Team-triggered coordination for real-time control of networked cyber-physical systems

Cameron Nowzari Jorge Cortés

***Abstract*—This paper studies the real-time implementation of distributed controllers on networked cyber-physical systems. We build on the strengths of event- and self-triggered control to synthesize a unified approach, termed team-triggered, where agents make promises to one another about their future states and are responsible for warning each other if they later decide to break them. The information provided by these promises allows individual agents to autonomously schedule information requests in the future and sets the basis for maintaining desired levels of performance at lower implementation cost. We establish provably correct guarantees for the distributed strategies that result from the proposed approach and examine their robustness against delays, packet drops, and communication noise. The results are illustrated in simulations of a multi-agent formation control problem.**

I. Introduction

A growing body of work studies the design and real-time implementation of distributed controllers to ensure the efficient and robust operation of networked cyber-physical systems. In multi-agent scenarios, energy consumption is correlated with the rate at which sensors take samples, processors recompute control inputs, actuator signals are transmitted, and receivers are left on listening for potential incoming signals. Performing these tasks periodically is costly, might lead to inefficient implementations, or face hard physical constraints. 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, the emphasis is instead on developing tests that rely only on current information available to individual agents to schedule future actions. Event-triggered strategies generally result in less samples or controller updates but, when executed over networked systems, may be costly to implement because of the need for continuous availability of the information required to check the triggers. Self-triggered strategies are more easily amenable to distributed implementation but result in conservative executions because of the overapproximation by individual agents about the state of the environment and the network. These strategies might be also beneficial in scenarios where leaving receivers on to listen to potential messages is costly. Our objective in this paper is to build on the strengths of event- and self-triggered control to synthesize a unified approach for controlling networked systems in real time 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 cyber-physical systems cannot be overemphasized [3], [4]. Real-time controller implementation is an area of extensive research including periodic [5], [6], event-triggered [7], [8], [9], [10], and self-triggered [11], [12], [13] 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 cyber-physical 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. [14], [15], [16]. Fewer works have considered scenarios where multiple plants or agents together are the subject of the overall control design. Exceptions include consensus via event-triggered [17], [18], [19] or self-triggered control [17], [20], rendezvous [21], model predictive control [22], and model-based event-triggered control [23], [24]. The event-triggered controller designed in [17] for a decentralized system with multiple plants requires agents to have continuous information about each others' states. The works in [17], [25] implement self-triggered communication schemes to perform distributed control where agents assume worst-case conditions for other agents when deciding when new information should be obtained. Distributed strategies based on event-triggered communication and control are explored in [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 works [23], [28], [29] are closer in spirit to the ideas presented here. In the interconnected system considered in [23], each subsystem helps neighboring subsystems by monitoring their estimates and ensuring that they stay within some performance bounds. The approach requires different subsystems to have synchronized estimates of one another even though they do not communicate at all times. In [28], [29], agents do not have continuous availability of information from neighbors and instead decide when to broadcast new information to them.

Statement of contributions: We propose a novel scheme for the real-time control of networked cyber-physical systems that combines ideas from event- and self-triggered control.

Preliminary versions of this paper have appeared as [1] and [2].

The authors are with the Department of Mechanical and Aerospace Engineering, University of California, San Diego, California, USA, {cnowzari,cortes}@ucsd.edu

Our approach is based on agents making promises to one another about their future states and being responsible for warning each other if they later decide to break them. This is reminiscent of event-triggered implementations. Promises can be broad, from tight state trajectories to loose descriptions of reachability sets. With the information provided by promises, individual agents can autonomously determine when in the future fresh information will be needed to maintain a desired level of performance. This is reminiscent of self-triggered implementations. 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 require the continuous availability of the information necessary to check the relevant triggers. Second, because of the extra information provided by promises about what other agents plan to do, agents can generally wait longer periods of time before requesting new information and operate more efficiently than if only worst-case scenarios are assumed, as is done in self-triggered control. Less overall communication is beneficial in reducing the total network load and decreasing chances of communication delays or packet drops due to network congestion. Lastly, we provide theoretical guarantees for the correctness and performance of team-triggered strategies implemented over distributed networked systems. Our technical approach makes use of set-valued analysis, invariance sets, and Lyapunov stability. We also show that, in the presence of physical sources of error and under the assumption that 1-bit messages can be sent reliably with negligible delay, the team-triggered approach can be slightly modified to be robust to delays, packet drops, and communication noise. Interestingly, the self-triggered approach can be seen as a particular case of the team-triggered approach where promises among agents simply consist of their reachability sets (and hence do not actually constrain their state). We illustrate the convergence and robustness results through simulation in a multi-agent formation control problem, paying special attention to the implementation costs and the role of the tightness of promises in the algorithm performance.

Organization: Section II lays out the problem of interest. Section III briefly reviews current real-time implementation approaches based on agent triggers. Section IV presents the team-triggered approach for networked cyber-physical systems. Sections V and VI analyze the correctness and robustness, respectively, of team-triggered strategies. Simulations illustrate our results in Section VII. Finally, Section VIII 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 of a vector is $\|\cdot\|_2$. 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 $A_i \in \mathbb{R}^{m_i \times n_i}$ 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 $\mathbb{P}^{\text{c}}(S)$, respectively $\mathbb{P}^{\text{cc}}(S)$, denote the collection of compact, respectively, compact and connected, subsets of S. The Hausdorff distance between $S_1, S_2 \subset \mathbb{R}^d$ 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 bounded set-valued functions $C_1, C_2 \in \mathcal{C}^0(I\subset\mathbb{R};\mathbb{P}^{\text{c}}(\mathbb{R}^d))$, its distance is

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

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 $(j,i)\in E$. The set of neighbors of a vertex i is $\mathcal{N}(i)=\{j\in V \mid (i,j)\in E\}$. Given $v\in\prod_{i=1}^N \mathbb{R}^{n_i}$, we let $v_{\mathcal{N}}^i = (v_i,\{v_j\}_{j\in\mathcal{N}(i)})$ denote the components of v that correspond to vertex i and its neighbors in $\mathcal{G}$.

II. Network modeling and problem statement

We consider a distributed control problem carried out over an 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 agents i can communicate with are its neighbors $\mathcal{N}(i)$ in $\mathcal{G}$. The state of $i\in\{1,\dots,N\}$, 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 $x_{\mathcal{N}}^i$ when it communicates with its neighbors. By assumption, 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 the existence of a *safe-mode* controller $u_i^{\text{sf}}:\mathcal{X}_i\to\mathcal{U}_i$,

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

i.e., a controller able to keep agent i's state fixed. The existence of a safe-mode controller for a general controlled system may seem restrictive, but there exist many cases, including nonlinear systems, that admit one, such as single integrators or vehicles with unicycle dynamics. Letting $u=(u_1,\dots,u_N)\in\mathcal{U}=\prod_{i=1}^N\mathcal{U}_i$, the dynamics can be described by

$$\dot{x} = Ax + Bu, \tag{3}$$

with $A = \operatorname{diag}(A_1,\dots,A_N)\in\mathbb{R}^{n\times n}$ and $B = \operatorname{diag}(B_1,\dots,B_N)\in\mathbb{R}^{n\times m}$, where $n=\sum_{i=1}^N n_i$, and $m=\sum_{i=1}^N m_i$. We refer to the team of agents with communication topology $\mathcal{G}$ and dynamics (3), where each agent has a safe-mode controller and access to its own state at all times, as a *networked cyber-physical system*. The goal is to drive the agents' states to some desired closed set of configurations $D\subset\mathcal{X}$ and ensure that it stays there. Depending on how D is defined, this objective can capture different coordination tasks, including deployment, rendezvous, and formation control. The goal 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.

Given the agent dynamics, the communication graph $\mathcal{G}$, and the set D, our starting point is the availability of a control law that drives the system asymptotically to D. Formally, we assume that a continuous map $u^* : \mathcal{X} \to \mathcal{U}$ and a continuously differentiable function $V : \mathcal{X} \to \mathbb{R}$, bounded from below exist such that D is the set of minimizers of V and, for all $x \notin D$,

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

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

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 $x_{\mathcal{N}}^i$, rather than on the full network state x. For simplicity, and with a slight abuse of notation, we write $u_i^*(x_{\mathcal{N}}^i) \in \mathcal{U}_i$ and $\nabla_i V(x_{\mathcal{N}}^i) \in \mathbb{R}^{n_i}$ to emphasize this fact when convenient. This property has the important consequence that agent i can compute these quantities with the exact information it can obtain through communication on $\mathcal{G}$.

Remark II.1 (Assumption on non-negative contribution of each agent to task completion) Note that (4b) simply states that V is a Lyapunov function for the closed-loop system. Instead, (4a) is a more restrictive assumption that essentially states that each agent does not individually contribute in a negative way to the evolution of the Lyapunov function. This latter assumption can in turn be relaxed [14] by selecting parameters $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$ with $\sum_{i=1}^{N} \alpha_i = 0$ (note that some α_i would be positive and others negative) and specifying instead that, for each $i \in \{1, \ldots, N\}$, the left-hand side of (4a) should be less than or equal to α_i. Along these lines, one could envision the design of distributed mechanisms to dynamically adjust these parameters, but we do not go into details here for space reasons. •

From an implementation viewpoint, the controller u^* requires continuous agent-to-agent communication and continuous updates of the actuator signals, making it unfeasible for practical scenarios. In the next section we develop a self-triggered communication and control strategy to address the issue of selecting time instants for information sharing.

III. Self-triggered communication and control

This section provides an overview of the self-triggered communication and control approach to solve the problem described in Section II. In doing so, we also introduce several concepts that play an important role in our discussion later. The general idea is to guarantee that the time derivative of the Lyapunov function V along the trajectories of the networked cyber-physical system (3) is less than or equal to 0 at all times, even when the information used by the agents is inexact.

To model the case that agents do not have perfect information about each other at all times, we let each agent $i \in \{1, \ldots, N\}$ keep an estimate $\widehat{x}_j^i$ of the state of each of its neighbors $j \in \mathcal{N}(i)$. Since i always has access to its own state, $\widehat{x}_{\mathcal{N}}^i(t) = (x_i(t), \{\widehat{x}_j^i(t)\}_{j \in \mathcal{N}(i)})$ is the information available to agent i at time t. Since agents do not have access to exact information at all times, they cannot implement the controller u^* exactly, but instead use the feedback law

$$u_i^{\text{self}}(t) = u_i^*(\widehat{x}_{\mathcal{N}}^i(t)).$$

We are now interested in designing a triggering method such that agent i can decide when $\widehat{x}_{\mathcal{N}}^i(t)$ needs to be updated. Let t_{last} be the last time at which all agents have received information from their neighbors. Then, the time t_{next} at which the estimates should be updated is when

$$\begin{aligned}&\frac{d}{dt} V(x(t_{\text{next}})) \\ &\quad = \sum_{i=1}^{N} \nabla_i V(x(t_{\text{next}})) \left(A_i x_i(t_{\text{next}}) + B_i u_i^{\text{event}}(t_{\text{last}})\right) = 0.\end{aligned} \tag{5}$$

Unfortunately, (5) requires global information and cannot be checked in a distributed way. Instead, one can define a local event that defines when a single agent $i \in \{1, \ldots, N\}$ should update its information as any time that

$$\nabla_i V(x(t)) \left(A_i x_i(t) + B_i u_i^{\text{self}}(t)\right) = 0. \tag{6}$$

As long as each agent i can ensure the local event (6) has not yet occurred, it is guaranteed that (5) has not yet occurred either. The problem with this approach is that each agent $i \in \{1, \ldots, N\}$ needs to have continuous access to information about the state of its neighbors $\mathcal{N}(i)$ in order to evaluate $\nabla_i V(x) = \nabla_i V(x_{\mathcal{N}}^i)$ and check condition (6). The self-triggered approach removes this requirement on continuous availability of information by having each agent employ instead the possibly inexact information about the state of their neighbors. The notion of reachability set plays a key role in achieving this. Given $y \in \mathcal{X}_i$, the *reachable set* of points under (2) starting from y in s seconds is,

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

Using this notion, if agents have exact knowledge about the dynamics and control sets of its neighboring agents (but not their controllers), each agent can construct, each time state information is received, sets that are guaranteed to contain their neighbors' states.

Definition III.1 (Guaranteed sets) If t_{last}^i is the time at which agent i receives state information $x_j(t_{\text{last}}^i)$ from its neighbor $j \in \mathcal{N}(i)$, then the *guaranteed set* is given by

$$\mathbf{X}_j^i(t, t_{\text{last}}^i, x_j(t_{\text{last}}^i)) = \mathcal{R}_j(t - t_{\text{last}}^i, x_j(t_{\text{last}}^i)) \subset \mathcal{X}_j, \tag{7}$$

and is guaranteed to contain $x_j(t)$ for $t \geq t_{\text{last}}^i$.

We let $\mathbf{X}_j^i(t) = \mathbf{X}_j^i(t, t_{\text{last}}^i, x_j(t_{\text{last}}^i))$ when the starting state $x_j(t_{\text{last}}^i)$ and time t_{last}^i do not need to be emphasized. We denote by $\mathbf{X}_{\mathcal{N}}^i(t) = (x_i(t), \{\mathbf{X}_j^i(t)\}_{j \in \mathcal{N}(i)})$ the information available to agent i at time t.

Remark III.2 (Computing reachable sets) Finding the guaranteed or reachable sets (7) can be in general computationally expensive. A common approach consists of computing over-approximations to the actual reachable set via convex polytopes or ellipsoids. There exist efficient algorithms to calculate and store these for various classes of systems, see e.g., [30], [31]. Furthermore, agents can deal with situations where they do not have exact knowledge about the dynamics of their neighbors (so that the guaranteed sets cannot be computed exactly) by employing overapproximations of the actual guaranteed sets. •

With the guaranteed sets in place, we can now provide a test that allows agents to determine when they should update their current information and control signals. At time t^i_{last}, agent i computes the next time $t^i_{\text{next}} \geq t^i_{\text{last}}$ to acquire information via

$$\sup_{y_{\mathcal{N}} \in \mathbf{X}^i_{\mathcal{N}}(t^i_{\text{next}})} \nabla_i V(y_{\mathcal{N}}) \left(A_i x_i(t^i_{\text{next}}) + B_i u^{\text{self}}_i(t^i_{\text{next}}) \right) = 0. \quad (8)$$

By (4a) and the fact that $\mathbf{X}^i_j(t^i_{\text{last}}) = \{x_j(t^i_{\text{last}})\}$, at time t^i_{last},

$$\begin{aligned} \sup_{y_{\mathcal{N}} \in \mathbf{X}^i_{\mathcal{N}}(t^i_{\text{last}})} & \nabla_i V(y_{\mathcal{N}}) \left(A_i x_i(t^i_{\text{last}}) + B_i u^{\text{self}}_i(t^i_{\text{last}}) \right) \\ &= \nabla_i V(x^i_{\mathcal{N}}(t^i_{\text{last}})) \left(A_i x_i(t^i_{\text{last}}) + B_i u^{\text{self}}_i(t^i_{\text{last}}) \right) \leq 0. \end{aligned}$$

If all agents use this triggering criterion for updating information, it is guaranteed that $\frac{d}{dt} V(x(t)) \leq 0$ at all times because, for each $i \in \{1, \dots, N\}$, the true state $x_j(t)$ is guaranteed to be in $\mathbf{X}^i_j(t)$ for all $j \in \mathcal{N}(i)$ and $t \geq t^i_{\text{last}}$.

The condition (8) is appealing because it can be evaluated by agent i with the information it possesses at time t^i_{last}. Once determined, agent i schedules that, at time t^i_{next}, it will request updated information from its neighbors. We refer to $t^i_{\text{next}} - t^i_{\text{last}}$ as the *self-triggered request time* for agent i. Due to the conservative way in which t^i_{next} is determined, it is possible that $t^i_{\text{next}} = t^i_{\text{last}}$ for some i, which would mean that instantaneous information updates are necessary (note that this cannot happen for all $i \in \{1, \dots, N\}$ unless the network state is already in D). This can be dealt with by introducing a dwell time such that a minimum amount of time must pass before an agent can request new information and using the safe-mode controller while waiting for the new information. We do not enter into details here and defer the discussion to Section IV-C.

The problem with the self-triggered approach is that the resulting times are often conservative because the guaranteed sets can grow large quickly as they capture all possible trajectories of neighboring agents. It is conceivable that improvements can be made from tuning the guaranteed sets based on what neighboring agents *plan* to do rather than what they *can* do. This observation is at the core of the team-triggered approach proposed next.

IV. Team-triggered coordination

This section presents the team-triggered approach for the real-time implementation of distributed controllers on networked cyber-physical systems. The team-triggered approach incorporates the reactive nature of event-triggered approaches and, at the same time, endows individual agents with the autonomy characteristic of self-triggered approaches to determine when and what information is needed. Agents make promises to their neighbors about their future states and inform them if these promises are violated later (hence the connection with event-triggered control). With the extra information provided by the availability of the promises, each agent computes the next time that an update is required and requests information from their neighbors accordingly to guarantee the monotonicity of the Lyapunov function V introduced in Section II (hence the connection with self-triggered control).

A. 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.

Definition IV.1 (State promises and rules) A *state promise* that agent $j \in \{1, \dots, N\}$ makes to agent i at time t is a set-valued, continuous (with respect to the Hausdorff distance) function $X^i_j[t] \in \mathcal{C}^0([t,\infty); \mathbb{P}^{\text{cc}}(\mathcal{X}_j))$. 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} defined in (1)) map of the form $R^{\text{s}}_j : \mathcal{C}^0 \left([t,\infty); \prod_{i \in \mathcal{N}(j) \cup \{j\}} \mathbb{P}^{\text{cc}}(\mathcal{X}_i) \right) \to \mathcal{C}^0 \left([t,\infty); \mathbb{P}^{\text{cc}}(\mathcal{X}_j) \right)$.

The notation $X^i_j[t](t')$ conveys the promise $x_j(t') \in X^i_j[t](t')$ that agent j makes at time t to agent i about time $t' \geq t$. A state promise rule is simply a way of generating state promises. This means that if agent j must send information to agent i at time t, it sends the state promise $X^i_j[t] = R^{\text{s}}_j(X^j_{\mathcal{N}}[\cdot]_{|[t,\infty)})$. We require that, in the absence of communication delays or noise in the state measurements, the promises generated by a rule have the property that $X^i_j[t](t) = \{x_j(t)\}$. For simplicity, when the time at which a promise is received is not relevant, we use the notation $X^i_j[\cdot]$, or simply X^i_j. All promise information available to agent $i \in \{1, \dots, N\}$ at some time t is given by $X^i_{\mathcal{N}}[\cdot]_{|[t,\infty)} = (x_{i|[t,\infty)}, \{X^i_j[\cdot]_{|[t,\infty)}\}_{j \in \mathcal{N}(i)}) \in \mathcal{C}^0 \left([t,\infty); \prod_{j \in \mathcal{N}(i) \cup \{i\}} \mathbb{P}^{\text{cc}}(\mathcal{X}_j) \right)$. To extract information from this about a specific time t', we use $X^i_{\mathcal{N}}[\cdot](t')$ or simply $X^i_{\mathcal{N}}(t') = (x_i(t'), \{X^i_j[\cdot](t')\}_{j \in \mathcal{N}(i)}) \in \prod_{j \in \mathcal{N}(i) \cup \{i\}} \mathbb{P}^{\text{cc}}(\mathcal{X}_j)$. The generality of the above definitions allow promise sets to be arbitrarily complex but we restrict ourselves to promise sets that can be described with a finite number of parameters.

Remark IV.2 (Example promise and rule) Alternative to directly sending state promises, agents can share their promises based on their control rather than their state. The notation $U^i_j[t](t')$ conveys the promise $u_j(t') \in U^i_j[t](t')$ that agent j makes at time t to agent i about time $t' \geq t$. Given the dynamics of agent j and state $x_j(t)$ at time t, agent i can compute the state promise for $t' \geq t$,

$$\begin{aligned} X^i_j[t](t') = \{ & z \in \mathcal{X}_j \mid \exists\, u_j : [t,t'] \to \mathcal{U}_j \text{ with} \\ & u_j(s) \in U^i_j[t](s) \text{ for } s \in [t,t'] \text{ such that} \\ & z = e^{A_j(t'-t)} x_j(t) + \int_t^{t'} e^{A_j(t'-\tau)} B_j u_j(\tau) d\tau \}. \quad (9) \end{aligned}$$

As an example, given $j \in \{1, \dots, N\}$, a continuous control law $u_j : \prod_{i \in \mathcal{N}(j) \cup \{j\}} \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_i) \to \mathcal{U}_j$, and $\delta_j > 0$, the *ball-radius control promise rule* for agent j generated at time t is

$$R_j^{\mathrm{cb}}(X_{\mathcal{N}}^j[\cdot]_{|[t,\infty)})(t') = \overline{B}(u_j(X_{\mathcal{N}}^j(t)), \delta_j) \cap \mathcal{U}_j \ t' \geq t. \quad (10)$$

Note that this promise is a ball of radius δ_j in the control space $\mathcal{U}_j$ centered at the control signal used at time t. Depending on whether δ_j is constant or changes with time, we refer to it as the static or dynamic ball-radius rule, respectively. The promise can be sent with three parameters, the state $x_j(t)$ when the promise was sent, the control action $u_j(X_{\mathcal{N}}^j(t))$ at that time, and the radius δ_j of the ball. The state promise can then be generated using (9). •

Promises allow agents to predict the evolution of their neighbors more accurately, which directly affects the network behavior. In general, tight promises correspond to agents having good information about their neighbors, which at the same time may result in an increased communication effort (since the promises cannot be kept for long periods of time). On the other hand, loose promises correspond to agents having to use more conservative controls due to the lack of information, while at the same time potentially being able to operate for longer periods of time without communicating (because promises are not violated).

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 in Section IV-B. The additional information that promises represent is beneficial to the agents because it decreases the amount of uncertainty when making action plans. Section IV-C discusses this in detail. Finally, these advantages rely on the assumption that promises hold throughout the evolution. As the state of the network changes and the level of task completion evolves, agents might decide to break former promises and make new ones. We examine this in Section IV-D.

B. Controllers on set-valued information models

Here we 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. There are different ways of designing controllers that operate with set-valued information depending on the type of system, its dynamics, or the desired task, see e.g., [32]. For the problem of interest here, we offer the following possible goals. One may be interested in simply decreasing the value of a Lyapunov function as fast as possible, at the cost of more communication or sensing. Alternatively, one may be interested in choosing the stabilizing controller such that the amount of required information is minimal at a cost of slower convergence time. We consider continuous (with respect to the Hausdorff distance) controllers of the form $u^{**} : \prod_{j \in \{1,\dots,N\}} \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j) \to \mathbb{R}^m$ that satisfy

$$\nabla_i V(x)\,(A_i x_i + B_i u_i^{**}(\{x\})) \leq 0, \ i \in \{1, \dots, N\}, \quad (11a)$$

$$\sum_{i=1}^{N} \nabla_i V(x)\,(A_i x_i + B_i u_i^{**}(\{x\})) < 0. \quad (11b)$$

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 above form can be derived from the availability of the controller $u^* : \mathcal{X} \to \mathcal{U}$ introduced in Section II. Specifically, let $E : \prod_{j=1}^{N} \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j) \to \mathcal{X}$ be a continuous map that is distributed over $\mathcal{G}$ 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 $y \in \mathcal{X}$. Essentially, what the map E does for each agent is select a point from the set-valued information that it possesses. Now, define

$$u^{**}(Y) = u^*(E(Y)). \quad (12)$$

Note that this controller satisfies (11a) and (11b) because u^* satisfies (4a) and (4b).

Example IV.3 (Controller definition with the ball-radius promise rules) Here we construct a controller u^{**} using (12) for the case when promises are generated according to the ball-radius control rule described in Remark IV.2. To do so, note that it is sufficient to define the map $E : \prod_{j=1}^{N} \mathbb{P}^{\mathrm{cc}}(\mathcal{X}_j) \to \mathcal{X}$ only for tuples of sets of the form given in (9), where the corresponding control promise is defined by (10). With the notation of Remark IV.2, recall that the promise that an agent j sends at time t is conveyed through three parameters (y_j, v_j, δ_j), the state $y_j = x_j(t)$ when the promise was sent, the control action $v_j = u_j(X_{\mathcal{N}}^j(t))$ at that time, and the radius δ_j of the ball. We can then define the jth component of the map E as

$$E_j(X_1[t](t'), \dots, X_N[t](t')) = e^{A_j(t'-t)} y_j + \int_t^{t'} e^{A_j(t'-\tau)} B_j v_j d\tau,$$

which is guaranteed to be in $X_j[t](t')$ for $t' \geq t$. This specification amounts to each agent i calculating the evolution of its neighbors $j \in \mathcal{N}(i)$ as if they were using a zero-order hold control. •

C. Self-triggered information updates

Here we discuss 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 $X^i_{\mathcal{N}}$ describing its neighbors' state and can compute its evolution under the controller u^{**} via

$$x_i(t) = e^{A_i(t-t^i_{\text{last}})}x_i(t^i_{\text{last}}) + \int_{t^i_{\text{last}}}^{t} e^{A_i(t-\tau)}B_i u^{**}_i(X^i_{\mathcal{N}}(\tau))d\tau, \quad t \geq t^i_{\text{last}}. \tag{13}$$

Note that this evolution of agent i can be viewed as a promise that it makes to itself, i.e., $X^i_i[\cdot](t) = \{x_i(t)\}$. With this in place, i can schedule the next time t^i_{next} at which it will need updated information from its neighbors by computing the worst-case time evolution of V along its trajectory among all the possible evolutions of its neighbors given the information contained in their promises. Formally, we define, for $Y_{\mathcal{N}} \in \prod_{j\in\mathcal{N}(i)\cup\{i\}} \mathbb{P}^{\text{cc}}(\mathcal{X}_j)$,

$$\mathcal{L}_i V^{\sup}(Y_{\mathcal{N}}) = \sup_{y_{\mathcal{N}}\in Y_{\mathcal{N}}} \nabla_i V(y_{\mathcal{N}})\left(A_i y_i + B_i u^{**}_i(Y_{\mathcal{N}})\right), \tag{14}$$

where y_i is the element of $y_{\mathcal{N}}$ corresponding to i. Then, the trigger for when agent i needs new information from its neighbors is similar to (8), where we now use the promise sets instead of the guaranteed sets. Specifically, the critical time at which information is requested is given by $t^i_{\text{next}} = \max\{t^i_{\text{last}} + T_{\text{d,self}}, t^*\}$, where $T_{\text{d,self}} > 0$ is an a priori chosen parameter that we discuss below and t^* is implicitly defined by

$$t^* = \min\{t \geq t^i_{\text{last}} \mid \mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t)) = 0\}. \tag{15}$$

This ensures that for $t \in [t^i_{\text{last}}, t^*)$, agent i is guaranteed to be contributing positively to the desired task. We refer to $t^i_{\text{next}} - t^i_{\text{last}}$ as the self-triggered request time. The parameter $T_{\text{d,self}} > 0$ is 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 $t^* \leq t^i_{\text{last}} + T_{\text{d,self}}$, agent i 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 V along the network execution. The team-triggered controller is defined by

$$u^{\text{team}}_i(t) = \begin{cases} u^{**}_i(X^i_{\mathcal{N}}(t)), & \text{if } t \leq t^*, \\ u^{\text{sf}}_i(x_i(t)), & \text{if } t > t^*, \end{cases} \tag{16}$$

for $t \in [t^i_{\text{last}}, t^i_{\text{next}})$, where t^* is given by (15). Note that the self-triggered dwell time $T_{\text{d,self}}$ only limits the frequency at which an agent i can *request* information from its neighbors and does not provide guarantees on inter-event times of when its memory is updated or its control is recomputed. If a neighboring agent sends information to agent i before this dwell time has expired (because that agent has broken a promise), this triggers agent i to update its memory and potentially recompute its control law.

D. 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, causing it to change its former plans. Another example is given by an agent that made a promise that is not able to keep for as long as it anticipated. Consider an agent $i \in \{1,\dots,N\}$ that has sent a promise $X^j_i[t_{\text{last}}]$ 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., $x_i(t^*) \notin X^j_i[t_{\text{last}}](t^*)$, then it is responsible for sending a new promise $X^j_i[t_{\text{next}}]$ 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 using the guaranteed sets (7) as follows,

$$X^j_i[\cdot](t) = \bigcup_{x_i \in X^j_i[\cdot](t^*)} \mathbf{X}^j_i(t, x_i) = \bigcup_{x_i \in X^j_i[\cdot](t^*)} \mathcal{R}_i(t - t^*, x_i) \tag{17}$$

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 $X^j_i[\cdot](t)$ is guaranteed to contain $x_i(t)$ for $t \geq t^*$.

Remark IV.4 (Promise expiration times) It is also possible to set an expiration time $T_{\text{exp}} > T_{\text{d,event}}$ for the validity of promises. If this in effect and a promise is made at t_{last}, it is only valid for $t \in [t_{\text{last}}, t_{\text{last}} + T_{\text{exp}}]$. The expiration of the promise triggers the formulation of a new one. •

The combination of the self- and event-triggered information updates described above together with the team-triggered controller u^{team} as defined in (16) gives rise to the TEAM-TRIGGERED LAW, which is formally presented in Algorithm 1. The self-triggered information request in Algorithm 1 is executed by an agent anytime new information is received, whether it was actively requested by the agent, or was received from some neighbor due to the breaking of a promise.

V. Convergence of the team-triggered law

Here we analyze the convergence properties of the TEAM-TRIGGERED LAW. Our first result establishes the monotonic evolution of the Lyapunov function V along the network trajectories.

Proposition V.1 *Consider a networked cyber-physical system as described in Section II executing the* TEAM-TRIGGERED LAW *(cf. Algorithm 1) based on a continuous controller $u^{**} : \prod_{j\in\{1,\dots,N\}} \mathbb{P}^{cc}(\mathcal{X}_j) \to \mathbb{R}^m$ that satisfies* (11) *and is distributed over the communication graph $\mathcal{G}$. Then, the*

Algorithm 1: TEAM-TRIGGERED LAW

```
(Self-trigger information update)
At any time $t$ agent $i \in \{1,\dots,N\}$ receives new promise(s) $X_j^i[t]$ from neighbor(s) $j \in \mathcal{N}(i)$, agent $i$ performs:
1: compute own control $u_i^{\text{team}}(t')$ for $t' \geq t$ using (16)
2: compute own state evolution $x_i(t')$ for $t' \geq t$ using (13)
3: compute first time $t^* \geq t$ such that $\mathcal{L}_i V^{\sup}(X_{\mathcal{N}}^i(t^*)) = 0$
4: schedule information request to neighbors in $\max\{t^* - t, T_{\text{d,self}}\}$ seconds
(Respond to information request)
At any time $t$ a neighbor $j \in \mathcal{N}(i)$ requests information, agent $i$ performs:
1: send new promise $X_i^j[t] = R_i^{\text{s}}(X_{\mathcal{N}}^i[\cdot]_{[t,\infty)})$ to agent $j$
(Event-trigger information update)
At all times $t$, agent $i$ performs:
1: if there exists $j \in \mathcal{N}(i)$ such that $x_i(t) \notin X_i^j[\cdot](t)$ then
2:     if agent $i$ has sent a promise to $j$ at some time $t_{\text{last}} \in (t - T_{\text{d,event}}, t]$ then
3:         send warning message WARN to agent $j$ at time $t$
4:         schedule to send new promise $X_i^j[t_{\text{last}} + T_{\text{d,event}}] = R_i^{\text{s}}(X_{\mathcal{N}}^i[\cdot]_{|[t_{\text{last}}+T_{\text{d,event}},\infty)})$ to agent $j$ in $t_{\text{last}} + T_{\text{d,event}} - t$ seconds
5:     else
6:         send new promise $X_i^j[t] = R_i^{\text{s}}(X_{\mathcal{N}}^i[\cdot]_{|[t,\infty)})$ to agent $j$ at time $t$
7:     end if
8: end if
(Respond to warning message)
At any time $t$ agent $i \in \{1,\dots,N\}$ receives a warning message WARN from agent $j \in \mathcal{N}(i)$
1: redefine promise set $X_j^i[\cdot](t') = \cup_{x_j \in X_j^i[\cdot](t)} \mathcal{R}_j(t' - t, x_j)$ for $t' \geq t$
```

function V is monotonically nonincreasing along any network trajectory.

Proof: We start by noting that the time evolution of V under Algorithm 1 is continuous and piecewise continuously differentiable. Moreover, at the time instants when the time derivative is well-defined, one has

$$\frac{d}{dt}V(x(t)) = \sum_{i=1}^{N} \nabla_i V(x_{\mathcal{N}}^i(t)) \left(A_i x_i(t) + B_i u_i^{\text{team}}(t)\right) \quad (18)$$
$$\leq \sum_{i=1}^{N} \sup_{y_{\mathcal{N}} \in X_{\mathcal{N}}^i(t)} \nabla_i V(y_{\mathcal{N}}) \left(A_i x_i(t) + B_i u_i^{\text{team}}(t)\right) \leq 0.$$

As we justify next, the last inequality follows by design of the TEAM-TRIGGERED LAW. For each $i \in \{1,\dots,N\}$, if $\mathcal{L}_i V^{\sup}(X_{\mathcal{N}}^i(t)) \leq 0$, then $u_i^{\text{team}}(t) = u_i^{**}(X_{\mathcal{N}}^i(t))$ (cf. (16)). In this case the corresponding summand of (18) is exactly $\mathcal{L}_i V^{\sup}(X_{\mathcal{N}}^i(t))$, as defined in (14). If $\mathcal{L}_i V^{\sup}(X_{\mathcal{N}}^i(t)) > 0$, then $u_i^{\text{team}}(t) = u_i^{\text{sf}}(x_i(t))$, for which the corresponding summand of (18) is exactly 0. ■

The next result characterizes the convergence properties of team-triggered coordination strategies.

Proposition V.2 *Consider a networked cyber-physical system as described in Section II executing the* TEAM-TRIGGERED LAW *(cf. Algorithm 1) with dwell times $T_{d,self}, T_{d,event} > 0$ based on a continuous controller $u^{**} : \prod_{j\in\{1,\dots,N\}} \mathbb{P}^{cc}(\mathcal{X}_j) \to \mathbb{R}^m$ that satisfies* (11) *and is distributed over the communication graph $\mathcal{G}$. Then, any bounded network trajectory with uniformly bounded promises asymptotically approaches the desired set D.*

The requirements of uniformly bounded promises in Proposition V.2 means that there exists a compact set that contains all promise sets. Note that this is automatically guaranteed if the network state space is compact. Alternatively, if the sets of allowable controls are bounded, a bounded network trajectory with expiration times for promises implemented as outlined in Remark IV.4 would result in uniformly bounded promises. There are two main challenges in proving Proposition V.2, which we discuss next.

The first challenge is that agents operate asynchronously, i.e., agents receive and send information, and update their control laws possibly at different times. To model asynchronism, we use a procedure called analytic synchronization, see e.g. [33]. Let the time schedule of agent i be given by $\mathcal{T}^i = \{t_0^i, t_1^i, \dots\}$, where t_ℓ^i corresponds to the ℓth time that agent i receives information from one or more of its neighbors (the time schedule $\mathcal{T}^i$ is not known a priori by the agent). Note that this information can be received because i requests it itself, or a neighbor sends it to i because an event is triggered. Analytic synchronization simply consists of merging together the individual time schedules into a global time schedule $\mathcal{T} = \{t_0, t_1, \dots\}$ by setting

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

Note that more than one agent may receive information at any given time $t \in \mathcal{T}$. This synchronization is done for analysis purposes only. For convenience, we identify $\mathbb{Z}_{\geq 0}$ with $\mathcal{T}$ via $\ell \mapsto t_\ell$.

The second challenge is that a strategy resulting from the team-triggered approach has a discontinuous dependence on the network state and the agent promises. More precisely, 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, as described in Section II. 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 network at some time t is collected in

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

where $X^i[\cdot]_{|[t,\infty)} = \big(X_1^i[\cdot]_{|[t,\infty)}, \dots, X_N^i[\cdot]_{|[t,\infty)}\big) \in S_i$. Here, $[\cdot]$ is shorthand notation to denote the fact that promises might have been made at different times, earlier than t. The TEAM-TRIGGERED LAW corresponds to a 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 of defining a discrete-time set-valued map $M : S \times \mathbb{Z}_{\geq 0} \rightrightarrows S \times \mathbb{Z}_{\geq 0}$ whose trajectories contain the trajectories of the 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 models the case when agent i does not receive any information from its neighbors. In this case, the ith component of the elements in $M(Z, \ell)$ is simply the ith component of Z,

$$\left(Z^i_{1\,|[t_{\ell+1},\infty)}, \dots, Z^i_{N\,|[t_{\ell+1},\infty)} \right), \tag{19}$$

The second possibility models the case when agent i has received information (including a WARN message) from at least one neighbor: the ith component of the elements in $M(Z, \ell)$ is

$$\left(Y^i_{1\,|[t_{\ell+1},\infty)}, \dots, Y^i_{N\,|[t_{\ell+1},\infty)} \right), \tag{20}$$

where each agent has access to its own state at all times,

$$\begin{aligned} Y^i_i(t) &= e^{A_i(t - t_{\ell+1})} Z^i_i(t_{\ell+1}) \\ &\quad + \int_{t_{\ell+1}}^{t} e^{A_i(t-\tau)} B_i u^{\text{team}}_i(\tau) d\tau, \quad t \geq t_{\ell+1}, \end{aligned} \tag{21a}$$

(here, with a slight abuse of notation, we use u^{team} to denote the controller evaluated at Y^i) and,

$$Y^i_{j\,|[t_{\ell+1},\infty)} = \tag{21b}$$
$$\begin{cases} Z^i_{j\,|[t_{\ell+1},\infty)}, & \text{if } i \text{ does not receive info from } j, \\ W^i_{j\,|[t_{\ell+1},\infty)}, & \text{if } i \text{ receives a warning from } j, \\ R^{\text{s}}_j(Z^j_{\mathcal{N}\,|[t_{\ell+1},\infty)}), & \text{otherwise}, \end{cases}$$

for $j \neq i$, where $W^i_j(t) = \bigcup_{z_i \in Z^i_j(t_{\ell+1})} \mathbf{X}^i_j(t, z_i)$ corresponds to the redefined promise (17) for $t \geq t_{\ell+1}$ as a result of the warning message.

We emphasize two properties of the set-valued map M. First, any trajectory of the 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, although the map defined by the TEAM-TRIGGERED LAW is discontinuous, the set-valued map M is closed, as we show next (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 $y \in T(x)$).

Lemma V.3 (Set-valued map is closed) *The set-valued map $M : S \times \mathbb{Z}_{\geq 0} \rightrightarrows S \times \mathbb{Z}_{\geq 0}$ is closed.*

Proof: To show this we appeal to the fact that a set-valued map composed of a finite collection of continuous maps is closed [34, E1.9]. Given (Z, ℓ), the set $M(Z, \ell)$ is finitely comprised of all possible combinations of whether or not updates occur for every agent pair $i, j \in \{1, \dots, N\}$. In the case that an agent i does not receive any information from its neighbors, it is trivial to show that (19) is continuous in (Z, ℓ) because $Z^i_{j\,[t_{\ell+1},\infty)}$ is simply the restriction of $Z^i_{j\,[t_\ell,\infty)}$ to the interval $[t_{\ell+1}, \infty)$, for each $i \in \{1, \dots, N\}$ and $j \in \mathcal{N}(i)$. In the case that an agent i does receive updated information, the above argument still holds for agents j that did not send information to agent i. If an agent j sends a warning message to agent i, $W^i_{j\,|[t_{\ell+1},\infty)}$ is continuous in (Z, ℓ) by continuity of the reachable sets on their starting point. If an agent j sends a new promise to agent i, $Y^i_{j\,|[t_{\ell+1},\infty)}$ is continuous in (Z, ℓ) by definition of the function R^{s}_j. Finally, one can see that $Y^i_{i\,|[t_{\ell+1},\infty)}$ is continuous in (Z, ℓ) from (21a). ■

We are now ready to prove Proposition V.2.

Proof of Proposition V.2: Here we resort to the LaSalle Invariance Principle for set-valued discrete-time dynamical systems [34, Theorem 1.21]. Let $W = S \times \mathbb{Z}_{\geq 0}$, which is closed and strongly positively invariant with respect to M. A similar argument to that in the proof of Proposition V.1 shows that the function V is nonincreasing along M. Combining this with the fact that the set-valued map M is closed (cf. Lemma V.3), the application of the LaSalle Invariance Principle implies that the trajectories of M that are bounded in the first N components approach the largest weakly positively invariant set contained in

$$\begin{aligned} S^* &= \{(Z, \ell) \in S \times \mathbb{Z}_{\geq 0} \mid \exists (Z', \ell + 1) \in M(Z, \ell) \\ &\qquad \text{such that } V(Z') = V(Z)\}, \\ &= \{(Z, \ell) \in S \times \mathbb{Z}_{\geq 0} \mid \mathcal{L}_i V^{\text{sup}}(Z^i_{\mathcal{N}}) \geq 0 \\ &\qquad \text{for all } i \in \{1, \dots, N\}\}. \end{aligned} \tag{22}$$

We now restrict our attention to those trajectories of M that correspond to the TEAM-TRIGGERED LAW. For convenience, let $\text{loc}(Z, \ell) : S \times \mathbb{Z}_{\geq 0} \to \mathcal{X}$ be the map that extracts the true position information in (Z, ℓ), i.e.,

$$\text{loc}(Z, \ell) = \left(Z^1_1(t_\ell), \dots, Z^N_N(t_\ell) \right).$$

Given a trajectory γ of the TEAM-TRIGGERED LAW that satisfies all the assumptions of the statement of Proposition V.2, the bounded evolutions and uniformly bounded promises ensure that the trajectory γ is bounded. Then, the omega limit set $\Omega(\gamma)$ is weakly positively invariant and hence is contained in S^*. Our objective is to show that, for any $(Z, \ell) \in \Omega(\gamma)$, we have $\text{loc}(Z, \ell) \in D$. We show this reasoning by contradiction. Let $(Z, \ell) \in \Omega(\gamma)$ but suppose $\text{loc}(Z, \ell) \notin D$. This means that $\mathcal{L}_i V^{\text{sup}}(Z^i_{\mathcal{N}}) \geq 0$ for all $i \in \{1, \dots, N\}$. Take any agent i, by the SELF-TRIGGERED INFORMATION UPDATES, agent i will request new information from neighbors in at most $T_{\text{d,self}}$ seconds. This means there exists a state $(Z', \ell + \ell') \in \Omega(\gamma)$ for which agent i has just received updated information from its neighbors $j \in \mathcal{N}(i)$. Since $(Z', \ell + \ell') \in S^*$, we know $\mathcal{L}_i V^{\text{sup}}({Z^i_{\mathcal{N}}}') \geq 0$. We also know, since information was just updated, that ${Z^i_j}' = \text{loc}_j(Z', \ell + \ell')$ is exact for all $j \in \mathcal{N}(i)$. But, by (11a), $\mathcal{L}_i V^{\text{sup}}({Z^i_{\mathcal{N}}}') \leq 0$ because $\text{loc}(Z', \ell + \ell') \notin D$. This means that each time any agent i updates its information, we must have $\mathcal{L}_i V^{\sup}({Z^i_{\mathcal{N}}}') = 0$. However, by (11b), there must exist at least one agent i such that $\mathcal{L}_i V^{\sup}({Z^i_{\mathcal{N}}}') < 0$ since $\operatorname{loc}(Z', \ell + \ell') \notin D$, which yields a contradiction. Thus for the trajectories of the TEAM-TRIGGERED LAW, $(Z, \ell) \in S^*$ implies that $\operatorname{loc}(Z, \ell) \in D$. ■

Given the convergence result of Proposition V.2, a termination condition for the TEAM-TRIGGERED LAW could be included via the implementation of a distributed algorithm that employs tokens identifying what agents are using safe-model controllers, see e.g., [35], [36]. Also, according to the proof of Proposition V.2, the actual value of the event-triggered dwell time $T_{\text{d,event}}$ does not affect the convergence property of the trajectories of the constructed discrete-time set-valued system. However, the dwell time does affect the rate of convergence of the actual continuous-time system (as a larger dwell time corresponds to more time actually elapsing between each step of the constructed discrete-time system).

Remark V.4 (Availability of a safe-mode controller) The assumption on the availability of the safe-mode controller plays an important role in the proof of Proposition V.2 because it provides individual agents with a way of avoiding having a negative impact on the monotonic evolution of the Lyapunov function. We believe this assumption can be relaxed for dynamics that allow agents to execute maneuvers that bring them back to their current state. Under such maneuvers, the Lyapunov function will not evolve monotonically but, at any given time, will always guarantee to be less than or equal to its current value at some future time. We have not pursued this approach here for simplicity and instead defer it for future work. •

The next result states that, under the TEAM-TRIGGERED LAW with positive dwell times, the system does not exhibit Zeno behavior.

Lemma V.5 (No Zeno behavior) *Under the assumptions of Proposition V.2, the network executions do not exhibit Zeno behavior.*

Proof: Due to the self-triggered dwell time $T_{\text{d,self}}$, the self-triggered information request steps in Algorithm 1 guarantee that the minimum time before an agent i asks its neighbors for new information is $T_{\text{d,self}} > 0$. Similarly, due to the event-triggered dwell time $T_{\text{d,event}}$, agent i will never receive more than two messages (one accounts for promise information, the other for the possibility of a WARN message) from a neighbor j in a period of $T_{\text{d,event}} > 0$ seconds. This means that any given agent can never receive an infinite amount of information in finite time. When new information is received, the control law (16) can only switch a maximum of two times until new information is received again. Specifically, if an agent i is using the normal control law when new information is received, it may switch to the safe-mode controller at most one time until new information is received again. If instead an agent i is using the safe-mode control controller when new information is received, it may immediately switch to the normal control law, and then switch back to the safe-mode controller some time in the future before new information is received again. The result follows from the fact that $|\mathcal{N}(i)|$ is finite for each $i \in \{1, \dots, N\}$. ■

Remark V.6 (Adaptive self-triggered dwell time) Dwell times play an important role in preventing Zeno behavior. However, a constant self-triggered dwell time throughout the network evolution might result in wasted communication effort because some agents might reach a state where their effect on the evolution of the Lyapunov function is negligible compared to others. In such case, the former agents could implement larger dwell times, thus decreasing communication effort, without affecting the overall performance. Next, we give an example of such an adaptive dwell time scheme. Let t be a time at which agent $i \in \{1, \dots, N\}$ has just received new information from its neighbors $\mathcal{N}(i)$. Then, the agent sets its dwell time to

$$T^i_{\text{d,self}}(t) = \max \Big\{ \delta_d \sum_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} \frac{\|u^{**}_j(X^j_{\mathcal{N}}(t)) - u^{\text{sf}}_j(x_j(t))\|_2}{\|u^{**}_i(X^i_{\mathcal{N}}(t)) - u^{\text{sf}}_i(x_i(t))\|_2}, \Delta_d \Big\}, \tag{23}$$

for some a priori chosen δ_d, $\Delta_d > 0$. The intuition behind this design is the following. The value $\|u^{**}_j(X^j_{\mathcal{N}}(t)) - u^{\text{sf}}_j(x_j(t))\|_2$ can be interpreted as a measure of how far agent j is from reaching a point where it cannot no longer contribute positively to the global task. As agents are nearing this point, they are more inclined to use the safe mode control to stay put and hence do not require fresh information. Therefore, if agent i is close to this point but its neighbors are not, (23) sets a larger self-triggered dwell time to avoid excessive requests for information. Conversely, if agent i is far from this point but its neighbors are not, (23) sets a small dwell time to let the self-triggered request mechanism be the driving factor in determining when new information is needed. For agent i to implement this, in addition to current state information and promises, each neighbor $j \in \mathcal{N}(i)$ also needs to send the value of $\|u^{**}_j(X^j_{\mathcal{N}}(t)) - u^{\text{sf}}_j(x_j(t))\|_2$ at time t. In the case that information is not received from all neighbors, agent i simply uses the last computed dwell time. Section VII illustrates this adaptive scheme in simulation. •

VI. Robustness against unreliable communication

This section studies the robustness of the team-triggered approach in scenarios with packet drops, delays, and communication noise. We start by introducing the possibility of packet drops in the network. For any given message an agent sends to another agent, assume there is an unknown probability $0 \leq p < 1$ that the packet is dropped, and the message is never received. We also consider an unknown (possibly time-varying) communication delay $\Delta(t) \leq \bar{\Delta}$ in the network for all t where $\bar{\Delta} \geq 0$ 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 [28], [37] and references therein. Lastly, we also account for the possibility of communication noise or quantization. We assume that messages among agents are corrupted with an error which is upper bounded by some $\bar{\omega} \geq 0$ known to the agents.

With this model, the TEAM-TRIGGERED LAW as described in Algorithm 1 does not guarantee convergence because the monotonic behavior of the Lyapunov function no longer holds. The problem occurs when an agent j breaks a promise to agent i at some time t. If this occurs, agent i will operate with invalid information (due to the sources of error described above) and compute $\mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t'))$ (as defined in (14)) incorrectly for $t' \geq t$.

Next, we discuss how the TEAM-TRIGGERED LAW can be modified in scenarios with unreliable communication. To deal with 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 promise set X^i_j that contains the actual promise that agent j intended to send. We refer to this action as making a promise set valid. The following example shows how it can be done for the promises described in Remark IV.2.

Example VI.1 (Ball-radius promise rule with communication noise) In the scenario with bounded communication noise, agent j sends the control promise conveyed through $x_j(t), u_j(X^j_{\mathcal{N}}(t))$, and δ_j, to agent i at time t as defined in Remark IV.2, but i receives instead $\widehat{x}_j(t), \widehat{u}_j(X^j_{\mathcal{N}}(t))$, and $\widehat{\delta}_j$, where it knows that $\|x_j(t) - \widehat{x}_j(t)\|_2 \leq \bar{\omega}$, $\|u_j(X^j_{\mathcal{N}}(t)) - \widehat{u}_j(X^j_{\mathcal{N}}(t))\|_2 \leq \bar{\omega}$, and $|\delta_j - \widehat{\delta}_j| \leq \bar{\delta}$, given that $\bar{\omega}$ and $\bar{\delta}$ are known a priori. To ensure that the promise agent i operates with about agent j contains the true promise made by j, agent i can set

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

To create the state promise from this, i would need the true state $x_j(t)$ of j at time t. However, since only the estimate $\widehat{x}^i_j(t)$ is available, we modify (9) by

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

We deal with the packet drops and communication delays with warning messages similar to the ones introduced in Section IV-D. Let an agent j break its promise to agent i at time t, then agent j sends i a new promise set $X^i_j[t]$ for $t' \geq t$ and warning message WARN. Since agent i only receives WARN at time t, the promise set $X^i_j[t]$ may not be available to agent i for $t' \geq t$. If the packet is dropped, then the message never comes through, if the packet is successfully transmitted, then $X^i_j[t](t')$ is only available for $t' \geq t + \Delta(t)$. In either case, we need a promise set $X^i_j[\cdot](t')$ for $t' \geq t$ that is guaranteed to contain $x_j(t')$. We do this by redefining the promise using the reachable set, similarly to (17). Note that this does not require the agents to have a synchronized global clock, as the times t' and t are both monitored by the receiving agent i. In other words, it is not necessary for the message sent by agent j to be timestamped. By definition of reachable set, the promise $X^i_j[\cdot](t')$ is guaranteed to contain $x_j(t')$ for $t' \geq t$. If at time $t + \bar{\Delta}$, agent i has still not received the promise $X^i_j[t]$ from j, it can send agent j a request REQ for a new message at which point j would send i a new promise $X^i_j[t + \bar{\Delta}]$. Note that WARN is not sent in this case because the message was requested from j by i and not a cause of j breaking a promise to i. The ROBUST TEAM-TRIGGERED LAW, formally presented in Algorithm 2, ensures the monotonic evolution of the Lyapunov function V even in the presence of packet drops, communication delays, and communication noise.

Algorithm 2: ROBUST TEAM-TRIGGERED LAW

```
(Self-trigger information update)
At any time $t$ agent $i \in \{1, \dots, N\}$ receives new promise(s) $\widehat{X}^i_j[t]$ from
neighbor(s) $j \in \mathcal{N}(i)$, agent $i$ performs:
 1: create valid promise $X^i_j[t]$ with respect to $\bar{\omega}$
 2: compute own control $u^{\text{team}}_i(t')$ for $t' \geq t$ using (16)
 3: compute own state evolution $x_i(t')$ for $t' \geq t$ using (13)
 4: compute first time $t^* \geq t$ such that $\mathcal{L}_i V^{\sup}(X^i_{\mathcal{N}}(t^*)) = 0$
 5: schedule information request to neighbors in $\max\{t^* - t, T_{\text{d,self}}\}$ seconds
 6: while message from $j$ has not been received do
 7:     if current time equals $t + \max\{t^* - t, T_{\text{d,self}}\} + k\bar{\Delta}$ for $k \in \mathbb{Z}_{\geq 0}$ then
 8:         send agent $j$ a request REQ for new information
 9:     end if
10: end while
(Respond to information request)
At any time $t$ a neighbor $j \in \mathcal{N}(i)$ requests information, agent $i$ performs:
 1: send new promise $Y^j_i[t] = R^s_i(X^i_{\mathcal{N}}[\cdot]|_{[t,\infty)})$ to agent $j$
(Event-trigger information update)
At all times $t$, agent $i$ performs:
 1: if there exists $j \in \mathcal{N}(i)$ such that $x_i(t) \notin Y^j_i[\cdot](t)$ then
 2:     send warning message WARN to agent $j$
 3:     if agent $i$ has sent a promise to $j$ at some time $t_{\text{last}} \in (t - T_{\text{d,event}}, t]$ then
 4:         schedule to send new promise $Y^j_i[t_{\text{last}} + T_{\text{d,event}}] = R^s_i(X^i_{\mathcal{N}}[\cdot]|_{[t_{\text{last}}+T_{\text{d,event}},\infty)})$ to agent $j$ in $t_{\text{last}} + T_{\text{d,event}} - t$ seconds
 5:     else
 6:         send new promise $Y^j_i[t] = R^s_i(X^i_{\mathcal{N}}[\cdot]|_{[t,\infty)})$ to agent $j$
 7:     end if
 8: end if
(Respond to warning message)
At any time $t$ agent $i \in \{1, \dots, N\}$ receives a warning message WARN from
agent $j \in \mathcal{N}(i)$
 1: redefine promise set $X^i_j[\cdot](t') = \cup_{x^0_j \in X^i_j[\cdot](t)} \mathcal{R}_j(t' - t, x^0_j)$ for $t' \geq t$
 2: while message from $j$ has not been received do
 3:     if current time equals $t + k\bar{\Delta}$ for $k \in \mathbb{Z}_{\geq 0}$ then
 4:         send agent $j$ a request REQ for new information
 5:     end if
 6: end while
```

The next result establishes the asymptotic correctness guarantees on the ROBUST TEAM-TRIGGERED LAW. In the presence of communication noise or delays, convergence can be guaranteed only to a set that contains the desired set D.

Corollary VI.2 *Consider a networked cyber-physical system as described in Section II 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 2) with dwell times $T_{d,self}, T_{d,event} > 0$ based on a continuous controller $u^{**} : \prod_{j\in\{1,\dots,N\}} \mathbb{P}^{cc}(\mathcal{X}_j) \to \mathbb{R}^m$ that satisfies* (11) *and is distributed over the communication graph $\mathcal{G}$. Let*

$$D'(\bar\Delta,\bar\omega) = \{x \in \mathcal{X} \mid \inf_{x^i_{\mathcal{N}}{}' \in \overline{B}(x^i_{\mathcal{N}},\bar\omega)} \mathcal{L}_i V^{sup}\Big(\{x_i\} \times \prod_{j\in\mathcal{N}(i)} \cup_{y_j\in\overline{B}(x^i_j{}',\bar\omega)} \mathcal{R}_j(\bar\Delta, y_j)\Big) \geq 0 \quad \text{for all } i \in \{1,\dots,N\}\}, \tag{24}$$

Then, any bounded network trajectory with uniformly bounded promises asymptotically converges to $D'(\bar\Delta,\bar\omega) \supset D$ with probability 1.

Proof: We begin by noting that by equation (11b), the definition (14), and the continuity of u^{**}, D can be written as

$$D'(0,0) = \{x \in \mathcal{X} \mid \sum_{i=1}^N \nabla_i V(x)(A_i x_i + B_i u_i^{**}(\{x^i_{\mathcal{N}}\})) \geq 0\}.$$

One can see that $D \subset D'(\bar\Delta,\bar\omega)$ by noticing that, for any $x \in D$, $\bar\omega, \bar\Delta \geq 0$, no matter which point $x^i_{\mathcal{N}}{}' \in \overline{B}(x^i_{\mathcal{N}},\bar\omega)$ is taken, one has $x^i_{\mathcal{N}} \in \{x_i\} \times \prod_{j\in\mathcal{N}(i)} \cup_{y_j\in\overline{B}(x^i_j{}',\bar\omega)} \mathcal{R}_j(\bar\Delta, y_j)$. To show that the bounded trajectories of the ROBUST TEAM-TRIGGERED LAW converge to D', we begin by noting that all properties of M used in the proof of Proposition V.2 still hold in the presence of packet drops, delays, and communication noise as long as the time schedule $\mathcal{T}^i$ is unbounded for each agent $i \in \{1,\dots,N\}$. In order for the time schedule $\mathcal{T}^i$ to be unbounded, 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 over an infinite time horizon is 0. Thus, with probability 1, there are an infinite number of information updates for each agent. Using a similar argument to that of Lemma V.5, one can show that the positive dwell times $T_{\text{d,self}}, T_{\text{d,event}} > 0$ ensure that Zeno behavior does not occur, meaning that $t^i_\ell \to \infty$. Then, by the analysis in the proof of Proposition V.2, the bounded trajectories of M still converge to S^* as defined in (22).

For a bounded evolution γ of the ROBUST TEAM-TRIGGERED LAW, we have that $\Omega(\gamma) \subset S^*$ is weakly positively invariant. Note that, since agents may never have exact information about their neighbors, we can no longer leverage properties (11a) and (11b) to precisely characterize $\Omega(\gamma)$. We now show that for any $(Z,\ell) \in \Omega(\gamma)$, we have $\operatorname{loc}(Z,\ell) \in D'$. Let $(Z,\ell) \in \Omega(\gamma)$. This means that $\mathcal{L}_i V^{\text{sup}}(Z^i_{\mathcal{N}}) \geq 0$ for all $i \in \{1,\dots,N\}$. Take any agent i, by the ROBUST TEAM-TRIGGERED LAW, agent i will request new information from neighbors in at most $T_{\text{d,self}}$ seconds. This means there exists a state $(Z',\ell+\ell') \in \Omega(\gamma)$ for which agent i has just received updated, possibly delayed, information from its neighbors $j \in \mathcal{N}(i)$. Since $(Z',\ell+\ell') \in S^*$, we know $\mathcal{L}_i V^{\text{sup}}(Z^i_{\mathcal{N}}{}') \geq 0$. We also know, since information was just updated, that $Z^i_{\mathcal{N}}{}' \subset \{Z^{i\prime}_i\} \times \prod_{j\in\mathcal{N}(i)} \cup_{y_j\in\overline{B}(z^i_j{}',\bar\omega)} \mathcal{R}(\bar\Delta, y_j)$. Since $(Z',\ell+\ell') \in S^*$, we know that $\mathcal{L}_i V^{\text{sup}}(Z^i_{\mathcal{N}}{}') \geq 0$, for all $i \in \{1,\dots,N\}$. This means that $\operatorname{loc}(Z',\ell+\ell') \subset D'$, thus $\operatorname{loc}(Z,\ell) \in S^* \subset D'$. ∎

From the proof of Corollary VI.2, one can see that the modifications made to the ROBUST TEAM-TRIGGERED LAW make the omega limit sets of its trajectories larger than those of the TEAM-TRIGGERED LAW, resulting in $D \subset D'$. The set D' depends on the Lyapunov function V. However, the difference between $D'(\bar\Delta,\bar\omega)$ and D vanishes as $\bar\omega$ and $\bar\Delta$ vanish.

VII. SIMULATIONS

In this section we present simulations of coordination strategies derived from the team- and self-triggered approaches in a planar multi-agent formation control problem. Our starting point is the distributed coordination algorithm based on graph rigidity analyzed in [38], [39] which makes the desired network formation locally (but not globally) asymptotically stable. In this regard, the state space $\mathcal{X}$ of Section II corresponds to the domain of attraction of the desired equilibria and, as long as the network trajectories do not leave this set, the convergence results still hold. The local convergence result of the team-triggered approach here is only an artifact of the specific example and, in fact, if the assumptions (4) are satisfied globally, then the system is globally asymptotically stabilized. The interested reader is referred to [2] for a similar study in a optimal networked deployment problem where the assumptions hold globally.

Consider 4 agents communicating over a graph which is only missing the edge $(1,3)$ from the complete graph. The agents seek to attain a rectangle formation of side lengths 1 and 2. Each agent has unicycle dynamics,

$$\dot x_i = u_i \begin{bmatrix}\cos\theta_i \\ \sin\theta_i\end{bmatrix}$$
$$\dot\theta_i = v_i,$$

where $0 \leq u_i \leq u_{\max} = 5$ and $|v_i| \leq v_{\max} = 3$ are the control inputs. The safe-mode controller is then simply $u_i^{\text{sf}} \equiv 0$. To compute the distributed control law, each agent computes a goal point

$$p_i^*(x) = x_i + \sum_{j\in\mathcal{N}(i)} (\|x_j - x_i\|_2 - d_{ij}) \operatorname{unit}(x_j - x_i),$$

where d_{ij} is the pre-specified desired distance between agents i and j, and $\operatorname{unit}(x_j - x_i)$ denotes the unit vector in the direction of $x_j - x_i$. Then, the control law is then given by

$$u_i^* = \max\left\{\min\{k[\cos\theta_i \;\; \sin\theta_i]^T \cdot (p_i^*(x) - x_i), u_{\max}\}, 0\right\},$$
$$v_i^* = \max\left\{\min\{k(\angle(p_i^*(x) - x_i) - \theta_i), v_{\max}\}, -v_{\max}\right\},$$

where $k > 0$ is a design parameter. For our simulations we set $k = 150$. This continuous control law essentially ensures that the position x_i moves towards $p_i^*(x)$ when possible while the unicycle rotates its orientation towards this goal. This control law ensures that $V : (\mathbb{R}^2)^N \to \mathbb{R}_{\geq 0}$ given by

$$V(x) = \frac{1}{2} \sum_{(i,j)\in E} \left(\|x_j - x_i\|_2^2 - d_{ij}^2\right)^2,$$

is a nonincreasing function for the closed-loop system to establish the asymptotic convergence to the desired formation. For the team-triggered approach, we use both static and dynamic ball-radius promise rules. The controller u^{team} is then defined by (16), where controller u^{**} is given by (12) as described in Example IV.3. Note that although the agent has no forward velocity when using the safe controller, it will still rotate in place. The initial conditions are $x_1(0) = (6, 10)^T$, $x_2(0) = (7, 3)^T$, $x_3(0) = (14, 8)^T$, and $x_4(0) = (7, 13)^T$ and $\theta_i(0) = \pi/2$ for all i. We begin by simulating the team-triggered approach using fixed dwell times of $T_{\text{d,self}} = 0.3$ and $T_{\text{d,event}} = 0.003$ and the static ball-radius promise of Remark IV.2 with the same radius $\delta = 1$ for all agents. Figure 1 shows the trajectories of the TEAM-TRIGGERED LAW.

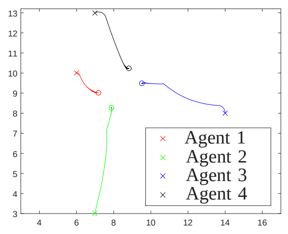


Fig. 1. Trajectories of an execution of the TEAM-TRIGGERED LAW with fixed dwell times and promises. The initial and final condition of each agent is denoted by an 'x' and an 'o', respectively.

To compare the team- and self-triggered approaches, we let N_S^i be the number of times i has requested new information and thus has received a message from each of its neighbors and N_E^i be the number of messages i has sent to a neighboring agent because it decided to break its promise. The total number of messages for an execution is $N_{\text{comm}} = \sum_{i=1}^4 |\mathcal{N}(i)|N_S^i + N_E^i$. Figure 2 compares the number of required communications in both approaches. Remarkably, for this specific example, the team-triggered approach outperforms the self-triggered approach in terms of required communication without sacrificing any performance in terms of time to convergence (the latter is depicted through the evolution of the Lyapunov function in Figure 4(b) below). Less overall communication has an important impact on reducing network load. In Figure 2(a), we see that very quickly all agents are requesting information as often as they can (as restricted by the self-triggered dwell time), due to the conservative nature of the self-triggered time computations. In the execution of the TEAM-TRIGGERED LAW in Figure 2(b), we see that the agents are requesting information from one another less frequently. Figure 2(c) shows that agents were required to break a few promises early on in the execution.

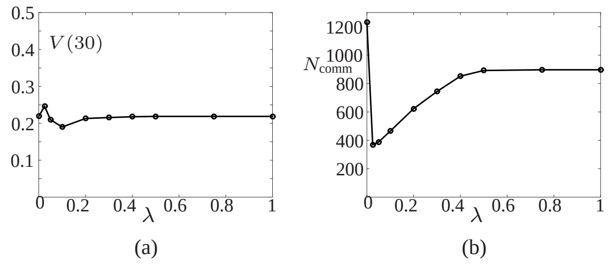


Fig. 3. Plots of (a) the value of the Lyapunov function at a fixed time (30 sec) and (b) the total number of messages exchanged in the network by this time for the team-triggered approach with varying tightness of promises λ.

Next, we illustrate the role that the tightness of promises has on the network performance. With the notation of Remark IV.2 for the static ball-radius rule, let $\lambda = \frac{\delta}{2u_{\max}}$. Note that when $\lambda = 0$, the promise generated by (10) is a singleton, i.e., an exact promise. On the other hand, when $\lambda = 1$, the promise generated by (10) contains the reachable set, corresponding to no actual commitment being made (i.e., the self-triggered approach). Figure 3 compares the value of the Lyapunov function after a fixed amount of time (30 seconds) and the total number of messages sent N_{comm} between agents by this time for varying tightness of promises. The dwell times here are fixed at $T_{\text{d,self}} = 0.3$ and $T_{\text{d,event}} = 0.003$. Note that a suitable choice of λ helps greatly reduce the amount of communication compared to the self-triggered approach ($\lambda = 1$) while maintaining a similar convergence rate.

Finally, we demonstrate the added benefits of using adaptive promises and dwell times. Figure 4(a) compares the total number of messages sent in the self-triggered approach and the team-triggered approaches with fixed promises and dwell times (FPFD), fixed promises and adaptive dwell times (FPAD), adaptive promises and fixed dwell times (APFD), and adaptive promises and dwell times (APAD). The parameters of the adaptive dwell time used in (23) are $\delta_d = 0.15$ and $\Delta_d = 0.3$. For agent $j \in \{1, \dots, 4\}$, the radius δ_j of the dynamic ball-radius rule of Remark IV.2 is $\delta_j(t) = 0.50\|u_j^{**}(X_{\mathcal{N}}^j(t)) - u_j^{\text{sf}}(x_j(t))\|_2 + 10^{-6}$. This plot shows the advantage of the team-triggered approach in terms of required communication over the self-triggered one and also shows the additional benefits of implementing the adaptive promises and dwell time. This is because by using the adaptive dwell time, agents decide to wait longer periods for new information while their neighbors are still moving. By using the adaptive promises, as agents near convergence, they are able to make increasingly tighter promises, which allows them to request information from each other less frequently. As Figure 4(b) shows, the network performance is not compromised despite the reduction in communication.

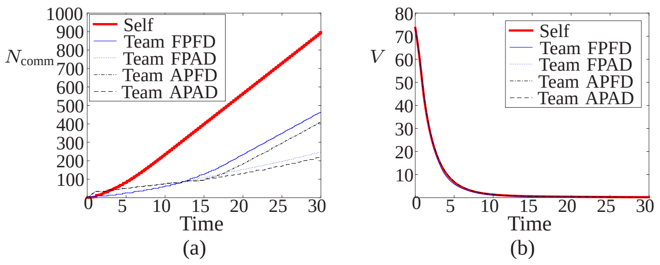
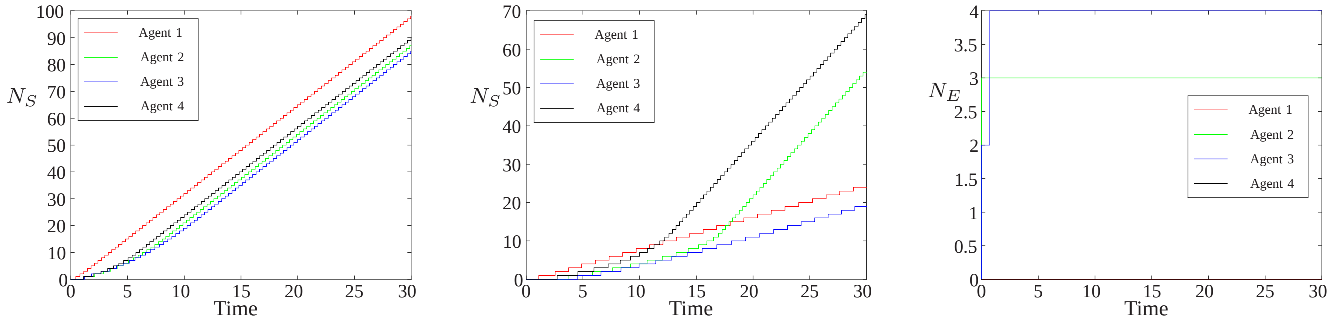


Fig. 4. Plots of (a) the total number of messages sent and (b) the evolution of the Lyapunov function V for executions of self-triggered approach and the team-triggered approaches with fixed promises and dwell times (FPFD), fixed promises and adaptive dwell times (FPAD), adaptive promises and fixed dwell times (APFD), and adaptive promises and dwell times (APAD).

VIII. CONCLUSIONS

We have proposed a novel approach, termed team-triggered, that combines ideas from event- and self-triggered control for the implementation of distributed coordination strategies for networked cyber-physical systems. Our approach is based



(a) (b) (c)

Fig. 2. Number of self-triggered requests made by each agent in an execution of the (a) self-triggered approach and (b) team-triggered approach with fixed dwell times and promises. For the latter execution, (c) depicts the number of event-triggered messages sent (broken promises) by each agent.

on agents making promises to each other about their future states. If a promise is broken, this triggers an event where the corresponding agent provides a new commitment. As a result, the information available to the agents is set-valued and can be used to schedule when in the future further updates are needed. We have provided a formal description and analysis of team-triggered coordination strategies and have also established robustness guarantees in scenarios where communication is unreliable. The proposed approach opens up numerous venues for future research. Among them, we highlight the robustness under disturbances and sensor noise, more general models for individual agents, the design of team-triggered implementations that guarantee the invariance of a desired set in distributed scenarios, the relaxation of the availability of the safe-mode control via controllers that allow agents to execute maneuvers that bring them back to their current state, relaxing the requirement on the negative semidefiniteness of the derivative of the Lyapunov function along the evolution of each individual agent, methods for the systematic design of controllers that operate on set-valued information models, understanding the implementation trade-offs in the design of promise rules, analytic guarantees on the performance improvements with respect to self-triggered strategies, and the impact of evolving topologies on the generation of promises.

References

[1] C. Nowzari and J. Cortés, "Team-triggered coordination of networked systems," in *American Control Conference*, (Washington, D.C.), pp. 3827–3832, June 2013.

[2] C. Nowzari and J. Cortés, "Robust team-triggered coordination of networked cyber-physical systems," in *Control of Cyber-Physical Systems* (D. C. Tarraf, ed.), vol. 449 of *Lecture Notes in Control and Information Sciences*, pp. 317–336, New York: Springer, 2013.

[3] K. D. Kim and P. R. Kumar, "Cyberphysical systems: A perspective at the centennial," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1287–1308, 2012.

[4] J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang, "Toward a science of cyberphysical system integration," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 29–44, 2012.

[5] D. Hristu and W. Levine, *Handbook of Networked and Embedded Control Systems*. Boston, MA: Birkhäuser, 2005.

[6] K. J. Åström and B. Wittenmark, *Computer Controlled Systems: Theory and Design*. Englewood Cliffs, NJ: Prentice Hall, 3rd ed., 1996.

[7] P. Wan and M. D. Lemmon, "Event-triggered distributed optimization in sensor networks," in *Symposium on Information Processing of Sensor Networks*, (San Francisco, CA), pp. 49–60, 2009.

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

[9] P. Tabuada, "Event-triggered real-time scheduling of stabilizing control tasks," *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1680–1685, 2007.

[10] 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*, vol. 81, no. 4, pp. 571–590, 2008.

[11] M. Velasco, P. Marti, and J. M. Fuertes, "The self triggered task model for real-time control systems," in *Proceedings of the 24th IEEE Real-Time Systems Symposium*, pp. 67–70, 2003.

[12] R. Subramanian and F. Fekri, "Sleep scheduling and lifetime maximization in sensor networks," in *Symposium on Information Processing of Sensor Networks*, (New York, NY), pp. 218–225, 2006.

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

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

[15] X. Wang and N. Hovakimyan, "L_1 adaptive control of event-triggered networked systems," in *American Control Conference*, (Baltimore, MD), pp. 2458–2463, 2010.

[16] M. C. F. Donkers and W. P. M. H. Heemels, "Output-based event-triggered control with guaranteed L_∞-gain and improved and decentralised event-triggering," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1362–1376, 2012.

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

[18] G. Shi and K. H. Johansson, "Multi-agent robust consensus-part II: application to event-triggered coordination," in *IEEE Conf. on Decision and Control*, (Orlando, FL), pp. 5738–5743, Dec. 2011.

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

[20] M. Mazo Jr. and P. Tabuada, "On event-triggered and self-triggered control over sensor/actuator networks," in *IEEE Conf. on Decision and Control*, (Cancun, Mexico), pp. 435–440, 2008.

[21] Y. Fan, G. Feng, Y. Wang, and C. Song, "Distributed event-triggered control of multi-agent systems with combinational measurements," *Automatica*, vol. 49, no. 2, pp. 671–675, 2013.

[22] A. Eqtami, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Event-triggered strategies for decentralized model predictive controllers," in *IFAC World Congress*, (Milano, Italy), Aug. 2011.

[23] E. Garcia and P. J. Antsaklis, "Model-based event-triggered control for systems with quantization and time-varying network delays," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 422–434, 2013.

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

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

[26] X. Wang and M. D. Lemmon, "Event-triggered broadcasting across dis-

tributed networked control systems," in *American Control Conference*, (Seattle, WA), pp. 3139–3144, June 2008.

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

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

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

[30] M. Althoff, C. L. Guernic, and B. H. Krogh, "Reachable set computation for uncertain time-varying linear systems," in *Proceedings of the 14th international conference on Hybrid Systems: Computation and Control*, (Chicago, IL), pp. 93–102, 2011.

[31] G. Frehse, "PHAVer: algorithmic verification of hybrid systems past HyTech," in *Hybrid Systems: Computation and Control* (M. Morari and L. Thiele, eds.), vol. 3414 of *Lecture Notes in Computer Science*, pp. 258–273, Heidelberg, Germany: Springer, 2005.

[32] F. Blancini and S. Miani, *Set-theoretic Methods in Control*. Boston, MA: Birkhäuser, 2008.

[33] J. Lin, A. S. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem. Part 2: The asynchronous case," *SIAM Journal on Control and Optimization*, vol. 46, no. 6, pp. 2120–2147, 2007.

[34] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*. Applied Mathematics Series, Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[35] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1997.

[36] D. Peleg, *Distributed Computing. A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications, SIAM, 2000.

[37] M. Guinaldo, D. Lehmann, J. S. Moreno, S. Dormido, and K. H. Johansson, "Distributed event-triggered control with network delays and packet losses," in *IEEE Conf. on Decision and Control*, (Hawaii, USA), pp. 1–6, Dec. 2012.

[38] L. Krick, M. E. Broucke, and B. Francis, "Stabilization of infinitesimally rigid formations of multi-robot networks," *International Journal of Control*, vol. 82, no. 3, pp. 423–439, 2009.

[39] F. Dorfler and B. Francis, "Geometric analysis of the formation problem for autonomous robots," *IEEE Transactions on Automatic Control*, vol. 55, no. 10, pp. 2379–2384, 2010.