A Lease based Hybrid Design Pattern for Proper-Temporal-Embedding of Wireless CPS Interlocking (Supplementary File)

Feng Tan, Yufei Wang, Qixin Wang, Lei Bu, and 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 requires safety guarantees despite communication faults. This paper focuses on one important set of such safety rules: Proper-Temporal-Embedding (PTE), where distributed CPS entities must enter/leave risky states according to properly nested temporal pattern and certain duration spacing. Our solution introduces hybrid automata to formally describe and analyze CPS design patterns. We propose a novel leasing based design pattern, along with closed-form configuration constraints, to guarantee PTE safety rules under arbitrary wireless communication faults. We propose a formal procedure to transform the design pattern hybrid automata into specific wireless CPS designs. This procedure can effectively isolate physical world parameters from affecting the PTE safety of the resultant specific designs. We conduct two wireless CPS case studies, one on medicine and the other on control, to show that the resulted system is safe against communication failures. We also compare our approach with a polling based approach. Both approaches support PTE under arbitrary communication failures. The polling approach performs better under severely adverse wireless medium conditions; while ours performs better under benign or moderately adverse wireless medium conditions.

Keywords—Cyber-physical systems, properly nested locking, mutual exclusion, hybrid systems model checking, real-time.

(Note: Appendices start from page 15)

1 Introduction

To introduce the CPS context [1], 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 required temporal spacing; and then (after the intended task is done) exit to the respective safe states in exactly the reverse order, and with certain required temporal spacing. 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 these discrete ordering and continuous-time temporal conditions define a temporal interlocking pattern, and is termed as *Proper-Temporal-Embedding (PTE) safety rules*.

As an example of PTE safety (see Fig. 1), in the classic medical CPS of laser tracheotomy [2], 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*.

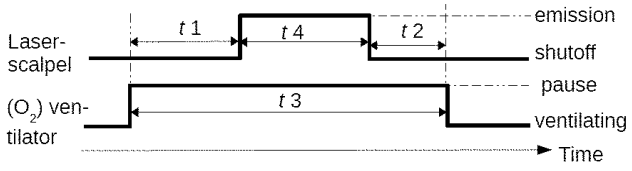


Fig. 1. Proper-Temporal-Embedding Example

Furthermore, as CPS environment entails wireless-connected sensing, control and computing entities, guaranteeing PTE safety rules necessitates consideration of

- *F. Tan, Y. Wang, and Q. Wang are with Dept. of Computing, The Hong Kong Polytechnic Univ., Hong Kong S. A. R. Email: csqwang@comp.polyu.edu.hk*
- *L. Bu is with State Key Lab for Novel Software Technology, Dept. of Computer Sci. and Tech., Nanjing Univ., Nanjing, 210093, P. R. China. Email: bulei@nju.edu.cn*
- *N. Suri is with Dept. of Computer Science, TU Darmstadt, Germany. Email: suri@cs.tu-darmstadt.de*
- *Corresponding Author: Qixin Wang*

unreliable wireless communication. Thus, we utilize and adapt the established design pattern of "*leasing*" [3]–[8], 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 (aka *leasing* 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.

Lease based design pattern 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 leasing based designs in computer systems, the wireless CPS leasing based design faces the following paradigm shifts.

First, leasing based designs in computer (i.e. cyber) systems are often integrated with distributed check-point and roll-back [3]–[6]. 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, in addition to logic-time, continuous-time durations (e.g. the maximal dwelling duration and safeguard interval in PTE safety rules) matter.

Considering the above paradigm shifts, our leasing based design pattern shall not use check-point or roll-back. Instead, its safety is guaranteed by properly configuring continuous-time temporal parameters.

These heuristics are systematically developed into a lease based design pattern for wirelss CPS PTE safety guarantee in this paper. Specifically, this paper's contributions include:

1. We formalize a temporal interlocking/mutual-exclusion pattern (i.e. PTE safety rules) for CPS physical component interactions.
2. We propose a rigorous leasing 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 utilizing *hybrid modeling* [9]–[11] 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.
4. We propose a formal methodology to refine the design pattern hybrid automata into specific wireless CPS designs. This methodology can effectively isolate physical world parameters (which are much harder to control, compared to the software/cyber parameters) from affecting the PTE safety of the resultant specific wireless CPS designs.
5. We conduct two case studies, respectively on wireless medical CPS and wireless control CPS, to validate our proposed approach. We also compare our approach with a polling based approach proposed by Kim et al [12]. The comparison results show that both approaches can guarantee PTE safety against arbitrary communication failures. In terms of resource occupation efficiency and user experience, the polling based approach performs better under severely adverse wireless medium conditions; while ours performs better under benign or moderately adverse wireless medium conditions.

The rest of the paper is organized as follows. Section 2 introduces the CPS hybrid modeling background; Section 3 describes the requirements to guarantee PTE safety rules; Section 4 formally defines the leasing based design pattern, proves its guarantee of PTE safety rules, and describes how to elaborate the design pattern into specific designs. Section 5 and 6 respectively evaluate our proposed approach with emulation/experiment based case studies and simulation based comparisons. Section 7 discusses related work. Section 8 concludes paper. Appendices are included in the Supplementary File [13] as an indispensable and integral part of this paper.

The conference version of this paper is published in [14]. Compared to [14], in this paper, we give more case studies and comparisons, and fixed several typos (see Supplementary File Appendix I [13] of this paper).

2 Background, Terms and Models

2.1 The Hybrid Modeling Terminology

Hybrid modeling is based on hybrid automaton [9]–[11], [15], [16], a tool that suits CPS modeling extremely well because it can formally describe/analyze both discrete (cyber) and continuous (physical) dynamics. For example, Fig. 2 illustrates a hybrid automaton A'_{vent} that describes the discrete/continuous behaviors of a stand-alone ventilator (see Appendix A of [13] for a more detailed description on the ventilator's working mechanism). $H_{\text{vent}}(t)$ is the height of the ventilator piston at time t. The hybrid automaton execution initially dwells in the location of "PumpOut": the piston continuously moves downward at velocity $\dot{H}_{\text{vent}}(t) = -0.1$(m/s). When the piston hits bottom ($H_{\text{vent}} = 0$), a discrete event happens: the execution moves to location "PumpIn". Once in location "PumpIn", the piston continuously moves upward at velocity $\dot{H}_{\text{vent}}(t) = +0.1$(m/s). When the piston hits ceiling ($H_{\text{vent}} = 0.3$(m)), a discrete event happens: the execution moves to location "PumpOut" again, so on and so forth.

In the rest of the paper, we reuse the notations proposed by Alur et al. [11] to formally describe hybrid automata. For reader's convenience, the notation list is

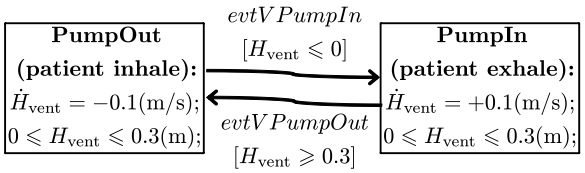


Fig. 2. Hybrid Automaton A'_{vent} of a Stand-Alone Ventilator. $H_{\text{vent}}(t)$ is the data state variable denoting the ventilator's piston height at time t. "PumpOut" is the only initial location.

also re-presented in our conference version paper [14], and in Supplementary File Appendix B [13] of this paper.

2.2 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[1].

A distributed sink-based wireless CPS consists of the following *entities*: a *base station* ξ_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.

3 Specification of PTE Safety Rules

For the wireless CPS system and communications fault model described in Section 2.2, 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.

1. To make an analogy, each hybrid automaton is like a class in Object-Oriented programming. Data state variables and locations are like class members, hence are "local" ("encapsulated") to their respective hybrid automata (classes). Interactions between hybrid automata are carried out via message (aka event) passing.

Let hybrid system $\mathcal{H} = \{A_i | (i = 0, 1, \ldots, N)\}$ describe 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 positive 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 positive 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 $T^{\mathsf{min}}_{\text{risky}:i\to j}$. 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 $T^{\mathsf{min}}_{\text{safe}:j\to i}$.

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

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

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*".

4 Design Pattern based Solutions

To guarantee PTE safety rules described in the previous section, we propose a leasing based design pattern approach.

4.1 Leasing 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. 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 "*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.

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 one 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 is 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 Supervisor has not yet aborted/cancelled the lease, ξ_i will exit to safe-location automatically.

The above thinking guides us to propose the design of Supervisor, Initializer, and Participant as shown in Table 1. We respectively denote the Supervisor, Initializer, and (the ith) Participant's defining hybrid automata (see Table 1) as A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$. These hybrid automata's diagrams in Table 1 (and the respective detailed diagrams in Appendix C of [13]) are elaborated in the following.

Supervisor:

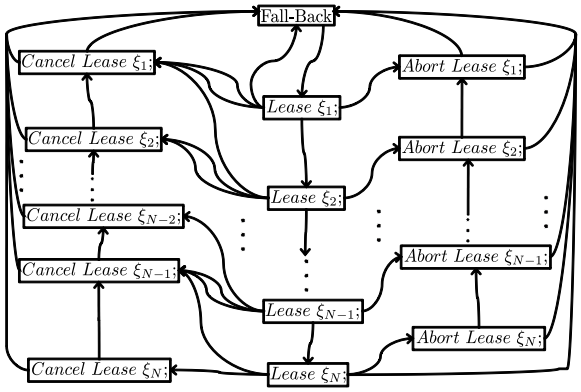
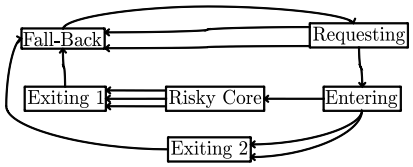
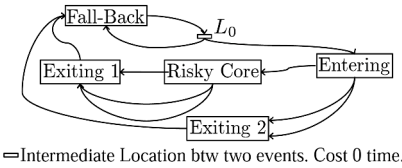
1. A_{supvsr}'s location set V_{supvsr} include the following locations: "Fall-Back", "Lease ξ_i" (where $i = 1 \sim N$), "Cancel Lease ξ_i" (where $i = 1 \sim N$), and "Abort Lease ξ_i" (where $i = 1 \sim N$).
2. Initially, the Supervisor dwells in location "Fall-Back", and all data state variables 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 paragraph), 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. 3 (a).
5. When in location "Lease ξ_N", the behavior of Supervisor can be described by Fig. 3 (b).
6. When in location "Cancel Lease ξ_i" (where $i = 1 \sim N$), the behavior of Supervisor can be described by Fig. 3 (c).
7. When in location "Abort Lease ξ_i" (where $i = 1 \sim N$), the behavior of Supervisor can also be described by Fig. 3 (c), except that every occurrence of "Cancel" is replaced by "Abort".

Initializer:

1. A_{initzr}'s location set V_{initzr} include the following locations: "Fall-Back", "Requesting", "Entering", "Risky Core", "Exiting 1", and "Exiting 2". $V^{\text{risky}}_{\text{initzr}}$

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.

TABLE 1
Specifications of Supervisor, Initializer, and Participant

Role	Conceptual Description of Behaviors	Hybrid Automata Specifications[1,2,3]
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 ξ_1, ξ_2, ..., ξ_{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/aborts are conducted in the reverse PTE order.	
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.	
(ith) 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.	

1. All hybrid automata diagrams here are sketches. Please refer to Appendix C in [13] for respective detailed diagrams.
2. The hybrid automata for Supervisor, Initializer, and (the ith) Participant are respectively denoted as A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$.
3. Note as mentioned in Section 2.2, all data state variable names and location names are local to the corresponding hybrid automata (just like in O-O programming, all class member variable names are local to the class, in contrast to global variable names). For example, A_{supvsr}'s "Fall-Back" location is not A_{initzr}'s "Fall-Back" locations, although the two locations has the same name. Same way, A_{supvsr}'s t_{clk} data state variable (see Fig. 11 in Appendix C of [13], the detailed diagram of A_{supvsr}) is not A_{initzr}'s t_{clk} data state variable (see Fig. 12 in Appendix C of [13], the detailed diagram of A_{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" with continuous dwelling duration over $T^{\text{min}}_{\text{fb},N}$, 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 $T^{\text{max}}_{\text{req},N}$, it will automatically transit back to "Fall-Back". Thirdly, if event $evt\xi_0 To \xi_N LeaseApprove$ 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 $T^{\text{max}}_{\text{enter},N}$, it transits to "Risky Core".
6. When in location "Risky Core", the Initializer ξ_N can send event $evt\xi_N To \xi_0 Cancel$ and transit to "Exiting 1". Secondly, if $evt\xi_0 To \xi_N Abort$ is received, ξ_N also transits to "Exiting 1". Thirdly, if ξ_N dwells continuously in "Risky Core" for $T^{\text{max}}_{\text{run},N}$, 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 event $evt\xi_N To \xi_0 Exit$.

Participant:

1. $A_{\text{ptcpnt},i}$'s location set $V_{\text{ptcpnt},i}$ include the following locations: "Fall-Back", "L0", "Entering", "Risky Core", "Exiting 1", and "Exiting 2". $V^{\text{risky}}_{\text{ptcpnt},i}$ include location "Risky Core" and "Exiting 1"; all other locations belong to $V^{\text{safe}}_{\text{ptcpnt},i}$.
2. Initially, Participant ξ_i dwells in location "Fall-Back"; and all data state variables initial values are zero.

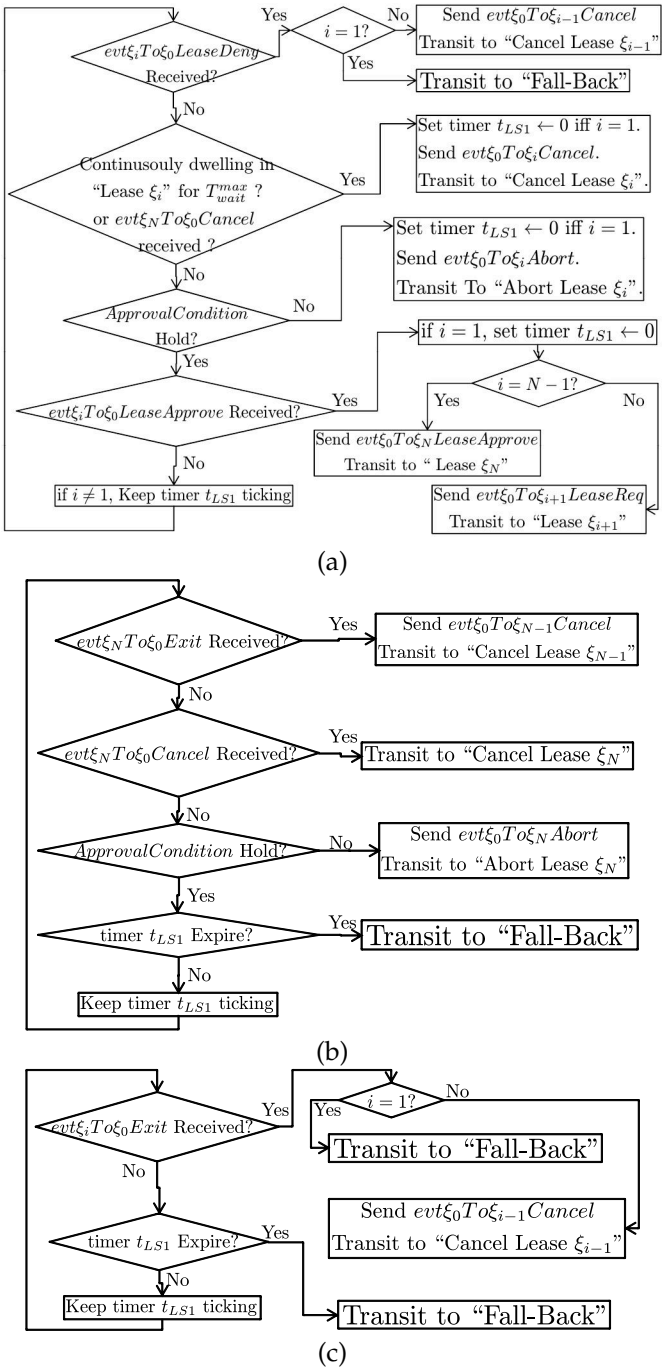


Fig. 3. Flow block diagram at location (a) "Lease ξ_i" ($i = 1 \sim N-1$); (b) "Lease ξ_N"; (c) "Cancel Lease ξ_i" ($i = 1 \sim N$). Note "t_{LS1} expire" means $t_{LS1} \geqslant T_{\text{LS1}}^{\max}$.

3. When in location "Fall-Back" with continuous dwelling duration over $T_{\text{fb},i}^{\min}$, 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".
5. When in location "Entering", if event $evt\xi_0To\xi_iCancel$ or $evt\xi_0To\xi_iAbort$ is received, ξ_i transits to "Exiting 2". Otherwise, if ξ_i dwells continuously in "Entering" for $T_{\text{enter},i}^{\max}$, it transits to "Risky Core".
6. When in location "Risky Core", if event $evt\xi_0To\xi_iCancel$ or $evt\xi_0To\xi_iAbort$ is received, ξ_i transits to "Exiting 1". Otherwise, if ξ_i dwells continuously in "Risky Core" for $T_{\text{run},i}^{\max}$, 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 event $evt\xi_iTo\xi_0Exit$.

4.2 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 $A_{\text{ptcpnt},i}$, PTE safety rules are guaranteed despite any communication faults. Specifically, we have the following result.

Theorem 1 (Design Pattern Validity): Given 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 $A_{\text{ptcpnt},i}$). Suppose $\mathcal{H}$ starts with all entities (i.e. $\xi_0 \sim \xi_N$) residing in location "Fall-Back", and satisfies conditions c1 $\sim$ c7:

- c1. All configuration time constants ($T_{\text{wait}}^{\max}$, $T_{\text{fb},0}^{\min}$, $T_{\text{LS1}}^{\max}$, $T_{\text{req},N}^{\max}$, $T_{\text{fb},i}^{\min}$, $T_{\text{enter},i}^{\max}$, $T_{\text{run},i}^{\max}$, $T_{\text{exit},i}$, where $i = 1 \sim N$) are positive.
- c2. $T_{\text{LS1}}^{\max} \stackrel{\text{def}}{=} T_{\text{enter},1}^{\max} + T_{\text{run},1}^{\max} + T_{\text{exit},1} > NT_{\text{wait}}^{\max}$.
- c3. $(N-1)T_{\text{wait}}^{\max} < T_{\text{req},N}^{\max} < T_{\text{LS1}}^{\max}$.
- c4. $\forall i \in \{1, 2, \ldots, N\}$, there is
$$(i-1)T_{\text{wait}}^{\max} + T_{\text{enter},i}^{\max} + T_{\text{run},i}^{\max} + T_{\text{exit},i} \leq T_{\text{LS1}}^{\max}.$$
- c5. $\forall i \in \{1, 2, \ldots, N-1\}$, there is
$$T_{\text{enter},i}^{\max} + T_{\text{risky}:i \to i+1}^{\min} < T_{\text{enter},i+1}^{\max}.$$

c6. $\forall i \in \{1, 2, \ldots, N-1\}$, there is

$$T^{\max}_{\text{enter},i} + T^{\max}_{\text{run},i} > T^{\max}_{\text{wait}} + T^{\max}_{\text{enter},i+1} + T^{\max}_{\text{run},i+1} + T_{\text{exit},i+1}.$$

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

Then we have:

Claim 1 (Safety): Even if events sent *between* entities can be arbitrarily lost, $\mathcal{H}$ still guarantees PTE safety rules. That is, 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.

Claim 2 (Liveness): Let $P^{\text{PER}}_{N,0}$ denote the packet error rate of the communication channel from the "Initializer" ξ_N to "Supervisor" ξ_0. If $P^{\text{PER}}_{N,0} < 100\%$, i.e. ξ_N can send events to ξ_0 after all. Then *i*) suppose at t_0 all entities (i.e. $\xi_0 \sim \xi_N$) reside in "Fall-Back", then starting from t_0, every $T^{\min}_{\text{fb},N} + T^{\max}_{\text{req},N}$ second, ξ_N has at least one chance to send $evt\xi_N To \xi_0 Req$ to ξ_0, until ξ_0 leaves location "Fall-Back"; *ii*) suppose ξ_0 non-zeno-ly leaves location "Fall-Back" at t_{00} (i.e. ξ_0 is not at "Fall-Back" at t^+_{00}), let $T_{\text{reset}} \overset{\text{def}}{=} (N-1)T^{\max}_{\text{wait}} + T^{\max}_{\text{LS1}} + T^{\min}_{\text{fb},N} + T^{\max}_{\text{req},N} + T^{\max}_{\text{enter},N} + T^{\max}_{\text{run},N} + T_{\text{exit},N}$, then $\exists t \in (t_{00}, t_{00} + T_{\text{reset}}]$, such that all entities (i.e. $\xi_0 \sim \xi_N$) return to location "Fall-Back" at t.

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 $evt\xi_0 To \xi_1 LeaseReq$ 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 $evt\xi_0 To \xi_1 LeaseReq$ 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 in Supplementary File Appendix. D [13] of this paper. ■

4.3 Methodology to Transform Design Pattern into Specific Designs

In the conference version of this paper [14], we further proposed a methodology to transform the aforementioned design pattern hybrid automata A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$ into specific PTE compliant wireless CPS designs. We call this methodology "*elaboration*".

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

Fig. 4 illustrates an example of elaboration. Denote the hybrid automaton of Fig. 2 to be A'_{vent}. We use A'_{vent} to elaborate hybrid automaton A of Fig. 4 (a) at location "Fall-Back". The resulted elaboration is the hybrid automaton A'' of Fig. 4 (b).

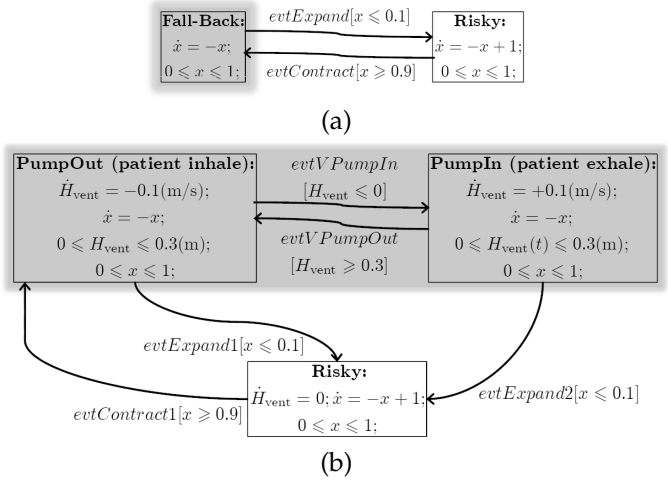


Fig. 4. Elaboration Example (compare the shaded areas in (a) and (b)). (a) Hybrid Automaton A, which has one data state variable x; the shaded location is to be elaborated. (b) Hybrid Automaton A'', which is the elaboration of A (see (a)) at location "Fall-Back" with hybrid automaton A'_{vent} (see Fig. 2); note no edge exists from "Risky" to "PumpIn" because "PumpIn" is not an initial location of A'_{vent}.

The formal description on elaboration is provided Supplementary File Appendix E [13] of this paper for reader's convenience. One important feature of this elaboration methodology is summarized by Theorem 2 in Supplementary File Appendix E [13] of this paper. Sketch of Theorem 2 is re-presented in the following for reader's convenience:

Sketch of Theorem 2 (Design Pattern Compliance): if the design pattern hybrid automata (i.e. A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$) satisfy Condition c1 $\sim$ c7 of Theorem 1, hence guarantee PTE safety rules and liveness described in Theorem 1 Claim 1 and 2, then any specific design resulted from elaborating the design pattern hybrid automata still guarantees the same PTE safety rules and liveness.

5 Case Study

Next, we carry out two case studies to validate our proposed leasing based hybrid design pattern approach. The case studies are respectively on medical CPS and control CPS, two major categories of CPS applications.

5.1 Laser Tracheotomy Wireless Medical CPS

Scenario and Design:

In laser tracheotomy wireless medical CPS (see Fig. 5 (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 $T^{\mathsf{min}}_{\mathrm{risky}:1\to2}$ (we regard the ventilator as entity ξ_1, the Participant; and the laser-scalpel as entity ξ_2, the Initializer); after the emission of laser, the ventilator must wait for at least $T^{\mathsf{min}}_{\mathrm{safe}:2\to1}$ 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.

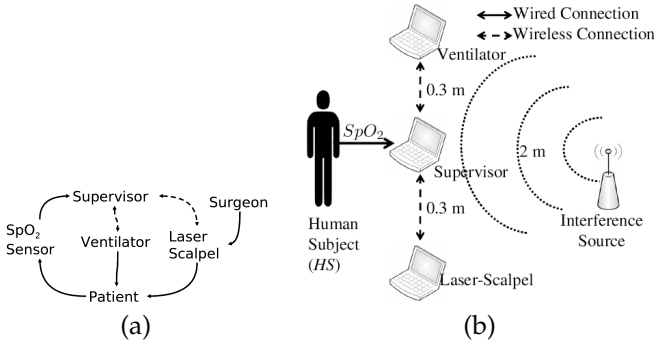


Fig. 5. (a) Laser tracheotomy wireless medical CPS, figure quoted from [2]; (b) Emulation Layout

The ventilator and the laser-scalpel are wirelessly connected via a base station, which also plays the role of the Supervisor (i.e. entity ξ_0). The supervisor/initializer can abort/cancel laser emission at any time (e.g., when the supervisor detects the patient's blood oxygen level SpO_2 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 ventilator are connected via wireless, message losses are possible. Therefore, we carry out our leasing based design approach, so that even with message losses, the wireless CPS can maintain PTE safety rules.

Interested readers can refer to Supplementary File Appendix F [13] of this paper for the resulted detailed design hybrid automata diagrams.

We configure the time parameters of the above detailed design hybrid automata according to commonsense laser tracheotomy requirements [19] 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 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 $T^{\mathsf{min}}_{\mathrm{risky}:1\to2} = 3$(s) and $T^{\mathsf{min}}_{\mathrm{safe}:2\to1} = 1.5$(s).

Per Theorem 2 (see Supplementary File Appendix E [13] of this paper), the above configurations guarantee PTE safety rules. To further validate this, we implemented and carried out emulations of the above design.

Emulation Setup:

Fig. 5 (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.

We also emulate the following three kinds of events, which cause all other events in the emulated system.

The first is the Initializer event $evt\xi_2To\xi_0Req$, 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 (re-)initializing a timer T_{on} (T_{on} follows 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.

The second kind is the Initializer event $evt\xi_2To\xi_0Cancel$, 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 (re-)initializing a timer T_{off} (T_{off} follows exponential distribution) whenever the laser-scalpel enters "Risky Core" (i.e. starts emission). When in "Risky Core" and T_{off} sets off, the (emulated) surgeon requests to cancel laser emission.

The third kind is the Supervisor event $evt\xi_0To\xi_iAbort$ ($i = 1 \sim N$), 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 [20]), 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 2.45GHz ZigBee TMote-Sky motes [21]. In addition, there is an IEEE 802.11g WiFi interference source 2 meters away from the supervisor. The interference source runs Iperf (a standard network evaluation software, see http://iperf.sourceforge.net) to generate 3Mbps interfering data traffic to be broadcasted through a WiFi 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.

5.2 Inverted Pendulum Remote Monitoring Wireless Control CPS

Scenario and Design:

Inverted Pendulum (IP) is a metal rod (the pendulum) with one end hinged on a cart, and the other end free rotating. The cart can move along a rail (the "x-axis") to keep the hinged rod standing up-right still. Due to its inborn instability, IP is a widely adopted test bed for various control strategies, including control CPS [22] [23].

In our IP remote monitoring case study (see Fig 6 for the application layout), the IP (entity ξ_2, the Initializer) may randomly request for a random walk, i.e. randomly adjust the cart's *reference location* (i.e. the target stabilization location) on the rail. Because random walk is considered a risky operation, the entire duration of random walk, including $T^{\mathsf{min}}_{\mathrm{risky:1\to 2}}$ seconds right before the random walk, and $T^{\mathsf{min}}_{\mathrm{safe:2\to 1}}$ seconds right after the random walk, must be continuously monitored/recorded by a remote video camera (entity ξ_1, the Participant). The video record can be used for real-time decision making, or for future analysis, debugging, or accident-forensics. Meanwhile, we do not allow infinite random walk, hence the duration of each random walk, and the corresponding duration of remote monitoring are upper bounded.

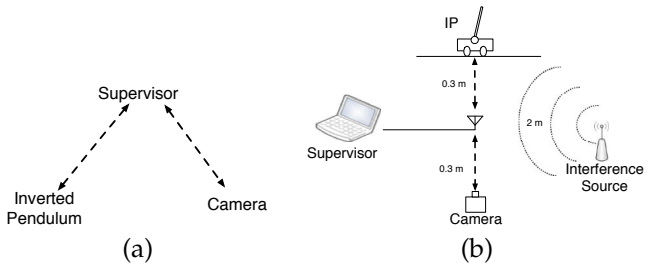


Fig. 6. (a) Inverted Pendulum (IP) remote monitoring wireless control CPS; (b) Experiment Layout

Similar to the laser tracheotomy case, the monitoring camera and the IP are wirelessly connected to the supervisor (entity ξ_0). The supervisor/initializer can abort/cancel the random walk at any time, but the PTE safety rules must be maintained.

Meanwhile, as the supervisor, IP, and camera are connected via wireless, message losses are possible. Therefore, we carry out our leasing based design approach, so that even with message losses, the wireless CPS can maintain PTE safety rules.

The detailed design of hybrid automata in IP remote monitoring (see Appendix F in [13]) is similar to the laser tracheotomy case, except that in "Risky Core" location, the IP conducts random walk, and the camera conducts video recording. We configure the time parameters of the detailed design hybrid automata as follows. For the Supervisor, $T^{\mathsf{min}}_{\mathrm{fb},0} = 0.1$(s), $T^{\mathsf{max}}_{\mathrm{wait}} = 0.1$(s). For the Initializer (i.e. the IP), $T^{\mathsf{max}}_{\mathrm{req},2} = 0.1$(s), $T^{\mathsf{max}}_{\mathrm{enter},2} = 3.0$(s), $T^{\mathsf{max}}_{\mathrm{run},2} = 20$(s), $T_{\mathrm{exit},2} = 2.5$(s). For the Participant 1 (i.e. the camera), $T^{\mathsf{max}}_{\mathrm{enter},1} = 1.0$(s), $T^{\mathsf{max}}_{\mathrm{run},1} = 35.0$(s), $T_{\mathrm{exit},1} = 6$(s). The PTE enter-risky/exit-risky safeguard intervals are $T^{\mathsf{min}}_{\mathrm{risky:1\to 2}} = 1.0$(s) and $T^{\mathsf{min}}_{\mathrm{safe:2\to 1}} = 1.5$(s). The above settings satisfy condition c1 $\sim$ c7 in Theorem 1, meanwhile allow reasonable duration length for random walk and monitoring.

Experiment Setup:

We implemented the IP remote monitoring detailed design, and carried out experiment evaluation. Fig. 6 (b) shows our experiment layout. The layout and settings are the same as those of our laser tracheotomy emulation, except that the laser scalpel emulator, ventilator emulator, and (laser tracheotomy) supervisor are respectively replaced by the IP, camera, and the (IP remote monitoring) supervisor.

5.3 Trials and Results

For laser tracheotomy wireless medical CPS (IP remote monitoring wireless control CPS), we ran two emulation (experiment) trials, each of 30 minutes duration. During the trials, the PTE safety rules are:

1. Neither ventilator pause (camera monitoring) nor laser emission (IP random walk) can last for more than 1 minute;
2. Ventilator pause (camera monitoring) duration must always properly-temporally-embedding laser emission (IP random walk) duration, with entering/exiting safeguard interval of $T^{\mathsf{min}}_{\mathrm{risky:1\to 2}} = 3$ seconds (1 second) and $T^{\mathsf{min}}_{\mathrm{safe:2\to 1}} = 1.5$ second (1.5 second).

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

As mentioned before, in the two trials, the emulated surgeon (IP) requests to emit/cancel-emit laser (start/cancel random walk) according to timer T_{on} and T_{off}, which are both random numbers following exponential distribution. The expectation of T_{on} is 30 seconds. The expectations of T_{off} are 18 seconds and 6 seconds respectively in the two trials.

Because of the use of our proposed leasing based design pattern, and the configuration of parameters satisfying Theorem 2 (see Supplementary File Appendix E [13] of this paper), although packets/events between the ventilator emulator (camera), supervisor, and laser-scalpel emulator (IP) can be arbitrarily lost, the PTE safety rules are never violated. This is shown in Table 2, the rows corresponding to "with Lease" always have 0 failures.

For comparison, for each case study, we also ran two additional emulation (experiment) trials with the same configurations but without using the leasing mechanism. Specifically, the ventilator (camera) does not set up a

TABLE 2
PTE Safety Rule Violation (Failure) Statistics

(a) Laser Tracheotomy Emulation

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

(b) IP Remote Monitoring Experiment

Trial Mode	$E(T_{\text{off}})$ (sec)	# of IP random walks	# of Failures	# of $evtRunEnded$
with Lease	18	12	0	8
without Lease	18	11	6	0
with Lease	6	15	0	10
without Lease	6	13	7	0

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. $evtRunEnded$ occurs when lease expiration forces the laser-scalpel (IP) to stop emitting (random walk), i.e. when lease mechanism takes effect to rescue the system from violating the PTE safety rules.

lease timer when it starts pausing (monitoring), neither does the laser-scalpel (IP) set up a lease timer when it starts emitting laser (random walk). When the surgeon's cancel laser emission event (the IP's cancel random walk event) is lost or the supervisor's abort event is lost, no one can terminate the ventilator's pause (the camera's monitoring) or the laser's emission (the IP's random walk). Thus, as shown in Table 2, the rows corresponding to "without Lease" all result in many failures.

To facilitate understanding of the above results, in the following, we provide some more intuitive explanations.

Without loss of generality, let us focus on the laser tracheotomy case study.

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 1 further guarantee that the automatic returns to "Fall-Back" of ventilator and laser-scalpel both conform to proper-temporal-embedding even under arbitrary packet/event losses.

Interested readers can refer to Appendix G of [13] for even more intuitive explanations.

6 COMPARISONS

PTE safety is a relatively new issue raised by CPS. To our best knowledge, the state-of-the-art solution is a polling based approach proposed by Kim et al. [12].

6.1 Polling Based Approach

Kim et al. [12]'s polling based approach also adopts a layout of distributed entities, with a base station and several remote entities. The base station also serves the role of the Supervisor; one remote entity is the Initializer, and the other remote entities serve the role of Participants. However, different from our leasing based approach, the Supervisor does not passively wait for messages sent from the Initializer to trigger a sequence of PTE operations. Instead, it periodically polls remote entities for their current states. The polling message is also piggy backed with a plan vector. The plan vector is basically instructions on what the remote entity shall do in the current and future periods, assuming communication link with the base station will be broken in the future periods. Also, the remote entities cannot change their (cyber) states (though the Initializer can *request* to start/cancel a sequence of PTE operations), unless instructed by the plan vector from the Supervisor.

For example, the plan vector for laser-scalpel may set the laser-scalpel to keep emitting for the next two periods and then deactivate in the third period; whereas the plan vector for ventilator may ask the ventilator to keep pausing in the next four periods. The Supervisor coordinates these plan vectors, ensuring the PTE safety rules are guaranteed. The polling temporal sequence (exemplified by the case of laser tracheotomy) is shown in Fig 7.

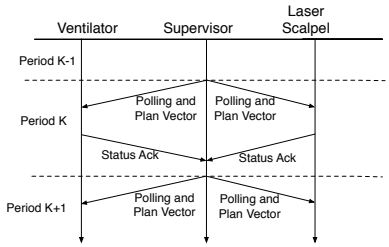


Fig. 7. The Polling Temporal Sequence for Laser Tracheotomy Wireless CPS (quoted from [12]).

6.2 Simulation Setup

We compare our leasing based approach with Kim et al. [12]'s polling based approach with simulation.

We reuse laser tracheotomy and IP remote monitoring described in Section 5 as the application background. Particularly, the PTE safety demands remain the same.

For our leasing based approach, the simulation setup matches what is described in Section 5. The only exception is that to improve the approach's robustness, each wireless packet is consecutively retransmitted ten times from the application layer, once per 10ms.

For the polling based approach, we follow the instructions in [12]. The detailed designs and parameter configurations are given in Appendix H of [13]. There are two issues worth particular mentioning.

The first issue is on the choice of polling period. The longer the polling period, the less wireless messages sent per second, and hence better wireless medium occupation efficiency (when the wireless medium is benign or moderately adverse). On the other hand, the polling

period can neither be too long, due to two reasons. First, the polling based approach assumes all remote entities' physical state change within a polling period is negligible (unless the plan vector instructs the remote entity to conduct a state change). Second, response time to user requests is at least one polling period; and for good user experience, under benign or moderately adverse wireless medium conditions, the response time should be within tens of milliseconds [24] [25]. Considering the above factors, we set the polling period at 50(ms).

The second issue is on the choice of other configuration parameters. Note PTE safety rules only care about worst case time bounds (e.g., the laser emission *must* take place *at least* 1.5 seconds after the ventilator has paused). Within these time bound constraints, many feasible configurations exist for both the polling and the leasing based approaches (e.g. the laser emission *can* take place 2 seconds, or 3 seconds after the ventilator has paused). To make our comparisons fair, the polling and leasing based approach parameters are configured so that the default (i.e. when there is no cancellation nor abort, and no communication packet loss) behaviors of ξ_1, ..., ξ_N are the same under both approaches[3].

6.3 Results and Analysis

Based on aforementioned analysis, emulations, and experiments, we know that both our leasing based approach and Kim et al. [12]'s polling based approach can guarantee PTE safety rules under arbitrary wireless communication failures. However, their performance, specifically, wireless medium occupation efficiency and user experience, may be different.

To strictly quantify the two performance indicators, we further consider three wireless medium conditions: benign, moderately adverse, and adverse, respectively corresponds to a *Packet Error Rate* (PER) of 0.5%, 5%, and 50%.

For each approach (leasing based vs. polling based), each application (laser tracheotomy, IP remote monitoring), and each wireless medium condition, we run 1000 simulation trials. In each trail, all entities start from "Fall-Back"-equivalent locations/states [4] , and then the Initializer will request to run a complete sequence of PTE operations. The Initializer will keep requesting until approval is received from the Supervisor [5] . Suppose the Initializer started to request at t_0, and first received the Supervisor's approval at t_1, we call the duration $(t_1 - t_0)$ the "*initialization response time*".

Once the Initializer enters its "Risky-Core" location/state, it (re-)initializes a timer T_{off}. T_{off} follows exponential distribution with an expectation of 18(s). If T_{off} sets off and the Initializer is still in "Risky-Core" location/state, the Initializer requests to cancel the risky activity (i.e. laser emission or IP random walk). Suppose the Initializer requests to cancel the risky activity at t_2, and suppose if no message is lost, the Participant can return to "Fall-Back" location/state at t_3^*. Meanwhile, denote the actual time the Participant returns to "Fall-Back" location/state to be t_3. Then we call the difference $(t_3 - t_3^*)$ the "*extra suffering time*". Extra suffering time quantifies the extra suffering time endured by the Participant due to message losses in the cancellation process. That is, the Initializer has cancelled the risky activity, but the Participant is not notified, hence has to suffer longer, waiting for the leasing or polling mechanism to return it to "Fall-Back" location/state.

We use initialization response time and extra suffering time to quantify user experiences. For both metrics, the shorter means better user experience (quicker response or less extra suffering). Wireless medium occupation, however, is quantified by the ratio of time used for wireless transmission during the whole interval of $[t_0, t_3]$. The higher the ratio, the worse the wireless medium occupation efficiency (i.e. the more wasteful of the wireless medium).

The simulations results for laser tracheotomy are summarized in Fig. 8. We can make several observations from these figures.

First, our leasing based approach incurs less wireless medium occupation ratio than polling based approach. As shown in Fig. 8 (a)(d), leasing's wireless medium occupation ratio is upper bounded by 0.65%; while polling's is lower bounded by 5.69%. Later we will see the impact of this difference on system scalability. This difference is intuitive: our leasing based approach is an event based approach, no messages are sent unless certain event takes place; while polling based approach sends messages every period no matter what. The benefit of wireless medium occupation efficiency will become more significant when we evaluate system scalability (see later paragraphs).

Second, when wireless medium is benign (e.g. PER = 0.5%) or moderately adverse (e.g. PER = 5%), leasing based approach can provide slightly better user experience. As shown in Fig. 8 (b)(e), leasing's initialization response time is upper bounded by 44ms, while polling's is lower bounded by 100ms; and as shown in Fig. 8 (c)(f), leasing's extra suffering time statistics (1st/3rd quartile, median, maximum) are all roughly one order of magnitude shorter than polling's. This is because under benign or moderately adverse wireless medium condition, packet loss is rare. Under leasing based approach, an event can be immediately responded to; while under

3. Here we ignore the delay differences caused by polling period and packet transmission, which is in $\leqslant 50$ms range.

4. For leasing based approach, this means the Supervisor and Initializer both start from the "Fall-Back" location (see Fig. 14, 15, 17, 18 in [13]); the Participant starts from "PumpOut" location in the case of laser tracheotomy (see Fig. 16 in [13]) , and from "Fall-Back" location in the case of IP remote monitoring (see Fig. 19 in [13]). For polling based approach, this means the Supervisor, Initializer, and Participant all start from the "Fall-Back" state (see Fig. 20 $\sim$ 25 in [13]).

5. For leasing based approach, this means message $evt\xi_0 To\xi_2 LeaseApprove$ is received by the Initializer (see Fig. 15, 18 in [13], the Supplementary File of this paper). For polling based approach, this means message $evt\xi_0 Ack$ is received by the Initializer (see Fig. 21, 24 in [13]).

polling based approach, every response must take at least one polling period. However, when wireless medium is severely adverse (e.g. PER = 50%), polling based approach provides better user experience than leasing based approach (see Fig. 8 (b)(c)(e)(f), leasing's maximum initialization response time and extra suffering time can respectively reach 10s and 31.03s, while polling's only respectively reach 2.52s and 0.671s). This is intuitive: polling based approach is basically continuously retransmitting messages every period, hence has better chance of delivering messages when packet loss rate is high[6].

Finally, we also evaluate the scalability of the two approaches. We study an N-IP remote monitoring scenario, where N pairs of IP-camera are being coordinated by a Supervisor. Fig. 9 compares the performances of leasing and polling based approaches when N scales up from 1 to 12 (wireless medium is set to moderately adverse, i.e. PER = 5%). We can see that polling based approach uses up wireless bandwidth quickly as N grows; when $N = 12$, nearly all wireless bandwidth is used up (91.72% in the worst case). In contrast, our leasing based approach only uses a small portion of the wireless bandwidth for all N values (5.86% in the worst case). This matches intuition, as our leasing based approach carries out event based interrupt-like communication, which is well-known to be more communication resource thrift than polling.

7 Related Work

Lease based design pattern was originally proposed by Gray et al. [3] and is used to provide efficient consistent access to cached data in distributed computer systems. In the past decades, various leasing based distributed computer systems have been implemented to achieve system consistency [4]–[8], [26]. As pointed out in Section 1, all these distributed computer systems are fundamentally different from CPS due to following reasons: 1) checkpoint and roll-back, two fundamental operations in lease-based distributed computer systems are often impossible for CPS (e.g. we cannot revive a killed patient); 2) PTE temporal ordering, particularly the continuous-time duration requirements (such as the minimal safeguard interval) are usually not present for distributed computer systems (which instead focus on logical-time, aka causal precedences).

Although formal methods have been applied to design pattern research [27], [28], hybrid modeling is mostly used for verification [2], [9]–[11], [15]. Recently, Tichakorn [29] 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 focus there is still verification, rather than design.

8 Conclusion and Future Work

In this paper, we formalize a temporal interlocking/mutual-exclusion pattern called PTE safety rules for CPS physical component interactions. We propose a leasing 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 derive a set of closed-form constraints, and prove that as long as system parameters are configured to satisfy these constraints, PTE safety rules are guaranteed under arbitrary wireless communication faults. Furthermore, we develop hybrid modeling approaches to describe the design patterns, and develope a formal methodology to elaborate the design pattern into specific designs that provide PTE safety guarantees. Our case studies on laser tracheotomy wireless CPS and inverted pendulum remote monitoring validate the proposed design methodology. We also compare our solutions with a polling based solution. The comparison results show that the polling based solution performs better under severely adverse wireless medium conditions, while ours performs better under benign or moderately adverse wireless medium conditions. As our future work, we will investigate additional network protocol technologies to enhance wireless communication reliability, hence to further improve the performance of our leasing based approach and the polling based approach. We will also explore more application domains for the proposed design pattern, such as chemical plant safety, anesthesiology, control, where timing (time duration) is an important parameter in defining safety rules.

Acknowledgement

The research project related to this paper is supported in part by Hong Kong RGC ECS PolyU 5328/12E, RGC GRF PolyU 5245/09E, DAAD/RGC Germany/HK Joint Research Scheme G_HK022/12, by The Hong Kong Polytechnic Univ. A-PJ68, A-PJ80, A-PK46, A-PL82, G-YN37. Lei Bu is supported by the National Key Basic Research Program of China2014CB340703) and the National NSFC (No.91318301, No.61321491, No.61100036) . Neeraj Suri is supported in part by TUD CASED. We thank Prof. Rong Zheng and reviewers for their advice on improving this paper. 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.

6. It is worth noting that initialization response time refers to the duration of a request-reply sequence taking place at the very beginning of PTE (and assuming all entities starts from "Fall-Back"). Therefore, it is irrelevant to most of PTE configuration parameters except $T^{\max}_{\text{req},N}$ (request time out). In both leasing and polling schemes, $T^{\max}_{\text{req},N}$ are the same, hence the comparison is fair. Meanwhile, extra suffering time refers to the *difference* between actual response time and the ideal response time when PER = 0. Most PTE parameters are cancelled out due to the substraction, leaving only ξ_1 (Participant)'s maximum dwelling time in risky-location relevant: in the worst case, the patient (camera) has to suffer this maximum dwelling time longer than the PER = 0 case. Again, this parameter settings are the same for both leasing and polling schemes, hence the comparison is fair.

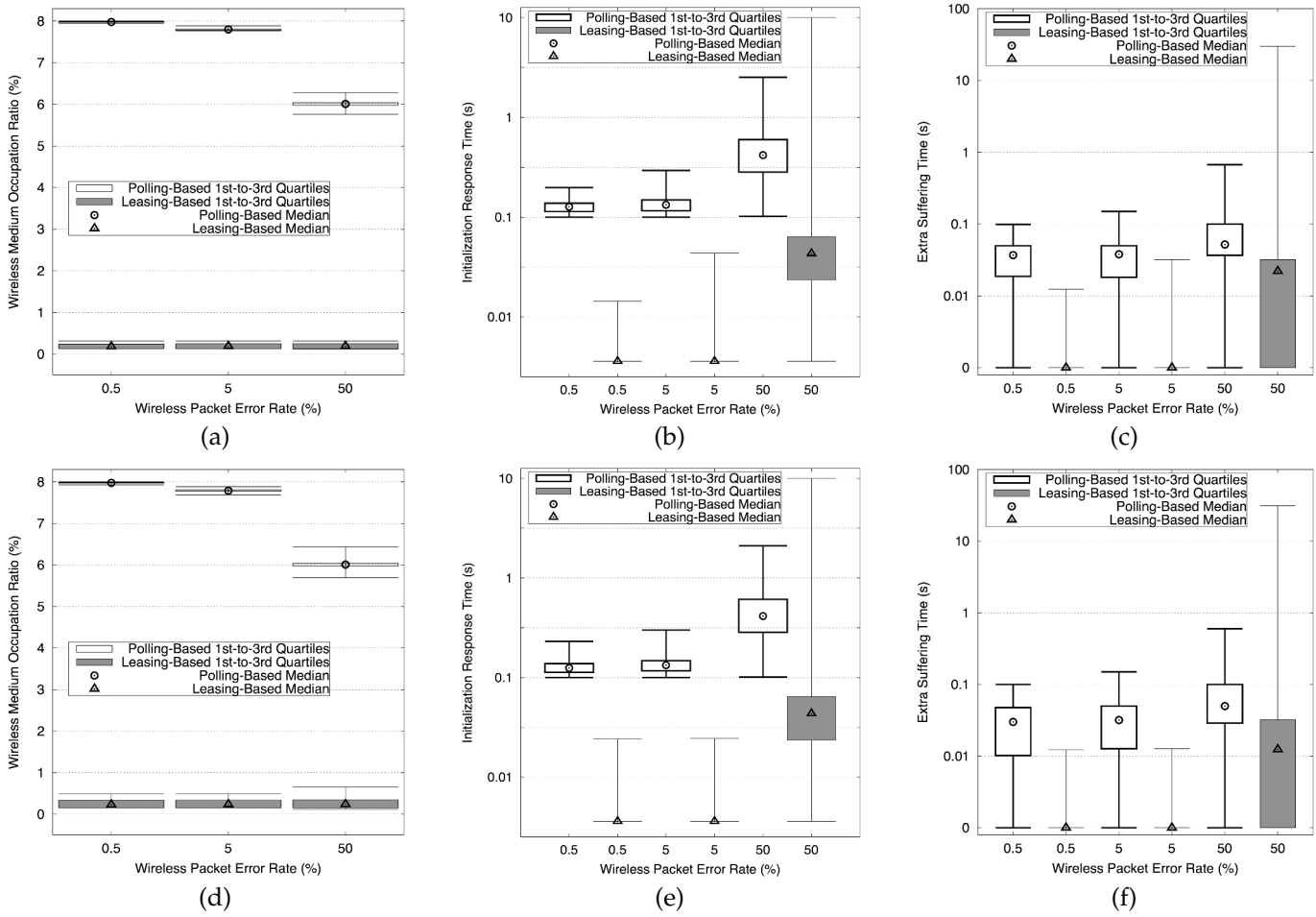


Fig. 8. Comparisons between Leasing-Based Approach and Polling-Based Approach in Laser Tracheotomy ((a) $\sim$ (c)) and IP Remote Monitoring ((d) $\sim$ (f)) Wireless CPS

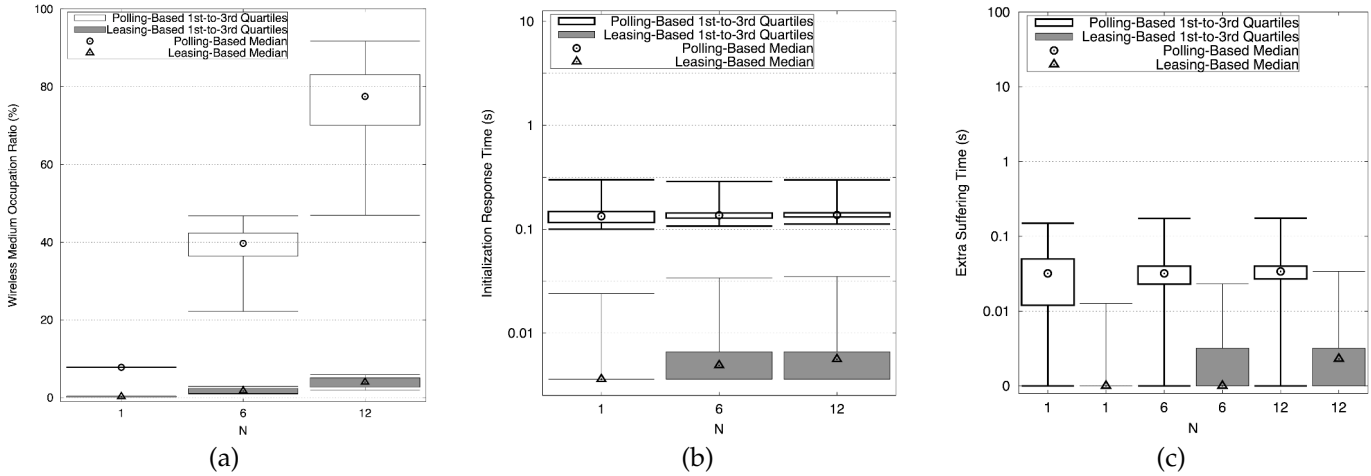


Fig. 9. Scalability Comparisons between Leasing-Based Approach and Polling-Based Approach (N is the number of IPs being remotely monitored)

REFERENCES

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

[2] 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.

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

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

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

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

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

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

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

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

[11] R. Alur *et al.*, "Automatic symbolic verification of embedded systems," *IEEE Trans. on Software Eng.*, vol. 22, no. 3, 1996.

[12] C. Kim *et al.*, "A framework for the safe interoperability of medical devices in the presence of network failures," *ICCPS'10*, Apr. 2010.

[13] F. Tan *et al.*, *A Lease based Hybrid Design Pattern for Proper-Temporal-Embedding of Wireless CPS Interlocking (Supplementary File)*. available in IEEEExplore TPDS as it is part of this paper, please ask TPDS for assistance if you cannot access it; or otherwise, it is also available at http://www.comp.polyu.edu.hk/%7Ecsqwang/appendix.html.

[14] ——, "Guaranteeing proper-temporal-embedding safety rules in wireless cps: A hybrid formal modeling approach," *Proc. of DSN*, 2013.

[15] M. Gribaudo *et al.*, "Fluid petri nets and hybrid model-checking: A comparative case study," *Reliability Engineering And System Safety*, vol. 81, 2003.

[16] J. Lunze and F. Lamnabhi-Lagarrigue, *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge Univ. Press, 2009.

[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 Trans. on Mobile Comp.*, vol. 6, no. 6, pp. 706–719, Jun. 2007.

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

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

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

[22] *Quanser*. http://www.quanser.com.

[23] G. F. Franklin *et al.*, *FeedBack Control of Dynamical Systems*. Addison-Wesley, Nov. 1993.

[24] M. Glencross *et al.*, "Exploiting perception in high-fidelity virtual environmentsadditional," in *ACM SIGGRAPH 2006 Courses*, ser. SIGGRAPH '06, 2006.

[25] B. Fisher *et al.*, "Seeing, hearing, and touching: Putting it all together," in *ACM SIGGRAPH 2004 Course Notes*, ser. SIGGRAPH '04, 2004.

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

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

[28] T. Mikkonen, "Formalizing design patterns," *Proc. of ICSE*, 1998.

[29] 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.

Feng Tan received his BE and ME degree in Industrial Engineering from University of Electronic Science and Technology of China, Chengdu, China in 2009 and 2012 respectively. He is currently pursuing the Ph.D degree in the Department of Computing in the Hong Kong Polytechnic University, Hong Kong. His research interests include Cyber-Physical Systems (CPS), particularly on CPS dependability. He is a student member of the IEEE.

Yufei Wang received his BS in Electronics and Information System and MS in Communication and Information System, from Nankai University, Tianjin, China. He received his PhD degree in Computer Science from the Hong Kong Polytechnic University in Hong Kong in 2013. He is currently a senior engineer in Hong Kong Applied Science and Technology Research Institute Company Ltd. His research interests include dependable systems, low latency CPS, etc.

Qixin Wang received his BE and ME degrees from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 1999 and 2001 respectively, and PhD degree from the Department of Computer Science, University of Illinois at Urbana-Champaign in 2008. He is currently an Assistant Professor in the Department of Computing at the Hong Kong Polytechnic University. Dr. Wang is a member of the IEEE and the ACM.

Lei Bu is an associate professor in the Department of Computer Science and Technology, State Key Laboratory for Novel Software Technology at Nanjing University, P.R.China. He received his BS and PhD degree in Computer Science from Nanjing University in 2004 and 2010 respectively. He has been a visiting scholar to CMU, Fondazione Bruno Kessler, and UT Dallas. Dr. Bu is also a member of the ACM.

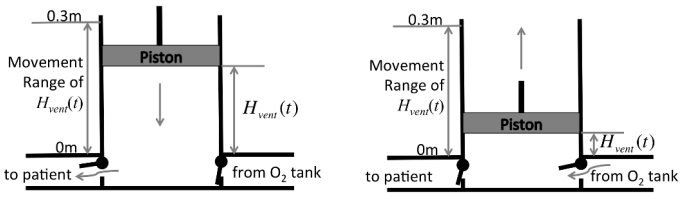
Neeraj Suri received his MS and PhD from the University of Massachusetts at Amherst. He is a Chair Professor at the Dept. of CS at TU Darmstadt and also with the University of Texas at Austin. His professional details are available at http://www.deeds.informatik.tu-darmstadt.de/suri

APPENDIX A
VENTILATOR WORKING MECHANISM

The working mechanism of our ventilator is illustrated by Fig. 10.

Basically, our ventilator consists of a cylinder, a piston, and two valves. The piston moves up/down the cylinder to pump oxygen from oxygen tank into patient. When the piston moves downward, the valve toward the patient is opened and the valve from the oxygen tank is closed, hence oxygen is pumped out to the patient, forcing the patient to inhale. When the piston moves upward, the valve from the oxygen tank is opened, hence oxygen is pumped into the cylinder; meanwhile the valve to the patient is closed, allowing the patient to naturally exhale due to his/her chest weight.

We denote the current height of the piston as $H_{\text{vent}}(t)$; its movement range is from 0(m) to 0.3(m). $\dot{H}_{\text{vent}}(t)$ is the velocity of the piston. When the piston moves downward, $\dot{H}_{\text{vent}}(t) = -0.1$(m/s). When the piston moves upward, $\dot{H}_{\text{vent}}(t) = +0.1$(m/s). The piston movement changes direction when $H_{\text{vent}}(t)$ hits its bottom 0(m) or ceiling 0.3(m).



(a) Pumping Out (b) Pumping In

Fig. 10. Ventilator Working Mechanism. $H_{\text{vent}}(t)$ is the piston height at time t. $\dot{H}_{\text{vent}}(t)$ is the piston velocity at time t. (a) when the piston moves downward, oxygen is to pumped out to patient (forcing patient to inhale); (b) when the piston moves upward, oxygen is pumped in from tank (meanwhile patient exhales naturally due to chest weight).

APPENDIX B
FORMAL DEFINITIONS OF HYBRID AUTOMATA

As one 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 [9]–[11], a hybrid automaton A is a tuple $(\vec{x}(t), V, \mathsf{inv}, F, E, g, R, L, \mathsf{syn}, \Phi_0)$ 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 $(H_{vent}(t))$, i.e. it contains only one data state variable: $H_{vent}(t)$, which is the height of the ventilator piston at time t.

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 piston height $H_{vent}(t)$ stays in the range $0 \leq H_{vent}(t) \leq 0.3$(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 piston 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 piston 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 *synchronization 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 piston height $H_{vent}(0) \in [0, 0.3]$(m).

APPENDIX C
DETAILED DIAGRAMS FOR DESIGN PATTERN HYBRID AUTOMATA

Please see Fig. 11, 12, and 13 for the detailed diagram of A_{supvsr}, A_{initzr}, and $A_{\text{ptcpnt},i}$, the hybrid automaton for Supervisor, Initializer, and the ith Participant respectively. Note all hybrid automata's initial locations are "Fall-Back", and all data state variables are initialized to 0.

APPENDIX D
PROOF OF THEOREM 1

First, we can prove the following lemma.

Lemma 1 (Guaranteed Resetting): Suppose $evt\xi_0To\xi_1LeaseReq$ happens at time t_0, then we have

$\forall i \in \{0, 1, \ldots, N\} \cdot \ell_i(t_0^-) \equiv$ "Fall-Back".

Proof: First, because $evt\xi_0To\xi_1LeaseReq$ happens at t_0 and ξ_0's location "L0" allows 0 dwelling time, we have $\ell_0(t_0^-) =$ "Fall-Back". ξ_0's location "L0" allows 0 dwelling time also implies $evt\xi_NTo\xi_0Req$ happens at t_0, so $\ell_N(t_0^-) =$ "Fall-Back" (see Fig. 12).

Now all we need to prove is

$$\ell_i(t_0^-) = \text{ "Fall-Back" } (i = 1 \sim N-1). \tag{2}$$

We can prove this inductively. As all hybrid automata of $\mathcal{H}$ start from respective "Fall-Back" locations, if t_0 is the first time $evt\xi_0To\xi_1LeaseReq$ happens, Claim (2) sustains.

Suppose Claim (2) sustains for t_0, a time instance that $evt\xi_0To\xi_1LeaseReq$ happens. Suppose $t_1 > t_0$ is the next time instance that $evt\xi_0To\xi_1LeaseReq$ happens. Then there can be two cases for the interval of $[t_0, t_1)$.

Case 1: During $[t_0, t_1)$, $evt\xi_1LeaseExpire$ never happens.

Suppose during $[t_0, t_1)$, ℓ_0 ever reaches location "Lease ξ_j" but not location "Lease ξ_{j+1}" ($j \in \{1, 2, \ldots, N-1\}$).

Then throughout $[t_0, t_1)$, ξ_{j+1}, ξ_{j+2}, ..., ξ_{N-1} never get a chance to leave their respective "Fall-Back" locations.

By checking all possible exit paths from "Lease ξ_j" in Fig. 11, and knowing that $evt\xi_1LeaseExpire$ never happens, we find for each entity ξ_k (where $k = j, j-1, \ldots, 1$), $\exists t \in [t_0, t_1) \cdot$ either $evt\xi_kTo\xi_0LeaseDeny$ or $evt\xi_kTo\xi_0Exit$ happens at t. For otherwise, ℓ_0 cannot be at "Fall-Back" at t_1^-, which has already been proven at the beginning of this proof. Meanwhile, according to Fig. 13, as soon as $evt\xi_kTo\xi_0LeaseDeny$ or $evt\xi_kTo\xi_0Exit$ happens, ℓ_k enters "Fall-Back", and stays there till t_1^- because there is no more $evt\xi_0To\xi_1LeaseReq$ (hence $evt\xi_0To\xi_kLeaseReq$) during $[t, t_1)$.

The same analysis applies to the case when ℓ_0 ever reaches location "Lease ξ_N" during $[t_0, t_1)$.

Therefore, Claim (2) sustains for t_1 in Case 1.

Case 2: $\exists t \in [t_0, t_1)$, $evt\xi_1LeaseExpire$ happens. Then according to Fig. 11, 12, and 13, $t_0 + T_{\text{LS1}}^{\max} \leq t$. As defined, $t < t_1$, so $t_0 + T_{\text{LS1}}^{\max} < t_1$.

On the other hand, the latest time during $[t_0, t_1)$ for ξ_i ($i \in \{1, 2, \ldots, N-1\}$) to leave "Fall-Back" (if it ever leaves) is at $t_0 + (i-1)T_{\text{wait}}^{\max}$ (see Fig. 11, 13). After that, ξ_i's maximal stay outside of "Fall-Back" is $T_{\text{enter},i}^{\max} + T_{\text{run},i}^{\max} + T_{\text{exit},i}$ (see Fig. 13). Because of Condition c4, this means by $t_0 + T_{\text{LS1}}^{\max} < t_1$, ξ_i should have returned "Fall-Back". Due to the same reason as in Case 1, after the return, ξ_i should have stayed in "Fall-Back" till t_1^-.

Therefore, Claim (2) sustains for t_1 in Case 2.

Due to Case 1 and 2, Claim (2) sustains for t_1. Induction sustains.

Note no matter there are infinite, or finite occurrences of $evt\xi_0To\xi_1LeaseReq$, the above induction based proof sustains. ■

Then we have the following lemma.

Lemma 2 (Single Visit between Resets): Let t_0, t_1 ($t_0 < t_1$) be the time instances that two consecutive $evt\xi_0To\xi_1LeaseReq$ happen; or let t_0 be the last time that $evt\xi_0To\xi_1LeaseReq$ happens and $t_1 = \infty$. Either

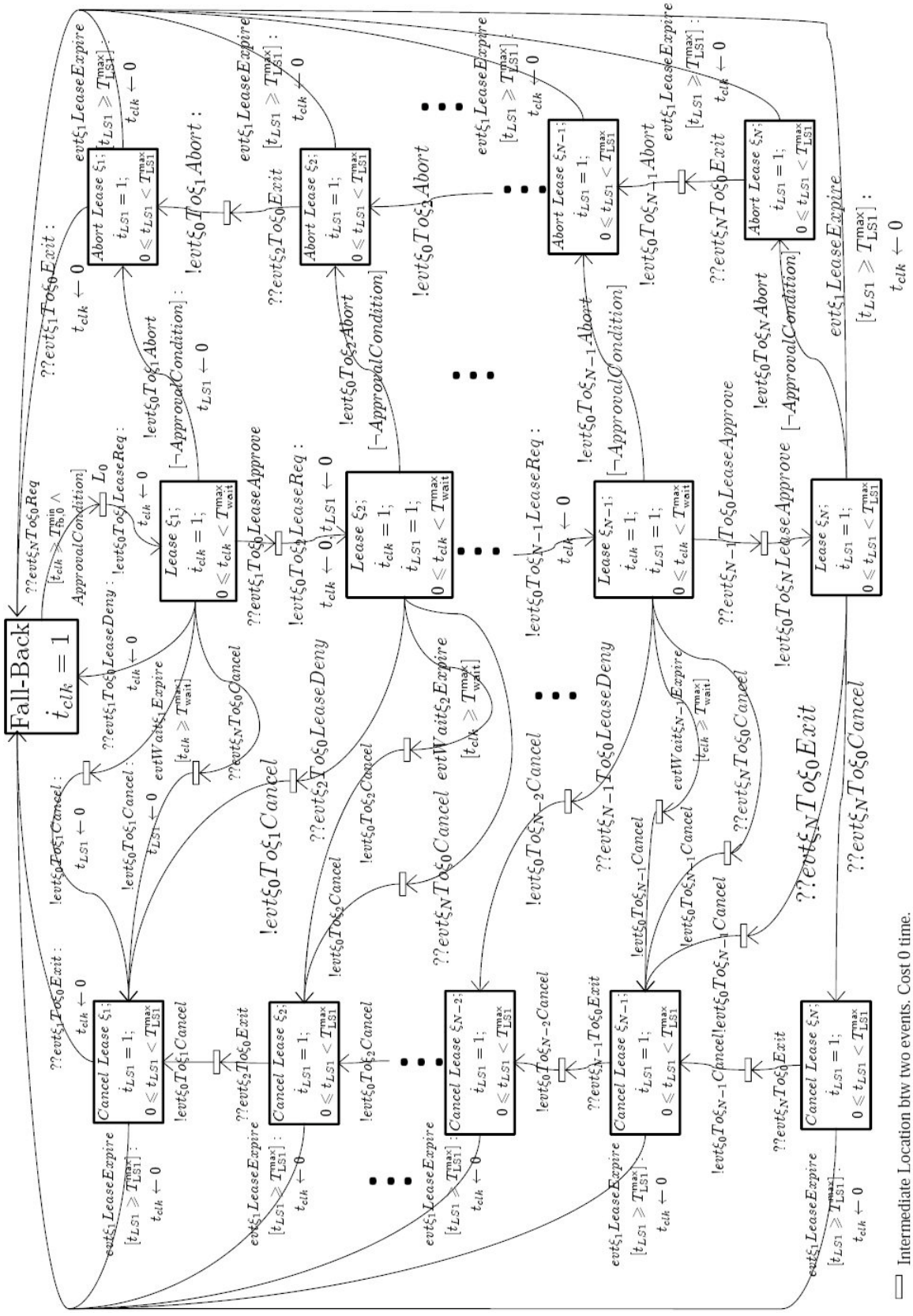


Fig. 11. Diagram of Hybrid Automaton A_{supvsr}, the Design Pattern for Supervisor. Each rectangle box indicates a location. Inside the box, the first line is the location's name (note state variable names and location names are local to their respective hybrid automata; hence two distinct locations of two distinct hybrid automata may have the same name). Annotations to each edge (aka transition) comply with the following conventions. Before the ':' are the synchronization label and the guard formula (quoted by brackets "[]") for the edge. After the ':' are the data state value resets ("$x \leftarrow a$" means "assigning x of value a"). The above notational conventions also apply to Fig. 12 and 13.

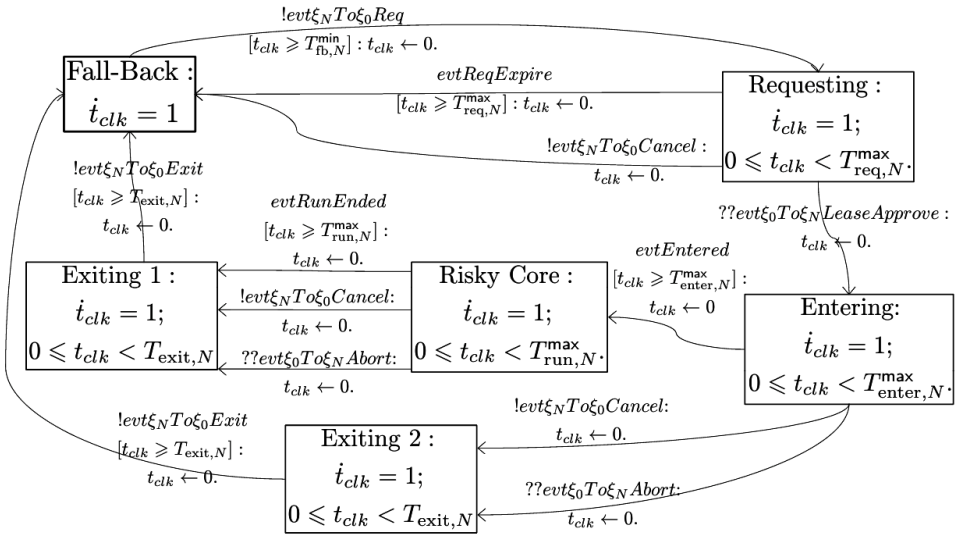


Fig. 12. Diagram of Hybrid Automaton A_{initzr}, the Design Pattern for Initializer

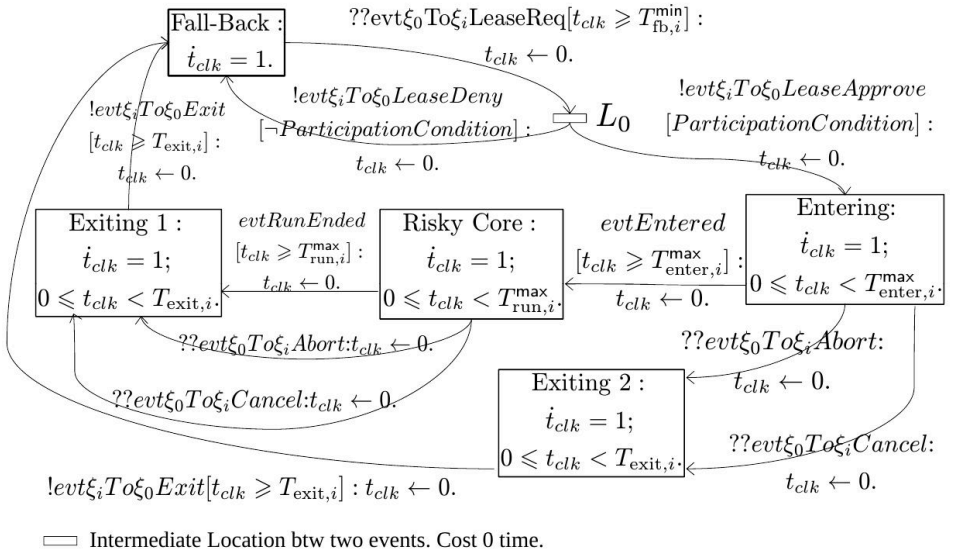


Fig. 13. Diagram of Hybrid Automaton $A_{\text{ptcpnt},i}$, the Design Pattern for the ith Participant.

way,

$$\forall t \in (t_0, t_1) \cdot \text{no } evt\xi_0To\xi_1LeaseReq \text{ happens at } t. \quad (3)$$

We then have

Claim 2.1: $\forall i \in \{1, 2, \ldots, N-1\}$, throughout interval $[t_0, t_1)$, ξ_i can respectively enter its location (set) "L0", "Entering", {"Risky Core", "Exiting 1"}, and "Exiting 2" (see Fig. 13) for at the most once, and continuously dwell there for no more than 0, $T^{\max}_{\text{enter},i}$, $T^{\max}_{\text{run},i}+T_{\text{exit},i}$, and $T_{\text{exit},i}$ respectively.

Claim 2.2: Throughout interval $[t_0, t_1)$, ξ_N can respectively enter its location (set) "Entering", {"Risky Core", "Exiting 1"}, and "Exiting 2" (see Fig. 12) for at the most once, and continuously dwell there for no more than $T^{\max}_{\text{enter},N}$, $T^{\max}_{\text{run},N} + T_{\text{exit},N}$, and $T_{\text{exit},N}$ respectively.

Proof: Due to Lemma 1, we have $\forall i \in \{0, 1, \ldots, N\} \cdot \ell_i(t_0^-) =$ "Fall-Back".

$\forall i \in \{1, 2, \ldots, N-1\}$, for ℓ_i to enter "L0" twice in interval $[t_0, t_1)$, $evt\xi_0To\xi_iLeaseReq$ must happen twice (see Fig. 13), which implies $evt\xi_0To\xi_1LeaseReq$ happen twice (see Fig. 11), which implies $\exists t_2 \in (t_0, t_1) \cdot evt\xi_0To\xi_1LeaseReq$ happens at t_2. This contradicts Formula (3) (note due to c1, $T^{\min}_{\text{fb},0} > 0$, hence no zeno can

happen). The continuous dwelling time upper bound of 0 follows naturally due to "L0"'s dwelling constraint.

Same reasoning also applies to location (set) "Entering", {"Risky Core", "Exiting 1"}, and "Exiting 2". Hence Claim 2.1 is proven.

Same way we can prove Claim 2.2, where $evt\xi_0To\xi_N LeaseApprove$ replaces $evt\xi_0To\xi_i LeaseReq$. ■

Furthermore, we have the following lemma.

Lemma 3 (PTE Compliance): Let t_0, t_1 ($t_0 < t_1$) be the time instances when two consecutive $evt\xi_0To\xi_1 LeaseReq$ happen; or let t_0 be the last time $evt\xi_0To\xi_1 LeaseReq$ happens and $t_1 = \infty$. In both cases, $\forall t \in [t_0, t_1)$, Ineq. (1) sustains.

Proof: Without loss of generality, let us first focus on entity ξ_i and ξ_{i-1} (where $i \in \{2, 3, \ldots, N-1\}$).

If ξ_i never entered risky-locations throughout interval $[t_0, t_1)$, then the PTE ordering of $\xi_{i-1} < \xi_i$ trivially sustains.

Otherwise, there must be a maximal interval $[t_2, t_3) \subseteq [t_0, t_1)$ that ℓ_i stays in risky-locations. That is, $evtEntered$ happens at t_2, $evt\xi_iTo\xi_0 Exit$ happens at $t_3 < t_1$ (no matter $t_1 < \infty$ or $t_1 = \infty$, we know $t_3 < t_1$ due to Lemma 1 and 2), and $\forall t \in (t_2, t_3) \cdot \ell_i(t) \in$ {"Risky Core", "Exiting 1"}.

Due to Lemma 2, (t_2, t_3) is the only maximal interval in $[t_0, t_1)$ that ℓ_i stays in risky-locations. Due to Lemma 1, $\ell_i(t_0^-) =$ "Fall-Back". By exhaustively examining all possible paths in $A_{\text{ptcpnt},i}$ of Fig. 13, we have

$$t_0 + T^{\max}_{\text{enter},i} \leq \quad t_2 \quad \leq t_0 + (i-1)T^{\max}_{\text{wait}} + T^{\max}_{\text{enter},i}; \tag{4}$$

$$\text{and } t_2 + T_{\text{exit},i} \leq \quad t_3 \quad \leq t_2 + T^{\max}_{\text{run},i} + T_{\text{exit},i}. \tag{5}$$

Now let us check the duration that ξ_{i-1} may stay in its risky-locations within $[t_0, t_1)$.

By exhaustively examining all possible paths in $A_{\text{ptcpnt},i}$ of Fig. 13, for ξ_i's $evtEntered$ to happen at t_2, $evt\xi_iTo\xi_0 LeaseApprove$ must happen at

$$t_4 = t_2 - T^{\max}_{\text{enter},i}. \tag{6}$$

Note $t_4 \geq t_0$ because of Ineq. (4). According to A_{supvsr} of Fig. 11, because $\ell_0(t_0^-) =$ "Fall-Back", for $evt\xi_iTo\xi_0 LeaseApprove$ to happen at $t_4 \in [t_0, t_2)$, $evt\xi_{i-1}To\xi_0 LeaseApprove$ must have happened at some time instance t_5, where

$$t_0 \leq t_5 \leq t_4. \tag{7}$$

Due to Condition c5,

$$\begin{aligned} & t_5 + T^{\max}_{\text{enter},i-1} + T^{\min}_{\text{risky}:i-1\to i} \\ < \quad & t_4 + T^{\max}_{\text{enter},i} = t_2 \quad \text{(due to Eq. (6))}. \end{aligned} \tag{8}$$

On the otherhand, because after t_0, the first occurrence of $evt\xi_iTo\xi_0 Exit$ happens at t_3, by exhaustively checking all possible paths in A_{supvsr} of Fig. 11, this implies the following proposition.

Proposition 1: No $evt\xi_0To\xi_j Abort$ nor $evt\xi_0To\xi_j Cancel$ ever happened during $[t_0, t_3)$ ($\forall j = i-1, i-2, \ldots, 1$). □

Because of Proposition 1, Ineq. (7)(8) imply that after ℓ_{i-1} enters location "Entering" at t_5, it enters "Risky Core", i.e. risky-locations, at

$$t_6 = t_5 + T^{\max}_{\text{enter},i-1}. \tag{9}$$

Due to Ineq. (8), we have

$$t_6 < t_2 - T^{\min}_{\text{risky}:i-1\to i} < t_3. \tag{10}$$

On the other hand, from Fig. 11, we see

$$t_5 \geq t_4 - T^{\max}_{\text{wait}}. \tag{11}$$

Ineq. (11) and Condition c6 together imply

$$\begin{aligned} & t_5 + T^{\max}_{\text{enter},i-1} + T^{\max}_{\text{run},i-1} \\ > \quad & t_4 + T^{\max}_{\text{enter},i} + T^{\max}_{\text{run},i} + T_{\text{exit},i} \\ = \quad & t_2 + T^{\max}_{\text{run},i} + T_{\text{exit},i} \quad \text{(due to Eq. (6))} \\ \geq \quad & t_3. \quad \text{(due to Ineq. (5))} \end{aligned} \tag{12}$$

Due to Proposition 1, during $[t_5, t_3)$, ξ_{i-1} never receives $evt\xi_0To\xi_{i-1}Abort$ or $evt\xi_0To\xi_{i-1}Cancel$. This fact, combined with Ineq. (12), implies the earliest time instance after t_5 that ξ_{i-1} may receive $evt\xi_0To\xi_{i-1}Abort$ or $evt\xi_0To\xi_{i-1}Cancel$ is t_3. Let $t_7 \in [t_0, t_1)$ be the time instance that ℓ_{i-1} exits risky-locations, then

$$\begin{aligned} t_7 \quad \geq \quad & \min\{t_5 + T^{\max}_{\text{enter},i-1} + T^{\max}_{\text{run},i-1} + T_{\text{exit},i-1}, \\ & \quad t_3 + T_{\text{exit},i-1}\} \\ = \quad & t_3 + T_{\text{exit},i-1} \quad \text{(due to Ineq. (12))} \\ > \quad & t_3 + T^{\min}_{\text{safe}:i\to i-1} \quad \text{(due to Condition c7)} \end{aligned} \tag{13}$$

Note Ineq. (10)(13) implies $t_6 < t_7$; and no matter $t_1 < \infty$ or $t_1 = \infty$, Lemma 1 and 2 imply $t_7 < t_1$. Therefore, to summarize, ξ_{i-1} must have visited risky-locations for once during interval $[t_0, t_1)$; and the visit starts at t_6, and ends at t_7, where t_6 compiles with Ineq. (10), and t_7 complies with Ineq. (13). In other words, the PTE ordering of $\xi_{i-1} < \xi_i$ sustains during interval $[t_0, t_1)$.

Using the same approach, we can also prove $\xi_{N-1} < \xi_N$ during $[t_0, t_1)$. ■

With the above lemmas, we can prove Theorem 1 as follows.

Proof of Claim 1:

Throughout the execution of hybrid system $\mathcal{H}$, if $evt\xi_0To\xi_1 LeaseReq$ never happens, as all entities are initialized from "Fall-Back" locations, the PTE safety rules trivially sustain.

If infinite number of $evt\xi_0To\xi_1 LeaseReq$ happen. Then before the first $evt\xi_0To\xi_1 LeaseReq$, all entities stay in "Fall-Back" locations, PTE safety rules trivially sustain. After that, between any two consecutive $evt\xi_0To\xi_1 LeaseReq$, due to Lemma 1 and 2, PTE Safety Rule 1 sustains, and a risky-locations continuous dwelling time upper bound is $T^{\max}_{\text{run},i} + T^{\max}_{\text{exit},i}$ for ξ_i

($i = 1, 2, \ldots, N$); due to Lemma 3, PTE Safety Rule 2 also sustains. Therefore, PTE safety rules sustain.

The same proving approach can be applied to the scenario where finite number of $evt\xi_0To\xi_1LeaseReq$ happen (we need to check the special case: the interval between the last occurrence of $evt\xi_0To\xi_1LeaseReq$ and time ∞; but the same proving approach can be applied, and the conclusion is the same). ■

Proof of Claim 2.i:

We can prove by contradiction. Suppose in any duration of length $T^{\mathsf{min}}_{\mathrm{fb},N} + T^{\mathsf{max}}_{\mathrm{req},N}$ in $[t_0, +\infty)$, ξ_N never get a chance to send $evt\xi_NTo\xi_0Req$. Then ξ_0 can never leave location "Fall-Back" in $[t_0, +\infty)$ (see Fig. 11, 12 in Appendix C of [13]); hence can never send $evt\xi_0To\xi_NLeaveApprove$; hence ξ_N can never leave the location set of {"Fall-Back", "Requesting"} in $[t_0, +\infty)$. Then in any duration $[t_a, t_b]$ of length $T^{\mathsf{min}}_{\mathrm{fb},N} + T^{\mathsf{max}}_{\mathrm{req},N}$ in $[t_0, +\infty)$, suppose

a) at t_a, ξ_N resides in "Fall-Back", then by $t_a + T^{\mathsf{min}}_{\mathrm{fb},N} \in [t_a, t_b]$, ξ_N get the chance to send $evt\xi_NTo\xi_0Req$, contradiction reached;

b) at t_a, ξ_N resides in "Requesting", then by $t_a + T^{\mathsf{max}}_{\mathrm{req},N} \in [t_a, t_b]$, ξ_N must have returned to "Fall-Back", suppose the return time instance is t_c, then $t_a \le t_c \le t_a + T^{\mathsf{max}}_{\mathrm{req},N} \le t_b$, then by $t_c + T^{\mathsf{min}}_{\mathrm{fb},N} \le t_a + T^{\mathsf{max}}_{\mathrm{req},N} + T^{\mathsf{min}}_{\mathrm{fb},N} = t_b$, ξ_N get the chance to send $evt\xi_NTo\xi_0Req$, also reached contradiction. ■

Proof of Claim 2.ii:

According to Supervisor's hybrid automata A_{supvsr} (see Fig. 11 in Appendix C of [13]), every location other than "Fall-Back" has a dwelling time upper bound. By checking all possible paths of A_{supvsr}, we know that ξ_0 can continuously stay away from "Fall-Back" for at the most $T_{\mathrm{reset},0} \overset{\mathsf{def}}{=} (N-1)T^{\mathsf{max}}_{\mathrm{wait}} + T^{\mathsf{max}}_{\mathrm{LS1}}$. Therefore, once ξ_0 non-zeno-ly leaves "Fall-Back" at t_{00}, ξ_0 will return to "Fall-Back" by $t^+_{00} + T_{\mathrm{reset},0}$. That is, $\exists t_a \in (t^+_{00}, t^+_{00} + T_{\mathrm{reset},0}]$, such that ξ_0 first returns to "Fall-Back" at t_a.

Meanwhile, according to Initializer's hybrid automata A_{initzr} (see Fig. 12 in Appendix C of [13]), every location other than "Fall-Back" has a dwelling time upper bound. By checking all possible paths of A_{initzr}, we know that ξ_N can continuously stay away from "Fall-Back" for at the most $T_{\mathrm{reset},N} \overset{\mathsf{def}}{=} T^{\mathsf{max}}_{\mathrm{req},N} + T^{\mathsf{max}}_{\mathrm{enter},N} + T^{\mathsf{max}}_{\mathrm{run},N} + T_{\mathrm{exit},N}$.

Therefore, $\exists t_b \in [t_a, t_a + T_{\mathrm{reset},N} + T^{\mathsf{min}}_{\mathrm{fb},N}]$, such that t_b is the first time instance in $[t_a, t_a + T_{\mathrm{reset},N} + T^{\mathsf{min}}_{\mathrm{fb},N}]$ that ξ_N have continuously resided in "Fall-Back" for at least $T^{\mathsf{min}}_{\mathrm{fb},N}$. In other words, t_b is the first time instance in $[t_a, t_a + T_{\mathrm{reset},N} + T^{\mathsf{min}}_{\mathrm{fb},N}]$ that ξ_N can send $evt\xi_NTo\xi_0Req$ to ξ_0.

As at t_a, ξ_0 resides in "Fall-Back", and ξ_0 cannot leave "Fall-Back" unless receiving $evt\xi_NTo\xi_0Req$ from ξ_N. Suppose ξ_N sends ξ_0 event $evt\xi_NTo\xi_0Req$ at t_b, and ξ_0 receives the event, then this will trigger ξ_0 to send $evt\xi_0To\xi_1LeaseReq$ at t_b. Then according to Lemma 1, we can infer that at t_b^-, all entities ($\xi_0 \sim \xi_N$) reside in "Fall-Back". As all entities' residing location at t_b^- is not determined by events happening in t_b, therefore, even if ξ_0 does not receive $evt\xi_NTo\xi_0Req$ at t_b, we can still conclude that at t_b^-, all entities are residing in "Fall-Back".

As $t_b \ge t_a > t^+_{00}$, we have $t_b^- > t_{00}$.

As $t_b \le t_a + T^{\mathsf{min}}_{\mathrm{fb},N} + T_{\mathrm{reset},N} \le t^+_{00} + T_{\mathrm{reset},0} + T^{\mathsf{min}}_{\mathrm{fb},N} + T_{\mathrm{reset},N} = t^+_{00} + T_{\mathrm{reset}}$, we have $t_b^- \le t_{00} + T_{\mathrm{reset}}$.

Therefore, we conclude that $\exists t \in (t_{00}, t_{00} + T_{\mathrm{reset}}]$, such that all entities ($\xi_0 \sim \xi_N$) return to location "Fall-Back" at t (where $t = t_b$). ■

APPENDIX E
FORMAL DESCRIPTION ON ATOMIC ELABORATION OF HYBRID AUTOMATON

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[7].

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 $A' = (\vec{x}'(t), V', \mathsf{inv}', F', E', g', R', L', \mathsf{syn}', \Phi_0')$, 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 $i \ne j$, 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.
4. A has no time-convergent paths, no timelock locations, and no zeno paths.

7. For the aforementioned design pattern hybrid automata A_{supvsr}, A_{initzr}, and $A_{\mathrm{ptcpnt},i}$, as long as Condition c1 ∼ 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.

Let us first describe the intuition on how to *elaborate* a given hybrid automaton at one location with one child 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).

The above intuitive methodology can be formalized as follows.

Formal Description on Atomic Elaboration of Hybrid Automaton:

The formal description assumes the following.

1. Given $\vec{x}^a = (x^a_1, x^a_2, \ldots, x^a_n) \in \mathbb{R}^n$ and $\vec{x}^b = (x^b_1, x^b_2, \ldots, x^b_m) \in \mathbb{R}^m$, $(\vec{x}^a, \vec{x}^b) \stackrel{\mathsf{def}}{=} (x^a_1, x^a_2, \ldots, x^a_n, x^b_1, x^b_2, \ldots, x^b_m) \in \mathbb{R}^{n+m}$.
2. Given an n-element tuple $X = (x_1, x_2, \ldots, x_n)$, we use $X|_i$ ($i \in \{1, 2, \ldots, n\}$) to denote X's ith element: x_i. We use $\mathsf{elements}(X) \stackrel{\mathsf{def}}{=} \{x_1, x_2, \ldots, x_n\}$ to denote the set of all elements of X.

Under the above assumptions, the formal description on how to carry out atomic elaboration of hybrid automaton runs as follows.

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 steps.

1. $\vec{x}''(t) \stackrel{\mathsf{def}}{=} (\vec{x}(t), \vec{x}'(t))$; denote A and A''s dimensions as respectively n and n', then $\vec{x}''(t) \in \mathbb{R}^{n+n'}$.
2. $V'' \stackrel{\mathsf{def}}{=} (V \cup V') \backslash \{v\}$.
3. $\forall u \in V \backslash \{v\}$, $\mathsf{inv}''(u) \stackrel{\mathsf{def}}{=} \mathsf{inv}(u) \times \mathsf{inv}'(v')$, where "$\times$" means Cartesian product, v' is an arbitrary location in V'; $\forall u \in V'$, $\mathsf{inv}''(u) \stackrel{\mathsf{def}}{=} \mathsf{inv}(v) \times \mathsf{inv}''(u)$.
4. $F'' \stackrel{\mathsf{def}}{=} \{f''_u | f''_u : \mathbb{R}^{n+n'} \mapsto \mathbb{R}^{n+n'}, \forall u \in V''\}$ such that at location $u \in V''$, we have
   4.1. if $u \in V \backslash \{v\}$, then
   $$f''_u\big((x_1, x_2, \ldots, x_n, x'_1, x'_2, \ldots, x'_{n'})\big)|_i \stackrel{\mathsf{def}}{=} \begin{cases} f_u\big((x_1, x_2, \ldots, x_n)\big)|_i & \\ & \text{(when } 1 \le i \le n\text{)}, \\ 0 & \text{(otherwise)}; \end{cases}$$
   4.2. if $u \in V'$, then
   $$f''_u\big((x_1, x_2, \ldots, x_n, x'_1, x'_2, \ldots, x'_{n'})\big)|_i \stackrel{\mathsf{def}}{=} \begin{cases} f_v\big((x_1, x_2, \ldots, x_n)\big)|_i & \\ & \text{(when } 1 \le i \le n\text{)}, \\ f'_u\big((x'_1, x'_2, \ldots, x'_{n'})\big)|_{i-n} & \text{(otherwise)}. \end{cases}$$
5. $L'' \stackrel{\mathsf{def}}{=} L \cup L'$.
6. E'', g'', R'', syn'' are created according to the following process.
   6.1. Initially $E'' = \varnothing$.
   6.2. For each $e = (v_1, v_2) \in E$, where $v_1, v_2 \neq v$ (hence $v_1, v_2 \in V''$), we add edge $e'' = (v_1, v_2)$ into E''. Furthermore, we define $g''(e'') \stackrel{\mathsf{def}}{=} g(e) \times \mathbb{R}^{n'}$;
   $$r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \stackrel{\mathsf{def}}{=} r_e\big((s_1, s_2, \ldots, s_n)\big) \times \{(s'_1, s'_2, \ldots, s'_{n'})\}, \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v_1);$$
   and $\mathsf{syn}''(e'') \stackrel{\mathsf{def}}{=} \mathsf{syn}(e)$.
   6.3. For each $e = (v_1, v) \in E$, where $v_1 \neq v$ (hence $v_1 \in V''$), we add for each $v' \in \Phi'_0|_{V'}$ (i.e. Φ'_0's projection on V') an edge $e'' = (v_1, v')$ into E''. Furthermore, we define $g''(e'') \stackrel{\mathsf{def}}{=} g(e) \times \mathbb{R}^{n'}$;
   $$r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \stackrel{\mathsf{def}}{=} r_e\big((s_1, s_2, \ldots, s_n)\big) \times \{(s'_1, s'_2, \ldots, s'_{n'})\}, \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v_1);$$
   and $\mathsf{syn}''(e'') \stackrel{\mathsf{def}}{=} \mathsf{syn}(e)$.
   6.4. For each $e = (v, v_2) \in E$, where $v_2 \neq v$ (hence $v_2 \in V''$), we add for each $v' \in V'$ an edge $e'' = (v', v_2)$ into E''. Furthermore, we define $g''(e'') \stackrel{\mathsf{def}}{=} g(e) \times \mathbb{R}^{n'}$;
   $$r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \stackrel{\mathsf{def}}{=} r_e\big((s_1, s_2, \ldots, s_n)\big) \times \{(s'_1, s'_2, \ldots, s'_{n'})\}, \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v');$$
   and $\mathsf{syn}''(e'') \stackrel{\mathsf{def}}{=} \mathsf{syn}(e)$.
   6.5. For each $e = (v, v) \in E$, we add for each $v' \in V'$ an edge $e'' = (v', v')$ into E''. Furthermore,

we define $g''(e'') \stackrel{\text{def}}{=} g(e) \times \mathbb{R}^{n'}$;

$$\begin{aligned} & r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \\ \stackrel{\text{def}}{=} \; & r_e\big((s_1, s_2, \ldots, s_n)\big) \times \{(s'_1, s'_2, \ldots, s'_{n'})\}, \\ & \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v'); \end{aligned}$$

and $\mathsf{syn}''(e'') \stackrel{\text{def}}{=} \mathsf{syn}(e)$.

6.6. For each $e' = (v_1, v_2) \in E'$ (hence $v_1, v_2 \in V''$), we add an edge $e'' = (v_1, v_2)$ into E''. Furthermore, we define $g''(e'') \stackrel{\text{def}}{=} \mathbb{R}^n \times g'(e')$;

$$\begin{aligned} & r''_{e''}\big((s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'})\big) \\ \stackrel{\text{def}}{=} \; & \{(s_1, s_2, \ldots, s_n)\} \times r'_{e'}\big((s'_1, s'_2, \ldots, s'_{n'})\big); \\ & \forall (s_1, s_2, \ldots, s_n, s'_1, s'_2, \ldots, s'_{n'}) \in \mathsf{inv}''(v_1); \end{aligned}$$

and $\mathsf{syn}''(e'') \stackrel{\text{def}}{=} \mathsf{syn}'(e)$.

7. Φ''_0 is created according to the following process.
   7.1. Initially $\Phi''_0 = \varnothing$.
   7.2. For each $(v_1, \vec{s}) \in \Phi_0$, where $v_1 \neq v$, we add $(v_1, (\vec{s}, \underbrace{0, 0, \ldots, 0}_{n' \text{ zeros}}))$ into Φ''_0.
   7.3. For each $(v, \vec{s}) \in \Phi_0$, we add for each $(v', \vec{s}') \in \Phi'_0$ a state value $(v', (\vec{s}, \vec{s}'))$ into Φ''_0.
8. For PTE CPS, there is the issue of partitioning V'' into safe-locations V''^{safe} and risky-locations V''^{risky}. In case $v \in V^{\text{safe}}$, $V''^{\text{safe}} \stackrel{\text{def}}{=} V' \cup (V^{\text{safe}} \backslash \{v\})$ and $V''^{\text{risky}} \stackrel{\text{def}}{=} V'' \backslash V''^{\text{safe}}$; otherwise, $V''^{\text{risky}} \stackrel{\text{def}}{=} V' \cup (V^{\text{risky}} \backslash \{v\})$ and $V''^{\text{safe}} \stackrel{\text{def}}{=} V'' \backslash V''^{\text{risky}}$.

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

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

With atomic elaboration at hand, 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

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

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^0_1, v^0_2, …, $v^0_{k_0} \in V_{\text{supvsr}}$, and simple hybrid automata A^0_1, A^0_2, …, $A^0_{k_0}$, such that A_{supvsr} and A^0_j ($j = 1 \sim k_0$) are independent, and A'_0 elaborates A_{supvsr} at v^0_1, v^0_2, …, $v^0_{k_0}$ with A^0_1, A^0_2, …, $A^0_{k_0}$ respectively;
2. For each $i \in \{1, 2, \ldots, N-1\}$, there are distinct locations v^i_1, v^i_2, …, $v^i_{k_i} \in V_{\text{ptcpnt},i}$, and simple hybrid automata A^i_1, A^i_2, …, $A^i_{k_i}$, such that $A_{\text{ptcpnt},i}$ and A^i_j ($j = 1 \sim k_i$) are independent, and A'_i elaborates $A^i_{\text{ptcpnt},i}$ at v^i_1, v^i_2, …, $v^i_{k_i}$ with A^i_1, A^i_2, …, $A^i_{k_i}$ respectively;
3. There are distinct locations $v^N_1, v^N_2, \ldots, v^N_{k_N} \in V_{\text{initzr}}$, and simple hybrid automata $A^N_1, A^N_2, \ldots, A^N_{k_N}$, such that A_{initzr} and A^N_j ($j = 1 \sim k_N$) are independent, and A'_N elaborates A_{initzr} at v^N_1, v^N_2, …, $v^N_{k_N}$ with A^N_1, A^N_2, …, $A^N_{k_N}$ respectively;
4. Hybrid automata A^i_j 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 and liveness described in Theorem 1 Claim 1 and 2.

Proof: If not, there must be an execution trace $\phi'(t)$ (see [11] for the rigorous definition of "execution trace", aka "trajectory" of a hybrid system) of $\mathcal{H}'$ that violates PTE safety rules (liveness). 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 (liveness). This contradicts Theorem 1. ■

APPENDIX F
DETAILED DESIGN OF LEASING BASED APPROACH FOR CASE STUDIES

We start our design of laser tracheotomy wireless CPS per proposed leasing-based design approach.

First, we see the wireless laser tracheotomy 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 4.1-Participant; also see Fig. 13 for the diagram of the hybrid automaton) at location "Fall-Back" with A'_{vent}, using the elaboration method described in Section 4.3.

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.

The resulted detailed designs for the wireless laser tracheotomy entities are shown in Fig. 14, 15, and 16 respectively. Some data state variable names and/or locations names in the corresponding design patterns are modified to better reflect their meanings in laser tracheotomy.

Via the same approach, we derive the detailed designs for the wireless IP remote monitoring entities, which are shown in Fig. 17, 18, and 19 respectively. Some data state variable names and/or locations names in the corresponding design patterns are modified to better reflect their meanings in IP remote monitoring.

APPENDIX G
EXAMPLE SCENARIOS WHERE LEASING PROTECTS PTE SAFETY RULES

Let us further consider a number of typical scenarios to get better intuitions on how leasing approach works in the laser tracheotomy case study.

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). Without leasing, 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: $evt\xi_0To\xi_2Abort$, followed by $evt\xi_2To\xi_0Exit$, and followed by $evt\xi_0To\xi_1Abort$. Losing any one of these events at the receiver end will cause PTE safety rules violation. For example, losing $evt\xi_2To\xi_0Exit$, 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 $T^{\mathsf{max}}_{\text{run},2} = 20$(s) with or without surgeon's request to cancel; and the ventilator resumes within the lease $T^{\mathsf{max}}_{\text{run},1} = 35$(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. $evt\xi_2To\xi_0Cancel$) 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 $T^{\mathsf{max}}_{\text{run},1} = 35$(s) at the most, hence cannot suffocate the patient.

A third scenario involves the parameter configuration constraints. Suppose we set $T^{\mathsf{max}}_{\text{enter},2} = T^{\mathsf{max}}_{\text{enter},1} = 0$(s) (or any other value so that $T^{\mathsf{max}}_{\text{enter},2} = T^{\mathsf{max}}_{\text{enter},1}$), then because $T^{\mathsf{min}}_{\text{risky:}1\to2} = 3\text{(s)} > 0$, 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 $T^{\mathsf{min}}_{\text{risky:}1\to2} = 3$(s): that the laser-scalpel must wait for another 3(s) after the ventilator pauses, and then can it emit laser.

In summary, if we follow the proposed lease based design approach, Theorem 1 and 2 can guarantee PTE safety rules.

APPENDIX H
DETAILED DESIGN OF POLLING BASED APPROACH FOR CASE STUDIES

The detailed design state diagrams of laser tracheotomy and IP remote monitoring per polling based approach [12] are shown in Fig 20 $\sim$ 25 respectively.

In these figures, $CurrentTime$ refers to the wall clock time, $T_{period} = 50$(ms) is the polling period. For laser tracheotomy (see Fig. 20, 21, and 22), $T^{\xi_1}_{length} = 44$(s), $T^{\xi_1}_{Entering} = 3$(s), $T^{\xi_2}_{length} = 31.5$(s), $T^{\xi_2}_{Entering} = 10$(s), and $T^{\xi_2}_{Exiting} = 1.5$(s). For IP remote monitoring (see Fig. 23, 24, and 25), $T^{\xi_1}_{length} = 42$(s), $T^{\xi_1}_{Entering} = 1$(s), $T^{\xi_2}_{length} = 25.5$(s), $T^{\xi_2}_{Entering} = 3$(s), and $T^{\xi_2}_{Exiting} = 2.5$(s).

APPENDIX I
CORRIGENDA TO CONFERENCE VERSION

We found the following editing errors/omissions in the conference version of this paper [14] (in this section, unless otherwise mentioned, all reference numbers refer to [14]). These editing errors/omissions occurred in [14] when reformatting Fig.8, 9, 10 (respectively the full-fledged hybrid automata diagrams of Supervisor, Initializer, and Participant in the appendices) to the narrations of Section IV for reader's ease-of-understanding. Fortunately, all analyses and evaluations of the conference paper are still correct, as they are based on Fig.8, 9, 10 of the appendices.

1) In Fig.4 (a), the block of "Send $evt\xi_0To\xi_iCancel$. Transit to "Cancel Lease ξ_i"." missed "Set timer $t_{LS1} \leftarrow 0$ iff $i = 1$." as its first line.

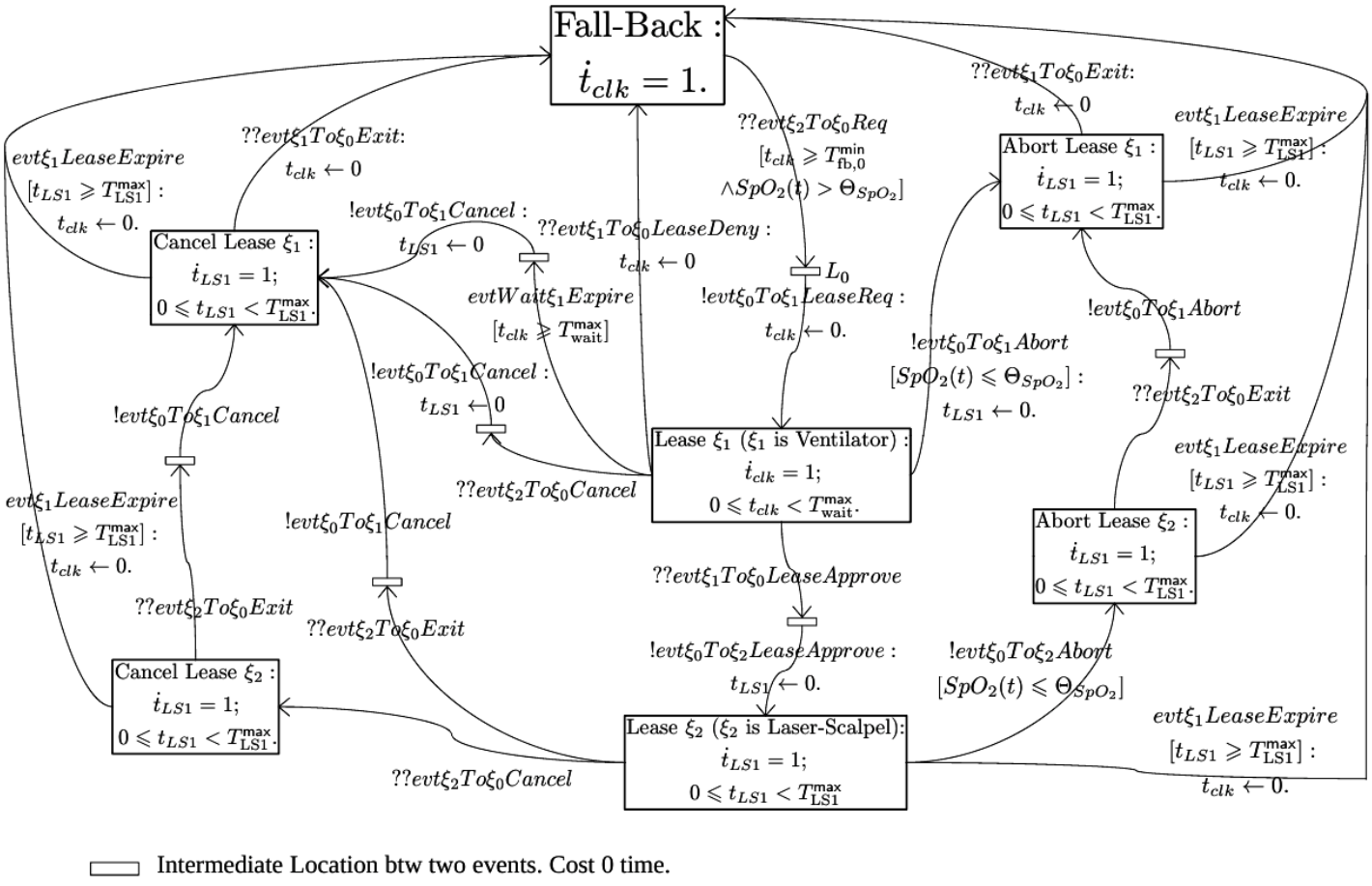


Fig. 14. Laser Tracheotomy Supervisor Detailed Design. Note entity ξ_1 refers to the ventilator, and ξ_2 refers to the laser-scalpel.

2) In Fig.4 (a), the block of "Send $\xi_0To\xi_iAbort$. Transit to "Abort Lease ξ_i"." should be "Set timer $t_{LS1} \leftarrow 0$ iff $i = 1$. Send $evt\xi_0To\xi_iAbort$. Transit to "Abort Lease ξ_i"."
3) Step 4 of the narration for Initializer, last sentence: "$evt\xi_0To\xi_NApprove$" should be "$evt\xi_0To\xi_NLeaseApprove$". Correspondingly, in Fig. 9 (Diagram of Hybrid Automaton A_{initzr}) and Fig. 12 (Laser Tracheotomy Laser-Scalpel Detailed Design) of appendices, the respective edge label of "$evt\xi_0To\xi_NApprove$" should be $evt\xi_0To\xi_NLeaseApprove$".
4) Step 7 of the narration for Initializer missed "and send event $evt\xi_NTo\xi_0Exit$" in its end. Similarly, Step 7 of the narration for Participant missed "and send event $evt\xi_iTo\xi_0Exit$" in its end.
5) In Step 5 and 6 of the narration for Participant, the two occurrences of "$evt\xi_0To\xi_NAbort$" should both be "$evt\xi_0To\xi_iAbort$".
6) In Fig. 11 (Laser Tracheotomy Supervisor Detailed Design), the outgoing transition $evtWait\xi_1Expire$ from location $Lease\xi_1$, the guard condition variable "t_{idle}" should be "t_{clk}".

We would also like to add the following notes.

1) In Fig.4, "t_{LS1} expire" means "$t_{LS1} \geq T^{\max}_{\mathrm{LS1}}$".
2) In Fig.13 of appendices, note the two edges of "$!evt\xi_1To\xi_0LeaseDeny$" actually can never happen because the guard condition is always false. This means either of the two edges can be removed, and location "$L_{0,0}$" and "$L_{0,1}$" can be merged.
3) To rigorously remove zenos in the model, it is better to add a positive minimum dwelling time constraint to both Initializer and Participant's "Fall-Back" locations.
4) Lemma 1 Claim 1.2 is not necessary to the proof; in fact, a better time upper bound can be derived during the proof of Lemma 2. Please refer to our journal publication for a better proof.

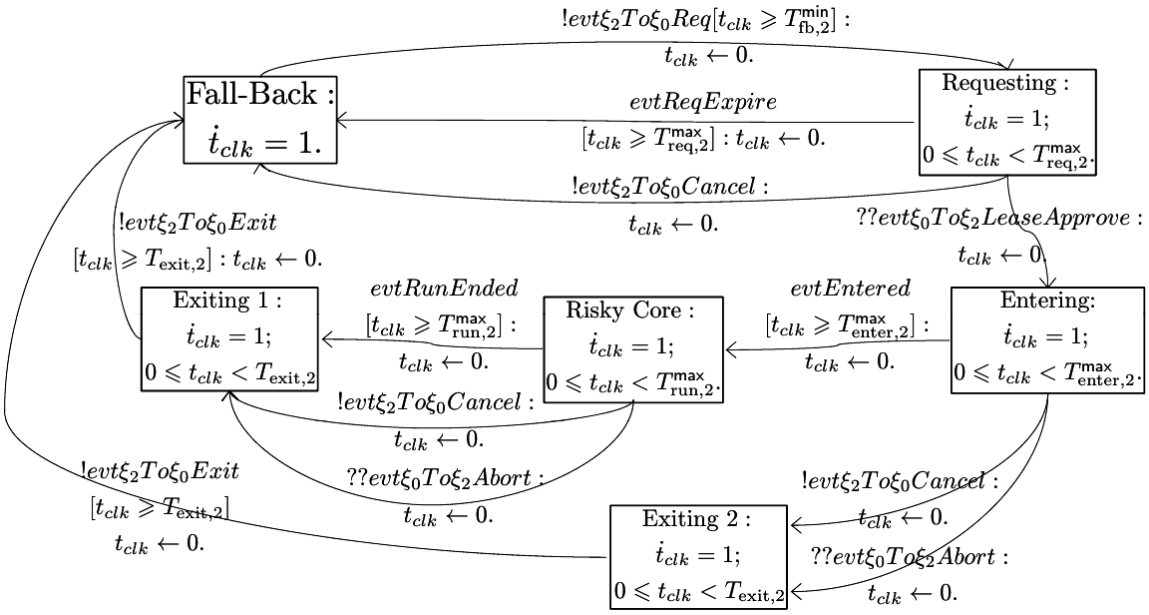


Fig. 15. Laser Tracheotomy Laser-Scalpel Detailed Design. Note the laser-scalpel emits and only emits laser when dwelling in location "Risky Core".

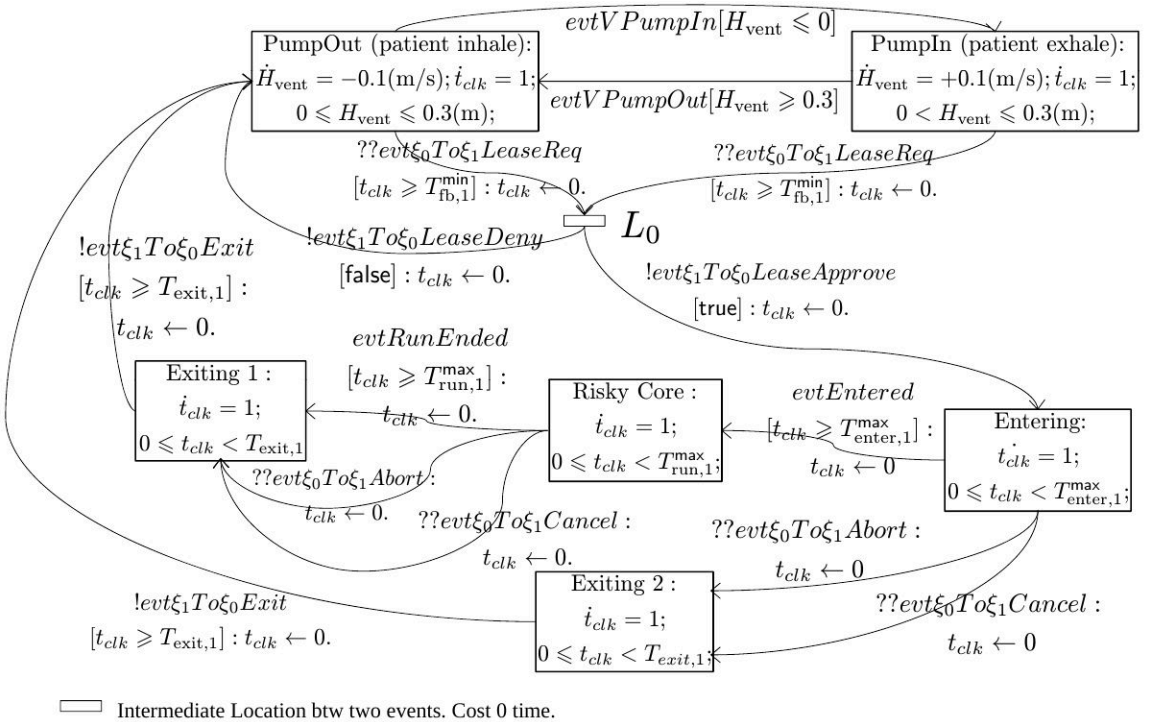


Fig. 16. Laser Tracheotomy Ventilator Detailed Design, by elaborating Participant Design Pattern.

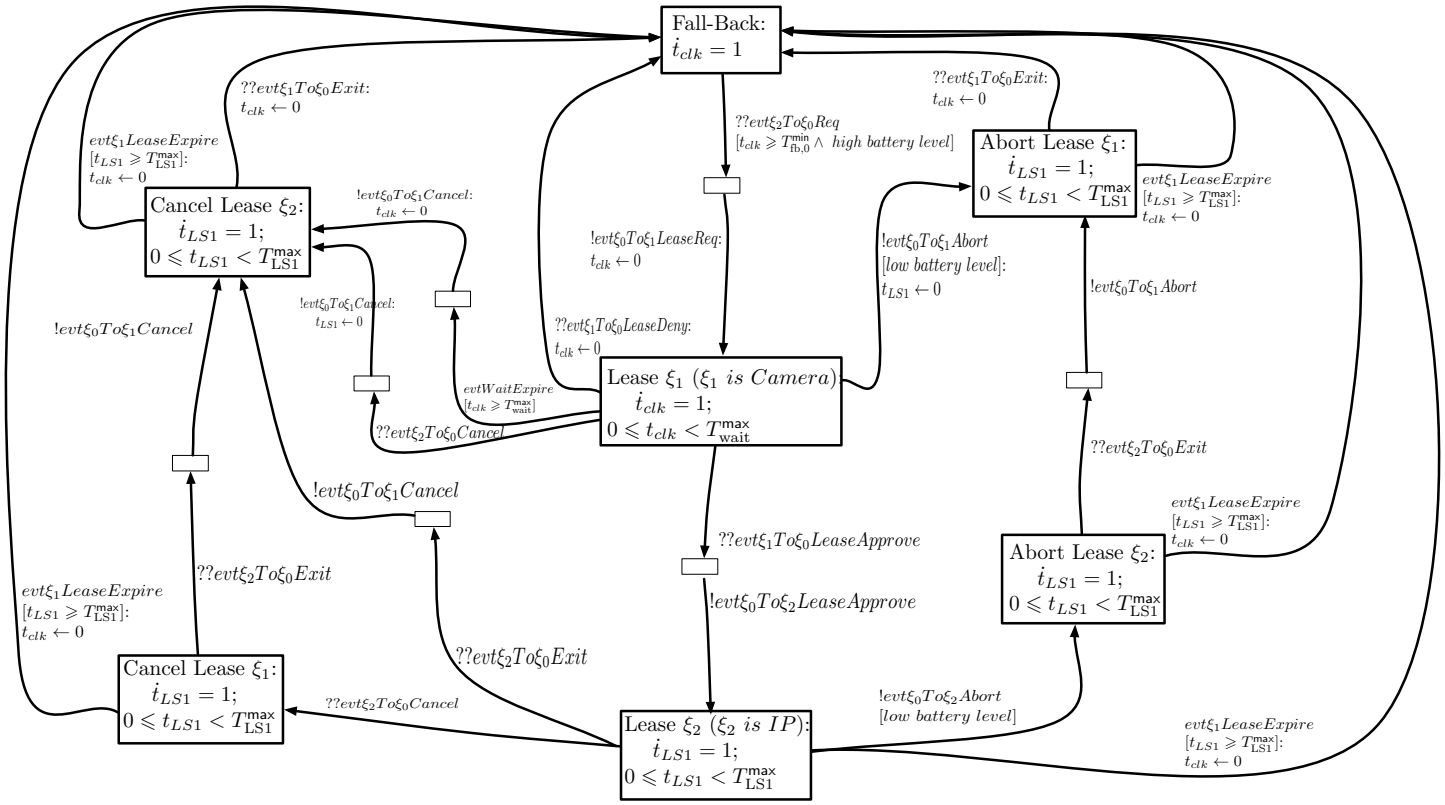


Fig. 17. IP Remote Monitoring Supervisor Detailed Design. Note entity ξ_1 refers to the camera, and ξ_2 refers to the IP.

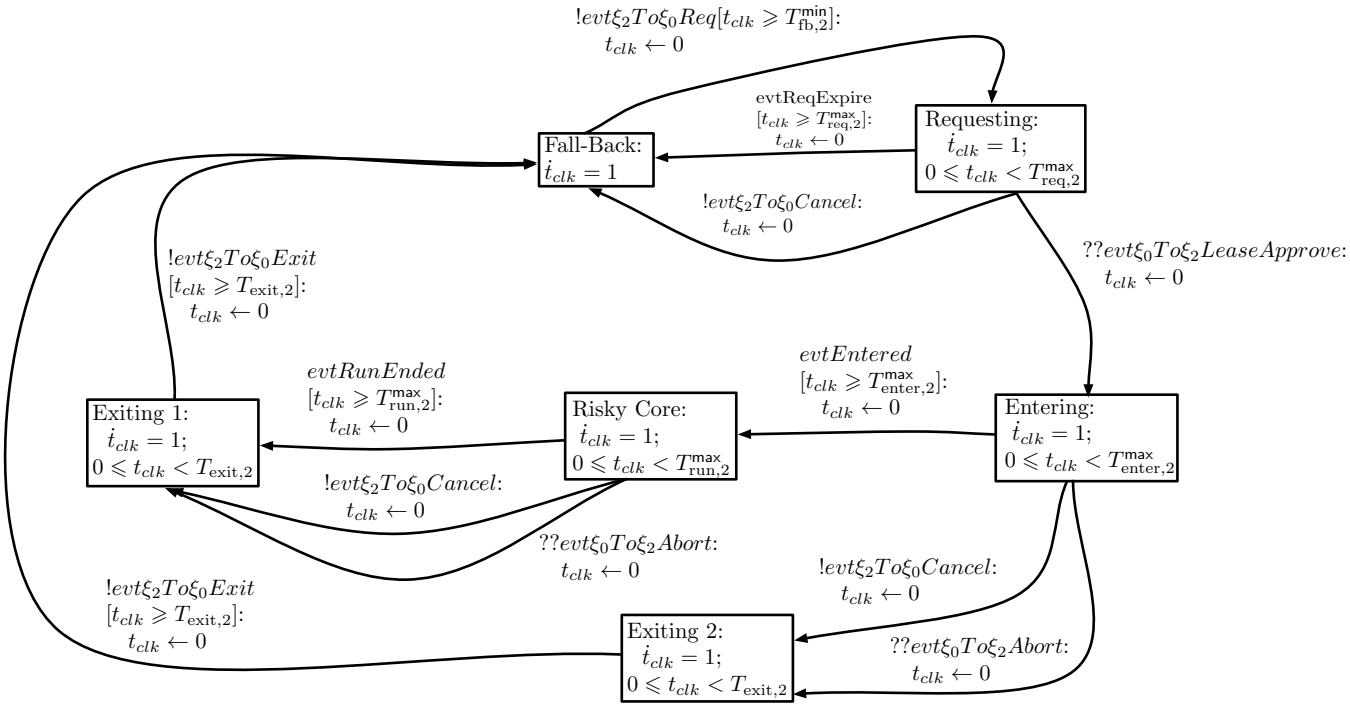


Fig. 18. IP Remote Monitoring IP Detailed Design. Note the IP conducts random walk when and only when dwelling in location “Risky Core”.

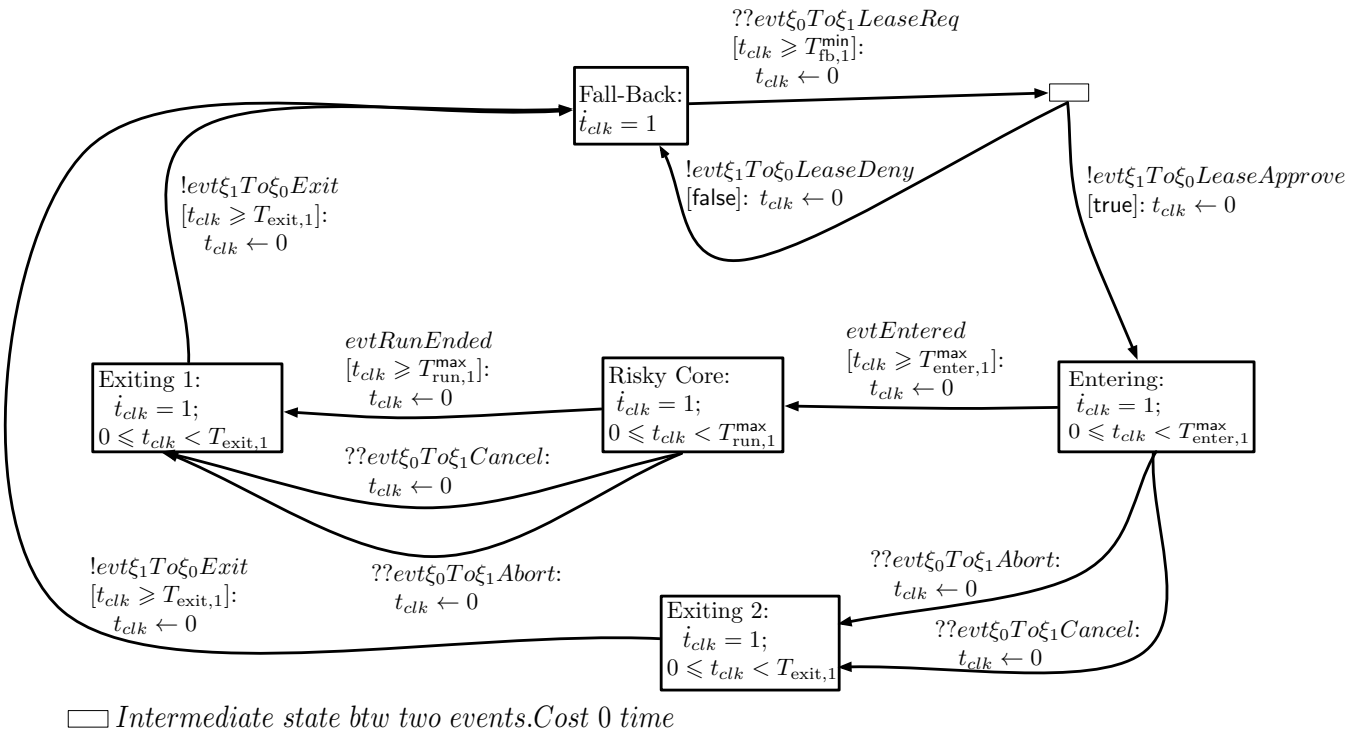


Fig. 19. IP Remote Monitoring Camera Detailed Design.

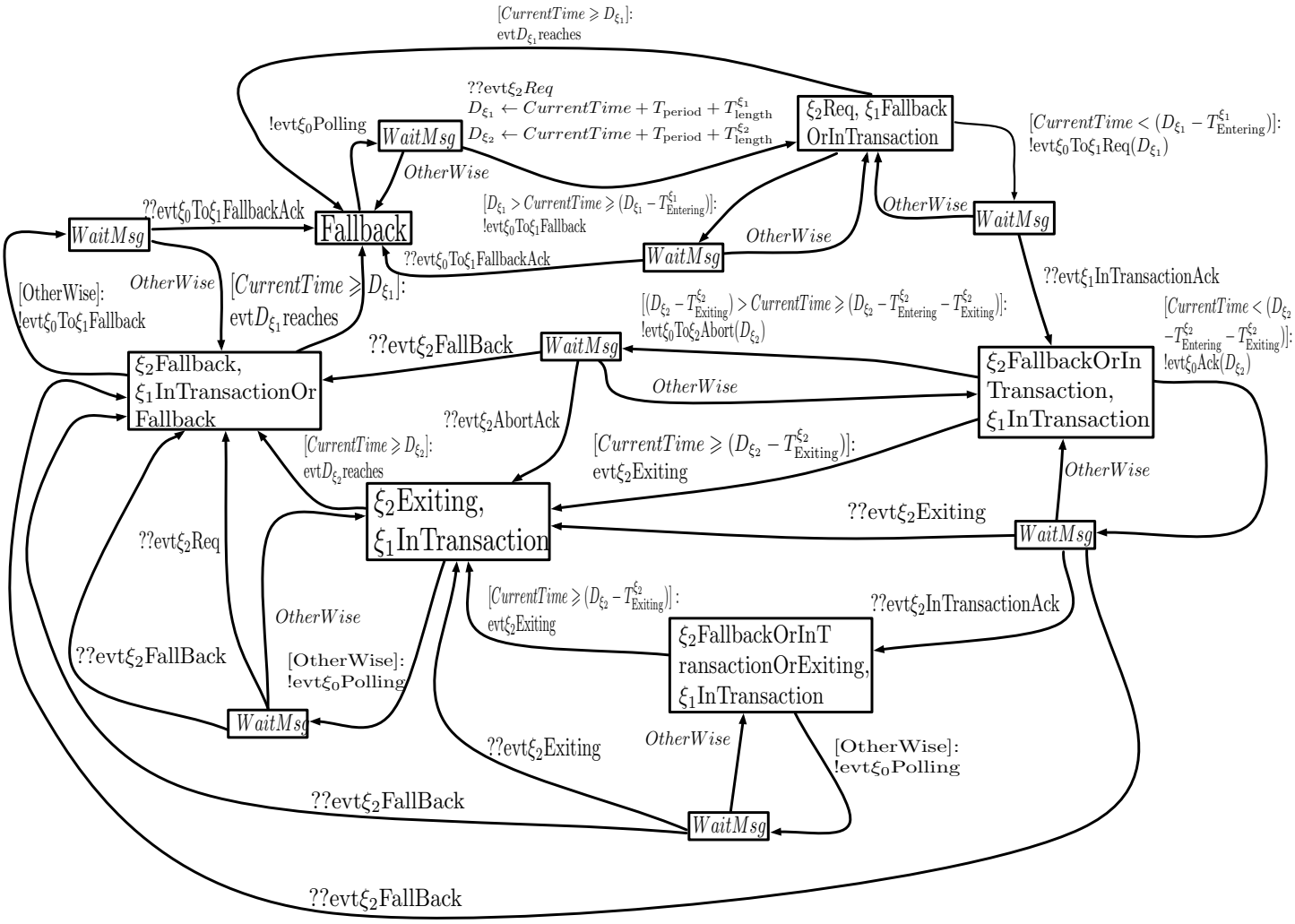


Fig. 20. Laser Tracheotomy Supervisor (aka Entity ξ_0) Detailed Design, per Kim et al [12]'s Polling-Based Approach

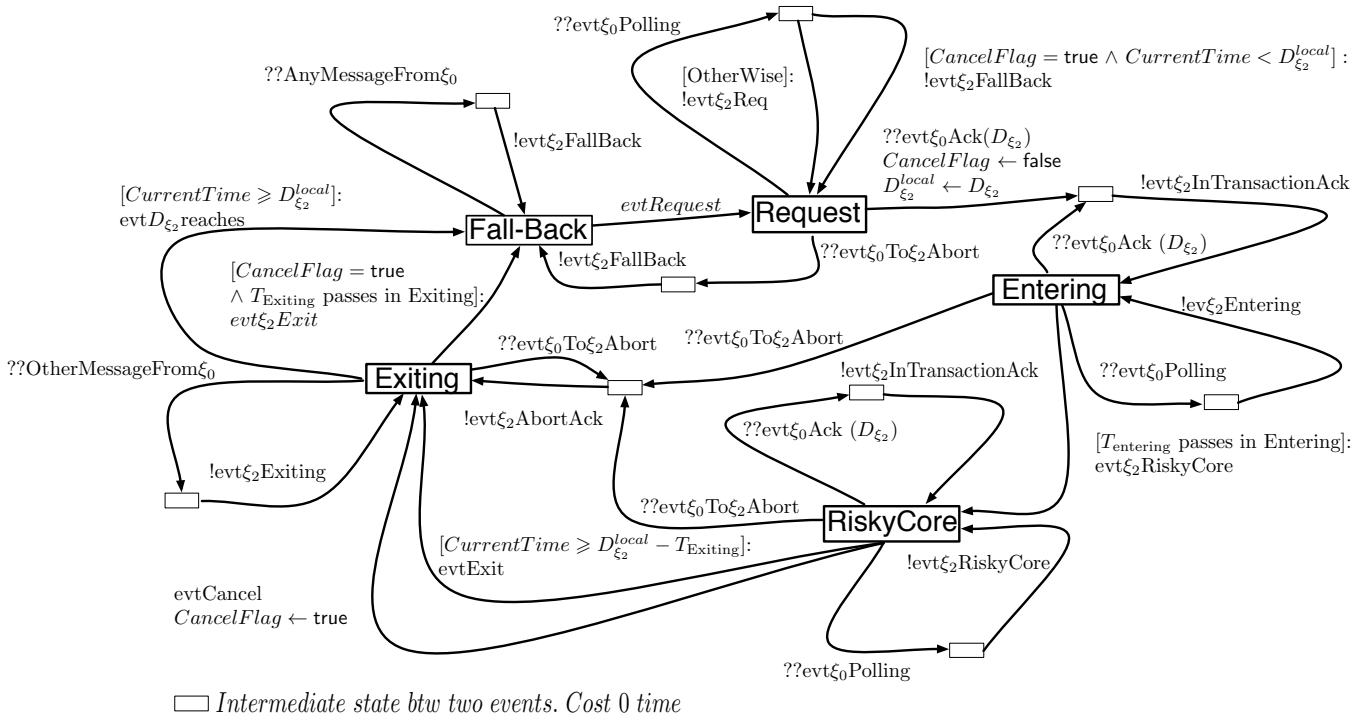


Fig. 21. Laser Tracheotomy Laser Scalpel (the Initializer, aka Entity ξ_2) Detailed Design, per Kim et al [12]'s Polling-Based Approach

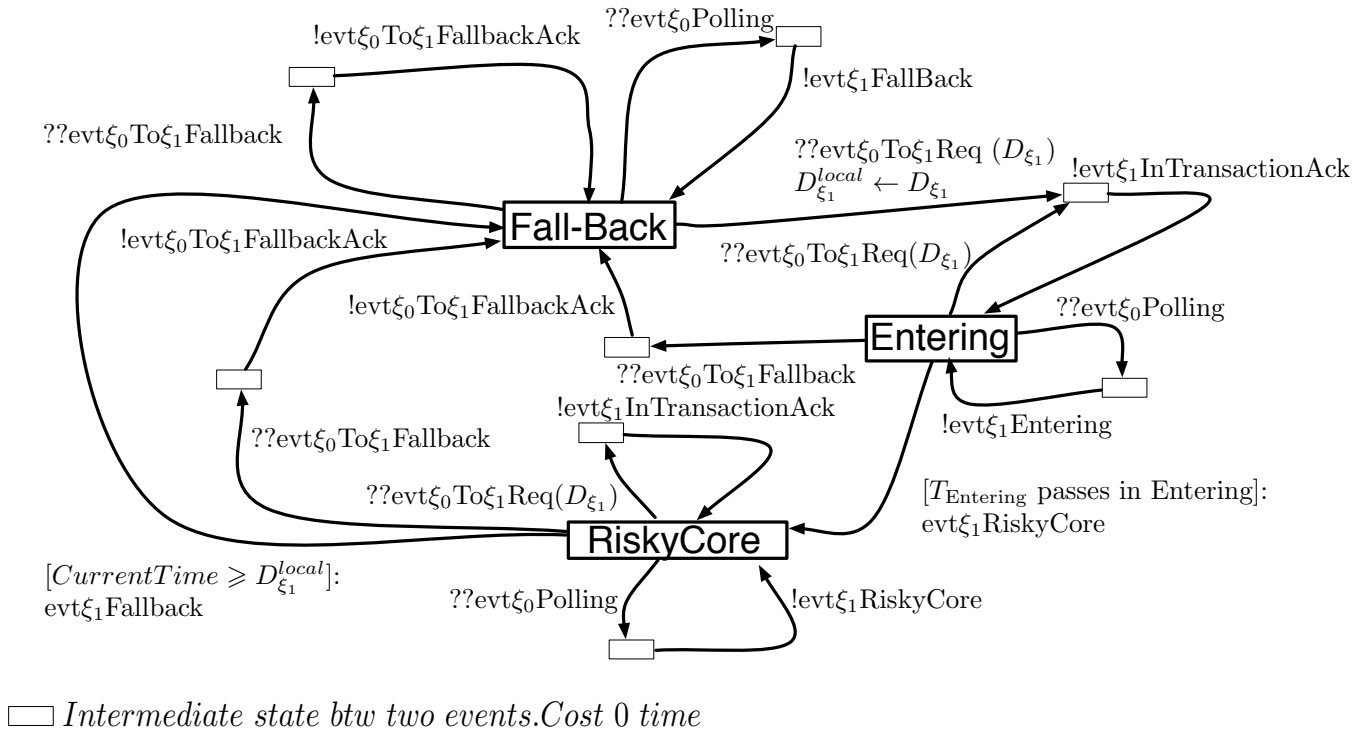


Fig. 22. Laser Tracheotomy Ventilator (the Participant, aka Entity ξ_1) Detailed Design, per Kim et al [12]'s Polling-Based Approach

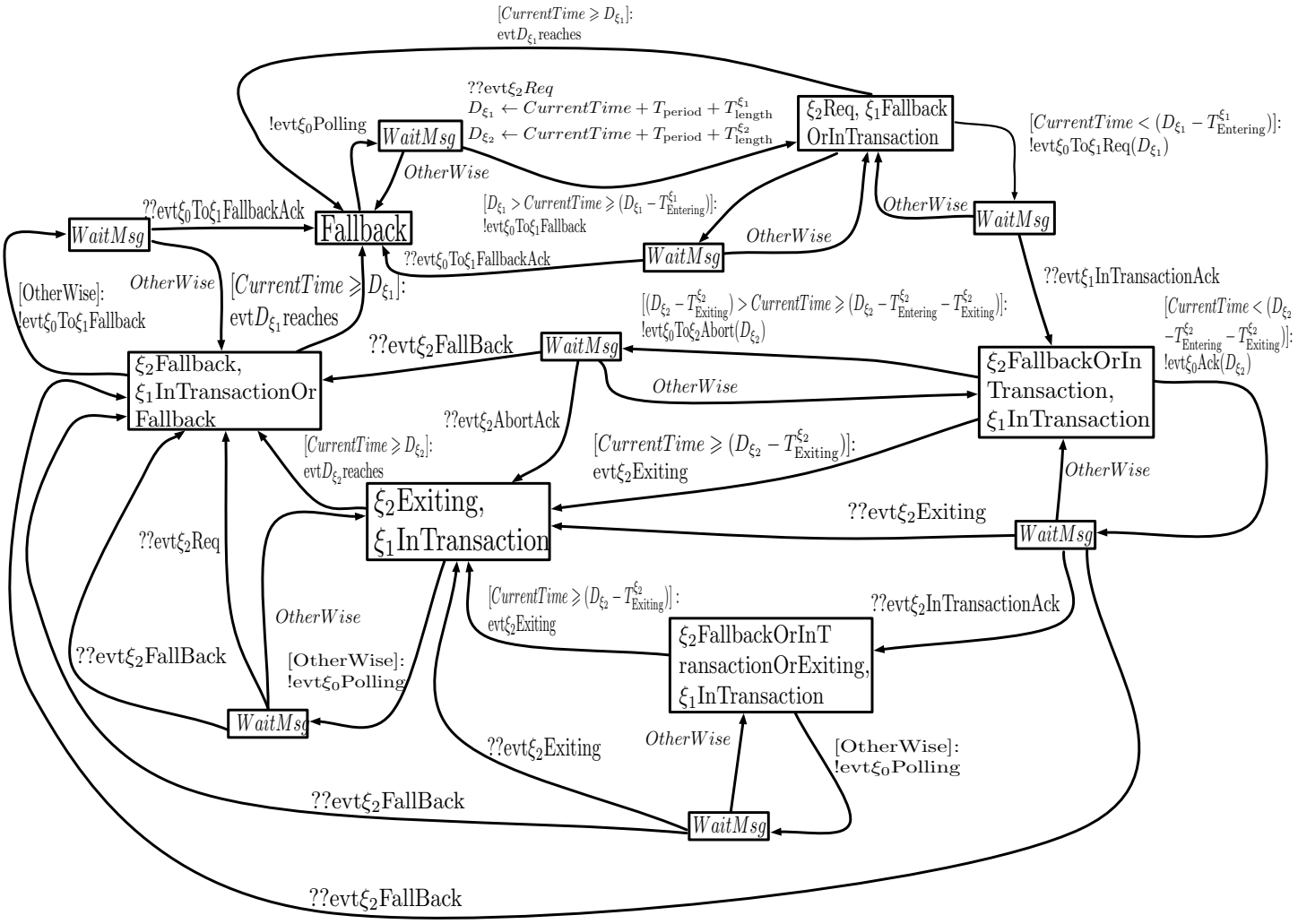


Fig. 23. IP Remote Monitoring Supervisor (aka Entity ξ_0) Detailed Design, per Kim et al [12]'s Polling-Based Approach

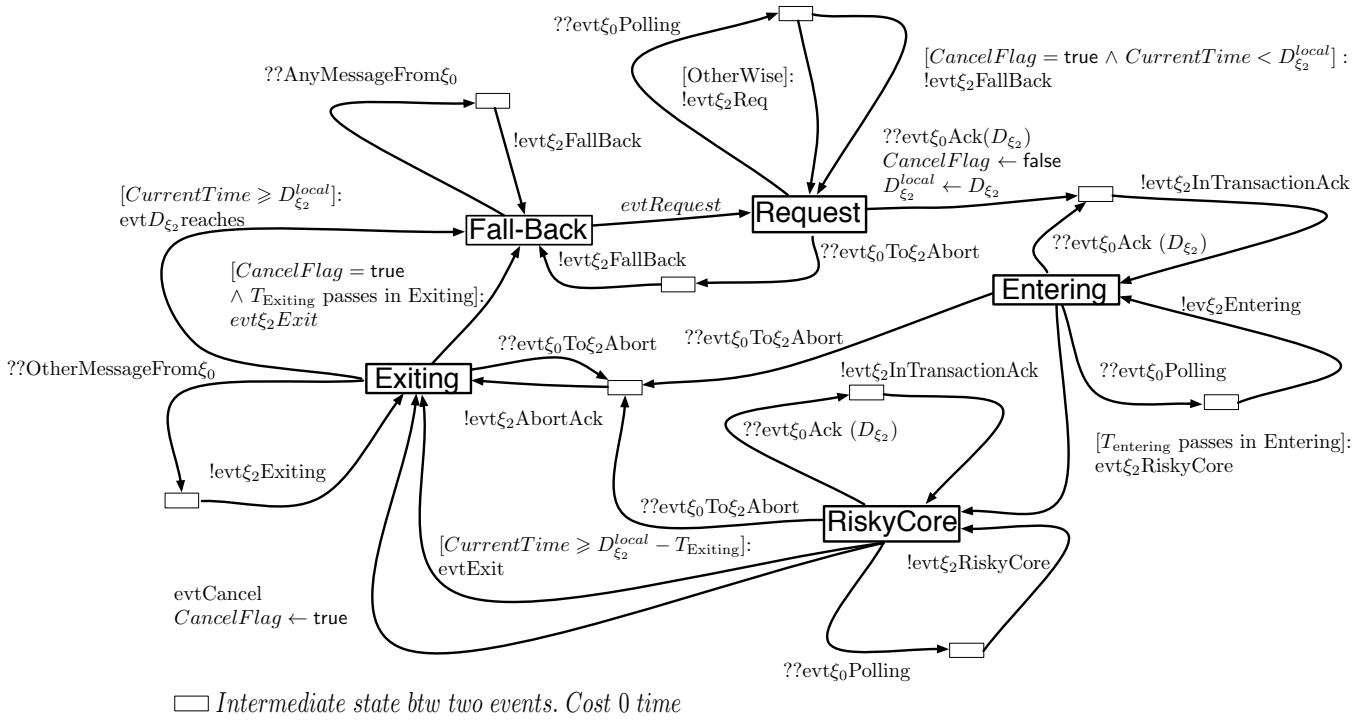


Fig. 24. IP Remote Monitoring IP (the Initializer, aka Entity ξ_2) Detailed Design, per Kim et al [12]'s Polling-Based Approach

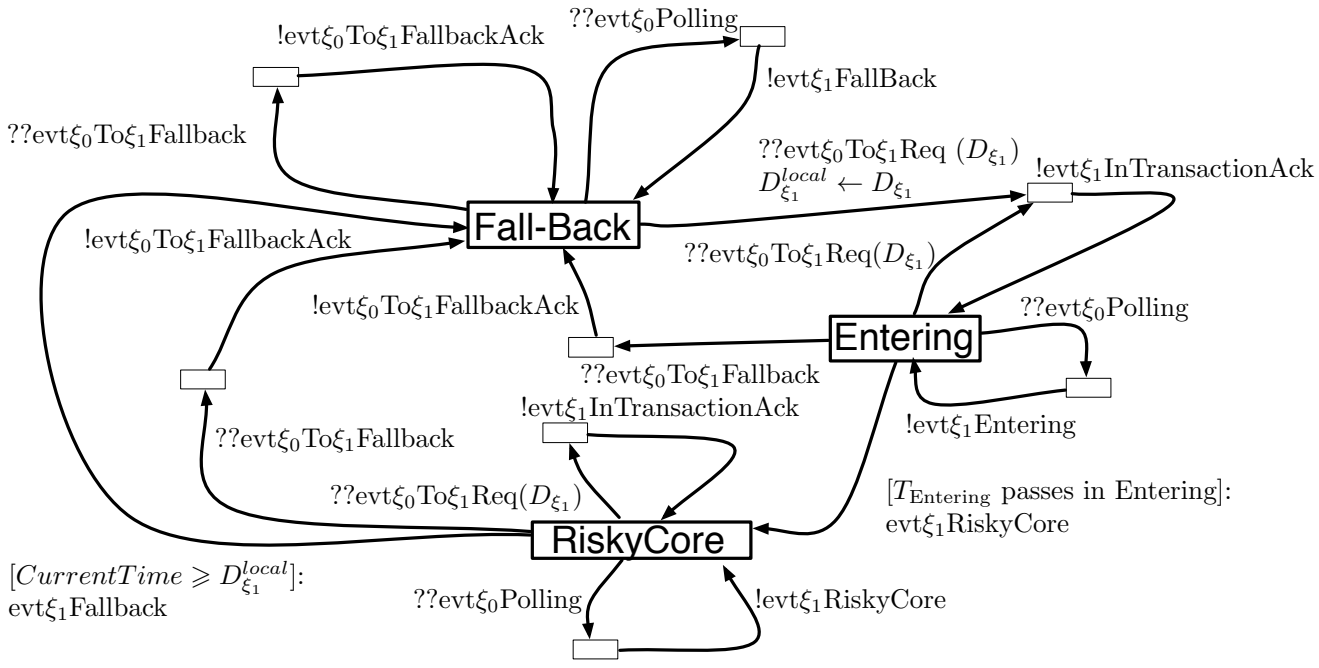


Fig. 25. IP Remote Monitoring Camera (the Participant, aka Entity ξ_1) Detailed Design, per Kim et al [12]'s Polling-Based Approach