Robust team-triggered coordination of networked cyberphysical systems

Cameron Nowzari and Jorge Cortés

University of California, San Diego
9500 Gilman Dr, La Jolla, California 92093

Abstract. This paper proposes a novel approach, termed team-triggered, to the real-time implementation of distributed controllers on networked cyberphysical systems. We build on the strengths of event- and self-triggered control to synthesize a single, unified approach that combines aspects of both and is implementable over distributed networked systems, while maintaining desired levels of performance. We establish provably correct guarantees of the distributed strategies resulting from the proposed approach and examine their robustness against multiple sources of errors including communication delays, packet drops, and communication noise. The results are illustrated in simulations of a multi-agent coverage control problem.

1 Introduction

The interest in the efficient and robust operation of cyberphysical systems has motivated a growing body of work that studies the distributed design and real-time implementation of controllers for networked sensors and actuators. In these systems, energy consumption is correlated with the rate at which sensors take samples, processors recompute control inputs, and actuator signals are transmitted. Performing these tasks periodically is costly and might be, at times, unnecessary or unfeasible due to physical constraints. Examples of unnecessary actions include sampling a part of the state that is already known or can be reconstructed with already available information, or recomputing a control signal that has not changed substantially. To address these issues, the goal of triggered control is to identify criteria that allow agents to tune the implementation of controllers and sampling schemes to the execution of the task at hand and the desired level of performance.

In event-triggered control, the focus is on detecting events during the network execution that are relevant from the point of view of task completion and should trigger specific agent actions. In self-triggered control, instead, the emphasis is on developing tests that rely only on the information available to individual agents to schedule future actions. Event-triggered control results in better performance but is costly to implement over fully distributed networked scenarios because of the need for continuous availability of information to check the triggers. Self-triggered control is more easily amenable to distributed implementation but results in conservative executions because of overapproximation by individual

agents about the state of the environment and the network. This paper builds on the strengths of event- and self-triggered control to propose a single, unified approach for networked cyberphysical systems that combines the best of both worlds.

Literature review. The need for systems integration and the importance of bridging the gap between computing, communication, and control in the study of cyberphysical systems cannot be overemphasized [1, 2]. Real-time controller implementation is an area of extensive research, including periodic [3–5], event-triggered [6–10], and self-triggered [11–14] procedures. Our approach shares with these works the aim of trading computation and decision making for less communication, sensor, or actuator effort while still guaranteeing a desired level of performance.

Of particular relevance to this paper are works that study self- and event-triggered implementations of controllers for networked cyberphysical systems. The predominant paradigm is that of a single plant that is stabilized through a decentralized triggered controller over a sensor-actuator network, see e.g. [15–17]. Fewer works have considered scenarios where multiple plants or agents together are the subject of the overall control design, as is the case of this paper. Exceptions include consensus via event-triggered [18, 19, 15] or self-triggered control [18, 20], model predictive control [21], and model-based event-triggered control [22, 23]. The work [24] implements self-triggered communication schemes to perform distributed control where agents assume worst-case conditions occur for other agents when deciding when new information should be obtained. An idea to extend the event-triggered tools to decentralized systems with multiple plants is presented in [18]; however, agents require continuous information about each others' states in order to implement the resulting event-triggered controller. Distributed strategies based on event-triggered communication and control are explored in [25, 26], where each agent has an a priori computed local error tolerance and once it violates it, the agent broadcasts its updated state to its neighbors. The same event-triggered approach is taken in [27] to implement gradient control laws that achieve distributed optimization. The work [22], closer in spirit to the ideas presented here, considers an interconnected system in which each subsystem helps neighboring subsystems by monitoring their estimates and ensuring that they stay within some performance bounds. The approach, however, requires different subsystems to have synchronized estimates of one another even though they do not communicate at all times. A main novelty of this manuscript with respect to the works listed above is the combination of elements of both event- and self-triggered strategies into a single unified approach.

Statement of contributions. We propose a novel scheme for networked cyberphysical systems that combines ideas from event- and self-triggered control. The basic concept is for agents to make promises to one another about their future states and being responsible for warning each other if they later decide to break them (event-triggering). Such promises can be broad, from very tight state

trajectories to loose descriptions of reachability sets. The information provided by these promises allows individual agents to autonomously determine when fresh information is needed (self-triggering). We refer to the approach as team-triggered because of the fact that agents need to make sure that their neighbors are operating with correct information about them. The team-triggered approach incorporates the reactive nature of event-triggered implementations while at the same time endows individual agents with autonomous tools and criteria to determine when and what information is needed, as self-triggered implementations do. The benefits of the proposed scheme are threefold. First, because of the availability of the promises, agents do not require continuous state information about neighbors, in contrast to event-triggered strategies implemented over distributed systems that assume continuous information is available in order to be able to detect the relevant triggers. Second, because of the extra information provided by promises about what other agents plan to do, agents can operate more efficiently and less conservatively than if only worst-case scenarios are assumed, as is done in self-triggered control. Lastly, we show that the networked system is guaranteed to maintain desired levels of performance while being robust to multiple physical sources of error such as communication delays, packet drops, and communication noise. We illustrate our results through simulations in a multi-agent coverage control problem. The journal version [28] of this work contains all proofs and additional results.

Organization. Section 2 lays out the problem of interest. Section 3 presents the team-triggered approach and Section 4 discusses correctness and robustness guarantees. Simulations illustrate our results in Section 5. Section 6 gathers our conclusions and ideas for future work.

Notation. We let $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{Z}_{\geq 0}$ denote the sets of real, nonnegative real, and nonnegative integer numbers, respectively. The two-norm a vector is denoted by $\|\cdot\|_2$. Given $x \in \mathbb{R}^d$ and $\delta \in \mathbb{R}_{\geq 0}$, $\overline{B}(x, \delta)$ denotes the closed ball centered at x with radius δ. For $A_i \subset \mathbb{R}^{m_i \times n_i}$ with $i \in \{1, \dots, N\}$, we denote by $\operatorname{diag}(A_1, \dots, A_N) \in \mathbb{R}^{m \times n}$ the block-diagonal matrix with A_1 through A_N on the diagonal, where $m = \sum_{i=1}^N m_i$ and $n = \sum_{i=1}^N n_i$.

Given a set S, we denote by $|S|$ its cardinality. We let $\mathbb{P}^{\mathrm{cc}}(S)$ denote the collection of all closed and connected subsets of S. Similarly, we let $\mathbb{P}^{\mathrm{c}}(S)$ denote the collection of all closed subsets of S. Given $S \subset \mathbb{R}^d$, we denote the distance from a point x to S as $\operatorname{dist}(x, S) = \inf_{y \in S} \|x - y\|_2$. Given $S_1, S_2 \subset \mathbb{R}^d$, the Hausdorff distance between S_1 and S_2 is

$$d_H(S_1, S_2) = \max\{\sup_{x \in S_1} \inf_{y \in S_2} \|x - y\|_2, \sup_{y \in S_2} \inf_{x \in S_1} \|x - y\|_2\}.$$

The Hausdorff distance is a metric on the set of all non-empty compact subsets of $\mathbb{R}^d$. Given two set-valued functions $C_1, C_2 \in \mathcal{C}^0(I \in \mathbb{R}; \mathbb{P}^{\mathrm{c}}(\mathbb{R}^d))$, we define the distance between the set-valued functions as

$$d_{\mathrm{func}}(C_1, C_2) = \sup_{t \in I} d_H(C_1(t), C_2(t)). \tag{1}$$

An undirected graph $\mathcal{G} = (V, E)$ is a pair consisting of a set of vertices $V = \{1, \dots, N\}$ and a set of edges $E \subset V \times V$ such that if $(i, j) \in E$, then $(j, i) \in E$ as well. The set of neighbors of a vertex i is given by $\mathcal{N}(i) = \{j \in V \mid (i, j) \in E\}$. Given $v \in \prod_{i=1}^{N} \mathbb{R}^{n_i}$, we let $v^i_{\mathcal{N}} = (v_i, \{v_j\}_{j \in \mathcal{N}(i)})$ denote the components of v that correspond to vertex i and its neighbors in $\mathcal{G}$.

2 Problem statement

We consider a distributed control problem carried out over a realistic, unreliable wireless network. Consider N agents whose communication topology is described by an undirected graph $\mathcal{G}$. The fact that (i, j) belongs to E models the ability of agents i and j to communicate with one another. The set of all agents that i can communicate with is then given by its set of neighbors $\mathcal{N}(i)$ in the graph $\mathcal{G}$. The state of agent $i \in \{1, \dots, N\}$, denoted x_i, belongs to a closed set $\mathcal{X}_i \subset \mathbb{R}^{n_i}$. The network state $x = (x_1, \dots, x_N)$ therefore belongs to $\mathcal{X} = \prod_{i=1}^{N} \mathcal{X}_i$. According to the discussion above, agent i can access $x^i_{\mathcal{N}}$ when it communicates with its neighbors. We assume that each agent has access to its own state at all times.

We consider linear dynamics for each $i \in \{1, \dots, N\}$,

$$\dot{x}_i = f_i(x_i, u_i) = A_i x_i + B_i u_i, \tag{2}$$

with $A_i \in \mathbb{R}^{n_i \times n_i}$, $B_i \in \mathbb{R}^{n_i \times m_i}$, and $u_i \in \mathcal{U}_i$. Here, $\mathcal{U}_i \subset \mathbb{R}^{m_i}$ is a closed set of allowable controls for agent i. We assume that the pair (A_i, B_i) is controllable with controls taking values in $\mathcal{U}_i$. Further, we assume there exists a *safe-mode* controller $u^{\mathrm{sf}}_i : \mathcal{X}_i \to \mathcal{U}_i$ such that

$$A_i x_i + B_i u^{\mathrm{sf}}_i(x_i) = 0, \quad \text{for all } x_i \in \mathcal{X}_i, \tag{3}$$

i.e., a controller able to keep agent i's state fixed. Note that this means that the null space of B_i should be contained in the null space of A_i.

2.1 Controller and system certification

The goal of the network is to drive the agents' states to some desired set of configurations $D \subset \prod_{i=1}^{N} \mathcal{X}_i$ and ensure that it stays there. Depending on how the set D is defined, this objective can capture different coordination tasks including deployment, rendezvous, or formation control. The scope of the paper is not to design the controller that achieves this, but rather synthesize efficient strategies for the real-time implementation of a given controller in the presence of communication delays, packet drops, and communication noise that still certifies convergence to the desired set D.

Given the agent dynamics, the communication graph $\mathcal{G}$, and the set D, our starting point is the availability of a continuous control law that drives the system asymptotically to D. Formally, we assume that a continuous map $u^* :$

$\prod_{i=1}^{N} \mathcal{X}_i \to \mathbb{R}^m$ and a continuously differentiable function $V : \prod_{i=1}^{N} \mathcal{X}_i \to \mathbb{R}$, bounded from below, exist such that for all $x \notin D$,

$$\nabla_i V(x)\,(A_i x_i + B_i u_i^*(x)) \leq 0, \quad i \in \{1, \ldots, N\}, \tag{4a}$$

$$\sum_{i=1}^{N} \nabla_i V(x)\,(A_i x_i + B_i u_i^*(x)) < 0. \tag{4b}$$

We assume that both the control law u^* and the gradient ∇V are distributed over $\mathcal{G}$. By this we mean that, for each $i \in \{1, \ldots, N\}$, the ith component of each of these objects only depends on $x^i_{\mathcal{N}}$, rather than on the full network state x. For simplicity, and with a slight abuse of notation, we write $u_i^*(x^i_{\mathcal{N}})$ and $\nabla_i V(x^i_{\mathcal{N}})$ to emphasize this fact when convenient. This property has the important consequence that agent i can compute u_i^* and $\nabla_i V$ with the exact information it can obtain through communication on $\mathcal{G}$.

The controller u^* requires continuous agent-to-agent communication in order to be implemented. However, this is unrealistic and inefficient in practical networked scenarios, especially in the presence of communication delays and packet drops. Our goal is to relax this continuous information requirement to provide robust and efficient real-time controller implementations over networked cyberphysical systems.

2.2 Physical sources of error

We are ultimately interested in scenarios with unreliable communication among agents. More specifically, we look at three things. The first is communication noise, which we assume is bounded as follows: given a state x_i, a message y_i can be sent such that $\|y_i - x_i\|_2 \leq \bar{\omega}$ for some known $\bar{\omega} \in \mathbb{R}_{\geq 0}$. Second is the possibility of packet drops in the network. For any given message an agent sends to another agent, there is an unknown probability $0 \leq p < 1$ that the packet is dropped, and the message is never received. Lastly, we also consider the possibility that, at any time t, there is an unknown (possibly time-varying) communication delay $\Delta(t) \leq \bar{\Delta}$ in the network, where $\bar{\Delta} \in \mathbb{R}_{\geq 0}$ is known. In other words, if agent j sends agent i a message at time t, agent i will not receive it with probability p or receive it at time $t + \Delta(t)$ with probability $1 - p$. We assume that small messages (i.e., 1-bit messages) can be sent reliably with negligible delay. This assumption is similar to the "acknowledgments" and "permission" messages used in other works, see [25, 29] and references therein.

3 Team-triggered coordination for real-time networked control

Here we present a novel communication strategy to implement distributed controllers in real time on cyberphysical systems. Our strategy, termed team-triggered, combines ideas from event- and self-triggered approaches. Agents make promises to their neighbors about their future states and inform them later if these

promises are violated (hence the connection with event-triggered control). With the information available to them, each agent computes the next time that an update is required based on the evolution of the Lyapunov function (hence the connection with self-triggered control).

3.1 Promises

A *promise* can be either a time-varying set of states (state promise) or controls (control promise) that an agent sends to another agent. Specifically, a state promise that agent j makes to agent i at time t is a set-valued, continuous (with respect to the Hausdorff distance) function $X_j^i[t] \in \mathcal{C}^0([t,\infty);\mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j))$. This means that agent j promises to agent i that its state at any time $t' \geq t$ will satisfy $x_j(t') \in X_j^i[t](t')$. Similarly, a control promise can be conveyed by a set-valued, continuous function $U_j^i[t] \in \mathcal{C}^0([t,\infty);\mathbb{P}^{\mathrm{c}}(\mathcal{U}_j))$. This means that agent j promises to only use controls $u_j(t') \in U_j^i[t](t')$ for all $t' \geq t$. With this information and knowledge of the dynamics of agent j, agent i can compute the state promise

$$X_j^i[t](t') = \{z \in \mathcal{X}_j \mid \exists\, u_j : [t,t'] \to \mathcal{U}_j \text{ with } u_j(s) \in U_j^i[t](s) \text{ for } s \in [t,t'] \\ \text{such that } z = e^{A_j(t'-t)}x_j(t) + \int_t^{t'} e^{A_j(t'-\tau)}B_j u_j(\tau)d\tau\}. \quad (5)$$

For simplicity, when the time at which the promise is received is not relevant, we use the notation $X_j^i[\cdot]$ and $U_j^i[\cdot]$ or simply X_j^i and U_j^i, respectively. All promise information available to agent $i \in \{1,\dots,N\}$ at some time t is given by $X_{\mathcal{N}}^i[\cdot]_{|[t,\infty)} = (x_{i|[t,\infty)}, \{X_j^i[\cdot]_{|[t,\infty)}\}_{j\in\mathcal{N}(i)}) \in \mathcal{C}^0\left([t,\infty);\prod_{j\in\mathcal{N}(i)\cup\{i\}}\mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j)\right)$. To extract information from this about a specific time t', we use $X_{\mathcal{N}}^i[\cdot](t')$ or simply $X_{\mathcal{N}}^i(t') = (x_i(t'), \{X_j^i[\cdot](t')\}_{j\in\mathcal{N}(i)}) \in \prod_{j\in\mathcal{N}(i)\cup\{i\}}\mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j)$. The generality of the above definitions allow promise sets to be arbitrarily complex. Here, we restrict ourselves to promise sets that can be described with a finite number of parameters so that these promises can be conveyed to one another in a realistic manner, i.e., without requiring an infinite number of bits.

The method of generating promises is not unique and can be done in a number of ways. A *promise rule* is a method to create promises. Formally, a state promise rule for agent $j \in \{1,\dots,N\}$ generated at time t is a continuous (with respect to the distance d_{func} between set-valued functions, c.f. (1)) map $R_j^{\mathrm{s}} : \mathcal{C}^0\left([t,\infty);\prod_{i\in\mathcal{N}(j)\cup\{j\}}\mathbb{P}^{\mathrm{cc}}(\mathcal{X}_i)\right) \to \mathcal{C}^0\left([t,\infty);\mathbb{P}^{\mathrm{cc}}\left(\mathcal{X}_j\right)\right)$. This means that if agent j must send information to agent i at time t, it sends the state promise $X_j^i[t] = R_j^{\mathrm{s}}(X_{\mathcal{N}}^j[t])$. A control promise rule for agent $j \in \{1,\dots,N\}$ generated at time t is a continuous map $R_j^{\mathrm{c}} : \mathcal{C}^0\left([t,\infty);\prod_{i\in\mathcal{N}(j)\cup\{j\}}\mathbb{P}^{\mathrm{cc}}(\mathcal{X}_i)\right) \to \mathcal{C}^0\left([t,\infty);\mathbb{P}^{\mathrm{c}}\left(\mathcal{U}_j\right)\right)$. This means that when agent j must send information to agent i at time t, it sends the control promise $U_j^i[t] = R_j^{\mathrm{c}}(X_{\mathcal{N}}^j[t])$, from which the state promise is computed by (5). We make the assumption that, in the absence of communication delays or noise in the state measurements, the promises

generated by these rules have the property that $X_j^i[t](t) = \{x_j(t)\}$. Note that because of this fact, it is unnecessary to send the current state $x_j(t)$ in addition to a state promise, since this information is already contained in the promise $X_j^i[t]$. However, when a control promise $U_j^i[t]$ is sent, the current state $x_j(t)$ should also be sent.

Example 1 (Static ball-radius promise rule). Here we describe one simple control promise rule, termed the static ball-radius rule, to create promises that can be described with a finite number of parameters. Given $j \in \{1, \dots, N\}$, a continuous control law $u_j : \prod_{i \in \mathcal{N}(j) \cup \{j\}} \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_i) \to \mathbb{R}^{m_j}$, and $\delta > 0$, the static ball-radius control promise rule for agent j generated at time t is

$$R_j^{\mathrm{c,sb}}(X_{\mathcal{N}}^j[\cdot]_{|[t,\infty)})(t') = \overline{B}(u_j(X_{\mathcal{N}}^j(t)), \delta) \cap \mathcal{U}_j \qquad t' \geq t. \tag{6}$$

Note that this promise is a fixed ball of radius δ in the control space $\mathcal{U}_j$ centered at the control signal used at time t. This promise can be sent with two parameters (assuming δ is known by all agents), the state $x_j(t)$ when the promise was sent, and the control action $u_j(X_{\mathcal{N}}^j(t))$ at that time. •

Having introduced the notion of promise, several observations can be made. First, the availability of promises equips agents with set-valued information models about the state of other agents. This fact makes it necessary to address the definition of distributed controllers that operate on sets, rather than points. We discuss this point in Section 3.2. Then, based on the promises that agent i receives from its neighbors at a given time t, it is responsible for computing the next time it will require updated information. We discuss this in Section 3.3. On the other hand, if at any time agent j breaks its promise to agent i, i.e., its state no longer belongs to its promise set, this triggers an event requiring agent j to send updated information to agent i. We discuss this in Section 3.4.

3.2 Controllers on set-valued information models

Here we briefly discuss the type of controllers that the team-triggered approach relies on. The underlying idea is that since agents possess set-valued information about the state of other agents through promises, controllers themselves should be defined on sets, rather than on points. Our starting point is therefore the availability of a continuous controller of the form $u^{**} : \prod_{j \in \{1,\dots,N\}} \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j) \to \mathbb{R}^m$ that satisfies

$$\nabla_i V(x) \left(A_i x_i + B_i u_i^{**}(\{x\})\right) \leq 0, \quad i \in \{1, \dots, N\}, \tag{7a}$$

$$\sum_{i=1}^{N} \nabla_i V(x) \left(A_i x_i + B_i u_i^{**}(\{x\})\right) < 0. \tag{7b}$$

In other words, if exact, singleton-valued information is available to the agents, then the controller u^{**} guarantees the monotonic evolution of the Lyapunov function V. We assume that u^{**} is distributed over the communication graph $\mathcal{G}$.

As before, this means that for each $i \in \{1, \dots, N\}$, the ith component u_i^{**} can be computed with information in $\prod_{j \in \mathcal{N}(i) \cup \{i\}} \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j)$ rather than in the full space $\prod_{j \in \{1,\dots,N\}} \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j)$.

Controllers of the form described above can be obtained using a number of design methods. We do not enter into the specific details, but briefly mention how one such controller can be derived from the availability of the controller $u^* : \mathcal{X} \to \mathbb{R}^m$ introduced in Section 2. The intuitive idea is that, given the promise information $X_{\mathcal{N}}^i(t)$ that an agent i has about some time t, one can simply apply the controller u^* to any point $y_{\mathcal{N}} \in X_{\mathcal{N}}^i(t)$. This can be formalized as follows: let $E : \prod_{j=1}^N \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j) \to \mathcal{X}$ be a continuous map that is distributed over the communication graph $\mathcal{G}$ and satisfies, for each $i \in \{1, \dots, N\}$, that $E_i(Y) \in Y_i$ for each $Y \in \prod_{j=1}^N \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j)$ and $E_i(\{y\}) = y_i$ for each $y \in \mathcal{X}$. Now, define

$$u^{**}(Y) = u^*(E(Y)). \tag{8}$$

Note that this controller satisfies (7a) and (7b) because u^* satisfies (4a) and (4b).

3.3 Self-triggered information updates

Here, we discuss in detail how agents use the promises received from other agents to generate self-triggered information requests in the future. Let t_{last}^i be some time at which agent i receives updated information (i.e., promises) from its neighbors. Until the next time information is obtained, agent i has access to the collection of functions $X_{\mathcal{N}}^i$ describing its neighbors' states and can compute its own evolution under the controller u^{**} via

$$x_i(t) = e^{A_i(t-t_{\mathrm{last}}^i)} x_i(t_{\mathrm{last}}^i) + \int_{t_{\mathrm{last}}}^t e^{A_i(t-\tau)} B_i u_i^{**}(X_{\mathcal{N}}^i(\tau)) d\tau, \quad t \geq t_{\mathrm{last}}^i. \tag{9}$$

With this in place, agent i can schedule the next time t_{next}^i at which it will need updated information from its neighbors. To do so, we define, for any $Y_{\mathcal{N}} \in \prod_{j \in \mathcal{N}(i) \cup \{i\}} \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j)$,

$$\mathcal{L}_i V^{\sup}(Y_{\mathcal{N}}) = \sup_{y_{\mathcal{N}} \in Y_{\mathcal{N}}} \nabla_i V(y_{\mathcal{N}}) \left(A_i y_i + B_i u_i^{**}(Y_{\mathcal{N}})\right), \tag{10}$$

where y_i is the element of $y_{\mathcal{N}}$ corresponding to i. Using this, we create an implementable trigger that computes when an agent requires new information to guarantee the monotonically decreasing evolution of V. Specifically, the critical time at which information is requested is given by $t_{\mathrm{next}}^i = \max\{t_{\mathrm{last}}^i + T_{\mathrm{d,self}}, t^*\}$, where $T_{\mathrm{d,self}} > 0$ is an a priori chosen parameter that we discuss below and t^* is implicitly defined as the first time $t^* \geq t_{\mathrm{last}}^i$ such that

$$\mathcal{L}_i V^{\sup}(X_{\mathcal{N}}^i(t^*)) = 0. \tag{11}$$

Note that as long as (11) has not yet occurred for all agents $i \in \{1, \dots, N\}$ for some time t and the promises that agents $j \in \mathcal{N}(i)$ have made to i have not

been broken, one can guarantee that

$$\frac{d}{dt}V(x(t)) \leq \sum_{i=1}^{N} \mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t)) < 0.$$

This follows from assumptions (7a) and (7b) and the continuity of (10) when evaluated at the promise sets that each agent possesses.

Although an agent i sends a request REQ for new information from its neighbors $j \in \mathcal{N}(i)$ at time t^i_{next}, it does not mean that it arrives at this time. We assume that the REQ message can be sent reliably with negligible delay since it is a very small message. Given the model for delays and uncertainties, as long as agent i has still not received information from some neighbors j, it will continue to send j requests for new information every $\bar{\Delta}$ seconds. This is because if at time $t^i_{\text{next}} + \bar{\Delta}$ the message still has not arrived, it means the packet was dropped since $\bar{\Delta}$ is the maximum allowable delay.

The parameter $T_{\text{d,self}} > 0$ is known as the *self-triggered dwell time*. We introduce it because, in general, it is possible that $t^* = t^i_{\text{last}}$, implying that instantaneous communication is required. The dwell time is used to prevent this behavior as follows. Note that $\mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t')) \leq 0$ is only guaranteed while $t' \in [t^i_{\text{last}}, t^*]$. Therefore, in case that $t^i_{\text{next}} = t^i_{\text{last}} + T_{\text{d,self}}$, i.e., if $t^* < t^i_{\text{last}} + T_{\text{d,self}}$, the agent uses the safe-mode control during $t' \in (t^*, t^i_{\text{last}} + T_{\text{d,self}}]$ to leave its state fixed. This design ensures the monotonicity of the evolution of the Lyapunov function V along the network execution. The team-triggered controller is therefore defined, for $t \in [t^i_{\text{last}}, t^i_{\text{next}})$, by

$$u_i^{\text{team}}(t) = \begin{cases} u_i^{**}(X^i_{\mathcal{N}}(t)), & \text{if } \mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t)) \leq 0, \\ u_i^{\text{sf}}(x_i(t)), & \text{if } \mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t)) > 0. \end{cases} \tag{12}$$

Next, we discuss how to ensure promises are still valid in the presence of communication noise. To deal with the communication noise, when an agent i receives an estimated promise $\widehat{X}^i_j$ from another agent j, it must be able to create a valid promise X^i_j that contains the promise that agent j intended to send. Note that the specific way of doing this depends on how promises are exchanged between agents. We refer to this action as making a promise set valid. The following example shows how it can be done for the static ball-radius promises described in Example 1.

Example 2 (Valid static ball-radius promise rule with communication noise). In the scenario with communication noise bounded by $\bar{\omega}$, when agent j attempts to send the control promise $\overline{B}(u_j(X^j_{\mathcal{N}}(t)), \delta)$ to agent i at time t as defined in Example 1, it will instead receive $\widehat{U}^i_j[t] = \overline{B}(\widehat{u}^i_j(X^j_{\mathcal{N}}(t)), \delta)$, where $u_j(X^j_{\mathcal{N}}(t)) \in \overline{B}(\widehat{u}^i_j(X^j_{\mathcal{N}}(t)), \bar{\omega})$. To ensure that the promise agent i operates with about agent j contains the true promise made by agent j, it can set

$$U^i_j[t](t') = \overline{B}(\widehat{u}^i_j(X^j_{\mathcal{N}}(t)), \delta + \bar{\omega}) \cap \mathcal{U}_j \qquad t' \geq t.$$

To create the state promise from this agent i would need the true state $x_j(t)$ of j at time t. However, since only the estimate $\widehat{x}_j^i(t)$ is available, we modify (5) by

$$X_j^i[t](t') = \cup_{x_j \in \overline{B}(\widehat{x}_j^i(t), \bar{\omega})} \{z \in \mathcal{X}_j \mid \exists\, u_j : [t,t'] \to \mathcal{U}_j \text{ with } u_j(s) \in U_j^i[t](s)$$
$$\text{for } s \in [t,t'] \text{ such that } z = e^{A_j(t'-t)}x_j + \int_t^{t'} e^{A_j(t'-\tau)}B_j u_j(\tau)d\tau\}. \quad \bullet$$

3.4 Event-triggered information updates

Agent promises may need to be broken for a variety of reasons. For instance, an agent might receive new information from its neighbors and, based on it, decide to change its former plans. Disturbances in the agent dynamics or new requirements imposed by the level of completion of the network task are yet more reasons for why promises might be broken. In the team-triggered approach, promises get updated through events triggered by the agents that decide to break them. Specifically, consider an agent i that made a promise $X_i^j[t]$ to agent j at some time t that is not able to keep after time $t' \geq t$, i.e., $x_i(t') \notin X_i^j[t](t')$. Then, agent i must send a new feasible promise $X_i^j[t']$ to agent j. This event-triggered mechanism requires each agent to keep track of the last promise it made to each one of its neighbors and constantly monitor them to detect when they are broken.

If the message arrived exactly at time t', there would be no issue because agent j will always be operating with correct information, namely that $x_i(t') \in X_i^j[t'](t')$. However, this message might arrive with a delay $\Delta(t')$ or even be dropped altogether. To deal with this, we require agent i to also send agent j a small message WARN at time t' that warns agent j that agent i has broken its promise at time t'. As before, we assume that the message WARN can be sent reliably with negligible delay. With this warning, agent j can make sure it is operating with correct information at all times as follows. Let us first define the notation of a reachable set. Given $y \in \mathcal{X}_i$, let $\mathcal{R}_i(s,y)$ be the reachable set of points under (2) starting from y in s seconds,

$$\mathcal{R}_i(s,y) = \{z \in \mathcal{X}_i \mid \exists\, u_i : [0,s] \to \mathcal{U}_i \text{ s.t. } z = e^{A_i s}y + \int_0^s e^{A_i(s-\tau)}B_i u_i(\tau)d\tau\}.$$

Assuming here that each agent has exact knowledge about the dynamics and control sets of its neighboring agents, each agent can construct, each time the WARN message is received, sets that are guaranteed to contain their neighbors' states. Formally, given the original promise $X_i^j[t]$ received by agent i at time t, and letting t^* be the time at which the WARN message was received from agent i, we then define the new promise for agent i as

$$X_i^j[t^*](t') = \cup_{x_i \in X_i^j[t](t^*)} \mathcal{R}_i(t'-t^*, x_i) \subset \mathcal{X}_i.$$

This new promise $X_i^j[t^*](t')$ is guaranteed to contain $x_i(t')$ for $t' \geq t^*$. As before, if the full message is not received after $\bar{\Delta}$ seconds, agent j will send requests REQ to agent i until the message is successfully transmitted.

Consider an agent $i \in \{1, \dots, N\}$ that has sent a promise $X_i^j[t_{\text{last}}]$ to a neighboring agent j at some time t_{last}. If agent i ends up breaking its promise at time $t^* \geq t_{\text{last}}$, i.e., $x_i(t^*) \notin X_i^j[t_{\text{last}}](t^*)$, then it is responsible for sending a new promise $X_i^j[t_{\text{next}}]$ to agent j at time $t_{\text{next}} = \max\{t_{\text{last}} + T_{\text{d,event}}, t^*\}$, where $T_{\text{d,event}} > 0$ is an a priori chosen parameter that we discuss below. This implies that agent i must keep track of promises made to its neighbors and monitor them in case they are broken. Note that this mechanism is implementable because each agent only needs information about its own state and the promises it has made to determine whether the trigger is satisfied.

The parameter $T_{\text{d,event}} > 0$ is known as the *event-triggered dwell time*. We introduce it because, in general, the time $t^* - t_{\text{last}}$ between when agent i makes and breaks a promise to an agent j might be arbitrarily small. The issue, however, is that if $t^* < t_{\text{last}} + T_{\text{d,event}}$, agent j operates under incorrect information about agent i for $t \in [t^*, t_{\text{last}} + T_{\text{d,event}})$. We deal with this by introducing a warning message WARN that agent i must send to agent j when it breaks its promise at time $t^* < t_{\text{last}} + T_{\text{d,event}}$. If agent j receives such a warning message, it redefines the promise X_j^i as follows,

$$X_i^j[\cdot](t) = \bigcup_{x_i \in X_i^j[\cdot](t^*)} \mathcal{R}(t - t^*, x_i), \tag{13}$$

for $t \geq t^*$, until the new message arrives at time $t_{\text{next}} = t_{\text{last}} + T_{\text{d,event}}$. By definition of the reachable set, the promise $X_i^j[\cdot](t)$ is guaranteed to contain $x_i(t)$ for $t \geq t^*$.

This algorithm ensures that promises are kept at all times. Even if promises are broken, the WARN message allows agents to redefine promises such that they always contain the true states of the relevant agents. The following remark points out minor modifications that would need to be made to the above discussion in the case of modeling uncertainties.

Remark 1 (Modeling uncertainties). In the case of modeling uncertainties or if each agent i does not know exactly the dynamics of its neighbors $j \in \mathcal{N}(i)$, the reachable sets $\mathcal{R}_j(s, y)$ can be replaced by any other set $\widehat{\mathcal{R}}_j(s, y)$ that contains $\mathcal{R}_j(s, y)$ for all $s \geq 0$. •

Combining the controller u_i^{team} with the event- and self-triggered methods of sharing information yields the ROBUST TEAM-TRIGGERED LAW, cf. Algorithm 1.

4 Analysis of the robust team-triggered law

In this section we analyze the convergence, performance, and robustness properties of the ROBUST TEAM-TRIGGERED LAW proposed in Section 3. We begin by noting the monotonic behavior of V with respect to the algorithm executions.

Proposition 1. *The function V is monotonically nonincreasing along the network dynamics* (2) *under the* ROBUST TEAM-TRIGGERED LAW*, with packet drops occurring with some unknown probability $0 \leq p < 1$ messages being delayed by some known maximum delay $\bar{\Delta}$, and communication noise bounded by $\bar{\omega}$.*

Algorithm 1: ROBUST TEAM-TRIGGERED LAW

(Self-triggered information update)
At any time t agent $i \in \{1,\dots,N\}$ receives new promise(s) $\widehat{X}_j^i[t]$ from neighbor(s) $j \in \mathcal{N}(i)$, agent i performs:

create valid promise $X_j^i[t]$ with respect to $\bar{\omega}$
compute own state evolution $x_i(t')$ for $t' \geq t$ using (9)
schedule information request to neighbors in $\max\{t^* - t, T_{\text{d,self}}\}$ seconds
apply controller $u^{\text{team}}(t')$ for $t' \geq t$
while message from j has not been received **do**
  if current time equals $t + \max\{t^* - t, T_{\text{d,self}}\} + k\bar{\Delta}$ for $k \in \mathbb{Z}_{\geq 0}$ **then**
    send agent j a request REQ for new information
  end if
end while

(Respond to information request)
At any time t agent $j \in \mathcal{N}(i)$ requests information, agent i performs:

send new promise $Y_i^j[t] = R_i^{\text{s}}(X_{\mathcal{N}}^i[\cdot]_{|[t,\infty)})$ to agent j

(Event-triggered information update)
At all times t, agent i performs:

if there exists $j \in \mathcal{N}(i)$ such that $x_i(t) \notin Y_i^j[\cdot](t)$ **then**
  send warning message WARN to agent j
  if agent i has sent a promise to j at some time $t_{\text{last}} \in (t - T_{\text{d,event}}, t]$ **then**
    schedule to send new promise $Y_i^j[t_{\text{last}} + T_{\text{d,event}}] = R_i^{\text{s}}(X_{\mathcal{N}}^i[\cdot]_{|[t_{\text{last}}+T_{\text{d,event}},\infty)})$ to agent j in $t_{\text{last}} + T_{\text{d,event}} - t$ seconds
  else
    send new promise $Y_i^j[t] = R_i^{\text{s}}(X_{\mathcal{N}}^i[\cdot]_{|[t,\infty)})$ to agent j
  end if
end if

(Respond to warning message)
At any time t agent $i \in \{1,\dots,N\}$ receives a warning message WARN from agent $j \in \mathcal{N}(i)$

redefine promise set $X_j^i[\cdot](t') = \cup_{x_j^0 \in X_j^i[\cdot](t)} \mathcal{R}_j(t' - t, x_j^0)$ for $t' \geq t$
while message from j has not been received **do**
  if current time equals $t + k\bar{\Delta}$ for $k \in \mathbb{Z}_{\geq 0}$ **then**
    send agent j a request REQ for new information
  end if
end while

Proof. To prove the result, we write the time derivative of the Lyapunov function using the controller u_i^{team} for all agents $i \in \{1,\dots,N\}$,

$$\begin{aligned}\frac{d}{dt}V(x(t)) &= \sum_{i=1}^N \nabla_i V(x_{\mathcal{N}}^i(t))\left(A_i x_i(t) + B_i u_i^{\text{team}}(X_{\mathcal{N}}^i(t))\right) \qquad (14)\\ &\leq \sum_{i=1}^N \sup_{x_{\mathcal{N}}^i(t) \in X_{\mathcal{N}}^i(t)} \nabla_i V(x_{\mathcal{N}}^i(t))\left(A_i x_i(t) + B_i u_i^{\text{team}}(X_{\mathcal{N}}^i(t))\right) \leq 0.\end{aligned}$$

This is guaranteed by the design of the ROBUST TEAM-TRIGGERED LAW. In fact, the WARN messages allow agents to ensure that the information they are operating with is correct, i.e., that promises are valid at all times. When $\mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t)) \leq 0$, the control policy as defined in (12) is $u_i^{\text{team}}(X^i_{\mathcal{N}}(t)) = u_i^{**}(X^i_{\mathcal{N}}(t))$. In this case the summands of (14) are exactly $\mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t)$ as defined in (10). When $\mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t)) > 0$, the control $u_i^{\text{team}}(X^i_{\mathcal{N}}(t)) = u_i^{\text{sf}}(x_i)$, for which the summands of (14) are exactly 0. □

The following result states that, under the ROBUST TEAM-TRIGGERED LAW, Zeno behavior does not occur when using positive dwell times.

Lemma 1 (No Zeno behavior). *Consider a networked cyberphysical system as described in Section 2 executing the* ROBUST TEAM-TRIGGERED LAW*, cf. Algorithm 1, with dwell times $T_{d,self}, T_{d,event} > 0$. Then the network does not exhibit Zeno behavior.*

We are now ready to study the algorithm convergence. For simplicity, we first consider the case with no packet drops, communication delays, or communication noise and later extend the result.

Proposition 2. *Given the network dynamics* (2) *under the* ROBUST TEAM-TRIGGERED LAW *with no packet drops, communication delays, or communication noise, the system asymptotically approaches the set D.*

There are two main challenges in proving Proposition 2. The first is that we need a way to model the asynchronous executions of the agents' actions. The second is that because of the intermittent communications between agents, the memories of the agents evolve discontinuously in time, making it difficult to use standard stability methods of analyzing the trajectories of the system.

Proof sketch. To model the asynchronism, let the time schedule of agent i be given by $\mathcal{T}^i = \{t^i_0, t^i_1, \dots\}$ where t^i_ℓ corresponds to the ℓth time that agent i receives information from one or more of its neighbors. Note that this information can be received because i requests it itself, or j sends it to i because an event is triggered. Analytic synchronization is a procedure of merging together the individual time schedules into a global one $\mathcal{T} = \{t_0, t_1, \dots\}$ by setting

$$\mathcal{T} = \cup_{i=1}^N \mathcal{T}^i.$$

This synchronization is done for analysis purposes only and the time schedules $\mathcal{T}^i$ are not known by the agents themselves. Note that more than one agent may receive information at any given time $t \in \mathcal{T}$.

The information possessed by any given agent are trajectories of sets for each of their neighbors, i.e., promises. For convenience, we denote by

$$S = \prod_{i=1}^N S_i, \quad \text{where}$$
$$S_i = \mathcal{C}^0\Big(\mathbb{R}; \mathbb{P}^{\text{cc}}(\mathcal{X}_1) \times \cdots \times \mathbb{P}^{\text{cc}}(\mathcal{X}_{i-1}) \times \mathcal{X}_i \times \mathbb{P}^{\text{cc}}(\mathcal{X}_{i+1}) \times \cdots \times \mathbb{P}^{\text{cc}}(\mathcal{X}_N)\Big),$$

the space that the state of the entire network lives in. Note that this set allows us to capture the fact that each agent i has perfect information about itself. Although agents only have information about their neighbors, the above space considers agents having promise information about all other agents to facilitate the analysis. This is only done to allow for a simpler technical presentation, and does not impact the validity of the arguments made here. The information possessed by all agents of the networks at some time t is collected in

$$\left(X^1[\cdot]_{|[t,\infty)}, \dots, X^N[\cdot]_{|[t,\infty)}\right) \in S,$$

where $X^i[\cdot]_{|[t,\infty)} = \left(X^i_1[\cdot]_{|[t,\infty)}, \dots, X^i_N[\cdot]_{|[t,\infty)}\right) \in S_i$. We can then formulate the ROBUST TEAM-TRIGGERED LAW discontinuous map of the form $S \times \mathbb{Z}_{\geq 0} \to S \times \mathbb{Z}_{\geq 0}$. This fact makes it difficult to use standard stability methods to analyze the convergence properties of the network. Our approach to this problem consists then of defining a set-valued map $M : S \times \mathbb{Z}_{\geq 0} \rightrightarrows S \times \mathbb{Z}_{\geq 0}$ whose trajectories contain the trajectories of the ROBUST TEAM-TRIGGERED LAW. Although this 'overapproximation procedure' enlarges the set of trajectories to consider, the gained benefit is that of having a set-valued map with suitable continuity properties that is amenable to set-valued stability analysis. We describe this in detail next.

We start by defining the set-valued map M. Let $(Z, \ell) \in S \times \mathbb{Z}_{\geq 0}$. We define the $(N+1)$th component of all the elements in $M(Z, \ell)$ to be $\ell + 1$. The ith component of the elements in $M(Z, \ell)$ is given by one of following possibilities. The first possibility is simply the ith component of Z,

$$\left(Z^i_1[\cdot]_{|[t_{\ell+1},\infty)}, \dots, Z^i_N[\cdot]_{|[t_{\ell+1},\infty)}\right), \tag{15}$$

which models the case when the agent i does not receive any information from its neighbors. The second is

$$\left(Y^i_1[\cdot]_{|[t_{\ell+1},\infty)}, \dots, Y^i_N[\cdot]_{|[t_{\ell+1},\infty)}\right), \tag{16}$$

where for $j \neq i$

$$Y^i_j[\cdot]_{|[t_{\ell+1},\infty)} = \begin{cases} Z^i_j[\cdot]_{|[t_{\ell+1},\infty)}, & \text{if } i \text{ does not receive information from } j, \\ R^{\mathrm{s}}_j(Z^j_{\mathcal{N}}[\cdot]_{|[t_{\ell+1},\infty)}), & \text{otherwise,} \end{cases} \tag{17a}$$

and

$$Y^i_i[\cdot](t) = e^{A_i(t-t_{\ell+1})} Z^i_i(t_{\ell+1}) + \int_{t_{\ell+1}}^{t} e^{A_i(t-\tau)} B_i u^{\mathrm{team}}_i(\tau) d\tau, \quad t \geq t_{\ell+1}, \tag{17b}$$

which models the case when the agent i has received updated information from at least one neighbor (here, with a slight abuse of notation, we use the notation u^{team} to denote the controller evaluated at the set $Y^i[\cdot]$).

Two properties regarding the set-valued map M are worth emphasizing. First, any trajectory of the ROBUST TEAM-TRIGGERED LAW is also a trajectory of the nondeterministic dynamical system defined by M,

$$(Z(t_{\ell+1}), \ell+1) \in M(Z(t_\ell), \ell).$$

Second, unlike the map defined by the ROBUST TEAM-TRIGGERED LAW, which is discontinuous, it can be shown that the set-valued map M is closed (a set-valued map $T : X \rightrightarrows Y$ is closed if $x_k \to x$, $y_k \to y$ and $y_k \in T(x_k)$ imply that $y \in T(x)$). Using this and the monotonicity of the Lyapunov function V, cf. Proposition 1, we can resort to a form of the LaSalle Invariance Principle for set-valued discrete-time dynamical systems to show that all trajectories of T converge to the largest weakly invariant set contained in

$$\begin{aligned} S^* &= \{(Z,\ell) \in S \times \mathbb{Z}_{\geq 0} \mid \exists (Z', \ell+1) \in M(Z,\ell) \text{ such that } V(Z') = V(Z)\}, \\ &= \{(Z,\ell) \in S \times \mathbb{Z}_{\geq 0} \mid \mathcal{L}_i V^{\sup}(Z^i_{\mathcal{N}}) \geq 0 \text{ for all } i \in \{1,\dots,N\}\}. \end{aligned} \tag{18}$$

Using this information, we are able to show that the omega-limit set of any trajectory of M specific to the ROBUST TEAM-TRIGGERED LAW corresponds to the set D. This concludes our proof sketch for Proposition 2.

In the scenario with possible packet drops, communication delays and sensor noise, we are able to state the following result.

Corollary 1. *Consider a networked cyberphysical system as described in Section 2 with packet drops occurring with some unknown probability $0 \leq p < 1$, messages being delayed by some known maximum delay $\bar{\Delta}$, and communication noise bounded by $\bar{\omega}$, executing the* ROBUST TEAM-TRIGGERED LAW *(cf. Algorithm 1) with dwell times $T_{d,self}, T_{d,event} > 0$. Then, any bounded network evolution with uniformly bounded promises asymptotically converges to the neighborhood of D given by*

$$D'(\bar{\Delta},\bar{\omega}) = \{x \in \mathcal{X} \mid \inf_{x^i_{\mathcal{N}}{}' \in \overline{B}(x^i_{\mathcal{N}},\bar{\omega})} \mathcal{L}_i V^{\sup}\Big(\{x_i\} \times \prod_{j\in\mathcal{N}(i)} \bigcup_{y_j \in \overline{B}(x^i_j{}',\bar{\omega})} \mathcal{R}_j(\bar{\Delta}, y_j)\Big) \geq 0, \text{ for all } i \in \{1,\dots,N\}\}, \tag{19}$$

with probability 1.

Note that by equation (7b), the definition (10), and the continuity of u^{**}, D precisely corresponds to $D'(0,0)$. We only provide a proof sketch of the result. Note that, under the hypotheses of the corollary, agents might never know the exact state of themselves or their neighbors at any time. The basic idea is the observation that all properties of M used in the proof of Proposition 2 still hold in the presence of packet drops, delays, and communication noise as long as the time schedule $\mathcal{T}^i$ remains unbounded for each agent $i \in \{1,\dots,N\}$. For this to happen, each agent i must receive an infinite number of messages, and $t^i_\ell \to \infty$. Since packet drops have probability $0 \leq p < 1$, the probability that there is a finite number of updates for any given agent i is 0. Thus, with probability 1,

there are an infinite number of information updates for each agent. Using a similar argument to that in the proof of Proposition 2, one can establish that the bounded trajectories of M still converge to S^* as defined in (18). Finally, one can use this fact to conclude that the omega-limit set of any trajectory of M specific to the ROBUST TEAM-TRIGGERED LAW corresponds to the set $D'(\bar{\Delta}, \bar{\omega})$.

5 Simulations

This section presents simulations of a planar coverage control problem to illustrate the performance of the team-triggered approach and compare it with periodic and self-triggered approaches. Our starting point is the distributed coordination algorithm based on Voronoi partitioning introduced in [30]. The dynamics of each agent is a single integrator

$$\dot{x}_i = u_i, \quad i \in \{1, \dots, N\}, \tag{20}$$

where $\|u_i\|_2 \leq u_{\max}$. Given a convex polygon $Q \subset \mathbb{R}^2$ and some known density function $\phi : Q \to \mathbb{R}_{\geq 0}$, consider the objective function

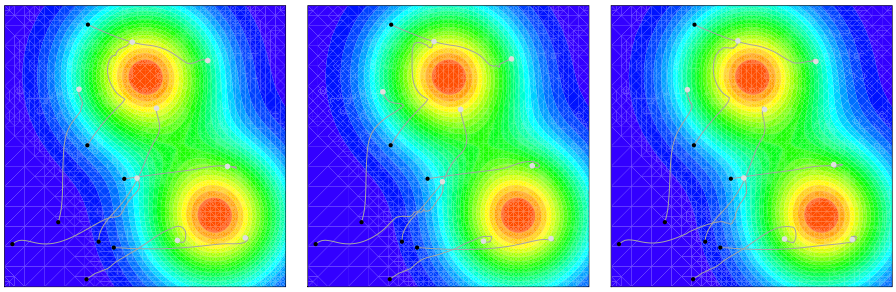
$$\mathcal{H}(x) = E_\phi \left[\min_{i \in \{1,\dots,N\}} \|q - x_i\|^2 \right] = \sum_{i=1}^{N} \int_{V_i} \|q - x_i\|^2 \phi(q) dq. \tag{21}$$

Here, $\{V_1, \dots, V_N\}$ denotes the Voronoi partition of Q, cf. [31]. Roughly speaking, the value $\mathcal{H}$ encodes the expected value of the minimum distance from some agent in the network to a generic point in the polygon, given the density function ϕ. The continuous control law $u^* = (u_1^*, \dots, u_N^*)$ is the gradient of $\mathcal{H}$,

$$u_i^* = -2M_{V_i}(p_i - C_{V_i}),$$

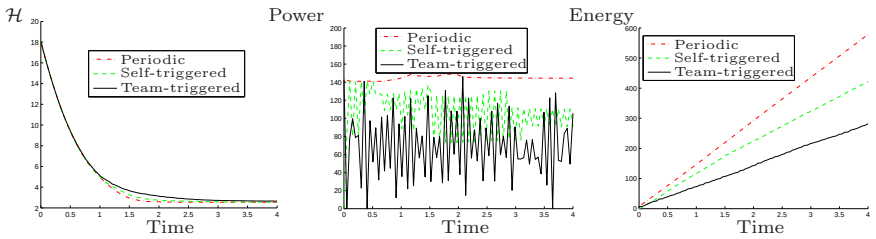
where M_{V_i} and C_{V_i} are the mass and centroid of V_i, respectively. Note that this control law is distributed on the Delaunay graph, i.e., where each agent's neighbors are its Voronoi neighbors. The system (20) under the control law u^* converges to the set of centroidal Voronoi configurations, i.e., configurations where each agent is at the centroid of its own Voronoi cell.

In the following simulations, we consider $N = 8$ agents operating in a square environment of side lengths 4 with $u_{\max} = 1$. The density function is given by $\phi(q) = e^{-\|q-p_1\|_2} + e^{-\|q-p_2\|_2}$, where $p_1 = (2, 3)$ and $p_2 = (3, 1)$. The promises among agents are generated using the static ball-radius rule described in Example 1 with $\delta = 0.5u_{\max}$. The controller we use in the team-triggered approach is defined from u^* using the procedure described in Section 3.2, using $y_j^i = \text{cc}(X_j^i) \subset X_j^i$ for each $j \in \mathcal{N}(i)$ (here, $\text{cc}(S)$ is the circumcenter of S). The dwell time in the team-triggered execution is $T_{\text{d,self}} = 0.05$. According to Corollary 1, under communication delays bounded by $\bar{\Delta}$ and sensor noise bounded by $\bar{\omega}$, the system converges to a neighborhood of the set of centroidal Voronoi configurations. In this case, one can actually provide a characterization of this asymptotic behavior as follows: in the limit, each agent is within $2(\bar{\Delta}u_{\max} + \bar{\omega})$ of the centroid of its own Voronoi cell.



(a) (b) (c)

Fig. 1. Executions of the (a) periodic, (b) self-triggered, and (c) team-triggered implementations of the gradient-based continuous controller for optimal deployment in [30]. The black and gray dots correspond to initial and final conditions, respectively.



(a) (b) (c)

Fig. 2. (a) shows the evolution of the objective function (21), (b) shows the communication power (Watts) consumption over time and (b) shows the total transmission energy used (Joules) for the three executions in Figure 1.

Figure 1 shows network executions under periodic, self-triggered, and team-triggered implementations of the controller. This figure also compares the evolution of the objective function (21). Figure 2 compares the total energy used to transmit messages by the entire network as the system evolves. For each agent $i \in \{1, \dots, N\}$, we quantify the power $\mathcal{P}_i$ used by i to communicate using [32],

$$\mathcal{P}_i = 10 \log_{10} \Big[\sum_{j \in \{1,\dots,n\}, i \neq j}^{n} \beta 10^{0.1 P_{i \to j} + \alpha \|x_i - x_j\|_2} \Big],$$

where $\alpha > 0$ and $\beta > 0$ depend on the characteristics of the wireless medium and $P_{i \to j}$ is the power received by j of the signal transmitted by i. In our simulations, all these values are set to 1. We can see from Figure 2(b) that the total amount of transmission energy used with the team-triggered approach is significantly less than those of the periodic and self-triggered approaches. Remarkably, this comes without compromising the stability of the system, cf. Figure 1. For instance, Figure 1(b) shows that the speed of convergence is a little slower in the triggered strategies, but yields a large amount of savings in terms of message transmission energy.

We conclude this section by illustrating how tightness of promises affect the performance of the network. We do this by varying the parameter δ in the definition (6) of the static-ball radius rule. This parameter captures how large the promise sets that agents send to each other are. We define $\lambda = \frac{\delta}{2}$, so that $\lambda = 0$ corresponds to exact information (the control promise is a point in the allowable control set) and $\lambda = 1$ corresponds to no promises at all (the control promise is the entire allowable control set). Note that the latter case exactly corresponds to a self-triggered strategy because agents are simply using reachability sets about their neighbors as the promises. Figure 3 shows the average power consumption and the time to converge to 99% of the final value of the objective function for varying tightness on the promises. Note that for small λ, the amount of energy required for sending messages is minimal while the time to convergence only increases slightly.

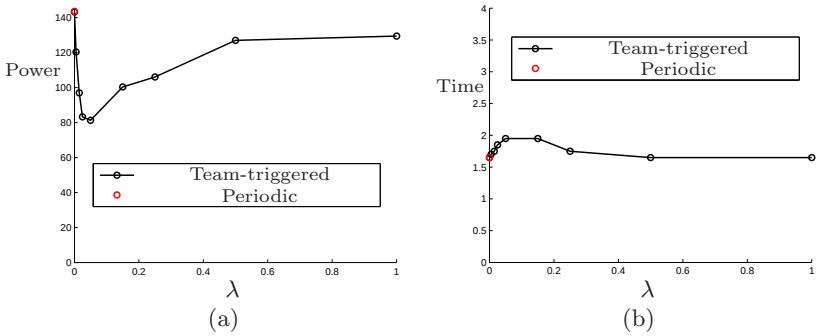


Fig. 3. Implementation of the team-triggered strategy with varying tightness of promises. Plot (a) shows average communication power consumption (Watts) by the whole network and (b) shows time to converge to 99% of the final value (seconds). The parameter λ captures tightness of promises, with $\lambda = 0$ corresponding to exact information and $\lambda = 1$ corresponding to the self-triggered case (no promises at all, just the description of the reachability set).

6 Conclusions

We have proposed a novel approach, termed team-triggered, for the real-time control of networked cyberphysical systems. When information between subsystems is obtained through wireless communication, event-triggered strategies may be costly to implement because they need continuous availability of information to check the triggers, and self-triggered strategies are conservative because they tend to generate more communications than strictly necessary. The ROBUST TEAM-TRIGGERED LAW combines ideas from both event- and self-triggered control into a unified paradigm that incorporates their strengths while maintaining desired levels of performance. The backbone of the team-triggered approach is

the quality of promises that agents make to one another. Future work will be devoted to tuning the generation of promises to optimize system properties such as communication energy and time to convergence, analyzing the algorithm performance under other sources of errors such as disturbances in the dynamics, and relaxing our assumptions to make the approach more generally applicable.

Acknowledgments

This research was partially supported by NSF award CCF-0917166.

References

1. Kim, K.D., Kumar, P.R.: Cyberphysical systems: A perspective at the centennial. Proceedings of the IEEE **100** (2012) 1287–1308
2. Sztipanovits, J., Koutsoukos, X., Karsai, G., Kottenstette, N., Antsaklis, P., Gupta, V., Goodwine, B., Baras, J., Wang, S.: Toward a science of cyberphysical system integration. Proceedings of the IEEE **100** (2012) 29–44
3. Hristu, D., Levine, W.: Handbook of Networked and Embedded Control Systems. Birkhäuser, Boston, MA (2005)
4. Åström, K.J., Wittenmark, B.: Computer Controlled Systems: Theory and Design. 3rd edn. Prentice Hall, Englewood Cliffs, NJ (1996)
5. Laila, D.S., Nesic, D., Astolfi, A.: Sampled-data control of nonlinear systems. In Loria, A., Lamnabhi-Lagarrigue, F., Panteley, E., eds.: Advanced Topics in Control Systems Theory: Lecture Notes from FAP. Volume 328. Springer, New York (2005) 91–137
6. Wan, P., Lemmon, M.D.: Event-triggered distributed optimization in sensor networks. In: Symposium on Information Processing of Sensor Networks, San Francisco, CA (2009) 49–60
7. Eqtami, A., Dimarogonas, D.V., Kyriakopoulos, K.J.: Event-triggered control for discrete-time systems. In: American Control Conference, Baltimore, MD (2010) 4719–4724
8. Åström, K.J., Bernhardsson., B.M.: Comparison of Riemann and Lebesgue sampling for first order stochastic systems. In: IEEE Conf. on Decision and Control, Las Vegas, NV (2002) 2011–2016
9. Tabuada, P.: Event-triggered real-time scheduling of stabilizing control tasks. IEEE Transactions on Automatic Control **52** (2007) 1680–1685
10. Heemels, W.P.M.H., Sandee, J.H., van den Bosch, P.P.J.: Analysis of event-driven controllers for linear systems. International Journal of Control **81** (2008) 571–590
11. Velasco, M., Marti, P., Fuertes, J.M.: The self triggered task model for real-time control systems. In: Proceedings of the 24th IEEE Real-Time Systems Symposium. (2003) 67–70
12. Subramanian, R., Fekri, F.: Sleep scheduling and lifetime maximization in sensor networks. In: Symposium on Information Processing of Sensor Networks, New York, NY (2006) 218–225
13. Wang, X., Lemmon, M.D.: Self-triggered feedback control systems with finite-gain L_2 stability. IEEE Transactions on Automatic Control **54** (2009) 452–467
14. Anta, A., Tabuada, P.: To sample or not to sample: self-triggered control for nonlinear systems. IEEE Transactions on Automatic Control **55** (2010) 2030–2042

15. Mazo Jr., M., Tabuada, P.: Decentralized event-triggered control over wireless sensor/actuator networks. IEEE Transactions on Automatic Control **56** (2011) 2456–2461
16. Wang, X., Hovakimyan, N.: L_1 adaptive control of event-triggered networked systems. In: American Control Conference, Baltimore, MD (2010) 2458–2463
17. Donkers, M.C.F., Heemels, W.P.M.H.: Output-based event-triggered control with guaranteed L_∞-gain and improved and decentralised event-triggering. IEEE Transactions on Automatic Control **57** (2012) 1362–1376
18. Dimarogonas, D.V., Frazzoli, E., Johansson, K.H.: Distributed event-triggered control for multi-agent systems. IEEE Transactions on Automatic Control **57** (2012) 1291–1297
19. Shi, G., Johansson, K.H.: Multi-agent robust consensus-part II: application to event-triggered coordination. In: IEEE Conf. on Decision and Control, Orlando, FL (2011) 5738–5743
20. Mazo Jr., M., Tabuada, P.: On event-triggered and self-triggered control over sensor/actuator networks. In: IEEE Conf. on Decision and Control, Cancun, Mexico (2008) 435–440
21. Eqtami, A., Dimarogonas, D.V., Kyriakopoulos, K.J.: Event-triggered strategies for decentralized model predictive controllers. In: IFAC World Congress, Milano, Italy (2011)
22. Garcia, E., Antsaklis, P.J.: Model-based event-triggered control for systems with quantization and time-varying network delays. IEEE Transactions on Automatic Control **58** (2013) 422–434
23. Heemels, W.P.M.H., Donkers, M.C.F.: Model-based periodic event-triggered control for linear systems. Automatica **49** (2013) 698–711
24. Nowzari, C., Cortés, J.: Self-triggered coordination of robotic networks for optimal deployment. Automatica **48** (2012) 1077–1087
25. Wang, X., Lemmon, M.D.: Event-triggering in distributed networked control systems. IEEE Transactions on Automatic Control **56** (2011) 586–601
26. Wang, X., Sun, Y., Hovakimyan, N.: Relaxing the consistency condition in distributed event-triggered networked control systems. In: IEEE Conf. on Decision and Control, Atlanta, GA (2010) 4727–4732
27. Zhong, M., Cassandras, C.G.: Asynchronous distributed optimization with event-driven communication. IEEE Transactions on Automatic Control **55** (2010) 2735–2750
28. Nowzari, C., Cortés, J.: Team-triggered coordination for real-time control of networked cyberphysical systems. IEEE Transactions on Automatic Control (2013) Submitted.
29. Guinaldo, M., Lehmann, D., Moreno, J.S., Dormido, S., Johansson, K.H.: Distributed event-triggered control with network delays and packet losses. In: IEEE Conf. on Decision and Control, Hawaii, USA (2012) 1–6
30. Cortés, J., Martínez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Transactions on Robotics and Automation **20** (2004) 243–255
31. Okabe, A., Boots, B., Sugihara, K., Chiu, S.N.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. 2 edn. Wiley Series in Probability and Statistics. Wiley (2000)
32. Firouzabadi, S.: Jointly optimal placement and power allocation in wireless networks. Master's thesis, University of Maryland at College Park (2007)