Guaranteeing Proper-Temporal-Embedding Safety Rules in Wireless CPS: A Hybrid Formal Modeling Approach

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" shutoff 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 $t1$ and $t2$ in Fig. 1 (e.g., $t1$ means that only after the oxygen ventilator has paused for $t1$ can laser start emission, otherwise the patient's trachea may still have high enough oxygen concentration to catch fire; note this "pause $t1$ 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 $t3$ and $t4$ in Fig. 1, must each be upper bounded by a constant (e.g., the ventilator pause duration $t3$ must be upper bounded, for otherwise the patient may suffocate to death). Modeling these sequenced CPS operations constitute *design patterns*.

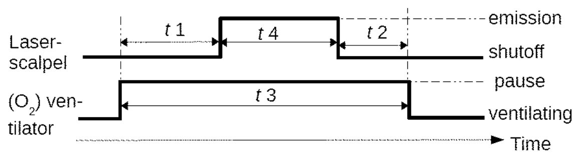


Fig. 1. Proper-Temporal-Embedding Example

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

* Dept. of Computing, The Hong Kong Polytechnic Univ., Hong Kong S.A.R.
† State Key Lab for Novel Software Technology, Dept. of Computer Sci. and Tech., Nanjing Univ., China.
‡ Dept. of Computing and Software, McMaster Univ., Canada.
** Dept. of Computer Science, TU Darmstadt, Germany.
Email: csqwang@comp.polyu.edu.hk

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 designs[1].
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, inv, F, E, g, R, L, syn, Φ_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 $(H_{vent}(t))$, i.e. it contains only one data state variable: $H_{vent}(t)$, which is the height of the ventilator cylinder at time t.

[1]Note a key difference between distributed computer systems and wireless (hence distributed) CPS lies in the continuous variables, which represent physical world states and are not fully controllable by the computer commands. For example, in surgery, the surgeon is a human-in-the-loop entity of the holistic CPS, where his/her actions are not fully controlled by computer commands. Similarly, the patient's precise blood oxygen level is only partially controllable by computer control as many cyber and physical factors can affect it, even including the patient's emotion.

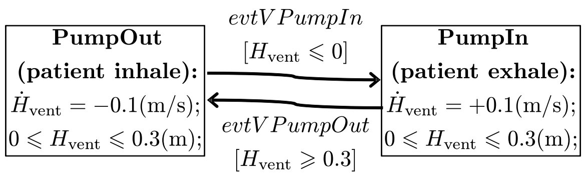

$evtVPumpIn$
$[H_{\text{vent}} \leqslant 0]$
PumpOut
(patient inhale):
$\dot{H}_{\text{vent}} = -0.1(\text{m/s});$
$0 \leqslant H_{\text{vent}} \leqslant 0.3(\text{m});$
PumpIn
(patient exhale):
$\dot{H}_{\text{vent}} = +0.1(\text{m/s});$
$0 \leqslant H_{\text{vent}} \leqslant 0.3(\text{m});$
$evtVPumpOut$
$[H_{\text{vent}} \geqslant 0.3]$

Fig. 2. Hybrid Automaton A'_{vent} of a Stand-Alone Ventilator. $H_{\text{vent}}(t)$ is the data state variable denoting the ventilator's cylinder height at time t. "PumpOut" is the only initial location.

2. A finite set V of vertices called *locations*. The *state* of A (at time t) is a tuple $\phi(t) = (\ell(t), \vec{x}(t))$ of two variables of time t: the aforementioned data state variables vector $\vec{x}(t)$, and the *location counter* $\ell(t) \in V$, 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 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 $\vec{x}(t) \in \mathsf{inv}(v)$. In the example of Fig. 2, in location PumpOut, the invariant is that the ventilator cylinder height $H_{vent}(t)$ stays in the range $0 < H_{vent}(t) \leq 0.3$(m).

4. A set of *flow maps* $F = \{f_v | f_v : \mathbb{R}^n \mapsto \mathbb{R}^n, \forall v \in V\}$, 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: $\dot{H}_{vent}(t) = -0.1$(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 $\mathsf{src}(e)$; while the destination location as $\mathsf{des}(e)$. An edge $e = (v, v')$ specifies the possible *discrete dynamics* of A's state: it can switch from $\ell(t) = v$ to $\ell(t^+) = v'$. 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* $g(e) \subseteq \mathsf{inv}(\mathsf{src}(e))$. 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. $H_{vent}(t) = 0$.

7. A finite set of *reset functions* $R = \{r_e | r_e : \mathsf{inv}(\mathsf{src}(e)) \mapsto 2^{\mathsf{inv}(\mathsf{des}(e))}, \forall e \in E\}$. When the A's state switches from $\ell(t) = \mathsf{src}(e)$ to $\ell(t^+) = \mathsf{des}(e)$ 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 $((H_{vent}(t))$ 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-*

nization labeling function syn that assigns to each edge $e \in E$ a synchronization label $\mathsf{syn}(e) \in L$. A synchronization label consists of a *root* and a *prefix*, which respectively represent a *event* and the *role* of the hybrid automaton for that event.

When entity ξ_1 (whose hybrid automaton is A_1) sends an event l to entity ξ_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, $!l$, $?l$, $??l$ 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 *external*; otherwise, the corresponding synchronization labels are *internal*. 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 $evtVPumpIn$ happens; in the other way around, event $evtVPumpOut$ happens. The ! prefix to $evtVPumpIn$ and $evtVPumpOut$ in the figure indicates the events are broadcasted. If there are other hybrid automata in the system, some transitions may be triggered on receiving these events, the corresponding transitions are labeled with $?evtVPumpIn$ or $?evtVPumpOut$. In case the reception of events are via unreliable (e.g. wireless) communication links, the corresponding labels should be $??evtVPumpIn$ or $??evtVPumpOut$.

9. A set of *possible initial states* $\Phi_0 \subseteq \{(v, \vec{s}) \in V \times \mathbb{R}^n | v \in V, \vec{s} \in \mathsf{inv}(v)\}$. We also call Φ_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 $H_{vent}(0) \in [0, 0.3]$(m).

B. System and Fault Model

A *hybrid system* $\mathcal{H}$ is a collection of *hybrid automata* (each is called a *member hybrid automaton* of $\mathcal{H}$), 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* ξ_0 and N (*in this paper, we require* $N \geq 2$) *remote entities* ξ_1, ξ_2, ..., ξ_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 $\mathcal{H} = \{A_i| \ (i = 0, 1, \ldots, N)\}$ describes a wireless CPS. The hybrid automaton A_i describes wireless CPS member entity ξ_i. The synchronization labels/functions describe the communication relationships between these hybrid automata.

We assume that for each hybrid automaton $A_i = (\vec{x}_i(t), V_i, \mathsf{inv}_i, F_i, E_i, g_i, r_i, L_i, \mathsf{syn}_i, \Phi_{0,i})$ (where $i = 1 \sim N$), V_i is partitioned into two subsets: V_i^{safe} and V_i^{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 ξ_0).

There are two types of PTE safety rules, namely:

PTE Safety Rule 1 (Bounded Dwelling): Each entity ξ_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 ξ_i and ξ_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 ξ_i dwells in safe-locations at time t (i.e. A_i's location counter $\ell_i(t) \in V_i^{\text{safe}}$), then throughout interval $[t, t + T^{\mathsf{min}}_{\text{risky}:i \to j}]$, ξ_j dwells in safe-locations, where constant $T^{\mathsf{min}}_{\text{risky}:i \to j}$ is the ξ_i *to* ξ_j *enter-risky safeguard interval*.

p2. Whenever ξ_j dwells in risky-locations, ξ_i dwells in risky-locations.

p3. If ξ_j dwells in risky-locations at time t, then throughout interval $[t, t + T^{\mathsf{min}}_{\text{safe}:j \to i}]$, ξ_i dwells in risky-locations, where constant $T^{\mathsf{min}}_{\text{safe}:j \to i}$ is the ξ_j *to* ξ_i *exit-risky safeguard interval*.

Intuitively, Property p2 implies that whenever entity ξ_j is in risky-locations, then entity ξ_i is already in risky-locations. Property p1 and p3, in addition, specify the safeguard interval requirements that ξ_i and ξ_j enter/exit respective risky-locations. Specifically, Property p1 implies that before ξ_j enters its risky-locations, ξ_i should have already been in risky-locations for at least $T^{\min}_{\text{risky}:i\to j}$. Property p3 implies that after ξ_j exits its risky-locations (i.e. returns to safe-locations), ξ_i must stay in risky-locations for at least $T^{\min}_{\text{safe}:j\to i}$.

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 ξ_1, ξ_2, ..., ξ_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

$$\xi_1 < \xi_2 < \ldots < \xi_N. \tag{1}$$

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 ξ_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 ξ_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 ξ_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 ξ_N.

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

We require that every entity ξ_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 ξ_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 to rigorously eliminate zeno, better add a positive min dwelling requirement in Fall-Back.

Supervisor has not yet aborted/cancelled the lease, ξ_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 ξ_0 shall start from a "Fall-Back" location. Whenever the Initializer ξ_N requests leasing itself to enter risky-locations, the Supervisor shall lease Participants $\xi_1, \xi_2, \ldots, \xi_{N-1}$ 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 ξ_N's lease request to enter risky-location. The Initializer ξ_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 blood oxygen level SpO_2 is higher than threshold Θ_{SpO_2}), Supervisor ξ_0 can abort leases. Lease cancellations/abortions are conducted in the reverse PTE order.

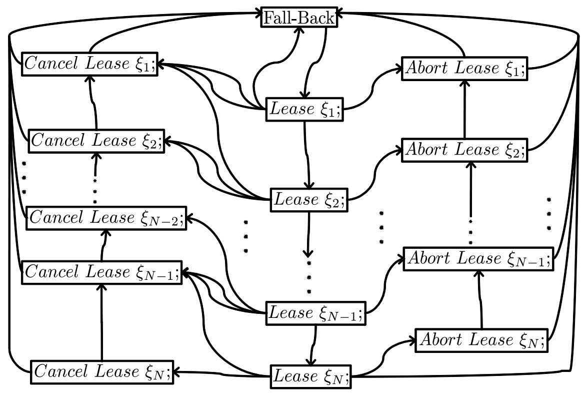


Fig. 3. Sketch of Hybrid Automaton A_{supvsr}, the Design Pattern for Supervisor.

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

1. A_{supvsr}'s location set V_{supvsr} include the following locations: "Fall-Back", "Lease ξ_i" (where $i = 1 \sim N$), "Cancel Lease ξ_i" (where $i = 1 \sim N$), and "Abort Lease ξ_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 $evt\xi_N To\xi_0 Req$ is received (which is sent by the Initializer requesting for entering risky-locations, see the descriptions for A_{initzr} in the following), and the Supervisor has been continuously dwelling in "Fall-Back" for at least $T^{\min}_{\text{fb},0}$, and the application dependent proposition $ApprovalCondition$ holds, then the Supervisor transits to location "Lease ξ_1". Along this transition[2], the Supervisor sends out event $evt\xi_0 To\xi_1 LeaseReq$, requesting leasing Participant ξ_1.
4. When in location "Lease ξ_i" (where $i = 1 \sim N-1$), the behavior of Supervisor can be described by Fig. 4 (a).
5. When in location "Lease ξ_N", the behavior of Supervisor can be described by Fig. 4 (b).
6. When in location "Cancel Lease ξ_i" (where $i = 1 \sim N$), the behavior of Supervisor can be described by Fig. 4 (c).
7. When in location "Abort Lease ξ_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 ξ_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 ξ_0, ξ_N enters risky-locations. The dwelling in risky-locations can be cancelled by ξ_N or aborted by ξ_0 at any time; otherwise, ξ_N returns to "Fall-Back" when the lease expires.

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

1. A_{initzr}'s location set V_{initzr} include the following locations: "Fall-Back", "Requesting", "Entering", "Risky Core", "Exiting 1", and "Exiting 2". $V^{\text{risky}}_{\text{initzr}}$ include location "Risky Core" and "Exiting 1"; all other locations belong to $V^{\text{safe}}_{\text{initzr}}$.
2. Initially, the Initializer ξ_N dwells in location "Fall-Back"; and all data state variables initial values are zero.
3. When in location "Fall-Back", the Initializer ξ_N can send event $evt\xi_N To\xi_0 Req$ and transit to "Requesting" at any time.
4. When in location "Requesting", the Initializer ξ_N can send event $evt\xi_N To\xi_0 Cancel$ and transit back to "Fall-Back" at any time. Secondly, if ξ_N dwells continuously in "Requesting" for $T^{\max}_{\text{req},N}$, it will automatically transit back to "Fall-Back". Thirdly, if event ~~$evt\xi_0 To\xi_N Approve$~~ is received, ξ_N transits to "Entering".
5. When in location "Entering", the Initializer ξ_N can send event $evt\xi_N To\xi_0 Cancel$ and transit to "Exiting 2". Secondly, if $evt\xi_0 To\xi_N Abort$ is received, ξ_N also transits to "Exiting 2". Thirdly, if ξ_N dwells continuously in "Entering" for $T^{\max}_{\text{enter},N}$, it transits to "Risky Core".

[2] In fact, this "transition" includes two consecutive transitions, the first one is on receiving event $evt\xi_N To\xi_0 Req$, Supervisor enters an *intermediate location* of 0 dwelling time; and then transit from this intermediate location to "Lease ξ_1" and send out $evt\xi_0 To\xi_1 LeaseReq$. For narrative simplicity, in the following, such intermediate locations between two consecutive events are not elaborated.

$evt\xi_0To\xi_NLeaseApprove$

Set timer $t_{LS1} \leftarrow 0$ iff $i = 1$.

evt

to rigorously eliminate zeno, better add a positive min dwelling requirement in Fall-Back.

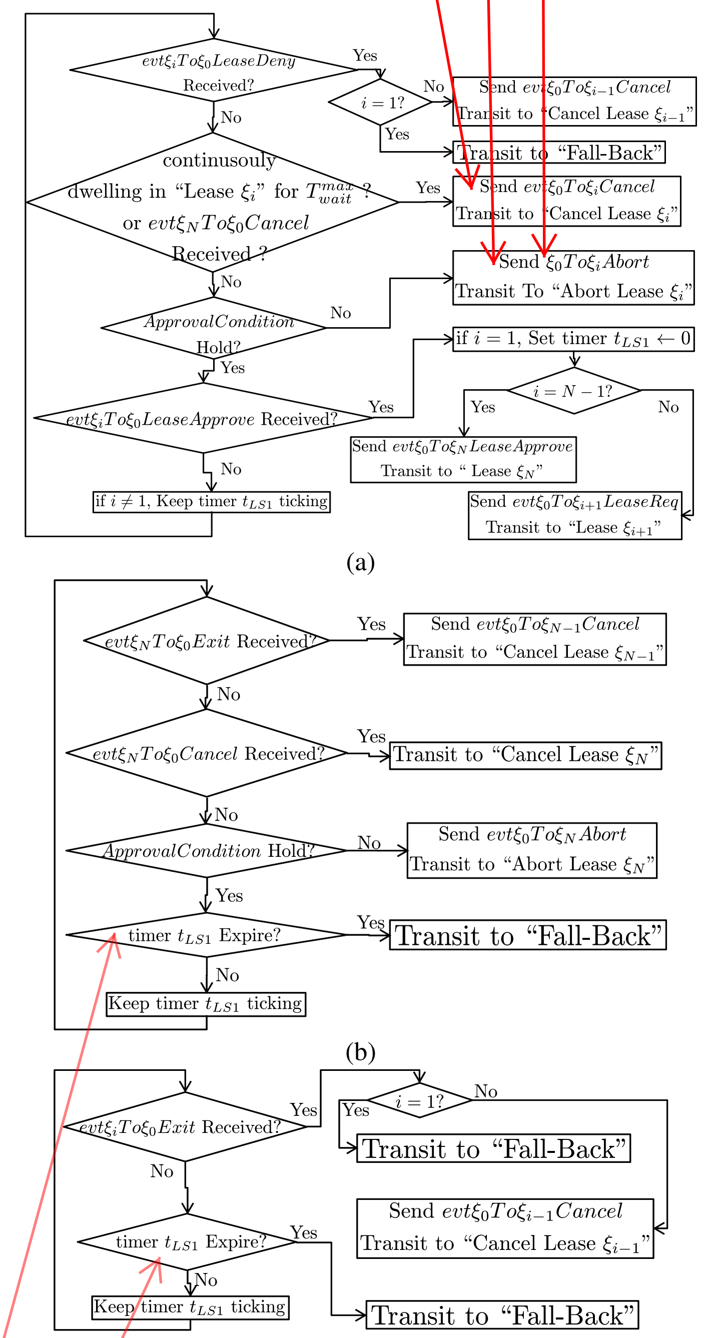

$evt\xi_iTo\xi_0LeaseDeny$ Received?
Yes
No
$i = 1$?
No
Yes
Send $evt\xi_0To\xi_{i-1}Cancel$ Transit to "Cancel Lease ξ_{i-1}"
Transit to "Fall-Back"
continusouly dwelling in "Lease ξ_i" for T_{wait}^{max} ? or $evt\xi_NTo\xi_0Cancel$ Received ?
Yes
Send $evt\xi_0To\xi_iCancel$ Transit to "Cancel Lease ξ_i"
No
Send $\xi_0To\xi_iAbort$ Transit To "Abort Lease ξ_i"
$ApprovalCondition$ Hold?
No
Yes
if $i = 1$, Set timer $t_{LS1} \leftarrow 0$
$i = N-1$?
Yes
No
$evt\xi_iTo\xi_0LeaseApprove$ Received?
Yes
Send $evt\xi_0To\xi_NLeaseApprove$ Transit to " Lease ξ_N"
No
Send $evt\xi_0To\xi_{i+1}LeaseReq$ Transit to "Lease ξ_{i+1}"
if $i \neq 1$, Keep timer t_{LS1} ticking
(a)
$evt\xi_NTo\xi_0Exit$ Received?
Yes
Send $evt\xi_0To\xi_{N-1}Cancel$ Transit to "Cancel Lease ξ_{N-1}"
No
$evt\xi_NTo\xi_0Cancel$ Received?
Yes
Transit to "Cancel Lease ξ_N"
No
$ApprovalCondition$ Hold?
No
Send $evt\xi_0To\xi_NAbort$ Transit to "Abort Lease ξ_N"
Yes
timer t_{LS1} Expire?
Yes
Transit to "Fall-Back"
No
Keep timer t_{LS1} ticking
(b)
$evt\xi_iTo\xi_0Exit$ Received?
Yes
$i = 1$?
No
Yes
Transit to "Fall-Back"
Send $evt\xi_0To\xi_{i-1}Cancel$ Transit to "Cancel Lease ξ_{i-1}"
No
timer t_{LS1} Expire?
Yes
No
Keep timer t_{LS1} ticking
Transit to "Fall-Back"
(c)

Fig. 4. Flow block diagram at location (a) "Lease ξ_i" ($i = 1 \sim N-1$); (b) "Lease ξ_N"; (c) "Cancel Lease ξ_i" ($i = 1 \sim N$).

i.e. $t_{LS1} \geq T^{max}_{LS1}$.

6. When in location "Risky Core", the Initializer ξ_N can send event $evt\xi_NTo\xi_0Cancel$ and transit to "Exiting 1". Secondly, if $evt\xi_0To\xi_NAbort$ is received, ξ_N also transits to "Exiting 1". Thirdly, if ξ_N dwells continuously in "Risky Core" for $T^{\max}_{\text{run},N}$, it also transits to "Exiting 1".

7. When in location "Exiting 1" or "Exiting 2", the Initializer ξ_N must continuously dwell in the location for $T_{\text{exit},N}$, and then transit to "Fall-Back".

and send $evt\xi_NTo\xi_0Exit$

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

Participant:

Conceptually, a Participant ξ_i ($i = 1 \sim N-1$) shall start from a "Fall-Back" location. Upon receiving lease request from the Supervisor ξ_0, and if the lease is approved, ξ_i enters risky-locations. The dwelling in risky-locations can be cancelled by the Initializer ξ_N or aborted by the Supervisor ξ_0 at any time; otherwise, ξ_i returns to "Fall-Back" when the lease expires.

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

1. A_{ptcpnt}'s location set V_{ptcpnt} include the following locations: "Fall-Back", "L0", "Entering", "Risky Core", "Exiting 1", and "Exiting 2". $V^{\text{risky}}_{\text{ptcpnt}}$ include location "Risky Core" and "Exiting 1"; all other locations belong to $V^{\text{safe}}_{\text{ptcpnt}}$.
2. Initially, Participant ξ_i dwells in location "Fall-Back"; and all data state variables initial values are zero.
3. When in location "Fall-Back", upon receiving event $evt\xi_0To\xi_iLeaseReq$, ξ_i transits to a temporary location "L0".
4. When in "L0", if an application dependent proposition $ParticipationCondition$ sustains, ξ_i sends event $evt\xi_iTo\xi_0LeaseApprove$ and transits to "Entering"; otherwise, ξ_i sends event $evt\xi_iTo\xi_0LeaseDeny$ and transits back to "Fall-Back".

i

5. When in location "Entering", if event $evt\xi_0To\xi_iCancel$ or $evt\xi_0To\xi_NAbort$ is received, ξ_i transits to "Exiting 2". Otherwise, if ξ_i dwells continuously in "Entering" for $T^{\max}_{\text{enter},i}$, it transits to "Risky Core".
6. When in location "Risky Core", if event $evt\xi_0To\xi_iCancel$ or $evt\xi_0To\xi_NAbort$ is received, ξ_i transits to "Exiting 1". Otherwise, if ξ_i dwells continuously in "Risky Core" for $T^{\max}_{\text{run},i}$, it also transits to "Exiting 1".
7. When in location "Exiting 1" or "Exiting 2", Participant ξ_i must continuously dwell in the location for $T_{\text{exit},i}$, and then transit to "Fall-Back".

and send $evt\xi_iTo\xi_0Exit$

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_{supvsr}, A_{initzr}, and A_{ptcpnt}, PTE safety rules are guaranteed

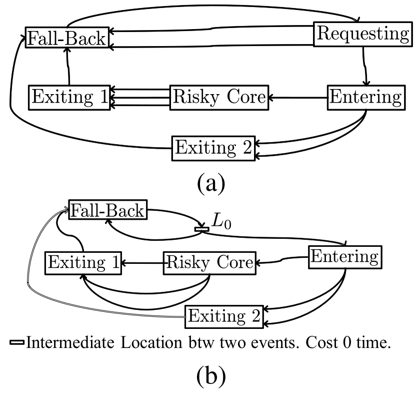


Fig. 5. (a) Sketch of Hybrid Automaton A_{initzr}, the Design Pattern for Initializer; (b) Sketch of Hybrid Automaton $A_{\text{ptcpnt},i}$, the Design Pattern for the ith Participant.

despite any communication faults. Specifically, we have the following result.

Theorem 1 (Design Pattern Validity): If a hybrid system $\mathcal{H}$ of ξ_0 as "Supervisor" (i.e. behaves per A_{supvsr}), ξ_N ($N \geq 2$) as "Initializer" (i.e. behaves per A_{initzr}), and ξ_i ($i = 1, 2, \ldots, N-1$) as "Participants" (i.e. behaves per $A_{\text{ptcpnt},i}$) satisfies conditions c1 $\sim$ c7:

c1. All configuration time constants ($T^{\max}_{\text{wait}}$, $T^{\min}_{\text{fb},0}$, $T^{\max}_{\text{LS1}}$, $T^{\max}_{\text{req},N}$, $T^{\max}_{\text{enter},i}$, $T^{\max}_{\text{run},i}$, $T_{\text{exit},i}$, where $i = 1 \sim N$) are positive.

c2. $T^{\max}_{\text{LS1}} \overset{\text{def}}{=} T^{\max}_{\text{enter},1} + T^{\max}_{\text{run},1} + T_{\text{exit},1} > NT^{\max}_{\text{wait}}$.

c3. $(N-1)T^{\max}_{\text{wait}} < T^{\max}_{\text{req},N} < T^{\max}_{\text{LS1}}$.

c4. $\forall i \in \{1, 2, \ldots, N\}$, there is
$$(i-1)T^{\max}_{\text{wait}} + T^{\max}_{\text{enter},i} + T^{\max}_{\text{run},i} + T_{\text{exit},i} \leq T^{\max}_{\text{LS1}}.$$

c5. $\forall i \in \{1, 2, \ldots, N-1\}$, there is
$$T^{\max}_{\text{enter},i} + T^{\min}_{\text{risky}:i\to i+1} < T^{\max}_{\text{enter},i+1}.$$

c6. $\forall i \in \{1, 2, \ldots, N-1\}$, there is
$$\begin{aligned} T^{\max}_{\text{enter},i} + T^{\max}_{\text{run},i} \quad > \quad & T^{\max}_{\text{wait}} + T^{\max}_{\text{enter},i+1} + T^{\max}_{\text{run},i+1} \\ & + T_{\text{exit},i+1}. \end{aligned}$$

c7. $\forall i \in \{1, 2, \ldots, N-1\}$, there is $T_{\text{exit},i} > T^{\min}_{\text{safe}:i+1\to i}$.

Then $\mathcal{H}$ satisfies PTE safety rules. Specifically, every entity's continuous dwelling time in risky-locations is upper bounded by $T^{\max}_{\text{wait}} + T^{\max}_{\text{LS1}}$, and the PTE full ordering of $\xi_1 < \xi_2 < \ldots < \xi_N$ 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 $T^{\max}_{\text{wait}} + T^{\max}_{\text{LS1}}$ every time $evt\xi_0To\xi_1LeaseReq$ 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 $evt\xi_0To\xi_1LeaseReq$ events (or the last such event and time ∞), any entity can only dwell in the risky-locations for once.

Third, due to Conditions c1 $\sim$ c7, for each ξ_i and ξ_{i+1} ($i = 1 \sim N-1$), the aforementioned single dwelling intervals of ξ_i and ξ_{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_{supvsr}, A_{initzr}, and A_{ptcpnt} into specific PTE wireless CPS designs.

The intuition is that every location v of A_{supvsr}, A_{initzr}, and A_{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_{supvsr}, A_{initzr}, and A_{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_{supvsr}, A_{initzr}, and A_{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 $H_{vent}(t)$, 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.

Unless explicitly denoted, the rest of the paper assumes every hybrid automaton to be time-block-free and non-zeno[3].

We now define *hybrid automata independence*.

Definition 2 (Hybrid Automata Independence): Given hybrid automata $A = (\vec{x}(t), V, \mathsf{inv}, F, E, g, R, L, \mathsf{syn}, \Phi_0)$ and

[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.

$A' = (\vec{x}'(t),\ V',\ \mathsf{inv}',\ F',\ E',\ g',\ R',\ L',\ \mathsf{syn}',\ \Phi_0')$, we say "*$A$ and A' are independent*" iff

1. $\mathsf{elements}(\vec{x}(t)) \cap \mathsf{elements}(\vec{x}'(t)) = \varnothing$;
2. and $V \cap V' = \varnothing$;
3. and $L \cap L' = \varnothing$.

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

We further define *simple* hybrid automaton.

Definition 3 (Simple Hybrid Automaton): A hybrid automaton $A = (\vec{x}(t),\ V,\ \mathsf{inv},\ F,\ E,\ g,\ R,\ L,\ \mathsf{syn},\ \Phi_0)$ is *simple* iff

1. $\forall v_1, v_2 \in V$, $\mathsf{inv}(v_1) = \mathsf{inv}(v_2)$.
2. $\forall v \in \Phi_0|_V \cdot \forall \vec{s} \in \mathsf{inv}(v) \cdot (v, \vec{s}) \in \Phi_0$, where $\Phi_0|_V$ means Φ_0's projection on V.
3. $\forall v \in \Phi_0|_V \cdot (v, \mathbf{0}) \in \Phi_0$, where $\mathbf{0}$ 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 $A = (\vec{x}(t),\ V,\ \mathsf{inv},\ F,\ E,\ g,\ R,\ L,\ \mathsf{syn},\ \Phi_0)$, location $v \in V$, and a simple hybrid automaton $A' = (\vec{x}'(t),\ V',\ \mathsf{inv}',\ F',\ E',\ g',\ R',\ L',\ \mathsf{syn}',\ \Phi_0')$ 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 $A'' = (\vec{x}''(t),\ V'',\ \mathsf{inv}'',\ F'',\ E'',\ g'',\ R'',\ L'',\ \mathsf{syn}'',\ \Phi_0'')$, according to the following intuitions.

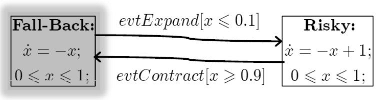
1. Location v of hybrid automaton A is replaced by simple hybrid automaton A'.
2. All former ingress edges to v in A become ingress edges to A' (A''s initial locations to be more specific).
3. All former egress edges from v in A become egress edges from A'.
4. When in A', the data state variables $\vec{x}(t)$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

We denote A'', the elaboration of A at v with A', as

$$A'' = \mathcal{E}(A, v, A').$$

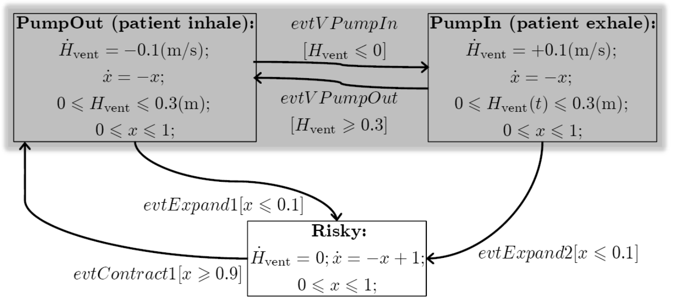
The rigorous formal description on atomic elaboration of hybrid automaton is provided in [19]

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



Fall-Back:
$\dot{x} = -x$;
$0 \leqslant x \leqslant 1$;
$evtExpand[x \leqslant 0.1]$
$evtContract[x \geqslant 0.9]$
Risky:
$\dot{x} = -x + 1$;
$0 \leqslant x \leqslant 1$;

(a)



PumpOut (patient inhale):
$\dot{H}_{\text{vent}} = -0.1$(m/s);
$\dot{x} = -x$;
$0 \leqslant H_{\text{vent}} \leqslant 0.3$(m);
$0 \leqslant x \leqslant 1$;
$evtVPumpIn$
$[H_{\text{vent}} \leqslant 0]$
$evtVPumpOut$
$[H_{\text{vent}} \geqslant 0.3]$
PumpIn (patient exhale):
$\dot{H}_{\text{vent}} = +0.1$(m/s);
$\dot{x} = -x$;
$0 \leqslant H_{\text{vent}}(t) \leqslant 0.3$(m);
$0 \leqslant x \leqslant 1$;
$evtExpand1[x \leqslant 0.1]$
Risky:
$\dot{H}_{\text{vent}} = 0; \dot{x} = -x + 1$;
$0 \leqslant x \leqslant 1$;
$evtContract1[x \geqslant 0.9]$
$evtExpand2[x \leqslant 0.1]$

(b)

Fig. 6. Atomic Elaboration Example (compare the shaded areas in (a) and (b)). (a) Hybrid Automaton A, which has one data state variable x; the shaded location is to be elaborated. (b) Hybrid Automaton A'', which is the atomic elaboration of A (see (a)) at location "Fall-Back" with simple hybrid automaton A'_{vent} (see Fig. 2); note there is no edge from "Risky" to "PumpIn" because "PumpIn" is not an initial location of A'_{vent}.

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, ..., A_k are mutually independent, then we can carry out "(parallel) elaboration of A at v_1, v_2, ..., v_k with A_1, A_2, ..., A_k", denoted as

$$\begin{aligned} &\mathcal{E}(A, (v_1, v_2, \ldots, v_k), (A_1, A_2, \ldots, A_k)) \\ \overset{\mathsf{def}}{=}\ &\underbrace{\mathcal{E}(\ldots\mathcal{E}(\mathcal{E}(}_{\text{repeat } k \text{ times}} A, v_1, A_1), v_2, A_2)\ldots), v_k, A_k). \end{aligned}$$

Denote $A' = \mathcal{E}(A,\ (v_1, v_2, \ldots, v_k),\ (A_1, A_2, \ldots, A_k))$, we also say "*$A'$ elaborates A at v_1, v_2, ..., v_k with A_1, A_2, ..., A_k respectively*".

Intuitively, parallel elaboration of A at v_1, v_2, ..., v_k with A_1, A_2, ..., 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 $\mathcal{H}'$, has its member hybrid automata respectively elaborating the Supervisor, Initializer, and Participant design pattern hybrid automata (i.e. A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$), then the design $\mathcal{H}'$ 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 $\mathcal{H}'$ 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 $v_1^0, v_2^0, \ldots, v_{k_0}^0 \in V_{\text{supvsr}}$, 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'_0 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 $v_1^i, v_2^i, \ldots, v_{k_i}^i \in V_{\text{ptcpnt},i}$, and simple hybrid automata $A_1^i, A_2^i, \ldots, A_{k_i}^i$, such that $A_{\text{ptcpnt},i}$ and A_j^i ($j = 1 \sim k_i$) are independent, and A'_i elaborates $A_{\text{ptcpnt},i}^i$ at $v_1^i, v_2^i, \ldots, v_{k_i}^i$ with $A_1^i, A_2^i, \ldots, A_{k_i}^i$ respectively;

3. There are distinct locations $v_1^N, v_2^N, \ldots, v_{k_N}^N \in V_{\text{initzr}}$, and simple hybrid automata $A_1^N, A_2^N, \ldots, A_{k_N}^N$, such that A_{initzr} and A_j^N ($j = 1 \sim k_N$) are independent, and A'_N elaborates A_{initzr} at $v_1^N, v_2^N, \ldots, v_{k_N}^N$ with $A_1^N, A_2^N, \ldots, A_{k_N}^N$ respectively;

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

5. Condition c1 $\sim$ c7 of Theorem 1 sustain;

where V_{supvsr}, $V_{\text{ptcpnt},i}$, and V_{initzr} are respectively A_{supvsr}, $A_{\text{ptcpnt},i}$, and A_{initzr}'s location sets, then $\mathcal{H}'$ 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 $\mathcal{H}'$ that violates PTE safety rules. According to the methodology we elaborate hybrid automata, $\phi'(t)$ corresponds to an execution trace $\phi(t)$ of $\mathcal{H}$ (the hybrid system of A_{supvsr}, $A_{\text{ptcpnt},i}$ ($i = 1, 2, \ldots, N-1$), 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 ξ_1 and the laser-scalpel as entity ξ_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

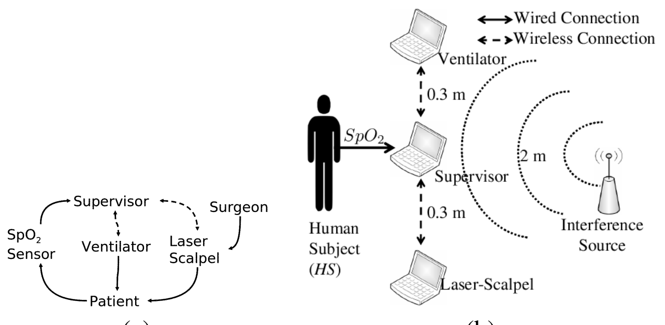


Fig. 7. (a) Laser Tracheotomy Wireless CPS Layout, figure quoted from [3]. The SpO_2 sensor (measuring the Patient's blood oxygen level) is wired to the Supervisor computer, forming entity ξ_0. The Laser-Scalpel (operated by the Surgeon) takes the role of Initializer entity, ξ_2. The ventilator is the Participant entity, ξ_1. (b) Emulation Layout

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 ξ_0; the (surgeon operated) laser-scalpel plays the role of Initializer, hence entity ξ_2; and the ventilator plays the role of Participant 1, hence entity ξ_1.

Next, we design the hybrid automata for the laser tracheotomy supervisor, laser-scalpel, and ventilator by respectively elaborating A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},1}$.

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'_{vent} of Fig. 2. The stand-alone design of A'_{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 $A_{\text{ptcpnt},i}$ (see Section IV-A-Participant; also see Fig. 5 (b) for the sketch of the hybrid automaton) at location "Fall-Back" with A'_{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), $T^{\mathsf{min}}_{\mathrm{fb},0} = 13$s, $T^{\mathsf{max}}_{\mathrm{wait}} = 3$s. For the Initializer (i.e. the laser-scalpel), $T^{\mathsf{max}}_{\mathrm{req},2} = 5$s, $T^{\mathsf{max}}_{\mathrm{enter},2} = 10$s, $T^{\mathsf{max}}_{\mathrm{run},2} = 20$s, $T_{\mathrm{exit},2} = 1.5$s. For the Participant 1 (i.e. the ventilator), $T^{\mathsf{max}}_{\mathrm{enter},1} = 3$s, $T^{\mathsf{max}}_{\mathrm{run},1} = 35$s, $T_{\mathrm{exit},1} = 6$s. The PTE enter-risky/exit-risky safeguard intervals are $T^{\mathsf{min}}_{\mathrm{risky}:1\to 2} = 3$s and $T^{\mathsf{min}}_{\mathrm{safe}:2\to 1} = 1.5$s.

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 $evt\xi_2 To\xi_0 Req$, 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_{on} (following exponential distribution) whenever the laser-scalpel enters "Fall-Back". When in "Fall-Back" and T_{on} sets off, the (emulated) surgeon requests to emit laser. Timer T_{on} is destroyed whenever the laser-scalpel leaves "Fall-Back" location.

The second kind is the Initializer event $evt\xi_2 To\xi_0 Cancel$, 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_{off} (following exponential distribution) whenever the laser-scalpel is emitting laser. When emitting laser and T_{off} sets off, the (emulated) surgeon requests to cancel laser emission. Timer T_{off} is destroyed whenever the laser-scalpel returns to "Fall-Back" location.

The third kind is the Supervisor event $evt\xi_0 To\xi_i Abort$ ($i = 1 \sim N$), triggered when the supervisor is in "Lease ξ_i" location and $ApprovalCondition$ 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 $ApprovalCondition$ is that the oximeter reading $SpO_2(t) > \Theta_{SpO_2}$, where Θ_{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.

TABLE I. PTE SAFETY RULE VIOLATION (FAILURE) STATISTICS OF EMULATION TRIALS

Trial Mode	$E(T_{\mathrm{off}})$ (sec)	# of Laser Emissions	# of Failures	# of $evtToStop$
with Lease	18	19	0	5
without Lease	18	11	4	0
with Lease	6	19	0	3
without Lease	6	12	3	0

1. Each trial lasts 30 minutes, and is under constant WiFi interference.
2. For each trial, the expectation $E(T_{\mathrm{on}}) \equiv 30$(sec).
3. $evtToStop$ 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.

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 $\leqslant 1$ minute is always safe);
2. Ventilator pause duration must always properly-temporally-embedding laser emission duration, with entering/exiting safeguard interval of $T^{\mathsf{min}}_{\mathrm{risky}:1\to 2} = 3$(s) and $T^{\mathsf{min}}_{\mathrm{safe}:2\to 1} = 1.5$(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_{on} and T_{off}, both are random numbers following exponential distribution. The expectation of T_{on} is 30(s). The expectations of T_{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 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_{off} is set to 1 hour). In this case, only 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: $evt\xi_0 To\xi_2 Abort$, followed by $evt\xi_2 To\xi_0 Exit$, and followed by $evt\xi_0 To\xi_1 Abort$. Losing any one of these events at the receiver end will cause PTE safety rules violation. For example, losing $evt\xi_2 To\xi_0 Exit$, 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 $T^{\text{max}}_{\text{run},2} = 20$s with or without surgeon’s request to cancel; and the ventilator resumes within the lease $T^{\text{max}}_{\text{run},1} = 35$s 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. $evt\xi_2 To\xi_0 Cancel$) 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 $T^{\text{max}}_{\text{run},1} = 35$s at the most, hence cannot suffocate the patient.

A third scenario involves the parameter configuration constraints. Suppose we set $T^{\text{max}}_{\text{enter},2} = T^{\text{max}}_{\text{enter},1} = 0$s (or any other value so that $T^{\text{max}}_{\text{enter},2} = T^{\text{max}}_{\text{enter},1}$), then because $T^{\text{min}}_{\text{risky}:1\to 2} = 3$s > 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 $T^{\text{min}}_{\text{risky}:1\to 2} = 3$s: 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 intensively 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.2012AA011205). Rong Zheng is supported in part by US NSF CNS-1117560, McMaster Startup fund, and the Canada NSERC Discovery grant. Neeraj Suri is supported in part by TUD CASED. This research is also 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

[1] L. Sha *et al.*, "Cyber-physical systems: A new frontier," *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*, 2009.

[2] PCAST, *Federal Networking and Information technology R&D (NITRD) Program Review*, 2007.

[3] T. Li *et al.*, "From offline toward real-time: A hybrid systems model checking and CPS co-design approach for medical device plug-and-play (MDPnP)," *Proc. of the ICCPS'12*, pp. 13–22, 2012.

[4] R. Poovendran *et al.*, "Special issue on Cyber-Physical Systems," *Proc. of IEEE*, vol. 100, no. 1, 2012.

[5] NITRD, *High-Confidence Medical Devices: Cyber-Physical Systems for 21st Century Health Care – A Research and Development Needs Report*, Feb. 2009.

[6] "Medical devices and medical systems - essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ice), part 1: General requirements and conceptual model," no. STAM F2761-2009, 2009.

[7] C. G. Gray *et al.*, "Leases: An efficient fault-tolerant mechanism for distributed file cache consistency," *Proc. of ACM SOSP'89*, 1989.

[8] C. A. Thekkath *et al.*, "Frangipani: a scalable distributed file system," *Proc. of ACM SOSP'97*, pp. 224–237, 1997.

[9] S. Annapureddy *et al.*, "Shark: scaling file servers via cooperative caching," *Proc. of the NSDI'05*, pp. 129–142, 2005.

[10] C. Kotselidis *et al.*, "Distm: A software transactional memory framework for clusters," *Proc. of the ICPP'08*, pp. 51–58, 2008.

[11] E. G. Boix *et al.*, "Context-aware leasing for mobile ad hoc networks," *3rd Workshop on OT4AmI co-located at ECOOP'07*, 2007.

[12] A. Adya *et al.*, "Centrifuge: Integrated lease management and partitioning for cloud services," *Proc. of the NSDI'10*, 2010.

[13] R. Alur *et al.*, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," *Hybrid Systems*, 1993.

[14] T. A. Henzinger *et al.*, "Hytech: The next generation," *Proc. of the RTSS'95*, pp. 56–65, 1995.

[15] R. Alur *et al.*, "Automatic symbolic verification of embedded systems," *IEEE Trans. on Software Engineering*, vol. 22, no. 3, pp. 181–201, 1996.

[16] M. Gribaudo, M. Gribaudo, A. Horvth, A. Bobbio, E. Tronci, E. Ciancamerla, and M. Minichino, "Fluid petri nets and hybrid model-checking: A comparative case study," *Reliability Engineering And System Safety*, vol. 81, pp. 239–257, 2003.

[17] Y. Wang *et al.*, "Wicop: Engineering wifi temporal white-spaces for safe operations of wireless body area networks in medical applications," *Proc. of the RTSS'11*, pp. 170 –179, 2011.

[18] Q. Wang *et al.*, "Building robust wireless LAN for industrial control with the DSSS-CDMA cell phone network paradigm," *IEEE Transactions on Mobile Computing*, vol. 6, no. 6, pp. 706–719, Jun. 2007.

[19] *Guaranteeing Proper-Temporal-Embedding Safety Rules in Wireless CPS: A Hybrid Formal Modeling Approach (Technical Report: Supplementary Materials)*. http://www.comp.polyu.edu.hk/~csqwang/research/appendix.html.

[20] C. Baier *et al.*, *Principles of Model Checking*. MIT Press, 2008.

[21] C. M. Townsend Jr. *et al.*, *Sabiston Textbook of Surgery: The Biological Basis of Modern Surgical Practice*, 19th ed. Elsevier Saunders, 2012.

[22] *Nonin 9843 oximeter/Co2 detector*. http://www.nonin.com.

[23] J. Yick *et al.*, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292 – 2330, 2008.

[24] X. Chen, H. Wang, and S. Ren, "DNScup: Strong cache consistency protocol for dns," *Proc. of the ICDCS'06*, pp. 40–48, 2006.

[25] I. Majzik, A. Bondavalli, S. Klapka, T. Madsen, and D. Iovino, "Formal methods in the evaluation of a safe driver-machine interface," *FORMS-FORMAT 2008*, 2008.

[26] H. V. Ramasamy, M. Cukier, and W. H. Sanders, "Formal specification and verification of a group membership protocol for an intrusion-tolerant group communication system," *Proc. of the PRDC '02*, pp. 9–18, 2002.

[27] S. Donatelli and J. Sproston, "CSL model checking for the GreatSPN tool," *In Proc. ISCIS 2004*, pp. 543–552, 2004.

[28] P. Buchholz and P. Kemper, "Model checking for a class of weighted automata," *Discrete Event Dynamic Systems*, vol. 20, no. 1, pp. 103–137, Mar. 2010.

[29] B. R. Haverkort, "Formal modeling and analysis of timed systems: Technology push or market pull?" *Formal Modeling and Analysis of Timed Systems*, pp. 18–24, 2011.

[30] D. Garlan, "The role of formal reusable frameworks," *SIGSOFT Softw. Eng. Notes*, vol. 15, no. 4, pp. 42–44, 1990.

[31] T. Mikkonen, "Formalizing design patterns," *Proc. of the ICSE '98*, pp. 115–124, 1998.

[32] S. Faheem *et al.*, "Designing verifiable and reusable data access layer using formal methods and design patterns," *Proc. of the ICCMS '09*, pp. 167–172, 2009.

[33] X. B. Li *et al.*, "Formal development of a washing machine controller by using formal design patterns," *Proceedings of the CEA'09*, pp. 127–132, 2009.

[34] W. Tichakorn, "Formal methods for design and verification of embedded control systems : application to an autonomous vehicle," *Dissertation (Ph.D.),California Institute of Technology*, 2010.