A Semantic Theory of the Internet of Things (extended abstract)

Ruggero Lanotte[1] and Massimo Merro[2]

[1] Dipartimento di Scienza e Alta Tecnologia, Università dell'Insubria, Como, Italy
[2] Dipartimento di Informatica, Università degli Studi di Verona, Italy

Abstract. We propose a process calculus for modelling and reasoning on systems in the *Internet of Things* paradigm. Our systems interact both with the physical environment, via *sensors* and *actuators*, and with *smart devices*, via short-range and Internet channels. The calculus is equipped with a standard notion of labelled *bisimilarity* which represents a fully abstract characterisation of a well-known contextual equivalence. We use our semantic proof-methods to prove run-time properties of a non-trivial case study as well as system equalities.

1 Introduction

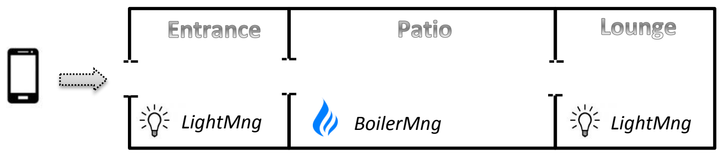
In the *Internet of Things* (IoT) paradigm, *smart devices*, such as smartphones, automatically collect information from shared resources (e.g. Internet access or physical devices) and aggregate them to provide new services to end users [13]. The "things" commonly deployed in IoT systems are: *RFID tags*, for unique identification, *sensors*, to detect physical changes in the environment, and *actuators*, to pass information to the environment.

The research on IoT is currently focusing on practical applications such as the development of enabling technologies, ad hoc architectures, semantic web technologies, and cloud computing [13]. However, as pointed out by Lanese et al. [16], there is a lack of research in formal methodologies to model the interactions among system components, and to verify the correctness of the network deployment before its implementation.

The main goal of this paper is to propose a process calculus with a clearly-defined semantic theory, for specifying and reasoning on IoT applications. Designing a calculus for modelling a new paradigm requires understanding and distilling, in a clean algebraic setting, the main features of the paradigm. Let us try to figure out what the main ingredients of IoT are, by means of an example.

Suppose a simple *smart home* (see Fig. 1) in which the user can use her smartphone to remotely control the heating boiler of her house, and automatically turn on lights when entering a room. The house consists of an entrance and a lounge, separated by a patio. Entrance and lounge have their own lights (actuators) which are governed by different light manager processes, $LightMng$. The boiler is in the patio and is governed by a boiler manager process, $BoilerMng$. This process senses the local temperature (via a sensor) and decides whether to turn on/off the boiler, setting a proper actuator to signal the state of the boiler.

Fig. 1 A simple smart home



The smartphone executes two concurrent processes: $BoilerCtrl$ and $LightCtr$. The first one reads user's commands, submitted via the phone touchscreen (a sensor), and forward them to the process $BoilerMng$, via an Internet channel. Whereas, the process $LightCtrl$ interacts with the processes $LightMng$, via short-range wireless channels (e.g. Bluetooth), to automatically turn on lights when the smartphone physically enters either the entrance or the lounge.

The whole system is given by the parallel composition of the smartphone (a mobile device) and the smart home (a stationary entity).

On this kind of systems one may wish to prove interesting *run-time properties*. Think of a *fairness* property saying that the boiler will be eventually turned on/off whenever specific conditions are satisfied. Or consistency properties, saying the smartphone will never be in two rooms at the same time. Even more, one may be interested in understanding whether our system has the same *observable behaviour* of another system. Let us consider a variant of our smart home, where lights functionality depends on GPS coordinates of the smartphone (localisation is a common feature of today smartphones). Intuitively, the smartphone sends its GPS position to a centralised light manager, $CLightMng$ (possibly placed in the patio), via an Internet channel. The process $CLightMng$ will then interact with the two processes $LightMng$, via short-range channels, to switch on/off lights, depending on the position of the smartphone. Here comes an interesting question: Can these two implementations of the smart home, based on different light management mechanisms, be actually distinguished by an end user?

In the paper at hand we develop a fully abstract semantic theory for a process calculus of IoT systems, called `CaIT`. We provide a formal notion of when two systems in `CaIT` are indistinguishable, in all possible contexts, from the point of view of the end user. Formally, we adopt the approach of [15, 24], often called *reduction (closed) barbed congruence*, which relies on two crucial concepts: a *reduction semantics* to describe system computations, and the *basic observables* to represent what the environment can directly observe of a system. In `CaIT`, there are at least two possible observables: the ability to transmit along channels, *logical observation*, and the capability to diffuse messages via actuators, *physical observation*. We have adopted the second form as our contextual equality remains invariant when adding logical observation. However, the right definition of physical observation is far from obvious as it involves some technical challenges in the definition of the reduction semantics (see the discussion in Sec. 2.3).

Our calculus is equipped with two *labelled transition semantics* (LTS) in the SOS style of Plotkin: an *intensional* semantics and an *extensional* semantics. The adjective intensional is used to stress the fact that the actions here correspond to activities which can be performed by a system in isolation, without any interaction with the external environment. While, the extensional semantics focuses on those activities which require a contribution of the environment. We prove that the reduction semantics coincides with the intensional semantics (Harmony Theorem), and that they satisfy some desirable time properties such as *time determinism*, *patience*, *maximal progress* and *well-timedness* [14].

However, the main result of the paper is that weak *bisimilarity* in the extensional LTS is *sound* and *complete* with respect to our contextual equivalence, reduction barbed congruence. This required a non-standard proof of the congruence theorem (Thm. 2). Finally, in order to show the effectiveness of our bisimulation proof method, we prove a number of non-trivial system equalities.

In this extended abstract proofs are omitted; full details can be found in [7].

Outline Sec. 2 contains the calculus together with the reduction semantics, the contextual equivalence, and a discussion on design choices. Sec. 3 gives the details of our smart home example, and proves desirable run-time properties for it. Sec. 4 defines both the intensional and the extensional LTS. In Sec. 5 we define bisimilarity for IoT-systems, and prove the full abstraction result together with a number of non-trivial system equalities. Sec. 6 discusses related work.

2 The calculus

The syntax of our *Calculus of the Internet of Things*, shortly `CaIT`, is given in a two-level structure: a lower one for *processes* and an upper one for *networks* of smart devices.

$$\begin{array}{rcl} M, N & ::= & \mathbf{0} \quad \big| \quad n[\mathcal{I} \bowtie P]^{\mu}_{l} \quad \big| \quad M \mid N \quad \big| \quad (\boldsymbol{\nu} c)M \\ P, Q & ::= & \mathsf{nil} \quad \big| \quad \rho.P \quad \big| \quad P \mid Q \quad \big| \quad \lfloor \pi.P \rfloor Q \quad \big| \quad [b]P;Q \quad \big| \quad X \quad \big| \quad \mathsf{fix}\, X.P \end{array}$$

We use letters n, m to denote *nodes/devices*, c, g for *channels*, l, h, k for (physical) *locations*, s, s' for *sensors*, a, a' for *actuators* and x, y, z for *variables*. Our *values*, ranged over by v and w, are constituted by basic values, such as booleans and integers, sensor and actuator values, and coordinates of physical locations.

A network M is a pool of *distinct nodes* running in parallel and living in physical locations. We assume a discrete notion of *distance* between two locations h and k, i.e. $\mathrm{d}(h, k) \in \mathbb{N}$. We write $\mathbf{0}$ to denote the empty network, while $M \mid N$ represents the parallel composition of two networks M and N. In $(\boldsymbol{\nu} c)M$ channel c is private to the nodes of M. Each node is a term of the form $n[\mathcal{I} \bowtie P]^{\mu}_{l}$, where n is the device ID; $\mathcal{I}$ is the physical interface of n, represented as a partial mapping from sensor and actuator names to physical values; P is the process modelling the logics of n; l is the physical location of the device; $\mu \in \{\mathtt{s}, \mathtt{m}\}$ is a tag to distinguish between stationary and mobile nodes.

For security reasons, sensors in $\mathcal{I}$ can be read only by its *controller process* P. Similarly, actuators in $\mathcal{I}$ can be modified only by P. No other devices can access the physical interface of n. P is a timed concurrent processes which manages both the interaction with the physical interface $\mathcal{I}$ and channel communication. The communication paradigm is *point-to-point* via channels that may have different transmission ranges. We assume a global function rng() from channel names to $\mathbb{N} \cup \{-1, \infty\}$. A channel c can be used for: i) *intra-node communications*, if $\mathrm{rng}(c) = -1$; ii) *short-range inter-node communications* (such as Bluetooth, infrared, etc) if $0 \leq \mathrm{rng}(c) < \infty$; iii) *Internet communications*, if $\mathrm{rng}(c) = \infty$.

Technically, our processes build on CCS with discrete time [14]. We write $\rho.P$, with $\rho \in \{\sigma, @(x), s?(x), a!v\}$, to denote intra-node actions. The process $\sigma.P$ sleeps for one time unit. The process $@(x).P$ gets the current location of the enclosing node. Process $s?(x).P$ reads a value v from sensor s. Process $a!v.P$ writes the value v on the actuator a. We write $\lfloor \pi.P \rfloor Q$, with $\pi \in \{\bar{c}\langle v \rangle, c(x)\}$, to denote channel communication with timeout. This process can communicate in the current time interval and then continues as P; otherwise, after one time unit, it evolves into Q. We write $[b]P; Q$ for conditional (here guard $[\![b]\!]$ is always *decidable*). In processes of the form $\sigma.Q$ and $\lfloor \pi.P \rfloor Q$ the occurrence of Q is said to be *time-guarded*. The process $\mathsf{fix}\, X.P$ denotes *time-guarded recursion*, as all occurrences of the process variable X may only occur time-guarded in P. In processes $\lfloor c(x).P \rfloor Q$, $s?(x).P$ and $@(x).P$ the variable x is said to be bound. Similarly, in process $\mathsf{fix}\, X.P$ the process variable X is bound. In the term $(\boldsymbol{\nu} c)M$ the channel c is bound. This gives rise to the standard notions of *free/bound (process) variables*, *free/bound channels*, and *α-conversion*. A term is said to be *closed* if it does not contain free (process) variables, although it may contain free channels. We always work with closed networks: the absence of free variables is preserved at run-time. We write $T\{^v/_x\}$ for the substitution of the variable x with the value v in any expression T of our language. Similarly, $T\{^P/_X\}$ is the substitution of the process variable X with the process P in T.

Actuator names are metavariables for actuators like *display*$@n$ or *alarm*$@n$. As node names are unique so are actuator names: different nodes have different actuators. The sensors embedded in a node can be of two kinds: *location-dependent* and *node-dependent*. The first ones sense data at the current location of the node, whereas the second ones sense data within the node, independently on the node's location. Thus, node-dependent sensor names are metavariables for sensors like *touchscreen*$@n$ or *button*$@n$; whereas a sensor *temp*$@h$, for temperature, is a typical example of location-dependent sensor. Node-dependent sensor names are unique. This is not the case of location-dependent sensor names which may appear in different nodes. For simplicity, we use the same metavariables for both kind of sensors. When necessary we will specify the type of sensor in use.

We rule out ill-formed networks by means of the following definition.

Definition 1. *A network M is said to be* well-formed *if: (i) it does not contain two nodes with the same name; (ii) different nodes have different actuators and different node-dependent sensors; (iii) for each $n[\mathcal{I} \bowtie P]^{\mu}_{h}$ in M, with a prefix $s?(x)$ (resp. $a!v$) in P, $\mathcal{I}(s)$ (resp. $\mathcal{I}(a)$) is defined; (iv) for each $n[\mathcal{I} \bowtie P]^{\mu}_{h}$ in M with $\mathcal{I}(s)$ defined for some location-dependent sensor s, it holds that $\mu = \mathtt{s}$.*

Last condition says that location-dependent sensors may be used only in stationary nodes (see discussion in Sec. 2.3). Hereafter, we will always work with *well-formed networks*. It is easy to show that well-formedness is preserved at runtime.

We adopt the following notational conventions. $\prod_{i \in I} M_i$ denotes the parallel composition of all M_i, for $i \in I$. $\prod_{i \in I} M_i = \mathbf{0}$ and $\prod_{i \in I} P_i = \mathsf{nil}$, for $I = \emptyset$. We write $\prod_i M_i$ when I is not relevant. We write $\pi.P$ instead of $\mathsf{fix}\, X.\lfloor \pi.P \rfloor X$. We use $(\boldsymbol{\nu} \tilde{c})M$ as an abbreviation for $(\nu c_1) \ldots (\nu c_k)M$, with $\tilde{c} = c_1, \ldots, c_k$.

Table 1 Reduction semantics

$$\text{(pos)}\ \frac{-}{n[\mathcal{I} \bowtie @(x).P]^{\mu}_{h} \rightarrowtriangle_{\tau} n[\mathcal{I} \bowtie P\{^{h}/_{x}\}]^{\mu}_{h}} \qquad \text{(sensread)}\ \frac{\mathcal{I}(s) = v}{n[\mathcal{I} \bowtie s?(x).P]^{\mu}_{h} \rightarrowtriangle_{\tau} n[\mathcal{I} \bowtie P\{^{v}/_{x}\}]^{\mu}_{h}}$$

$$\text{(actunchg)}\ \frac{\mathcal{I}(a) = v}{n[\mathcal{I} \bowtie a!v.P]^{\mu}_{h} \rightarrowtriangle_{\tau} n[\mathcal{I} \bowtie P]^{\mu}_{h}} \qquad \text{(actchg)}\ \frac{\mathcal{I}(a) \neq v \quad \mathcal{I}' := \mathcal{I}[a \mapsto v]}{n[\mathcal{I} \bowtie a!v.P]^{\mu}_{h} \rightarrowtriangle_{a} n[\mathcal{I}' \bowtie P]^{\mu}_{h}}$$

$$\text{(loccom)}\ \frac{\mathrm{rng}(c) = -1}{n[\mathcal{I} \bowtie \lfloor \overline{c}\langle v \rangle.P \rfloor R \mid \lfloor c(x).Q \rfloor S]^{\mu}_{h} \ \rightarrowtriangle_{\tau}\ n[\mathcal{I} \bowtie P \mid Q\{^{v}/_{x}\}]^{\mu}_{h}}$$

$$\text{(timestat)}\ \frac{n[\mathcal{I} \bowtie \prod_i \lfloor \pi_i.P_i \rfloor Q_i \ \mid\ \prod_j \sigma.R_j]^{\mathtt{s}}_{h} \ \not\rightarrowtriangle_{\tau}}{n[\mathcal{I} \bowtie \prod_i \lfloor \pi_i.P_i \rfloor Q_i \ \mid\ \prod_j \sigma.R_j]^{\mathtt{s}}_{h} \rightarrowtriangle_{\sigma} n[\mathcal{I} \bowtie \prod_i Q_i \ \mid\ \prod_j R_j]^{\mathtt{s}}_{h}}$$

$$\text{(timemob)}\ \frac{n[\mathcal{I} \bowtie \prod_i \lfloor \pi_i.P_i \rfloor Q_i \ \mid\ \prod_j \sigma.R_j]^{\mathtt{m}}_{h} \ \not\rightarrowtriangle_{\tau} \qquad \mathrm{d}(h,k) \leq \delta}{n[\mathcal{I} \bowtie \prod_i \lfloor \pi_i.P_i \rfloor Q_i \ \mid\ \prod_j \sigma.R_j]^{\mathtt{m}}_{h} \rightarrowtriangle_{\sigma} n[\mathcal{I} \bowtie \prod_i Q_i \ \mid\ \prod_j R_j]^{\mathtt{m}}_{k}}$$

$$\text{(glbcom)}\ \frac{\mathrm{d}(h,k) \leq \mathrm{rng}(c)}{n[\mathcal{I} \bowtie \lfloor \overline{c}\langle v \rangle.P \rfloor R]^{\mu_1}_{h} \ \mid\ m[\mathcal{I} \bowtie \lfloor c(x).Q \rfloor S]^{\mu_2}_{k} \rightarrowtriangle_{\tau} n[\mathcal{I} \bowtie P]^{\mu_1}_{h} \ \mid\ m[\mathcal{I} \bowtie Q\{^{v}/_{x}\}]^{\mu_2}_{k}}$$

$$\text{(parp)}\ \frac{\prod_i n_i[\mathcal{I}_i \bowtie P_i]^{\mu_i}_{h_i} \rightarrowtriangle_{\omega} \prod_i n_i[\mathcal{I}'_i \bowtie P'_i]^{\mu_i}_{h_i} \quad \omega \in \{\tau, a\}}{\prod_i n_i[\mathcal{I}_i \bowtie P_i \mid Q_i]^{\mu_i}_{h_i} \rightarrowtriangle_{\omega} \prod_i n_i[\mathcal{I}'_i \bowtie P'_i \mid Q_i]^{\mu_i}_{h_i}} \qquad \text{(parn)}\ \frac{M \rightarrowtriangle_{\omega} M' \quad \omega \in \{\tau, a\}}{M \mid N \rightarrowtriangle_{\omega} M' \mid N}$$

$$\text{(timepar)}\ \frac{M \rightarrowtriangle_{\sigma} M' \quad N \rightarrowtriangle_{\sigma} N' \quad M \mid N \not\rightarrowtriangle_{\tau}}{M \mid N \rightarrowtriangle_{\sigma} M' \mid N'} \qquad \text{(timezero)}\ \frac{-}{\mathbf{0} \rightarrowtriangle_{\sigma} \mathbf{0}}$$

$$\text{(res)}\ \frac{M \rightarrowtriangle_{\omega} N \quad \omega \in \{\tau, a, \sigma\}}{(\boldsymbol{\nu} c)M \rightarrowtriangle_{\omega} (\boldsymbol{\nu} c)N} \qquad \text{(struct)}\ \frac{M \equiv N \ \ N \rightarrowtriangle_{\omega} N' \ \ \omega \in \{\tau, a, \sigma\} \ \ N' \equiv M'}{M \rightarrowtriangle_{\omega} M'}$$

2.1 Reduction semantics

The dynamics of the calculus is specified in terms of *reduction relations* over networks (see Tab. 1). As usual in process calculi, a reduction semantics [22] relies on an auxiliary standard relation, $\equiv$, called *structural congruence*, which brings the participants of a potential interaction into contiguous positions. For lack of space, we omit the formal definition of $\equiv$, as it is quite standard.

As `CaIT` is a timed calculus, with a discrete notion of time, it will be necessary to distinguish between *instantaneous reductions*, $M \rightarrowtriangle_{\mathrm{i}} N$, and *timed reductions*, $M \rightarrowtriangle_{\sigma} N$. Relation $\rightarrowtriangle_{\mathrm{i}}$ denotes activities which take place within one time interval, whereas $\rightarrowtriangle_{\sigma}$ represents the passage of one time unit. Instantaneous reductions are of two kinds: those which involve the change of the values associated to some actuator a, written $\rightarrowtriangle_{a}$, and the others, written $\rightarrowtriangle_{\tau}$. Intuitively, reductions of the form $M \rightarrowtriangle_{a} N$ denote *watchpoints* which cannot be ignored by the physical environment (in Ex. 2, and more extensively at the end of Sec. 2.3, we explain why this is important). Thus, we define the instantaneous reduction relation $\rightarrowtriangle_{\mathrm{i}} = \rightarrowtriangle_{\tau} \cup \rightarrowtriangle_{a}$, for any actuator a. We also define the reduction $\rightarrowtriangle = \rightarrowtriangle_{\tau} \cup \rightarrowtriangle_{\sigma}$.

The first seven rules in Tab. 1 model intra-node activities. Rule (pos) serves to compute the current position of a node. Rule (sensread) represents the reading of the current data detected at some sensor s. Rules (actunchg) and (actchg) implement the writing of some data v on an actuator a, distinguishing whether the value of the actuator changes or not. Rule (loccom) models intra-node communications on a local channel c ($\mathrm{rng}(c) = -1$). Rule (timestat) models the passage of time within a stationary node. Notice that all untimed intra-node actions are considered urgent actions as they must occur before the next timed action. Rule (timemob) models the passage of time for mobile nodes. This rule also serves to model *node mobility*. Mobile nodes can nondeterministically move from one physical location h to a (possibly different) location k, at the end of a time interval. Node mobility respects the following time discipline: in one time unit a node can move from h to k provided that $\mathrm{d}(h,k) \leq \delta$, for some fixed $\delta \in \mathbb{N}$ (if $h = k$ then $\mathrm{d}(h,k) = 0$). For the sake of simplicity, we fix the same constant δ for all nodes of our systems. Rule (glbcom) models inter-node communication along a global channel c ($\mathrm{rng}(c) \geq 0$). Intuitively, two nodes can communicate via a channel c only if they are within the transmission range of c. Rules (parp) and (parn) serve to propagate instantaneous reductions through parallel processes, and parallel networks, respectively. Rule (timepar) is for inter-node time synchronisation. The remaining rules are standard.

We write $\twoheadrightarrow_{\mathrm{i}}^{k}$ to denote k consecutive reductions $\twoheadrightarrow_{\mathrm{i}}$; $\twoheadrightarrow_{\mathrm{i}}^{*}$ is the reflexive and transitive closure of $\twoheadrightarrow_{\mathrm{i}}$. We use the same notation for the reduction relation $\twoheadrightarrow$.

Below we report a few standard time properties which hold in our calculus: *time determinism*, *maximal progress*, *patience* and *well-timedness*.

Proposition 1 (Time properties).

- *If* $M \twoheadrightarrow_{\sigma} M'$ *and* $M \twoheadrightarrow_{\sigma} M''$, *then* $M' \equiv \prod_{i\in I} n_i[\mathcal{I}_i \bowtie P_i]^{\mu_i}_{h_i}$ *and* $M'' \equiv \prod_{i\in I} n_i[\mathcal{I}_i \bowtie P_i]^{\mu_i}_{k_i}$, *with* $\mathrm{d}(h_i, k_i) \leq 2\delta$, *for all* $i \in I$.
- *If* $M \twoheadrightarrow_{\mathrm{i}} M'$, *then there is no* M'' *such that* $M \twoheadrightarrow_{\sigma} M''$.
- *If* $M \twoheadrightarrow_{\mathrm{i}} M'$ *for no* M', *then there is* N *such that* $M \twoheadrightarrow_{\sigma} N$.
- *For any* M *there is a* $z \in \mathbb{N}$ *such that if* $M \twoheadrightarrow_{\mathrm{i}}^{u} N$ *then* $u \leq z$.

In its standard formulation, time determinism says that a system reaches at most one new state by executing a reduction step $\twoheadrightarrow_{\sigma}$. However, by an application of Rule (timemob), our mobile nodes may change location when executing a reduction $\twoheadrightarrow_{\sigma}$. *Well-timedness* ensures the absence of infinite instantaneous traces which would prevent the passage of time.

2.2 Behavioural equivalence

Our contextual equivalence is *reduction barbed congruence* [15, 24], a standard contextually defined process equivalence that crucially relies on the definition of *basic observables* to represent what the environment can directly observe of a system[3]. As already said in the Introduction, we choose to observe the capability to publish messages via actuators (physical observation).

[3] See [24] for a comparison between this approach and the original barbed congruence.

Definition 2 (Barbs). *We write* $M \downarrow_{a@h!v}$ *if* $M \equiv (\boldsymbol{\nu}\tilde{g})\big(n[\mathcal{I} \bowtie P]^{\mu}_{h} \mid M'\big)$*, with* $\mathcal{I}(a) = v$*. We write* $M \Downarrow_{a@h!v}$ *if* $M \twoheadrightarrow^{*} M' \downarrow_{a@h!v}$*.*

The reader may wonder why our barb reports the location and not the node of the actuator. We also recall that actuator names are unique, so they somehow codify the name of their node. The location is then necessary because the environment is potentially aware of its position when observing an actuator: if every day at 6.00AM your smartphone rings to wake you up, then you may react differently depending whether you are at home or on holidays in the Bahamas!

Definition 3. *A binary relation* $\mathcal{R}$ *over networks is* barb preserving *if* $M \mathcal{R} N$ *and* $M \downarrow_{a@h!v}$ *implies* $N \Downarrow_{a@h!v}$*.*

Definition 4. *A binary relation* $\mathcal{R}$ *over networks is* reduction closed *if whenever* $M \; \mathcal{R} \; N$ *the following conditions are satisfied:*

– $M \twoheadrightarrow M'$ *implies* $N \twoheadrightarrow^{*} N'$ *and* $M' \; \mathcal{R} \; N'$
– $M \twoheadrightarrow_{a} M'$ *implies* $N \twoheadrightarrow^{*}\twoheadrightarrow_{a}\twoheadrightarrow^{*} N'$ *and* $M' \; \mathcal{R} \; N'$*.*

Here, we require reduction closure of both $\twoheadrightarrow$ and $\twoheadrightarrow_a$, for any a. This is a *crucial design decision* in `CaIT` (see Ex. 2 and Sec. 2.3 for details).

In order to model *sensor updates* made by the physical environment on a sensor s, in a given location h, we define the operator $[s@h \mapsto v]$ on networks.

Definition 5. *Given a location* h*, a sensor* s*, and a value* v*, we define:*

$$
\begin{aligned}
n[\mathcal{I} \bowtie P]^{\mu}_{h}[s@h \mapsto v] &\stackrel{\text{def}}{=} n[\mathcal{I}[s \mapsto v] \bowtie P]^{\mu}_{h}, \text{ if } \mathcal{I}(s) \text{ defined} \\
n[\mathcal{I} \bowtie P]^{\mu}_{k}[s@h \mapsto v] &\stackrel{\text{def}}{=} n[\mathcal{I} \bowtie P]^{\mu}_{k}, \text{ if } \mathcal{I}(s) \text{ undef. or } h \neq k \\
(M \mid N)[s@h \mapsto v] &\stackrel{\text{def}}{=} M[s@h \mapsto v] \;\mid N[s@h \mapsto v] \\
\big((\boldsymbol{\nu}c)M\big)[s@h \mapsto v] &\stackrel{\text{def}}{=} (\boldsymbol{\nu}c)\big(M[s@h \mapsto v]\big) \\
\mathbf{0}[s@h \mapsto v] &\stackrel{\text{def}}{=} \mathbf{0}.
\end{aligned}
$$

Notice that when updating a sensor we use its location, also for node-dependent sensors. This is because when changing a node-dependent sensor (e.g. touching a touchscreen of a smartphone) the environment is in general aware of its position.

Definition 6. *A binary relation* $\mathcal{R}$ *is* contextual *if* $M \; \mathcal{R} \; N$ *implies that*

– *for all networks* O*,* $M \mid O \;\; \mathcal{R} \;\; N \mid O$
– *for all channels* c*,* $(\boldsymbol{\nu}c)M \;\; \mathcal{R} \;\; (\boldsymbol{\nu}c)N$
– *for all* s*,* h*, and* v *in the domain of* s*,* $M[s@h \mapsto v] \; \mathcal{R} \; N[s@h \mapsto v]$*.*

The first two clauses requires closure under *logical contexts* (parallel systems), while the last clause involves *physical contexts*, which can *nondeterministically update* sensor values.

Finally, everything is in place to define our touchstone behavioural equality.

Definition 7. Reduction barbed congruence*,* $\cong$*, is the largest symmetric relation over networks which is reduction closed, barb preserving and contextual.*

Remark 1. Obviously, if $M \cong N$ then M and N will be still equivalent in any setting where sensor updates are governed by specific physical laws.

We recall that the reduction relation $\twoheadrightarrow$ ignores the passage of time, and therefore the reader might suspect that our reduction barbed congruence is impervious to the precise timing of activities. We show that this is not the case.

Example 1. Let $M = n[\emptyset \bowtie \sigma.\lfloor \bar{c}\langle\rangle \rfloor \mathsf{nil}]^{\mathsf{s}}_{h}$ and $N = n[\emptyset \bowtie \lfloor \bar{c}\langle\rangle \rfloor \mathsf{nil}]^{\mathsf{s}}_{h}$, with $\mathrm{rng}(c) = \infty$. Then, $M \twoheadrightarrow_{\sigma} N$. As $\twoheadrightarrow$ does not distinguish instantaneous from timed reductions, one may suspect that $M \cong N$, and that a prompt transmission along channel c is equivalent to the same transmission delayed of one time unit. However, the test $T = \mathit{test}[\mathcal{J} \bowtie \sigma.a!1.\lfloor c().a!0 \rfloor \mathsf{nil}]^{\mathsf{s}}_{l}$, with $\mathcal{J}(a) = 0$, for some actuator a, can distinguish the two networks. In fact, if $M \mid T \twoheadrightarrow\twoheadrightarrow_{a} O = n[\emptyset \bowtie \lfloor \bar{c}\langle\rangle \rfloor \mathsf{nil}]^{\mathsf{s}}_{h} \mid \mathit{test}[\mathcal{J}' \bowtie \lfloor c().a!0 \rfloor \mathsf{nil}]^{\mathsf{s}}_{l}$, with $\mathcal{J}'(a) = 1$, then there is no O' such that $N \mid T \twoheadrightarrow^{*}\twoheadrightarrow_{a}\twoheadrightarrow^{*} O'$ with $O \cong O'$. This is because O can perform a reduction sequence $\twoheadrightarrow\twoheadrightarrow_{a}$ that cannot be matched by any O'.

Behind this example there is the general principle that reduction barbed congruence is sensitive to the passage of time.

Proposition 2. *If $M \cong N$ and $M \twoheadrightarrow_{\sigma} M'$ then there is N' such that $N \twoheadrightarrow^{*}_{\tau}\twoheadrightarrow_{\sigma}\twoheadrightarrow^{*}_{\tau} N'$ and $M' \cong N'$.*

Now, we provide some insights into the design decision of having two different instantaneous reductions $\twoheadrightarrow_{\tau}$ and $\twoheadrightarrow_{a}$.

Example 2. Let $M{=}n[\mathcal{I} \bowtie a!1 \mid a!0.a!1]^{\mu}_{h}$ and $N{=}n[\mathcal{I} \bowtie a!1.a!0.a!1]^{\mu}_{h}$, with $\mathcal{I}(a){=}0$ and undefined otherwise. Then, within one time unit, M may display on the actuator a either the sequence of values 01 or the sequence 0101, while N can only display the sequence 0101. As a consequence, for a physical observer, the behaviours of M and N are clearly different. Now, if $M \twoheadrightarrow_{\tau}\twoheadrightarrow_{a} M' = n[\mathcal{J} \bowtie a!1]^{\mu}_{h}$, with $\mathcal{J}(a) = 1$, the only possible reply of N respecting reduction closure is $N \twoheadrightarrow^{*}\twoheadrightarrow_{a} N' = n[\mathcal{J} \bowtie a!0.a!1]^{\mu}_{h}$. However, it is evident that $M' \ncong N'$ because N' can turn the actuator a to 0 while M' cannot. Thus, $M \ncong N$.

Had we merged $\twoheadrightarrow_{a}$ with $\twoheadrightarrow_{\tau}$ then we would have $M \cong N$ because the capability to observe messages on actuators, given by the barb, would not be enough to observe changes on actuators within one time interval.

2.3 Design choices

`CaIT` is a value-passing process calculus, à la CCS, which can be easily adapted to deal with the transmission of channel names, à la π-calculus [24].

The time model we adopt is known as the fictitious clock approach (see [14]): a global clock is supposed to be updated whenever all nodes agree on this, by globally synchronising on a special timing action σ. Thus, time synchronisation relies on some clock synchronisation protocol for mobile wireless systems [27].

In *cyber-physical systems* [25], sensor changes are usually modelled either using continuous models (differential equations) or through discrete models (difference equations)[4]. However, in this paper we aim at providing a behavioural semantics for IoT applications from the point of the view of the end user. And the end user cannot directly observe changes on the sensors of an IoT application: she can only observe the effects of those changes via actuators and communication channels. Thus, in `CaIT` we do not represent sensor changes via specific models, but we rather abstract on them by supporting *nondeterministic sensor updates* (see Def. 5 and 6). Actually, as said in Rem. 1, behavioural equalities derived in our setting remains valid when adopting any specific model for sensor updates.

In `CaIT` the value associated to sensors and actuators can change more than once within the same time interval. At first sight this choice may appear weird as certain actuators may require some time to turn on. On the other hand, other actuators may have a very quick reaction. A similar argument applies to sensors. In this respect `CaIT` does not enforce a synchronisation of physical events as it happens for logical signals in synchronous languages [5]. In fact, actuator changes are under nodes' control: if an actuator is a slow device then it is under the responsibility of its controller to update the actuator with a proper delay. Similarly, a sensor should be read only when its value makes sense.

Unlike *mobile computations* [6], smart devices do not decide where to move to: an external agent moves them. Furthermore, Def. 1 imposes that location-dependent sensors can only occur in stationary nodes. This allows us a local, rather than a global, representation of those sensors. The representation of mobile location-dependent sensors would have the same technical challenges of *mobile wireless sensor networks* [27].

Finally, we would like to explain our choice of barb. As said in the Introduction there are other possible definitions. For instance, one could observe the capability to transmit along a channel c, by defining $M \downarrow_{\overline{c}@h}$ if $M \equiv (\boldsymbol{\nu}\tilde{g})\big(n[\mathcal{I} \bowtie \lfloor \overline{c}\langle v \rangle.P \rfloor Q \mid R]^{\mu}_{k} \mid N\big)$ with $c \notin \tilde{g}$ and $\mathrm{d}(h,k) \leq \mathrm{rng}(c)$. However, if you consider the system $S = (\boldsymbol{\nu}c)(M \mid m[\mathcal{J} \bowtie \lfloor c(x).a!1 \rfloor \mathsf{nil}]^{\mu}_{h})$, with $\mathcal{J}(a) = 0$, then it is easy to show that $M \downarrow_{\overline{c}@h}$ if and only if $S \twoheadrightarrow\!\!\twoheadrightarrow_{a} S' \downarrow_{a@h!1}$. Thus, the barb on channels can always be reformulated in terms of our barb. The vice versa is not possible. The reader may also wonder whether it is possible to turn the reduction $\twoheadrightarrow_{a}$ into $\twoheadrightarrow_{\tau}$ by introducing some special barb which would be capable to observe actuators changes. For instance, something like $M \downarrow_{a@h!v.w}$ if $M \equiv (\boldsymbol{\nu}\tilde{g})\big(n[\mathcal{I} \bowtie a!w.P \mid Q]^{\mu}_{h} \mid M'\big)$, with $\mathcal{I}(a) = v$ and $v \neq w$. It should be easy to see that this extra barb would not help in distinguishing the terms proposed in Ex. 2. Actually, here there is something deeper that needs to be spelled out. In process calculi, the term β of a barb $\downarrow_{\beta}$ is a concise encoding of a context C_{β} expressible in the calculus and capable to observe the barb $\downarrow_{\beta}$. However, our barb $\downarrow_{a@h!v}$ does not have such a corresponding *physical context* in our language. Said with an example, in `CaIT` we do not represent the "eyes of a person" looking at the values appearing to some display. Technically speaking, in our calculus we don't have terms of the form $a?(x).P$ to read values on the actuator a, simply because such terms would not be part of an IoT system. The lack of this physical

[4] Difference equations relate to differential equations as discrete math relate to continuous math.

Table 2 A smart home in `CaIT`

$$
\begin{array}{rcl}
Sys & \stackrel{\text{def}}{=} & Phone \;\big|\; Home \\
Phone & \stackrel{\text{def}}{=} & n_P[\mathcal{I}_P \bowtie BoilerCtrl \mid LightCtrl]^{\mathtt{m}}_{out} \\
Home & \stackrel{\text{def}}{=} & LM_1 \;\big|\; BM \;\big|\; LM_2 \\
LM_1 & \stackrel{\text{def}}{=} & n_1[\mathcal{I}_1 \bowtie LightMng_1]^{\mathtt{s}}_{loc_1} \\
LM_2 & \stackrel{\text{def}}{=} & n_2[\mathcal{I}_2 \bowtie LightMng_2]^{\mathtt{s}}_{loc_4} \\
BM & \stackrel{\text{def}}{=} & n_B[\mathcal{I}_B \bowtie BoilerMng]^{\mathtt{s}}_{loc_2} \\
BoilerCtrl & \stackrel{\text{def}}{=} & \mathsf{fix}\, X.mode?(z).\lfloor \bar{b}\langle z\rangle.\sigma.X \rfloor X \\
LightCtrl & \stackrel{\text{def}}{=} & \prod_{j=1}^{2} \mathsf{fix}\, X.\lfloor \overline{c_j}\langle\rangle.\sigma.X \rfloor X \\
LightMng_j & \stackrel{\text{def}}{=} & \mathsf{fix}\, X.\lfloor c_j().light_j!\mathsf{on}.\sigma.X \rfloor light_j!\mathsf{off}.X \quad \text{for } j \in \{1,2\} \\
BoilerMng & \stackrel{\text{def}}{=} & \mathsf{fix}\, X.\lfloor b(x).[x = \mathsf{man}]\, boiler!\mathsf{on}.\sigma.Manual; TempCtrl \rfloor TempCtrl \\
Manual & \stackrel{\text{def}}{=} & \mathsf{fix}\, Y.b(y).[y = \mathsf{auto}]X; \sigma.Y \\
TempCtrl & \stackrel{\text{def}}{=} & temp?(t).[t < \Theta]\, boiler!\mathsf{on}.\sigma.X; boiler!\mathsf{off}.\sigma.X
\end{array}
$$

context, together with the persistent nature of actuators' state, explains why our barb $\downarrow_{a@h!v}$ must work together with the reduction relation $\twoheadrightarrow_a$ to provide the desired distinguishing power of $\cong$. Further discussions can be found in [7].

3 Case study: a smart home

In Tab. 2, we model the smart home discussed in the Introduction, and represented in Fig. 1. Our house spans over 4 contiguous physical locations loc_i, for $i = [1..4]$, such that $\mathrm{d}(loc_i, loc_j) = \mid i - j \mid$. The entrance is in loc_1, the patio spans from loc_2 to loc_3 and the lounge is at loc_4. The house can only be accessed via its entrance.

Our system Sys consists of the parallel composition of the smartphone, $Phone$, and the smart home, $Home$. The smartphone is represented as a mobile node, with $\delta = 1$, initially placed outside the house: $out \neq loc_j$, for $j \in [1..4]$. As the phone can only access the house from its entrance, and $\delta = 1$, we have $\mathrm{d}(l, loc_i) \geq i$, for any $l \notin \{loc_1, loc_2, loc_3, loc_4\}$ and $i \in [1..4]$. Its interface $\mathcal{I}_P$ contains only one sensor, called $mode$, representing the touchscreen to control the boiler. This is a node-dependent sensor. The process $BoilerCtrl$ reads sensor $mode$ and forwards its value to the boiler manager in the patio, $BoilerMng$, via the Internet channel b ($\mathrm{rng}(b) = \infty$). The domain of $mode$ is $\{\mathsf{man}, \mathsf{auto}\}$, where man stands for manual and auto for automatic; initially, $\mathcal{I}_P(mode) = \mathsf{auto}$.

In $Phone$ there is a second process, called $LightCtrl$, which allows the smartphone to switch on lights *only when* getting in touch with the light managers installed in the rooms. Here, channels c_1 and c_2 serve to control the lights of entrance and lounge, respectively; these are short-range channels: $\mathrm{rng}(c_1) = \mathrm{rng}(c_2) = 0$.

The smart home $Home$ consists of three stationary nodes: LM_1, BM, and LM_2. The light managers processes $LightMng_1$, $LightMng_2$, are placed in LM_1 and LM_2, respectively. They manage the corresponding lights via the actuators $light_j$, for $j \in \{1, 2\}$. The domain of these actuators is $\{\mathsf{on}, \mathsf{off}\}$; initially, $\mathcal{I}_j(light_j) = \mathsf{off}$, for $j \in \{1, 2\}$.

Table 3 Smart home: a position based light management

$$\begin{aligned}
\overline{Sys} &\stackrel{\text{def}}{=} \overline{Phone} \;\big|\; \overline{Home}\\
\overline{Phone} &\stackrel{\text{def}}{=} n_P[\mathcal{I}_P \bowtie BoilerCtrl \mid \overline{LightCtrl}]^{\mathtt{m}}_{out}\\
\overline{Home} &\stackrel{\text{def}}{=} Home \;\big|\; \overline{CLM}\\
\overline{CLM} &\stackrel{\text{def}}{=} n_C[\emptyset \bowtie \overline{CLightMng}]^{\mathsf{s}}_{loc_3}\\
\overline{LightCtrl} &\stackrel{\text{def}}{=} \mathsf{fix}\, X.@(x).\lfloor \overline{g}\langle x\rangle.\sigma.X \rfloor X\\
\overline{CLightMng} &\stackrel{\text{def}}{=} \mathsf{fix}\, X.\lfloor g(y).[y = loc_1]\lfloor \overline{c_1}\langle\rangle.\sigma.X\rfloor X; [y = loc_4]\lfloor \overline{c_2}\langle\rangle.\sigma.X\rfloor X; \sigma.X \rfloor X
\end{aligned}$$

The boiler manager process $BoilerMng$ is placed in BM (node n_B). Here, the physical interface $\mathcal{I}_B$ contains a sensor named *temp* and an actuator called *boiler*; *temp* is a location-dependent temperature sensor, whose domain is $\mathbb{N}$, and *boiler* is an actuator to display boiler functionality, whose domain is $\{\mathsf{on}, \mathsf{off}\}$. The boiler manager can work either in automatic or in manual mode. In automatic mode, sensor *temp* is periodically checked: if the temperature is under a threshold Θ then the boiler will be switched on, otherwise it will be switched off. Conversely, in manual mode, the boiler is always switched on. Initially, the boiler is in automatic mode, $\mathcal{I}_B(temp) = \Theta$, and $\mathcal{I}_B(boiler) = \mathsf{off}$.

Our system Sys enjoys a number of desirable *run-time properties*. For instance, if the boiler is in manual mode or its temperature is under the threshold Θ then the boiler will get switched on, within one time unit. Conversely, if the boiler is in automatic mode and its temperature is higher than or equal to the threshold Θ, then the boiler will get switched off within one time unit. These three *fairness* properties can be easily proved because our calculus is well-timed. In general, similar properties cannot be expressed in untimed calculi. Finally, our last property states the phone cannot act on the lights of the two rooms at the same time, manifesting a kind of "ubiquity". For the sake of simplicity, in the following proposition we omit location names both in barbs and in sensor updates, writing $\downarrow_{a!v}$ instead of $\downarrow_{a@h!v}$, and $[s \mapsto v]$ instead of $[s@h \mapsto v]$. The system Sys' denotes an arbitrary (stable) derivative of Sys.

Proposition 3 (Run-time properties). *Let* $Sys\ (\twoheadrightarrow_{\mathrm{i}}^{*}\twoheadrightarrow_{\sigma})^{*}\ Sys'$.

- *If* $Sys'[mode{\mapsto}\mathsf{man}] \twoheadrightarrow_{\mathrm{i}}^{*} Sys'' \twoheadrightarrow_{\sigma}$ *then* $Sys'' \downarrow_{boiler!\mathsf{on}}$
- *If* $Sys'[temp \mapsto t] \twoheadrightarrow_{\mathrm{i}}^{*} Sys'' \twoheadrightarrow_{\sigma}$*, with* $t < \Theta$*, then* $Sys'' \downarrow_{boiler!\mathsf{on}}$
- *If* $Sys'[temp \mapsto t] \twoheadrightarrow_{\mathrm{i}}^{*} Sys'' \twoheadrightarrow_{\sigma}$*, with* $t \geq \Theta$*, then* $Sys'' \downarrow_{boiler!\mathsf{off}}$
- *If* $Sys' \twoheadrightarrow_{\mathrm{i}}^{*} Sys'' \downarrow_{light_1!\mathsf{on}}$ *then* $Sys'' \downarrow_{light_2!\mathsf{off}}$*, and vice versa.*

Finally, we propose a variant of our system, where lights functionality depends on the position of the smartphone. Intuitively, the smartphone detects is current GPS position, via the process $@(x).P$, and then sends it to a centralised light manager process, $\overline{CLightMng}$, via an Internet channel g. This process will interact with the local light managers to switch on/off lights, depending on the position of the smartphone. In Table 3, new components have been overlined. Channels c_1 and c_2 have different range now, as they serve to communicate with the centralised light manager: $\mathrm{rng}(c_1) = 2$ and $\mathrm{rng}(c_2) = 1$.

Prop. 3 holds for $\overline{Sys}$ as well. Actually, the two systems are closely related.

Table 4 Intensional semantics for processes

$$\text{(SndP)}\ \frac{-}{\lfloor \overline{c}\langle v\rangle.P\rfloor Q \xrightarrow{\overline{c}v} P} \qquad \text{(RcvP)}\ \frac{-}{\lfloor c(x).P\rfloor Q \xrightarrow{cv} P\{^{v}/_{x}\}}$$

$$\text{(PosP)}\ \frac{-}{@(x).P \xrightarrow{@h} P\{^{h}/_{x}\}} \qquad \text{(Com)}\ \frac{P \xrightarrow{\overline{c}v} P' \quad Q \xrightarrow{cv} Q' \quad \mathrm{rng}(c) = -1}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$$

$$\text{(Sensor)}\ \frac{-}{s?(x).P \xrightarrow{s?v} P\{^{v}/_{x}\}} \qquad \text{(Actuator)}\ \frac{-}{a!v.P \xrightarrow{a!v} P}$$

$$\text{(ParP)}\ \frac{P \xrightarrow{\lambda} P' \quad \lambda \neq \sigma}{P \mid Q \xrightarrow{\lambda} P' \mid Q} \qquad \text{(Fix)}\ \frac{P\{^{\mathsf{fix}\,X.P}/_{X}\} \xrightarrow{\lambda} Q}{\mathsf{fix}\,X.P \xrightarrow{\lambda} Q}$$

$$\text{(TimeNil)}\ \frac{-}{\mathsf{nil} \xrightarrow{\sigma} \mathsf{nil}} \qquad \text{(Delay)}\ \frac{-}{\sigma.P \xrightarrow{\sigma} P}$$

$$\text{(Timeout)}\ \frac{-}{\lfloor \pi.P\rfloor Q \xrightarrow{\sigma} Q} \qquad \text{(TimeParP)}\ \frac{P \xrightarrow{\sigma} P' \quad Q \xrightarrow{\sigma} Q' \quad P \mid Q \not\xrightarrow{\tau}}{P \mid Q \xrightarrow{\sigma} P' \mid Q'}$$

Proposition 4. *For $\delta = 1$, $(\boldsymbol{\nu}\tilde{c})Sys \cong (\boldsymbol{\nu}\tilde{c})(\boldsymbol{\nu}g)\overline{Sys}$.*

The bisimulation proof technique developed in the remainder of the paper will be very useful to prove equalities between systems of such size.

We end this section with a comment. While reading this case study the reader should have noticed that our reduction semantics does not model sensor updates. This is because sensor changes depend on the physical environment, and a reduction semantics models the evolution of a system in isolation. Interactions with the external environment will be treated in our *extensional semantics*.

4 Labelled transition semantics

In this section we provide two labelled semantic models in the SOS style of Plotkin: the *intensional semantics* and the *extensional semantics*.

Intensional semantics Since our syntax distinguishes between networks and processes, we have two different kinds of transitions:

- $P \xrightarrow{\lambda} Q$, with $\lambda \in \{\sigma, \tau, \overline{c}v, cv, @h, s?v, a!v\}$, for *process transitions*
- $M \xrightarrow{\nu} N$, with $\nu \in \{\sigma, \tau, a, \overline{c}v@h, cv@h\}$, for *network transitions*.

In Tab. 4 we report transition rules for processes, very much in the style of [14]. As in CCS, we assume $[b]P;Q = P$ if $[\![b]\!] = \mathsf{true}$, and $[b]P;Q = Q$ if $[\![b]\!] = \mathsf{false}$. Rules (SndP), (RcvP) and (Com) model communications along a channel c. Rule (PosP) is for extracting the physical position of the embedding node. Rules (Sensor) and (Actuator) serve to read sensors, and to write on actuators, respectively. Rules (ParP) and (Fix) are straightforward. The remaining rules allow us the derive the timed action σ. In Rule (Delay) a timed prefix is consumed. Rule (Timeout)

Table 5 Intensional semantics for networks

$$\text{(Pos)}\ \frac{P \xrightarrow{@h} P'}{n[\mathcal{I} \bowtie P]^{\mu}_{h} \xrightarrow{\tau} n[\mathcal{I} \bowtie P']^{\mu}_{h}} \qquad \text{(SensRead)}\ \frac{\mathcal{I}(s) = v \quad P \xrightarrow{s?v} P'}{n[\mathcal{I} \bowtie P]^{\mu}_{h} \xrightarrow{\tau} n[\mathcal{I} \bowtie P']^{\mu}_{h}}$$

$$\text{(ActUnChg)}\ \frac{\mathcal{I}(a) = v \quad P \xrightarrow{a!v} P'}{n[\mathcal{I} \bowtie P]^{\mu}_{h} \xrightarrow{\tau} n[\mathcal{I} \bowtie P']^{\mu}_{h}} \qquad \text{(LocCom)}\ \frac{P \xrightarrow{\tau} P'}{n[\mathcal{I} \bowtie P]^{\mu}_{h} \xrightarrow{\tau} n[\mathcal{I} \bowtie P']^{\mu}_{h}}$$

$$\text{(ActChg)}\ \frac{\mathcal{I}(a) \neq v \quad P \xrightarrow{a!v} P' \quad \mathcal{I}' := \mathcal{I}[a \mapsto v]}{n[\mathcal{I} \bowtie P]^{\mu}_{h} \xrightarrow{a} n[\mathcal{I}' \bowtie P']^{\mu}_{h}}$$

$$\text{(TimeStat)}\ \frac{P \xrightarrow{\sigma} P' \quad n[\mathcal{I} \bowtie P]^{\mathtt{s}}_{h} \not\xrightarrow{\tau}}{n[\mathcal{I} \bowtie P]^{\mathtt{s}}_{h} \xrightarrow{\sigma} n[\mathcal{I} \bowtie P']^{\mathtt{s}}_{h}} \qquad \text{(TimeMob)}\ \frac{P \xrightarrow{\sigma} P' \quad n[\mathcal{I} \bowtie P]^{\mathtt{m}}_{h} \not\xrightarrow{\tau} \quad \mathrm{d}(h,k) \leq \delta}{n[\mathcal{I} \bowtie P]^{\mathtt{m}}_{h} \xrightarrow{\sigma} n[\mathcal{I} \bowtie P']^{\mathtt{m}}_{k}}$$

$$\text{(Snd)}\ \frac{P \xrightarrow{\bar{c}v} P' \quad \mathrm{rng}(c) \geq 0}{n[\mathcal{I} \bowtie P]^{\mu}_{h} \xrightarrow{\bar{c}v@h} n[\mathcal{I} \bowtie P']^{\mu}_{h}} \qquad \text{(Rcv)}\ \frac{P \xrightarrow{cv} P' \quad \mathrm{rng}(c) \geq 0}{n[\mathcal{I} \bowtie P]^{\mu}_{h} \xrightarrow{cv@h} n[\mathcal{I} \bowtie P']^{\mu}_{h}}$$

$$\text{(GlbCom)}\ \frac{M \xrightarrow{\bar{c}v@k} M' \quad N \xrightarrow{cv@h} N' \quad \mathrm{d}(h,k) \leq \mathrm{rng}(c)}{M \mid N \xrightarrow{\tau} M' \mid N'}$$

$$\text{(ParN)}\ \frac{M \xrightarrow{\nu} M' \quad \nu \neq \sigma}{M \mid N \xrightarrow{\nu} M' \mid N} \qquad \text{(TimePar)}\ \frac{M \xrightarrow{\sigma} M' \quad N \xrightarrow{\sigma} N' \quad M \mid N \not\xrightarrow{\tau}}{M \mid N \xrightarrow{\sigma} M' \mid N'}$$

$$\text{(TimeZero)}\ \frac{-}{\mathbf{0} \xrightarrow{\sigma} \mathbf{0}} \qquad \text{(Res)}\ \frac{M \xrightarrow{\nu} N \quad \nu \notin \{\bar{c}v@h, cv@h\}}{(\boldsymbol{\nu}c)M \xrightarrow{\nu} (\boldsymbol{\nu}c)N}$$

models timeouts when channel communications are not possible in the current time interval. Rule (TimeParP) is for time synchronisation of parallel processes. The symmetric counterparts of Rules (ParP) and (Com) are omitted.

In Tab. 5 we report the rules for networks. Rule (Pos) extracts the position of a node. Rule (SensRead) models the reading of a sensor of the enclosing node. Rules (ActUnChg) and (ActChg) describes the writing of a value v on an actuator a of the node, distinguishing whether the value of the actuator is changed or not. Rule (LocCom) models intra-node communications. Rule (TimeStat) models the passage of time for a stationary node. Rule (TimeMob) models both time passing and node mobility at the end of a time interval. Rules (Snd) and (Rcv) model transmission and reception along an global channel. Rule (GlbCom) models inter-node communications. The remaining rules are straightforward. The symmetric counterparts of Rules (ParN) and (GlobCom) are omitted.

The reduction semantics and the labelled intensional semantics coincide.

Theorem 1 (Harmony theorem). *Let* $\omega \in \{\tau, a, \sigma\}$*:*

- $M \xrightarrow{\omega} M'$ *implies* $M \twoheadrightarrow_{\omega} M'$
- $M \twoheadrightarrow_{\omega} M'$ *implies* $M \xrightarrow{\omega} M''$*, for some* M'' *such that* $M' \equiv M''$*.*

Extensional semantics Here we redesign our LTS to focus on the interactions of our systems with the external environment. As the environment has a *logical*

Table 6 Extensional semantics: additional rules

(SndObs)	$\dfrac{M \xrightarrow{\overline{c}v@h} M' \quad \mathrm{d}(h,k) \leq \mathrm{rng}(c)}{M \xrightarrow{\overline{c}v \triangleright k} M'}$	(RcvObs)	$\dfrac{M \xrightarrow{cv@h} M' \quad \mathrm{d}(k,h) \leq \mathrm{rng}(c)}{M \xrightarrow{cv \triangleright k} M'}$
(SensEnv)	$\dfrac{v \text{ in the domain of } s}{M \xrightarrow{s@h?v} M[s@h \mapsto v]}$	(ActEnv)	$\dfrac{M \downarrow_{a@h!v}}{M \xrightarrow{a@h!v} M}$

part (the parallel nodes) and a *physical part* (the physical world) our extensional semantics distinguishes two different kinds of transitions:

- $M \xrightarrow{\alpha} N$, *logical transitions*, for $\alpha \in \{\tau, \sigma, a, \overline{c}v \triangleright k, cv \triangleright k\}$, to denote the interaction with the *logical environment*; here, actuator changes, τ- and σ-actions are inherited from the intensional semantics, so we don't provide inference rules for them;
- $M \xrightarrow{\alpha} N$, *physical transitions*, for $\alpha \in \{s@h?v, a@h!v\}$, to denote the interaction with the *physical world*.

In Tab. 6 the extensional actions deriving from rules (SndObs) and (RcvObs) mention the location k of the logical environment which can *observe* the communication occurring at channel c. Rules (SensEnv) and (ActEnv) model the interaction of a system M with the physical environment. The environment can *nondeterministically update* the current value of (location-dependent or node-dependent) sensors, and can read the information exposed on actuators.

Note that our LTSs are *image finite*. They are also *finitely branching*, and hence *mechanisable*, under the obvious assumption of finiteness of all domains of admissible values, and the set of physical locations.

5 Full abstraction

Based on our extensional semantics, we are ready to define a notion of bisimilarity. We adopt a standard notation for weak transitions. We denote with $\Longrightarrow$ the reflexive and transitive closure of τ-actions, namely $(\xrightarrow{\tau})^*$, whereas $\xRightarrow{\alpha}$ means $\Longrightarrow \xrightarrow{\alpha} \Longrightarrow$, and finally $\xRightarrow{\hat{\alpha}}$ denotes $\Longrightarrow$ if $\alpha = \tau$ and $\xRightarrow{\alpha}$ otherwise.

Definition 8 (Bisimulation). *A binary symmetric relation $\mathcal{R}$ over networks is a* bisimulation *if $M \mathrel{\mathcal{R}} N$ and $M \xrightarrow{\alpha} M'$ imply there exists N' such that $N \xRightarrow{\hat{\alpha}} N'$ and $M' \mathrel{\mathcal{R}} N'$. We say that M and N are* bisimilar*, written $M \approx N$, if $M \mathrel{\mathcal{R}} N$ for some bisimulation $\mathcal{R}$.*

Later on, we will take into account the number of τ-actions performed by a process. The *expansion* relation [1], written $\lesssim$, is a well-known asymmetric variant of $\approx$ such that $P \lesssim Q$ holds if $P \approx Q$ and Q has at least as many τ-moves as P.

A main result is that our bisimilarity is a *congruence*.

Theorem 2 (Congruence theorem). *The relation $\approx$ is contextual.*

The proof that $\approx$ is preserved by the sensor update operator $[s@h \mapsto v]$ is non-standard and *technically challenging*. It required a well-founded induction.

Thm. 2 is crucial to prove that our bisimilarity is sound with respect to reduction barbed congruence. Actually, our bisimilarity is both sound and complete.

Theorem 3 (Full abstraction). *$M \approx N$ iff $M \cong N$.*

Soundness follows from Thm. 1, Thm. 2, and the capability of extensional actions to capture barbs. As to completeness, for any extensional action α we exhibit an observing context C_α.

Remark 2. A consequence of Thm. 3 and Rem. 1 is that our bisimulation proof-technique remains sound in a setting with more restricted contexts, where nondeterministic sensor updates are replaced by some specific model for sensors.

As testbed for our notion of bisimilarity, we prove a number of algebraic laws on well-formed networks.

Theorem 4 (Some algebraic laws).

1. $n[\mathcal{I} \bowtie a!v.P \mid R]^{\mu}_{h} \gtrsim n[\mathcal{I} \bowtie P \mid R]^{\mu}_{h}$, *if* $\mathcal{I}(a) = v$ *and* a *does not occur in* R
2. $n[\mathcal{I} \bowtie @(x).P \mid R]^{\mu}_{h} \gtrsim n[\mathcal{I} \bowtie \{^{h}/_{x}\}P \mid R]^{\mu}_{h}$
3. $n[\mathcal{I} \bowtie \lfloor \bar{c}\langle v\rangle.P \rfloor S \mid \lfloor c(x).Q \rfloor T \mid R]^{\mu}_{h} \gtrsim n[\mathcal{I} \bowtie P \mid Q\{^{v}/_{x}\} \mid R]^{\mu}_{h}$, *$c$ not in R and* $\mathrm{rng}(c) = -1$
4. $(\boldsymbol{\nu} c)(n[\mathcal{I} \bowtie \lfloor \bar{c}\langle v\rangle.P \rfloor S \mid R]^{\mu}_{h} \mid m[\mathcal{J} \bowtie \lfloor c(x).Q \rfloor T \mid U]^{\mu'}_{k})$
   $\gtrsim (\boldsymbol{\nu} c)(n[\mathcal{I} \bowtie P \mid R]^{\mu}_{h} \mid m[\mathcal{J} \bowtie Q\{^{v}/_{x}\} \mid U]^{\mu'}_{k})$, *if* $\mathrm{rng}(c) = \infty$ *and* c *does not occur in* R *and* U
5. $n[\mathcal{I} \bowtie P]^{\mu}_{h} \gtrsim n[\mathcal{I} \bowtie \mathsf{nil}]^{\mu}_{h}$, *if subterms* $\lfloor \pi.P_1 \rfloor P_2$ *or* $a!v.P_1$ *do not occur in* P
6. $n[\mathcal{I} \bowtie \mathsf{nil}]^{\mu}_{h} \approx \mathbf{0}$, *if* $\mathcal{I}(a)$ *is undefined for any actuator* a
7. $n[\emptyset \bowtie P]^{\mathtt{m}}_{h} \approx m[\emptyset \bowtie P]^{\mathtt{s}}_{k}$, *if* P *does not contain terms of the form* $@(x).Q$, *and for any channel* c *in* P *either* $\mathrm{rng}(c) = \infty$ *or* $\mathrm{rng}(c) = -1$.

Laws 1-4 are a sort of tau-laws. Laws 5 and 6 models garbage collection of processes and nodes, respectively. Law 7 gives a sufficient condition for node anonymity as well as for non-observable node mobility.

Finally, we show that our labelled bisimilarity can be used to deal with more complicated systems. Let us prove that the two variants of the smart home mentioned in Prop. 4 are actually bisimilar.

Proposition 5. *If $\delta = 1$ then $(\boldsymbol{\nu}\tilde{c})Sys \approx (\boldsymbol{\nu}\tilde{c})(\boldsymbol{\nu} g)\overline{Sys}$.*

Due to the size of the systems involved, the proof of the proposition above is quite challenging. In this respect, the first four laws of Thm. 4 are fundamentals to apply non-trivial up-to expansion proof-techniques [24].

6 Related work

To our knowledge, the IoT-calculus [16] is the first (and only) process calculus for IoT systems. We report here the main differences between `CaIT` and the IoT-calculus. In `CaIT`, we can express desirable time and runtime properties (see Prop. 1 and Prop. 3). The nondeterministic link entailment of the IoT-calculus makes communication simpler than ours; on the other hand it does not allow to enforce that a smart device cannot be in two places at the same time. In `CaIT`, both sensors and actuators are under the control of a single entity, i.e. the process of the node where they were deployed. This was a security issue. `CaIT` has a finer control of inter-node communication: it takes into account both distance among nodes and transmission range of channels. Node mobility in `CaIT` is timed constrained: in one time unit at most a fixed distance δ may be covered.

Finally, Lanese et al. equip the IoT-calculus with an *end-user bisimilarity* which shares the same motivations of our bisimilarity. In the IoT-calculus, end users provide values to sensors and check actuators. Unlike us, they can also observe node mobility but they cannot observe channel communication. End-user bisimilarity is not preserved by parallel composition. Compositionality is recovered by strengthening its discriminating power.

Our calculus takes inspiration from algebraic models for wireless systems [17, 23, 19, 12, 3, 18, 20, 11, 26, 10, 21, 8, 4]. Most of these models adopt broadcast communication, while we consider point-to-point communication, as in [16, 10, 21]. We model network topology as in [17, 19]. Prop. 2 was inspired by [8]. Fully abstract observational theories for calculi of wireless systems appear in [12, 19, 8].

Vigo et al. [28] proposed a calculus for *wireless-based cyber-physical systems* endowed with a theory to model and reasoning on cryptographic primitives, together with explicit notions of communication failure and unwanted communication. However, as pointed out in [29], the calculus does not provide a notion of network topology, local broadcast and behavioural equivalence. It also lacks a clear distinction between physical components (sensor and actuators) and logical ones (processes). Compared to [28], paper [29] introduces a static network topology and enrich the theory with an harmony theorem.

`CaIT` shares some similarities with the *synchronous languages* of the Esterel family [5]. In synchronous languages, computations proceed in phases called instants, which have some similarity with our time intervals. For instance, our timed reduction semantics has some points in common with that of CRL [2].

Finally, `CaIT` is somehow reminiscent of the SCEL language [9]. A framework to model behaviour, knowledge, and data aggregation of Autonomic Systems.

Acknowledgements We thank Ilaria Castellani and Matthew Hennessy for their precious comments, and Valentina Castiglioni for an early proof of the harmony theorem. The anonymous referees provided useful comments.

References

1. Arun-Kumar, S., Hennessy, M.: An efficiency preorder for processes. Acta Informatica 29, 737–760 (1992)
2. Attar, P., Castellani, I.: Fine-Grained and Coarse-Grained Reactive Noninterference. In: Abadi, M., Lluch-Lafuente, A. (eds.) TGC 2013. LNCS, vol. 8358, pp. 159–179. Springer, Heidelberg (2013)
3. Benetti, D., Merro, M., Vigano, L.: Model checking ad hoc network routing protocols: Aran vs. endairA. In: Fiadeiro, J.L., Gnesi, S., Maggiolo-Schettini, M. (eds.) SEFM 2010. pp. 191–202. IEEE Computer Society (2010)
4. Borgström, J., Huang, S., Johansson, M., Raabjerg, P., Victor, B., Pohjola, J., Parrow, J.: Broadcast psi-calculi with an application to wireless protocols. Software and System Modeling 14(1), 201–216 (2015)

5. Boussinot, F., de Simone, R.: The SL synchronous language. IEEE Transactions on Software Engineering 22(4), 256–266 (1996)
6. Cardelli, L., Gordon, A.: Mobile ambients. TCS 240(1), 177–213 (2000)
7. Castiglioni, V., Lanotte, R., Merro, M.: A Semantic Theory of the Internet of Things. CoRR abs/1510.04854 (2015)
8. Cerone, A., Hennessy, M., Merro, M.: Modelling mac-layer communications in wireless systems. Logical Methods in Computer Science 11(1:18) (2015)
9. De Nicola, R., Loreti, M., Pugliese, R., Tiezzi, F.: A Formal Approach to Autonomic Systems Programming: The SCEL language. ACM TAAS 9, 7:1–7:29 (2014)
10. Fehnker, A., van Glabbeek, R., Höfner, P., McIver, A., Portmann, M., Tan, W.: A process algebra for wireless mesh networks. In: Seidl, H. (ed.) ESOP 2012. LNCS, vol. 7211, pp. 295–315. Springer, Heidelberg (2012)
11. Ghassemi, F., Fokkink, W., Movaghar, A.: Verification of Mobile Ad Hoc Networks: an Algebraic Approach. TCS 412(28), 3262–3282 (2011)
12. Godskesen, J.: A Calculus for Mobile Ad Hoc Networks. In: Murphy, A.L., Vitek, J. (eds.) COORDINATION 2007. LNCS, vol. 4467, pp. 132–150. Springer (2007)
13. Gubbi, J., Palaniswami, M.: Internet of things (IoT): A vision, architectural elements, and future directions. Fut. Gen. Comput. Syst. 29(7), 1645–1660 (2013)
14. Hennessy, M., Regan, T.: A process algebra for timed systems. Information and Computation 117(2), 221–239 (1995)
15. Honda, K., Yoshida, N.: On reduction-based process semantics. TCS 151(2), 437–486 (1995)
16. Lanese, I., Bedogni, L., Di Felice, M.: Internet of things: a process calculus approach. In: Shin, S.Y., Maldonado, J.C. (eds.) SAC 2013. pp. 1339–1346. ACM (2013)
17. Lanese, I., Sangiorgi, D.: An Operational Semantics for a Calculus for Wireless Systems. TCS 411, 1928–1948 (2010)
18. Lanotte, R., Merro, M.: Semantic analysis of gossip protocols for wireless sensor networks. In: Katoen, J.P., König, B. (eds.) CONCUR 2011. LNCS, vol. 6901, pp. 156–170. Springer, Heidelberg (2011)
19. Merro, M.: An Observational Theory for Mobile Ad Hoc Networks (full version). Information and Computation 207(2), 194–208 (2009)
20. Merro, M., Ballardin, F., Sibilio, E.: A timed calculus for wireless systems. TCS 412(47), 6585–6611 (2011)
21. Merro, M., Sibilio, E.: A calculus of trustworthy ad hoc networks. Formal Aspects of Computing 25(5), 801–832 (2013)
22. Milner, R.: The polyadic π-calculus: a tutorial. Tech. rep., LFCS (1991)
23. Nanz, S., Hankin, C.: A Framework for Security Analysis of Mobile Wireless Networks. TCS 367(1-2), 203–227 (2006)
24. Sangiorgi, D., Walker, D.: The Pi-Calculus a theory of mobile processes. Cambridge University Press (2001)
25. Schaft, A., Schumacher, H.: An introduction to hybrid dynamical systems, Lecture Notes in Control and Information Science, vol. 251. Springer, Heidelberg (2000)
26. Singh, A., Ramakrishnan, C., Smolka, S.: A process calculus for Mobile Ad Hoc Networks. SCP 75(6), 440–469 (2010)
27. Sundararaman, B., Buy, U., Kshemkalyani, A.D.: Clock synchronization for wireless sensor networks: a survey. Ad Hoc Networks 3(3), 281–323 (2005)
28. Vigo, R., Nielson, F., Nielson, H.: Broadcast, denial-of-service, and secure communication. In: Johnsen, E.B., Petre, L. (eds.) IFM 2013. LNCS, vol. 7940, pp. 412–427. Springer, Heidelberg (2013)
29. Wu, X., Zhu, H.: A calculus for wireless sensor networks from quality perspective. In: HASE 2015. pp. 223–231. IEEE Computer Society (2015)