
Feng Tan*, Yufei Wang*, Qixin Wang*, Lei Bu†, Rong Zheng‡, Neeraj Suri**

Abstract—Cyber-Physical Systems (CPS) integrate discrete-time computing and continuous-time physical-world entities, which are often wirelessly interlinked. The use of wireless safety critical CPS (control, healthcare etc.) requires safety guarantees despite communication faults. This paper focuses on one important set of such safety rules: Proper-Temporal-Embedding (PTE). Our solution introduces hybrid automata to formally describe and analyze CPS design patterns. We propose a novel lease based design pattern, along with closed-form configuration constraints, to guarantee PTE safety rules under arbitrary wireless communication faults. We propose a formal methodology to transform the design pattern hybrid automata into specific wireless CPS designs. This methodology can effectively isolate physical world parameters from affecting the PTE safety of the resultant specific designs. We conduct a case study on laser tracheotomy wireless CPS to show that the resulting system is safe and can withstand communication disruptions.

I. INTRODUCTION

To introduce the CPS context [1]–[6], we consider a classical system approach and annotate it with CPS specifics.

Consider a distributed CPS system where each entity has an abstract “safe” state and an abstract “risky” state. During idle time, all entities dwell in their safe states. However, to accomplish a collective task, a distributed procedure must be carried out: relevant entities must enter respective risky states in a fixed order and with certain minimal temporal spacing; and then (after the intended task is done) exit to the respective safe states in exactly the reverse order, and with the requisite temporal spacing. This is called Proper-Temporal-Embedding (PTE). Furthermore, each entity’s continuous dwelling time (i.e. the duration that it continuously stays in the state) in its “risky” state must be upper bounded by a constant. The safety rules encompassing the discrete ordering and continuous-time temporal conditions are termed as PTE safety rules.

As an example of PTE safety (see Fig. 1), in laser tracheotomy [3] (a classical case study in CPS), the oxygen ventilator has the “safe” ventilating state, and the “risky” pause state; the laser-scalpel has the “safe” shutdown state, and the “risky” emission state. In order to emit the laser, the oxygen ventilator must first enter the pause state, and only then can the laser-scalpel enter the emission state. Otherwise, the laser emission can trigger fire on the oxygen ventilated trachea of the patient. Conversely, the laser-scalpel must first exit the emission state, and then the ventilator can exit the pause state.

Thirdly, certain minimal temporal spacing must be maintained during enter/exit of “risky” states, as shown by \( t_1 \) and \( t_2 \) in Fig. 1 (e.g., \( t_1 \) means that only after the oxygen ventilator has paused for \( t_1 \) can laser start emission, otherwise the patient’s trachea may still have high enough oxygen concentration to catch fire; note this “pause \( t_1 \) before laser emission” approach is chosen in real practice because hard real-time and error-free trachea oxygen level sensing is impractical). Fourthly, the continuous dwelling time, as shown by \( t_3 \) and \( t_4 \) in Fig. 1, must each be upper bounded by a constant (e.g., the ventilator pause duration \( t_3 \) must be upper bounded, for otherwise the patient may suffocate to death). Modeling these sequenced CPS operations constitute design patterns.

Fig. 1. Proper-Temporal-Embedding Example

Note that the above CPS procedure (i.e. the PTE safety rules) is not a conventional distributed transaction. In a classical distributed transaction, each participating entity can check-point its current state, i.e., log based recovery for aborts. If the distributed transaction fails, each entity can roll-back to its check point. This is often infeasible for physical world entities in CPS, e.g., we cannot revive (roll-back) a killed patient; even reviving (roll-back) an injured patient to his/her check-pointed state when he/she enters hospital is hard.

As the CPS environment entails wireless-connected sensing, control and computing entities, guaranteeing PTE safety rules necessitates consideration of unreliable wireless communication. Thus, we utilize and adapt the established notion of “leasing” [7]–[12], to ensure auto-reset of distributed entities under communication faults. The basic idea is that each entity’s dwelling duration in risky state is “lease” based. A lease is a timer, which takes effect when the entity enters the risky state. When the lease expires, the entity exits the risky state, no matter if it receives exit command from another entity or not.

This paper develops a novel approach to utilize “leasing” to ensure PTE safety rules in wireless CPS. Our contributions being:

1. We propose utilizing hybrid modeling [13]–[15] to describe and analyze CPS design patterns. Hybrid modeling is a formal technique to describe/analyze both the discrete and continuous dynamics of a system, hence it is suitable for CPS. Recently, hybrid modeling has gained popularity for CPS, though to our best knowledge, it is mostly used for verification.
and we are the first to apply it to CPS design pattern research.

2. We propose a rigorous lease based design pattern for wireless CPS; and identify a set of closed-form constraints on software (i.e. cyber) configuration parameters. We prove that as long as these constraints are satisfied, the design pattern guarantees PTE safety rules under arbitrary packet losses over wireless.

3. We propose a formal methodology to transform the design pattern hybrid automata into specific wireless CPS designs. This methodology can effectively isolate physical world parameters (which are much harder to control, if not impossible, compared to the software/cyber parameters) from affecting the PTE safety of the resultant specific wireless CPS designs1.

4. A case study on laser tracheotomy wireless CPS is presented to validate our proposed approach and analysis.

The rest of the paper is organized as follows. Section II introduces the CPS hybrid modeling background; Section III describes the requirements to guarantee PTE safety rules; Section IV formally defines the lease based design pattern, and proves its guarantee over PTE safety rules. The methodology to elaborate the design patterns into specific designs is also presented here. Section V conducts a case study to validate our proposed approach. We present the related work in Section VI.

II. BACKGROUND, TERMINOLOGY AND MODELS

A. The Hybrid Modeling Terminology

Hybrid modeling is based on hybrid automaton [13]–[16], and has been adapted in modeling CPS since it can formally describe/analyze both discrete (cyber) and continuous (physical) dynamics. For example, Fig. 2 illustrates a hybrid automaton that describes the discrete/continuous behaviors of a stand-alone ventilator. \(H_{\text{vent}}(t)\) is the height of the ventilator cylinder at time \(t\). The ventilator initially dwells in the location of “PumpOut”: the cylinder moves downward at speed 0.1 (m/s). When it hits bottom (\(H_{\text{vent}} = 0\)), it moves to location “PumpIn”, where the cylinder moves upward at speed 0.1 (m/s). When it hits ceiling (\(H_{\text{vent}} = 0.3\) (m)), it moves to location “PumpOut” again, so on and so forth.

As the goal of this paper is to provide formal descriptions and analysis, it is necessary to first give the formal definition of hybrid automaton. We use the hybrid automaton of Fig. 2 to explain the following abstract definitions.

According to [13]–[15], a hybrid automaton \(A\) is a tuple \((\vec{x}(t), V, \text{inv}, F, E, g, R, L, \text{syn}, \Phi_0)\) of following components:

1. A data state variables vector \(\vec{x}(t) = (x_1(t), x_2(t), \ldots, x_n) \in \mathbb{R}^n\) of \(n\) data state variables of time \(t\), where \(n\) is called the dimension of \(A\). A possible evaluation of \(\vec{x}(t)\), denoted as \(\vec{s} \in \mathbb{R}^n\), is called a data state of \(A\) (at time \(t\)). In the example of Fig. 2, the data state variables vector is \((H_{\text{vent}}(t)), i.e. it contains only one data state variable: \(H_{\text{vent}}(t)\), which is the height of the ventilator cylinder at time \(t\).

2. A finite set \(V\) of vertices called locations. The state of \(A\) (at time \(t\)) is a tuple \(\phi(t) = (t, \vec{x}(t))\) of two variables of time \(t\); the aforementioned data state variables vector \(\vec{x}(t)\), and the location counter \(t(t) \in V\), which indicates the current location that \(A\) dwells at. In the example of Fig. 2, the ventilator hybrid automaton has two locations: PumpOut and PumpIn.

3. A function \(\text{inv}\) that assigns to each \(v \in V\) a subset of \(\mathbb{R}^n\), aka. the invariant set. As long as the location counter \(\ell(t) = v\), \(\vec{x}(t)\) must satisfy \(\vec{x}(t) \in \text{inv}(v)\). In the example of Fig. 2, in location PumpOut, the invariant is that the ventilator cylinder height \(H_{\text{vent}}(t)\) stays in the range \(0 < H_{\text{vent}}(t) \leq 0.3\) (m).

4. A set of flow maps \(F = \{f_v : \mathbb{R}^n \mapsto \mathbb{R}^n, \forall v \in V\}\), with each element \(f_v\) defining a set of differential equations \(\dot{\vec{x}} = f_v(\vec{x})\) over data state variables vector \(\vec{x}(t)\) for each location \(v \in V\). These differential equations specify the continuous dynamics of \(\vec{x}(t)\) when \(\ell(t) = v\). In the example of Fig. 2, in location PumpOut, the flow maps only involve one differential equation: \(H_{\text{vent}}(t) = -0.1\) (m/s), i.e. the ventilator cylinder pushes downward at a velocity of \(-0.1\) (m/s).

5. A finite set of edges \(E\). Each edge \(e \in E\) identifies a discrete transition \((v, v')\) from a source location \(v \in V\) to a destination location \(v' \in V\). We denote the source location of edge \(e\) as src(e); while the destination location as des(e). An edge \((v, v')\) specifies the possible discrete dynamics of \(A\)’s state: it can switch from \(t(t) = v\) to \(t(t') = v'\). In the example of Fig. 2, there are two edges: from location PumpOut to PumpIn, and vice versa.

6. A guard function \(g : E \mapsto \mathbb{R}^n\) that assigns each \(e \in E\) a guard set \(g(e) \subseteq \text{inv}(\text{src}(e))\). Discrete transition \(e\) can only take place when \(\vec{x}(t) \in g(e)\). In the example of Fig. 2, the guard condition for the edge (transition) from PumpOut to PumpIn is that the ventilator cylinder reaches the bottom of its movement range, i.e. \(H_{\text{vent}}(t) = 0\).

7. A finite set of reset functions \(R = \{r_e : \text{inv}(\text{src}(e)) \mapsto g(\text{inv}(\text{des}(e))), \forall e \in E\}\). When \(A\)’s state switches from \(t(t) = \text{src}(e)\) to \(t(t') = \text{des}(e)\) via transition \(e \in E\), \(\vec{x}(t')\) is assigned a new data state from set \(r_e(\vec{x})\). In the example of Fig. 2, the reset functions for both edges are the identity function, i.e., the state variables vector \((H_{\text{vent}}(t))\) does not change value after each transition (edge). We hence omit the reset functions in the figure.

8. A finite set \(L\) of synchronization labels and a synchro-

\[\text{Fig. 2. Hybrid Automaton } A'_{\text{vent}} \text{ of a Stand-Alone Ventilator. } H_{\text{vent}}(t) \text{ is the data state variable denoting the ventilator’s cylinder height at time } t. \]
nization labeling function $\text{syn}$ that assigns to each edge $e \in E$ a synchronization labeling function $\text{syn}(e) \in L$. A synchronization labeling consists of a root and a prefix, which respectively represent an event and the role of the hybrid automaton for that event.

When entity $\xi_1$ (whose hybrid automaton is $A_1$) sends an event $l$ to entity $\xi_2$ (whose hybrid automaton is $A_2$), a transition $e_1$ in $A_1$ takes place, and on receiving the event, transition $e_2$ is triggered in $A_2$. Correspondingly, we put a synchronization label $l$ to $e_1$ and $?l$ to $e_2$. We respectively add the prefixes ! and ? to the root $l$, to distinguish the sender and the receiver of event $l$. In case $l$ is received unreliably, which is typical for wireless, we use $??$ instead of a single ? prefix. Synchronization labels with different prefixes or roots are regarded as different. For example, !, ??, ??l are considered three different synchronization labels, though they are related to a same event by the root $l$.

If an event (correspondingly, a synchronization label root) is communicated across multiple hybrid automata, then the corresponding synchronization labels are internal; otherwise, the corresponding synchronization labels are external. For an internal synchronization label whose corresponding event does not have receiver(s), prefix ! is omitted.

In the example of Fig. 2, when the transition from location PumpOut to PumpIn happens, event $evtV\text{PumpIn}$ happens; in the other way around, event $evtV\text{PumpOut}$ happens. The ! prefix to $evtV\text{PumpIn}$ and $evtV\text{PumpOut}$ in the figure indicates the events are broadcasted. If there are other hybrid automata in the system, some transitions may be triggered when receiving these events, the corresponding transitions are labeled with $?evtV\text{PumpIn}$ or $?evtV\text{PumpOut}$. In case the reception of events are via unreliable (e.g. wireless) communication links, the corresponding labels should be $??evtV\text{PumpIn}$ or $??evtV\text{PumpOut}$.

9. A set of possible initial states $\Phi_0 \subseteq \{v, s \in V \times \mathbb{R}^n | v \in V, s \in \text{inv}(v)\}$. We also call $\Phi_0$'s projection on location set $V$ as initial locations, denoted as $\Phi_0|_V$. In the example of Fig. 2, the possible initial states can be $\Phi_0 = \{(\text{PumpOut}, (h_0))\}$, where $h_0 \in [0, 0.3]$: i.e., starting from location PumpOut and cylinder height $H_{\text{entr}}(0) \in [0, 0.3](m)$.

B. System and Fault Model

A hybrid system $H$ is a collection of hybrid automata (each is called a member hybrid automaton of $H$), which execute concurrently and coordinate with each other via event communications (i.e., the sending/receiving of synchronization labels). For simplicity, in this paper, we assume no shared data state variables nor shared locations between different hybrid automata of a hybrid system. That is, data state variable names or location names are local to their respective hybrid automata.

A distributed sink-based wireless CPS consists of the following entities: a central base station $\xi_0$ and $N$ (in this paper, we require $N \geq 2$) remote entities $\xi_1, \xi_2, \ldots, \xi_N$. A wireless communication link from the base station to a remote entity is called a downlink; and a wireless communication link from a remote entity to the base station is called an uplink. We assume that there is no direct wireless communication links between any two remote entities (such practice is desirable for wireless applications with high dependability requirements [17], [18]).

We assume that each packet’s checksum is strong enough to detect any bit error(s); a packet with bit error(s) is discarded at the receiver. Our fault model assumes that packets sent via wireless can be arbitrarily lost (not received at all, or discarded at the receiver due to checksum errors). As per PTE safety requirements, the uplink communication delays are specified and handled by the base station. For the downlink, the remote entities locally specify delays as acceptable or as lost-messages.

III. THE BASIS AND SPECIFICATION OF THE PTE SAFETY RULES

For the wireless CPS system and communications fault model described in Section II-B, various safety requirements can be proposed. Addressing all of them is beyond the scope of this paper. Instead, this paper considers a representative subset of such safety requirements, i.e., the requirement to guarantee PTE safety rules. We start by defining these safety rules.

Let hybrid system $H = \{A_i | i = 0, 1, \ldots, N\}$ describes a wireless CPS. The hybrid automaton $A_i$ describes wireless CPS member entity $\xi_i$. The synchronization labels/functions describe the communication relationships between these hybrid automata.

We assume that for each hybrid automaton $A_i = (\bar{x}_i(t), V_i, \text{inv}_i, F_i, E_i, g_i, r_i, L_i, \text{syn}_i, \Phi_i)$, where $i = 1 \sim N$. $V_i$ is partitioned into two subsets: $V_i^{\text{safe}}$ and $V_i^{\text{risky}}$. We call a location $v$ a “safe-location” iff $v \in V_i^{\text{safe}}$; and a “risky-location” iff $v \in V_i^{\text{risky}}$ (note we do not differentiate the safe/risky locations for $\xi_0$).

There are two types of PTE safety rules, namely:

**PTE Safety Rule 1 (Bounded Dwelling):** Each entity $\xi_i$’s ($i = 1 \sim N$) continuous dwelling time (i.e. continuous-stay time-span) in risky-locations is upper bounded by a constant.

To describe the second PTE safety rule, however, we must first introduce the following definition.

**Definition 1 (Proper-Temporal-Embedding Partial Order):** We say that entity $\xi_i$ and $\xi_j$ has a proper-temporal-embedding partial order $\xi_i \prec \xi_j$ iff their respective hybrid automata $A_i$ and $A_j$ always satisfy the following properties:

p1. If $\xi_i$ dwells in safe-locations at time $t$ (i.e. $A_i$’s location counter $\xi_i(t) \in V_i^{\text{safe}}$), then throughout interval $[t, t + T_0^{\text{safe} \rightarrow j}]$, $\xi_j$ dwells in safe-locations, where constant $T_0^{\text{safe} \rightarrow j}$ is the $\xi_i$ to $\xi_j$ enter-risky safeguard interval.

p2. Whenever $\xi_j$ dwells in risky-locations, $\xi_i$ dwells in risky-locations.

p3. If $\xi_j$ dwells in risky-locations at time $t$, then throughout interval $[t, t + T_0^{\text{risky} \rightarrow j}]$, $\xi_i$ dwells in risky-locations, where constant $T_0^{\text{risky} \rightarrow j}$ is the $\xi_j$ to $\xi_i$ exit-risky safeguard interval.
Intuitively, Property p2 implies that whenever entity $\xi_j$ is in risky-locations, then entity $\xi_i$ is already in risky-locations. Property p1 and p3, in addition, specify the safeguard interval requirements that $\xi_i$ and $\xi_j$ enter/exit respective risky-locations. Specifically, Property p1 implies that before $\xi_j$ enters its risky-locations, $\xi_i$ should have already been in risky-locations for at least $T_{\min_{\text{risky}i\rightarrow j}}$. Property p3 implies that after $\xi_j$ exits its risky-locations (i.e. returns to safe-locations), $\xi_i$ must stay in risky-locations for at least $T_{\min_{\text{risky}i\rightarrow j}}$.

The above intuition is illustrated by Fig. 1, where in laser tracheotomy, ventilator $\prec$ laser-scalpel, if we consider "pause" and "emission" are risky-locations and "ventilating" and "shutoff" are safe-locations.

With this notion of PTE partial ordering, the second PTE safety rule is defined as:

**PTE Safety Rule 2 (Proper-Temporal-Embedding):** The proper-temporal-embedding partial ordering between entities $\xi_1, \xi_2, \ldots, \xi_N$ forms a full ordering.

In the following, for narrative simplicity and without loss of generality, we assume that PTE Safety Rule 2 implies a full ordering of

$$\xi_1 < \xi_2 < \ldots < \xi_N.$$ (1)

We call a safety rule set belongs to the category of PTE safety rules iff the rule set consists of and only of PTE Safety Rule 1 and 2. As mentioned before, in this paper, we shall only focus on wireless CPS whose safety rules belong to the category of PTE safety rules. For simplicity, we call such wireless CPS "PTE wireless CPS".

IV. DEVELOPING DESIGN PATTERN BASED SOLUTIONS

To guarantee PTE safety rules described in the previous section, we propose a lease based design pattern approach. This approach is elaborated in the following three subsections. Subsection IV-A formally describes the proposed lease based design pattern. Subsection IV-B analyzes the validity of the proposed design pattern. Subsection IV-C describes how to transform the lease based design pattern into specific designs.

A. The Lease Based Design Pattern

For a PTE wireless CPS, we assume that safety is guaranteed if all its member entities stay in their safe-locations. The challenge arises when a remote entity needs to enter its risky-locations. In the following, we propose a "lease" based design pattern, and (in the subsequent subsections) show that as long as the PTE wireless CPS design complies with the proposed design pattern, the PTE safety rules are guaranteed.

When a remote entity $\xi_k$ ($k \in \{1, 2, \ldots, N\}$) of a PTE wireless CPS requests to enter its risky-locations, PTE Safety Rule 2 and Ineq. (1) imply that entity $\xi_0, \xi_1, \ldots, \xi_k$ must coordinate. This may be achieved through wireless communications (uplink/downlink) via the base station $\xi_0$. However, wireless communications are by nature unreliable. Messages may be lost, and the states of participating entities may become inconsistent, violating the PTE safety rules.

To deal with the unreliable wireless communications, we propose a design pattern based on the well-known "leasing" design philosophy [7]. The "leasing" design philosophy says that every (distributed) resource must be allocated according to a "lease", i.e. a contract specifying the start/expiration time of using that resource. If, by the lease' expiration, the resource has not yet been released by the user, the resource will release itself automatically. Therefore, even if the distributed systems' communication infrastructure fails, every resource will be released in the end (by the user or itself) after all. This can be used to eliminate/heal inconsistent states of distributed systems.

Lease-based design has been widely adopted in distributed computer systems, particularly distributed storage and database systems. We find it can also be applied to cyber-physical systems, where discrete and continuous states intermingle. Compared to the many existing lease based designs in computer systems, the CPS lease based design faces the following paradigm shifts.

First, lease based designs in computer (i.e. cyber) systems are often integrated with distributed check-point and roll-back [7]–[10]. However, in CPS, computers often have little control over the physical world states: these states cannot be check-pointed or rolled-back. For example, we cannot revive a killed patient; nor can we recover a piece of burnt wood.

Second, instead of carrying out check-point and roll-back, we need to enforce the PTE temporal ordering and corresponding safeguard intervals to guarantee safety.

Third, not only computers have little control over CPS physical world states, these states can adversely interfere with cyber system dynamics.

Considering the above paradigm shifts, our lease based design pattern shall not use check-point or roll-back. Instead, its safety is guaranteed by properly configuring temporal logic parameters and (physical world) continuous state variables.

Specifically, there are three roles for PTE wireless CPS entities: Supervisor, Initializer, and Participant. The base station $\xi_0$ serves the role of “Supervisor”. Initially, all entities stay in their respective safe-locations. We only allow a remote entity to proactively request switching to its risky-locations. Such a remote entity is called an “Initializer”.

For the time being, let us assume there is only one Initializer; and without loss of generality, assume the Initializer is remote entity $\xi_N$.

According to PTE Safety Rule 2 and Ineq. (1), when $\xi_N$ requests to enter risky-locations, remote entity $\xi_1, \xi_2, \ldots, \xi_{N-1}$ must enter respective risky-locations before $\xi_N$. Remote entities $\xi_1, \xi_2, \ldots, \xi_{N-1}$ hence play the role of “Participants”.

We require that every entity $\xi_i$’s ($i \in \{0, 1, 2, \ldots, N\}$) dwelling in risky-locations be based on a lease, i.e. a contract between the Supervisor and $\xi_i$. A lease specifies the expiration time of dwelling in the risky-locations, and takes effect upon the entrance to risky-locations. If by the lease expiration, the
Supervisor has not yet aborted/cancelled the lease, $\xi_i$ will exit to safe-location automatically.

Guided by the above design philosophies, we propose the design of Supervisor, Initializer, and Participant as following.

Supervisor:

Conceptually, the Supervisor $\xi_0$ shall start from a “Fall-Back” location. Whenever the Initializer $\xi_N$ requests leasing itself to enter risky-locations, the Supervisor shall lease Participants $\xi_1$, $\xi_2$, ..., $\xi_{N-1}$ according to PTE ordering first. After all $\xi_1 \sim \xi_{N-1}$ are leased (i.e. $\xi_1 \sim \xi_{N-1}$ enter respective risky-locations), the Supervisor approves $\xi_N$’s lease request to enter risky-location. The Initializer $\xi_N$ can also request to cancel the leases; or when an application dependent proposition ApprovalCondition is violated (e.g. in laser tracheotomy wireless CPS, ApprovalCondition means oxygen level $SpO_2$ is higher than threshold $\Theta_{SpO_2}$), Supervisor $\xi_0$ can abort leases. Lease cancellations/abortions are conducted in the reverse PTE order.

Fig. 3. Sketch of Hybrid Automaton $A_{\text{supvsr}}$, the Design Pattern for Supervisor.

The above conceptual design of the Supervisor is specified by a hybrid automaton $A_{\text{supvsr}}$ (see Fig. 3 for the sketch of $A_{\text{supvsr}}$) as follows.

1. $A_{\text{supvsr}}$’s location set $V_{\text{supvsr}}$ include the following locations: “Fall-Back”, “Lease $\xi_i$” (where $i = 1 \sim N$), “Cancel Lease $\xi_i$” (where $i = 1 \sim N$), and “Abort Lease $\xi_i$” (where $i = 1 \sim N$).
2. Initially, the Supervisor dwells in location “Fall-Back”, and all data state variables initial values are zero.
3. When in location “Fall-Back”, if an event $evt\xi_0To\xi_1\text{LeaseReq}$ is received (which is sent by the Initializer requesting for entering risky-locations, see the descriptions for $A_{\text{initzr}}$ in the following), and the Supervisor has been continuously dwelling in “Fall-Back“ for at least $T_{\text{min}}^{\text{FB,0}}$ and the application dependent proposition ApprovalCondition holds, then the Supervisor transits to location “Lease $\xi_1$”.
4. Along this transition\(^2\), the Supervisor sends out event $evt\xi_0To\xi_1\text{LeaseReq}$, requesting leasing Participant $\xi_1$.
5. When in location “Lease $\xi_i$” (where $i = 1 \sim N - 1$), the behavior of Supervisor can be described by Fig. 4 (a).
6. When in location “Lease $\xi_N$”, the behavior of Supervisor can be described by Fig. 4 (b).
7. When in location “Abort Lease $\xi_i$” (where $i = 1 \sim N$), the behavior of Supervisor can also be described by Fig. 4 (c), except that every occurrence of “Cancel" is replaced by “Abort”.

Initializer:

Conceptually, the Initializer $\xi_N$ shall start from a “Fall-Back” location. It can randomly request to lease itself to enter risky-locations. If this request is approved by the Supervisor $\xi_0$, $\xi_N$ enters risky-locations. The dwelling in risky-locations can be cancelled by $\xi_N$ or aborted by $\xi_0$ at any time; otherwise, $\xi_N$ returns to “Fall-Back” when the lease expires.

The above conceptual design of the Initializer is specified by a hybrid automaton $A_{\text{initzr}}$ (see Fig. 5 (a) for the sketch of $A_{\text{initzr}}$) as follows.

1. $A_{\text{initzr}}$’s location set $V_{\text{initzr}}$ include the following locations: “Fall-Back”, “Requesting”, “Entering”, “Risky Core”, “Exiting 1”, and “Exiting 2”. $V_{\text{initzr}}$ includes location “Risky Core” and “Exiting 1”; all other locations belong to $V_{\text{safeinitzr}}$.
2. Initially, the Initializer $\xi_N$ dwells in location “Fall-Back”; and all data state variables initial values are zero.
3. When in location “Fall-Back”, the Initializer $\xi_N$ can send event $evt\xi_NTo\xi_0\text{Req}$ and transit to “Requesting” at any time.
4. When in location “Requesting”, the Initializer $\xi_N$ can send event $evt\xi_NTo\xi_0\text{Cancel}$ and transit back to “Fall-Back” at any time. Secondly, if $\xi_N$ dwells continuously in “Requesting” for $T_{\text{req}}^{\text{max}}$, it will automatically transit back to “Fall-Back”. Thirdly, if event $evt\xi_0To\xi_N\text{Approve}$ is received, $\xi_N$ transits to “Entering”.
5. When in location “Entering”, the Initializer $\xi_N$ can send event $evt\xi_NTo\xi_0\text{Cancel}$ and transit to “Exiting 2”. Secondly, if $evt\xi_0To\xi_N\text{Abort}$ is received, $\xi_N$ also transits to “Exiting 2”. Thirdly, if $\xi_N$ dwells continuously in “Entering” for $T_{\text{enter}}^{\text{max}}$, it transits to “Risky Core”.

\(^2\)In fact, this “transition” includes two consecutive transitions, the first one is on receiving event $evt\xi_NTo\xi_0\text{Req}$, Supervisor enters an intermediate location of 0 dwelling time; and then transit from this intermediate location to “Lease $\xi_1$” and send out $evt\xi_0To\xi_1\text{LeaseReq}$. For narrative simplicity, in the following, such intermediate locations between two consecutive events are not elaborated.
6. When in location “Risky Core”, the Initializer \( \xi_N \) can send event \( \text{evt}_{\xi_N} \text{To}\xi_0 \text{Cancel} \) and transit to “Exiting 1”. Secondly, if \( \text{evt}_{\xi_0} \text{To}\xi_N \text{Abort} \) is received, \( \xi_N \) also transits to “Exiting 1”. Thirdly, if \( \xi_N \) dwells continuously in “Risky Core” for \( T_{\text{run},i} \), it also transits to “Exiting 1”.

7. When in location “Exiting 1” or “Exiting 2”, the Initializer \( \xi_N \) must continuously dwell in the location for \( T_{\text{exit},i} \), and then transit to “Fall-Back”.

Note that all state variable names and location names are local to the corresponding hybrid automata (so e.g. “Fall-Back” of \( A_{\text{supvsr}} \) and “Fall-Back” of \( A_{\text{inizr}} \) are two distinct locations).

**Participant:**

Conceptually, a Participant \( \xi_i \) (\( i = 1 \sim N - 1 \)) shall start from a “Fall-Back” location. Upon receiving lease request from the Supervisor \( \xi_0 \), and if the lease is approved, \( \xi_i \) enters risky-locations. The dwelling in risky-locations can be cancelled by the Initializer \( \xi_N \) or aborted by the Supervisor \( \xi_0 \) at any time; otherwise, \( \xi_i \) returns to “Fall-Back” when the lease expires.

The above conceptual design of Participant \( \xi_i \) (\( i = 1 \sim N - 1 \)) is specified by a hybrid automaton \( A_{\text{ptcpnt}} \) (see Fig. 5 (b) for the sketch of \( A_{\text{ptcpnt}} \)) as follows.

1. \( A_{\text{ptcpnt}} \)'s location set \( V_{\text{ptcpnt}} \) include the following locations: “Fall-Back”, “L0”, “Entering”, “Risky Core”, “Exiting 1”, and “Exiting 2”. \( V_{\text{risky}} \) include location “Risky Core” and “Exiting 1”; all other locations belong to \( V_{\text{safe}} \).

2. Initially, Participant \( \xi_i \) dwells in location “Fall-Back”; and all data state variables initial values are zero.

3. When in location “Fall-Back”, upon receiving event \( \text{evt}_{\xi_0} \text{To}\xi_i \text{LeaseReq} \), \( \xi_i \) transits to a temporary location “L0”.

4. When in “L0”, if an application dependent proposition \( \text{ParticipationCondition} \) sustains, \( \xi_i \) sends event \( \text{evt}_{\xi_i} \text{To}\xi_0 \text{LeaseApprove} \) and transits to “Entering”; otherwise, \( \xi_i \) sends event \( \text{evt}_{\xi_i} \text{To}\xi_0 \text{LeaseDeny} \) and transits back to “Fall-Back”.

5. When in location “Exiting 1” or “Exiting 2”, Participant \( \xi_i \) must continuously dwell in the location for \( T_{\text{exit},i} \), and then transit to “Fall-Back”.

**B. Ensuring Design Pattern Validity**

We now analyze the validity of the proposed design pattern. As mentioned before, the main threat to PTE wireless CPS is the unreliable wireless communications. Event reception between the Supervisor, Initializer, and Participants can be lossy. If some important events are not received, the holistic system can enter an inconsistent state, which jeopardizes PTE safety rules.

A main contribution of this paper is that we prove that by properly configuring the time constants of the aforementioned \( A_{\text{supvsr}} \), \( A_{\text{inizr}} \), and \( A_{\text{ptcpnt}} \), PTE safety rules are guaranteed.
by continuous dwelling time in risky-locations is upper bounded. Then can be arbitrarily lost. These events refer to those between distributed wireless entities.

A communication channel (in the case of our wireless CPS, the resulting specific designs maintains the PTE safety rules.

Finally, we prove following the proposed elaboration methodology, the resulted specific designs maintains the PTE safety rules guarantees.

In the following, we first propose the formal concept of independence between hybrid automata. We then propose a formal methodology on elaborating locations of design pattern hybrid automata with independent child hybrid automata. Finally, we prove following the proposed elaboration method, the resulted specific designs maintains the PTE safety rules guarantees.

Unless explicitly denoted, the rest of the paper assumes every hybrid automaton to be time-block-free and non-zeno.

We now define hybrid automata independence.

\[ T_{\text{wait}} \leq T_{\text{min}} < T_{\text{max}} \]

Then \( H \) satisfies PTE safety rules. Specifically, every entity’s continuous dwelling time in risky-locations is upper bounded by \( T_{\text{wait}} \), and the PTE full ordering of \( \xi_1 < \xi_2 < \cdots < \xi_N \) is maintained; also, events received through unreliable communication channels (in the case of our wireless CPS, these events refer to those between distributed wireless entities) can be arbitrarily lost.

**Proof:** The sketch of the proof is as follows.

First we can prove if the given parameters satisfy Conditions \( c1 \sim c7 \), and that all entities start from “Fall-Back” location, the system will reset itself to “Fall-Back” within \( T_{\text{wait}} + T_{\text{LS1}} \) every time \( \text{ev}_{\xi_0} L_{\xi_1} \text{LeaseReq} \) happens. This is mainly because of the leases: even if messages are lost, leases will expire to guarantee the return to “Fall-Back” of the Initializer and every Participant.

Second, we prove between any two consecutive \( \text{ev}_{\xi_0} L_{\xi_1} \text{LeaseReq} \) events (or the last such event and time \( \infty \)), any entity can only dwell in the risky-locations for once.

Third, due to Conditions \( c1 \sim c7 \), for each \( \xi_i \) and \( \xi_i+1 \) (\( i = 1 \sim N-1 \)), the aforementioned single dwelling intervals of \( \xi_i \) and \( \xi_i+1 \) satisfies PTE enter-risky/exit-risky safeguard interval requirements.

The detailed proof appears at [19].

C. Methodology to Transform Design Pattern into Specific Designs

In this subsection, we propose a methodology to transform the aforementioned design pattern hybrid automata \( A_{\text{supvsr}}, A_{\text{initzr}}, \) and \( A_{\text{ptcpnt}} \) into specific PTE wireless CPS designs.

The intuition is that every location \( v \) of \( A_{\text{supvsr}}, A_{\text{initzr}}, \) and \( A_{\text{ptcpnt}} \) can be expanded by a child hybrid automata \( A' \). However, \( A' \) must be sufficiently independent (i.e. orthogonal) from the rest part of \( A_{\text{supvsr}}, A_{\text{initzr}}, \) and \( A_{\text{ptcpnt}} \), so that it will not interfere the design pattern’s guarantee on PTE safety rules.

For example, except clock/time variables, the design pattern automata \( A_{\text{supvsr}}, A_{\text{initzr}}, \) and \( A_{\text{ptcpnt}} \) contains no other continuous state variables. However, in a specific PTE wireless CPS design, there will be various continuous state variables representing all kinds of physical world properties. For example, in laser tracheotomy PTE wireless CPS, there can be \( H_{\text{vent}}(t) \), the height of the ventilator cylinder at time \( t \), which decides the behavior of the ventilator. Their dynamics are not totally decided by the cyber-software. When elaborating design pattern into specific designs, we must guarantee these details will not affect the PTE safety rule guarantees.

In the following, we first propose the formal concept of independence between hybrid automata. We then propose a formal methodology on elaborating locations of design pattern hybrid automata with independent child hybrid automata. Finally, we prove following the proposed elaboration method, the resulted specific designs maintains the PTE safety rules guarantees.

**Definition 2 (Hybrid Automata Independence):** Given hybrid automata \( A = (\mathcal{X}(t), V, \text{inv}, F, E, g, R, L, \text{syn}, \Phi_0) \) and

\[ T_{\text{wait}} = T_{\text{max}} \]

\[ T_{\text{max}} = \sum_{i=1}^{N} T_{\text{max}}^{i} \]

\[ T_{\text{req},N} = T_{\text{max}}^{\text{req},N} \]

\[ T_{\text{run},i} = T_{\text{max}}^{\text{run},i} \]

\[ T_{\text{exit},i} = T_{\text{max}}^{\text{exit},i} \]

\[ T_{\text{wait}} = T_{\text{max}}^{\text{wait}} \]

\[ T_{\text{LS1}} = T_{\text{max}}^{\text{LS1}} \]

---

3 Please see [20] for the definitions of time-block-free and non-zeno. For the aforementioned design pattern hybrid automata in Fig. 3, 5 (a), 5 (b), as long as Condition \( c1 \sim c7 \) hold, they are time-block-free and non-zeno. Besides, time-block-free and non-zeno are well-known concepts in formal modeling, and most practical hybrid automata are time-block-free and non-zeno. Due to above reasons, we are not going to elaborate the definitions of these two concepts in this paper.
A' = (x'(t), V', inv', F', E', g', R', L', syn', Φ'_0), we say “A and A' are independent” iff
1. \( \text{elements}(x(t)) \cap \text{elements}(x'(t)) = \emptyset; \)
2. \( V \cap V' = \emptyset; \)
3. \( L \cap L' = \emptyset. \)

Furthermore, we say “a set of hybrid automata \( A_1, A_2, \ldots, A_k \) are mutually independent”, iff \( \forall i, j \in \{1, 2, \ldots, k\} \) and \( i \neq j, A_i \text{ and } A_j \) are independent.

We further define simple hybrid automaton.

**Definition 3 (Simple Hybrid Automaton):** A hybrid automaton \( A = (x(t), V, \text{inv}, F, E, g, R, L, \text{syn}, \Phi_0) \) is simple iff
1. \( \forall v_1, v_2 \in V, \text{inv}(v_1) = \text{inv}(v_2). \)
2. \( \forall v \in \Phi_0|_V \cdot \forall \tilde{s} \in \text{inv}(v) \cdot (v, \tilde{s}) \in \Phi_0, \) where \( \Phi_0|_V \) means \( \Phi_0 \)'s projection on \( V. \)
3. \( \forall v \in \Phi_0|_V \cdot (v, 0) \in \Phi_0, \) where \( 0 \) is the zero data state vector.

We can now describe the intuition on the how to elaborate a given hybrid automaton.

**Atomic Elaboration of Hybrid Automaton (Intuition):**

Given a hybrid automaton \( A = (x(t), V, \text{inv}, F, E, g, R, L, \text{syn}, \Phi_0) \), location \( v \in V \), and a simple hybrid automaton \( A' = (x'(t), V', \text{inv}', F', E', g', R', L', \text{syn}', \Phi'_0) \) such that \( A \) and \( A' \) are independent, then we can create the “atomic” elaboration of \( A \) at \( v \) with \( A' \), i.e. a hybrid automaton \( A'' = (x''(t), V'', \text{inv}'', F'', E'', g'', R'', L'', \text{syn}'', \Phi''_0) \), according to the following intuitions:

1. Location \( v \) of hybrid automaton \( A \) is replaced by simple hybrid automaton \( A'. \)
2. All former ingress edges to \( v \) in \( A \) become ingress edges to \( A' \) (\( A' \)'s initial locations to be more specific).
3. All former egress edges from \( v \) in \( A \) become egress edges from \( A'. \)
4. When in \( A' \), the data state variables \( x'(t) \) of \( A \) maintain their continuous behavior as if they are in \( v. \)
5. When out of \( A' \), the data state variables \( x'(t) \) of \( A' \) remain unchanged (until return to \( A' \) again in the future).

We denote \( A'' \), the elaboration of \( A \) at \( v \) with \( A' \), as
\[
A'' = E(A, v, A').
\]

The rigorous formal description on atomic elaboration of hybrid automaton is provided in [19].

Fig. 6 illustrates an example of atomic elaboration of hybrid automaton. Denote the hybrid automaton of Fig. 2 to be \( A''_{\text{entr}} \). We use \( A'_{\text{entr}} \) to elaborate hybrid automaton \( A \) of Fig. 6 (a) at location “Fall-Back”. The resulted elaboration is the hybrid automaton \( A''_{\text{entr}} \) of Fig. 6 (b).

With the above (atomic) elaboration methodology, we can go further.

Given hybrid automaton \( A, k \) distinct locations \( v_1 \sim v_k \in V \) (where \( V \) is \( A \)'s location set), and \( k \) simple hybrid automata \( A_1 \sim A_k \) such that \( A, A_1, \ldots, A_k \) are mutually independent, then we can carry out “(parallel) elaboration of \( A \) at \( v_1, v_2, \ldots, v_k \) with \( A_1, A_2, \ldots, A_k \)” denoted as
\[
E(A, (v_1, v_2, \ldots, v_k), (A_1, A_2, \ldots, A_k)) \quad \text{def} \quad E(\ldots E(E(A, v_1, A_1), v_2, A_2) \ldots, v_k, A_k).
\]

repeat \( k \) times

Denote \( A' = E(A, (v_1, v_2, \ldots, v_k), (A_1, A_2, \ldots, A_k)) \), we also say “\( A' \) elaborates \( A \) at \( v_1, v_2, \ldots, v_k \) with \( A_1, A_2, \ldots, A_k \) respectively”.

Intuitively, parallel elaboration of \( A \) at \( v_1, v_2, \ldots, v_k \) with \( A_1, A_2, \ldots, A_k \) can be implemented by elaborating \( A \) at \( v_1 \) with \( A_1 \), \( v_2 \) with \( A_2 \), so on and so forth, until \( v_k \) with \( A_k \).

If a specific wireless CPS design, described by hybrid system \( H' \), has its member hybrid automata respectively elaborating the Supervisor, Initiator, and Participant design pattern hybrid automata (i.e. \( H_{\text{supervisor}}, H_{\text{initiator}}, \) and \( H_{\text{participant}}, \)) then the design \( H' \) maintains the properties of our design pattern and guarantee of PTE safety rules. Formally, this is expressed in the form of the following theorem.

**Theorem 2 (Design Pattern Compliance):** Given a hybrid system \( H' \) consisting of entities \( \xi_0, \xi_1, \ldots, \xi_N \), which respectively corresponds to hybrid automata of \( A'_0, A'_1, \ldots, A'_N \). If the following conditions are satisfied:
1. There are distinct locations $v_{1}^0, v_{2}^0, \ldots, v_{k_0}^0 \in V_{\supvsr}$, and simple hybrid automata $A_{1}^0, A_{2}^0, \ldots, A_{k_0}^0$, such that $A_{\supvsr}$ and $A_{j}^0$ ($j = 1 \sim k_0$) are independent, and $A_{\ptcpnt,i}$ elaborates $A_{\supvsr}$ at $v_{1}^0, v_{2}^0, \ldots, v_{k_0}^0$ with $A_{1}^0, A_{2}^0, \ldots, A_{k_0}^0$, respectively;

2. For each $i \in \{1, 2, \ldots, N - 1\}$, there are distinct locations $v_{1,i}^0, v_{2,i}^0, \ldots, v_{k_i}^0 \in V_{\ptcpnt,i}$, and simple hybrid automata $A_{1,i}^0, A_{2,i}^0, \ldots, A_{k_i}^0$, such that $A_{\ptcpnt,i}$ and $A_{j}^0$ ($j = 1 \sim k_i$) are independent, and $A_{1,i}$ elaborates $A_{\ptcpnt,i}$ at $v_{1,i}^0, v_{2,i}^0, \ldots, v_{k_i}^0$ with $A_{1,i}, A_{2,i}, \ldots, A_{k_i}$, respectively;

3. There are distinct locations $v_{1,N}^N, v_{2,N}^N, \ldots, v_{k_N}^N \in V_{\initzr}$, and simple hybrid automata $A_{1,N}^N, A_{2,N}^N, \ldots, A_{k_N}^N$, such that $A_{\initzr}$ and $A_{1,N}^N$ ($j = 1 \sim k_N$) are independent, and $A_{1,N}$ elaborates $A_{\initzr}$ at $v_{1,N}^N, v_{2,N}^N, \ldots, v_{k_N}^N$ with $A_{1,N}, A_{2,N}, \ldots, A_{k_N}$, respectively;

4. Hybrid automata $A_{j,i}$ are mutually independent, where $i = 0, 1, \ldots, N, \ j = 1, 2, \ldots, k_i$;

5. Condition $c_1 \sim c_7$ of Theorem 1 sustain;

where $V_{\supvsr}, V_{\ptcpnt,i}$, and $V_{\initzr}$ are respectively $A_{\supvsr}$, $A_{\ptcpnt,i}$, and $A_{\initzr}$’s location sets, then $H'$ satisfies PTE safety rules.

Proof: If not, there must be an execution trace $\phi'(t)$ (see [15] for the rigorous definition of “execution trace”, aka “trajectory” of a hybrid system) of $H'$ that violates PTE safety rules. According to the methodology we elaborate hybrid automata, $\phi'(t)$ corresponds to an execution trace $\phi(t)$ of $H$ (the hybrid system of $A_{\supvsr}, A_{\ptcpnt,i}$ ($i = 1, 2, \ldots, N - 1$), and $A_{\initzr}$) that also violates PTE safety rules. This contradicts Theorem 1. 

V. CASE STUDY

To demonstrate the use of our proposed lease design pattern, we carry out a case study on wireless laser tracheotomy as introduced in earlier sections.

Setup and Modeling:

In wireless laser tracheotomy (see Fig. 7 (a) for the application layout), a patient is under anesthesia, hence must be connected to a ventilator to breathe oxygen. However, a surgeon may randomly request a laser-scalpel to emit laser, to cut the patient’s trachea. Therefore, PTE safety rules apply as follows. Before the emission of laser, the ventilator must have paused for at least $T_{\min}^{\text{risky}:1 \to 2}$ (we regard ventilator as entity $\xi_1$ and the laser-scalpel as entity $\xi_2$); after the emission of laser, the ventilator must wait for at least $T_{\min}^{\text{safe}:2 \to 1}$ before resuming. Otherwise, if high concentration of oxygen in the patient’s trachea (due to ventilation) is present when laser emits, the patient’s trachea can catch fire. In addition, the durations that the laser-scalpel can continuously emit and that the ventilator can continuously pause shall respectively be upper-bounded by a constant.

The ventilator and the laser-scalpel are wirelessly connected via a central base station, which also plays the role of supervisor. The supervisor/laser-scalpel can abort/cancel laser emission at any time (for example, when the supervisor detects the patient’s blood oxygen level reaches below a threshold, it can immediately request aborting laser emission and resuming ventilation), but the PTE safety rules must be maintained.

On the other hand, because the supervisor, laser-scalpel, and the ventilator are connected through wireless, message losses are possible. Therefore, we want all entities’ risky behaviors lease based, so that even with message losses, the wireless CPS can maintain the PTE safety rules.

In order to satisfy the above requirements, we start our design using the proposed methodology.

First, we see the laser tracheotomy PTE wireless CPS consists of three entities (i.e. $N = 2$): the laser tracheotomy supervisor (together with the $SpO_2$ sensor wired to it) plays the role of Supervisor, hence entity $\xi_0$; the (surgeon operated) laser-scalpel plays the role of Initializer, hence entity $\xi_2$; and the ventilator plays the role of Participant 1, hence entity $\xi_1$.

Next, we design the hybrid automata for the laser tracheotomy supervisor, laser-scalpel, and ventilator by respectively elaborating $A_{\supvsr}, A_{\initzr}$, and $A_{\ptcpnt,i}$.

Take the ventilator detailed design for example. The detailed design of a stand-alone ventilator has already been described by the simple hybrid automaton $A_{\text{vent}}'$ of Fig. 2. The stand-alone design of $A_{\text{vent}}'$, however, is not aware of the communications/collaborations with supervisor and laser-scalpel; hence cannot guarantee PTE safety rules. In order to guarantee PTE safety rules, we revise the ventilator design by elaborating the Participant Design Pattern hybrid automaton $A_{\ptcpnt,i}$ (see Section IV-A-Participant; also see Fig. 5 (b) for the sketch of the hybrid automaton) at location “Fall-Back” with $A_{\text{vent}}'$, using the elaboration method described in Section IV-C.

The Initializer hybrid automaton $A_{\initzr}$ and Supervisor hybrid automaton $A_{\supvsr}$ do not need to be further elaborated. They can be directly used to describe the behavior of laser-scalpel and laser tracheotomy supervisor respectively.

Interested readers shall refer to [19] for the diagrams of these detailed designs.
We configure the time parameters of the above detailed design hybrid automata according to common-sense laser tracheotomy requirements [21] as follows. For the Supervisor (i.e., the laser tracheotomy supervisor), $T_{\text{on},0}^\text{min} = 138$, $T_{\text{on},0}^\text{max} = 3s$. For the Initializer (i.e., the laser-scalpel), $T_{\text{max},0}^\text{rec} = 5s$, $T_{\text{max},0}^\text{ent} = 10s$, $T_{\text{max},0}^\text{run} = 20s$, $T_{\text{max},0}^{\text{exit}} = 1.5s$. For the Participant 1 (i.e., the ventilator), $T_{\text{max},1}^\text{ent} = 3s$, $T_{\text{max},1}^\text{run} = 35s$, $T_{\text{max},1}^{\text{exit}} = 6s$. The PTE enter-risky/exit-risky safeguard intervals are $T_{\text{min}}^{\text{risk}y:1\rightarrow 2} = 3s$ and $T_{\text{safe}:2\rightarrow 1} = 1.5s$.

Per Theorem 2, the above configurations guarantee PTE safety rules. To further validate this, we implemented and carried out emulations of the above design.

**Emulation Setup:**

Fig. 7 (b) illustrates the layout of our emulation. The laser tracheotomy ventilator, supervisor, and (surgeon operated) laser-scalpel are respectively emulated by three computers. The patient is emulated by a real human subject (HS).

Instead of actually ventilating the human subject HS, the ventilator emulator displays its current hybrid automata location (“PumpOut”, “PumpIn”, etc.). Human subject HS watches the display and breathe accordingly.

Another complex part of the experiments is the emulation of externally triggered events. We emulate three kinds of such events in the laser tracheotomy hybrid system (all other events are consequences of this set).

The first is the Initializer event $\text{evt}_i^\xi T_{0\xi}^\text{Req}$, triggered when the laser-scalpel is in “Fall-Back” and the surgeon requests to supervisor to emit laser. In the real system, this is triggered by the surgeon’s human will. In our emulation, however, this is emulated by setting up a random timer $T_{\text{on}}$ (following exponential distribution) whenever the laser-scalpel enters “Fall-Back”. When in “Fall-Back” and $T_{\text{on}}$ sets off, the (emulated) surgeon requests to emit laser. Timer $T_{\text{on}}$ is destroyed whenever the laser-scalpel leaves “Fall-Back” location.

The second kind is the Initializer event $\text{evt}_i^\xi T_{0\xi}^\text{Cancel}$, triggered when the laser-scalpel is emitting and the surgeon cancels the request to emit laser. Again in a real system, this is triggered by the surgeon’s human will. In our emulation, this is emulated by setting up a random timer $T_{\text{off}}$ (following exponential distribution) whenever the laser-scalpel is emitting laser. When emitting laser and $T_{\text{off}}$ sets off, the (emulated) surgeon requests to cancel laser emission. Timer $T_{\text{off}}$ is destroyed whenever the laser-scalpel returns to “Fall-Back” location.

The third kind is the Supervisor event $\text{evt}_i^\xi T_{0\xi}^\text{Abort}$ ($i = 1 \sim N$), triggered when the supervisor is in “Lease $\xi_i$” location and $\text{ApprovalCondition}$ becomes false. In our emulation, the human subject HS wears an oximeter (Nonin 9843 [22]), which measures HS’s blood oxygen level in real-time $t$ ($SpO_2(t)$). The oximeter is wired to the laser tracheotomy supervisor emulator. The $\text{ApprovalCondition}$ is that the oximeter reading $SpO_2(t) > \Theta_{SpO_2}$ where $\Theta_{SpO_2}$ is set to 92%.

The supervisor, ventilator, and laser-scalpel emulators communicate with each other via wireless, with supervisor as base station, and the other two as clients. Their wireless interfaces are implemented via ZigBee TMote-Sky motes [23]. In addition, there is an IEEE 802.11g WiFi interference source 2 meters away from the supervisor. The interference source broadcasts interfering WiFi packets at a data rate of 3Mbps at a radio band overlapping with that of the ZigBee TMote-Sky motes’. Because the interference broadcast is independent from the laser tracheotomy wireless CPS communications, any packets/events between the supervisor, ventilator, and laser-scalpel emulation computers can be lost.

**Trials and Results:**

We ran two emulation trials, each of 30 minutes duration. During the emulation, the PTE safety rules are:

1. Neither ventilator pause nor laser emission can last for more than 1 minute (we assume that holding breath for $\leq 1$ minute is always safe);
2. Ventilator pause duration must always properly-temporally-embedding laser emission duration, with entering/exit safeguard interval of $T_{\text{min}}^{\text{risk}y:1\rightarrow 2} = 3(s)$ and $T_{\text{safe}:2\rightarrow 1} = 1.5(s)$.

Violation of either of the PTE safety rules is a failure.

As mentioned before, in the two emulation trials, the emulated surgeon request to emit/cancel-emit laser according to timer $T_{\text{on}}$ and $T_{\text{off}}$, both are random numbers following exponential distribution. The expectation of $T_{\text{on}}$ is 30(s). The expectations of $T_{\text{off}}$ are 18(s) and 6(s) respectively in the two emulation trials.

Because of the use of our proposed lease based design pattern, and the configuration of parameters satisfying Theorem 2, although packets/events between ventilator, supervisor, and laser-scalpel emulator can be arbitrarily lost, the PTE safety rules are never violated. This is shown in Table I, the two rows corresponding to “with Lease” both have 0 failures.

For comparison, we also ran two additional emulation trials with the same configurations but without using the leasing mechanism. Specifically, the ventilator does not set up a lease timer when it is pausing, neither does the laser-scalpel set up a lease timer when it is emitting laser. When the surgeon’s cancel laser emission event is lost or the supervisor’s abort event is lost, no one can terminate the ventilator’s pause or the laser’s emission. Thus, as shown in Table I, the two rows corresponding to “without Lease” both result in many failures.

The intuitive explanations to the above empirical evidences are as follows. Because of leasing, the ventilator’s stay in

<table>
<thead>
<tr>
<th>Trial</th>
<th>$E(T_{\text{on}})$ (sec)</th>
<th># of Laser Emissions</th>
<th># of Failures</th>
<th># of $\text{evt}_{\text{ToStop}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>with Lease</td>
<td>18</td>
<td>19</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>without Lease</td>
<td>18</td>
<td>11</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>with Lease</td>
<td>6</td>
<td>19</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>without Lease</td>
<td>6</td>
<td>12</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

1. Each trial lasts 30 minutes, and is under constant WiFi interference.
2. For each trial, the expectation $E(T_{\text{on}}) \equiv 30$sec.
3. $\text{evt}_{\text{ToStop}}$ occurs when lease expiration forces the laser-scalpel to stop emitting (see Fig. 12), i.e. when lease mechanism takes effect to rescue the system from violating the PTE safety rules.
the pause state (i.e. risky-locations) expires on lease time-out; hence it will automatically return to “Fall-Back” to continue ventilating the patient, even when it is cut-off from communications. Same applies to the laser-scalpel’s stay in the emission state (i.e. risky-locations). Conditions c1 \sim c7 of Theorem 2 further guarantees that the automatic returns to “Fall-Back” of ventilator and laser-scalpel both conform to proper-temporal-embedding even under arbitrary packet/event losses.

We further consider a number of typical scenarios to get better intuitions on why the proposed leasing approach and parameter configuration constraints are critical in guaranteeing PTE safety rules.

One scenario is that after the ventilator is paused and the laser-scalpel is emitting, the surgeon may forget to cancel laser emission until too late (e.g. \( T_{\text{off}} \) is set to 1 hour). In this case, the abort request from the supervisor can stop laser emission and resume ventilator before it is too late. However, this requires a sequence of correct send/receive of events through wireless: \( \text{evt}_{\text{to}} \text{To}_{\text{ok}} \text{Abort} \), followed by \( \text{evt}_{\text{to}} \text{To}_{\text{ok}} \text{Exit} \), and followed by \( \text{evt}_{\text{to}} \text{To}_{\text{ok}} \text{Abort} \). Losing any one of these events at the receiver end will cause PTE safety rules violation. For example, losing \( \text{evt}_{\text{to}} \text{To}_{\text{ok}} \text{Exit} \), the supervisor may think the laser-scalpel is stuck and cannot stop laser emission, hence ventilator shall keep pausing.

With leasing, the laser emission terminates within the lease \( T_{\text{run2}} = 20s \) with or without surgeon’s request to cancel; and the ventilator resumes within the lease \( T_{\text{run1}} = 35s \) with or without supervisor’s requests. Hence PTE safety rules are protected.

Similar analysis applies to the scenario that the surgeon remembers to cancel laser emission, but his/her cancelling request (i.e. \( \text{evt}_{\text{to}} \text{To}_{\text{ok}} \text{Cancel} \)) is not received at the supervisor. Without lease, the ventilator may keep pausing till for too long; with lease, the ventilator will keep pausing for \( T_{\text{run1}} = 35s \) at the most, hence cannot suffocate the patient.

A third scenario involves the parameter configuration constraints. Suppose we set \( T_{\text{max enter,2}} = T_{\text{max enter,1}} = 0s \) (or any other value so that \( T_{\text{max enter,2}} = T_{\text{max enter,1}} \)), then because \( T_{\text{min risky:1}} = 3s > 0 \), Condition c5 of Theorem 1 is violated. Under such design, immediately after the ventilator is paused, the laser-scalpel can emit laser, violating the PTE requirement of \( T_{\text{min risky:1}} = 3s \); that the laser-scalpel must wait for another 3s after the ventilator pauses, and then can it emit laser.

More failure scenarios are possible. However, if we follow the proposed lease based design approach and meet parameter configuration constraints listed in Theorem 1, Theorem 1 and 2 guarantees PTE safety rules.

VI. RELATED WORK

Lease protocol was originally proposed by Gray et al. [7] and is used to provide efficient consistent access to cached data in distributed computer systems. With leases, the inconsistencies caused by communication faults can be recovered. In the past decades, various lease based distributed computer systems have been implemented to achieve system consistency [8]–[10]. Recently, Chen et al. [24] proposed a dynamic lease technique to keep track of the local DNS name servers.

Boix et al. [11] applied leases to mobile ad hoc networks; and Adya et al. [12] applied lease to cloud computing. As pointed out in Section IV-A, all these distributed computer systems are fundamentally different from CPS due to following reasons: 1) check-point and roll-back, which are extensively used in lease protocols for distributed computer systems are often impossible for CPS (e.g. we cannot revive a killed patient); 2) PTE temporal ordering, particularly the minimum safeguard interval requirements are usually not present for distributed computer systems (which usually focus on causal precedences); 3) in CPS, uncontrollable physical world parameters can often interfere with the computer software dynamics.

Formal methods and model checking techniques have been widely used in various applications. Majzik et al. [25] apply the formal methods in the quantitative evaluation of the Driver Machine Interface (DMI). Ramasamy et al. [26] employ the SPIN model checker to validate the correctness of a formal model of the intrusion-tolerant Group Membership Protocol (GMP). Donatelli et al. [27] solve the problem of Continuous Stochastic Logic (CSL) model checking in the context of Generalized Stochastic Petri Nets. Buchholz et al. [28] present a new framework for model checking techniques that can be applied to the general class of weighted automata. Haverkort [29] gives a short paper summarizing the formal modelling of timed system in practice.

In the design pattern formalization, formal methods have also been applied [30]–[33]. For the hybrid modelling, it is mostly used for verification [3], [13]–[16]. Recently, Tichakorn [34] proposed a subclass of hybrid automata for a class of hybrid control systems in which certain control actions occur roughly periodically and applied it to verify the safety of an autonomous vehicle. However, the intent there is verification, than a design methodology as in our work.

VII. CONCLUSION

In this paper, we have proposed a lease based design pattern to guarantee PTE safety rules in wireless CPS, as part of the effort to address challenges arising from poor reliability of wireless communication on CPS’ mission/life criticality. We derived a set of closed-form constraints, and proved that as long as system parameters are configured to satisfy these constraints, PTE safety rules are guaranteed under arbitrary wireless communication faults. Furthermore, we developed hybrid modeling approaches to describe the design patterns, and developed a formal methodology to elaborate the design pattern into specific designs that provide PTE safety guarantees. Our case study on laser tracheotomy wireless CPS validates the proposed design methodology.

ACKNOWLEDGEMENT

The research project related to this paper in Hong Kong Polytechnic University (HKPolyU) is supported in part by Hong Kong RGC General Research Fund (GRF) PolyU 5245/09E, RGC Early Career Scheme (ECS) PolyU 5328/12E, The HKPolyU Internal Competitive Research Grant (DA) A-PJ68, HKPolyU Newly Recruited Junior Academic Staff Grant A-PJ80, HK PolyU Fund for GRF Project Rated 3.5 (DA) grant A-PK46 (RGC No. PolyU 5300/10E), A-PL82 (RGC No. PolyU 5285/11E), and Department of Computing.
start up fund. Lei Bu is supported by the National Natural Science Foundation of China (No.91118007, No.61100036) and by the National 863 High-Tech Programm of China (No.2011AA010103, No.2012A011205). Rong Zheng is supported in part by the DAAD/RGC Germany/HK Joint Research Scheme G_HK022/12. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of sponsors.

The authors thank anonymous reviewers for their efforts/advice on improving this paper.

REFERENCES


