Predicate Monitoring in Distributed Cyber-physical Systems *

Anik Momtaz[1], Niraj Basnet[2], Houssam Abbas[2], and Borzoo Bonakdarpour[1]

[1] Michigan State University, East Lansing MI 48824, USA
[2] Oregon State University, Corvallis OR, 97331, USA

Abstract. This paper solves the problem of detecting violations of predicates over *distributed* continuous-time and continuous-valued signals in cyber-physical systems (CPS). We assume a partially synchronous setting, where a clock synchronization algorithm guarantees a bound on clock drifts among all signals. We introduce a novel *retiming* method that allows reasoning about the correctness of predicates among continuous-time signals that do not share a global view of time. The resulting problem is encoded as an SMT problem and we introduce techniques to solve the SMT encoding efficiently. Leveraging simple knowledge of physical dynamics allows further runtime reductions. We fully implement our approach on two distributed CPS applications: monitoring of a network of autonomous ground vehicles, and a network of aerial vehicles. The results show that in some cases, it is even possible to monitor a distributed CPS sufficiently fast for online deployment on fleets of autonomous vehicles.

1 Introduction

As the environment we live in develops, so does our dependency on safety-critical *cyber-physical systems* (CPS), along with the need for verifying their correctness. A particularly critical class of CPS includes software applications *distributed* over networked nodes, which we will refer to as *agents*. Examples include fleets of autonomous vehicles, network of sensors in infrastructures, health-monitoring wearables, and networks of medical devices. While the literature of distributed computing is decades old, and many important problems have been solved in the context of *discrete-event systems*, we currently lack a solid understanding of distributed CPS, as they are differentiated by three characteristics. **First**, their signals are *analog*; these signals contain an uncountable infinity of events which makes existing reasoning techniques from the discrete settings inapplicable in most cases. The applications we target, such as those above, care about continuous-time behavior: for instance, it is not enough to say that a voltage does not spike at sample times. Thus, adjusting the signal sampling rate does nothing to address the need for reasoning about the *analog* signals. **Second**, each agent in these CPS has a *local* clock that drifts from other agents' clocks; thus,

* This work is sponsored in part by the United States NSF FMitF-1917979 and CCF-2118356 awards.

the notion of time, taken for granted in centralized systems, must be revised, since it is unclear when exactly events are sequential and concurrent. In fact, it is not clear how continuous events in different processes obey the *happened before* relation [21], and how one can reason about the order of occurrence of continuous events. Robust controllers do not address the issues of asynchrony in time. **Third**, CPS signals obey physical laws and dynamics. Even a rough knowledge of these dynamics might be leveraged to reason about distributed signals and predict their behavior, thus increasing efficiency. In this paper, we take the first step towards rigorous, automated reasoning about distributed CPS whose *correctness* and integrity is vital to guaranteeing the safety of the environment they operate in. A popular and practical approach to reason about the health of CPS is to *monitor* them with respect to their formal specification, and detect violations. Currently, we lack techniques for monitoring CPS where analog signals are produced by distributed agents that do not share a global clock (see the related work in Section 7). Lack of synchrony in particular creates significant challenges, as the monitor has to reason about signal values at local times of different agents, which may lead to inconsistent monitoring verdicts. This difficulty is compounded by the fact that agents typically communicate with each other, which imposes additional constraints on event ordering.

Motivating Example. We illustrate the urgent need for monitoring distributed CPS by a critical application in automated air traffic control (AATC). The market for unmanned aerial vehicles (UAVs) is witnessing explosive growth [18]. The Federal Aviation Administration (FAA) in the United States is envisioning a federated framework, in which UAVs that collaborate in monitoring *global* air safety properties are rewarded with faster free-flight paths to their destinations [15,14]. To enable this federated framework, analog signals like UAV position and velocity must be monitored by the ATC tower software to see whether they violate *global* instantaneous safety properties, or *predicates*. These predicates are Boolean expressions defined over the simultaneous states of the different CPS agents, like mutual separation between agents, conditional speed limits, and minimal energy storage. These predicates must be evaluated on the global state, which is the state of all UAVs combined at the *same moment in time*. However, the absence of a perfect shared clock among all UAVs may result in a situation where UAV1's clock indicates $t = 5$ and UAV2's clock indicates $t = 5.2$, at the same physical 'real' moment. Equivalently, the same value on both clocks might mean different physical moments. If the central ATC monitor uses these two states to evaluate whether the predicate is violated, it might lead to false negatives (i.e., missing violations) or false positives (i.e., declaring a violation when none exists).

The UAV example has two characteristics that are present in many other distributed CPS: first, while it is generally impossible to guarantee perfect continuous-time synchrony, clock synchronization algorithms such as NTP [23] ensure that the drift among local clocks remains within some bounds. Secondly, it is often the case that the central monitor knows some bounds on the UAV dynamics, like velocity limits. In this example, the ATC tower itself would know the UAVs' speed limits. We leverage these two characteristics in developing our solution.

1.1 Our Solution and Contributions

In this paper, we propose a sound and complete solution to the problem of predicate monitoring for distributed CPS. Our contributions are as follows:

1. a Satisfiability Modulo Theory (SMT) based algorithm for centralized monitoring of distributed analog signals for predicate violations, augmented with a clock synchronization algorithm that guarantees *bounded skew* ε between all local clocks, using the classic *happened-before* relation [21];
2. a *retiming* technique that borrows the notion of retiming functions from stochastic processes;
3. a lightweight mechanism for incorporating bounds on system dynamics to reduce monitoring overhead;
4. an analysis of the sensitivity of monitoring overhead to the skew bound and the amount of communication between agents, and
5. a technique for parallelizing the monitoring algorithm to improve scalability.

We have fully implemented our techniques and report results of experiments on monitoring a network of autonomous ground vehicles (real-world experiment) and aerial vehicles (in simulation). It should be mentioned that due to using a central monitor, naturally the system is susceptible to a single point of failure. This paper is concerned in developing the proposed theory, not account for fault tolerance. We make the following observations. First, although our approach is based on SMT solving, it can be employed for online monitoring when the monitor is invoked with appropriate frequency (i.e., the monitoring overhead does not surpass the normal operation time of the system). Second, incorporating the knowledge of system dynamics is highly beneficial in reducing the overhead of monitoring. In some cases it leads to a speedup by one order of magnitude. Finally, monitoring overhead is *in*dependent of the clock skews when practical clock synchronization protocols (e.g., NTP and PTP) are applied.

2 Model of Computation

We first set some notation. The set of reals is $\mathbb{R}$, the set of non-negative reals is $\mathbb{R}_+$, and the set of positive reals is $\mathbb{R}_+^*$. The integer set $\{1, \ldots, N\}$ is abbreviated as $[N]$. *Global* time values (kept by an *imaginary* global clock) are denoted by χ, χ', χ_1, χ_2, etc, while the letters t, t', t_1, t_2, s, s', s_1, s_2, etc. denote *local* clock values specific to given agents which will always be clear from the context.

2.1 Signal Model

In this section, we introduce our signal model, i.e., our model of the output signal of an agent. Monitoring can be done regardless of the dynamics of the agents. However, as we see later, a rough knowledge of the dynamics can be helpful.

Definition 1. *An* output signal *(of some agent A) is a function* $x : [a, b] \to \mathbb{R}^d$*, which is right-continuous, left-limited, and is not Zeno. Here,* $[a, b]$ *is an interval in* $\mathbb{R}_+$*, and will be referred to as the timeline of the signal.* ■

Without loss of generality, we will henceforth assume that x is one-dimensional, i.e., $d = 1$. *Right-continuity* means that at all t in its support, $\lim_{s \to t_+} x(s) = x(t)$. *Left-limitedness* means the function has a finite left-limit at every t in its support: $\lim_{s \to t_-} x(s) < \infty$. *Not being Zeno* means that x has a finite number of discontinuities in any bounded interval in its support. This ensures that the signal cannot jump infinitely often in a finite amount of time. A *discontinuity* in a signal $x(\cdot)$ can be due to a discrete event internal to agent A (like a variable updated by software), or to a message sent to or received from another agent A'.

We assume a loosely coupled system with asynchronous message passing. Specifically, the system consists of N reliable *agents* that do not fail, denoted by $\{A_1, A_2, \ldots, A_N\}$, without any shared memory or global clock. The output signal of agent A_n is denoted by x_n, for $1 \leq n \leq N$. Agents can communicate via FIFO lossless channels. The contents of a message are immaterial to our purposes. We will need to refer to some global clock which acts as a '*real*' time-keeper. However, this global clock is a theoretical object used in definitions and theorems, and is *not* available to the agents. We make two assumptions:

- *(A1) Partial synchrony.* The *local clock* (or time) of an agent A_n can be represented as an increasing function $c_n : \mathbb{R}_+ \to \mathbb{R}_+$, where $c_n(\chi)$ is the value of the local clock at global time χ. Then, for any two agents A_n and A_m, we have:
$$\forall \chi \in \mathbb{R}_+ . |c_n(\chi) - c_m(\chi)| < \varepsilon$$
with $\varepsilon > 0$ being the maximum *clock skew*. The value ε is assumed fixed and known by the monitor in the rest of this paper. In the sequel, we make it explicit when we refer to 'local' or 'global' time.
- *(A2) Deadlock-freedom.* The agents being analyzed do not deadlock.

Assumption (A1) is met by using a clock synchronization algorithm, like NTP [23], to ensure bounded clock skew among all agents.

In the discrete-time setting, an event is a value change in an agent's variables. We now update this definition for the continuous-time setting of this paper. Specifically, in an agent A_n, an *event* is either a (i) a pair $(t, x_n(t))$, where t is the local time (i.e., returned by function c_n); (ii) a message transmission, or (iii) a message reception. There is no assumption on the messages that the agents send to each other. Messages that are sent to the monitor are timestamped by their respective local clocks. Since the agents evolve in continuous time and their output signals are defined for all local times t, a message transmission or reception always coincides with a signal value; i.e., if A_n receives a message at local time t, its signal has value $x_n(t)$ at that time. Thus, without loss of generality, every event will be represented as a (local time, value) pair $(t, x_n(t))$, often abbreviated as e^n_t (n and t will be omitted when irrelevant).

A *distributed signal* is modeled as a set of events partially ordered by Lamport's *happened-before* ($\rightsquigarrow$) relation [21], extended by our assumption (A1) on bounded clock skew among all agents. Namely, let

$$E = \{e_t^n \mid n \in [N] \wedge I_n \subseteq \mathbb{R}_+ \ \wedge \ t \in I_n\}$$

denote a set of events, where t is local time in agent A_n, and set I_n is a bounded nonempty interval. The following defines a continuous-time distributed signal under partial synchrony.

Definition 2. *A* distributed signal *on N agents is a tuple $(E, \rightsquigarrow)$, where E is a set of events obeying the restriction: for every $n \in [N]$. The relation $\rightsquigarrow$ is a relation between events such that:*

(1) In every agent A_n, all events are totally ordered, that is,

$$\forall t, t' \in I_n : (t < t') \to (e_t^n \rightsquigarrow e_{t'}^n).$$

(2) If e is a message send event in an agent and f is the corresponding receive event by another agent, then we have $e \rightsquigarrow f$.
(3) For any two events $e_t^n, e_{t'}^m \in E$, if $t + \varepsilon < t'$, then $e_t^n \rightsquigarrow e_{t'}^m$.
(4) If $e \rightsquigarrow f$ and $f \rightsquigarrow g$, then $e \rightsquigarrow g$. ■

Figure. 1 shows an example. The classical case of complete asynchrony is recovered by setting $\varepsilon = \infty$. The restriction on I_n is necessary in the continuous-time setting and will be revisited in the next section.

Because the agents are only synchronized to within an ε, it is not possible to actually evaluate all signals at the same moment in global time. The notion of *consistent cut* and its frontier, defined next, capture *possible* global states: that is, states that could be valid global states (see Fig. 1).

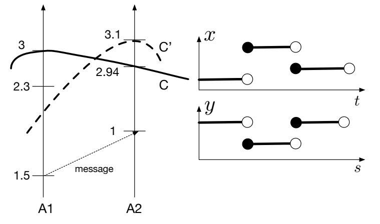


Fig. 1: Two partially synchronous continuous concurrent timelines with $\varepsilon = 0.1$, and corresponding signals x and y. (Solid dot indicates signal value at discontinuity). C is a consistent cut but C' is not.

Definition 3 (Consistent Cut). *Given a distributed signal $(E, \rightsquigarrow)$, a subset of events $C \subseteq E$ is said to form a* consistent cut *if and only if when C contains an event e, then it contains all events that happened-before e. Formally,*

$$\forall e \in E \ . \ (e \in C) \wedge (f \rightsquigarrow e) \Rightarrow f \in C. \qquad ■$$

From this definition and Definition 2.(3) it follows that if $e_{t'}^m$ is in C, then C also contains every event e_t^n s.t. $t + \varepsilon < t'$.

A consistent cut C can be represented by its *frontier* $\mathsf{front}(C) = \left(e_{t_1}^1, \ldots, e_{t_N}^N\right)$, in which each $e_{t_n}^n$ is the last event of agent A_n appearing in C. Formally:

$$\forall n \in [N] \ . \ e_{t_n}^n \in C \text{ and } t_n = \max\{t \in I_n \mid \exists e_t^n \in C\}$$

Example 1. Figure 1 shows two timelines, generated by two agents executing concurrently. Every moment in each timeline corresponds to an event e^n_t, where $n \in \{1, 2\}$. An arrow between the timelines indicates a message transmission and reception. Thus, we may see that the following hold: $e^1_{1.5} \rightsquigarrow e^1_{2.3}$, $e^2_1 \rightsquigarrow e^2_{3.1}$, and $e^1_{1.5} \rightsquigarrow e^2_1$. Assuming $\varepsilon = 0.1$, it comes that all events below (thus, before) the solid arc form a consistent cut C with frontier $\mathsf{front}(C) = (e^1_3, e^2_{2.94})$, On the other hand, all events below the dashed arc do *not* form a consistent cut since $e^1_{2.3} \rightsquigarrow e^2_{3.1}$ and $e^2_{3.1}$ is in the set C', but $e^1_{2.3}$ is not in C'.

2.2 Signal Transmission to the Monitor

Communication between nodes necessarily involves sampling the analog signal, transmitting the samples, and reconstructing the signal at the receiving node. Our objective is to monitor the *reconstructed analog signals*. This is different from monitoring a discrete-time signal consisting of the samples – the applications we target actually care about the value of the signal between samples, and potential violations they reveal. Methods for signal transmission, including sampling and reconstruction, are standard in Communication theory. Errors due to sampling and reconstruction (say, because of bandwidth limitations) can be accounted for by strictifying the STL formula using the methods of [16]. The choice of reconstruction algorithm is application-dependent and follows from domain knowledge. In this paper's experiments, we assume that every output signal x_n is reconstructed as *piece-wise linear* between the samples. We emphasize that other reconstructions, like cubic splines, can also be used with simple modification to our algorithms at the cost of additional runtime, and that the choice of reconstruction is orthogonal to our techniques and this paper's objectives. Since we assume the agents do not deadlock, this transmission happens in *segments* of length T: at the k^{th} transmission, agent A_n transmits $x_n|_{[(k-1)T,kT]}$, the restriction of its output signal to the interval $[(k-1)T, kT]$ as measured by its local clock. In the rest of this paper, we refer exclusively to the signal fragments received by the monitor in a given transmission.

We now re-visit the restriction placed on I_n in Definition 2, namely, that it is a non-empty bounded interval. Non-emptiness models that computation does not deadlock. That I_n is an interval expresses that no events are missed, or equivalently, that signal reconstruction is perfect at the monitor. The restriction that it be bounded models the above monitoring setup: the monitor is only ever dealing with bounded signal fragments $x_n|_{[(k-1)T,kT]}$, so

$$I_n = [(k-1)T, kT] \tag{1}$$

for every agent at the k^{th} transmission, *measured in local time*. By the bounded skew assumption, we have:

Lemma 1. *For any two agents A_n, A_m, $|\min I_n - \min I_m| \leq \varepsilon$ and $|\max I_n - \max I_m| \leq \varepsilon$.* ■

3 The Predicate Monitoring Problem

Many system requirements are often captured via predicates (e.g., invariants). A *predicate* ϕ is a global Boolean-valued function over the signal values of agents. For instance, $\phi(x_1, x_2) = (x_1 > 0) \wedge (\ln(x_2) < 3)$ is a predicate on two signals that evaluates to true when $x_1 > 0$ and $\ln(x_2) < 3$, otherwise false.

Because the agents are partially synchronized to within an ε, it is not possible to actually evaluate all signals at the same moment in global time. However, as noted above, the frontier of a consistent cut gives us a possible global state.

Definition 4 (Distributed satisfaction). *Given a distributed signal* $(E, \rightsquigarrow)$ *over* N *agents, and a predicate* ϕ *over the* N *agents, we say that* $(E, \rightsquigarrow)$ satisfies ϕ *iff for all consistent cuts* $C \subseteq E$ *with*

$$\mathsf{front}(C) = \Big((t_1, x_1(t_1)), \ldots, (t_N, x_N(t_N)) \Big)$$

we have $\phi\big(x_1(t_1), x_2(t_2), \ldots, x_N(t_N)\big) = \mathsf{true}$. *We write this as* $(E, \rightsquigarrow) \models \phi$. ■

Thus, we formally define the problem as follows.

Problem Statement

Continuous-Time Monitoring of Distributed CPS. Given a distributed signal $(E, \rightsquigarrow)$ and a predicate ϕ over N agents, determine whether $(E, \rightsquigarrow) \models \phi$.

When a distributed signal $(E, \rightsquigarrow)$ does not satisfy a predicate ϕ, we say that $(E, \rightsquigarrow)$ *violates* ϕ and write $(E, \rightsquigarrow) \not\models \phi$. In this paper, we want to detect whether there exists a consistent cut $C \subseteq E$, such that $(E, \rightsquigarrow) \not\models \phi$.

The main challenge in monitoring distributed signals is that the monitor has to reason about signals that are subject to time asynchrony. For instance, consider two signals x_1 and x_2 and the case where $x_1(2) = 5$, $x_2(3) = 1$, $\phi(x_1, x_2) = (x_1 > 4) \wedge (x_2 < 0)$, and $\epsilon = 2$ so that time points 2 and 3 form a consistent cut. In this case, since the above signal values are at local times within the possible clock skew, one has to (conservatively) consider that the predicate is violated. In the next section, we present our solution to the problem.

4 SMT-based Monitoring Algorithm

In a nutshell, our solution has the following features:

- **Central monitor.** We assume that there is a *central* monitor that solves, at regular intervals, the monitoring problem described in Section 3.
- **Signal retiming.** As signals are measured using their local clocks, the monitor should somehow align them to detect possible violations of the predicate. To this end, we propose a *retiming* technique that establishes the happened-before relation in the continuous-time setting, and stretches or compacts signals to align them with each other within the ε clock skew bound.

– **SMT encoding.** We transform the monitoring decision problem into an SMT-solving problem, whose components (like input signals and the happened-before relation) are modeled as SMT entities and constraints.

4.1 Retiming Functions

Our signal model is continuous-time, that is, the signals are maps from $\mathbb{R}_+$ to $\mathbb{R}_+$. Therefore, to model the approximate re-synchronizing action of the monitor, we use a *retiming function.*

Definition 5 (Retiming functions). *A* retiming *function, or simply retiming, is an increasing function $\rho : \mathbb{R}_+ \to \mathbb{R}_+$. An ε-retiming is a retiming function such that: $\forall t \in \mathbb{R}_+ : |t - \rho(t)| < \varepsilon$. Given a distributed signal $(E, \rightsquigarrow)$ over N agents and any two distinct agents A_i, A_j, where $i, j \in [N]$, a retiming ρ from A_j to A_i is said to* respect $\rightsquigarrow$ *if we have $(e^i_t \rightsquigarrow e^j_{t'}) \Rightarrow (t < \rho(t'))$ for any two events $e^i_t, e^j_{t'} \in E$.* ■

Fig. 2 shows examples of retimings and how they relate to predicate monitoring. To detect predicate violation, we must first retime y to the t axis via a retiming map ρ. (c) shows three different retimings, including the identity. (d)-(e) show the retimed y. For the predicate $x > y$, (e)-(f) show no violations, but (d) does. The conservative monitoring answer is that the predicate is violated. An ε-retiming ρ maps $\mathbb{R}_+$ to itself, but it is easy to see that the restriction of ρ to a bounded interval I is an increasing function from I to $\rho(I)$ that respects the constraint $|t - \rho(t)| < \varepsilon$ for all $t \in I$. Thus, in what follows we restrict our attention to the action of ε-retimings on bounded intervals.

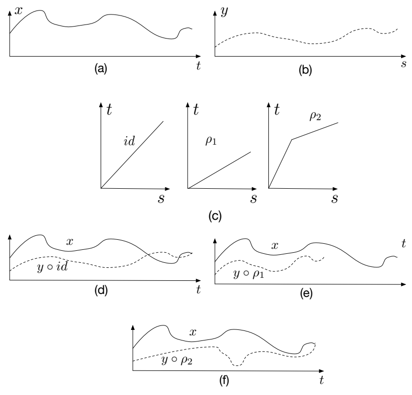


Fig. 2: Predicate violation between two signals x and y measured using partially synchronized clocks t and s.

We now state and prove the main technical result of this paper, which relates the existence of consistent cuts in distributed signals to the existence of retimings between the agents' local clocks.

Proposition 1. *Given a predicate ϕ and distributed signals $(E, \rightsquigarrow)$ over N agent, there exists a consistent cut $C \subseteq E$ that violates ϕ* if and only if *there exists a finite A_1-local clock value t and $N-1$ ε-retimings $\rho_n : I_n \to I_1$ that respect $\rightsquigarrow$, $2 \le n \le N$, such that:*

$$\phi\Big(x_1(t), x_2 \circ \rho_2^{-1}(t), \ldots, x_N \circ \rho_N^{-1}(t)\Big) = \mathsf{false} \tag{2}$$

and such that $\rho_m^{-1} \circ \rho_n : I_n \to I_n$ is an ε-retiming for all $n \neq m$. Here, '$\circ$' denotes the function composition operator. ■

Proof. ($\Leftarrow$) Suppose that such retimings exist. Define the local time values $t_1 := t$, $t_n = \rho_n^{-1}(t)$, $2 \leq n \leq N$, and the set $C = \{e_t^n \mid t \leq t_n\}$. By the construction of C and the fact that the retimings respect $\rightsquigarrow$, it holds that if $e \in C$ and $f \rightsquigarrow e$ then $f \in C$. For every $n, m \geq 2, n \neq m$, it holds that $t_m = \rho_m^{-1}(\rho_n(t_n))$ so $|t_n - t_m| \leq \varepsilon$. Thus C is a consistent cut with frontier $(e_{t_n}^n)_{n=1}^N$ that witnesses the violation of ϕ.

($\Rightarrow$) Suppose now that there exists a consistent cut C with frontier:

$$\mathsf{front}(C) = \Big((t_1, x_1(t_1)), \ldots, (t_N, x_N(t_N)) \Big)$$

that witnesses violation of ϕ. We need the following facts.

Fact 1. For every two events $e_{t_n}^n$ and $e_{t_m}^m$ in the frontier of a consistent cut, we have $|t_n - t_m| \leq \varepsilon$. Indeed, since $e_{t_n}^n \in \mathsf{front}(C)$, we have $e_s^m \in C$ for all s s.t. $s + \varepsilon \leq t_n$. Thus, $t_m \geq s$ for all such s and so $t_m \geq t_n - \varepsilon$. By symmetry of the argument, $t_n \geq t_m - \varepsilon$ holds as well.

Fact 2. Given intervals $[a, b]$ and $[c, d]$ s.t. $|a - c| \leq \varepsilon$ and $|b - d| \leq \varepsilon$, the map $L : [a, b] \to [c, d]$ defined by $L(t) = c + \frac{d-c}{b-a}(t - a)$ is a linear ε-retiming. This is immediate.

Suppose first that there are no message exchanges. For $2 \leq n \leq N$, we define the retiming $\rho_n : I_n \to I_1$ in two pieces. First, set $\rho_n(t_n) = t_1$. By preceding lemma, $|t_n - t_1| \leq \varepsilon$. Write $I_1 = [a, b]$ and $I_n = [c, d]$ for notational simplicity in this proof. Call a pair of intervals that satisfies the hypothesis of Fact 2 an *admissible pair*. Then, the following pairs are clearly admissible by Lemma 1: $[a, t_1]$ and $[c, t_n]$, and $[t_1, b]$ and $[t_n, d]$. Thus, there exist two linear retimings $L_n : [a, t_1] \to [c, t_n]$ and $L'_n : [t_1, b] \to [t_n, d]$, and we can define a piece-wise ρ_n: $\rho_n(t) = L_n(t)$ on $c \leq t \leq t_n$ and $\rho_n(t) = L'_n(t)$ on $t_n \leq t \leq d$. It is easy to establish that ρ_n is an ε-retiming.

It remains to show that $\rho_n^{-1} \circ \rho_m : I_m = [f, g] \to [c, d]$ is also an ε-retiming. This too can be established in parts, first over $[f, t_m]$ then over $[t_m, g]$, using the same arguments as above and exploiting the linearity of these retimings. For instance, if we write α_n for the slope of L_n, then over $[f, t_m]$

$$\begin{aligned}\rho_n^{-1}(\rho_m(s)) &= L_n^{-1}(L_m(s)) = L_n^{-1}(a + \alpha_m(s - c)) \\ &= \frac{1}{\alpha_n}[a + \alpha_m(s - c)] + f - a/\alpha_n = f + \frac{g - f}{d - c}(s - c)\end{aligned}$$

which is a linear ε-retiming by Fact 2.

If there are message exchanges, the above argument still applies but over a more fine-grained division of the timelines I_n obtained by partitioning each timeline at message transmission times. We sketch the proof: for the admissible pair $I_1 = [a, b]$ and $I_n = [c, d]$, suppose the first message is sent from A_n to A_1 at local time $s < t_n$ and is received at local time $r < t_1$. Define $t_{(s)} := \min(s + \varepsilon, r)$. Then the pair $[a, t_{(s)}]$, $[c, s]$ is admissible. Repeat this process for all messages. We end up with a collection of admissible pairs that can be retimed to each other, as above, without violating the $\rightsquigarrow$ relation. These are concatenated to yield the desired retiming ρ_n. ■

Thus, finding a consistent cut that violates the predicate can be achieved by finding such retimings. The proof of Prop. 1 further shows that the retimings can always be chosen as piece-wise linear (rather than any increasing function), which yields significant runtime savings in the SMT encoding in the next section.

Remark 1. An interesting consequence of Fact 2 in the proof is that it is enough to use piece-wise linear retimings. This results in the following concrete problem.

Concrete Problem Statement

Given $\varepsilon > 0$, a distributed signal $(E, \rightsquigarrow)$ over N agents, and a predicate ϕ over the N agents, find $N-1$ ε-retiming functions $\rho_2, \dots, \rho_N$ that satisfy the hypotheses of Prop. 1 and s.t.

$$\phi\Big(x_1(t_1), x_2(t_2), \dots, x_N(t_N)\Big) = \mathsf{false} \tag{3}$$

4.2 SMT Formulation

We solve the monitoring problem by transforming it into an instance of *satisfiability modulo theory* (SMT). Specifically, we ask whether there exists $N-1$ retimings, such that (3) holds; equivalently, whether there exists a consistent cut that witnesses satisfaction of $\neg\phi$.

Without loss of generality, we start with our encoding of two agents, A_1 and A_2 (shown in Fig. 1). A_1 outputs signal x supported over the bounded timeline I_1, which is discretized to $D_1 \subset I_1$ and sent to the monitor. Similarly, A_2 outputs signal y supported over the bounded timeline I_2, which is discretized to $D_2 \subset I_2$ and sent to the monitor. D_1 and D_2 are finite. Let $\delta_k > 0$ be the *sampling period* of agent A_k, so two consecutive elements of D_k differ by δ_k, $k \in \{1, 2\}$.

Consider further that A_2 transmits a message at local time t_1 and it is received by A_1 at local time t_2, and that A_1 sends a message at local time t_3 which is received by A_2 at local time t_4. The distributed signal violates the predicate iff the following SMT problem returns SAT.

SMT entities. In our encoding, the entities are the retimings ρ_n included as *uninterpreted* functions (the solver will interpret), signals x and y, intervals I_1 and I_2, real numbers t, s, s', t_1, t_2, t_3, and t_4. All these entities have been defined in the previous sections. The following quantities are all constants in the encoding, since they are known to the monitor: the sampling time sets D_k and sampling periods δ_k, the sampled values $\{x(t_i) \mid t_i \in D_1\}$ and $\{y(s_i) \mid s_i \in D_2\}$, and the message transmission and reception local times.

SMT constraints. The encoding is a conjunction of the following constraints:

– *(Predicate violation)* The first constraint 'finds' local times t and s at which predicate ϕ is violated (upto ε-synchrony):

$$\begin{aligned}
&\exists\, t \in I_1 .\, \exists s \in I_2 . && \text{(4a)}\\
&\quad\Big(\exists t^- \in D_1 .\; t^- \le t \le t^- + \delta_1\Big) \;\wedge && \text{(4b)}\\
&\quad\Big(\exists s^- \in D_2 \;.\; s^- \le s \le s^- + \delta_2\Big) \;\wedge && \text{(4c)}\\
&\quad\Big(\rho(s) = t\Big) \;\wedge && \text{(4d)}\\
&\quad\Big(\neg\phi(x(t^-), y(s^-))\Big) && \text{(4e)}
\end{aligned}$$

Eq. (4b) finds the time sample t^- such that $x(t) = x(t^-)$: this is the result of our assumption that signals are piece-wise constant. Eq.(4c) does the same for y. Eq. (4d) specifies that s is retimed to t: this is what guarantees that $(x(t), y(s))$ is a possible global state as per Proposition 1. Eq. (4e) checks violation of the predicate at $(x(t), y(s)) = (x(t^-), y(s^-))$.

– *(Valid retiming)* Eq. (5) ensures that ρ is a valid ε-retiming from I_2 to I_1:

$$\forall s \in I_2 .\; \exists t \in I_1 .\; (\rho(s) = t) \wedge (|t - s| < \varepsilon) \tag{5}$$

and Eq. (6) ensures that the retiming function is increasing:

$$\forall s \in I_2 .\; \forall s' \in I_2 .\, \Big(s < s' \;\Rightarrow\; \rho(s) < \rho(s')\Big) \tag{6}$$

– *(Happened-before)* Eq. (7) enforces the happened-before relation for message transmissions:

$$\Big(\rho(t_1) < t_2\Big) \;\wedge\; \Big(t_3 < \rho(t_4)\Big) \tag{7}$$

– *(Inverse retiming)* When there are more than 2 agents, we must also encode the constraint that for all $n \neq m$, $\rho_m^{-1} \circ \rho_n$ is an ε-retiming. Thus, for all $n \neq m$, letting f_m be the uninterpreted function that represents the inverse of the uninterpreted ρ_m, we add

$$\forall t \in I_n \cdot\;\; f_m(\rho_n(t)) = t \tag{8}$$

in addition to the analogs of Eqs. (6) and (5) for $f_m \circ \rho_n$.

Other signal models. If output signals were piece-wise linear, say, Eq. (4e) would be modified accordingly:

$$\begin{aligned}
\phi\Bigg(&x(t^-) + \frac{x(t^- + \delta_1) - x(t^-)}{\delta_1}(t - t^-),\\
&y(s^-) + \frac{y(s^- + \delta_2) - y(s^-)}{\delta_2}(s - s^-)\Bigg) = \mathsf{false}
\end{aligned} \tag{9}$$

Our choice of signal models is limited by the SMT solver: it must be able to handle the corresponding interpolation equations, like the piece-wise linear interpolation in Eq. (9). As an example, in Fig. 3, let x and y be two signals, where the violating predicate ϕ to be monitored is $x(t) = y(s)$. Let ρ be a retiming of y on x, such that $\rho(s^-) = t^-$ and $\rho(s^- + \delta_2) = t^- + \delta_1$. It can be observed that although the discretized signal samples do not violate ϕ, due to the signals being piece-wise linear, it is easy to identify a violation at time t and s on signals x and y respectively, where $x(t) = 3$, $y(s) = 3$ and $\rho(s) = t$.

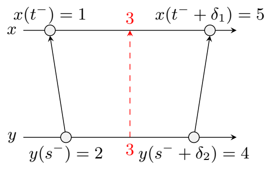


Fig. 3: Piece-wise Linear Interpolation

It is worth mentioning that restricting the SMT search to piece-wise linear retimings results in a significant decrease in run time, compared to the approach where the SMT is tasked with determining an interpolation. For example, for two UAVs with $\varepsilon = 1ms$ over $5s$-long signals, at segment count 5, the search for a general retiming requires $3.42s$, whereas searching for a piece-wise linear retiming requires only $1.01s$. Since, by Remark 1, there is no loss of generality in this restriction, from this point, all the reported experiments are obtained using the piece-wise linear retiming approach.

Remark 2. (i) $\rho_m^{-1} \circ \rho_n$ respects $\rightsquigarrow$ automatically so it is not necessary to encode that explicitly. (ii) Because we can restrict the SMT search to piece-wise linear retimings (see remark following proof of Prop. 1), constraint (8) can be simplified, namely, the expression for the inverse can be hard-coded. We don't show this to maintain clarity of exposition.

5 Exploiting the Knowledge of System Dynamics

Physical processes in a CPS follow the laws of physics. A runtime monitor can leverage this knowledge of the CPS dynamics to make monitoring more efficient.

We explain our idea by the following example (see Fig. 4). From knowing the rate bound $|\dot{x}| \leq 1$ (shown by a dashed line), the monitor concludes that the earliest x can satisfy the atom $x \leq 3$ is τ_1. Similarly for y. Given that $\tau_1 > \tau_2$, the monitor discards, roughly speaking, the fragment $[0, \tau_2]$ from each signal and monitors the remaining pieces. Note that $x(0) = 1$ and $y(0) = 2$.

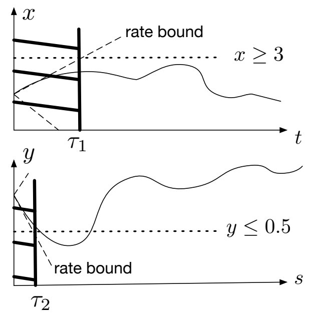


Fig. 4: Leveraging dynamics.

Consider the predicate: $\phi = \neg(a \vee b)$, where $a := x \geq 3$ and $b := y \leq 0.5$. Let a and b be *atoms* of predicate ϕ. There are 3 Boolean assignments to atoms a and b that falsify the predicate. Let us fix one such assignment, $a = b = \texttt{true}$. If the monitor knows a uniform bound on the rate of change $\dot{x}$ of x, say $\forall t. |\dot{x}(t)| \leq 1$, then it can infer that $a = \texttt{true}$ cannot hold before $\tau_1 = 2$ (local time). Similarly, if the monitor knows that $|\dot{y}| \leq 3$, then $b = \texttt{true}$ cannot hold before $\tau_2 = 0.5$

(local time). Taking into account the ε-synchrony, the monitor can limit itself to monitoring $x|_{[2,T]}$ (the restriction of x to $[2,T]$) and $y|_{[2-\varepsilon,T+\varepsilon]}$.

Now, if this yields UNSAT in the SMT instance, we select the next Boolean assignment (in terms of atoms a and b) that falsifies predicate ϕ (e.g., $a = \mathsf{false}$ and $b = \mathsf{true}$), derive the useful portion of signals x and y, and repeat the same procedure until the answer to the SMT instance is affirmative or all falsifying Boolean assignments are exhausted. Of course, this requires exploring all such assignments to atoms of the predicate, but since we expect the number of atoms in realistic predicates to be relatively small, the exhaustive nature of falsifying Boolean assignments will not be a bottleneck. We generalize this idea to N agents and arbitrary predicates in Algorithm 1. We assume without loss of generality that every atom a that appears in ϕ is of the form $x_n \geq v_a$ for some n and $v_a \in \mathbb{R}$. A Boolean assignment is a map σ from atoms to $\{\mathsf{false}, \mathsf{true}\}$, and a violating assignment is one that makes the predicate false. Thus, given a violating assignment σ, for every atom a, $a = \sigma(a)$ iff $x_n \geq v_a$ (if $\sigma(a) = \mathsf{true}$) or $x_n < v_a$ (if $\sigma(a) = \mathsf{false}$). Obvious modification to Algorithm 1 allows the monitor to take advantage of knowing different rate bounds at different points along the signals.

Algorithm 1: Dynamics-aware monitoring.

```
Data: Distributed signal $(E, \rightsquigarrow)$, $\varepsilon$,
      predicate $\phi$, bounds
      $|\dot{x}_n| \leq b_n, n \in [N]$
Result: $(E, \rightsquigarrow) \models \phi$
1  Set $t_n = \min I_n, n \in [N]$
2  while not done do
3      Get next violating assignment $\sigma$ to
         the atoms of $\phi$
4      if there are no more violating
         assignments then
5          done
6      else
7          for every atom $a$ in $\phi$ do
8              if $\sigma(a) = \mathsf{true}$ then
9                  $\tau_n = \min\{\tau \mid x(t_n+\tau) \geq$
                     $v_a\}, n \in [N]$
10             else
11                 $\tau_n = \min\{\tau \mid x(t_n+\tau) <$
                     $v_a\}, n \in [N]$
12         end
13         Set $\tau = \max_n \tau_n$ and
             $m = \mathrm{argmax}_n \tau_n$
14         SMT-monitor the distributed
             signal $E_\sigma$ made of the
             restrictions $x_n|_{[t_n+\tau-\varepsilon, \max I_n]}$,
             $n \neq m$ and $x_m|_{[t_m+\tau, \max I_m]}$
15         If SAT, done.
16     end
17 end
```

6 Case Studies and Evaluation

In this section, we evaluate our technique using two case studies on networks of autonomous ground and aerial vehicles.

6.1 Case Studies

Network of Ground Autonomous Vehicles We collected data from two $1/10^{th}$-scale autonomous cars competing in a race around a closed track. Each car carries a LiDAR for perceiving the world, and uses Wi-Fi antennas to communicate with the central monitor. Each car is running a model predictive controller to track its racing line and RRT to adjust its path. The trajectory data is sampled at $25Hz$. In this application, the useful signal length to monitor is $1-2s$, as this is the control horizon (i.e., the controller repeatedly plans the next $1-2s$). Thus, in Eq. (1), $T=1-2s$. A reasonable range for ε is interval $[1,5]ms$, guaranteed by ROS clock synchronization based on NTP. Unless otherwise indicated, we monitor the predicate $d(x_1,x_2) > \delta \wedge d(x_1,x_2) \leq \Delta$.

Network of UAVs We use Fly-by-Logic [27], a path planner software for UAVs, to simulate the operation of two UAVs performing various reach-avoid missions. In a reach-avoid mission, each UAV must reach a goal within a deadline, and must avoid static obstacles as well as other UAVs. The path planner uses a temporal logic robustness optimizer to find the most robust trajectory. The trajectories are sampled at $20Hz$. In this application, the useful signal length to monitor is around $2s$, as this is the UAV's 'reaction time' (depending on current speed). Thus, in Eq. (1), $T \approxeq 2s$. A reasonable range for ε is again $1-5ms$, guaranteed by ROS. Unless otherwise indicated, we monitor the predicate $d(x_1,x_2) \geq \delta$.

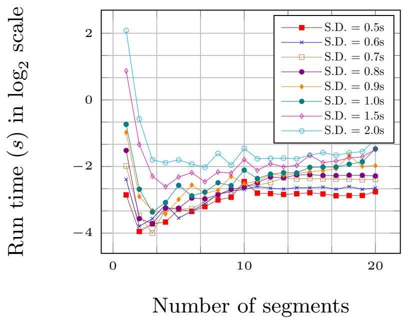
Note that the SMT solver's effort is mostly spent on finding retiming, instead of predicate complexity. Thus, we pick simpler predicates for our experiments.

6.2 Experimental Setup

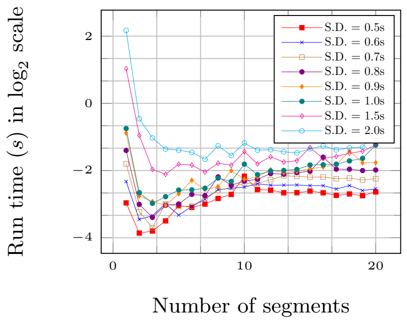
In our experiments, we choose the following parameters: (1) signal duration, (2) maximum clock skew ε, and (3) distribution of communication among agents. We measure the monitor run time. All experiments are replicated to exhibit %95 confidence interval to provide statistical significance. The experimental platform is a CentOS server with 112 Intel(R) Xeon(R) Platinum 8180 CPUs @ 3.80GHz CPU and 754G of RAM. Our implementation invokes the SMT-solver Z3 [10] to solve the problem described in Section 4. Color versions of all figures are available in the digital version of the paper.

6.3 Analysis of Results

Impact of signal segmentation Given a signal-to-be-monitored, we have a choice of either passing the entire signal to the monitor, or chopping it into segments and monitoring each segment separately (while accounting for ε-synchrony). Monitoring a signal in one shot is computationally more expensive than monitoring a number of shorter segments. Figure 5 shows the results of this claim. Note that all curves are plotted in $\log_2$ scale to provide more clarity. As can be seen, for any signal duration, chopping the signal and invoking the monitor for the shorter segments reduces the run time significantly. For example, in the case of the UAV network (Fig. 5b), for a signal duration of $2s$, it takes $4.5s$ to monitor the signal in one shot, but only $0.55s$ if the monitor is invoked 20 times over the signal duration. We observe the same behavior in Fig. 5a. This is due to the SMT-solver having to deal with much smaller search spaces in each invocation.



(a) Network of cars.



(b) Network of UAVs.

Fig. 5: Impact of signal segmentation on run time with varying signal duration (S.D.) and fixed $\varepsilon = 0.001s$.

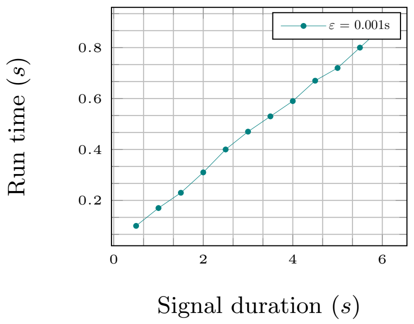


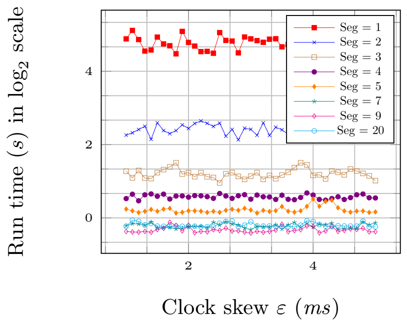
Fig. 6: Best run time (network of cars) for different signal duration.

Figure 6 shows the best achievable run time for different signal durations by searching over the segment count of range $[1, 25]$. For example, segment count of 4 is obtained for $1s$ signal to get minimum run time of $0.17s$, while segment count of 18 is obtained for $5s$ signal to get minimum run time of $0.72s$. The best run time shown is achieved by distributing the monitoring tasks across all the available cores (4) on the monitoring device. Notice that our predicate detection algorithm can be parallelized trivially, assigning one or a pool of segments to a different core.

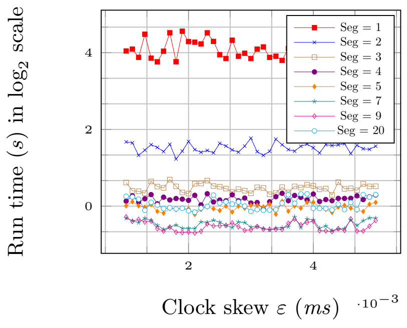
An important consequence of segmentation is that it enables us to monitor signals in real time, as for 3 or more segments, the run time of the monitor is less than the signal duration. For this reason, in all remaining experiments, the signal-to-monitor is chopped into 20 segments and each segment is monitored separately. Cumulative run times (of monitoring all 20 segments) are reported.

Impact of clock skew We now study the impact of different choices of ε on monitoring run time. We choose realistic values for ε with millisecond resolution. Figure 7 shows the monitoring run time for a $2s$ signal chopped into $1 - 20$ segments. Both Figs. 7a and 7b show that high resolution clock synchronization results in very stable execution time for the monitor. This is a positive result, showing that for practical clock synchronization algorithms, the actual value of ε does not have an impact on the monitoring overhead.

Impact of number of agents Now we observe the impact of the number of UAVs on the monitor. Fig. 8a shows the effect on run time for increasing the number of agents from 2 to 10 with $\varepsilon = 1ms$ over $5s$-long signals. As each segment of a signal

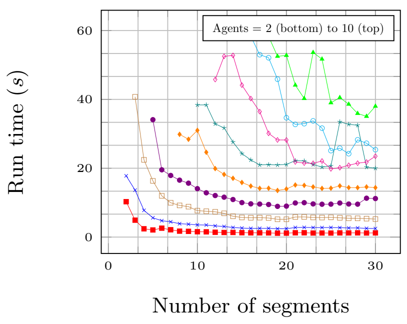


(a) Network of cars.

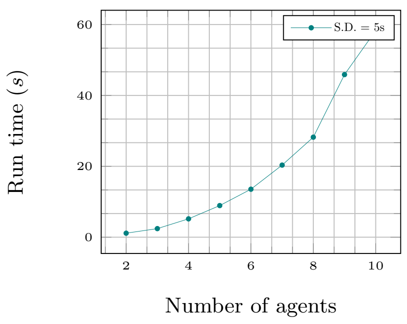


(b) Network of UAVs.

Fig. 7: Impact of clock skew on run time. Signal duration = $2s$.



(a) Signal Duration = 5s and ε = 0.001s



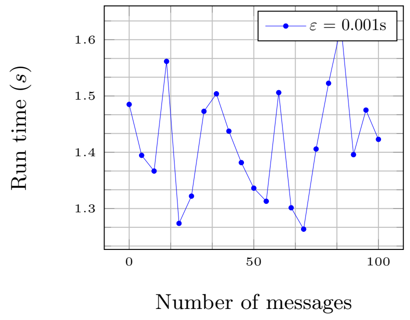
(b) Signal Duration = 5s and ε = 0.001s

Fig. 8: Impact of agents on run time.

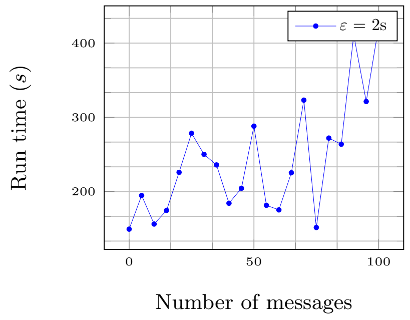
can be monitored independently, we improve our run time by distributing the monitoring tasks across all available cores on the monitoring device. Observe that initially the run time drastically improves as more segments are used. However, eventually the improvement becomes negligible, due to run time being dominated by non-SMT tasks, such as creating job queues, allocating jobs to cores, and so on. We refer to this run time as the *best run time*. Fig. 8b shows the best run times for different number of agents with $\varepsilon = 1ms$ over $5s$-long signals.

Impact of communication We examine whether the number of messages exchanged between agents has a significant impact on monitor run time. Two opposing mechanisms exist: on the one hand, messages impose an order between the send and receive moments and so reduce concurrency. In the discrete-time setting this normally reduces the asynchronous monitoring complexity. On the other hand, messages result in extra constraints in the SMT encoding via Eq. 7, which could increase SMT run time.

Fig. 9 shows the results. In (a) we use $\varepsilon = 1ms$ and a $1s$-long signal. Run time varies with no clear trend, suggesting that neither of the above two opposing mechanisms dominates. In (b), we use $\varepsilon = 2s$ for a $2s$-long signal: i.e., *all*

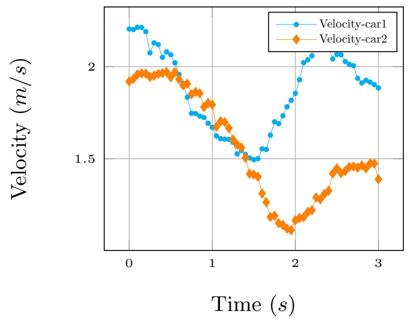


(a) Signal Duration = 1s and ε = 1s.

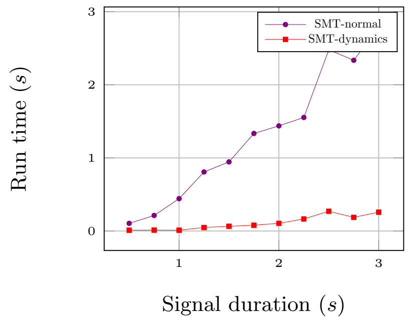


(b) Signal Duration = 2s and ε = 2s.

Fig. 9: Impact of communication (between two agents) on run time.



(a) Velocity profile of two cars.



(b) Run time vs. signal duration.

Fig. 10: Impact of Algorithm 1 on monitoring run time. $\varepsilon = 0.001s$.

events are concurrent. One can see the order introduced by messages are slightly increasing the runtime, instead of decreasing it. No conclusion can be drawn, and future work should study this more closely.

Impact of knowledge of dynamics bounds Here the predicate of interest is $\phi = (v_1 > 1.6) \vee (v_2 > 1.3)$, where v_i is the velocity of the i^{th} car. The acceleration limit from system dynamics is $1m/s^2$. The monitor samples the received signals (Fig. 10b) at $0.25s$ intervals and applies the acceleration bounds as explained in Section 5 to discard irrelevant pieces of the signal. As shown in Fig. 10, applying Algorithm 1 clearly reduces the monitor run time. In general, of course, the exact run time reduction varies. For instance, while the speedup is $\times 10$ for $3s$-long signals $3s$, it is $\times 15$ for $2s$-long signals.

7 Related Work

Runtime Monitoring of CPS Accurate time-keeping for CPS was investigated in the Roseline project [1]. Assuming perfect synchrony, [4] introduces a tool for offline monitoring of robust MTL semantics, and [12] performs online monitoring of STL [13]. The work [11] requires the full dynamical model of the system for predictive monitoring. Our work is closer to [19], which assumes worst-case a priori bounds on signal values (but without factoring dynamics). The works [29] and [2] study of how the satisfaction of a temporal property is affected by timing inaccuracies. Minimally intrusive CPS monitoring was studied in [22].

Decentralized Monitoring Lattice-theoretic centralized and decentralized online predicate detection in distributed systems has been studied in [7,24]. Extensions of this work to include temporal operators appear in [26,25]. In [30], the authors design a method for monitoring safety properties in distributed systems using the past-time linear temporal logic. This approach, however, suffers from producing false negatives. Runtime monitoring of LTL formulas for synchronous distributed systems has been studied in [9,8,5]. Finally, fault-tolerant monitoring has been investigated in [6] for asynchronous and in [20] for synchronous networks.

Partially Synchronous Monitoring The feasibility of monitoring partially synchronous distributed systems to detect latent bugs was first studied in [31]. This approach was generalized to the full LTL in [17]. The authors achieve this in a discrete-time/value setting by detecting the presence of latent bugs using SMT solvers. In [28], the authors propose a tool for identifying data races in distributed system traces. This approach is able to handle non-deterministic discrete event orderings. However, these approaches cannot not fully capture the continuous-time and continuous-valued behavior of CPS.

8 Conclusion

In this paper, we demonstrated a technique for online predicate detection for distributed signals that do not share a global clock. Our approach is based on causality analysis between real-valued signals, and integrates a realistic assumption on maximum clock skew among the local clocks, and rough knowledge of system dynamics, to make the problem tractable. We made several important observations by experimenting over a real network of autonomous vehicles and a simulated network of UAVs. Our approach can be effectively implemented to monitor a distributed CPS in an online fashion.

As for future work, there are many interesting research avenues. Our approach finds the first global states that violate a predicate. A crucial step in debugging distributed CPS is to find all such states. Thus, it is important to investigate data structures that can efficiently represent a set of global states of distributed continuous signals that violate a predicate. In the discrete setting, computation *slices* [24] are an example of such a data structure. One way to achieve this is by using the long-known notion of regions in timed automata [3]. Another future problem is to monitor distributed signals with respect to Signal Temporal Logic (STL) specifications.

References

1. https://sites.google.com/site/roselineproject/.
2. Houssam Abbas, Hand Mittelmann, and Georgios Fainekos. Formal property verification in a conformance testing framework. In *ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, October 2014.
3. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
4. Yashwanth Singh Rahul Annapureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.
5. A. Bauer and Y. Falcone. Decentralised LTL monitoring. *Formal Methods in System Design*, 48(1-2):46–93, 2016.
6. B. Bonakdarpour, P. Fraigniaud, S. Rajsbaum, D. A. Rosenblueth, and C. Travers. Decentralized asynchronous crash-resilient runtime verification. In *Proceedings of the 27th International Conference on Concurrency Theory (CONCUR)*, pages 16:1–16:15, 2016.
7. H. Chauhan, V. K. Garg, A. Natarajan, and N. Mittal. A distributed abstraction algorithm for online predicate detection. In *Proceedings of the 32nd IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 101–110, 2013.
8. C. Colombo and Y. Falcone. Organising LTL monitors over distributed systems with a global clock. *Formal Methods in System Design*, 49(1-2):109–158, 2016.
9. L. M. Danielsson and C. Sánchez. Decentralized stream runtime verification. In *Proceedings of the 19th International Conference on Runtime Verification (RV)*, pages 185–201, 2019.
10. L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 337–340, 2008.
11. A. Dokhanchi, B. Hoxha, and G. Fainekos. Online monitoring for temporal logic robustness. In *Proc. of Runtime Verification*, 2014.
12. A. Donzé, T. Ferrère, and O. Maler. Efficient robust monitoring for STL. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV)*, pages 264–279, 2013.
13. A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *Proceedings of the 8th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 92–106, 2010.
14. Drone Life. FAA UTM project: Decentralized uas traffic management demonstration, september 2019. https://dronelife.com/2019/09/09/decentralized-uas-traffic-management-demonstration.
15. FAA. DOT UAS initiatives, April 2019. https://www.faa.gov/uas/programs_partnerships/DOT_initiatives.
16. G. E. Fainekos and G. J. Pappas. Robust sampling for MITL specifications. In *Proceedings of 5th International Conference on the Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 147–162, 2007.
17. R. Ganguly, A. Momtaz, and B. Bonakdarpour. Distributed runtime verification under partial asynchrony. In *Proceedings of the 24nd International Conference on Principles of Distributed Systems (OPODIS)*, pages 20:1–20:17, 2020.

18. Meghan Hendry-Brogan. Global unmanned aerial vehicle (uav) market report. Technical report, May 2019.
19. V. Deshmukh J, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia. Robust online monitoring of signal temporal logic. *Formal Methods System Design*, 51(1):5–30, 2017.
20. S. Kazemloo and B. Bonakdarpour. Crash-resilient decentralized synchronous runtime verification. In *Proceedings of the 37th Symposium on Reliable Distributed Systems (SRDS)*, pages 207–212, 2018.
21. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.
22. R. Medhat, B. Bonakdarpour, D. Kumar, and S. Fischmeister. Runtime monitoring of cyber-physical systems under timing and memory constraints. *ACM Transactions of Embedded Computing Systems*, 14(4):79:1–79:29, 2015.
23. D. Mills. Network time protocol version 4: Protocol and algorithms specification. RFC 5905, RFC Editor, June 2010.
24. N. Mittal and V. K. Garg. Techniques and applications of computation slicing. *Distributed Computing*, 17(3):251–277, 2005.
25. M. Mostafa and B. Bonakdarpour. Decentralized runtime verification of LTL specifications in distributed systems. In *Proceedings of the 29th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 494–503, 2015.
26. V. A. Ogale and V. K. Garg. Detecting temporal logic predicates on distributed computations. In *Proceedings of the 21st International Symposium on Distributed Computing (DISC)*, pages 420–434, 2007.
27. Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. Smooth operator: Control using the smooth robustness of temporal logic. In *IEEE Conference on Control Technology and Applications,*, 2017.
28. Pereira, João Carlos, Nuno Machado, and Jorge Sousa Pinto. Testing for race conditions in distributed systems via smt solving. In *International Conference on Tests and Proofs, Bergen, Norway, June 22-26, 2020, Proceedings*, pages 122–140, 2020.
29. Jan-David Quesel. *Similarity, Logic, and Games: Bridging Modeling Layers of Hybrid Systems.* PhD thesis, Carl Von Ossietzky Universitat Oldenburg, July 2013.
30. K. Sen, A. Vardhan, G. Agha, and G.Rosu. Efficient decentralized monitoring of safety in distributed systems. In *ICSE*, 2004.
31. Vidhya Tekken Valapil, Sorrachai Yingchareonthawornchai, Sandeep S. Kulkarni, Eric Torng, and Murat Demirbas. Monitoring partially synchronous distributed systems using SMT solvers. In *Runtime Verification - 17th International Conference, RV 2017, Seattle, WA, USA, September 13-16, 2017, Proceedings*, pages 277–293, 2017.