), which acts independently from all other agents' states. The rest of the agents are referred to as the followers and they implement some form of consensus algorithm such that they essentially 'follow' the leader(s), perhaps while maintaining a specified formation.

For simplicity, consider a single leader with identity $i = 0$ which aims to lead a group of agents with identities $i = \{1, \ldots, N\}$. The leader is free to move or be controlled by a user. Not all agents will have access to direct information about the leaders' motion, and hence they implement average consensus to propagate it throughout the network and be able to follow the leader. Depending on constraints such as the maximum speed or acceleration of the leader, various results can be established regarding the global behavior and performance of the system. Many groups of researchers have looked at applying event-triggered coordination to many different variations of this leader-tracking problem. For instance, single- and double-integrator dynamics are considered in (Liu et al., 2016), homogeneous linear dynamics in (Cheng et al., 2014; Cheng and Ugrinovskii, 2016; Zhu and Jiang, 2015), heterogeneous linear dynamics in Garcia et al. (2017), and nonlinear dynamics in Adaldo et al. (2015); Zhang et al. (2015); Li et al. (2016). In addition to considering different types of dynamics, other groups have considered discrete-time systems (Chen et al., 2015). Finally, even more variations can be considered by imposing different types of constraints on the problems or solutions as discussed throughout this article. Examples include the addition of uncertainties/disturbances, specific goals of a network of leader-followers (e.g., containment control), dynamic topologies, and the different types of triggering mechanisms discussed in this article (Li et al., 2015, 2016; Xu et al., 2017; Mu et al., 2015).

Distributed estimation. A popular indirect application of consensus algorithms in general is distributed state estimation, see e.g. (Zou et al., 2017). Consensus protocols can then be used to allow distributed agents

to communicate and agree on a common state estimate; however, this generally assumes periodic communication among the agents. More specifically, distributed agents are sharing new samples with one another at all times to maintain both a good and consistent estimate of the quantity of interest. However, this can be wasteful in general, especially if new samples being shared are not providing much new information. Instead, applying event-triggered coordination to these algorithms can help reduce the amount of communication required by a network to maintain a state estimate. In (Battistelli et al., 2016), each node of the communication network implements a local Kalman filter and shares this information with neighbors to achieve consensus. The implementation of event-triggered communication strategies restricts inter-agent communication. In this case, it is necessary to evaluate the difference between probability density functions (PDF) at different time instants. This is achieved by applying the Kullback-Leibler divergence metric to the current local PDF and the last transmitted PDF. The result of this operation is compared against a time-dependent threshold of the form $c_0 + c_1 e^{-\alpha t}$. Similar to the ideas of this article, the agents only share their new information with neighbors if the new information is different enough from what is currently estimated according to the time-varying threshold. Ouimet et al. (2018) pursue similar ideas in the context of cooperative localization, with agents only sending measurements to neighbors when the expected innovation for state estimation is high. Instead, the work (Liu et al., 2015) considers a simpler positive threshold parameter to dictate events.

Clock synchronization. Clock synchronization is a particular problem of interest requiring agents in a network to synchronize their imperfect clocks via communication. This problem has been addressed in the past using consensus algorithms and assuming continuous or periodic communication such as in (Schenato and Fiorentin, 2011) and (Carli and Zampieri, 2014). Specifically, letting $t \in \mathbb{R}_{\geq 0}$ represent the true global time, we define the local clock time for a distributed agent $i \in \{1, \ldots, N\}$ as

$$s_i(t) = a_i t + b_i,$$

where $a_i > 0$ and $b_i \in \mathbb{R}$ represent the unknown drift and bias of clock i, respectively. In order to synchronize the clocks with varying and unknown drifts and biases, we define a virtual clock

$$T_i(t) = \alpha_i(s_i(t))s_i(t) + \beta_i(s_i(t)),$$

where α_i and β_i are the controlled drift and bias of node i, respectively, whose dynamics are to be designed. The goal is then to perform consensus on the virtual clock variables T_i such that $\|T_i - T_j\| \to 0$ for all pairs of agents by constantly sharing these values as they update their controlled drift and bias. Instead, recent works have studied applying event-triggered coordination to these problems

to reduce the communication required by the agents to synchronize their clocks (Kadowaki and Ishii, 2015; Chen et al., 2015; Garcia et al., 2017).

Distributed optimization. In many applications, it is of interest to solve optimization problems with separable objective functions of the form

$$f(x) = \sum_{i=1}^{N} f_i(x, y_i),$$

subject to various equality and/or inequality constraints, where x is some variable or state, y_i is some data or measurement local to agent i, and f_i is a function known to agent i, see (Wan and Lemmon, 2009; Nedić, 2015). Regardless of the objective or constraints, a popular approach to solve this problem is by having agents constantly share information (either their local state x_i or their about the global solution x) with their neighbors to cooperatively optimize the function while ensuring the global constraints are satisfied. Instead, several groups of researchers have studied the application of event-triggered coordination to these sorts of problems where the agents' decisions are coupled through the objective function, the constraints, or both. More specifically, agents employ event-triggered communication to trade computation for reduced overall information exchange in finding a solution of the optimization problem (Zhong and Cassandras, 2010; Kia et al., 2015; Richert and Cortés, 2016; Liu and Chen, 2016; Chen and Ren, 2016).

5 A Look Beyond

We have mainly focused on providing an introduction and survey of event-triggered coordination strategies applied specifically to the multi-agent consensus problem, but this article can also be viewed as a broader tutorial on how to apply event-triggered coordination to networked systems in general by thinking of the consensus problem as a specific case study. In fact, the work on event-triggered coordination for multi-agent consensus and networked systems in general has evolved in parallel over the last decade. For a tutorial introduction on event-triggered control of single plant systems we refer to (Heemels et al., 2012). Instead, we provide here a brief overview of distributed event-triggered control for networked systems in a more general context than just multi-agent consensus. Note that this is not meant to be a comprehensive survey of event-triggered coordination for networked systems in general but rather points to various other classes of networked problems that also benefit from ideas in event-triggered control; for this we refer the interested reader to (Tolić and Hirche, 2017). In this section, we provide a brief discussion on distributed event-triggered coordination of more general networked systems and finally take a look forward, identifying possible avenues for future research.

5.1 Distributed event-triggered control and stabilization

We consider here the distributed control and stabilization of interconnected systems. These interconnections not only capture the ability to share information, but also might represent inherent dynamic coupling between the subsystems. The overall objective is the stabilization of the overall system through coordinated communication and control. Initial work formally introducing and addressing this problem was presented in (Wang and Lemmon, 2008,), where each subsystem's dynamics are described by

$$\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) + \sum_{j \in \mathcal{N}_i} A_{ij} x_j(t), \qquad (76)$$

for $i = \{1, \ldots, N\}$, where $x_i \in \mathbb{R}^{n_i}$ and $u_i \in \mathbb{R}^{m_i}$ represent the state and the control input of subsystem i, respectively. The matrices $A_i \in \mathbb{R}^{n_i \times n_i}$, $B_i \in \mathbb{R}^{n_i \times m_i}$, $A_{ij} \in \mathbb{R}^{n_i \times n_j}$, represent the state, input, and coupling matrices, respectively. The starting point is the availability of ideal controllers

$$u_i(t) = K_i x_i + \sum_{j \in \mathcal{N}_i} L_{ij} x_j,$$

where the decoupling gains L_{ij} can be chosen such that $B_i L_{ij} = -A_{ij}$ and K_i has been designed such $x = 0$ is asymptotically stable. However, since implementing this solution requires each subsystem to have continuous access to the state of neighboring subsystems, the goal now is to instead design a distributed event-triggered coordination strategy to solve the problem.

Similar to our setup for consensus, each subsystem must now determine for itself when to broadcast its state to neighboring subsystems. Consequently, the actual control input of agent i is given by

$$u_i(t) = K_i \widehat{x}_i + \sum_{j \in \mathcal{N}_i} L_{ij} \widehat{x}_j, \qquad (77)$$

where $\widehat{x}_i(t) = x_i(t_\ell^i)$ for $t \in [t_\ell^i, t_{\ell+1}^i)$ is the last broadcast state of subsystem i at any given time $t \in \mathbb{R}_{\geq 0}$. Just as in the consensus case, we define $e_i(t) = \widehat{x}_i(t) - x_i(t)$ as the error between the last broadcast state of subsystem i and its current state. Following our discussion in Section 2.1, we are interested in designing a distributed event-triggering condition $f_i(\cdot)$ such that the closed-loop dynamics of (76) with control input (77) guarantees that the system still asymptotically converges.

To achieve this, under suitable conditions, Wang and Lemmon (2008) propose the trigger

$$f_i(e_i, x_i) \triangleq \beta_i \|e_i\|^2 - \alpha_i \|x_i\|^2 \leq 0,$$

for some positive constants β_i and α_i. This work was extended in (Wang and Lemmon, 2011; De Persis et al.,

2013) to nonlinear systems of the form

$$\dot{x}_i(t) = F_i(x_i, \{x_j\}_{j \in \mathcal{N}_i}, u_i).$$

Later works (Guinaldo et al., 2012, 2013) have further extended these results to consider robustness issues such as the presence of network delays and packet dropouts in the networked stabilization problem.

5.2 Decentralized event-triggered control over wireless sensor/actuator networks

A similar problem related to distributed event-triggered control was presented in (Mazo Jr. and Tabuada, 2011) and (Postoyan et al., 2011), where the states of a nonlinear system

$$\dot{x}(t) = F(x(t), u(t))$$

are measured by individual, decentralized sensors. More specifically, different components of the state of a single system is sampled by different sensors. As usual, the starting point here is the assumption that a stabilizing controller $u(t) = K(x(t))$ exists and is known. The actually used control input is then of the form $u(t) = K(\widehat{x}(t))$, where $\widehat{x}(t)$ represents the most up-to-date information about the true state $x(t)$. The goal is then for any given sensor to determine conditions for sharing its information based only on its local measurements in order to guarantee stabilization of the overall system.

5.3 Distributed event-triggered control for output feedback systems

Throughout this article we have assumed that individual agents or subsystems have access to their own exact state. Instead, many works (Tallapragada and Chopra, 2012; Donkers and Heemels, 2012; Heemels et al., 2013; Tallapragada and Chopra, 2014; Forni et al., 2014; Mazo Jr. and Cao, 2014; Yu and Antsaklis, 2012; Tolić and Fierrol, 2013; Zhang et al., 2014; Cui et al., 2016; Hu and Liu, 2017; Yu and Antsaklis, 2014; Hu et al., 2016; Liu et al., 2014; Zhou et al., 2016; Dolk et al., 2017; Hu et al., 2017; Abdelrahim et al., 2017; Mahmoud et al., 2016) consider the case of systems with output feedback, e.g.,

$$\begin{aligned} \dot{x}_i(t) &= A_i x_i(t) + B_i u_i(t), \\ y_i(t) &= C_i x_i(t), \end{aligned}$$

where each element of the output of the system is sampled by a different sensor. Depending on the specific application, the goals are similar to the problems we have already discussed; e.g., synchronizing the outputs (rather than the states) or guaranteeing asymptotic convergence using only an output feedback controller (rather than state feedback).

Similar to the technical issues we come across regarding Zeno executions for the multi-agent consensus problem, these types of problems exhibit the exact same types of

concerns due to the distributed and partial information available to the sensors Donkers and Heemels (2012).

5.4   Future Outlook

This article has provided an introduction to the field of event-triggered consensus. As consensus problems are widespread in terms of networked systems, the exposition has brought up many of the challenges and tools that are not specific to this particular problem, but underlie network coordination tasks in general. In particular, we have highlighted the importance of designing distributed event-triggers that still guarantee global properties and the technical difficulties that come with ensuring stability of the resulting asynchronous executions. The focus on consensus has enabled us to illustrate the motivation, the design methods, and the technical challenges that arise in carrying this through. Here, we offer some thoughts on additional lines of research that we believe are worthy of further exploration in the future.

Stochastic event-triggers.   Through this article we have focused only on deterministic triggering strategies that determine precise times at which events should be triggered. Alternatively, a number of works (Han et al., 2015; Brunner et al., 2018; Antunes, 2013) consider the design of events that are triggered stochastically. While this area is still in under development, there seem to be some benefits that may be applicable to both the multi-agent consensus problem and networked systems more generally. For instance,

(i) it may be easier to compare the performance of different algorithms against one another by considering average quantities or rates of transmissions rather than exact trajectories;

(ii) they require less precise specifications, in that triggers can be more loosely defined since it is not critical that an event be triggered at an exact specified time; and

(iii) it may be easier to ensure non-Zeno executions due to the less precise scheduling constraints.

We are not yet aware of any works that study this for networked systems but believe this to be a worthwhile avenue for future exploration. It should be noted that we view this differently from *stochastic event-triggered control*, in which the idea of event-triggered control updates is applied to stochastic optimal controllers (Xu and Hespanha, 2005; Rabi and Baras, 2007; Molin and Hirche, 2009, 2010).

Dynamic average consensus.   Throughout the article we have generally assumed that the final convergence value of the entire network depends on some function of the initial states of all agents. However, we can also imagine extensions of these ideas to dynamic average consensus problems, in which the group of agents is expected to track some time-varying quantity about all the agents.

For instance, the agents may be trying to agree on the average temperature in a room; but if the temperature in the room is changing, we need a dynamic consensus algorithm to track the average temperate of the room in real time. Of course of the temperature is changing too quickly, then we cannot expect accurate tracking of it in a distributed manner; however, depending on assumptions on how quickly the quantity of interest is changing, different bounds on the tracking error can be provided using dynamic consensus algorithms (Freeman et al., 2006; Spanos et al., 2005; Kia et al., 2015). In fact, the dynamic average consenus problem is a natural generalization of the static average consensus problem with applications in a wide variety of areas.

Existing dynamic average consensus algorithms generally require continuous or periodic communication between agents in order to adequately track the quantity of interest. Consequently, Kia et al. (2015) considers the application of event-triggered coordination to the dynamic average consensus problem to relax this requirement. Similar to some algorithms presented in this article, Kia et al. (2015) presents event-triggered versions of these algorithms that determine when communication should occur according to some time- or state-varying thresholds over multi-agent networks with single integrator and dynamic topologies. Convergence is only guaranteed to within a neighborhood of the time-varying average signal.

Cloud-based event-triggered control.   Throughout this article we have assumed various forms of peer-to-peer communication; in all cases the messages were directly sent from one agent to another. Instead, some recent works consider scenarios where agents communicate indirectly through the use of the cloud (Adaldo et al., 2015; Bowman et al., 2016; Adaldo et al., 2017). More specifically, we have thus far assumed that when an agent decides to send information to another, it is *passively* received by the receiving agent. It is possible that the packet can be dropped or delayed, but in either case the receiving agent does not have an active part in receiving the message. Instead, using a cloud-based communication model, an agent i can only publish things intended for other agents to a cloud repository, rather than sending it directly to them. In this case, until a receiving agent j *actively* decides to connect to the cloud and download the information that is available, it will not even be aware of a pending message from the transmitting agent i. And in turn, when the message is received, it reveals information about the past state and plans of agent i, and likely only partial information about the present ones. This raises a plethora of interesting opportunities for the design of promises among agents, and the technical analysis of the resulting coordination algorithms, see also (Nowzari and Cortés, 2016). The use of the cloud also opens the possibility of network agents with limited capabilities taking advantage of high-performance computation capabilities to deal with complex dynamical processes.

Performance guarantees. Something conspicuously missing from the event-triggered consensus literature in particular and the event-triggered control literature in general are performance guarantees that quantify the benefit of this approach over time-triggered or periodic implementations. The self-tuning nature of event-triggered control, where events are *tuned* to the execution of the task at hand, makes it appealing at a conceptual and design level. Periodic control, by contrast, requires the a priori selection of stepsizes and the consideration of worst cases in doing so. Simulations have consistently shown the promise of event-triggered algorithms over periodic ones in many cases. Apart from some early work examining this issue Åström and Bernhardsson (1999, 2002), it is only recently that some works have establishes results along these lines for systems with a centralized controller or decision maker, see e.g., (Antunes and Heemels, 2014; Dolk et al., 2017; Khashooei et al., 2017, 2018; Ong and Cortés, 2018). There is also some preliminary work to apply this to network settings (Ramesh et al., 2016; Borgers et al., 2017; Heijmans et al., 2017), but this area as a whole is still largely incomplete. We expect such guarantees and characterizations of average communication rates to be increasingly important as event-triggered coordination algorithms gain further popularity.

Related to this is the need to have ways of comparing different event-triggered algorithms that successfully achieve the same task in order to understand which one is better and under what conditions. Even in this article we have presented several algorithms that solve the same problem, and even assume the same capabilities of the agents. For example, Theorems 3.4 and 3.5 both solve Problem 2.5 under the same assumptions of the agents' abilities and are able to provide the same guarantee: asymptotic convergence (including non-Zeno guarantees). However, this gives us no insight into the transient performance of these algorithms. Consequently, it is unclear which algorithm would be better to implement to solve a given problem. Ultimately, the type of algorithm we want to design and implement should be optimized for a certain task. For instance, in some cases it may be desirable to reduce the communication burden of the network as much as possible, but in other scenarios it may make more sense to use frequent communication to yield faster convergence. In any case, we are still in the need as a community for established metrics and methods for comparing different algorithms against one another to be able to optimize them to meet varying performance needs.

6 Conclusions

The application of event-triggered coordination to large-scale networks is currently surging in interest due to the rising ubiquity of interconnected cyber-physical systems. As the number of devices connected to a shared network grows larger than previously dealt with in the past, distributed time-triggered coordination strategies do not scale well. Such limitations require a rethinking of the periodic control paradigm towards opportunistic schemes, as the ones discussed in this article, that take full advantage of the knowledge of the agents, the environment, and the task to efficiently manage the available resources.

The work Åström and Bernhardsson (2002) concluded that

> "There are an increasing number of applications where the assumption of constant sampling rate is no longer valid, typical examples are multi-rate sampling and networked systems. Lebesgue sampling (or event-triggered sampling) may be a useful alternative...
>
> ...It would be very attractive to have a system theory similar to the one for periodic sampling."

We believe the field of event-triggered coordination is shaping up to be precisely the type of theory needed in addressing the various network problems that exist today, where aperiodic sampling, communication, and control should be viewed as an opportunity rather than a disturbance. We hope that researchers find this article useful in developing a better understanding of event-triggered coordination and designing other general networks that operate in an efficient and adaptive fashion.

Acknowledgments

References

[1] M. Abdelrahim, R. Postoyan, J. Daafouz, and D. Nešić. Robust event-triggered output feedback controllers for nonlinear systems. *Automatica*, 75: 96–108, 2017.

[2] A. Adaldo, F. Alderisio, D. Liuzza, G. Shi, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Event-triggered pinning control of switching networks. *IEEE Transactions on Control of Network Systems*, 2(2):204–213, 2015.

[3] 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*, pages 954–961, Linz, Austria, 2015.

[4] A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Multi-agent trajectory tracking with self-triggered cloud access. In *IEEE Conf. on Decision and Control*, pages 2207–2214, Las Vegas, NV, Dec. 2016.

[5] A. Adaldo, D. Liuzza, D. V. Dimarogonas, and K. H. Johansson. Cloud-supported formation control of second-order multi-agent systems. *IEEE Transactions on Control of Network Systems*, PP (99):1–1, 2017.

[6] A. Albert and R. Bosch. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. In *Embedded World*, pages 235–252, Nurnberg, Germany, 2004.

[7] A. Anta and P. Tabuada. To sample or not to sample: self-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, 55(9): 2030–2042, 2010.

[8] D. Antunes. Event-triggered control under Poisson events: the role of sporadicity. In *IFAC World Congress*, pages 269–276, Koblenz, Germany, 2013.

[9] D. Antunes and W. P. M. H. Heemels. Rollout event-triggered control: Beyond periodic control performance. *IEEE Transactions on Automatic Control*, 59(12):3296–3311, 2014.

[10] R. Aragues, G. Shi, D. V. Dimarogonas, C. Sagues, and K. H. Johansson. Distributed algebraic connectivity estimation for adaptive event-triggered consensus. In *American Control Conference*, pages 32–37, Montreal, Canada, 2012.

[11] K. E. Årzén. A simple event based PID controller. In *IFAC World Congress*, pages 423–428, Beijing, China, 1999.

[12] K. J. Åström and B. M. Bernhardsson. Comparison of periodic and event based sampling for first-order stochastic systems. In *IFAC Proceedings Volumes*, volume 32, pages 5006–5011, 1999.

[13] 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*, pages 2011–2016, Las Vegas, NV, Dec. 2002.

[14] K. J. Åström and B. Wittenmark. *Computer Controlled Systems*. Prentice Hall, 1997.

[15] G. Battistelli, L. Chisci, and D. Selvi. Energy-efficient distributed state estimation via event-triggered consensus on exponential families. In *American Control Conference*, pages 6387–6392, Boston, MA, 2016.

[16] G. Bekey and R. Tomovic. Sensitivity of discrete systems to variation of sampling interval. *IEEE Transactions on Automatic Control*, 11(2):284–287, 1966.

[17] J. Berneburg and C. Nowzari. Distributed dynamic event-triggered coordination with a designable minimum inter-event time. In *American Control Conference*, Philadelphia, PA, July 2019. To Appear.

[18] D. P. Borgers and W. P. M. H. Heemels. Event-separation properties of event-triggered control systems. *IEEE Transactions on Automatic Control*, 59(10):2644–2656, 2014.

[19] D. P. Borgers, R. Geiselhart, and W. P. M. H. Heemels. Tradeoffs between quality-of-control and quality-of-service in large-scale nonlinear networked control systems. *Nonlinear Analysis: Hybrid Systems*, 23:142–165, 2017.

[20] S. L. Bowman, C. Nowzari, and G. J. Pappas. Coordination of multi-agent systems via asynchronous cloud communication. In *IEEE Conf. on Decision and Control*, pages 2215–2220, Las Vegas, NV, Dec. 2016.

[21] R. W. Brockett. Minimum attention control. In

IEEE Conf. on Decision and Control, pages 2628–2632, San Diego, CA, 1997.

[22] F. D. Brunner, D. Antunes, and F. Allgower. Stochastic thresholds in event-triggered control: a consistent policy for quadratic control. *Automatica*, 89:376–381, 2018.

[23] M. Cao, F. Xiao, and L. Wang. Second-order leader-following consensus based on time and event hybrid-driven control. *Systems & Control Letters*, 74:90–97, 2014.

[24] M. Cao, F. Xiao, and L. Wang. Event-based second-order consensus control for multi-agent systems via synchronous periodic event detection. *IEEE Transactions on Automatic Control*, 60(9):2452–2457, 2015.

[25] R. Carli and S. Zampieri. Networked clock synchronization based on the second-order linear consensus algorithm. *IEEE Transactions on Automatic Control*, 59(2):409–422, 2014.

[26] M. Chen, L. Zhang, H. Su, and C. Li. Event-based synchronisation of linear discrete-time dynamical networks. *IET Control Theory & Applications*, 9 (5):755–765, 2015.

[27] W. Chen and W. Ren. Event-triggered zero-gradient-sum distributed consensus optimization over directed networks. *Automatica*, 65:90–97, 2016.

[28] X. Chen and F. Hao. Event-triggered average consensus control for discrete-time multi-agent systems. *IET Control Theory and Applications*, 6(16): 2493–2498, 2012.

[29] X. Chen, F. Hao, and A. Rahmani. Asynchronous decentralised event-triggered control of multi-agent systems. *International Journal of Control*, 87(10): 2130–2139, 2014.

[30] Z. Chen, D. Li, Y. Huang, and C. Tang. Event-triggered communication for distributed time synchronization in WSNs. In *Chinese Control Conference*, pages 7789–7794, Hangzhou, China, 2015.

[31] T. Cheng, Z. Kan, J. M. Shea, and W. E. Dixon. Decentralized event-triggered control for leader-follower consensus. In *IEEE Conf. on Decision and Control*, pages 1244–1249, 2014.

[32] Y. Cheng and V. Ugrinovskii. Event-triggered leader-following tracking control for multivariable multi-agent systems. *Automatica*, 70:204–210, 2016.

[33] X. Chu, Z. Peng, G. Wen, and A. Rahmani. Distributed formation tracking of multi-robot systems with nonholonomic constraint via event-triggered approach. *Neurocomputing*, 275:121–131, 2018.

[34] D. Ciscato and L. Martiani. On increasing sampling efficiency by adaptive sampling. *IEEE Transactions on Automatic Control*, 12(3):318, 1967.

[35] Y. Cui, M. Fei, and D. Du. Event-triggered cooperative compensation control for consensus of heterogeneous multi-agent systems. *IET Control Theory & Applications*, 10(13):1573–1582, 2016.

[36] C. De Persis. On self-triggered synchronization of

linear systems. In *IFAC Workshop on Distributed Estimation and Control in Networked Systems*, pages 247–252, Koblenz, Germany, Sept. 2013.

[37] C. De Persis and P. Frasca. Robust self-triggered coordination with ternary controllers. *IEEE Transactions on Automatic Control*, 58(12):3024–3038, 2013.

[38] C. De Persis and R. Postoyan. A Lyapunov redesign of coordination algorithms for cyber-physical systems. *IEEE Transactions on Automatic Control*, 62(2):808–823, 2017.

[39] C. De Persis, R. Sailer, and F. Wirth. Parsimonious event-triggered distributed control: a zeno free approach. *Automatica*, 49(7):2116–2124, 2013.

[40] O. Demir and J. Lunze. Cooperative control of multi-agent systems with event-based communication. In *American Control Conference*, pages 4504–4509, 2012.

[41] O. Demir and J. Lunze. Event-based synchronisation of multi-agent systems. In *IFAC Conference on Analysis and Design of Hybrid Systems*, pages 1–6, 2012.

[42] D. V. Dimarogonas and E. Frazzoli. Distributed event-triggered control strategies for multi-agent systems. In *Allerton Conf. on Communications, Control and Computing*, pages 906–910, Monticello, IL, Sept. 2009.

[43] D. V. Dimarogonas and K. H. Johansson. Event-triggered control for multi-agent systems. In *IEEE Conf. on Decision and Control*, pages 7131–7136, Shanghai, China, 2009.

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

[45] D. Ding, Z. Wang, B. Shen, and G. Wei. Event-triggered consensus control for discrete-time stochastic multi-agent systems: the input-to-state stability in probability. *Automatica*, 62:284–291, 2015.

[46] V. S. Dolk and W. P. M. H. Heemels. Dynamic event-triggered control under packet losses: The case with acknowledgements. In *Event-based Control, Communication, and Signal Processing*, pages 1–7, Krakow, Poland, 2015.

[47] V. S. Dolk, M. Abdelrahim, and W. P. M. H. Heemels. Event-triggered consensus seeking under non-uniform time-varying delays. *IFAC-PapersOnLine*, 51(1):10096–10101, 2017.

[48] V. S. Dolk, D. P. Borgers, and W. P. M. H. Heemels. Output-based and decentralized dynamic event-triggered control with guaranteed L_p-gain performance and zeno-freeness. *IEEE Transactions on Automatic Control*, 62(1):34–49, 2017.

[49] M. C. F. Donkers and W. P. M. H. Heemels. Output-based event-triggered control with guaranteed l-gain and improved and decentralized event-triggering. *IEEE Transactions on Automatic Control*, 57(6):1362–1376, 2012.

[50] G. Duan, F. Xiao, and L. Wang. Asynchronous periodic edge-event triggered control for double-integrator networks with communication time delays. *IEEE Transactions on Cybernetics*, 2017. To appear.

[51] Y. Fan and G. Hu. Connectivity-preserving rendezvous of multi-agent systems with event-triggered controllers. In *IEEE Conf. on Decision and Control*, pages 234–239, Osaka, Japan, 2015.

[52] Y. Fan, L. Liu, G. Feng, and Y. Wang. Self-triggered consensus for multi-agent systems with zeno-free triggers. *IEEE Transactions on Automatic Control*, 60(10):2779–2784, 2015.

[53] F. Forni, S. Galeani, D. Nešić, and L. Zaccarian. Event-triggered transmission for linear control over communication channels. *Automatica*, 50(2):490–498, 2014.

[54] G. Franklin, J. Powel, and A. Emami-Naeini. *Feedback Control of Dynamical Systems*. Prentice Hall, 2010.

[55] R. A. Freeman, P. Yang, and K. M. Lynch. Stability and convergence properties of dynamic average consensus estimators. In *IEEE Conf. on Decision and Control*, pages 398–403, San Diego, CA, 2006.

[56] Y. Fuan, Y. Yang, and Y. Zhang. Sampling-based event-triggered consensus for multi-agent systems. *Neurocomputing*, 191:141–147, 2016.

[57] E. Garcia, Y. Cao, H. Yu, P. Antsaklis, and D. Casbeer. Decentralised event-triggered cooperative control with limited communication. *International Journal of Control*, 86(9):1479–1488, 2013.

[58] E. Garcia, Y. Cao, and D. W. Casbeer. Decentralized event-triggered consensus with general linear dynamics. *Automatica*, 50(10):2633–2640, 2014.

[59] E. Garcia, Y. Cao, and D. W. Casbeer. Decentralized event-triggered consensus of double integrator multi-agent systems with packet losses and communication delays. *IET Control Theory and Applications*, 10(15):1835–1843, 2016.

[60] E. Garcia, Y. Cao, and D. W. Casbeer. Periodic event-triggered synchronization on linear multi-agent systems with communication delays. *IEEE Transactions on Automatic Control*, 62(1):366–371, 2017.

[61] E. Garcia, Y. Cao, and D. W. Casbeer. An event-triggered control approach for the leader tracking problem with heterogeneous agents. *International Journal of Control*, 2017. To appear.

[62] E. Garcia, S. Mou, Y. Cao, and D. W. Casbeer. An event-triggered consensus approach for distributed clock synchronization. In *American Control Conference*, 2017. To Appear.

[63] B. Gharesifard and J. Cortés. Distributed strategies for generating weight-balanced and doubly stochastic digraphs. *European Journal of Control*, 18(6):539–557, 2012.

[64] A. Girard. Dynamic triggering mechanisms for event-triggered control. *IEEE Transactions on Automatic Control*, 60(7):1992–1997, 2017.

[65] M. Guinaldo, D. Lehmann, J. Sanchez, S. Dormido, and K. H. Johansson. Distributed event-triggered control with network delays and packet losses. In *IEEE Conf. on Decision and Control*, pages 1–6, Maui, HI, 2012.

[66] M. Guinaldo, D. V. Dimarogonas, K. H. Johansson, J. Sanchez, and S. Dormido. Distributed event-based control strategies for interconnected linear systems. *IET Control Theory & Applications*, 7(6): 877–886, 2013.

[67] G. Guo, L. Ding, and Q. Han. A distributed event-triggered transmission strategy for sampled-data consensus of multi-agent systems. *Automatica*, 50 (5):1489–1496, 2014.

[68] M. Guo and D. V. Dimarogonas. Nonlinear consensus via continuous, sampled, and aperiodic updates. *International Journal of Control*, 86(4):567–578, 2013.

[69] S. Gupta. Increasing the sampling efficiency for a control system. *IEEE Transactions on Automatic Control*, 8(3):263–264, 1963.

[70] D. Han, Y. Mo, J. Wu, S. Weerakkody, B. Sinopoli, and L. Shi. Stochastic event-triggered sensor schedule for remote state estimation. *IEEE Transactions on Automatic Control*, 60(10):2661–2675, 2015.

[71] W. P. M. H. Heemels, J. H. Sandee, and P. P. J. van den Bosch. Analysis of event-driven controllers for linear systems. *International Journal of Control*, 81(4):571–590, 2008.

[72] 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*, pages 3270–3285, Maui, HI, 2012.

[73] W. P. M. H. Heemels, M. C. F. Donkers, and A. R. Teel. Periodic event-triggered control for linear systems. *IEEE Transactions on Automatic Control*, 58(4):847–861, 2013.

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

[75] E. Henriksson, D. E. Quevedo, E. G. W. Peters, H. Sandberg, and K. H. Johansson. Multiple-loop self-triggered model predictive control for network scheduling and control. *IEEE Transactions on Control Systems Technology*, 23(6):2167–2181, 2015.

[76] L. Hetel, C. Fiter, H. Omran, A. Seuret, E. Fridman, J. Richard, and S. I. Niculescu. Recent developments on the stability of systems with aperiodic sampling: an overview. *Automatica*, 76:309–335, 2017.

[77] A. Hu and J. Cao. Consensus of multi-agent systems via intermittent event-triggered control. *International Journal of Systems Science*, 48(2):280–287, 2017.

[78] W. Hu and L. Liu. Cooperative output regulation of heterogeneous linear multi-agent systems by event-triggered control. *IEEE Transactions on Cybernetics*, 47(1):105–116, 2017.

[79] W. Hu, L. Liu, and G. Feng. Output consensus of heterogeneous linear multi-agent systems by distributed event-triggered/self-triggered strategy. *IEEE Transactions on Cybernetics*, PP(99), 2016.

[80] W. Hu, L. Liu, and G. Feng. Output consensus of heterogeneous linear multi-agent systems by distributed event-triggered/self-triggered strategy. *IEEE Transactions on Cybernetics*, 47(8): 1914–1924, 2017.

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

[82] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, 3 edition, 2002. ISBN 0130673897.

[83] E. Kharisov, X. Wang, and N. Hovakimyan. Distributed event-triggered consensus algorithm for uncertain multi-agent systems. In *AIAA Guidance, Navigation, and Control Conference*, pages 1–15. AIAA, 2010.

[84] B. A. Khashooei, D. J. Antunes, and W. P. M. H. Heemels. Output-based event-triggered control with performance guarantees. *IEEE Transactions on Automatic Control*, 62(7):3646–3652, 2017.

[85] B. A. Khashooei, D. J. Antunes, and W. P. M. H. Heemels. A consistent threshold-based policy for event-triggered control. *IEEE Control Systems Letters*, 2(3):447–452, 2018.

[86] S. S. Kia, J. Cortés, and S. Martínez. Distributed event-triggered communication for dynamic average consensus in networked systems. *Automatica*, 59:112–119, 2015.

[87] S. S. Kia, J. Cortés, and S. Martínez. Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication. *Automatica*, 55:254–264, 2015.

[88] S. S. Kia, J. Cortés, and S. Martínez. Dynamic average consensus under limited control authority and privacy requirements. *International Journal on Robust and Nonlinear Control*, 25(13):1941–1966, 2015.

[89] P. Kolios, C. Panayiotou, G. Ellinas, and M. Polycarpou. Data-driven event triggering for IoT applications. *IEEE Internet of Things Journal*, 3(6): 1146–1158, 2016.

[90] H. Li, X. Liao, T. Huang, and W. Zhu. Event-triggering sampling based leader-following consensus in second-order multi-agent systems. *IEEE Transactions on Automatic Control*, 60(7):1998–2003, 2015.

[91] H. Li, G. Chen, T. Huang, Z. Dong, W. Zhu, and L. Gao. Event-triggered distributed average consensus over directed digital networks with limited communication bandwidth. *IEEE Transactions on Cybernetics*, 46(12):3098–3110, 2016.

[92] H. Li, G. Chen, T. Huang, W. Zhu, and L. Xiao. Event-triggered consensus in nonlinear multi-agent

systems with nonlinear dynamics and directed network topology. *Neurocomputing*, 185:105–112, 2016.

[93] H. Li, G. Chen, X. Liao, and T. Huang. Leader-following consensus of discrete-time multiagent systems with encoding–decoding. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(4): 401–405, 2016.

[94] H. Li, G. Chen, and L. Xiao. Event-triggered non-linear consensus in directed multi-agent systems with combinational state measurements. *International Journal of Systems Science*, 47(14):3364–3377, 2016.

[95] H. Li, G. Chen, and L. Xiao. Event-triggered sampling scheme for pinning control in multi-agent networks with general nonlinear dynamics. *Neural Computing and Applications*, 27(8):2587–2599, 2016.

[96] L. Li, D. Ho, and S. Xu. A distributed event-triggered scheme for discrete-time multi-agent consensus with communication delays. *IET Control Theory & Apllications*, 8(10):830–837, 2014.

[97] Z. Li, Z. Duan, G. Chen, and L. Huang. Consensus of multiagent systems and synchronization of complex networks: a unified viewpoint. *IEEE Transactions on Circuits and Systems-I: Regular papers*, 57 (1):213–224, 2010.

[98] Z. Li, Z. Duan, and G. Chen. Dynamic consensus of linear multi-agent systems. *Control Theory & Applications, IET*, 5(1):19–28, 2011.

[99] A. Liff and J. Wolf. On the optimum sampling rate for discrete-time modeling of continuous-time systems. *IEEE Transactions on Automatic Control*, 11(2):288–290, 1966.

[100] J. Liu and W. Chen. Distributed convex optimisation with event-triggered communication in networked systems. *International Journal of Systems Science*, 47(16):3876–3887, 2016.

[101] K. Liu, Z. Ji, G. Xie, and R. Xu. Event-based broadcasting containment control for multi-agent systems under directed topology. *International Journal of Control*, 89(11):2360–2370, 2016.

[102] Q. Liu, Z. Wang, X. He, and D. H. Zhou. Event-based consensus control of multi-agent systems with relative output feedback: The finite-horizon case. *IEEE Transactions on Automatic Control*, 60 (9):2553–2558, 2015.

[103] Q. Liu, Z. Wang, X. He, and D. H. Zhou. Event-based recursive distributed filtering over wireless sensor networks. *IEEE Transactions on Automatic Control*, 60(9):2470–2475, 2015.

[104] T. Liu, D. J. Hill, and B. Liu. Synchronization of dynamical networks with distributed event-based communication. In *IEEE Conf. on Decision and Control*, pages 7199–7204, 2012.

[105] T. Liu, M. Cao, and D. J. Hill. Distributed event-triggered control for output synchronization of dynamical networks with non-identical nodes. In *IEEE Conf. on Decision and Control*, pages 3554–

3559, Los Angeles, CA, 2014.

[106] X. Liu, C. Du, P. Lu, and D. Yang. Distributed event-triggered feedback consensus control with state-dependent threshold for general linear multi-agent systems. *International Journal of Robust and Nonlinear Control*, 2016.

[107] Y. Liu, C. Nowzari, Z. Tian, and Q. Ling. Asynchronous periodic event-triggered coordination of multi-agent systems. In *IEEE Conf. on Decision and Control*, pages 6696–6701, Melbourne, Australia, Dec. 2017.

[108] Z. Liu and Z. Chen. Reaching consensus in networks of agents via event-triggered control. *Journal of Information and Computational Science*, 8 (3):393–402, 2011.

[109] Z. Liu, Z. Chen, and Z. Yuan. Event-triggered average-consensus of multi-agent systems with weighted and direct topology. *Journal of Systems Science and Complexity*, 25(5):845–855, 2012.

[110] D. Luzza, D. V. Dimarogonas, M. di Bernardo, and K. H. Johansson. Distributed model based event-triggered control for synchronization of multi-agent systems. *Automatica*, 73:1–7, 2016.

[111] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1997. ISBN 1558603484.

[112] C. Q. Ma and J. F. Zhang. Necessary and sufficient conditions for consensusability of linear multi-agent systems. *IEEE Transactions on Automatic Control*, 55(5):1263–1268, 2010.

[113] M. S. Mahmoud, M. Sabih, and M. Elshafei. Event-triggered output feedback control for distributed networked systems. *Automatica*, 60:294–302, 2016.

[114] M. Mazo Jr. and M. Cao. Asynchronous decentralized event-triggered control. *Automatica*, 50(12): 3197–3203, 2014.

[115] M. Mazo Jr. and P. Tabuada. Decentralized event-triggered control over wireless sensor/actuator networks. *IEEE Transactions on Automatic Control*, 56(10):2456–2461, 2011.

[116] X. Meng and T. Chen. Event based agreement protocols for multi-agent networks. *Automatica*, 49 (7):2125–2132, 2013.

[117] 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*, pages 4151–4156, Osaka, Japan, Dec. 2015.

[118] X. Meng, L. Xie, and Y. C. Soh. Asynchronous periodic event-triggered consensus for multi-agent systems. *Automatica*, 2017. To appear.

[119] M. Miskowicz. Send-on-delta concept: An event-based data reporting strategy. *Sensors*, 6(1):49–63, 2006.

[120] J. Mitchell and W. McDaniel. Adaptive sampling technique. *IEEE Transactions on Automatic Control*, 14(2):200–201, 1967.

[121] A. Molin and S. Hirche. On LQG joint optimal scheduling and control under communication constraints. In *IEEE Conf. on Decision and Control*,

pages 5832–5838, Shanghai, China, 2009.

[122] A. Molin and S. Hirche. Stuctrural characterization of optimal event-based controllers for linear stochastic systems. In *IEEE Conf. on Decision and Control*, pages 3227–3233, Atlanta, GA, 2010.

[123] N. Mu, X. Liao, and T. Huang. Leader-following consensus in second-order multiagent systems via event-triggered control with nonperiodic sampled data. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(10):1007–1011, 2015.

[124] N. Mu, X. Liao, and T. Huang. Event-based consensus control for a linear directed multiagent system with time delay. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(3):281–285, 2015.

[125] A. Nedić. Distributed optimization. In J. Baillieul and T. Samad, editors, *Encyclopedia of Systems and Control*. Springer, New York, 2015.

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

[127] C. Nowzari and J. Cortés. Zeno-free, distributed event-triggered communication and control for multi-agent average consensus. In *American Control Conference*, pages 2148–2153, Portland, OR, 2014.

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

[129] C. Nowzari and J. Cortés. Team-triggered coordination for real-time control of networked cyber-physical systems. *IEEE Transactions on Automatic Control*, 61(1):34–47, 2016.

[130] C. Nowzari and G. J. Pappas. Multi-agent coordination with asynchronous cloud access. In *American Control Conference*, pages 4649–4654, Boston, MA, June 2016.

[131] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[132] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[133] P. Ong and J. Cortés. Event-triggered control design with performance barrier. In *IEEE Conf. on Decision and Control*, pages 951–956, Miami, FL, Dec. 2018.

[134] P. G. Otanez, J. R. Moyne, and D. M. Tilbury. Using deadbands to reduce communication in networked control systems. In *American Control Conference*, pages 3015–3020, Anchorage, AK, 2002.

[135] M. Ouimet, D. Iglesias, N. Ahmed, and S. Martínez. Cooperative robot localization using event-triggered estimation. *Journal of Aerospace and Information Systems*, 2018. To appear.

[136] C. Perera, C. H. Liu, S. Jayawardena, and M. Chen. A survey on internet of things from industrial mar-

ket perspective. *IEEE Access*, 2:1660–1679, 2014.

[137] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014.

[138] R. Postoyan, P. Tabuada, D. Nešić, and A. Anta. Event-triggered and self-triggered stabilization of distributed networked control systems. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 2565–2570, Orlando, FL, 2011.

[139] M. Rabi and J. S. Baras. Level-triggered control of a scalar linear system. In *Mediterranean Conference on Control and Automation*, pages 1–6, Athens, Greece, 2007.

[140] C. Ramesh, H. Sandberg, L. Bao, and K. H. Johansson. On the dual effect in state-based scheduling of networked control systems. In *American Control Conference*, pages 2216–2221, 2011.

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

[142] W. Ren. Synchronization of coupled harmonic oscillators with local interaction. *Automatica*, 44(12):3195–3200, 2008.

[143] W. Ren. On consensus algorithms for double-integrator dynamics. *IEEE Transactions on Automatic Control*, 53(6):1503–1509, 2008.

[144] W. Ren and E. Atkins. Distributed multi-vehicle coordinated control via local information exchange. *International Journal of Robust and Nonlinear Control*, 17(10-11):1002–1033, 2007.

[145] W. Ren and R. W. Beard. *Distributed Consensus in Multi-Vehicle Cooperative Control*. Communications and Control Engineering. Springer, 2008. ISBN 978-1-84800-014-8.

[146] W. Ren, R. W. Beard, and E. M. Atkins. Information consensus in multivehicle cooperative control. *IEEE Control Systems Magazine*, 27(2):71–82, 2007.

[147] D. Richert and J. Cortés. Distributed linear programming with event-triggered communication. *SIAM Journal on Control*, 54(3):1769–1797, 2016.

[148] A. Rikos, T. Charalambous, and C. N. Hadjicostis. Distributed weight balancing over digraphs. *IEEE Transactions on Control of Network Systems*, 1(2):190–201, 2014.

[149] J. Sandee, W. P. M. H. Heemels, and P. van den Bosch. Event-driven control as an opportunity in the multidisciplinary development of embedded controllers. In *American Control Conference*, pages 1776–1781, Portland, OR, 2005.

[150] L. Scardovi and R. Sepulchre. Synchronization in networks of identical linear systems. *Automatica*, 45(11):2557–2562, 2009.

[151] L. Schenato and F. Fiorentin. Average timesynch: a consensus-based protocol for clock synchronization in wireless sensor networks. *Automatica*, 47

(9):1878–1886, 2011.

[152] D. Senejohnny, P. Tesi, and C. De Persis. A jamming-resilient algorithm for self-triggered network coordination. *IEEE Transactions on Control of Network Systems*, PP(99):1–10, 2018.

[153] J. H. Seo, H. Shim, and J. Back. Consensus of high-order linear systems using dynamic output feedback compensator: low gain approach. *Automatica*, 45(11):2659–2664, 2009.

[154] A. Seuret, C. Prieur, and N. Marchand. Stability of non-linear systems by means of event-triggered sampling algorithms. *IMA Journal of Mathematical Control and Information*, 31(3):415–433, 2014.

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

[156] D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Distributed sensor fusion using dynamic consensus. In *IFAC World Congress*, Prague, CZ, July 2005. Electronic proceedings.

[157] Y. Su and J. Huang. Stability of a class of linear switching systems with applications to two consensus problems. *IEEE Transactions on Automatic Control*, 57(6):1420–1430, 2012.

[158] Z. Sun, N. Huang, B. D. O. Anderson, and Z. Duan. A new distributed Zeno-free event-triggered algorithm for multi-agent consensus. In *IEEE Conf. on Decision and Control*, pages 3444–3449, Las Vegas, NV, 2016.

[159] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

[160] P. Tallapragada and N. Chopra. Event-triggered decentralized dynamic output feedback control for lti systems. *IFAC Proceedings Volumes*, 45(26):31–36, 2012.

[161] P. Tallapragada and N. Chopra. Decentralized event-triggering for control of nonlinear systems. *IEEE Transactions on Automatic Control*, 59(12):3312–3324, 2014.

[162] D. Tolić and R. Fierrol. Decentralized output synchronization of heterogeneous linear systems with fixed and switching topology via self-triggered communication. In *American Control Conference*, pages 4648–4653, Washington, DC, 2013.

[163] D. Tolić and S. Hirche. *Networked Control Systems with Intermittent Feedback*. CRC Press, 2017.

[164] R. Tomovic and G. Bekey. Adaptive sampling based on amplitude sensitivity. *IEEE Transactions on Automatic Control*, 11(2):282–284, 1966.

[165] S. E. Tuna. Synchronizing linear systems via partial-state coupling. *Automatica*, 44(8):2179–2184, 2008.

[166] S. E. Tuna. Conditions for synchronizability in arrays of coupled linear systems. *IEEE Transactions on Automatic Control*, 54(10):2416–2420, 2009.

[167] P. Wan and M. D. Lemmon. Event-triggered distributed optimization in sensor networks. In *Sym-*

[168] posium on Information Processing of Sensor Networks*, pages 49–60, San Francisco, CA, 2009.

[168] X. Wang and M. D. Lemmon. Event-triggered broadcasting across distributed networked control systems. In *American Control Conference*, pages 3139–3144, Seattle, WA, 2008.

[169] X. Wang and M. D. Lemmon. Decentralized event-triggered broadcasts over networked control systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 674–677, 2008.

[170] X. Wang and M. D. Lemmon. Event-triggering in distributed networked control systems. *IEEE Transactions on Automatic Control*, 56(3):586–601, 2011.

[171] B. Wei, F. Xiao, and M. Dai. Edge event-triggered control for multi-agent systems under directed communication topologies. *International Journal of Control*, 91(4):887–896, 2018.

[172] Y. Wu, X. Meng, L. Xie, R. Lu, H. Su, and Z. Wu. An input-based triggering approach to leader-following problems. *Automatica*, 75:221–228, 2016.

[173] F. Xiao and T. Chen. Sampled-data consensus in multi-agent systems with asynchronous hybrid event-time driven interactions. *Systems & Control Letters*, 89:24–34, 2016.

[174] F. Xiao, T. Chen, and H. Gao. Synchronous hybrid event-and time-driven consensus in multiagent networks with time delays. *IEEE Transactions on Cybernetics*, 46(5):1165–1174, 2016.

[175] D. Xie and J. Xie. Consensus control of multi-agent systems with event-triggered strategy. *Information Technology Journal*, 13(14):2250–2259, 2014.

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

[177] Y. Xiuxia and Y. Dong. Event-triggered consensus of nonlinear multi-agent systems with nonlinear interconnections. In *Chinese Control Conference*, pages 7087–7092, Xi'an, China, 2013.

[178] W. Xu, D. W. C. Ho, J. Zhong, and B. Chen. Distributed edge event-triggered consensus protocol of multi-agent systems with communication buffer. *International Journal of Robust and Nonlinear Control*, 27(3):483–496, 2016.

[179] W. Xu, D. Ho, L. Li, and J. Cao. Event-triggered schemes on leader-following consensus of general linear multiagent systems under different topologies. *IEEE Transactions on Cybernetics*, 47(1):212–223, 2017.

[180] Y. Xu and J. P. Hespanha. Estimation under uncontrolled and controlled communications in networked control systems. In *IEEE Conf. on Decision and Control*, pages 842–847, Seville, Spain, 2005.

[181] D. Xue and S. Hirche. Event-triggered consensus of heterogeneous multi-agent systems with double integrator dynamics. In *European Control Confer-*

ence, pages 1162–1167, 2013.

[182] H. Yan, Y. Shen, H. Zhang, and H. Shi. Decentralized event-triggered consensus control for second-order multi-agent systems. *Neurocomputing*, 133: 18–24, 2014.

[183] D. Yang, W. Ren, and X. Liu. Decentralized consensus for linear multi-agent systems under general directed graphs based on event-triggered/self-triggered strategy. In *IEEE Conf. on Decision and Control*, pages 1983–1988, Los Angeles, CA, 2014.

[184] D. Yang, X. Liu, and W. Chen. Periodic event/self-triggered consensus for general continuous-time linear multi-agent systems under general directed graphs. *IET Control Theory & Applications*, 9(3): 428–440, 2015.

[185] D. Yang, W. Ren, X. Liu, and W. Chen. Decentralized event-triggered consensus for linear multi-agent systems under general directed graphs. *Automatica*, 69:242–249, 2016.

[186] P. Yang, R. A. Freeman, G. J. Gordon, K. M. Lynch, S. S. Srinivasa, and R. Sukthankar. Decentralized estimation and control of graph connectivity for mobile sensor networks. *Automatica*, 46(2): 390–396, 2010.

[187] X. Yi, J. Wei, and K. H. Johansson. Self-triggered control for multi-agent systems with quantized communication or sensing. In *IEEE Conf. on Decision and Control*, pages 2227–2232, Las Vegas, NV, Dec. 2016.

[188] X. Yi, K. Liu, D. V. Dimarogonas, and K. H. Johansson. Distributed dynamic event-triggered control for multi-agent systems. In *IEEE Conf. on Decision and Control*, pages 6683–6688, Melbourne, Australia, 2017.

[189] X. Yi, J. Wei, D. V. Dimarogonas, and K. H. Johansson. Formation control for multi-agent systems with connectivity preservation and event-triggered controllers. *IFAC-PapersOnLine*, 50(1): 9367–9373, 2017.

[190] X. Yin and D. Yue. Event-triggered tracking control for heterogeneous multi-agent systems with Markov comunication delays. *Journal of the Franklin Institute*, 350(5):1312–1334, 2013.

[191] X. Yin, D. Yue, and S. Hu. Distributed event-triggered control of discrete-time heterogeneous multi-agent systems. *Journal of the Franklin Institute*, 350(3):651–669, 2013.

[192] H. Yu and P. Antsaklis. Quantized output synchronization of networked passive systems with event-driven communication. In *American Control Conference*, pages 5706–5711, Montreal, Canada, 2012.

[193] H. Yu and P. J. Antsaklis. Formation control of multi-agent systems with connectivity preservation by using both event-driven and time-driven communication. In *IEEE Conf. on Decision and Control*, pages 7218–7223, Maui, HI, 2012.

[194] H. Yu and P. J. Antsaklis. Output synchronization of networked passive systems with event-driven communication. *IEEE Transactions on Automatic Control*, 59(3):750–756, 2014.

[195] P. Yu, L. Ding, Z. Liu, and Z. Guan. Leader-follower flocking based on distributed event-triggered hybrid control. *International Journal of Robust and Nonlinear Control*, 26(1):143–153, 2016.

[196] H. Zhang, G. Feng, H. Yan, and Q. Chen. Consensus of multi-agent systems with linear dynamics using event-triggered control. *IET Control Theory & Applications*, 8(18):2275–2281, 2014.

[197] H. Zhang, G. Feng, H. Yan, and Q. Chen. Observer-based output feedback event-triggered control for consensus of multi-agent systems. *IEEE Transactions on Industrial Electronics*, 61(9):4885–4894, 2014.

[198] X. Zhang, M. Chen, and L. Wang. Distributed event-triggered consensus in multi-agent systems with non-linear protocols. *IET Control Theory and Applications*, 9(18):2626–2633, 2015.

[199] X. Zhang, L. Liu, and G. Feng. Leader-follower consensus of time-varying nonlinear multi-agent systems. *Automatica*, 52:8–14, 2015.

[200] Z. Zhang, F. Hao, L. Zhang, and L. Wang. Consensus of linear multi-agent systems via event-triggered control. *International Journal of Control*, 87(6):1243–1251, 2014.

[201] Z. Zhang, L. Zhang, F. Hao, and L. Wang. Distributed event-triggered consensus for multi-agent systems with quantization. *International Journal of Control*, 88(6):1112–1122, 2015.

[202] M. Zhong and C. G. Cassandras. Asynchronous distributed optimization with event-driven communication. *IEEE Transactions on Automatic Control*, 55(12):2735–2750, 2010.

[203] B. Zhou, X. Liao, T. Huang, and G. Chen. Leader-following exponential consensus of general linear multi-agent systems via event-triggered control with combinational measurements. *Applied Mathematics Letters*, 40:35–39, 2015.

[204] B. Zhou, X. Liao, T. Huang, H. Li, and G. Chen. Event-based semiglobal consensus of homogeneous linear multi-agent systems subject to input saturation. *Asian Journal of Control*, 19(2):564–574, 2016.

[205] F. Zhou, Z. Huang, Y. Yang, J. Wang, L. Li, and J. Peng. Decentralized event-triggered cooperative control for multi-agent systems with uncertain dynamics using local estimators. *Neurocomputing*, 237:388–396, 2017.

[206] W. Zhu and Z. Jiang. Event-based leader-following consensus of multi-agent systems with input time delay. *IEEE Transactions on Automatic Control*, 60(5):1362–1367, 2015.

[207] W. Zhu, Z. Jiang, and G. Feng. Event-based consensus of multi-agent systems with general linear models. *Automatica*, 50(2):552–558, 2014.

[208] W. Zhu, H. Li, and Z. Jiang. Consensus of multi-agent systems with time-varying topology: An event-based dynamic feedback scheme. *International Journal of Robust and Nonlinear Control*,

27(8):1339–1350, 2016.

[209] W. Zhu, H. Pu, D. Wang, and H. Li. Event-based consensus of second-order multi-agent systems with discrete time. *Automatica*, 79:78–83, 2017.

[210] L. Zou, Z. Wang, and D. Zhou. Event-based control and filtering of networked systems: A survey. *International Journal of Automation and Computing*, 14(3):239–253, 2017.

Appendix

Proof of Theorem 2.2 Consider the Lyapunov function

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

Then, given the dynamics (10) and the continuous control law (11),

$$\dot{V}(x) = x^T L \dot{x} = -x^T L^T L x = -\|Lx\|^2,$$

where we have used the fact that L is symmetric. It is now clear that using the continuous control law (11) we have $\dot{V}(x) < 0$ for all $Lx \neq 0$. Using LaSalle's Invariance Principle [82], it can then be shown that

$$x(t) \to \{Lx = 0\} = \{x_i = x_j \ \forall i, j \in \{1, \dots, N\}\}$$

as $t \to \infty$. Combining this with the fact that the sum of all states is an invariant quantity concludes the proof,

$$\frac{d}{dt}\left(\mathbf{1}_N^T x(t)\right) = \mathbf{1}_N^T \dot{x}(t) = -\mathbf{1}_N^T L x(t) = 0.$$

■

Proof of Theorem 3.1 Consider again the Lyapunov function

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

Then, we see that the trigger (33) guarantees that (32) is satisfied at all times. Combined with the closed-loop dynamics (30), we have

$$\dot{V} \leq \sum_{i=1}^{N}(\sigma_i - 1)(1 - a|\mathcal{N}_i|)\hat{z}_i^2,$$

which is strictly negative for all $L\hat{x} \neq 0$. Similar to the conclusion in the proof of Theorem 2.2, we can similarly show that $\hat{x} \to \{L\hat{x} = 0\}$. By noticing that $\hat{x}_i$ is simply a sampled subset of the trajectory of x_i, we have that $x \to \{Lx = 0\}$. Finally, combining this again with the fact that the sum of all states is an invariant quantity concludes the proof.

■

Proof of Lemma 3.2 Given the Lyapunov function

$$V(x) = \frac{1}{2}(x - \bar{x}\mathbf{1})^T(x - \bar{x}\mathbf{1})$$

and the closed-loop dynamics (30), we have

$$\dot{V} = x^T \dot{x} - \bar{x}^T \dot{x} = -x^T L\hat{x} - \bar{x}L\hat{x} = -x^T L\hat{x}.$$

Recalling $e_i(t) = \hat{x}_i(t) - x_i(t)$, we can expand this out to

$$\dot{V} = -\hat{x}^T L\hat{x} + e^T L\hat{x}$$

$$= -\sum_{i=1}^{N}\sum_{j \in \mathcal{N}_i}\left(\frac{1}{2}(\hat{x}_i - \hat{x}_j)^2 - e_i(\hat{x}_i - \hat{x}_j)\right).$$

Using Young's inequality for each product we can bound (see [128] for why this choice)

$$e_i(\hat{x}_i - \hat{x}_j) \leq e_i^2 + \frac{1}{4}(\hat{x}_i - \hat{x}_j)^2$$

which yields

$$\dot{V} \leq -\sum_{i=1}^{N}\sum_{j \in \mathcal{N}_i}\left(\frac{1}{2}(\hat{x}_i - \hat{x}_j)^2 - e_i^2 - \frac{1}{4}(\hat{x}_i - \hat{x}_j)^2\right)$$

$$= -\sum_{i=1}^{N}\sum_{j \in \mathcal{N}_i}\left(\frac{1}{4}(\hat{x}_i - \hat{x}_j)^2 - e_i^2\right)$$

$$= \sum_{i=1}^{N}e_i^2|\mathcal{N}_i| - \sum_{j \in \mathcal{N}_i}\left(\frac{1}{4}(\hat{x}_i - \hat{x}_j)^2\right).$$

■

Proof of Theorem 3.6 Let $\delta(t) = x(t) - \bar{x}\mathbf{1}$, where $\bar{x} = \frac{1}{N}\sum_{i=1}^{N}x_i(0)$ is the average of all initial conditions. Then, $\dot{\delta}(t) = -L\delta(t) - Le(t)$, yielding

$$\delta(t) = e^{-Lt}\delta(0) - \int_0^t e^{-L(t-s)}Le(s)ds.$$

Taking norms,

$$\|\delta(t)\| \leq \|\delta(0)e^{-Lt}\| + \int_0^t \|e^{-L(t-s)}Le(s)\|ds$$

$$\leq e^{-\lambda_2(L)t}\|\delta(0)\| + \int_0^t e^{-\lambda_2(L)(t-s)}\|Le(s)\|ds,$$

where the second inequality follows from [155, Lemma 2.1].

Using the condition

$$|e_i(t)| \leq c_0 + c_1 e^{-\alpha t},$$

it follows that

$$\|\delta(t)\| \le e^{-\lambda_2 t}\|\delta(0)\| + \|L\|\sqrt{N}\int_0^t e^{-\lambda_2(t-s)}(c_0 + c_1 e^{-\alpha s})ds$$

$$= e^{-\lambda_2 t}\left(\|\delta(0)\| - \|L\|\sqrt{N}\left(\frac{c_0}{\lambda_2} + \frac{c_1}{\lambda_2 - \alpha}\right)\right)$$

$$+ e^{-\alpha t}\frac{\|L\|\sqrt{N}c_1}{\lambda_2 - \alpha} + \frac{\|L\|\sqrt{N}c_0}{\lambda_2}.$$

The convergence result then follows by taking $t \to \infty$. ∎

Proof of Theorem 3.5 Consider the Lyapunov function

$$V(x) = \frac{1}{2}(x - \bar{x}\mathbf{1})^T(x - \bar{x}\mathbf{1}).$$

Following the discussing after Lemma 3.2, we know that when (35) is satisfied, we have $\dot{V}$ is strictly negative for all $L\hat{x} \neq 0$. However, since the agents can now only evaluate the trigger (56) at the sampling times under the periodic event-triggered coordination algorithm presented in Theorem 3.5, we lose the guarantee that $\dot{V} \le 0$ at all times. Thus, we must now analyze what happens to the Lyapunov function V in between these sampling times. Explicitly considering $t \in [t_{\ell'}, t_{\ell'+1})$, note that

$$e(t) = e(t_{\ell'}) + (t - t_{\ell'})L\hat{x}(t_{\ell'}).$$

Substituting this expression into $\dot{V}(t) = -\hat{x}^T(t)L\hat{x}(t) + e^T(t)L\hat{x}(t)$, we obtain

$$\dot{V}(t) = -\hat{x}^T(t_{\ell'})L\hat{x}(t_{\ell'}) + e^T(t_{\ell'})L\hat{x}(t_{\ell'})$$
$$+ (t - t_{\ell'})\hat{x}^T(t_{\ell'})L^T L\hat{x}(t_{\ell'}),$$

for all $t \in [t_{\ell'}, t_{\ell'+1})$. For a simpler exposition, we drop all arguments referring to time $t_{\ell'}$ in the sequel. Then, using (35) to bound $e(t'_\ell)$, we can show

$$\dot{V}(t) \le \sum_{i=1}^N \frac{\sigma_i - 1}{4}\sum_{j \in \mathcal{N}_i}(\hat{x}_i - \hat{x}_j)^2 + (t - t_{\ell'})\hat{x}^T L^T L\hat{x}.$$

Note that the first term is exactly what we have when we are able to monitor the trigger continuously (36).

Using the fact that $\left(\sum_{k=1}^p y_k\right)^2 \le p\sum_{k=1}^p y_k^2$ (which follows directly from the Cauchy-Schwarz inequality), we bound

$$\hat{x}^T L^T L\hat{x} = \sum_{i=1}^N \left(\sum_{j \in \mathcal{N}_i}(\hat{x}_i - \hat{x}_j)\right)^2$$

$$\le \sum_{i=1}^N |\mathcal{N}_i|\sum_{j \in \mathcal{N}_i}(\hat{x}_i - \hat{x}_j)^2.$$

Hence, for $t \in [t_{\ell'}, t_{\ell'+1})$,

$$\dot{V}(t) \le \sum_{i=1}^N \left(\frac{\sigma_i - 1}{4} + h|\mathcal{N}_{\max}|^2\right)\sum_{j \in \mathcal{N}_i}(\hat{x}_i - \hat{x}_j)^2$$

$$\le \left(\frac{\sigma_{\max} - 1}{2} + 2h|\mathcal{N}_{\max}|^2\right)\hat{x}^T L\hat{x}.$$

Then, by using (45), it can be shown that there exists $\mathcal{B} > 0$ such that

$$\dot{V}(t) \le \frac{1}{2\mathcal{B}}\left(\sigma_{\max} + 4h|\mathcal{N}_{\max}|^2 - 1\right)V(x(t)),$$

which implies the result. See [128] for more details. ∎

Proof of Theorem 3.11 Consider the Lyapunov function

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

Then, we see that the trigger (56) ensures that (54) is satisfied at all times. Then, leveraging Lemma 3.10, we have

$$\dot{V} \le \sum_{i=1}^N \frac{\sigma_i - 1}{4}\sum_{j \in \mathcal{N}_i^{\mathrm{out}}} w_{ij}(\hat{x}_i - \hat{x}_j)^2,$$

which is strictly negative for all $L\hat{x} \neq 0$. Following the discussion in the proof of Theorem 3.1, we have that $x \to \{Lx = 0\}$. Finally, because the graph is weight-balanced the sum of all states is an invariant quantity,

$$\frac{d}{dt}\left(\mathbf{1}_N^T x(t)\right) = \mathbf{1}_N^T \dot{x}(t) = -\mathbf{1}_N^T L\hat{x}(t) = 0,$$

which concludes the proof. ∎

Proof of Theorem 3.15 Let $\delta(t) = x(t) - \bar{x}\mathbf{1}$, where $\bar{x} = \frac{1}{N}\sum_{i=1}^N x_i(0)$ is the average of all initial conditions. Then, $\dot{\delta}(t) = -L\delta(t) - Le(t)$, yielding

$$\delta(t) = e^{-Lt}\delta(0) - \int_0^t e^{-L(t-s)}Le(s)ds.$$

Taking norms,

$$\|\delta(t)\| \le \|\delta(0)e^{-Lt}\| + \int_0^t \|e^{-L(t-s)}Le(s)\|ds$$

$$\le e^{-\lambda_2(L)t}\|\delta(0)\| + \int_0^t e^{-\lambda_2(L)(t-s)}\|Le(s)\|ds,$$

where the second inequality follows from [155, Lemma 2.1].

Using the condition

$$|e_i(t)| \le c_0 + c_1 e^{-\alpha t},$$

it follows that

$$\|\delta(t)\| \le e^{-\lambda_2 t} \|\delta(0)\| + \|L\| \sqrt{N} \int_0^t e^{-\lambda_2 (t-s)} (c_0 + c_1 e^{-\alpha s}) ds$$

$$= e^{-\lambda_2 t} \left(\|\delta(0)\| - \|L\| \sqrt{N} \left(\frac{c_0}{\lambda_2} + \frac{c_1}{\lambda_2 - \alpha} \right) \right)$$

$$+ e^{-\alpha t} \frac{\|L\| \sqrt{N} c_1}{\lambda_2 - \alpha} + \frac{\|L\| \sqrt{N} c_0}{\lambda_2}.$$

The convergence result then follows by taking $t \to \infty$. See [155] for details on excluding Zeno behavior. $\blacksquare$

Proof of Lemma 3.16 Define $x = [x_1, ..., x_N]^T$. Then, using the Kronecker product, the dynamics of the overall system can be expressed as follows

$$\dot{x} = (\bar{A} + \bar{B})x$$

where $\bar{A} = I_N \otimes A$ and $\bar{B} = cL \otimes BF$. There exists a similarity transformation S such that $L_J = S^{-1} LS$ is in Jordan canonical form. Define $\bar{S} = S \otimes I_n$ and calculate the following:

$$\bar{S}^{-1} (\bar{A} - \bar{B}) \bar{S} = \bar{S}^{-1} \bar{A} \bar{S} + \bar{S}^{-1} (L \otimes BF) \bar{S}$$

$$= I_N \otimes A + cL_J \otimes BF.$$

By applying the similarity transformation we obtain that the eigenvalues of $\bar{A} + \bar{B}$ are given by the eigenvalues of $A + c\lambda_j BF$, where $\lambda_j = \lambda_j(L)$. $\blacksquare$