POWVER

Technical Report 2020-15

Title: **Connection Models for the Internet-of-Things**

Authors: Kangli He, Holger Hermanns, Hengyang Wu, Yixiang Chen

Report Number: 2020-15

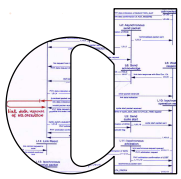
ERC Project: Power to the People. Verified.

ERC Project ID: 695614

Funded Under: H2020-EU.1.1. – EXCELLENT SCIENCE

Host Institution: Universität des Saarlandes, Dependable Systems and Software
Saarland Informatics Campus

Published In: Frontiers of Computer Science 14



POWER TO THE PEOPLE.
VERIFIED.
erc

Connection Models for the Internet-of-Things

Kangli HE [1], **Holger HERMANNS** [2], **Hengyang WU** [1], **Yixiang CHEN** (✉)[1]

1 MoE Engineering Research Center for Software/Hardware Co-design Technology and Application, East China Normal University, 200062 Shanghai, China
2 Saarland University, Saarland Informatics Campus, 66123 Saarbrücken, Germany

Abstract The Internet-of-Things (IoT) is expected to swamp the world. In order to study and understand the emergent behaviour of connected things, effective support for their modelling is needed. At the heart of IoT are flexible and adaptive connection patterns between things, which can naturally be modelled by channel-based coordination primitives, and characteristics of connection failure probabilities, execution and waiting times, as well as resource consumption. The latter is especially important in light of severely limited power and computation budgets inside the things. In this paper, we tackle the IoT modelling challenge, based on a conservative extension of channel-based Reo circuits. We introduce a model called *Priced Probabilistic Timed Constraint Automaton*, which combines models of probabilistic and timed aspects, and integrates pricing information. An expressive logic called *priced probabilistic timed scheduled data stream logic* is presented, so as to enable the specification and verification of properties, which characterize dataflow streams and prices. A small but illustrative IoT case demonstrates the principal benefits of the proposed approach.

Keywords IoT, Reo, cost, time, probability, automaton.

1 Introduction

As one of the most prominent emerging technologies, the Internet of Things (IoT) is expected to change the world as we know it. The IoT connects not only traditional end devices in the internet, but also more general physical things, such as a robot that can sense the outer world and give an independent interaction. Any system composed of such things and the associated coordination and communication infrastructure such as the internet, real-time systems or local networks can be considered as fractions of IoT.

The essential characteristics of IoT systems cover *real-time behaviour* (e.g., a smart pen is able to indicate to the outside ink tank to stop refilling no more than 0.01s after reaching its maximum capacity bound), *probabilistic behaviour* (e.g., the chance of the ink tank failing to release ink is at most 5% after a request from the pen) and *non-deterministic behaviour* (e.g., once the ink tank is empty, one can either refill the tank or replace with a new tank). These characteristics are partly present in ordinary distributed systems. But in addition, *power considerations* (e.g., the initial electric power budget of a smart pen is 30 Kilo Joule) is at the core of IoT systems, because of their predominantly wireless nature. Other *resources* (e.g., the memory) are also severely limited due to portability and budget considerations. These aspects are central to IoT systems, in comparison with most other distributed systems. As mentioned by Borgia [1], *self-organization* and *flexibility* are general features of IoT systems. In other words, despite the diverse, often intricate and topologically varying phenotypes, we find the need for *flexible and adaptive connection patterns* between things being at the heart of IoT. These are meant to enable advanced as well as radically new emerging functionalities and components. To ease their development, effective support for the modelling of IoT systems is forthwith needed.

Furthermore, the diversity on research areas and professional abilities of stakeholders (which for instance consist of researchers, project leaders, company managers etc.) in IoT systems makes an important motivation for a lightweight for-

E-mail: *yxchen@sei.ecnu.edu.cn*

mal notation that, on the one hand, provides an easy interface for design engineers (which can be any stakeholder) and, on the other hand, supports the description of quantitative IoT system aspects.

The formal treatment of IoT behaviour was pioneered by Lanese et al. [2] and later taken up by Lanotte et al. [3], employing approaches based on process calculi. This line of work focusses on compositionality principles, networking primitives, and their theoretical underpinning, but without considering quantitative notions like time, probability or cost. The work presented here instead puts all those aspects in its joint focus, so as to make formal verification possible for IoT systems.

Arbab [4] proposed a coordination language Reo, which offers a powerful glue language for implementation of coordination mechanisms via connectors, based on a calculus of mobile channels. This naturally yields a friendly and easy-understanding user interface. The extensions of non-function/quantitative requirements have been studied, for instance by Meng et al. [5] and Arbab et al. [6]. IoT systems consist of distributed *components* (i.e., physical things) coordinated with each other under different distributed *communication mechanisms*, and are maintained by resources. One of the most important aspects in IoT is the data-acquisition-driven interaction among such components which can naturally be viewed as exogenous interactions. In this paper, we focus on that exogenous interactions in IoT systems, leaving the interesting interior mechanisms of each component for later research. The Reo language appears as a convenient lightweight formalism, where *physical things* are represented as *nodes* and *communication mechanisms* give rise to *channels* or *connectors*.

When exploring this approach, it soon becomes apparent that extensions to the original Reo syntax and underlying semantics are needed to better fit relevant aspects of the IoT domain. In IoT systems, resource consumption is pivotal for mission accomplishment and self-sustainability. This asks for matching support in the language to specify such aspects, in the form of a notation for reward or cost decorations of the model, together with appropriate extensions of the semantic model. This extension enables analyses of several other features including, for instance, the amount of tasks finished within a given duration. The present paper develops such an extension, and places it into the system context of IoT.

On the semantic level, Reo has a basic semantics defined in terms of so-called Constraint Automaton (CA) [7] with several variants [8]. It is understood how to include nondeterministic aspects in CA, as well as timed aspects – forming Timed Constraint Automaton (TCA) [9] – and probabilistic aspects – forming so-called Probabilistic Constraint Automaton (PCA) [10]. Since these aspects exist naturally and are of crucial relevance to interesting properties in the IoT domain, our approach is based on these extensions.

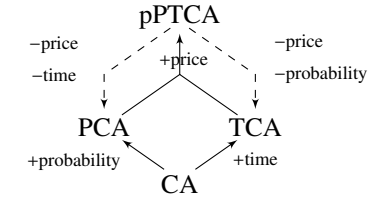


Fig. 1 Pedigree of pPTCA.

1.1 Our contribution

In order to enable a faithful modelling of IoT systems, we propose a model called *Priced Probabilistic Timed Constraint Automaton* (pPTCA). pPTCA combines features to represent nondeterministic, probabilistic, and timed aspects with aspects of energy consumption (or any kind of resource consumption considered important). The model is constructed on the basis of PCA and TCA, and it combines these two models together with dedicated support for cost aspects. The extension is strictly conservative in the sense that all existing PCA and TCA models are valid pPTCA models as well, and pPTCA is more expressive and capable to represent timed phenomena (which PCA can not), probabilistic phenomena (which TCA can not) and cost considerations (which PCA and TCA both can not). The relationship between our model and the base models (i.e., CA, TCA and PCA) is shown in Figure 1.

This work constitutes the first approach joining probabilistic, timed and priced aspects effectively in a single model within the domain of Reo and Constraint Automata. It notably goes beyond a mere combination and conservative extension, by an elaboration that provides compact and consistent value on both theorem and practice. With our proposed method, one can straightforwardly generate the IoT systems without much learning efforts, and work on the semantic models for further formal verification.

Due to the data-acquisition-driven feature in IoT systems, interesting properties for the Reo or pPTCA models are to be constructed with relation to data flows. For instance, ‘a sequence of data monitored through input/output ports with specific patterns, the values of which form a Fibonacci sequence, are required in the system’. Moreover, the pricing information is also another essential component in the prop-

erties, like 'the highest energy cost is within the battery storage'. In this paper, we extend the traditional PCTL* logic [11] with reward structure, to model the pricing information, and so-called pTsD expressions, to model the data flows. In virtue of pPTDL, properties involving probabilistic, timed, priced and data-flow features can be expressed integrally and verified for pPTCA models.

1.2 Organisation of the paper

We first review a primer of Reo circuits, and the semantic model of constraint automata together with the variants adding timed and probabilistic aspects to it in Section 2. Section 3 presents the details of Reo with priced, probabilistic and timed features and its semantic constraint automata model. Section 4 gives the semantics of pPTCA in terms of probabilistic times systems with reward structures. The pPTDL logic are introduced in Section 5. A case study of an IoT scenario is presented in Section 6, demonstrating the expressiveness and principal benefits of our model in such domains. Section 7 presents the related work. Section 8 concludes this paper.

This paper is an extended and substantially revised version of a paper published at COMPSAC [12].

2 Preliminaries

2.1 A primer of Reo

Reo is a channel-based exogenous coordination model where complex coordinators for component instances, called *connectors*, are compositionally built out of simpler ones. Reo is entirely exogenous in that only the communication and coordination among components are taken into account without considering the inner activities or communications of each component. The basic connectors are a set of *channels* with well-defined behaviour. Each channel has two *channel ends*, which can be seen as ports through which data items can enter or leave the channel. The channel ends are classified as *source* ends, providing input data items into the channel through write operations, and *sink* ends, taking data items from the channel by read operations. Reo generalizes this channel notion by allowing arbitrary channel ends according to different channel types with user-defined semantics, as shown partly in Figure 2. Synchronous channels require that write operations at source ends synchronize with matching read operations at sink ends, whilst asynchronous channels (e.g., FIFO-1, t-timer) do not. A classical channel type is

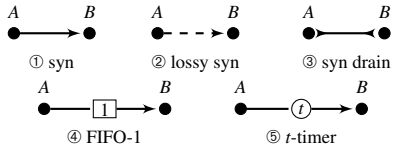


Fig. 2 Some basic Reo channels.

a *synchronous channel* (i.e., Figure 2.①), which has a source end and a sink end. The write operations at its source end and the matching read operations at its sink end are restricted to succeed only simultaneously. Other two variants of synchronous channels are introduced. A *lossy synchronous channel* (i.e., Figure 2.②) allows write operations on the source end are always enabled. If no matching read operations on the sink node, then the input data items are lost. Otherwise, it behaves like a standard synchronous channel. A *synchronous drain* (i.e., Figure 2.③) has two source ends, so no read operations can be taken to obtain the input data items. The write operations on two source ends are required to be synchronized, and both written values are lost. FIFO-1 *channel* (i.e., Figure 2.④) is a typical type of asynchronous channel with one buffer cell. It has a source end and a sink end, and the box in the middle stands for the buffer cell. The buffer can be initiated empty or with a data item defined by users (here is the latter case). The write operations on source end can only succeed if the buffer is empty, while the read operations on sink end can only fire when some data item is stored in the buffer. A t-timer (i.e., Figure 2.⑤) is triggered by some write operation on the source end, and outputs time-out after exact t time units.

A complex connector is built out of basic channels of arbitrary types by so-called *topological operations*, namely join and hide. The result is a graphical representation, called a Reo *circuit*. Every basic channel is straightforwardly one Reo circuit. The compositional framework provides features of composability and dynamic reconfigurability in Reo. The nodes of a Reo circuit are considered as pair-wise disjoint and non-empty sets of channel ends. The edges represent the connecting channels. For a node A, $Src(A)$ denotes the set of source ends coinciding on A, and $Snk(A)$ denotes the set of coincident sink ends. A node A in Reo circuit is classified as a *source node* (where $Src(A) \neq \emptyset$ and $Snk(A) = \emptyset$), a *sink node* (where $Src(A) = \emptyset$ and $Snk(A) \neq \emptyset$), or a *mixed node* (where $Src(A) \neq \emptyset$ and $Snk(A) \neq \emptyset$). A write operation on one node succeeds only if *all* the coincident source ends accept the data item. A read operation on one node succeeds if and only if at least *one* of the coincident sink ends offers

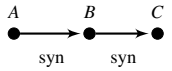


Fig. 3 A complex Reo circuit.

suitable data items, and only one is selected nondeterministically. The join operation on two nodes with the same name creates a new node (with the same name) and combines all channel ends coincident on original ones. We use hide operations to encapsulate mixed nodes inside a circuit, making them invisible and unaccessible to the environment.

Since flexible and adaptive communication and interaction patterns in IoT systems are determined by topological characteristics, they can naturally be modelled by channel-based Reo circuits.

Example 1. As in Figure 3, a complex Reo circuit is built by join operation on two synchronous channels AB and BC on the common node B. Nodes A, B, C are source, mixed and sink nodes respectively. If we take hide operation on the mixed node B, then we get a new Reo circuit of one synchronous channel AC, where B is invisible to the environment.

2.2 Constraint Automata

The semantics of Reo can be defined in terms of relations on *timed data streams* (TDSs) [13]. In [7], Baier et al. introduced *Constraint Automaton* (CA) as an operational semantics for describing the behaviour of Reo circuits. They relate the languages of these automata with TDSs, where CA accept TDS-tuples rather than strings, as for ordinary automata. CA are variants of labelled transition systems where transitions are labelled with pairs $\langle N, g\rangle$ instead of action labels, where $N \subseteq \mathcal{N}$ stands for a finite set of nodes and g is a boolean condition on the observed data items. The nodes play the role of input and output ports of components and connectors to model several channels gluing together. The locations of a CA refer to the network configurations, and transitions out of a location represent the possible data-flow according to the current configuration and the effect on it.

Notation 1 (Data assignments and data constraints)**.** In the sequel, we assume a finite and non-empty set *Data* consisting of data items that can be transmitted through channels, and a finite set of nodes $\mathcal{N}$. A *data assignment* denotes a function $\delta : N \to Data$ where $\emptyset \neq N \subseteq \mathcal{N}$. We use $\delta.A \in Data$ to denote the data item assigned to every node $A \in N$ under δ and $DA(N)$ for the set of all data assignments of node set N. If $M \subseteq N \subseteq \mathcal{N}$ and $\delta \in DA(N)$ then $\delta|_M$ stands for the data assignment for M that assigns data item $\delta.A$ to each $A \in M$, and $\delta|_\emptyset = \emptyset$. Given $\delta_1 \in DA(N_1)$ and $\delta_2 \in DA(N_2)$, if $\delta_1.A = \delta_2.A$ for all $A \in N_1 \cap N_2$, then $\delta_1 \uplus \delta_2$ stands for the data assignment for $N_1 \cup N_2$ that assigns data item $\delta_1.B_1$ to each $B_1 \in N_1$ and data item $\delta_2.B_2$ to each $B_2 \in N_2$.

Data constraints can be viewed as a symbolic representation of data assignments. Formally, *data constraints* are propositional formulae built from the atoms $d_A = d_B$, $d_A = d$, and $d_A \in D$ (plus the standard boolean connectors $\wedge, \vee, \neg$, etc.) where $A, B \in \mathcal{N}$, d_A and d_B are symbols for the observed data item at node A and B respectively, $d \in Data$ and $D \subseteq Data$. For a node set $N \in \mathcal{N}$, $DC(N)$ denotes the set of data constraints that refer to the terms d_A for $A \in N$. We write DA for $\bigcup_{\emptyset \neq N \subseteq \mathcal{N}} DA(N)$ and DC for $\bigcup_{\emptyset \neq N \subseteq \mathcal{N}} DC(N)$. Given a data constraint $g \in DC(N)$, the semantics for each g is a data assignment $\delta \in DA(N)$, where $\delta \models g$. Here $\models$ stands for the general satisfaction relation which results from interpreting data constraints over data assignments. $g =$ true is equivalent with $\delta = \emptyset$ which stands intuitively for no constraints on the set of nodes N.

Definition 1 (CA)**.** A *constraint automaton* (CA) is a tuple $\mathcal{A} = (L, \mathcal{N}, \longrightarrow, L_0)$ where L is a finite set of locations, $\mathcal{N}$ is a finite set of nodes, disjointly partitioned into $\mathcal{N}^{src}$, $\mathcal{N}^{snk}$, and $\mathcal{N}^{mix}$. The transition relation $\longrightarrow$ is a subset of $L \times 2^{\mathcal{N}} \times DC \times L$, and $L_0 \subseteq L$ the set of initial locations.

We write $l \xrightarrow{N,g} l'$ instead of $(l, N, g, l') \in \longrightarrow$ and refer to N as the node-set and g the guard of the transition, where $N \neq \emptyset$ and $g \in DC(N)$. The semantics for an instance of $l \xrightarrow{N,g} l'$ is a transition of the form $l \xrightarrow{N,\delta} l'$ where $\delta \models g$. If the current location is l then an instance of the outgoing transitions from l is chosen nondeterministically, and the corresponding I/O-operation (i.e., $\langle N, \delta\rangle$) is taken leading to the next location l'. The formalization of such behaviour can be specified by the notion of a *path*. Given a CA $\mathcal{A} = (L, \mathcal{N}, \longrightarrow, L_0)$, a path for $\mathcal{A}$ is a (finite or infinite) sequence of consecutive transition instances $\epsilon = l_0 \xrightarrow{N_0,\delta_0} l_1 \xrightarrow{N_1,\delta_1} l_2 \xrightarrow{N_2,\delta_2} \dots$ where $l_0 \in L_0$. We require that paths are either infinite or end in a location that does not have any outgoing transition, where the node-set N only consists of mixed nodes (i.e., $N \subseteq N^{mix}$). This requirement can be understood as a *maximal progress assumption* for the mixed nodes. For a finite path $\epsilon_{fin} = l_0 \xrightarrow{N_0,\delta_0} l_1 \xrightarrow{N_1,\delta_1} \dots \xrightarrow{N_{i-1},\delta_{i-1}} l_i, i \in \mathbb{N}^+$, we define $\mathsf{Last}(\epsilon_{fin}) = l_i$ as the last reachable location on this path.

Example 2. An FIFO-1 (first-in first-out with one buffer place) channel and its CA are shown in Figure 4. Loca-

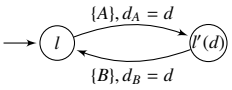


Fig. 4 CA for FIFO-1 channel.

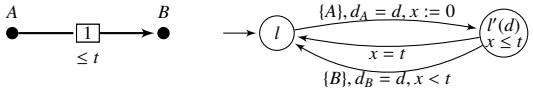


Fig. 5 Timed Reo circuit and TCA for an expiring FIFO-1 channel.

tion l represents the initial configuration with the buffer being empty, while $l'(d)$ stands for the configuration where data item d is stored in the buffer. Note that this is a simplified parametric model where we use parameter d ranging over all data items. A corresponding non-parametric CA has for each $d \in Data$ a location $l'(d)$ and transitions $l \xrightarrow{\{A\},d_A=d} l'(d)$ and $l'(d) \xrightarrow{\{B\},d_B=d} l$.

Reo supports a hide operation, realized on CA by a hiding structure, and a join operation, realized by two constructions. For the join of a source node with another (sink, source or mixed) node, we can use their product, while joining sink or mixed nodes can be specified by a merger CA. The reader interested in more details with respect to basic Reo channels and the corresponding CA is referred to [4, 7].

2.2.1 Timed Constraint Automata

Arbab et al. introduced *timed constraint automata* (TCA) in [9] extended from the original CA by adding real-time aspects to describe the behaviour specification of channels and component interfaces involving timing constraints. As in classical timed automata with location invariants [14, 15], TCA have two kinds of transitions: (i) Internal changes of the locations caused by some time constraints and (ii) transitions that represent the synchronized execution of I/O-operations at some of the ports.

Notation 2 (Clock assignments and clock constraints)**.** Let C be a finite set of clocks. A function $\nu : C \to \mathbb{R}_{\geq 0}$ is referred to be as a *clock assignment* ($\mathbb{CA}$). For a clock assignment ν and a time $t \in \mathbb{R}_{\geq 0}$, $\nu + t$ denotes the clock assignment that assigns the value $\nu(x)+t$ to every clock $x \in C$. If $C \subseteq C$ then $\nu[C := 0]$ stands for the clock assignment that returns the value 0 for every clock $x \in C$ and the value $\nu(x)$ for every clock $x \in C \setminus C$. A *clock constraint* ($\mathbb{CC}$) for C is defined as $cc ::= \mathsf{true} | x \odot n | cc \wedge cc$, where $x \in C, \odot \in \{<, \leq, \geq, >, =\}, n \in \mathbb{N}$, and $\mathbb{CA}(C)$ (or $\mathbb{CA}$) denotes the set of all clock assignments and $\mathbb{CC}(C)$ (or $\mathbb{CC}$) the set of all clock constraints. If $B \subseteq C \subseteq C$ and $cc \in \mathbb{CC}(C)$ then $cc|_B$ stands for the clock constraint for B out of C, i.e., a conjunction of atoms of the form $y \odot n$ where $y \in B$ and $\odot$ is defined as above. With $\mathbf{0}$ we denote the clock assignment with all clocks reset to 0, namely $\mathbf{0}(x) = 0$ for all $x \in C$.

Example 3. Now let us consider expiring FIFO-1 channels in Figure 5, which extend from FIFO-1 with a max time constraint for the data residing in the buffer. After that time, the data is forced to be lost. A clock x is declared in TCA. A time constraint $\leq t$ equipped under the buffer in Reo circuit is used to specify the expiring time aspect, which is denoted by a clock constraint $\mathbb{CC}(x)$ as an invariance condition for location $l'(d)$ in TCA. The two edges from $l'(d)$ to l represent the event where a data item is discarded upon timer expiration and the event where B reads the data out of the buffer respectively.

2.2.2 Probabilistic Constraint Automata

In [10], Baier introduced *probabilistic constraint automata* (PCA) for describing probabilistic connectors in Reo circuits built out of unreliable channels with known failure probabilities, so as to support the modelling of probabilistic lossy synchronous channels or randomized synchronous channels. Baier also defines a probabilistic model, called *simple probabilistic constraint automata* (SPCA), that appear natural to model various kinds of unreliable FIFO channels. SPCA only treat probabilistic choices over configurations but fail to model two important aspects: (i) Channels where synchronous I/O-operations might fail with some probability, such as in probabilistic lossy synchronous channels; (ii) Coin tossing actions where different data items are produced through sink nodes. PCA allow to model these common aspects of IoT systems. In this paper, we consider probabilistic aspects as those representable in PCA.

Merger structures are (again) used for joining sink or mixed nodes in a preprocessing step. The product operation on PCA constitutes the semantics for the join operation (i.e., joining one source node with another one) in Reo circuits. This enables to construct complex connectors out of simpler ones. According to Baier, generating the product of two PCA is much more difficult compared to SPCA that we might need to match (i) one transition with one transition in the other PCA deterministically, or (ii) one transition where none of the sink or mixed nodes of the other PCA is involved with several transitions in the other PCA. This is rooted in the fact that PCA allow distinct I/O-operations on probabilis-

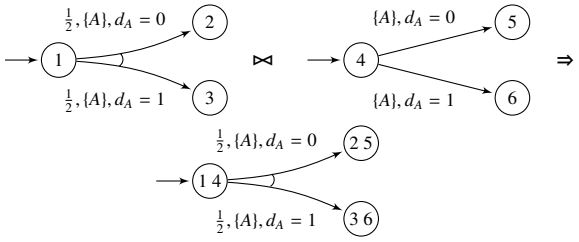


Fig. 6 A case for product for PCA $\mathcal{A}_1$ and $\mathcal{A}_2$.

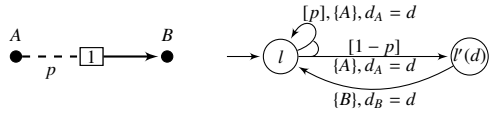


Fig. 7 Probabilistic Reo circuit and PCA for faulty FIFO-1 channel.

tic branches. Let us consider the case in Figure 6: $\mathcal{A}_1$ offers I/O-operations with a probabilistic effect for a sink node A that in turn is a source node in $\mathcal{A}_2$. Then $\mathcal{A}_2$ can react on the outcome of $\mathcal{A}_1$'s probabilistic choice (namely $d_A = 0$ or $d_A = 1$). In this case, the conventional product operation is not handleable, and $\mathcal{A}_1$ is a perfect example for the inability of SPCA.

Notation 3 (Probability distribution)**.** A (discrete) *probability distribution* over a countable set S is a function $\pi : S \to [0, 1]$ satisfying $\sum_{s \in S} \pi(s) = 1$. We call the elements of S *probabilistic events*. We use $Distr(S)$ to denote the set of distributions over S.

We use $\mathcal{D}(s)$ to denote the *deterministic* (often called *Dirac*) distribution that assigns probability 1 for s, defined by $\mathcal{D}(s)(s) = 1$ and $\mathcal{D}(s)(s') = 0$ for all $s' \neq s$. Let $\mathsf{support}(\pi)$ be the subset of S such that $s \in \mathsf{support}(\pi)$ if and only if $\pi(s) > 0$.

Example 4. We consider another variant of FIFO-1, called faulty FIFO-1 in Figure 7, where the data fails to be written into the buffer with some probability p. In PCA, the 'arc' notation is used to connect probabilistic branches of one transition. The I/O-operations on each pair of probabilistic branches out of l (modelled by $\langle \{A\}, d_A = d \rangle$ and $\langle \{A\}, d_A = d \rangle$) stand for the writing failure and success respectively. In this case, SPCA is capable of modelling.

3 Priced Probabilistic Timed Constraint Automata

As mentioned above, real time aspects and probabilistic aspects are two vital aspects to be considered in models of IoT. In these contexts, distributed components (i.e., physical *things*) coordinate with each other using different kinds of communication channels. We intend to employ Reo and CA (and their underlying semantics) as the scripting language to specify components as nodes in Reo and connections as channels or connectors. To this end, they are to be enhanced to support those two features properly. For instance, let us consider a faulty expiring FIFO-1 channel where the data fails to be written into the buffer with some probability and the data is going to be lost after certain time units once it has entered the buffer. On the technical level, we can model the time expiration using TCA, and the possibility to lose data by PCA respectively, but there is still no variant of CA that combines these two aspects effectively. In practice, another vital characteristic of IoT systems is that they need to meet quantitative requirements, because *things* are operating with limited power, memory, computation budgets and other resources. One might be interested in computing some expected values(e.g., the average ink usage costs per refill). A way to handle such quantitative requirements is to use reward/cost notation to specify interesting pricing informations.

This motivates us to propose a model, called *Priced Probabilistic Timed Constraint Automata* (pPTCA), which combines PCA and TCA and is equipped with prices to model nondeterministic, probabilistic, real-time and reward/cost aspects of a system composed by channel-based component (distributed) connectors. Price rates are attached to locations, indicating the cost or reward to reside in a location per time unit. Furthermore, instant prices are attached to transitions, indicating the cost to take the transition or the reward produced, when changing from one location to another. In the following definition, we explicitly use data assignments instead of the symbolic representation data constraints on the transitions in our model to be consistent with the definition of PCA. We use $\mathcal{E} \subseteq (2^N \times DA \times 2^C \times L)$ to denote the set of probabilistic events, and Ct for the set of costs (or rewards).

Definition 2 (pPTCA)**.** A *priced probabilistic timed constraint automaton* (pPTCA) is a tuple $\mathcal{A} = (L, C, \mathcal{N}, \longrightarrow, L_0, ic, \rho)$ where L is a countable set of locations, $L_0 \subseteq L$ the set of initial locations, C a finite set of clocks, and $\mathcal{N}$ is a finite set of nodes disjointly partitioned into $\mathcal{N}^{src}, \mathcal{N}^{snk}$ and $\mathcal{N}^{mix}$.

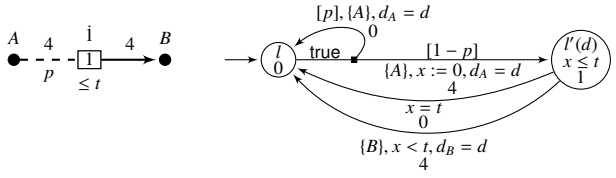


Fig. 8 Priced probabilistic timed Reo circuit and pPTCA for faulty expiring FIFO-1 channel with energy consumption.

$ic : L \to \mathbb{CC}$ is a function that assigns to any location l an invariance condition, and $\longrightarrow \subseteq L \times \mathbb{CC} \times Distr(2^{\mathcal{N}} \times DA \times 2^{C} \times L)$. $\rho : Ct \to (L \cup \mathcal{E} \to \mathbb{R})$ is a price function that, for each *cost* $ct \in Ct$, assigns *price rate* to the locations and *price* to its probabilistic branches. We require that for any transition $l \xrightarrow{cc} \pi$ we have: If $\pi(N_1, \delta_1, C_1, l_1) > 0$ and $\pi(N_2, \delta_2, C_2, l_2) > 0$ then $N_1 \cap \mathcal{N}^{src} = N_2 \cap \mathcal{N}^{src}$ and $\delta_1.A = \delta_2.A$ for all source nodes $A \in N_1 \cap \mathcal{N}^{src}$.

We claim that the probability distribution does not effect the data items source nodes take in, which is formalized in the last statement. A transition fires if the data items are observed in the respective nodes of the component and the clock constraint is satisfied, and the data assignment is performed (except for the empty node-set). We write $l \xrightarrow{cc} \pi(N, \delta, C, l')$ instead of $(l, cc, \pi(N, \delta, C, l')) \in \longrightarrow$, where N denotes the nodes, $\delta \in DA(N)$ the data assignment on N, cc the clock constraint, C the set of clocks to be reset, l and l' represent the source and target locations respectively. Transitions with Dirac distribution (i.e., $\mathcal{D}(N, \delta, C, l') = 1$) are called *Dirac transitions*, and we simply write $l \xrightarrow{cc} \mathcal{D}$.

Intuitively, a pPTCA behaves as follows: It starts in an initial location $l_0 \in L_0$. Then, whenever a location l is occupied with a valid invariance condition ic and each cost ct incurred until now, it is chosen nondeterministically whether to delay or to take a transition which satisfies the above observation constraint and data/clock requirements. Delaying will increase each clock by the delay units and each accumulated cost by the product of delay units and corresponding price rate $\rho(ct)(l)$. A transition instance $l \xrightarrow{cc} \pi(e)$ (where $e = \langle N, \delta, C, l' \rangle \in \mathcal{E}$) is taken, resetting each clock $x \in C$ to 0, and increasing the cost by $\rho(ct)(e)$. The configuration moves to l' with probability $\pi(N, \delta, C, l')$. Note that it is reasonable practice (rooted in the work of Segala [16]) that each transition has a single clock constraint, while the sub-probabilistic branches can be equipped with different data assignments and prices. In the sequel, clock constraint $cc = \mathsf{true}$, data assignment $\delta = \emptyset$ and price or price rate 0 are often left out for simplicity.

Example 5. Figure 8 presents a faulty expiring FIFO-1 channel equipped with prices to reflect the energy consumption on channels and within the buffer, representing that the system needs to consume energy for transmitting data as well for keeping data in the buffer. The semantic pPTCA for this Reo circuit is depicted on the right. This simplified parametric model combines the features from TCA and PCA. Beyond these, in this Reo circuit, the positive constant 4 above the channels represents the instantaneous energy cost for data transmission from node A to the buffer and from the buffer to node B respectively. We use the form $\dot{\$}$ (so as to distinguish it from price) in pictorial Reo circuits to denote the price rate $\$ \in \mathbb{R}$. Thus, $\dot{1}$ above the buffer specifies the energy cost increases by 1 each time unit. Accordingly in the pPTCA, the price rate 0 on location l results from no energy being consumed when the buffer is empty, and 1 on location $l'(d)$ to represent 1 energy unit cost per time unit. The energy consumption for data transmission is encoded into the price 4 on the probabilistic branch from l to $l'(d)$ and on the lowest transition. The clock constraint 'true' is usually omitted in pPTCA for simplicity.

3.1 Target run

Given a pPTCA $\mathcal{A} = (L, C, \mathcal{N}, \longrightarrow, L_0, ic, \rho)$, a *target run* r for $\mathcal{A}$ represents a finite sequence of consecutive transition instances that ends up at a target location without delaying of the following form,

$$\mathsf{r} = l_0 \xrightarrow{cc_0, t_0, \pi_0} l_1 \xrightarrow{cc_1, t_1, \pi_1} l_2 \xrightarrow{cc_2, t_2, \pi_2} \dots \xrightarrow{cc_{n-1}, t_{n-1}, \pi_{n-1}} l_n$$

where $l_i \in L, cc_i \in \mathbb{CC}, (l_i, cc_i, \pi_i) \in \longrightarrow$, and there exist $N_i \subseteq \mathcal{N}$, $\delta_i \in DA(N_i)$, $C_i \in C$, $e_i = \langle N_i, \delta_i, C_i, l_{i+1} \rangle$, such that $\pi_i(e_i) > 0$, and $t_i > 0$ satisfying:

(i) $\nu_i + t_i' \models ic(l_i)$ for all $0 < t_i' \leq t_i$,

(ii) $(\nu_i + t_i)[C_i := 0] \models ic(l_{i+1})$ and

(iii) $\nu_i + t_i \models cc_i$

where $\nu_i \models ic(l_i), 0 \leq i \leq n$. We denote the target location as $\mathsf{Last}(\mathsf{r}) = l_n$. The total cost of r corresponding to $ct \in Ct$ is $\mathsf{TC}_{ct}(\mathsf{r}) = \sum_{i=0}^{n-1} (\rho(ct)(l_i) \cdot t_i + \rho(ct)(e_i))$, i.e., the sum of accumulated prices on locations and instantaneous prices on transitions. The subscript ct is ususally omitted if it is clear from the context. The probability for reaching the target location along r is defined as: $\mathsf{P}(\mathsf{r}) = \prod_i \pi_i(e_i)$.

3.2 Priced probabilistic timed Reo circuits

Electric energy is the key resource needed for *things* to sense, to calculate, to store and to interact under IoT, and thus each

step of those *things* consumes electric power. We detail the pricing model from this very natural IoT perspective now. Nevertheless we mention that we take energy resources as an example, users are nevertheless free to build and work with any pricing structure they wish.

We usually want *things* to accomplish a task or sustain a series of interactions with a limited budget of energy. We furthermore might want to test or verify whether the energy consumption is acceptable (possibly under some fairness conditions) for each *thing* or for the whole IoT system. For this purpose, we extend several useful primitive channels in the Reo framework by adding prices (and price rates) together with timing constraints and probabilistic choices for enhancement of their I/O-operations, resulting in *priced probabilistic timed Reo* (pPTReo for short). We let $b, b_1, b_2 \in \mathbb{R}$ be instantaneous prices on channels (or on transitions in pPTCA), and $\dot{\$}$ (represented by $\$ \in \mathbb{R}$ on the locations in pPTCA) be price rate on buffers or time counting nodes. For clarity, we explicitly present all price rates and prices including 0 valuation.

3.2.1 Basic channels

In Figure 9 we present conservative extensions of part of interesting channels from the original Reo publication. We also present a new communication channel named Zigbee, defined for the purpose of modelling realistic connection means, and a new timer named $t_{\geq}$-timer, introduced to support lower-bound timers (i.e., the time for alerting time-out, represented by TO, is greater than or equal to t). Naturally, a Zigbee channel is equipped with data transmission time t through the channel, failure probability p and energy consumption b. These channels will reappear in the case study section that follows.

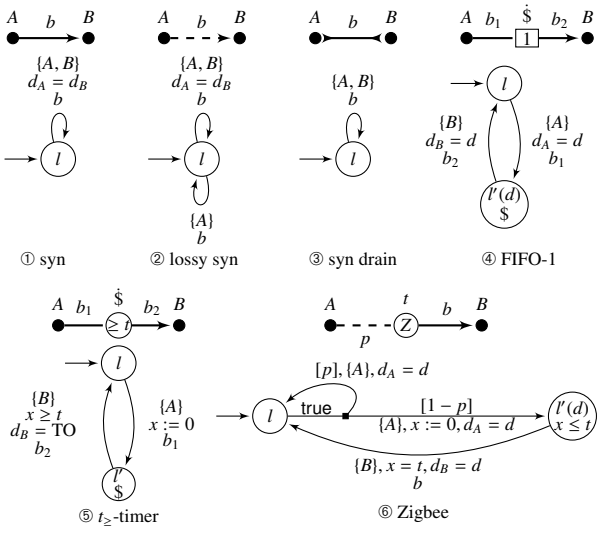


Fig. 9 Six basic Reo channels and their pPTCA.

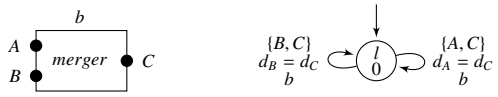


Fig. 10 *Merger* structure.

3.2.2 Merger

For the *merger* structure to be equipped with prices, we decide to equip it with a price rate of 0 on the location and some energy cost $b \in \mathbb{R}$ on the transitions, as depicted in Figure 10. This is necessary because the *merger* has a genuine logic functionality, trying to control the nondeterministic choices between two sink or mixed nodes, as well as implementing the join of two synchronous channels (with ends A, C and B, C respectively).

3.3 Product of pPTCA

After using common mergers to reprocess the automata taking care of the join of non-source nodes, we build the product of two pPTCA to semantically specify the join of one source node with another node in Reo circuits. As mentioned above, one transition is either matched by one or several transitions in another pPTCA. In the latter case, we require all matched transitions have the same clock guard for technical consideration. Assuming independence of energy consumption, for each involved cost, the synchronisation of two pPTCA induces the sum of the price rates on each sub-location to be used as the price rate for the combined location. Once two matching transitions (or one transition matching several transitions) fire at the same time, the price on every probabilistic branch of the combined transition is set to the sum of that on each original branch.

Definition 3 (Product)**.** Given two pPTCA $\mathcal{A}_i = (L_i, C_i, \mathcal{N}_i, \longrightarrow_i, L_{0,i}, ic_i, \rho_i)$, $i = 1, 2$, with $C_1 \cap C_2 = \emptyset$, the product of $\mathcal{A}_1$ and $\mathcal{A}_2$ is $\mathcal{A}_1 \bowtie \mathcal{A}_2 = (L_1 \times L_2, C_1 \cup C_2, \mathcal{N}_1 \cup \mathcal{N}_2, \longrightarrow, L_{0,1} \times L_{0,2}, ic, \rho)$ where $ic(\langle l_1, l_2 \rangle) = ic_1(l_1) \wedge ic_2(l_2)$, $\rho(ct)(\langle l_1, l_2 \rangle) = \rho_1(ct)(l_1) + \rho_2(ct)(l_2)$ for each $ct \in Ct$, $\longrightarrow$ is defined by the following rules:

$$\frac{l_1 \xrightarrow{cc_1}_1 \pi_1, \bigwedge_{j=1}^{j=k}(l_2 \xrightarrow{cc_2}_2 \pi_{2,j})}{\langle l_1, l_2 \rangle \xrightarrow{cc_1 \wedge cc_2} \pi} \quad \text{(initialized by } l_1\text{)}$$

where $k \in \mathbb{N}^+$, and $\bigwedge_{j=1}^{j=k}(l_2 \xrightarrow{cc_2}_2 \pi_{2,j})$ stands for all the transitions in $\mathcal{A}_2$ with same current location l_2 and clock con-

straint cc_2, *if* for each pair of $(M, \sigma) = (N_1 \cap \mathcal{N}_2, \delta_1)$ where $\pi_1(N_1, \delta_1, C_1, l_1') > 0$, either

(i) $M \neq \emptyset$, there exists a probability distribution $\pi_{2,j}$ such that $\pi_{2,j}(N_{2,j}, \delta_{2,j}, C_{2,j}, l_{2,j}') > 0$ implies $N_2 \cap \mathcal{N}_1 = M$ and $\delta_{2,j}.A = \sigma.A$ for all $A \in M$, then

$$\pi(\overbrace{N_1 \cup N_{2,j}, \delta_1 \uplus \delta_{2,j}, C_1 \cup C_{2,j}, \langle l_1', l_{2,j}' \rangle}^{e}) = \pi_1(\underbrace{N_1, \delta_1, C_1, l_1'}_{e_1}) \cdot \pi_{2,j}(\underbrace{N_{2,j}, \delta_{2,j}, C_{2,j}, l_{2,j}'}_{e_2})$$

where $\rho(ct)(e) = \rho_1(ct)(e_1) + \rho_2(ct)(e_2)$, or

(ii) $M = \emptyset$, then

$$\pi(\underbrace{N_1, \delta_1, C_1, \langle l_1', l_2 \rangle}_{e}) = \pi_1(\underbrace{N_1, \delta_1, C_1, l_1'}_{e_1})$$

where $\rho(ct)(e) = \rho_1(ct)(e_1)$.

Transitions out of $\langle l_1, l_2 \rangle$ initialized by l_2 are defined in the symmetric way.

The above definition is a conservative extension of the one for PCA to timed and priced features, with the handling of prices being motivated by [17]. It reformulates the original PCA definition and thereby enables to drop a restriction that originally excludes composition of transitions that are neither input-independent[1)] nor I/O-deterministic[2)]. Support for this case arises naturally from the reformulation.

Remark 1 (Nonassociativity)**.** Similar to PCA, since the product operator $\bowtie$ treats $\mathcal{A}_1$ and $\mathcal{A}_2$ in a symmetric way, $\bowtie$ on pPTCA is commutative up to isomorphism. However, $\bowtie$ is not associative.

Remark 2 (Synchronous and asynchronous product)**.** Let us consider two transitions, for instance as in Figure 11, where contradictory data constraints exist. The product (or so-called parallel composition in ordinary probabilistic automata) on these transitions in generative settings is not straightforward as discussed by Segala [18]. The existing methods take into account the synchronous (where the product transition distribution is generated by the disjoint union on probabilistic branches of the two transitions) [19–21] and asynchronous (where several parallel composition operators were defined in the literature that use bias factors) [21, 22] styles respectively. However in the domain of Reo circuits, such case will

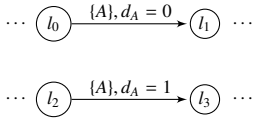


Fig. 11 Differing from the synchronous and asynchronous methods.

be handled by introducing the *merger* structure in the preprocessing stage. As a consequence in the semantic level, the product for such transitions like in Figure 11 induces no transitions in the product automata. This gives a support for not using those synchronous and asynchronous methods.

3.4 Hiding for pPTCA

The hiding structure for pPTCA is extended from that of the PCA by augmenting it with a timed and a price structure. Given one pPTCA $\mathcal{A} = (L, C, \mathcal{N}, \longrightarrow, L_0, ic, \rho)$, a new clock $y \notin C$ and a non-empty node-set $M \subseteq \mathcal{N}^{mix}$. Then, the hide operation $\mathsf{hide}(\mathcal{A}, M)$ on $\mathcal{A}$ results a pPTCA $\exists M[\mathcal{A}] = (L, C \cup \{y\}, \mathcal{N} \setminus M, \longrightarrow_M, L_0, ic, \rho_M)$ where $\longrightarrow_M$ is given by the rules:

$$\frac{l \xrightarrow{cc} \pi, (N = \emptyset \vee N \setminus M \neq \emptyset)}{l \xrightarrow{cc}_M \pi_M} \quad \text{or} \quad \frac{l \xrightarrow{cc} \pi, \emptyset \neq N \subseteq M}{l \xrightarrow{cc \wedge (y>0)}_M \pi_M}$$

where $\pi_M(\overbrace{N \setminus M, \delta|_{N \setminus M}, C \cup \{y\}, l'}^{e_M}) = \pi(\overbrace{N, \delta, C, l'}^{e})$, and for each $ct \in Ct$ $\rho_M(ct)(e_M) = \rho(ct)(e), \rho_M(ct)(l) = \rho(ct)(l)$ for all e_M and $l \in L$. The second rule is used to specify the case that when all the nodes and corresponding data constraints are hidden (as the result of a hiding operation), we need to ensure this transition can fire only after some positive delay. This is achieved by adding an additional clock.

Example 6. Figure 12 illustrates the process of building a connector out of a priced faulty expiring FIFO-1 channel and a priced synchronous channel. On the level of the Reo circuit, these two channels join over B, then hiding it leads to the absence of B and the sum of b_2 and b_3 as the new price associated to the buffer to C. The semantic operations for those behaviours are denoted by the product of the two pPTCA and hiding B and all data constraints involved with d_B on transitions.

Remark 3 (Time-locks problem in Reo circuits)**.** Due to the arbitrary combinations of timed channels, time-locks may happen.

In order to present the interesting logic for properties on the domain of pPTCA models, we first give the semantics of pPTCA by means of Markov decision process (MDP).

[1)] $l_1 \longrightarrow_1 \pi_1$ in $\mathcal{A}_1$ is called *input-independent* on $\mathcal{A}_2$ if $\pi_1(N_1, \delta_1, l_1') > 0$ implies $N_1 \cap \mathcal{N}_1^{src} \cap (\mathcal{N}_2^{snk} \cup \mathcal{N}_2^{mix}) = \emptyset$.

[2)] $l_1 \longrightarrow_1 \pi_1$ in $\mathcal{A}_1$ is called *I/O-deterministic* for $\mathcal{A}_2$ if $\pi_1(N_1, \delta_1, l_1') > 0$ and $\pi_1(N_2, \delta_2, l_2') > 0$ implies $N_1 \cap \mathcal{N}_2 = N_2 \cap \mathcal{N}_2$ and $\delta_1.A = \delta_2.A$ for all $A \in N_1 \cap \mathcal{N}_2$.

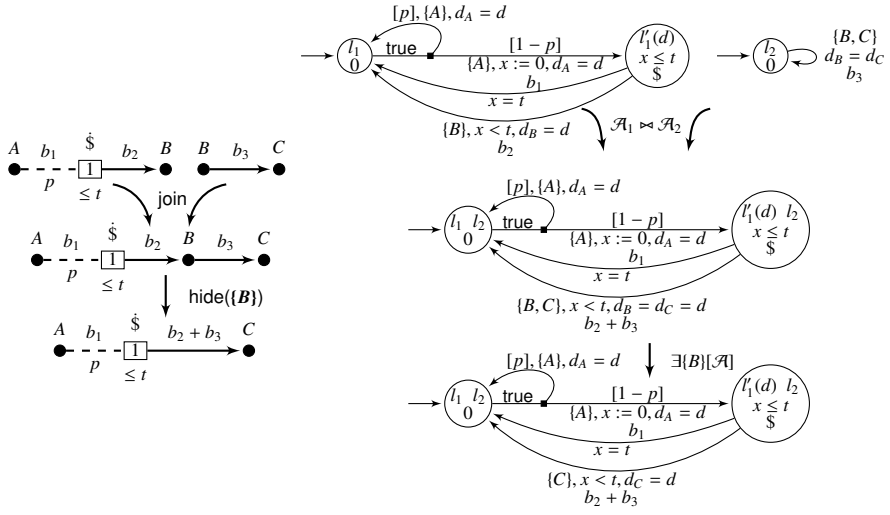


Fig. 12 Example on join and hide, and their semantics.

4 Semantics of pPTCA

4.1 Probabilistic Timed Systems

The semantics for a pPTCA is given by a probabilistic timed system (PTS for short) with reward structures extended from [23]. Specifically a PTS is an infinite MDP where transitions are decorated with time durations and a particular set of labels. The latter is encoded inside the probability distributions to inherit the information of I/O-operations from pPTCA.

Definition 4 (PTS)**.** A *probabilistic timed system* (PTS) is a *Markov Decision Process* $\mathcal{M} = (S, IO, Steps, S_0)$, where S is a set of states, $S_0 \subseteq S$ the set of initial states, and $IO \subseteq 2^{\mathcal{N}} \times DA$ a countable set of *I/O-operations*. $Steps : S \to 2^{\mathbb{R}_{\geq 0} \times Distr(IO \times S)}$ is a *transition function* that assigns to each state $s \in S$ a set of pairs (t, π) where $t \in \mathbb{R}_{\geq 0}$ and $\pi \in Distr(IO \times S)$.

A PTS starts at an initial state $s_0 \in S_0$. When in state $s \in S$, there is a nondeterministic choice between available duration-distribution pairs $(t, \pi) \in Steps(s)$. After the choice, a transition is taken and after t time units a successor state s' together with an I/O-operation $io \in IO$ is selected by probability $\pi(io, s')$. Such *transition steps* are denoted by $s \xrightarrow{t,\pi,io} s'$. In order to make this understood more intuitively, we would like to split the duration and distribution aspects apart. For a transition step $s \xrightarrow{t,\pi,io} s'$, there exists a *time* step $s \xrightarrow{1:t} s''$, and respectively a *discrete* step $s'' \xrightarrow{p:io} s'$ where $p = \pi(io, s')$. The number before the colon stands for the probability taking that step. A state s is called *terminal* iff $Steps(s) = \emptyset$. Given a transition step $s \xrightarrow{t,\pi,io} s'$, $s + t$ denotes s'.

4.1.1 Paths

A path in PTS is gained by resolving nondeterministic and probabilistic choices and defined as the following finite or infinite transition steps:

$$\omega = s_0 \xrightarrow{t_0,\pi_0,io_0} s_1 \xrightarrow{t_1,\pi_1,io_1} s_2 \xrightarrow{t_2,\pi_2,io_2} \dots$$

where $s_i \in S, t_i \in \mathbb{R}_{\geq 0}, io_i \in IO, (t_i, \pi_i) \in Steps(s_i)$, and $(io_i, s_i) \in \mathsf{support}(\pi_i)$ for all $0 \leq i < |\omega|$, where $|\omega|$ denotes the number of transition steps in ω. $|\omega|$ is defined as ∞ for infinite paths, and $\mathsf{Last}(\omega)$ stands for the state where a finite path ω ends. ω is called *maximal* iff $\mathsf{Last}(\omega)$ is terminal. ω is called initial if it starts from one initial state. $\omega(i)$ denotes the $(i+1)$-th state of ω, i.e., $\omega(i) = s_i$. ω^i denotes the prefix sequence of ω till s_i, i.e., $\omega^i = s_0 \xrightarrow{t_0,\pi_0,io_0} s_1 \xrightarrow{t_1,\pi_1,io_1} \dots \xrightarrow{t_{i-1},\pi_{i-1},io_{i-1}} s_i$. We use FiPath to denote the set of finite paths in PTS and InPath the set of infinite paths. $\mathsf{FiPath}(s)$ and $\mathsf{InPath}(s)$ are the sets of finite paths and infinite paths respectively that start from $s \in S$. $\mathsf{FuPath}(s)$ refers to the set of *fulpaths* (starting from s) that are either maximal finite paths or infinite paths.

For a path ω in PTS, $\mathsf{D}_\omega(n)$ denotes the duration up to state s_n. Formally, $\mathsf{D}_\omega(n) = \sum_0^{n-1} t_i$. An infinite path $\omega \in \mathsf{InPath}$ is *time divergent*, if for any $t \in \mathbb{R}_{\geq 0}$, there exists $j \in \mathbb{N}$ such that $\mathsf{D}_\omega(j) > t$. An example of non-divergent path, specifically a zeno path, is: $\omega = s_0 \xrightarrow{2,\pi_0,io_0} s_1 \xrightarrow{1,\pi_1,io_1} s_2 \xrightarrow{0.5,\pi_2,io_2} s_3 \xrightarrow{0.25,\pi_3,io_3} \dots$

Definition 5 (Reward structure)**.** A *reward structure* for a PTS $\mathcal{M} = (S, IO, Steps, S_0)$ is a pair $rew = (rew^{st}, rew^{di})$ where $rew^{st} : S \to \mathbb{R}$ is a *state reward function* and $rew^{di} : IO \times S \to \mathbb{R}$ is a *discrete reward function*.

Given a reward structure $rew = (rew^{st}, rew^{di})$, for a state s, $rew^{st}(s)$ denotes the *rate* at which the reward is accumulated when in s. Whilst given a probability distribution π, for an I/O-operation io and a state s, $rew^{di}(io, s)$ denotes the reward required for choosing pair $(io, s) \in \mathsf{support}(\pi)$. Formally, given an infinite path $\omega = s_0 \xrightarrow{t_0,\pi_0,io_0} s_1 \xrightarrow{t_1,\pi_1,io_1} s_2 \xrightarrow{t_2,\pi_2,io_2} \ldots$, the step reward accumulated from s_i to s_{i+1} is defined as: $rew(\omega, i) = rew^{st}(s_i) \cdot t_i + rew^{di}(io_i, s_{i+1})$.

4.1.2 Schedulers

A (history-dependent) scheduler resolves the nondeterministic choices by choosing a duration-distribution pair depending on the current state and on the current prefix of the path up to that state.

Definition 6 (Scheduler of a PTS)**.** Given a PTS $\mathcal{M} = (S, IO, Steps, S_0)$, a *scheduler* of $\mathcal{M}$ is a function $\zeta : \mathsf{FiPath} \to \mathbb{R}_{\geq 0} \times Distr(IO, S)$ such that $\zeta(\omega) \in Steps(\mathsf{Last}(\omega))$ for all $\omega \in \mathsf{FiPath}$.

Intuitively, if a PTS has progressed so far arriving at s along a finite path ω under a scheduler ζ, then it will take time duration $t \in \mathbb{R}_{\geq 0}$ and choose an I/O-operation $io \in IO$ leading to state s' in the next step with probability $\pi(io, s')$, where $(t, \pi) = \zeta(\omega) \in Steps(s)$. Scheduler ζ on PTS $\mathcal{M}$ induces a discrete time Markov chain (DTMC) $\mathcal{M}^\zeta$. Each state in $\mathcal{M}^\zeta$ is a finite path ω in $\mathcal{M}$. The transition probability distribution is determined by ζ. We denote the set of all schedulers on $\mathcal{M}$ as $\mathsf{SCH}_{\mathcal{M}}$.

Given a scheduler ζ, we define $\mathsf{FiPath}^\zeta(s)$ (respectively $\mathsf{InPath}^\zeta(s)$ and $\mathsf{FuPath}^\zeta(s)$) as the set of finite paths (respectively infinite paths and fulpaths) $\omega = s \xrightarrow{t_0,\pi_0,io_0} s_1 \xrightarrow{t_1,\pi_1,io_1} \ldots$ (starting from s) such that for any $i \in \mathbb{N}$, $\pi(io_{i-1}, s_i) > 0$ where $\pi_i = \zeta(\omega^{i-1})$. Then we define the probability for a finite path induced by a scheduler ζ. For $\omega = s_0 \xrightarrow{t_0,\pi_0,io_0} s_1 \xrightarrow{t_1,\pi_1,io_1} \ldots \xrightarrow{t_{k-1},\pi_{k-1},io_{k-1}} s_k \in \mathsf{FiPath}^\zeta$, if $k = 0$, $\mathsf{P}^\zeta(\omega) = 1$, else if $k \geq 1$, $\mathsf{P}^\zeta(\omega) = \mathsf{P}^\zeta(\omega^{k-1}) \cdot \zeta(\omega^{k-1})(io_{k-1}, s_k)$.

Built on the basic notions of probability theory [24], for each state s, we show that P^ζ induces a probability space on $\mathsf{FuPath}^\zeta(s)$ as follows. We define $\sigma Field^\zeta_s$ as the smallest σ-field on $\mathsf{FuPath}^\zeta(s)$ containing the basic *cylinders* $\omega \uparrow$[3], where $\omega \in \mathsf{FiPath}^\zeta(s)$, i.e., ω ranges over all finite paths starting in s. The probability measure $Prob^\zeta_s$ is the unique measure on $\sigma Field^\zeta(s)$ such that $Prob^\zeta_s(\omega \uparrow) = \mathsf{P}^\zeta(\omega)$.

[3] $\sigma \uparrow = \{\omega \in \mathsf{FuPath}^\zeta(s) : \sigma \leq_{pre} \omega\}$ where $\leq_{pre}$ is the usual prefix relation on paths.

4.2 Semantics of pPTCA

The semantics of a pPTCA is a PTS where the states keep track of the current location and the current values of all clock variables, the labels on probabilistic branches keep track of I/O-operations in pPTCA, and the attached reward structures record prices. Given a pPTCA $\mathcal{A} = (L, C, \mathcal{N}, \longrightarrow, L_0, ic, \rho)$, a *state* is a pair of $\langle l, \nu \rangle$, where $l \in L$ is a location of $\mathcal{A}$, $\nu \in \mathbb{CA}$ is a clock assignment over C and $\nu \models ic(l)$. The initial state of $\mathcal{A}$ is represented by $\langle l_0, \mathbf{0} \rangle$ for all $l_0 \in L_0$.

Definition 7 (Semantics of pPTCA)**.** The semantics of a pPTCA $\mathcal{A} = (L, C, \mathcal{N}, \longrightarrow, L_0, ic, \rho)$ is a PTS $\mathcal{M}^{\mathcal{A}} = (S, IO, Steps, S_0)$ with:

- $S = \{\langle l, \nu \rangle \mid \nu \models ic(l), \nu \in \mathbb{CA}(C), l \in L\}$,
- $S_0 = \{\langle l_0, \mathbf{0} \rangle \mid l_0 \in L_0\}$,
- $IO = \{\langle N, \delta \rangle \mid N \subseteq \mathcal{N}, \delta \in DA(N)\}$,
- Given $(t, \pi) \in Steps(\langle l, \nu \rangle)$, the transition $\longrightarrow$ is defined by the following rules:
  - *discrete transition:* $\langle l, \nu \rangle \xrightarrow{0,\pi,\langle N,\delta \rangle} \langle l', \nu' \rangle$ if there exists $l \xrightarrow{cc} \pi'$ in $\mathcal{A}$ such that:
    (i) $\pi'(N, \delta, C, l') > 0$,
    (ii) $\nu \models cc$,
    (iii) $\nu' = \nu[C := 0]$ and $\nu' \models ic(l')$, and
    (iv)
    $$\pi(\langle N, \delta \rangle, \langle l', \nu' \rangle) = \sum_{C \subseteq C, \nu' = \nu[C:=0]} \pi'(N, \delta, C, l')$$
    We simply write this transition as $\langle l, \nu \rangle \longrightarrow \pi$.
  - *time transition:* $\langle l, \nu \rangle \xrightarrow{t'} \mathcal{D}(\emptyset, \langle l, \nu + t' \rangle)$ for all $0 \leq t' \leq t$, if $\nu + t' \models ic(l)$.
    We simply write this transition as $\langle l, \nu \rangle \xrightarrow{t'} \langle l, \nu + t' \rangle$.

For each *reward structure* $rew_{ct} = (rew^{st}_{ct}, rew^{di}_{ct})$ corresponding to a *cost* $ct \in Ct$, and any $\langle l, \nu \rangle \in S$, we define:

- $rew^{st}_{ct}(\langle l, \nu \rangle) = \rho(ct)(l)$ and
- $rew^{di}_{ct}(\langle N, \delta \rangle, \langle l, \nu \rangle) = \rho(ct)(\langle N, \delta, C, l \rangle)$ for all $(N, \delta, C, l) \in \mathcal{E}$ where $\nu[C := 0] \models ic(l)$.

Followed by the notion of reward structures, we can define the transition reward as:

- for a discrete transition:
$$rew_{ct}(\langle l, \nu \rangle \longrightarrow \pi) = \sum_{e = (\langle N,\delta \rangle, \langle l', \nu' \rangle) \in \mathsf{support}(\pi)} \pi(e) \cdot rew^{di}_{ct}(e),$$

and

- for a time transition:

$$rew_{ct}(\langle l, \nu\rangle \xrightarrow{t} \langle l, \nu + t\rangle) = t \cdot rew^{st}_{ct}(\langle l, \nu\rangle).$$

Given a pPTCA $\mathcal{A}$ and $s = \langle l, \nu\rangle$ a state in $\mathcal{M}^{\mathcal{A}}$, an s-path (or simply path) in $\mathcal{A}$ denotes any path in $\mathcal{M}^{\mathcal{A}}$ from s.

Remark 4. In this paper, we restrict our attention to time divergent behaviour, a common restriction imposed in real-time systems. So unrealised behaviour, on which the corresponding time does not advance beyond a time bound, is disregarded. The eligible behaviour is also called non-zeno. In another word, all infinite paths in $\mathcal{M}^{\mathcal{A}}$ are time divergent.

5 Priced Probabilistic Timed Scheduled Data Stream Logic

Since our IoT tailored extension of Reo and its underlying pPTCA models are driven by data and pricing information, it is worthwhile to consider formally the observable data-flow together with particular pricing informations contained in paths of a pPTCA. For this, we introduce a priced extension of timed scheduled data streams [9]. Each such stream keeps track of cost $ct \in Ct$ accumulating along a sequence of triples $(\mathring{t}, io, b)$ where $\mathring{t}$ stands for a time point, $io = \langle N, \delta\rangle$ denotes I/O-operations where $N \in \mathcal{N}$ is a non-empty node-set and $\delta \in DA(N)$ a data assignment, and $b \in \mathbb{R}$ being the cost value accumulated up to $\mathring{t}$. Intuitively, $(\mathring{t}, io, b)$ means that at time $\mathring{t}$ the I/O-operation io specified by $\langle N, \delta\rangle$ is performed, incurring an accumulated cost value of b.

Notation 4 (pTsD stream)**.** Given a node set $\mathcal{N}$, a *priced timed scheduled data stream* (pTsD) is defined as a finite or infinite sequence $\Theta = (\mathring{t}_0, io_0, b_0), (\mathring{t}_1, io_1, b_1), \ldots \in (\mathbb{R}_{>0} \times IO \times \mathbb{R})^{\infty}$ such that $0 < \mathring{t}_0 \leq \mathring{t}_1 \leq \ldots$, for all $io_i = \langle N_i, \delta_i\rangle, \emptyset \neq N_i \in \mathcal{N}, \delta_i \in DA(N_i)$, and Θ is time divergent. The empty pTsD stream is denoted by ε. The length $|\Theta|$ stands for the number of tuples $(\mathring{t}, io, b)$ in Θ. $\mathsf{t}(\Theta)$ denotes the execution time of Θ. $\mathsf{t}(\Theta)$ is defined as ∞ if Θ is infinite, $\mathring{t}_k$ if $|\Theta| = k+1$, and 0 if $\Theta = \epsilon$. $\mathsf{b}(\Theta)$ stands for the accumulated cost value of Θ after the execution time $\mathsf{t}(\Theta)$, and defined as ∞ if Θ is infinite, b_k if $|\Theta| = k+1$, and 0 if $\Theta = \epsilon$. **pTsD** denotes the set of all pTsD streams.

Definition 8 (pTsD-language of a pPTCA)**.** Given a path $\omega = s_0 \xrightarrow{t_0,\pi_0,io_0} s_1 \xrightarrow{t_1,\pi_1,io_1} \ldots$ from the semantic PTS $\mathcal{M}^{\mathcal{A}}$ of a pPTCA $\mathcal{A}$. We define $\Theta(\omega)$ induced from ω by (i) retrieving t_i and io_i from the step labels together with the step rewards $b_i = rew(\omega, i)$, (ii) replacing t_i with the accumulated duration $\mathring{t}_i = t_0 + t_1 + \ldots + t_i$, b_i with the total accumulated cost value $b'_i = b_0 + b_1 + \ldots + b_i$, and (iii) removing all $(\mathring{t}_i, io_i, b'_i)$ with $io_i = \emptyset$. The generated *pTsD-language* of a state s in $\mathcal{M}^{\mathcal{A}}$ is $\mathcal{L}(\mathcal{A}, s) = \{\Theta(\omega) \in \mathbf{pTsD} \mid \omega \in \mathsf{FuPath}(s)\}$. Language $\mathcal{L}(\mathcal{A})$ represents all pTsD streams $\Theta(\omega)$ where ω is a maximal and initial path.

In order to verify pPTCA models against priced probabilistic timed properties, one can use temporal logic to express these quantitative properties. The basis for this is PCTL*, a probabilistic variant of CTL [25]. Here we present Priced Probabilistic Timed scheduled Data stream Logic (pPTDL for short) that is a real-time variant of PCTL* together with operators as in [26] to specify prices (or costs or rewards). pPTDL allows us to reason about the observable pricing data flow of a Reo circuit by means of the pTsD streams generated by its underlying pPTCA. The probabilistic behaviour can also be quantified supported through probabilistic operators. This puts us in the position to describe interesting properties for IoT, such as 'reliability', 'dependability', and 'performability' [27].

5.1 Syntax of pPTDL

In this paper, we take the way in [9] to describe the modality $\bigcirc$ (next step), which is one standard operator in traditional probabilistic CTL [28], as an operator $\langle\!\langle\gamma\rangle\!\rangle\psi$ consisting of a pTsD expression γ and a formula ψ. pTsD expressions extend timed regular expressions [29] so as to specify sets of finite pTsD streams. Intuitively, $\langle\!\langle\gamma\rangle\!\rangle\psi$ holds for a pPTCA $\mathcal{A}$ iff *each* pTsD stream $\Theta \in \mathcal{L}(\mathcal{A})$ *has a (finite) prefix* that generates a γ-word (i.e. a word expressed by γ) and ψ holds for its remaining suffix. The $\langle\!\langle\cdot\rangle\!\rangle$ operator can be considered as a 'universal-hold' case for the non-deterministic structures. On the other hand, we can use the dual operator $[\![\cdot]\!]$ for specifying a 'selective-hold' case. $[\![\gamma]\!]\psi = \neg(\langle\!\langle\gamma\rangle\!\rangle\neg\psi)$ is satisfied for $\mathcal{A}$ iff *whenever* a pTsD stream $\Theta \in \mathcal{L}(\mathcal{A})$ *possesses a (finite) prefix* generating a γ-word, ψ holds for its corresponding suffix.

The pTsD expression is defined as:

$$\gamma ::= \langle N, \delta\rangle \mid \gamma_1 \vee \gamma_2 \mid \gamma_1 \wedge \gamma_2 \mid \gamma_1;\gamma_2 \mid \gamma^* \mid \gamma^I \mid \gamma^b$$

where N is a non-empty set of nodes and $\delta \in DA(N)$, b is a cost value, and I is a time interval of the form $[0, a]$ or $[0, a)$ where $a \in \mathbb{R}_{\geq 0} \cup \{\infty\}$. Usually we simply write $< a$, $\leq a$ or ∞ instead of I. As introduced in regular expressions, $\gamma_1 \vee \gamma_2$ refers to *union*, $\gamma_1 \wedge \gamma_2$ *intersection*, $\gamma_1;\gamma_2$ *concatenation*, and γ^* *Kleene closure*. γ^I and γ^b specify the same properties as γ

except for a constraint that the total execution time falls in the time interval I and that the accumulated cost value is within the bound of b respectively.

The syntax of pPTDL is given by:

$$\phi ::= \mathsf{true} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathbf{P}_{\sim p}[\psi] \mid \mathbf{R}^{rew}_{\sim b}[\iota]$$
$$\psi ::= \mathsf{true} \mid \langle\!\langle \gamma \rangle\!\rangle \psi \mid \psi_1 \mathsf{U} \psi_2$$
$$\iota ::= \mathsf{A}^k \mid \mathsf{F}\phi \mid \mathsf{L}\psi$$

where $p \in [0,1], b \in \mathbb{R}, \sim \in \{=, <, >, \leq, \geq\}$, and *rew* is a reward structure corresponding to some cost $ct \in Ct$. ϕ is called *state* formula, which refers to the satisfied properties on states of the semantic model. **P** and **R** stand for *probabilistic* and *reward* operators respectively. $\mathbf{P}_{\sim p}[\psi]$ specifies that the probability of *stream* formula ψ being true is within the bound $\sim p$. $\langle\!\langle \gamma \rangle\!\rangle \psi$ indicates a time point such that the induced prefix and suffix of the stream satisfy γ and ψ respectively. $\psi_1 \mathsf{U} \psi_2$ means that a suffix of the stream satisfying ψ_2 exists, and ψ_1 is met for all suffix starting from each time point prior to that. Notably, the constituents of stream formulae in our definition are also stream formulae, which differ from those of the *path* formulae in PCTL* where the constituents are allows to be state or path formulae. This is due to the pTsD-stream-based semantics which is detailed in the next sub-section. The embraced LTL-flavour in pPTDL also forms the main reason to not remould another CTL variant PCTL [30, 31]. $\mathbf{R}^{rew}_{\sim b}[\iota]$ specifies the expected pricing value under condition ι with the given reward structure *rew* (on its semantic PTS model) is within the bound $\sim b$. A^k models the accumulated reward until time k, $\mathsf{F}\phi$ the accumulated reward until reaching a state satisfying ϕ, and $\mathsf{L}\psi$ the lowest cost for streams satisfying ψ. Intuitively, those conditions for expected pricing value refer to three aspects respectively: (i) cost under a period of time constraint, (ii) cost to reach some state inducing the target set of pTsD streams, and (iii) cost on some target set of pTsD streams. Note that pTDL does not contain a time-bounded operator on stream formula, which in practice can be modelled with the help of γ^I.

In order to specify the properties in a more friendly and easy-understanding way, we derive the common operators on stream formulae as follows. *Next*: $\bigcirc\psi = \langle\!\langle \bigvee_i \langle N_i, \mathsf{true}\rangle \rangle\!\rangle \psi$ where N_i ranges over all non-empty subsets of $\mathcal{N}$. *Eventually*: $\Diamond\psi = \mathsf{true}\mathsf{U}\psi$. *Always*: $\Box\psi = \neg\Diamond\neg\psi$.

Example 7. The CA for FIFO-1 channel in Figure 4 satisfies the property

$$\phi_1 = \bigwedge_{d \in Data} \mathbf{P}_{=1}(\Box(\llbracket \langle \{A\}, d_A = d\rangle \rrbracket \langle\!\langle \langle \{B\}, d_B = d\rangle \rangle\!\rangle \mathsf{true}))$$

that states 'for all incoming data d from node A, after d is written into the buffer, node B always read the same d'.

For simplicity, we would like to omit the brackets for the node sets and the true statements. The $t_{\geq}$-timer introduced in Figure 9.⑤ satisfies the following property

$$\phi_2 = \mathbf{P}_{=0}(\Box(\llbracket \langle A \rangle^* \rrbracket \langle\!\langle \langle B, d_B = \mathrm{TO} \rangle^{<t} \rangle\!\rangle))$$

which describes 'the probability that after node A inputs a (arbitrary) data for several times, node B outputs the message "Time Out" within (excluding) t time units is zero'.

In Figure 12, the compositional channel built from the FIFO-1 and a synchronous channel satisfies the property

$$\phi_3 = \mathbf{R}^{rew_{\mathbf{energy}}}_{\geq (b_1+b_2+b_3)}[\mathsf{A}^t]$$

that shows 'under the reward structure $rew_{\mathbf{energy}}$ corresponding to the cost **energy**, the accumulated energy cost until t time units is at least $(b_1 + b_2 + b_3)$'.

5.2 Semantics of pPTDL

Since we focus on the priced observable data-flow feature, the semantics of pPTDL model is studied from the perspective of satisfaction on streams. Before giving the formal definition of the semantics, we first introduce the basic theorem of *time cuts* and *stream concatenation*.

5.2.1 Time cuts

Given a pTsD stream $\Theta = (\mathring{t}_0, io_0, b_0), (\mathring{t}_1, io_1, b_1), \ldots$ and a time point $t \in \mathbb{R}_{>0}$, we use $\Theta{\uparrow}t$ and $\Theta{\downarrow}t$ to denote the pTsD stream that specifies the data-flow in the time interval $[t, \infty)$ and $[0, t)$ respectively. Formally,

- $\Theta{\uparrow}t = \epsilon$ if $|\Theta| = k + 1 < \infty$ and $\mathring{t}_k < t$, and
- $\Theta{\uparrow}t = (\mathring{t}_k, io_k, b_k), (\mathring{t}_{k+1}, io_{k+1}, b_{k+1}), \ldots$ if $|\Theta| = \infty$ and k is the smallest index such that $\mathring{t}_k \geq t$.
- $\Theta{\downarrow}t = \epsilon$ if $|\Theta| = \epsilon$ or $\mathring{t}_0 \geq t$, and
- $\Theta{\downarrow}t = (\mathring{t}_0, io_0, b_0), \ldots, (\mathring{t}_k, io_k, b_k)$ if $|\Theta| \neq \infty$ and k is the largest index such that $\mathring{t}_k < t$.

5.2.2 Concatenation

Consider pTsD streams $\Theta_1 = (\mathring{t}^1_0, io^1_0, b^1_0), \ldots, (\mathring{t}^1_n, io^1_n, b^1_n)$, and $\Theta_2 = (\mathring{t}^2_0, io^2_0, b^2_0), \ldots, (\mathring{t}^2_m, io^2_m, b^2_m)$, we define the concatenation of Θ_1 and Θ_2 as

$$\begin{aligned}\Theta_1;\Theta_2 =& (\mathring{t}^1_0, io^1_0, b^1_0), \ldots, (\mathring{t}^1_n, io^1_n, b^1_n),\\ & (\mathring{t}^2_0 + \mathring{t}^1_n, io^2_0, b^2_0 + b^1_n), \ldots, (\mathring{t}^2_m + \mathring{t}^1_n, io^2_m, b^2_m + b^1_n).\end{aligned}$$

Now we give the formal definition of the semantics of pTsD expressions and pPTDL formulae.

5.2.3 Semantics of pTsD expression

We define the concatenation of finite pTsD streams as follows. $\Theta;\epsilon = \epsilon;\Theta = \Theta$. For $\mathcal{L}_1$ and $\mathcal{L}_2$ that are pTsD-languages with the same node-set $\mathcal{N}$ and the same involved cost $ct \in Ct$, we define $\mathcal{L}_1;\mathcal{L}_2 = \{\Theta_1;\Theta_2 \mid \Theta_1 \in \mathcal{L}_1, \Theta_2 \in \mathcal{L}_2\}$. $\mathcal{L}^* = \cup_{n\geq 0}\mathcal{L}^n$ where $\mathcal{L}^0 = \{\epsilon\}$, $\mathcal{L}^{n+1} = \mathcal{L}^n;\mathcal{L}$.

The semantics of pTsD expression γ is defined by structural induction as $\mathcal{L}(\gamma) \subseteq \mathbf{pTsD}$. $\mathcal{L}(\langle N, \delta\rangle)$ is the set of all pTsD streams of length one with the form of $(\mathring{t}, \langle N, \delta\rangle, b)$. $\mathcal{L}(\gamma_1 \vee \gamma_2) = \mathcal{L}(\gamma_1) \cup \mathcal{L}(\gamma_2)$, $\mathcal{L}(\gamma_1 \wedge \gamma_2) = \mathcal{L}(\gamma_1) \cap \mathcal{L}(\gamma_2)$, $\mathcal{L}(\gamma_1;\gamma_2) = \mathcal{L}(\gamma_1);\mathcal{L}(\gamma_2)$, and $\mathcal{L}(\gamma^*) = \mathcal{L}(\gamma)^*$. $\mathcal{L}(\gamma^I) = \{\Theta \in \mathcal{L}(\gamma) \mid \mathsf{t}(\Theta) \in I\}$. $\mathcal{L}(\gamma^b) = \{\Theta \in \mathcal{L}(\gamma) \mid \mathsf{b}(\Theta) \leq b\}$.

5.2.4 Semantics of pPTDL

Given a pPTCA $\mathcal{A}$, the semantics of pPTDL (state) formulae is given in terms of the satisfaction relation $\models$ on the states of semantics model $\mathcal{M}^{\mathcal{A}}$, which is presented by the structural induction as follows.

$$
\begin{array}{lll}
s \models \mathsf{true} & & \\
s \models \neg\phi & \text{iff} & s \not\models \phi \\
s \models \phi_1 \wedge \phi_2 & \text{iff} & s \models \phi_1 \text{ and } s \models \phi_2 \\
s \models \mathbf{P}_{\sim p}[\psi] & \text{iff} & Prob_s^{\zeta}(\{\omega \in \mathsf{FuPath}^{\zeta}(s) \mid \Theta(\omega) \models \psi\}) \sim p \\
 & & \text{for all } \zeta \in \mathsf{SCH}_{\mathcal{M}^{\mathcal{A}}} \\
s \models \mathbf{R}^{rew}_{\sim b}[\iota] & \text{iff} & Expv_s^{\zeta}(\mathsf{rv}(rew, \iota)) \sim b \\
 & & \text{for all } \zeta \in \mathsf{SCH}_{\mathcal{M}^{\mathcal{A}}}
\end{array}
$$

where the semantics of stream formula ψ is defined with regard to the data-flow streams of pPTCA $\mathcal{A}$ as follows:

$$
\begin{array}{lll}
\Theta \models \mathsf{true} & & \\
\Theta \models \neg\psi & \text{iff} & \Theta \not\models \psi \\
\Theta \models \psi_1 \wedge \psi_2 & \text{iff} & \Theta \models \psi_1 \text{ and } \Theta \models \psi_2 \\
\Theta \models \langle\!\langle\gamma\rangle\!\rangle\psi & \text{iff} & \exists t \in \mathbb{R}_{\geq 0} \text{ such that } \Theta{\downarrow}t \in \mathcal{L}(\gamma) \wedge \Theta{\uparrow}t \models \psi \\
\Theta \models \psi_1 \mathsf{U} \psi_2 & \text{iff} & \exists t \in \mathbb{R}_{\geq 0} \text{ such that } \Theta{\uparrow}t \models \psi_2 \\
 & & \text{and } \Theta{\uparrow}t' \models \psi_1 \text{ for all } t' \text{ with } 0 \leq t' \leq t
\end{array}
$$

and $Expv_s^{\zeta}(\mathsf{rv})$ refers to the expected value for random variable rv based on the paths ω starting from s and induced by ζ, which for reward structure $rew = (rew^{st}, rew^{di})$ over $\mathcal{M}^{\mathcal{A}}$ is defined as follows:

$$
\begin{aligned}
\mathsf{rv}(rew, \mathsf{A}^k) &= \sum_{i=0}^{j_k-1} rew(\omega, i) + (k - \mathsf{D}_{\omega}(j_k)) \cdot rew^{st}(\omega(j_k)) \\
\mathsf{rv}(rew, \mathsf{F}\phi) &= \begin{cases} \sum_{i=0}^{j_\phi-1} rew(\omega, i) + t_\phi \cdot rew^{st}(\omega(j_\phi)) & \text{if } (j_\phi, t_\phi) \text{ exists} \\ \infty & \text{otherwise} \end{cases} \\
\mathsf{rv}(rew, \mathsf{L}\psi) &= \min\{\mathsf{b}(\Theta(\omega)) \mid \Theta(\omega) \models \psi\}
\end{aligned}
$$

where $j_k = \max\{i \mid \mathsf{D}_{\omega}(i) < k\}$[4], j_ϕ is the minimum position such that $\omega(j_\phi) + t_\phi \models \phi$.

Is it easy to find the semantics for operator $[\![\gamma]\!]\psi$ as follows:

$$\Theta \models [\![\gamma]\!]\psi \quad \text{iff} \quad \text{for all } t \geq 0,\ \Theta{\downarrow}t \in \mathcal{L}(\gamma) \text{ implies } \Theta{\uparrow}t \models \psi.$$

We say $\mathcal{A} \models \phi$ iff for all initial states $s \in S_0$ of $\mathcal{M}^{\mathcal{A}}$ such that $s \models \phi$. We define the induced pTsD-language for a pPTDL-formula ϕ as:

$$\mathcal{L}(\phi) = \{\Theta(\omega) \in \mathbf{pTsD} \mid \omega \in \mathsf{FuPath}(s) \wedge s \models \phi\}.$$

Then the equivalence $\equiv$ of pPTDL-formulae is defined as $\phi_1 \equiv \phi_2$ iff $\mathcal{L}(\phi_1) = \mathcal{L}(\phi_2)$.

5.3 Verification of pPTCA against pPTDL

One can build the pPTReo circuit for the modelling of IoT system, and the underlying semantic behaviour is represented by the corresponding pPTCA model. The model reflects how the real system behaves, and naturally it is interesting to formally understand what properties the model satisfies. In another word, we apply the pPTDL formulae to verify the pPTCA models. The criteria for such process is presented as the relationships between the state/model-property satisfaction and the pTsD-language inclusion. Intuitively, we need to clarify when a state in a pPTCA model satisfies a formula expressed by pPTDL, whether the corresponding pTsD-language generated by the state itself is included in that induced by the formula. Formally, we have the following propositions.

Proposition 1. Given a pPTCA $\mathcal{A}$, its semantic model $\mathcal{M}^{\mathcal{A}}$ and a pPTDL-formula ϕ, $s \models \phi$ iff $\mathcal{L}(\mathcal{A}, s) \subseteq \mathcal{L}(\phi)$.

Proof. We consider two directions respectively.

(i) $s \models \phi$ implies $\mathcal{L}(\mathcal{A}, s) \subseteq \mathcal{L}(\phi)$. According to Definition 8, $\mathcal{L}(\mathcal{A}, s) \overset{def}{=} \{\Theta(\omega) \in \mathbf{pTsD} \mid \omega \in \mathsf{FuPath}(s)\}$,

[4] Using $< k$ instead of $\leq k$ guaranteens that the last discrete reward is excluded when some transition step is about to happen at time k.

and according to the above, $\mathcal{L}(\phi) \stackrel{def}{=} \{\Theta(\omega) \in \mathbf{pTsD} \mid \omega \in \mathsf{FuPath}(s) \wedge s \models \phi\}$. Since $s \models \phi$, it is straightforward that $\mathcal{L}(\mathcal{A}, s) = \mathcal{L}(\phi)$.

(ii) $\mathcal{L}(\mathcal{A}, s) \subseteq \mathcal{L}(\phi)$ implies $s \models \phi$. For each $\Theta(\omega)$ where $\omega \in \mathsf{FuPath}(s)$, it is also an element of $\mathcal{L}(\phi)$. According to the definition of $\mathcal{L}(\phi)$, we have $s \models \phi$. □

Proposition 2. Given a pPTCA $\mathcal{A}$, its semantic model $\mathcal{M}^{\mathcal{A}}$ and a pPTDL-formula ϕ, $\mathcal{A} \models \phi$ iff $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\phi)$.

Proof. The proof can be done similar to Proposition 1. □

Remark 5. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two pPTCA with the same node-set $\mathcal{N}$. Clearly if $\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$ then, for any pPTDL-formula ϕ, $\mathcal{A}_2 \models \phi$ implies $\mathcal{A}_1 \models \phi$. Thus if $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$ then $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy the same pPTDL-formulae.

6 Case Study

In this section, we consider an Internet-of-Things scenario where a remote and isolated factory unit answers to customers' requests. We assume in this IoT system that one *sink device* **SD** is in charge of receiving orders and distributing the tasks to *robots*. The robots are deployed in hostile environments and their main duty is to sense, process and transmit data. They need to be able to operate as long as possible. Due to cost economics, they are small in size and provide limited power (supported by batteries). At the beginning, only one robot named **R1** is deployed, and can handle one task at a time. After completing the task, **R1** sends data to one nearby *workstation* **W** through Zigbee channels for further processing. In this case, we consider one energy cost function ρ, and therefore write ρ instead of $\rho(ct)$ where ct stands for the only energy cost.

6.1 Reo circuit for the system

We can easily construct the model of connecting every *thing* in this IoT system as layed out in Figure 13. We use one connector composed of one FIFO-1, one $t_{\geq}$-timer and one synchronous drain channel to model task processing in **R1**, where the data from the buffer can only be fetched after the execution time (i.e. ≥ 5). The drain channel restricts time-out and buffer-clear to coincide. Price rates equipping FIFO-1 and $t_{\geq}$-timer represent the energy consumption rate for data storing and processing respectively. The energy consumption for data transmission is encoded in Zigbee channels. Due to the unreliable wireless network, these communication channels have a loss probabilities associated. **SD** writes

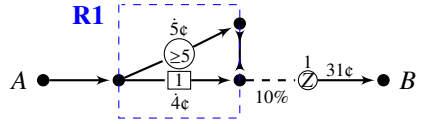


Fig. 13 Reo circuit for the exemplary IoT system.

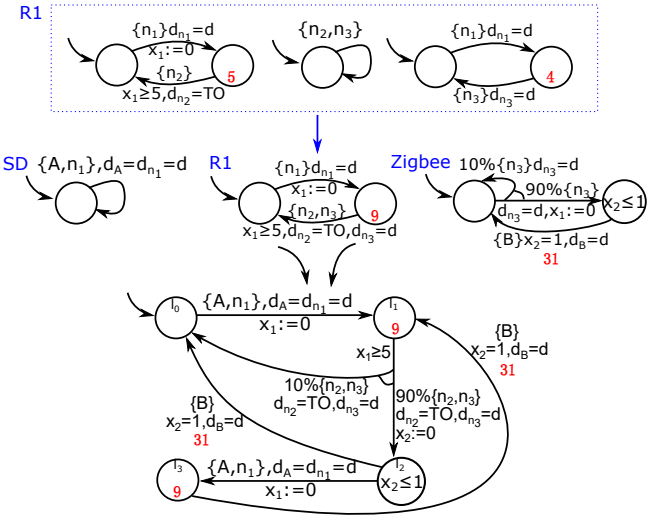


Fig. 14 pPTCA for the exemplary IoT system.

data through A and **W** reads data from B. We use numbers with ¢(i.e. Kilo Joule) to denote energy consumption, and numbers with no units for time.

Based on the pPTCA $\mathcal{A}$ corresponding to this example in Figure 14, one can identify the target run that transmits data to **W** with lowest energy cost as follows:

$$\mathsf{r}_1 = l_0 \xrightarrow{\text{true},t_1,\pi_0} l_1 \xrightarrow{l_1=5,5,\pi_1} l_2 \xrightarrow{x_2=1,1,\pi_2} l_0$$

where $e_0 = \langle\{A,\mathrm{n}_1\}, d_A = d_{\mathrm{n}_1} = d, x_1 := 0, l_1\rangle$, $e_1 = \langle\{\mathrm{n}_2,\mathrm{n}_3\}, d_{\mathrm{n}_2} = \mathrm{TO} \wedge d_{\mathrm{n}_3} = d, x_1 := 0, l_2\rangle$, $e_2 = \langle\{B\}, d_B = d, \emptyset, l_0\rangle$, $\pi_0(e_0) = 1, \pi_1(e_1) = 0.9, \pi_2(e_2) = 1$. The energy cost of r_1 is $\mathsf{TC}(\mathsf{r}_1) = (\rho(l_0) \cdot t_1 + \rho(e_0)) + (\rho(l_1) \cdot 5 + \rho(e_1)) + (\rho(l_2) \cdot 1 + \rho(e_2)) = 76¢$. The reach probability is $\mathsf{P}_1 = 1 \cdot 0.9 \cdot 1 = 0.9$.

For this system, one interesting property is that 'it is possible whenever the robot sends some data, less than 6 time units within which time no new data is sent by the robot, the workstation receives the same data', which is specified by pPTDL-formula

$$\phi_1 = \bigwedge_{d \in Data} \mathbf{P}_{>0}(\Box([\![\langle A, d_A = d\rangle]\!]\langle\!\langle((\neg\langle A\rangle)^*\langle B, d_B = d\rangle)^{<6}\rangle\!\rangle))$$

where $\neg\langle A, \delta_A\rangle$ refers to the disjunction of all pTsD expressions $\langle N, \delta_N\rangle$ that are not pairwise equal to $\langle A, \delta_A\rangle$. However ϕ_1 is not satisfied by $\mathcal{A}$ of the system, i.e., $s_0 = \langle l_0, \mathbf{0}\rangle \not\models \phi_1$ where s_0 is the initial state in $\mathcal{M}^{\mathcal{A}}$. Let us consider the data-

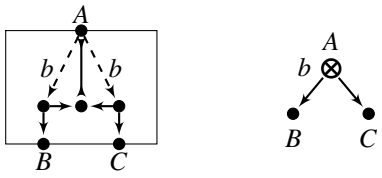


Fig. 15 Router connector and its instance.

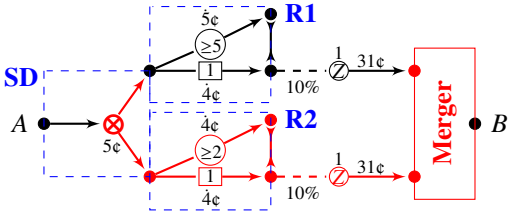


Fig. 16 Reo circuit for the updated exemplary system.

flow stream with the possible shortest execution time. Assume initially at s_0, one data $d{\in}Data$ is written by the node A at time 0, then a transition is taken moving to the state $s_1 = \langle l_1, x_1{=}x_2{=}0\rangle$. After 5 time units for the data processing, the probabilistic choice to successfully send the data by Zigbee channel is determined immediately moving to state $s_2 = \langle l_2, x_1{=}5{\wedge}x_2{=}0\rangle$. Then after 1 time unit, d is read from node B without any delay, and state $s_3 = \langle l_0, x_1{=}6{\wedge}x_2{=}1\rangle$ is reached. This procedure coincides with the target run r_1 where $t_1 = 0$, and the induced pTsD stream is

$$\Theta(\mathsf{r}_1) = (0, \langle\{A, \mathrm{n}_1\}, d_A{=}d_{\mathrm{n}_1}{=}d\rangle, 0)$$
$$(5, \langle\{\mathrm{n}_2, \mathrm{n}_3\}, d_{\mathrm{n}_2}{=}\mathrm{TO}{\wedge}d_{\mathrm{n}_3}{=}d\rangle, 45)(6, \langle\{B\}, d_B{=}d\rangle, 76).$$

It is easy to see ϕ_1 does not hold for $\Theta(\mathsf{r}_1)$, since the shortest execution time is 6 time units. Therefore we have $\mathcal{A} \not\models \phi_1$. Similarly, we can verify the system satisfies the following property 'once the robot sends a data, the workstation will eventually receive the same data' presented by

$$\phi_2 = \bigwedge_{d\in Data} \mathbf{P}_{=1}(\Box(\llbracket\langle A, d_A = d\rangle\rrbracket\langle\!\langle\langle B, d_B = d\rangle^{\infty}\rangle\!\rangle)).$$

Properties like ϕ_2 are called *liveness* properties, which informally specify that 'something good' will happen in the future. One can also construct so-called *safety* properties stating 'something bad' never happens, by the use of probabilistic operators. For instance, the following pPTDL formula

$$\phi_3 = \bigwedge_{d\in Data} \mathbf{P}_{=0}(\Box(\langle\!\langle(\neg\langle A, d_A = d\rangle)^*\langle B, d_B = d\rangle\rangle\!\rangle))$$

shows that 'the workstation never receives a data that is not sent by the robot'. Apparently $\mathcal{A} \models \phi_3$, and therefore from such point of view we say this system is 'safe'. Particularly, the stream formulae within the probabilistic operator refer to the sets of pTsD streams induced from the *bad prefix* of this safety property.

Regarding the energy cost for the system, we can develop a property specifying 'once a data is sent by the robot, the accumulated cost of the system up to 5 time units, no matter whether the workstation receives the data or not, is at least 45 Kilo Joule' as

$$\phi_4 = \mathbf{R}^{rew_\rho}_{\geq 45}[\mathsf{A}^5].$$

With the help of energy price function ρ, it is not difficult to verify that ϕ_4 holds for the system.

6.2 Updating the system

Now let us consider an update to this system: Due to the increased financial budget and technical development, a new advanced robot **R2** is deployed in this unit. **R2** is faster and more effective (modelled with lower execution time and energy consumption rate for processing).

We first construct a compositional connector called *Router* in Figure 15 to model the task distribution where incoming tasks are routed to one of the robots. A Router is built out of four synchronous channels, two lossy synchronous channels, and one synchronous drain channel. A inputs some data item d, and either B or C outputs d simultaneously. The nondeterminism for the case that both B and C are ready is determined by the middle mixed node taking the data from only one of the two synchronous channels coinciding on it. On the right a simple mark is used to refer to the instance for Router. b represents the energy price for effectuating the distribution.

The Reo circuit for this updated system is shown in Figure 16. Notably, realizing such a flexible and adaptive communication and interaction pattern is readily possible by means of priced probabilistic timed Reo. The updated connectors are depicted in red. The corresponding pPTCA $\mathcal{B}$ in Figure 17 can be mechanically inspected to compute the lowest energy consumption and the corresponding probability for transmitting data to **W** either by **R1** or **R2**.

6.2.1 Sequential process

If two tasks are assigned for **R1** and **R2** respectively and by the order that one after another one finishes, we can find the following target runs:

$$\mathsf{r}_2 = l_0 \xrightarrow{\mathsf{true},t_2,\pi_0} l_1 \xrightarrow{x_1=5,5,\pi_1} l_2 \xrightarrow{x_3=1,1,\pi_2} l_0$$

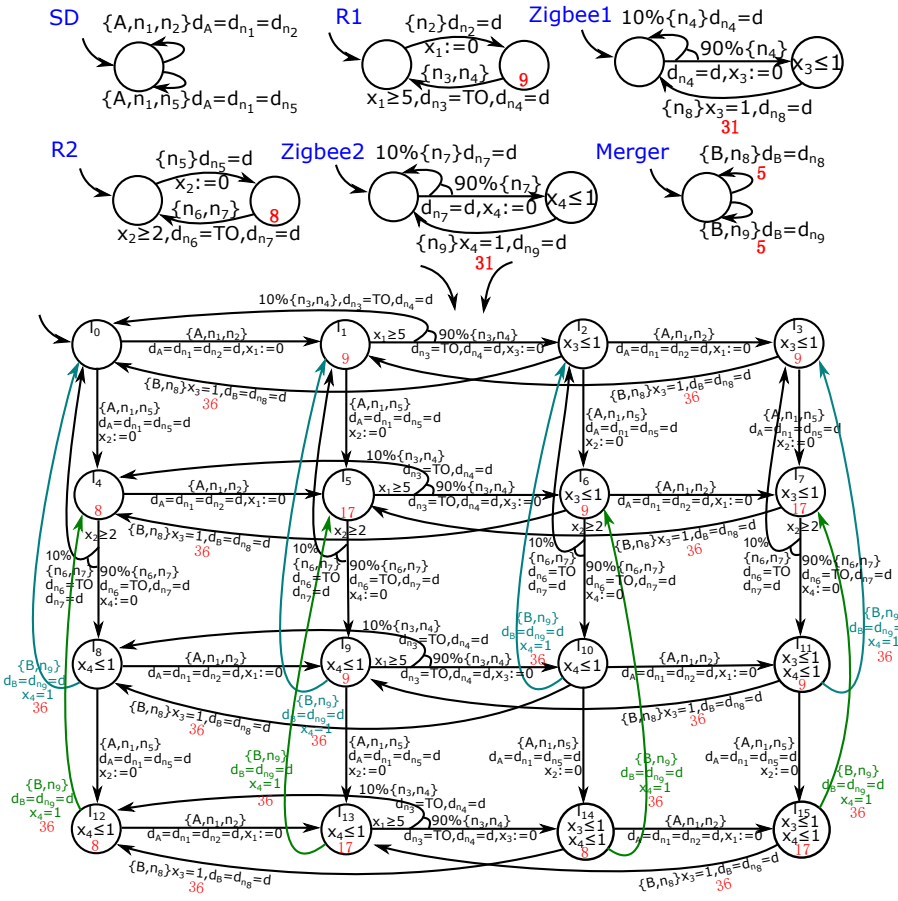


Fig. 17 pPTCA for the exemplary IoT system.

where e_0=⟨{A,n$_1$,n$_2$}, d_A=d_{n_1}=d_{n_2}=d, x_1:=0, l_1⟩, e_1=⟨{n$_3$,n$_4$}, d_{n_3}=TO∧d_{n_4}=d, x_3:=0, l_2⟩, e_2=⟨{B,n$_8$}, d_B=d_{n_8}=d, ∅, l_0⟩, $\pi_0(e_0)$=1, $\pi_1(e_1)$=0.9, $\pi_2(e_2)$=1, and

$$\mathsf{r}_3 = l_0 \xrightarrow{\text{true},t_3,\pi_0} l_4 \xrightarrow{x_2=2,2,\pi_1} l_8 \xrightarrow{x_4=1,1,\pi_2} l_0$$

where e_0=⟨{A,n$_1$,n$_5$}, d_A=d_{n_1}=d_{n_5}=d, x_2:=0, l_4⟩, e_1=⟨{n$_6$,n$_7$}, d_{n_6}=TO∧d_{n_7}=d, x_4:=0, l_8⟩, e_2=⟨{B,n$_9$}, d_B=d_{n_9}=d, ∅, l_0⟩, $\pi_0(e_0)$=1, $\pi_1(e_1)$=0.9, $\pi_2(e_2)$=1. Due to the task distribution, the average energy cost is $\frac{\mathsf{TC}(\mathsf{r}_2)+\mathsf{TC}(\mathsf{r}_3)}{2}$ = 66.5¢. And the probability for successful sequential process is $\mathsf{P}_2 = 0.9 \cdot 0.9 = 0.81$.

6.2.2 Concurrent process

Alternatively in another case that two tasks are processed concurrently (which is the exclusive feature of the updated system), we can find a target run as:

$$\mathsf{r}_4 = l_0 \xrightarrow{\text{true},t_4,\pi_0} l_1 \xrightarrow{\text{true},0,\pi_1} l_5 \xrightarrow{x_2=2,2,\pi_2} l_9 \xrightarrow{x_4=1,1,\pi_3} l_1 \xrightarrow{x_1=5,2,\pi_4} l_2 \xrightarrow{x_3=1,1,\pi_5} l_0$$

where e_0 = ⟨{A,n$_1$,n$_2$}, d_A = d_{n_1} = d_{n_2} = d, x_1 := 0, l_1⟩, e_1 = ⟨{A,n$_1$,n$_5$}, d_A=d_{n_1}=d_{n_5}=d, x_2:=0, l_5⟩, e_2=⟨{n$_6$,n$_7$}, d_{n_6}=TO, d_{n_7}=d, x_4 := 0, l_9⟩, e_3 = ⟨{B,n$_9$}, d_B = d_{n_9} = d, ∅, l_1⟩, e_4 = ⟨{n$_3$,n$_4$}, d_{n_3}= TO∧d_{n_4}=d, x_3:=0, l_2⟩, e_5=⟨{B,n$_8$}, d_B=d_{n_8}=d, ∅, l_0⟩, $\pi_0(e_0)$= 1, $\pi_1(e_1)$ = 1, $\pi_2(e_2)$ = 0.9, $\pi_3(e_3)$ = 1, $\pi_4(e_4)$ = 0.9, $\pi_5(e_5)$ = 1. The energy cost is $\mathsf{TC}(\mathsf{r}_4)$ = 133¢, and the probability is P_3 = 0.81.

Under both circumstances, the old system can only take sequential process, and it is easy to calculate the lowest energy consumption is TC = 76 · 2 = 152¢, the successful probability P_4 = 0.9 · 0.9 = 0.81. Obviously, in the case of consuming the lowest energy the updated system performs better than the old one while maintaining the same probability of success. This can be analysed in the perspective of pPTDL formula as follows. We first denote three stream formulae as

$$\psi_1 = \langle\!\langle \langle A, d_A = d_1\rangle\langle B, d_B = d_1\rangle\langle A, d_A = d_2\rangle\langle B, d_B = d_2\rangle \rangle\!\rangle$$
$$\psi_2 = \langle\!\langle \langle A, d_A = d_1\rangle\langle A, d_A = d_2\rangle\langle B, d_B = d_1\rangle\langle B, d_B = d_2\rangle \rangle\!\rangle$$
$$\psi_3 = \langle\!\langle \langle A, d_A = d_1\rangle\langle A, d_A = d_2\rangle\langle B, d_B = d_2\rangle\langle B, d_B = d_1\rangle \rangle\!\rangle$$

which respectively specify three independent cases about the order of data flows on robot and workstation. Then the property stating that 'the lowest energy cost for the workstation receiving the data, which consists of three independent cases

that guarantee the data is first sent by the robot, is greater than 152 Kilo Joule' is presented by

$$\phi_5 = \mathbf{R}^{rew_\rho}_{\geq 152}[\mathsf{L}(\psi_1)] \vee \mathbf{R}^{rew_\rho}_{\geq 152}[\mathsf{L}(\psi_2)] \vee \mathbf{R}^{rew_\rho}_{\geq 152}[\mathsf{L}(\psi_3)]$$

Now we need to verify this property of $\mathcal{A}$ and $\mathcal{B}$ respectively. With regards to $\mathcal{B}$, for each scheduler ζ, assume the sets of pTsD streams satisfying ψ_1, ψ_2, ψ_3 are respectively $\mathbf{pTsD}_i = \{\Theta(\omega) \models \psi_i\}$, where $i = 1, 2, 3$, $\omega \in \mathsf{FiPath}^{\zeta}_{s_0} \cup \mathsf{InPath}^{\zeta}_{s_0}$, and $s_0 = \langle l_0, \mathbf{0}\rangle$ is the initial state of $\mathcal{M}^{\mathcal{B}}$. It is straightforward to see the lowest energy costs for $\mathbf{pTsD}_i$ are all 133¢. The value of $\mathsf{L}(\psi_i)$ are all 133, and therefore $\mathcal{B} \not\models \phi_5$. Similarly, we can verify that $\mathcal{A} \models \phi_5$. The violation of ϕ_5 directly shows a lower value than 152 is gained for $\mathcal{B}$, which represents a lower energy cost by the updated system.

Based on pPTCA and pPTDL, one can explore more evaluations and also apply exhaustive verification techniques. For instance, we might be interested in the situation where **R1** and **R2** both have just completed a task and the resulting data is in transmission while no new tasks come in (modelled by location l_{10} in Figure 17). It is not difficult to calculate the minimal expected time for reaching that situation. Moreover the numbers of complete task execution (i.e., reaching l_0 from l_2 or l_8) under some bounded time can be computed, just as many other quantities of interest.

7 Related Work

7.1 IoT modelling languages

In recent years, varied efforts have been devoted to develop the formal modelling involving pricing information in the IoT domain. Li and Jin et al. developed an approach to model the reliability and cost of service composition in the IoT on the basis of Markov Decision Processes with cost structure [32]. Then in [33], Li and Wei et al. extended this work to model real-time constraints, where the IoT services and their corresponding environment can be described in Probabilistic Timed Automata. However, they focus on a level of services in SOC-based IoT. Martinez et al. [34] proposed a methodology for the power consumption of wireless network devices at the system level. Through this approach, application engineers can foretell how parameters impact power consumption and make estimates without a complete implementation of the application. This pricing model exclusively aims at analysing the energy life-cycles in applications, making its limitations on IoT systems with probabilistic and timed aspects. Costa et al. [35] proposed an approach to model IoT systems based on SysML profile, and further to apply NuSMV tool for model checking against CTL and LTL properties. This approach provides a similar idea for choosing graphical interface as the high-level modelling language, however it lacks the key feature of probabilistic aspect in IoT systems.

7.2 CA variants

Specifically, Lee has claimed that Reo plays a significant role on an emerging means to model Cyber Physical Systems (CPS) at the component interaction level [36]. Palomar et al. [37] developed a case study on the scalable smart city systems in CPS using Reo as the modelling language. A closely related model considering non-function requirements is the so-called resource-sensitive TCA (RSTCA) [5] where execution times for interactions are made dependent on resource availability and timeout behaviours. Relative to our approach RSTCA seems more restrictive, using implicit clocks on each transition. Another model, called Quantitative Constraint Automata [6], considers quantitative aspects which can be specified by so-called Q-algebra as constraints on transition. The motivation of this model is close in spirit to the one considered here, but it avoids to consider global time advance explicitly. In [38], Baier and Wolf developed Continuous-time CA (CCA) as extensions of CA with soft (memoryless) time, instead of hard time bounds (i.e., exact upper and lower bounds of time). Stochastic Reo is a variant of Reo appended with data arrival rates and processing delay rates. The approach targets soft real-time behaviours, too, and is thus similar in capabilities to the CCA approach. Two different underlying semantics have been studied [39, 40]. Unfortunately CCA can not represent nondeterminism. Motivated by the IoT context, we are interested in hard real-time behaviours mixed with probabilistic effects and nondeterministic behaviours.

7.3 IoT supported logics

The very basic logics for describing quantitative properties with priced, probabilistic and timed aspects in IoT systems are the reward extensions of PCTL. Like in [26], Norman et al. decorated a reward operator on the state formulae of PCTL to specify the reward value under three specific conditions. However, PCTL reflects directly how the states and paths behave, and abstracts the performance of data flows. And it is not able to express the LTL-flavour properties, which on the contrary are supported in PCTL*. Arbab et al. in [9] proposed a logic based on LTL [41] and equipped with a stream

modal operator to present properties of data flows. This logic naturally lacks the expression for probabilistic and priced aspects. We hence delicately combine parts of these logics in such a way that the reward operator from [26] is appended within the state formulae of PCTL*, the so-called path formulae of which are replaced by stream formulae as shown in Section 5, and the stream operator from [9] is inserted as a component of the stream formulae. By this means, the novel logic pPTDL is expressive enough for the required properties for IoT systems modelled by Reo and pPTCA.

8 Conclusion and Future Work

In this paper, we develop a priced, probabilistic and timed extension of Reo and Constraint Automata (which forms pPTCA) for the purpose of enabling a faithful modelling of IoT systems. Therefore, from the level of Reo, the modeller can easily construct (possible) large scalable IoT systems. Within the framework of pPTCA, where cost, time and probabilities are taken into consideration, the modeller can describe, on a single model, different aspects of an IoT system, and analyze real-time properties, performance, QoS and reliability properties. An expressive logic called pPTDL is proposed for the supporting properties and to verify the pPTCA models in terms of inclusion on pTsD-languages. A small example has demonstrated the principal expressiveness and modelling conveniences.

As future work, two problems are worth considering. One is pPTDL model checking where the key lies in the development of an automated verification algorithm. The other aspect that we consider worthwhile to explore is a semantics preserving translation to Modest [42, 43], which can then enable discrete event simulations of models developed for IoT and thus support the feasibility of our modelling approach from the experimental perspective.

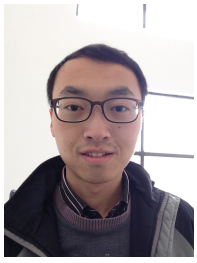
Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No.61370100, Grant No.61321064 and No.61773019), Shanghai Knowledge Service Platform for Trustworthy Internet of Things (Grant No.ZF1213), Shanghai Municipal Science and Technology Commission Project (No.1451100400), and Defense Industrial Technology Development Program JCKY 2016212B004-2, by the ERC Advanced Grant 695614 (POWVER), and by the Sino-German Center for Research Project CAP (GZ 1023).

References

1. Borgia E. The internet of things vision: key features, applications and open issues. Computer Communications, 2014, 54:1-31
2. Lanese I, Bedogni L, Felice M D. Internet of things: a process calculus approach. In: Proceedings of Annual ACM Symposium on Applied Computing, 2013, 1339-1346
3. Lanotte R, Merro M. A semantic theory of the internet of things. Information and Computation, 2018, 259(1):72-101
4. Arbab F. Reo: a channel-based coordination model for component composition. Mathematical Structures in Computer Science, 2004, 14(3):329-366
5. Meng S, Arbab F. On resource-sensitive timed component connectors. In: Proceedings of International Conference on Formal Methods for Open Object-Based Distributed Systems, 2007, 301-316
6. Arbab F, Chothia T, Meng S, Moon Y J. Component connectors with qos guarantees. In: Proceedings of International Conference on Coordination Models and Languages, 2007, 286-304
7. Baier C, Sirjani M, Arbab F, Rutten J J M M. Modeling component connectors in reo by constraint automata. Science of Computer Programming, 2006, 61(2):75-113
8. Jongmans S S T Q, Arbab F. Overview of thirty semantic formalisms for reo. Scientific Annals of Computer Science, 2012, 22(1):201-251
9. Arbab F, Baier C, Boer F S, Rutten J J M M. Models and temporal logical specifications for timed component connectors. Software and System Modeling, 2007, 6(1):59-82
10. Baier C. Probabilistic models for reo connector circuits. Journal of Universal Computer Science, 2005, 11(10):1718-1748
11. Aziz A, Singhal V, Balarin F. It usually works: the temporal logic of stochastic systems. In: Proceedings of International Conference on Computer Aided Verification, 1995, 155-165
12. He K, Hermanns H, Chen Y. Models of connected things: on priced probabilistic timed reo. In: Proceedings of IEEE Annual Computer Software and Applications Conference, 2017, 234-243
13. Arbab F, Rutten J J M M. A coinductive calculus of component connectors. In: Proceedings of International Workshop on Recent Trends in Algebraic Development Techniques, 2002, 34-55
14. Alur R, Dill D L. A theory of timed automata. Theoretical Computer Science, 1994, 126(2):183-235
15. Henzinger T A, Nicollin X, Sifakis J, Yovine S. Symbolic model checking for real-time systems. Information and Computation, 1994, 111(2):193-244
16. Segala R, Lynch N A. Probabilistic simulations for probabilistic processes. Nordic Journal of Computing, 1995, 2(2):250-273
17. Turrini A, Hermanns H. Cost preserving bisimulations for probabilistic automata. Logical Methods in Computer Science, 2014, 10(4):1-58
18. Segala R. Modelling and verification of randomized distributed real time systems. PhD thesis, Cambridge: Massachusetts Institute of Tech-

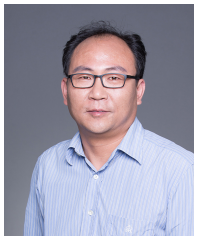
nology, 1995
19. Glabbeek R V, Smolka S A, Steffen B, Tofts C M N. Reactive, generative, and stratified models of probabilistic processes. In: Proceedings of Annual Symposium on Logic in Computer Science, 1990, 130-141
20. Glabbeek R V, Smolka S A, Steffen B. Reactive, generative and stratified models of probabilistic processes. Information and Computation, 1995, 121(1):59-80
21. D' Argenio P R, Hermanns H, Katoen J P. On generative parallel composition. Electronic Notes in Theoretical Computer Science, 1999, 22:30-54
22. Baeten J C M, Bergstra J A, Smolka S A. Axiomatizing probabilistic processes: acp with generative probabilities. Information and Computation, 1995, 121(2):234-255
23. Kwiatkowska M Z, Norman G, Parker D, Sproston J. Performance analysis of probabilistic timed automata using digital clocks. Formal Methods in System Design, 2006, 29(1):33-78
24. Halmos P R. Measure theory. Berlin: Springer-Verlag, 1950
25. Clarke E M, Emerson E A. Design and synthesis of synchronization skeletons using branching-time temporal logic. In: Proceedings of Workshop on Logics of Programs 1981, 52-71
26. Norman G, Parker D, Sproston J. Model checking for probabilistic timed automata. Formal Methods in System Design, 2013, 43(2):164-190
27. Baier C, Haverkort B R, Hermanns H, Katoen J P. Performance evaluation and model checking join forces. Communications of the ACM, 2010, 53(9):76-85
28. Bianco A, Alfaro L D. Model checking of probabilistic and nondeterministic systems. In: Proceedings of International Conference on Foundations of Software Technology and Theoretical Computer Science, 1995, 499-513
29. Asarin E, Caspi P, Maler O. Timed regular expressions. Journal of the ACM, 2002, 49(2):172-206
30. Hansson H, Jonsson B. A logic for reasoning about time and reliability. Formal Aspects of Computing, 1994, 6(5):512-535
31. Bianco A, Alfaro L D. Model checking of probabalistic and nondeterministic systems. In: Proceedings of International Conference on Foundations of Software Technology and Theoretical Computer Science, 1995, 499-513
32. Li L, Jin Z, Li G, Zheng L, Wei Q. Modeling and analyzing the reliability and cost of service composition in the iot: a probabilistic approach. In: Proceedings of International Conference on Web Services, 2012, 584-591
33. Li G, Wei Q, Li X, Jin Z, Xu Y, Zheng L. Environment based modeling approach for services in the internet of things. Scientia Sinica, 2013, 43(10):1198-1218
34. Martinez B, Montón M, Vilajosana I, Prades J D. The power of models: modeling power consumption for iot devices. IEEE Sensors Journal, 2015, 15(10):5777-5789
35. Costa B, Pires P F, Delicato F C, Li W, Zomaya A Y. Design and analysis of iot applications: a model-driven approach. In: Proceedings of IEEE International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress, 2016, 392-399
36. Lee E A. Cyber physical systems: design challenges. In: Proceedings of IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2008, 363-369
37. Palomar E, Chen X, Liu Z, Maharjan S, Bowen J P. Component-based modelling for scalable smart city systems interoperability: a case study on integrating energy demand response systems. Sensors, 2016, 16(11):1810
38. Baier C, Wolf V. Stochastic reasoning about channel-based component connectors. In: Proceedings of International Conference on Coordination Models and Languages, 2006, 1-15
39. Moon Y J, Silva A, Krause C, Arbab F. A compositional semantics for stochastic reo connectors. In: Proceedings of International Workshop on the Foundations of Coordination Languages and Software Architectures, 2010, 93-107
40. Oliveira N, Silva A, Barbosa L A. Imcreo: interactive markov chains for stochastic reo. Journal of Internet Services and Information Security, 2015, 5(1):3-28
41. Pnueli A. The temporal logic of programs. In: Proceedings of Annual Symposium on Foundations of Computer Science, 1977, 46-57
42. Bohnenkamp H C, D' Argenio P R, Hermanns H, Katoen J P. Modest: a compositional modeling formalism for hard and softly timed systems. IEEE Transactions on Software Engineering, 2006, 32(10):812-830
43. Hahn E M, Hartmanns A, Hermanns H, Katoen J P. A compositional modelling and analysis framework for stochastic hybrid systems. Formal Methods in System Design, 2013, 43(2):191-232



Kangli He is a PhD student in the School of Computer Science and Software Engineering in the East China Normal University, Shanghai, China. He received the B.S. degree from East China Normal University in 2009. His research interests include formal methods, Internet of Things and Bisimulation. Now his research topic concerns the formal modelling and verification of Internet of Things.

Holger Hermanns is a full professor at Saarland University, Saarbrücken, Germany, holding the chair of Dependable Systems and Software on Saarland Informatics Campus. He is an ERC Advanced Grantee and member of Academia Europaea. His research interests include perspicuous computing, modeling and verification of concurrent systems, resource-aware

embedded systems, compositional performance and dependability evaluation, and their applications to energy informatics. Holger Hermanns has co-authored more than 200 peer-reviewed scientific papers (ha-index 92, h-index 50). He co-chaired the program committees of major international conferences such as CAV, CONCUR, TACAS and QEST, and delivered keynotes at about a dozen international conferences and symposia. He serves on the steering committees of ETAPS and TACAS. He is president of the association "Friends of Dagstuhl".



Hengyang Wu received the B.S. degree from Xuzhou Normal University, Xuzhou, China, in 1996, and the M.Sc. and Ph.D. degrees from Shanghai Normal University, Shanghai, China, in 2004 and 2007, respectively, all in mathematics. From 2008 to 2011, he was a Postdoctoral Researcher with East China Normal University, Shanghai, China, where he is currently an Associate Research Fellow. From 2011 to July 2016, he was an Associate Professor of computer science with Hangzhou Dianzi University. His current research interests include formal methods and domain theory. He serves on the thechnical committees of Fuzzy Systems and Mathemantics and CCF Theoretical Computer Science.

Yixiang Chen is a full professor in the School of Computer Science and Software Engineering, East China Normal University, Shanghai, China. Where he is coordinating trustworthy software, Internet of things and Human-Cyber-Physical System related research activities. Professor Chen is the director of the MoE Engineering Research Center for Software/Hardware Co-design Technology and Application. He is a vice-chairman of technical committee for Embedded System China Computer Federation, Fuzzy Systems and Mathemantics, Chnese Association for Artificial Intelligence.