When Every Transmission Counts: Event-Trigger Threshold Regulation for STL Properties

***Abstract*—We propose a novel event-trigger threshold (ETT) regulation mechanism ETT^ρ based on Signal Temporal Logic (STL) properties significantly extending recent work on ETT regulation for Propositional Logic properties. We utilize the quantitative semantics of STL to construct a method for computing and merging suitable ETTs for different requirements in complex STL specifications. Contrary to related work on event-triggered control for STL properties, we apply the event-triggering logic to the measured signals individually rather than the control output and assume that an existing periodic controller is defined. By exploiting the early satisfaction detection capabilities of STL and analyzing the property structure, our method aims to reduce the number of triggered events, while maintaining satisfaction of system properties. To evaluate ETT^ρ, we consider a simulated adaptive cruise control case-study where STL is used to encode complex safety and performance properties and the ETTs of measured signals are regulated accordingly. We test three different properties in two different scenarios to showcase how STL and ETT^ρ can identify intricate circumstances where it is possible to significantly reduce the number of triggered events relative to a constant ETT.**

***Index Terms*—Event-triggering mechanisms, Signal Temporal Logic, Cyber-physical systems**

I. Introduction

Matching communication frequency to the requirements of cyber-physical systems (CPS) has attracted increasing interest with the advance of networked control systems (NCS) [1]. Matching communication frequency entails not only ensuring that communication is sufficiently frequent to enable the CPS to satisfy its requirements, but also to identify and act on circumstances where a reduced communication frequency is possible. In bandwidth-limited and shared medium communication systems such as wireless systems, the available resources are shared among system components. Thus, reducing communication for safe/well performing (sub)systems, can allow other presently more unsafe/less performant systems to communicate more frequently and thus improve the performance and safety bottom line.

Event-triggering mechanisms: A popular approach to reduce communication is to introduce an event-triggering mechanism which can trigger either control, sampling, measurement transmission [2] or a combination of these. Introducing the event-triggering condition allows the system to act on new information when necessary (i.e. a triggered event) as opposed to periodical updates. The introduction of the event-triggering condition can significantly reduce communication frequency [3] as opposed to periodic sampling, transmission and/or control. We further detail relevant event-triggering mechanisms in Section II-B.

Specification-based monitoring: As an alternative to providing formal guarantees for performance and safety for a well-defined system (e.g. stability or convergence), specification languages have been developed to specify correct operation of systems based on system traces [4]. Such specification languages are typically not restricted to certain types or classes of systems and instead only require that the system behavior can be modelled as signals or discrete events. For many specification languages it is then possible to produce a monitor which determines whether the system satisfies the specification, either at runtime or in an offline setting [4]. STL is one such well-established formalism for specifying requirements in CPSs where it has seen a wide range of applications from smart grids [5] to safety-critical medical devices [6].

Contribution: We propose an algorithm that can generate a run-time ETT regulation mechanism tailored to the quantitative semantics or arbitrary STL properties. Our method accounts for the accuracy requirements of a controlled CPS, how well the requirements are satisfied at runtime, and introduces parameters to allow for fine-tuning to different systems. Informally, the ETT regulation mechanism is designed to regulate signal ETTs to ensure minimum performance requirements (i.e. property satisfaction) while reducing communication. We formalize this later in Problem 1. While our method uses some elements of the STL quantitative semantics for ETT computation, we stress that our method is auxiliary to STL, and uses STL semantics and system requirements to infer ETT regulation behavior based on the structure of the specific property.

Previous work: Recently, in [7], an ETT regulation mechanism for Propositional Logic (PL) was presented, where PL is a subset of STL excluding temporal operators. Naturally, the method proposed in this paper builds upon the work in [7], but the addition of temporal operators significantly increases the expressiveness compared to PL and requires careful consideration to define a suitable ETT regulation synthesis mechanism. Besides [7], the closest related works are [8], [9] which both consider the problem of designing an event-triggered prescribed performance controller to satisfy STL properties. The STL properties are however limited to fragment of STL, as opposed to our approach which defines an ETT regulation mechanism for arbitrary STL properties. Furthermore, our approach does not consider event-triggered control, but rather event-triggered state estimation with an existing periodic control, and does not provide formal guarantees for property satisfaction. Instead, we show how ETT regulation

Manuscript created February 2025; Revised September 2025 Valdemar Tang (valdemar.tang@ece.au.dk), Claudio Gomes and Daniel Lucani are with the Department of Electrical and Computer Engineering, Aarhus University, 8200 Aarhus, Denmark.

parameters can be determined empirically.

TABLE I: Overview of common notation.

Notation	Description
$\varphi, \varphi^p, \boldsymbol{\Phi}$	Arbitrary, inequality and set of STL properties respectively.
$\mathbf{x}, \hat{\mathbf{x}}, \hat{\mathbf{x}}_k, \hat{\mathbf{x}}_{[k,k+a]}$	Full true and estimated system state trace and estimated system state trace at the k-th sampling/update instant (time t_k) and in the interval $[t_k, t_k + a]$, respectively where $a \geq 0$. We note that $\hat{\mathbf{x}}$ is a sequence of discrete time state estimates: $\hat{\mathbf{x}} = (\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_{\text{end}})$.
$y_i, \mathbf{y}$	Measurable system output i and set of all measurable system outputs.
$\zeta^*_\rho(\varphi, \hat{\mathbf{x}}_k)$	Normalized robustness for arbitrary STL property.
β_k	Parameter calculated based on $\zeta^*_\rho(\varphi, \hat{\mathbf{x}}_k)$.
$\delta^+_{y_i,k+1}(\varphi^p, \hat{\mathbf{x}}_k, \epsilon_{y_i,\varphi^p})$ $\delta^*_{y_i,k+1}(\varphi, \hat{\mathbf{x}}_k, \beta_k)$	ETT for signal y_i at the $(k+1)$-th sampling instant for an inequality and arbitrary STL property.
ϵ_{y_i,φ^p}	Scaling parameter for ETT regulation based on inequality property.
ETT^ρ, ETT^C, ETT^0	Proposed ETT regulation, constant ETT and time-triggered schemes.
$PT(\varphi)$	Propositional transformation of an arbitrary STL property.

II. Background

A. Signal Temporal Logic

Signal Temporal Logic [10] (STL) is a formal specification language used to express signal-based temporal logic properties. STL is defined by the following recursive syntax

$$\varphi ::= \varphi^p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 \tag{1}$$

where φ^p is of the form $p(\hat{\mathbf{x}}_k) \sim c$ where $\sim \in \{>, <\}$, $\hat{\mathbf{x}}_k$ is the estimated system state vector at time t_k, c is a constant, $\neg$ is the negation operator, $\wedge$ and $\vee$ are the logical *and* and *or* operators, and $\varphi_1 \mathcal{U}_{[a,b]} \varphi_2$ is the temporal *until* operator which verifies that φ_1 is satisfied at all times in the interval $[t_k + a, t_k + b]$, until φ_2 becomes satisfied in the same interval. As defined in Definition 1, the STL operators in Eq. (1) can be combined to form additional useful operators; the *eventually* operator ($\diamond_{[a,b]}\varphi \equiv \text{True}\ \mathcal{U}_{[a,b]}\varphi$); the *always* operator ($\square_{[a,b]}\varphi \equiv \neg \diamond_{[a,b]} \neg\varphi$) and the implication operator ($\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$).

STL properties can produce both qualitative and quantitative metrics, where the latter is also known as the robustness [11]. The robustness indicates the degree of satisfaction of a property at a given time. The qualitative metric is equivalent to determining whether the property is satisfied.

Definition 1 (Robustness from [11]). *The robustness $\rho(\varphi, \hat{\mathbf{x}}, t_k)$ of an STL property φ based on the system state trace $\hat{\mathbf{x}}$, at time t_k, is defined recursively by:*

$$\begin{aligned}
\rho(p(\hat{\mathbf{x}}) > c, t_k) &= p(\hat{\mathbf{x}}_k) - c \\
\rho(\neg\varphi, \hat{\mathbf{x}}, t_k) &= -\rho(\varphi, \hat{\mathbf{x}}, t_k) \\
\rho(\varphi_1 \wedge \varphi_2, \hat{\mathbf{x}}, t_k) &= \min(\rho(\varphi_1, \hat{\mathbf{x}}, t_k), \rho(\varphi_2, \hat{\mathbf{x}}, t_k)) \\
\rho(\varphi_1 \vee \varphi_2, \hat{\mathbf{x}}, t_k) &= \max(\rho(\varphi_1, \hat{\mathbf{x}}, t_k), \rho(\varphi_2, \hat{\mathbf{x}}, t_k)) \\
\rho(\diamond_{[a,b]}\varphi, \hat{\mathbf{x}}, t_k) &= \max_{t_i \in [t_k+a, t_k+b]} \rho(\varphi, \hat{\mathbf{x}}, t_i) \\
\rho(\square_{[a,b]}\varphi, \hat{\mathbf{x}}, t_k) &= \min_{t_i \in [t_k+a, t_k+b]} \rho(\varphi, \hat{\mathbf{x}}, t_i) \\
\rho(\varphi_1 \mathcal{U}_{[a,b]} \varphi_2, \hat{\mathbf{x}}, t_k) &= \\
\max_{t_i \in [t_k+a, t_k+b]} (\min(\rho(\varphi_2, \hat{\mathbf{x}}, t_i), &\min_{t_j \in [t_k+a, t_i]} \rho(\varphi_2, \hat{\mathbf{x}}, t_j)))
\end{aligned}$$

1) Runtime STL monitoring: According to Definition 1, the robustness of temporal operators at a time t_k depends on states at a future time, motivating runtime STL semantics. A notable insight, which we exploit to reduce the number of triggered events, is that a signal prefix may be sufficient to determine the satisfaction of a property before the entire interval has elapsed. The Robust Satisfaction Interval (RoSI) [12], defined in Definition 2, provides an upper and lower bound of the robustness for an STL property which evolves as time progresses.

Definition 2 (Robust Satisfaction Interval (RoSI) [12]). *A robust satisfaction interval $RoSI(\varphi, \hat{\mathbf{x}}_{[0,i]}, t_k)$ of an STL property φ evaluated at time t_k on a set of signals $\hat{\mathbf{x}}_{[0,i]}$ available in the temporal interval $[t_0, t_i]$, is an interval I which satisfies:*

$$\inf(I) = \inf_{\hat{\mathbf{x}} \in \mathcal{C}(\hat{\mathbf{x}}_{[0,i]})} \rho(\varphi, \hat{\mathbf{x}}, t_k) \tag{2}$$

$$\sup(I) = \sup_{\hat{\mathbf{x}} \in \mathcal{C}(\hat{\mathbf{x}}_{[0,i]})} \rho(\varphi, \hat{\mathbf{x}}, t_k) \tag{3}$$

where $\mathcal{C}(\hat{\mathbf{x}}_{[0,i]})$ is all the possible (future) sequences of $\hat{\mathbf{x}}_{[0,\text{end}]}$ that have $\hat{\mathbf{x}}_{[0,i]}$ as a prefix where $t_{\text{end}} \geq t_i$.

A positive lower bound before the interval has elapsed is equivalent to detecting early property satisfaction. We note that early satisfaction detection (ESD) is only possible for the $\diamond_{[a,b]}$ and $\mathcal{U}_{[a,b]}$ operators. An efficient algorithm for runtime computation of the RoSI and additional details can be found in [12]. We utilize RoSI for ETT regulation in Section IV-A and refer to Example 2 for an example on ESD.

For properties with nested temporal operators, the satisfaction of the property may somewhat counter-intuitively depend on signal values beyond the duration of the outermost temporal operator. For example, consider the property $\varphi_1 \rightarrow \diamond_{[0,10]}\square_{[0,1]}\varphi_2$. One might expect the satisfaction of the property to depend on the satisfaction of φ_2 in the interval $[0, 10]$. However, it actually depends on the interval $[0, 10+1]$ as the inner $\square_{[0,1]}$ may be satisfied at time 10 of the outer interval, meaning that if φ_2 is satisfied in the interval $[10, 11]$, the overall property is satisfied. The extent of this dependence can be formally defined as the *temporal depth* in Definition 3.

Definition 3 (Temporal depth ([13])). *The temporal depth is the maximum future duration relevant to compute the* *satisfaction (and/or robustness) of a property at the current time step and is computed inductively as:*

$$\begin{aligned}
H(\varphi^p) &= 0\\
H(\neg\varphi) &= H(\varphi)\\
H(\varphi_1 \wedge \varphi_2) &= \max(H(\varphi_1), H(\varphi_2))\\
H(\varphi_1 \vee \varphi_2) &= \max(H(\varphi_1), H(\varphi_2))\\
H(\Box_{[a,b]}\varphi) &= b + H(\varphi)\\
H(\diamond_{[a,b]}\varphi) &= b + H(\varphi)\\
H(\varphi_1 \mathcal{U}_{[a,b]}\varphi_2) &= b + \max(H(\varphi_1), H(\varphi_2))
\end{aligned}$$

B. Event-triggering mechanisms

As detailed later in Section III-A, we consider a discrete time sampled-data event-triggered transmission system [14] with periodic control. For sensors, this entails that measured signals are sampled periodically with an interval T_s but only transmitted once an event-triggering condition is satisfied. We provide an overview of the system architecture in Fig. 1. An event triggering mechanism is characterized by the event-triggering condition, which can be stated generally as

$$e(y_i(t_k), ...) > \delta_{y_i,k} \tag{4}$$

where $y_i(t_k)$ is related to the system state by $y_i(t_k) = q_i(\mathbf{x}(t_k)) + w_i(t_k)$, $w_i(t_k)$ is noise and $\delta_{y_i,k}$ is the ETT for signal y_i at time t_k. The function $e(\cdot)$ implements the update error where two popular options are 1) the Send-On-Delta (SOD) update error $e(\cdot) = |y_i(t_k) - y_{i,\tau^i_{j-1}}|$ [15] where $y_{i,\tau^i_{j-1}}$ refers to the value of signal y_i at the most recent event-triggering instant, and 2) the innovation [3] update error:

$$e(\cdot) = |\hat{y}_{i,k} - y_i(t_k)| \tag{5}$$

where $\hat{y}_{i,k}$ is the predicted value of $y_i(t_k)$. This prediction can be obtained through extrapolation [14]. The value of $\delta_{y_i,k}$ is scaled to ensure that the events are triggered under the required conditions. For example, the ETTs shall be small in situations where high accuracy is needed and may be increased in non-critical situations to communication resources. Thus, the ETT typically depends on the system state and goal. As a result, various ETT regulation mechanisms have been proposed. Examples include time-dependent ETTs for Multi-Agent-Systems (MASs) [16], ETTs which adapt to the state of the network [17], and ETTs taking Denial-of-Service attacks into account [18].

ETT regulation for Propositional Logic properties: We devote the remainder of this Section to introduce the method proposed in [7].

Definition 4 (ETT regulation for inequality properties [7, Eq. 29]). *Let $\mathbf{y}_{\sim\varphi^p}$ denote the set of measured signals to which ETT regulation is applied based on the robustness of φ^p. An ETT regulation mechanism for an inequality property φ^p and measurable signal $y_i \in \mathbf{y}_{\sim\varphi^p}$, is given by:*

$$\delta^+_{y_i,k+1}(\varphi^p, \hat{\mathbf{x}}_k, \epsilon_{y_i,\varphi^p}) = \max(\frac{\rho(\varphi^p, \hat{\mathbf{x}}, t_k)}{\epsilon_{y_i,\varphi^p}}, 0) \tag{6}$$

where $\epsilon_{y_i,\varphi^p} \in \mathbb{R}_{>0}$ and $\rho(\varphi^p, \hat{\mathbf{x}}, t_k)$ is the robustness from Definition 1. The set $\mathbf{y}_{\sim\varphi^p}$ is a design parameter.

We note that in Definition 4, the ETT at the next time step t_{k+1}, is based on the state at the current time step t_k. As shown in [7], along with certain ϵ_{y_i,φ^p} parameters, this may result in the system incorrectly believing that a property is satisfied when it is not. To guarantee correct satisfaction detection, [7] provides a modified version of the ETT regulation algorithm. We refer to [7] for more information.

When we consider propositional properties, multiple inequality properties may define an ETT for the same signal(s). This is handled by applying the minimum defined ETT for each signal, which in [7, Definition 4] is assumed to still guarantee the satisfaction of all affected properties. We also make this assumption and refer to [7] for details.

To refine the ETTs based on the propositional operators, the authors in [7] define a metric called the normalized robustness $\zeta_\rho(\varphi, \hat{\mathbf{x}}_k)$. To avoid repetition, we note that the normalized robustness for a PL property is identical to our extension (Definition 7), without the temporal and *otherwise* cases. Similarly, the ETT regulation mechanism for PL properties from [7] can be obtained by our extension in Definition 9 by omitting the temporal property cases. The key feature of the PL ETT regulation mechanism, is that when $\vee$ operators are present, the ETT regulation mechanism only has to provide sufficient accuracy to enable the system to satisfy at least one of the sub-properties at any time. This relaxation is then used to enlarge the ETTs related to the sub-property with a lower normalized robustness, thus saving communication resources.

III. Assumed system architecture and problem formulation

A. The assumed event-triggered system architecture

We study the generic system architecture in Fig. 1 with multiple smart sensors (i.e. a sensor with built-in but limited computing capacity) measuring various outputs of a process.

The system has a set of properties/requirements ($\boldsymbol{\Phi}$) from which our proposed method synthesizes a runtime ETT regulation mechanism.

Clarifying assumptions: We assume that the smart sensor sampling, remote state estimator and control system update frequency are periodic and synchronized with interval T_s. The synchronization of control update frequency is enabled by the remote state estimator, as it produces state estimates at every time step regardless of whether measurements have been transmitted [14]. Additionally, we assume that ETTs and measurement predictions are instantly available at the smart sensors without need for communication and that measurements which trigger events are instantly available at the remote state estimator.

In real-world applications, perfect synchronization and instant packet reception are not possible, but accurate synchronization protocols are available [19] and significant research has been conducted to minimize communication delays for wireless networks [20]. These assumptions therefore do not generally harm the usefulness of our proposed method. Like the event-triggering condition, random communication delays introduce additional uncertainty into the system, which in turn could be mitigated with smaller ETTs. Similarly, precise and

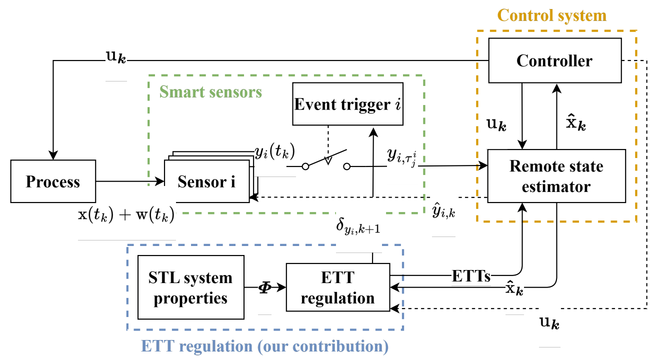

Fig. 1: The sampled-data event-triggered transmission architecture considered in this paper. Dotted black lines indicate an optional data flow, where the necessity of communicating this information depends on the monitored requirements and the event-triggering condition. Adapted from [7].

frequent ETT updates could be replaced with less precise or frequent but generally smaller ETT updates. Both these uncertainties are expected to result in an additional communication overhead in real-world systems. Overcoming the impact of real-word restrictions will be a core part of our future work.

B. Problem formulation

We start by formalizing the requirements of an ETT regulation mechanism in Definition 5, state a key assumption in Assumption 1 and present the formal problem formulation in Problem 1.

Definition 5. *An ETT regulation mechanism $ETT(\cdot)$, is a function which returns a set $\{\delta_{y_i,k} \geq 0\}_{y_i \in \mathbf{y}}$ for any time $t_k \in [t_0, t_{end}]$.*

The ETTs from $ETT(\cdot)$, are then applied in Eq. 4 at every time step to obtain the event-triggered transmission strategy. With an slight abuse of notation and for brevity, from now on we refer to an ETT regulation mechanism $ETT(\cdot)$ as ETT when parameters are not relevant to the context.

Assumption 1 (Satisfaction feasibility [7])**.** *We assume that the system under the time-triggered transmission approach, ensures property satisfaction, i.e.*

$$\forall \varphi \in \boldsymbol{\Phi}, \forall t_k \in [t_0, t_{\text{end}}] \;\; \rho(\varphi, \hat{\mathbf{x}}, t_k) > 0, \tag{7}$$

for some adequate sampling interval T_s. The time-triggered transmission approach is equivalent to the zero-ETT scheme, i.e. $ETT^0(\cdot) = \{0\}_{y_i \in \mathbf{y}} \forall t_k \in [t_0, t_{\text{end}}]$.

The sufficient T_s value in Assumption 1 is then used as the sampling and controller update frequency for any subsequent event-triggering scheme. We now present the main problem in Problem 1.

Problem 1. *Find an ETT regulation mechanism $ETT^*(\cdot)$ s.t. Eq. 7 and*

$$E_{\#}[ETT^*(\cdot)] \leq E_{\#}[ETT^0(\cdot)] \tag{8}$$

holds, where $E_{\#}[ETT(\cdot)]$ denotes the expected number of triggered events across all signals for $ETT(\cdot)$.

Combining properties: Instead of considering multiple disjoint properties, all properties may be combined using the $\wedge$ operator. As shown in [7], ETT regulation is unaffected by this operation and will remain so for the proposed method. To simplify notation, we assume that all properties have been combined into a single property.

C. Constant ETT policy

To provide a more comparable method for our proposed ETT regulation mechanism, later in Section V we compare our proposed mechanism with a constant ETT scheme (ETT^C). The ETT^C scheme entails finding a set of constant ETTs $\{\delta_{y_i} \geq 0\}_{y_i \in \mathbf{y}}$, which when applied to the signals in $\mathbf{y}$, enables the system to satisfy the constraint in Eq. (7).

Next, we present our parameterized ETT regulation mechanism $\boldsymbol{ETT^\rho}$ based on the quantitative semantics of STL to provide a candidate event-triggering mechanism for Problem 1.

D. STL vs other logic languages

In this Section, we briefly detail why STL is a suitable choice for ETT regulation compared to other logics. Many families of logics exist for property specification [21] and an exhaustive description is out of scope for this paper. Instead, we briefly account for the desirable characteristics of STL and compare with other languages. STL is developed specifically for time-series data and is widely used to specify properties in CPSs. The strict adherence to real-valued signals ensure well-defined quantitative semantics which we can base the ETT regulation mechanism on. This is in contrast to other temporal logics which only produce a binary verdict such as Linear Temporal Logic (LTL) [22]. Furthermore, STL has a quantitative view of time, i.e. data points are associated with a timestamp and real-time constraints are an integrated part of the language, whereas LTL has a qualitative view of time concerned with the ordering of events and consequently does not allow specifying real-time constraints. Some stream-based languages like TeSSLa [23] allow quantitative semantics and have a quantitative view of time, but the quantitative semantics must be defined by the user. Additionally, STL has a future-time perspective, i.e. the satisfaction metric related to the current time depends on signals in the future. Stream-based runtime verification languages instead map an input stream to an output stream and natively have a past-time view. Although the past-time view eases the process of producing an executable monitor, it makes it harder to define ETT regulation behavior, as ETT regulation must ensure satisfaction of the property in the future and naturally cannot affect the satisfaction once the verdict has already been determined. With that being said, a myriad of extensions of LTL, STL and other logics exist [21] extending existing languages with additional features such as quantitative semantics and real-time constraints, spatial operators etc. However, we settle on STL as it possesses the above-mentioned characteristics out of the box and is widely used in CPSs [4]. Additionally, we emphasize that even though STL is a specification language, an STL property in combination with ETT^ρ could also be used in scenarios where there is no notion of correct behavior, and instead to control the quantization level (and consequently communication rate) using a logic-based approach. This makes ETT^{ρ} potentially applicable to a wider range of IoT use-cases.

IV. ETT REGULATION FOR STL PROPERTIES

Similarly to [7], we take a constructive approach to solve Problem 1 where the ETT regulation mechanism is synthesized from the structure of the STL property and the semantics of the STL operators. Initially in Sections IV-A and IV-B, we introduce two central considerations which determine our ETT regulation synthesis mechanism; *duration of ETT regulation for temporal properties* and *propositional-temporal property interaction* respectively. We then present our proposed method in Section IV-C.

A. ETT regulation duration for temporal properties

According to Definition 1, the robustness of temporal STL properties (e.g. $\Diamond_{[a,b]}\varphi^p$) at a time t_k depends on the values of the signals in the interval $[t_k + a, t_k + b]$ which are unknown at time t_k. Thus, using the robustness of the temporal property to regulate the ETTs of the underlying properties is infeasible. Instead, we choose to base the ETT regulation on the robustness of the underlying inequality and propositional properties which can be computed at every timestep.

Since temporal properties may refer to tight temporal intervals in the future, the ETT regulation mechanism must provide sufficient accuracy for the controller to satisfy the requirement in the future. We demonstrate this necessity in Example 1.

Example 1. *Consider the property $\varphi = (y > 10) \to (\Diamond_{[10,12]}\Box_{[0,1]}(|y| \leq 1))$ which can be thought of as a time-bounded stabilization of the signal y. Only applying ETT regulation in the interval $[t_k + 10, t_k + 12]$ may not give the controller sufficient time to stabilize y within the required time and satisfy the property.*

To avoid this issue, if we detect that the satisfaction of the overall property depends on a temporal property at time t_k, we conservatively apply ETT regulation to the temporal property's sub-properties starting at time t_k rather than at $t_k + a$.

To determine the end of the ETT regulation interval, we detect the time at which the temporal property is known to be satisfied. As mentioned in Section II-A1, a positive lower bound of the RoSI indicates (a potentially early) satisfaction of the property. To explicitly indicate the temporal interval on which the current ETT regulation decision depends, we conveniently let $\underline{RoSI}(\varphi, \hat{\mathbf{x}}_{[k+a,k+i]}, t_k)$ denote the lower bound of the RoSI for time t_k, based on the information available in the interval $[t_k + a, t_k + i]$ s.t. $a \leq i \leq H(\varphi)$ where $H(\varphi)$ is the temporal depth of φ, defined in Definition 3. In Definition 6, we use $\underline{RoSI}(\cdot)$ to define the time instant at which a temporal property is satisfied and later use this to define the end of the ETT regulation interval for temporal properties.

Definition 6 (Satisfaction time)**.** *We define the satisfaction time $S(\varphi, t_k)$ for a temporal STL property φ with the associated temporal interval $[a, b]$ based on evaluating $\underline{RoSI}(\cdot)$ for the property φ at time t_k:*

$$S(\varphi, t_k) = \begin{cases} \begin{cases} \min\limits_{t_i \in [t_k+a, t_k+H(\varphi)]} \{t_i : \underline{RoSI}(\varphi, \hat{\mathbf{x}}_{[k+a,k+i]}, t_k) > 0\} \\ \quad \textit{if } \exists t_i \in [t_k + a, t_k + H(\varphi)] : \\ \quad\quad \underline{RoSI}(\varphi, \hat{\mathbf{x}}_{[k+a,k+i]}, t_k) > 0, \\ t_k + H(\varphi) \qquad \textit{otherwise}, \end{cases} \\ \qquad \textit{if } \varphi = \Diamond_{[a,b]}\varphi_1 \textit{ or } \varphi = \varphi_1 \mathcal{U}_{[a,b]}\varphi_2, \\ t_k + H(\varphi) \quad \textit{if } \varphi = \Box_{[a,b]}\varphi_1, \end{cases}$$

We note that the approach of applying ETT regulation to the interval $[t_k, S(\varphi, t_k)]$ is similar to the previously mentioned event-triggered control methods which consider a subset of STL [8], [9]. In Example 2, we show an example of early satisfaction detection and the corresponding desired ETT regulation behavior.

Example 2 (Early satisfaction detection (ESD) and ETT regulation)**.** *Consider a modified version of the property from Example 1: $\varphi = (y > 10) \to \Diamond_{[0,10]}\Box_{[0,1]}(|y| \leq 1)$. If at some time t_k, $y_k > 10$, then within 10 seconds the system must stabilize the signal y to reside in the interval $[-1, 1]$ for a consecutive period of 1 second. If the system stabilizes earlier (i.e. $|y| \leq 1$ for a consecutive time interval $[t_j, t_j + 1]$ where $t_j < 10$ thus satisfying the inner $\Box_{[0,1]}$ property), the overall property is satisfied before the last possible satisfaction time $(t_k + 11)$ meaning that from then on, we can safely disregard ETT regulation to satisfy the temporal requirement. A figure illustrating this scenario is provided in Fig. 2. After we define the ETT regulation mechanism in Section IV-C, we will revisit this property in Example 3 to clarify the interaction between temporal and $\vee$ operators and how ETT decisions for nested temporal operators work.*

B. Property interaction for temporal STL operators

To enable property interaction between temporal and propositional properties, we extend the concept of normalized robustness from [7, Definition 7] to include temporal STL properties in Definition 7. Thereafter, in Lemma 1, we prove the soundness of the extension in Definition 7.

Definition 7. *We extend the normalized robustness $\tilde{\zeta}_\rho(\varphi, \hat{\mathbf{x}}_k)$ for a PL property to the normalized robustness $\zeta^*_\rho(\varphi, \hat{\mathbf{x}}_k)$ for an arbitrary STL property φ as follows:*

$$\zeta^*_\rho(\varphi, \hat{\mathbf{x}}_k) = \begin{cases} \min\limits_{\varphi_s \in \{\varphi_1,\varphi_2\}} (\zeta^*_\rho(\varphi_s, \hat{\mathbf{x}}_k)) & \textit{if } \varphi = \varphi_1 \mathcal{U}_{[a,b]}\varphi_2 \text{ and } a = 0, \\ \min\limits_{\varphi_s \in \{\varphi_1,\varphi_2\}} (\zeta^*_\rho(\varphi_s, \hat{\mathbf{x}}_k)) & \textit{if } \varphi = \varphi_1 \wedge \varphi_2, \\ \max\limits_{\varphi_s \in \{\varphi_1,\varphi_2\}} (\zeta^*_\rho(\varphi_s, \hat{\mathbf{x}}_k)) & \textit{if } \varphi = \varphi_1 \vee \varphi_2, \\ \zeta^*_\rho(\varphi_1, \hat{\mathbf{x}}_k) & \textit{if } \varphi = \Diamond_{[a,b]}\varphi_1 \text{ and } a = 0, \\ \zeta_\rho(\varphi, \hat{\mathbf{x}}_k) & \textit{if } \varphi = \varphi^p \\ 0 & \textit{if } \varphi = \Box_{[a,b]}\varphi_1, \\ 0 & \textit{otherwise}, \end{cases}$$

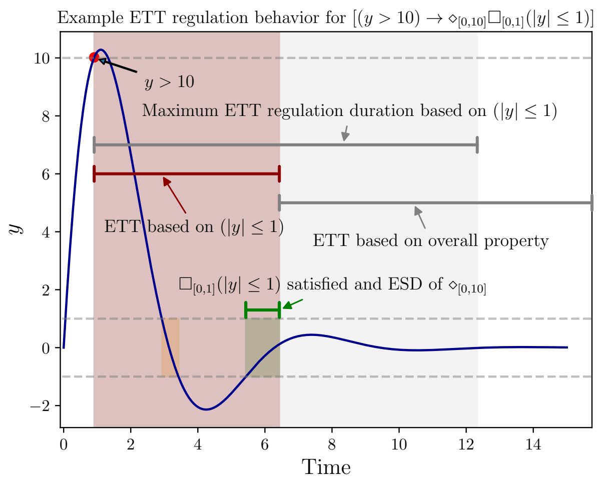

Fig. 2: An example of the desired ETT regulation behavior for the property $\varphi = (y > 10) \rightarrow \diamond_{[0,10]}\square_{[0,1]}(|y| \leq 1)$. While the property is yet to be satisfied (duration indicated by red area), the ETT of signal y will be regulated to ensure the eventual satisfaction of the property. The green box indicates the satisfaction of the $\square_{[0,1]}(|y| \leq 1)$ sub-property which leads to an ESD of the outer $\diamond_{[0,10]}$ operator. The yellow earlier box indicates the scenario where $|y| \leq 1$ is true, but for a shorter consecutive interval than the required duration and thus does not constitute and early satisfaction. After the property is satisfied (non-red area), the ETT for y will be regulated based on the overall property, as specified later in Definition 9.

*where the **otherwise** catches temporal operators where* $a > 0$.

Lemma 1. $(\zeta^*_\rho(\varphi, \hat{\mathbf{x}}_k) > 0) \Rightarrow (\rho(\varphi, \hat{\mathbf{x}}, t_k) > 0)$.

Proof. As the propositional cases were considered in [7], we only consider the temporal operator cases in Definition 7. We will prove Lemma 1 using induction.

Case: $\varphi = \varphi_1 \mathcal{U}_{[a,b]} \varphi_2$ and $a = 0$*:*
Assume $\min_{\varphi_s \in \{\varphi_1, \varphi_2\}} (\zeta^*_\rho(\varphi_s, \hat{\mathbf{x}}_k)) > 0$, which implies $\zeta^*_\rho(\varphi_1, \hat{\mathbf{x}}_k) > 0$ and $\zeta^*_\rho(\varphi_2, \hat{\mathbf{x}}_k) > 0$. By the inductive hypothesis; $\rho(\varphi_1, \hat{\mathbf{x}}, t_k) > 0$ and $\rho(\varphi_2, \hat{\mathbf{x}}, t_k) > 0$. Thus, if $a = 0$, then $\rho(\varphi_1 \mathcal{U}_{[a,b]} \varphi_2, \hat{\mathbf{x}}, t_k) > 0$ by Definition 1.

Case: $\varphi = \diamond_{[a,b]}\varphi_1$ and $a = 0$*:*
The proof for the $\diamond_{[a,b]}\varphi_1$ case can be obtained following a similar approach to the $\varphi_1 \mathcal{U}_{[a,b]} \varphi_2$ case.

Case: $\varphi = \square_{[a,b]}\varphi_1$ and *otherwise:* These cases are trivially true since $\zeta^*_\rho(\varphi, \hat{\mathbf{x}}_k)$ always equals 0.

Since each operation in Definition 7 preserves Lemma 1, Lemma 1 is preserved for an arbitrary STL property.

□

As proven in Lemma 1, the satisfaction of a property with temporal operators will not depend on future states if $\zeta^*_\rho(\varphi, \hat{\mathbf{x}}_k) > 0$. In this case, we propose to temporarily treat an STL property as a propositional property for ETT regulation purposes until the satisfaction of the overall property again depends on one or more temporal operators. The reasoning behind this choice will be explained in Example 4. To treat an arbitrary STL property as a propositional property, we propose the transformation in Definition 8.

Definition 8 (Propositional transformation)**.** *Let φ be an arbitrary STL property. We define the transformation $PT(\varphi)$ from an arbitrary STL property φ, to an STL property only containing propositional operators:*

$$PT(\varphi) = \begin{cases} \varphi & \text{if } \varphi = \varphi^p \\ PT(\varphi_1) \vee PT(\varphi_2) & \text{if } \varphi = \varphi_1 \vee \varphi_2 \\ PT(\varphi_1) \wedge PT(\varphi_2) & \text{if } \varphi = \varphi_1 \wedge \varphi_2 \\ PT(\varphi_1) & \text{if } \varphi = \diamond_{[a,b]}\varphi_1 \\ PT(\varphi_1) & \text{if } \varphi = \square_{[a,b]}\varphi_1 \\ PT(\varphi_1) \wedge PT(\varphi_2) & \text{if } \varphi = \varphi_1 \mathcal{U}_{[a,b]} \varphi_2 \end{cases}$$

C. *STL property ETT regulation (ETT^ρ - **main contribution**)*

In Definition 9, we combine the previous considerations, and introduce our proposed ETT regulation mechanism ***ETT$^\rho$***, for arbitrary STL properties. In Example 3, we guide the reader through ETT regulation decisions for an example property involving temporal and propositional operators, and in Example 4 we show the quantitative application of Definition 9. Later in IV-D we argue why Definition 9 is a suitable event-triggering mechanism to solve Problem 1 and further elaborate on some of the reasoning that led to Definition 9.

Definition 9 (ETT^ρ)**.** *We define the ETT for a signal y_i at time t_{k+1}, for the STL property φ, $\forall t_k \in [t_0, t_{end}]$ recursively as:*

$$\delta^*_{y_i,k+1}(\varphi, \hat{\mathbf{x}}_k, \beta_k) =$$
$$\begin{cases} \delta^{\in}_{y_i,k+1}(\varphi, \hat{\mathbf{x}}_k, \epsilon_{y_i,\varphi}, \beta_k) \\ \qquad \text{if } \varphi = \varphi^p, \\ \min\limits_{\varphi_s \in \{\varphi_1, \varphi_2\}} \delta^*_{y_i,k+1}(\varphi_s, \hat{\mathbf{x}}_k, \beta_k) \\ \qquad \text{if } \varphi = \varphi_1 \wedge \varphi_2, \\ \min\limits_{(\varphi_s, \varphi_l) \in \{(\varphi_1,\varphi_2),(\varphi_2,\varphi_1)\}} \\ \quad \delta^*_{y_i,k+1}(\varphi_s, \hat{\mathbf{x}}_k, \max(\beta_k, \zeta^*_\rho(\varphi_l, \hat{\mathbf{x}}_k))) \\ \qquad \text{if } \varphi = \varphi_1 \vee \varphi_2, \\ \min\limits_{\{t_j \in [\max(t_0, t_k - H(\varphi)), t_k]\}} T(y_i, \varphi, \varphi_1, \beta_k, t_k, t_j) \\ \qquad \text{if } \varphi = \diamond_{[a,b]}\varphi_1 \text{ or } \varphi = \square_{[a,b]}\varphi_1, \\ \min\limits_{\{t_j \in [\max(t_0, t_k - H(\varphi)), t_k]\}} \min\limits_{\varphi_s \in \{\varphi_1, \varphi_2\}} T(y_i, \varphi, \varphi_s, \beta_k, t_k, t_j) \\ \qquad \text{if } \varphi = \varphi_1 \mathcal{U}_{[a,b]} \varphi_2, \end{cases}$$

where $T(y_i, \varphi, \varphi_s, \beta_k, t_k, t_j) =$

$$\begin{cases} \delta^*_{y_i,k+1}(\varphi_s, \hat{\mathbf{x}}_k, 0) \\ \quad \text{if } t_k \in [t_j, S(\varphi, t_j)) \text{ and } \beta_j = 0 \\ \delta^*_{y_i,k+1}(PT(\varphi_s), \hat{\mathbf{x}}_k, \beta_k) \\ \quad \text{otherwise}, \end{cases}$$

and $\delta^{\in}_{y_i,k+1}(\varphi^p, \hat{\mathbf{x}}_k, \epsilon_{y_i,\varphi^p}, \beta_k) =$

$$\begin{cases} \delta^+_{y_i,k+1}(\varphi^p, \hat{\mathbf{x}}_k, \epsilon_{y_i,\varphi^p}) \\ \quad + \max(\beta_k - \zeta_\rho(\varphi^p, \hat{\mathbf{x}}_k), 0)\frac{\rho_{max}(\varphi^p)}{\epsilon_{y_i,\varphi^p}} \\ \qquad \text{if } y_i \in \mathbf{y}_{\sim\varphi^p}, \\ \infty \qquad \text{otherwise}. \end{cases}$$

The value of β_k *is initialized to* $\zeta^*_\rho(\varphi, \hat{\mathbf{x}}_k)$ *of* φ *at the root of the syntax tree at every time step, and* $\rho_{max}(\varphi^p)$ *denotes the maximum possible robustness of the inequality property* φ^p*. We apply this Definition to build the set in Definition 5 at every time step to obtain our proposed ETT regulation mechanism* ETT^ρ*.*

Example 3. *In this example, we will walk through the ETT decisions of Definition 9 for the property* $\varphi = (y > 10) \rightarrow \Diamond_{[0,10]}\Box_{[0,1]}(|y| \leq 1)$ *from Ex. 2 and the trace in Fig. 2. Before the first time that* $y > 10$*, the property is satisfied (recall* $\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$*), so when* $y > 10$ *is false, the overall property is true. This implies that the* β_k *values are positive until* $y > 10$ *at time* t_j*. Consequently, the propositional transformation is invoked in the* $T(\cdot)$ *function, removing the temporal operator conditions and enlarging the ETT based on the current robustness of* $|y| < 1$*. At time* t_j*, the* β_j *value becomes 0, and will remain so until* $y \leq 10$*. The system must now regulate the ETT to satisfy a temporal property based on a past condition. Therefore, once* $\beta_j = 0$*, the propositional transformation is no longer applicable, and we stop the propagation of* β_j *values until the robustness becomes positive for all times* t_j *where* $\beta_j = 0$ *was observed. For this duration, the ETT will be regulated based on* $|y| < 1$ *as the* β_k *values is not propagated, and the outer* min *operation in Definition 9 makes sure that the smallest ETT is chosen (i.e. based on* $|y| < 1$ *rather than* $\neg(y > 10)$*). Once the property becomes satisfied for all time steps where* $\beta_j = 0$ *(indicated by the green area in Fig. 2), the propagation of* β_k *values is again enabled. Alternatively, should the property end up not being satisfied, the ETT regulation "obligation" expires at the end of the temporal scope of the property as per Definition 6. Furthermore, for nested temporal operators we see that the "outer" temporal operator, in this case* $\Diamond_{[a,b]}$*, determines whether an outer* β_k *value is propagated to its sub-properties. We note that while temporal operators stop propagation of outer* β_k *values, this does not limit the propagation of* β_k *values within the temporal property. For example, in the property* $\varphi = \varphi_1^p \rightarrow \Diamond_{[0,10]}\Box_{[0,1]}(\varphi_2^p \vee \varphi_3^p)$ *the inequality properties* φ_2^p, φ_3^p *can still benefit from the ETT relaxation of the* $\vee$ *operator due to the* $\max(\beta_k, \zeta^*_\rho(\varphi, \hat{\mathbf{x}}_k))$ *operation in Definition 9. This is allowed because* φ_2^p *and* φ_3^p *are in the same "scope" and thus related to the satisfaction at the same time step.*

Example 4. *Consider the property* $\varphi = \neg\varphi_1^p \vee (\Box_{[2,5]}\varphi_2^p)$ *where* $\varphi_1^p = y_1 > 5, \varphi_2^p = y_2 > 10$*,* $y_1 \in [0, 20]$ *and* $y_2 \in [5, 50]$ *with* $\mathbf{y}_{\sim\varphi_1} = \{y_1\}$ *and* $\mathbf{y}_{\sim\varphi_2} = \{y_2\}$*. We provide an example trace of the robustness of the inequality properties* φ_1^p *and* φ_2^p *in Fig. 3 to visualize the ETT regulation behavior of Definition 9 for* $\vee$ *and temporal property interaction.*

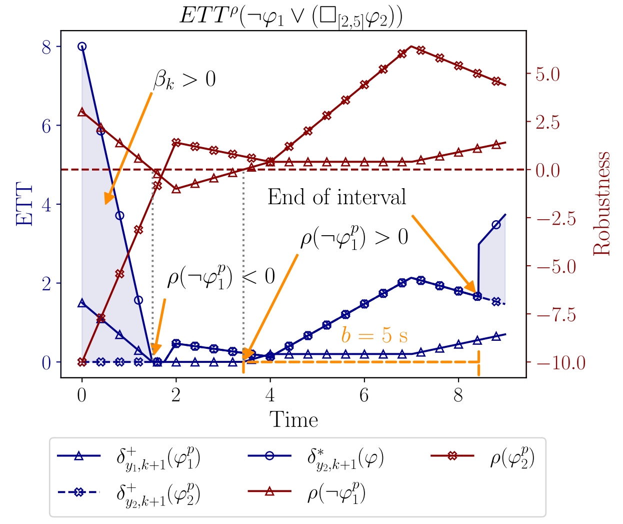

Fig. 3: A visualization of ETTs produced by ETT^ρ for the property $\varphi = \neg\varphi_1^p \vee (\Box_{[2,5]}\varphi_2^p)$ described in Example 4. We choose $\epsilon_{y_1,\varphi_1^p} = 2$, $\epsilon_{y_2,\varphi_2^p} = 3$ and for simplicity write $\rho(\varphi_1^p)$ instead of $\rho(\varphi_1^p, \hat{\mathbf{x}}, t_k)$ and omit irrelevant parameters for the ETT signals. The shaded blue area indicates where the ETTs related to φ_2^p are enlarged using Definition 9 since $\beta_k > 0$ and the property is transformed using the propositional transformation in Definition 8. We see that as $\delta^*_{y_2,k+1}(\varphi, \hat{\mathbf{x}}_k, \beta_k)$ gradually approaches $\delta^+_{y_2,k+1}(\varphi_2^p, \hat{\mathbf{x}}_k, \epsilon_{y_2,\varphi_2^p})$ as $\rho(\neg\varphi_1^p, \hat{\mathbf{x}}, t_k)$ approaches 0. Thereafter, for the duration of the interval (i.e. between $\approx 1.7 - 8.5$ seconds) $\delta^*_{y_2,k+1}(\varphi, \hat{\mathbf{x}}_k, \beta_k) = \delta^+_{y_2,k+1}(\varphi_2^p, \hat{\mathbf{x}}_k, \epsilon_{y_2,\varphi_2^p})$. Then, at the end of the interval (Time $\approx$ 8.5), the ETTs are again affected by the propositional transformation.

D. Putting it all together – Solving Problem 1

Definition 9 comprises the first of two parts of how we solve Problem 1, where the second part entails finding suitable ϵ_{y_i,φ^p} parameters which enable the system to satisfy its properties while triggering as few events as possible. In Theorem 1, we prove that Definition 9 is a suitable candidate for solving Problem 1. Subsequently, we discuss how ETT^ρ leverages STL semantics to increase ETTs resulting in fewer triggered events.

Theorem 1. ETT^ρ *is a suitable candidate to solve Problem 1 under Assumption 1.*

Proof. By setting all $\epsilon_{y_i,\varphi^p} = \infty$, the ETT regulation policy converges to the time-triggered policy described in Assumption 1 as

$$\lim_{\epsilon_{y_i,\varphi^p} \to \infty} \delta^{\in}_{y_i,k+1}(\varphi^p, \hat{\mathbf{x}}_k, \epsilon_{y_i,\varphi^p}, \beta_k) = 0$$

and because in ETT^ρ, the final ETT for a signal is a minimum over all defined ETTs at the inequality property level.

Since the time-triggered approach is assumed to guarantee property satisfaction, so will the ETT regulation mechanism in Definition 9 for sufficiently large ϵ_{y_i,φ^p} values, thus satisfying Eq. (7). Furthermore, as both triggering mechanisms use the same sampling interval, they also have the same upper bound on the number of triggered events, thus ensuring satisfaction of Eq. (8) which concludes the proof. □

Definition 9 employs two techniques to reduce the number of triggered events: **early satisfaction detection (ESD)** and **ETT relaxation** through the $\vee$ operator. The ETT relaxation through the $\vee$ operator, utilizes the fact that the system only has to satisfy at least one of the immediate sub-properties of a $\vee$ property. The propagation of the β_k value then enlarges the ETTs where applicable (i.e. for inequality properties $(\beta_k - \zeta_\rho^*(\varphi, \hat{\mathbf{x}}_k)) > 0$ as per $\delta^{\in}_{y_i,k+1}(\varphi^p, \hat{\mathbf{x}}_k, \epsilon_{y_i,\varphi^p}, \beta_k)$ in Definition 9). Additionally, **ESD** is employed to allow propagation of β_k values as soon as the property is known to be satisfied, increasing the ETTs of its sub-properties resulting in fewer triggered events.

E. Implementation and complexity

Since we are dealing with real-time systems and resource constrained devices, computational and spatial complexity are of significant importance. Below we outline the steps necessary to apply Definition 4 and subsequently analyze the computational and spatial complexity. Finally, we discuss distribution of computation and the computational complexity of other parts of the system in a real-world setting.

1) Algorithm summary: Applying Definition 9 involves 3 steps as outlined below.

S1 Calculate $\beta_k = \zeta_\rho^*(\varphi, \hat{\mathbf{x}}_k)$
S2 Update satisfaction of all temporal sub-properties
S3 Calculate $\{\delta^*_{y_i,k+1}(\varphi, \hat{\mathbf{x}}_k, \beta_k)\}_{y_i \in \mathbf{y}}$

Since we must only keep track of the satisfaction rather than the robustness to apply Definition 9, it suffices to monitor an equivalent Metric Interval Temporal Logic [24] (MITL) formula. Fortunately, MITL monitors which keep track of the satisfaction of the sub-properties have already been implemented where we choose the algorithm described in [25] as they consider a setup applicable to MITL formulas with a bounded number of events per time unit. The number of events per time unit is bounded by default in our case due to the periodic state estimator update.

2) Computational complexity: Computing S1 involves simple $\min$ and $\max$ operations and computing $\zeta_\rho(\varphi^p, \hat{\mathbf{x}}_k)$ of inequality properties. Assuming that the complexity of computing $\zeta_\rho(\varphi^p, \hat{\mathbf{x}}_k)$ is constant, the complexity of computing $\zeta_\rho^*(\varphi, \hat{\mathbf{x}}_k)$ is thus proportional to $|\varphi|$, where $|\varphi|$ is the size of the formula (i.e. the number of operators including inequalities).

The authors in [25] report an amortized computational complexity of $\mathcal{O}(|\varphi|)$. A small extension to the monitoring algorithm in [25] is needed to mark the sub-property as non-satisfaction-dependent at time t_k if $\beta_k > 0$ (i.e. the satisfaction of the overall property does not depend on a specific temporal sub-property at a given time). This can easily be implemented in the ETT calculation downward pass when the β_k value is propagated, where the entry corresponding to the current time is overwritten thus requiring no additional memory and only a simple constant-time update. S3 involves checking the satisfaction status of temporal properties, propagating β_k values accordingly and computing and propagating ETTs upwards. Checking temporal property satisfaction is proportional to $H(\varphi)/T_s$ (i.e. the satisfaction status of all relevant timesteps must be checked) but must only be done once at each timestep. This is in contrast to the ETT calculation and upward propagation where the number of computations and comparisons scales with $|\mathbf{y}| \cdot |\varphi|$ as the ETTs must be computed/compared for all signals at each sub-property.

In summary, the amortized computational complexity for determining the ETTs at every time-step is given by

$$\mathcal{O}((|\mathbf{y}| + H(\varphi)/T_s) \cdot |\varphi|).$$

3) Spatial complexity: The spatial complexity of the algorithm in [25] is reported as $\mathcal{O}(H(\varphi)/T_s \cdot |\varphi|)$. In addition, we will have to store the signal values as well as ϵ_{y_i,φ^p} and $\rho_{\max}(\varphi^p)$ parameters to compute ETTs and $\zeta_\rho(\varphi^p, \hat{\mathbf{x}}_k)$ respectively. Both parameters are needed for every inequality-property, and given that an STL property syntax tree is structurally equivalent to that of a binary tree (i.e. inequality properties are equivalent to leaf nodes), the maximum number of parameters that need to be stored is $|\varphi| + 1$. In summary, the overall spatial complexity is given by

$$\mathcal{O}(|\mathbf{y}| + H(\varphi)/T_s|\varphi|).$$

4) Real-world considerations: Besides determining ETTs using ETT^ρ, several other parts of the system require computation. Most notably, this includes running the state estimator and controller and the communication between smart sensors and the control system. As shown in Fig. 1, the smart sensors receive ETTs at every time step and measurement predictions if the innovation update error is used, which will require significant communication in a real-world scenario. Some solutions to this were briefly discussed in Section III-A, and here we elaborate more on solutions related to distribution of computation. If the model and computational resources of the smart sensors allow it, a local (perhaps reduced-order) model of the system can be maintained, alleviating the need for transmitting signal predictions. Additionally, if the necessary information is available to monitor properties relevant for ETT computation locally, the ETTs may be computed by the smart-sensors removing the need for communication of ETTs. In fact, this problem is similar to that of distributed or decentralized monitoring [26]. In any case, if a centralized monitor and/or control system is required in a real-time scenario, these are best run on an edge node ensuring low-latency. For a reasonably sized model and property to be monitored, this should be possible. When low latency is not necessary or the model is substantially complex, the model/control system and the monitor may run in a cloud environment.

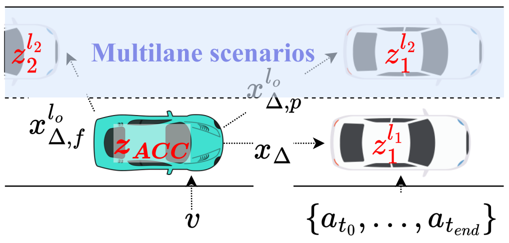

Fig. 4: An outline of simulated scenarios in the case-study adapted from [7]. The turquoise ACC vehicle employs smart sensors to measure the distance between $z_1^{l_1}$ and z_{ACC} (x_Δ), and the lateral distance to $z_1^{l_2}$, $x^{l_o}_{\Delta,p}$ and $z_2^{l_2}$, $x^{l_o}_{\Delta,f}$. Additionally, z_{ACC} measures its own speed v. The sensors communicate measurements to a centralized control system (orange box in Fig. 1). We use $z_n^{l_i}$ to refer to the n-th non-ACC vehicle in the i-th lane based on their original positions.

V. Case study

We apply ETT^ρ from Definition 9 in a simulated case-study where we compare ETT^ρ with the ETT^0 (Assumption 1) and ETT^C (Section III-C) mechanisms. We extend the adaptive-cruise control (ACC) scenario from [7] to incorporate additional lane-switching behavior and more complex temporal properties primarily linked to performance. A sketch of the simulated environment is provided in Fig. 4. For brevity, we refer to [7] for details on simulation setup. Any differences in the simulation setup will be explicitly stated where applicable. Three different STL properties are tested; one property in a single-lane scenario (Section V-A) and two properties in two different multilane scenarios (Sections V-C and V-C1).

We extend the single-lane and multi-lane propositional properties from [7] to include temporal operators which verify performance properties.

A. Single-lane scenario with temporal property

Let $x_{ss}(d_0, v, T) = d_0 + vT$, where $v[m/s] > 0$ is the speed, $T[s] > 0$ is known as the time headway and $d_0[m] \geq 0$ is a constant. The function $x_{ss}(d_0, v, T)$ determines a speed-dependent safe distance which the ACC vehicle controller attempts to keep to the preceding vehicle in the same lane [27]. The behavior verified by $\varphi_a = x_\Delta > x_{ss}(0, v, T)$, is then sufficient to verify that vehicles do not crash. However, typically $d_0 > 0$ and we could be interested in verifying that whenever $x_\Delta < x_{ss}(d_0, v, T)$ the controller can recover to or close to $x_{ss}(d_0, v, T)$ within some duration. Such an STL property can be constructed as follows: $\neg\varphi^p_{c_1} \rightarrow \diamond_{[a,b]}\varphi^p_{c_2}$ where $\varphi^p_{c_1} = x_\Delta > x_{ss}(d_0, v, T)$, $\varphi^p_{c_2} = x_\Delta > x_{ss}(d_0 - \eta, v, T)$ and $d_0 > \eta > 0$. Furthermore, we chose $a = 1$ s, $b = 5.5$ s, $\eta = 0.2$ and $d_0 = 2.7$. Combined with the property φ_a, we have $\boldsymbol{\varphi_c} = \varphi_a \wedge (\neg\varphi^p_{c_1} \rightarrow \diamond_{[1,5.5]}\varphi^p_{c_2})$.

Parameter selection: Initially, we assign $\mathbf{y}_{\sim\varphi^p_{c_2}} = \mathbf{y}_{\sim\varphi^p_a} = \{v, x_\Delta\}, \mathbf{y}_{\sim\varphi^p_{c_1}} = \{\}$, whereafter we consider selecting the ϵ_{y_i,φ^p} parameters. As a starting point, we choose the same ϵ_{y_i,φ^p} parameters as in [7]. We could use identical ϵ_{y_i,φ^p} parameters for $\mathbf{y}_{\sim\varphi^p_{c_2}}$ as for $\mathbf{y}_{\sim\varphi_a}$. However, realizing that $\rho(\varphi_{c_2}, \hat{\mathbf{x}}, t_k) = \rho(\varphi_a, \hat{\mathbf{x}}, t_k) - d_0$ meaning that for any time step, the final ETTs will then always be defined based on $\varphi^p_{c_2}$ and the ETTs based on φ_a are never applied. Instead, we choose small values for the $\epsilon_{x_\Delta,\varphi^p_{c_2}}$ and $\epsilon_{v,\varphi^p_{c_2}}$ parameters, such that $\varphi^p_{c_2}$ only affects ETT regulation once $\rho(\varphi^p_{c_1}, \hat{\mathbf{x}}, t_k) < 0$. For the ETT^0 and ETT^C approaches, we use similar or identical parameters to [7] where small adjustments are necessary to the ETT^C parameters to ensure satisfaction.

Strategy	**Parameters**	$\boldsymbol{\varphi_c}$ (Single lane)		
		$\rho_{min}(\varphi_c)$	$m \pm \sigma(m)$	$T_r \pm \sigma(T_r)$
$ETT^0(\varphi_c)$	$T_s = 0.01$	0.359	7000	5.03 ± 0.03
$ETT^C(\varphi_c)$	$\delta_v = 0.16$ $\delta_{x_\Delta} = 0.49$	0.012	1367 ± 27	5.17 ± 0.04
$ETT^\rho(\varphi_c)$	$\epsilon_{v,\varphi a} = 16.64$ $\epsilon_{x_\Delta,\varphi_a} = 4.95$ $\epsilon_{v,\varphi^p_{c_2}} = 5e^{-5}$ $\epsilon_{x_\Delta,\varphi^p_{c_2}} = 5e^{-5}$	0.074	1158 ± 14 1159 ± 14	5.18 ± 0.04
$ETT^\rho(\varphi_a)$	$\epsilon_{v,\varphi_a} = 16.64$ $\epsilon_{x_\Delta,\varphi_a} = 4.95$	0.007	777 ± 14	5.22 ± 0.04

TABLE II: Parameter configurations resulting in the lowest average number of triggered events (m) while maintaining $\rho_{min}(\varphi_c) > 0$.

Numerical results: Let $ETT^0(\varphi)$, $ETT^C(\varphi)$ and $ETT^\rho(\varphi)$ denote the application of the ETT^0, ETT^C and ETT^ρ triggering mechanisms based on the property φ. We evaluate the single-lane scenario for $ETT^0(\varphi_c)$, $ETT^C(\varphi_c)$ and $ETT^\rho(\varphi_c)$. Additionally, we apply $ETT^\rho(\varphi_a)$ to the single-lane scenario, but where the robustness is calculated based on φ_c. For each event-triggering mechanism, we run 20 simulations with different measurement noise sequences for all parameter configurations. In Table II, we note the parameter configuration resulting in the fewest number of average triggered events (m) with a corresponding minimum robustness $\rho_{min}(\varphi) > 0$ where $\rho_{min}(\varphi) = \min\limits_{\hat{\mathbf{x}} \in \chi_\varphi} \rho(\Box\varphi, \hat{\mathbf{x}}, t_0)$ where χ_φ is the set of all simulation traces for φ. Additionally, we note $\rho_{min}(\varphi_c)$ and the average recovery time T_r (i.e. the time it takes to recover to the desired distance) for the parameter configuration. The results in Table II show that very low values of $\epsilon_{x_\Delta,\varphi^p_{c_2}}$ and $\epsilon_{v,\varphi^p_{c_2}}$ are sufficient and that $ETT^\rho(\varphi_c)$ triggers fewer events than $ETT^C(\varphi_c)$. However, the results in Table II also show that the recovery times for $ETT^\rho(\varphi_a)$ and $ETT^\rho(\varphi_c)$ are nearly identical while $ETT^\rho(\varphi_c)$ triggers $\approx$ **49%** more events compared to $ETT^\rho(\varphi_a)$.

B. Investigation of triggered events relative to robustness

The lack of increase in performance for $ETT^\rho(\varphi_c)$ in the single-lane scenario motivates a deeper investigation. In Fig. 5, we provide a plot of the rate of triggered events relative to the ETT of x_Δ based on φ_c and φ_a from the single lane scenario. As Fig. 5 clearly shows, when the robustness decreases, the rate of triggered events significantly increases. On average, events triggered for ETTs in the range $[0.00, 0.16]$ account for $\approx$ **82%** of the total number of triggered events and only $\approx$ **14%** of the simulation time. Both numbers are significantly lower for φ_a. When regulating the ETTs based on φ_a, ETTs are generally larger, resulting in fewer triggered events. Thus, there is a clear motivation to either avoid low robustness (e.g.

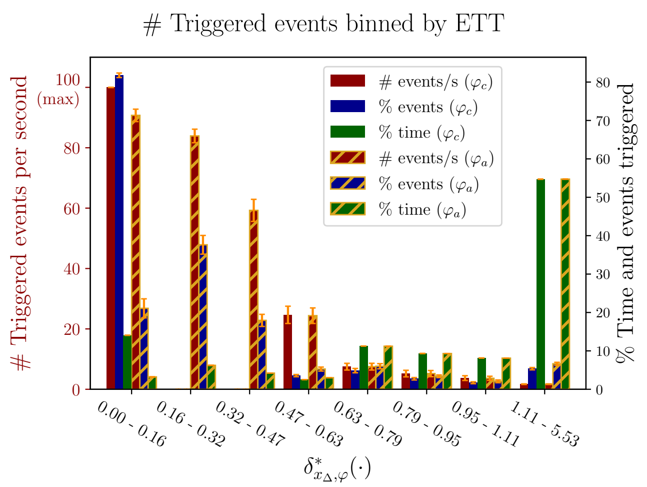

Fig. 5: Number of average triggered events per second, fraction of time spent, and fraction of triggered events for the x_Δ signal in a given ETT interval in the single-lane scenario under $ETT^\rho(\varphi_c)$ and $ETT^\rho(\varphi_a)$. The used parameters can be found in Table II. Orange error-bars indicate one standard deviation. The empty bins for φ_c are a result of the ETT regulation switching from $\delta^*_{x_\Delta,k+1}(\varphi_a, \hat{\mathbf{x}}_k, \beta_k)$ to $\delta^*_{x_\Delta,k+1}(\varphi^p_{c_2}, \hat{\mathbf{x}}_k, \beta_k)$ when $\varphi^p_{c_1}$ is violated.

Strategy	Parameters	$\boldsymbol{\varphi_d}$ (Temporal multilane)		
		$\rho_{min}(\varphi_d)$	$m \pm \sigma(m)$	$T_{l_2} \pm \sigma(T_{l_2})$
$ETT^0(\varphi_d)$	$T_s = 0.01s$	0.833	12644 ± 1	4.05 ± 0.01
$ETT^C(\varphi_d)$	$\delta_v = 0.16$ $\delta_{x_\Delta} = 0.5$ $\delta_{x^{l_o}_{\Delta,f}} = 2.7$ $\delta_{x^{l_o}_{\Delta,p}} = 0.5$	2.898	1524 ± 23	4.17 ± 0.02
$ETT^\rho(\varphi_d)$	$\epsilon_{v,\varphi^p_{b_1}} = 16.64$ $\epsilon_{x_\Delta,\varphi^p_{b_1}} = 4.95$ $\epsilon_{x^{l_o}_{\Delta,f},\varphi^p_{b_2}} = 4.6$ $\epsilon_{x^{l_o}_{\Delta,f},\varphi^p_{b_2}} = 5.32$ $\epsilon_{v,\varphi^p_{b_3}} = 16.64$ $\epsilon_{x^{l_o}_{\Delta,p},\varphi^p_{b_3}} = 4.95$	1.741 1.453	885 ± 9 860 ± 7	4.05 ± 0.01 4.03 ± 0.01

		$\boldsymbol{\varphi_e}$ (Temporal multilane modified)		
		$\rho_{min}(\varphi_e)$	$m \pm \sigma(m)$	$T_{l_2} \pm \sigma(T_{l_2})$
$ETT^0(\varphi_e)$	$-\vert\vert-$	1.977	13646 ± 1	n/a
$ETT^C(\varphi_e)$	$-\vert\vert-$	1.969	1984 ± 36	n/a
$ETT^\rho(\varphi_e)$	$-\vert\vert-$	1.978 1.973	145 ± 8 150 ± 6	n/a

TABLE III: Parameter configurations resulting in the lowest number of triggered events (m) while maintaining a positive minimum robustness ρ_{min} for the temporal properties φ_d and φ_e in the respective multi-lane scenarios. We denote the time the ACC vehicle spends in l_2 as T_{l_2}, and $-||-$ indicates that parameters are reused from the φ_d scenario in the φ_e scenario.

by adapting the controller if possible) or by slightly modifying the ETT regulation mechanism in Definition 4, for example by having some minimum ETT > 0. As per the ETT^C results, we know that ETTs > 0 can enable the system to satisfy its properties, and thus enforcing an ETT lower bound > 0 for ETT^ρ could be viable. We leave this exploration for future work.

C. Multi-lane scenario with temporal property

In addition to the single lane scenario, we study a multilane scenario with a more complex temporal property and control behavior.

Multilane scenario setup: The multilane scenario is identical to the single-lane scenario, except that there is now an additional lane (l_2) with vehicles traveling at a faster constant speed of 35 m/s. The vehicles in l_2 are spaced adequately apart such that the ACC vehicle can safely switch to l_2 when $z^{l_1}_1$ brakes. As opposed to [7], to allow other potentially faster vehicles to overtake in the multi-lane scenario, the ACC vehicle shall eventually change back to l_1. We choose that this should be done as soon as possible within 5 seconds of changing to l_2 if possible. We start with the propositional multilane property $\varphi_b = ((\varphi_{b_1}) \vee (\varphi_{b_2} \wedge \varphi_{b_3}))$ where $\varphi_{b_1} = x_\Delta > x_{ss}(0, v, T)$, $\varphi_{b_2} = x^{l_o}_{\Delta,f} > x_{ss}(0, v^{l_o}_f, T)$, $\varphi_{b_3} = x^{l_o}_{\Delta,p} > x_{ss}(0, v, T)$. Additionally, we choose $\mathbf{y}_{\sim\varphi^p_{b_1}} = \{x_\Delta, v\}, \mathbf{y}_{\sim\varphi^p_{b_2}} = \{x^{l_o}_{\Delta,f}\}, \mathbf{y}_{\sim\varphi^p_{b_3}} = \{x^{l_o}_{\Delta,p}, v\}$. In the case where the ACC vehicle has overtaken $z^{l_1}_1$, the $\vee$ operator will assign a high ETT to the $x^{l_o}_{\Delta,p}$ and $x^{l_o}_{\Delta,f}$ signals if the robustness of the property monitoring the preceding vehicle (i.e. $z^{l_2}_1$ in this case) is large resulting in an inaccurate estimate of the state of $z^{l_1}_1$. The inaccurate estimate of $z^{l_1}_1$ may result in an unsafe lane change if the uncertainty associated with the estimate is not considered by the controller in the lane-switching decision. To provide an accurate estimate of the vehicle(s) in l_1 when the ACC vehicle is in l_2, we construct a temporal property to verify that a safe lane change back to l_1 occurs within 5 seconds. We formulate the STL property $\varphi_d = \varphi_b \wedge (\varphi^p_{d_1} \rightarrow \diamond_{[0,5]}(\varphi^p_{b_2} \wedge \varphi^p_{b_3} \wedge \varphi^p_{d_1} \wedge \diamond_{[0,T_s]}\varphi^p_{d_2}))$ where $\varphi^p_{d_1} = (l_c > l_1), \varphi^p_{d_2} = (l_c < l_2)$, l_c is the current lane of the ACC vehicle, and we assign numbers to l_1 and l_2: $l_1 = 0, l_2 = 100$. The reason for choosing $l_2 = 100$ is to avoid the final robustness being equal to that of $\varphi^p_{d_1}$ or $\varphi^p_{d_2}$.

Parameter selection and switching behavior: We choose $\mathbf{y}_{\sim\varphi^p_{d_1}} = \mathbf{y}_{\sim\varphi^p_{d_2}} = \{\}$ and use the same signal sets for $\varphi^p_{b_1}, \varphi^p_{b_2}$ and $\varphi^p_{b_3}$ and original $\epsilon_{v,\varphi^p_{b_1}}, \epsilon_{x_\Delta,\varphi^p_{b_1}}, \epsilon_{v,\varphi^p_{b_3}}$ and $\epsilon_{x^{l_o}_{\Delta,p},\varphi^p_{b_3}}$ parameters from [7], for $\varphi^p_{b_1}$ and both instances of the inequality property $\varphi^p_{b_3}$. As we are interested in an accurate estimate of $z^{l_1}_1$ when switching back to l_1, we test various values of the $\epsilon_{x^{l_o}_{\Delta,f},\varphi^p_{b_2}}$ parameter for $ETT^\rho(\varphi_d)$ and $\delta_{x^{l_o}_{\Delta,f}}$ for ETT^C.

The lane-switching decision from l_2 to l_1 is implemented such that a lane-switch occurs at the first time step with sufficient distance to the relevant vehicles to safely switch back into l_1.

Numerical results: We provide the number of triggered events, $\rho_{min}(\varphi_d)$ and time spent in l_2 (T_{l_2}) in Table III for $ETT^\rho(\varphi_d)$, $ETT^C(\varphi_d)$ and $ETT^0(\varphi_d)$. We note that no events are triggered when no vehicles are present (i.e. when the ACC vehicle has overtaken $z^{l_1}_1$). $ETT^\rho(\varphi_d)$ triggers significantly more events compared to φ_b from [7], but only slightly more than the single-lane scenario with φ_a. $ETT^\rho(\varphi_d)$ triggers $\approx$ **42%43%** fewer events compared to the ETT^C approach while on average also spending less time in l_2 which we deem desirable. We note that spending less time in l_2 (i.e. switching back to l_1 earlier) results in a lower $\rho_{min}(\varphi_d)$. Thus, attempting to maximize both leads to conflicting requirements.

1) Multi-lane scenario with potential lane-switching detection: The behavior monitored in φ_d prioritizes accuracy to enable the ACC vehicle to change back into l_1 regardless of whether there is room to do so. There may be additional potentially slower vehicles, that the ACC vehicle may also wish to overtake or alternatively, there may not be enough room to safely switch back into l_1. We can capture this conditional behavior, by replacing $\varphi^p_{d_1}$ in φ_d with $(\varphi^p_{d_1} \wedge \varphi^p_{d_3} \wedge \varphi^p_{d_4})$ where $\varphi^p_{d_3} = v_p^{l_o} \geq v_{des}$, v_{des} is the desired speed (a parameter for the IDM) and $\varphi^p_{d_4} = x^{l_o}_{\Delta,p} > x_{ss}(d_0, v, T)$. Thus, the modified version of φ_d becomes $\varphi_e = \varphi_b \wedge ((\varphi^p_{d_1} \wedge \varphi^p_{d_3} \wedge \varphi^p_{d_4}) \rightarrow \Diamond_{[0,5]}(\varphi^p_{b_2} \wedge \varphi^p_{b_3} \wedge \varphi^p_{d_1} \wedge \Diamond_{[0,T_s]}\varphi^p_{d_2}))$. We update the multilane scenario such that there are now 3 vehicles in l_1. The new vehicle in front of $z_1^{l_1}$, which we refer to as $z_0^{l_1}$, is placed $20 + x_{ss}(2.7, 30, 2) = 82.7\ m$ in front of $z_1^{l_1}$ and maintains a constant speed of 30 m/s. In this scenario, there will not be enough room for the ACC vehicle to switch back into l_1. We note the results for the ETT^0, ETT^C and ETT^ρ approaches in Table III under "φ_e (Temporal multilane modified)". The results in Table III show that the added constraints for the implication correctly identify that there is not enough room to switch back to l_1 resulting in generally larger ETTs and $\approx$ **93%92%** fewer triggered events compared to the ETT^C approach.

This concludes the presentation of the case-study. In the following Section, we summarize our contribution.

VI. Concluding remarks

We have proposed and evaluated a novel ETT regulation synthesis mechanism ETT^ρ that extends ETT regulation for propositional properties [7] to arbitrary STL properties. Using the property structure and quantitative semantics and early satisfaction detection capabilities of STL, our method is able to seamlessly synthesize ETT regulation from propositional and temporal logic requirements. This enables ETT^ρ to identify situations where system properties are known to be satisfied and the communication frequency can be reduced. Our algorithm focuses on event-triggered transmission of system signals rather than event-triggered control, which means that our algorithm does not constrain the system dynamics and can potentially also be applied to systems without controllers. The effectiveness of ETT^ρ was demonstrated in a case-study concerning an adaptive cruise control scenario, and resulted in a significant reduction (between $\approx$ **4243**% and **9392**%) in the number of triggered events compared to a constant ETT approach. Based on these promising initial results, we believe that ETT^ρ and potential future extensions will be applicable to a wide range of real-world cyber-physical systems due to its ability to adaptively match communication frequency to complex runtime requirements specified in STL. Applying ETT^ρ to real-world cyber-physical systems will be a core part of our future work and will require developing an efficient runtime algorithm for (distributed) computation of ETTs for ETT^ρ on resource constrained devices. Algorithms for runtime temporal logic monitoring on resource-constrained devices have already been proposed (e.g. [28]), further strengthening our confidence in practical applications of ETT^ρ.

References

[1] Z. Lu and G. Guo, "Control and communication scheduling co-design for networked control systems: A survey," *International Journal of Systems Science*, vol. 54, no. 1, pp. 189–203, Jan. 2023.

[2] C. Peng and F. Li, "A survey on recent advances in event-triggered communication and control," *Information Sciences*, vol. 457–458, pp. 113–125, Aug. 2018.

[3] W. Chen, D. Shi, J. Wang, and L. Shi, "Event-Triggered State Estimation: Experimental Performance Assessment and Comparative Study," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 5, pp. 1865–1872, Sep. 2017.

[4] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan, "Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications," in *Lectures on Runtime Verification: Introductory and Advanced Topics*, ser. Lecture Notes in Computer Science, E. Bartocci and Y. Falcone, Eds. Cham: Springer International Publishing, 2018, pp. 135–175.

[5] I. Haghighi, A. Jones, Z. Kong, E. Bartocci, R. Gros, and C. Belta, "SpaTeL: A novel spatial-temporal logic and its applications to networked systems," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '15. New York, NY, USA: Association for Computing Machinery, Apr. 2015, pp. 189–198.

[6] S. Bufo, E. Bartocci, G. Sanguinetti, M. Borelli, U. Lucangelo, and L. Bortolussi, "Temporal Logic Based Monitoring of Assisted Ventilation in Intensive Care Patients," in *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*, T. Margaria and B. Steffen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, vol. 8803, pp. 391–403.

[7] V. Tang, C. Gomes, and D. E. Lucani, "Precision on Demand: Propositional Logic for Event-Trigger Threshold Regulation," *IEEE Internet of Things Journal*, vol. 12, no. 3, pp. 2674–2689, Feb. 2025.

[8] L. Lindemann, D. Maity, J. S. Baras, and D. V. Dimarogonas, "Event-triggered Feedback Control for Signal Temporal Logic Tasks," in *2018 IEEE Conference on Decision and Control (CDC)*. Miami Beach, FL: IEEE, Dec. 2018, pp. 146–151.

[9] L. Long and G. Gao, "Dynamic event-triggered prescribed performance control for nonlinear systems with signal temporal logic," *International Journal of Robust and Nonlinear Control*, p. rnc.7226, Feb. 2024.

[10] O. Maler and D. Nickovic, "Monitoring Temporal Properties of Continuous Signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, ser. Lecture Notes in Computer Science, Y. Lakhnech and S. Yovine, Eds. Berlin, Heidelberg: Springer, 2004, pp. 152–166.

[11] A. Donzé and O. Maler, "Robust Satisfaction of Temporal Logic over Real-Valued Signals," in *Formal Modeling and Analysis of Timed Systems*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 92–106.

[12] J. V. Deshmukh, A. Donzé, S. Ghosh, X. Jin, G. Juniwal, and S. A. Seshia, "Robust online monitoring of signal temporal logic," *Formal Methods in System Design*, vol. 51, no. 1, pp. 5–30, Aug. 2017.

[13] T. Yamaguchi, B. Hoxha, and D. Ničković, "RTAMT – Runtime Robustness Monitors with Application to CPS and Robotics," *International Journal on Software Tools for Technology Transfer*, Oct. 2023.

[14] D. Shi, L. Shi, and T. Chen, *Event-Based State Estimation*, ser. Studies in Systems, Decision and Control. Cham: Springer International Publishing, 2016, vol. 41.

[15] M. Miskowicz, "Send-On-Delta Concept: An Event-Based Data Reporting Strategy," *Sensors*, vol. 6, no. 1, pp. 49–63, Jan. 2006.

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

[17] X. Ge, S. Xiao, Q.-L. Han, X.-M. Zhang, and D. Ding, "Dynamic Event-Triggered Scheduling and Platooning Control Co-Design for Automated Vehicles Over Vehicular Ad-Hoc Networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 1, pp. 31–46, Jan. 2022.

[18] Y. Pan, Y. Wu, and H.-K. Lam, “Security-Based Fuzzy Control for Nonlinear Networked Control Systems With DoS Attacks via a Resilient Event-Triggered Scheme,” *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 10, pp. 4359–4368, Oct. 2022.

[19] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, “Clock synchronization for wireless sensor networks: A survey,” *Ad Hoc Networks*, vol. 3, no. 3, pp. 281–323, May 2005.

[20] M. Doudou, D. Djenouri, and N. Badache, “Survey on Latency Issues of Asynchronous MAC Protocols in Delay-Sensitive Wireless Sensor Networks,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 528–550, 2013.

[21] E. Bartocci, Y. Falcone, A. Francalanza, and G. Reger, “Introduction to Runtime Verification,” in *Lectures on Runtime Verification: Introductory and Advanced Topics*, ser. Lecture Notes in Computer Science, E. Bartocci and Y. Falcone, Eds. Cham: Springer International Publishing, 2018, pp. 1–33.

[22] A. Pnueli, “The temporal logic of programs,” in *18th Annual Symposium on Foundations of Computer Science (Sfcs 1977)*, Oct. 1977, pp. 46–57.

[23] L. Convent, S. Hungerecker, M. Leucker, T. Scheffel, M. Schmitz, and D. Thoma, “TeSSLa: Temporal Stream-Based Specification Language,” in *Formal Methods: Foundations and Applications*, T. Massoni and M. R. Mousavi, Eds. Cham: Springer International Publishing, 2018, pp. 144–162.

[24] R. Alur, “The Benefits of Relaxing Punctuality.”

[25] H.-M. Ho, J. Ouaknine, and J. Worrell, “Online Monitoring of Metric Temporal Logic,” in *Runtime Verification*, B. Bonakdarpour and S. A. Smolka, Eds. Cham: Springer International Publishing, 2014, vol. 8734, pp. 178–192.

[26] A. Francalanza, J. A. Pérez, and C. Sánchez, “Runtime Verification for Decentralised and Distributed Systems,” in *Lectures on Runtime Verification: Introductory and Advanced Topics*, E. Bartocci and Y. Falcone, Eds. Cham: Springer International Publishing, 2018, pp. 176–210.

[27] M. Treiber and A. Kesting, *Traffic Flow Dynamics: Data, Models and Simulation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[28] J. Geist, K. Y. Rozier, and J. Schumann, “Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems,” in *Runtime Verification*, B. Bonakdarpour and S. A. Smolka, Eds. Cham: Springer International Publishing, 2014, pp. 215–230.