# Norm-based and Commitment-driven Agentification of the Internet of Things

Khouloud Boukadi[a], Noura Faci[b], Zakaria Maamar[c], Emir Ugljanin[d], Mohamed Sellami[e], Thar Baker[f], Mohammed Al-Khafajiy[f]

[a]*University of Sfax, Sfax, Tunisia*
[b]*University of Lyon, Lyon, France*
[c]*Zayed University, Dubai, UAE*
[d]*State University of Novi Pazar, Novi Pazar, Serbia*
[e]*INT ParisSud, Every, France*
[f]*Liverpool JM University, Liverpool, United Kingdom*

**Abstract**

There are no doubts that the Internet-of-Things (IoT) has conquered the ICT industry to the extent that many governments and organizations are already rolling out many anywhere,anytime online services that IoT sustains. However, like any emerging and disruptive technology, multiple obstacles are slowing down IoT practical adoption including the passive nature and privacy invasion of things. This paper examines how to empower things with necessary capabilities that would make them proactive and responsive. This means things can, for instance reach out to collaborative peers, (un)form dynamic communities when necessary, avoid malicious peers, and be "questioned" for their actions. To achieve such empowerment, this paper presents an approach for agentifying things using norms along with commitments that operationalize these norms. Both norms and commitments are specialized into social (i.e., application independent) and business (i.e., application dependent), respectively. Being proactive, things could violate commitments at run-time, which needs to be detected through monitoring. In this paper, thing agentification is illustrated with a case study about missing children and demonstrated with a testbed that uses different IoT-related technologies such as Eclipse Mosquitto broker and Message Queuing Telemetry Transport protocol. Some experiments conducted upon this testbed are also discussed.

*Keywords:* Agentification, Commitment, Internet of Things, Norm, Violation.

## 1. Introduction

There is a consensus among the Information and Communication Technology ICT community that the Internet of Things (IoT) is helping materialize Mark Weiser's vision about ubiquitous computing that is "*...The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indis-*

*tinguishable from it*" [42]. Today's things (e.g., ambient sensors, smart watches, and RFID tags) are everywhere producing massive amount of data (with reference to Big Data era) about other "things" like vegetable freshness in a transit facility, number of vehicles on a highway, and patients' vitals in an ICU. Building upon these massive data, a plethora of services and products are being or already developed like driverless cars (e.g., Waymo by Google), healthcare applications (e.g., closed-loop insulin delivery and IBM Watson Healthcare application in the city of Bolzano [38]), and green buildings (e.g., Bahrain world trade-center and Munich Genius of Things building [16]). It is predicted that the total economic impact of IoT will reach between $3.9 trillion and $11.1 trillion per year by the year 2025 [13]. It is, also, predicted that 30 billion devices will be wirelessly connected to IoT by 2020[1].

Like any emerging and disruptive technology, IoT expansion and practical adoption are encountering multiple obstacles. We cite diversity and multiplicity of things' development and communication technologies [3], users' reluctance and sometimes rejection because of privacy invasion that things cause [24], lack of killer applications that would demonstrate the necessity of things [21], lack of an IoT-oriented software engineering discipline that would guide thing design and development [46], missing interoperability on IoT [8], and, finally, passive nature of things that mainly supply data (with limited actuating capabilities) [13, 27]. Backing the passive nature of things, Green states, in a 2015 IBM white-paper [15], that IoT needs to be smarter so, that, current things would go beyond sensing and sometimes actuating. Wu et al., also, argue that "*without comprehensive cognitive capability, IoT is just like an awkward stegosaurus: all brawn and no brains*" [43]. Simply put, "*The Internet of Things Depends on the Intelligence of Things*"[2].

In this paper, we empower things with particular capabilities that would make them proactive. These capabilities correspond to norms and commitments that would regulate things' operations when they need, for instance to reach out to peers that expose collaborative attitude, to (un)form dynamic communities when necessary, and

[1] `tinyurl.com/yadq3u2d`.

[2] `www.mouser.com/pdfdocs/Technologies-and-Applications-for-the-IoT.pdf`.

to avoid peers that expose malicious attitude. In fact, things will be accountable for their actions. While we already see some positive and encouraging signs of thing empowerment through initiatives like intelligent things [18], wisdom Web of things [9], semantic things [17], Internet of social things [4], Internet of agents [33], agents of things [27, 34], agentified use of IoT [19], process of things [22], and organizational structures for IoT [35], we discuss, here, our work on agentifying[3] (with reference to software agents [23]) things. This agentification is done from 2 perspectives: conceptual using norms to guide things (what can be done) and operational using commitments to allow things to act in compliance with these norms. No-compliance leads to sanctions, which should affect things' credibility and reputation, for example. According to Chesani et al., commitments have a dual-role [10]. At design-time, they represent the consequences of an entity actions on a system. And at run-time, they provide a reference model for monitoring if this entity is behaving as prescribed.

In a previous work [44], we defined an ecosystem of agentified things in which norms and commitments regulate the operations of these things and then, specialized these norms and commitments into social (i.e, user-application independent) and business (i.e., user-application dependent). In this paper, we (*i*) stress out the benefits of agentifying IoT, (*ii*) discuss how things engage in collaborative sessions, (*iii*) identify potential social relations between things, (*iv*) ensure the compliance of things with norms to avoid violations, and finally, (*v*) demonstrate thing agentification through a testbed incorporating different IoT-related technologies like Eclipse Mosquitto broker and Message Queuing Telemetry Transport (MQTT) protocol. Some experiments conducted over the testbed are also presented in the paper.

The rest of this paper is organized as follows. Section 2 discusses the current IoT limitations, norm and commitment use in IoT, and then a case study. Section 3 describes our approach for thing agentification based on norms and commitments. Section 4 presents the approach implementation before drawing some conclusions and discussing some future work in Section 5.

[3]The appropriateness of software agents' characteristics like autonomy, social ability, and mobility, for IoT applications is discussed in [12].

## 2. Background

This section discusses IoT in terms of limitations, related works, and adoption of norms and commitments. It, also, presents a case study that will be used throughout the paper to illustrate thing agentification based on norms and commitments.

### *2.1. IoT limitations*

Pico-Valencia and Holgado-Terriza shed the light on some limitations of IoT that result from both the no-reasoning over things' surroundings and the no-handling of social aspects [33]. These limitations prevent things from being proactive and, thus, confine them into a role of data supplier, only. Acting on behalf of things, software agents could address these limitations. As a result, the Internet of Things will transition to the Internet of Agents. To ensure this transition, Pico-Valencia and Holgado-Terriza present a semantic-contract model that is built upon an OWL-based ontology to describe agents in the context of the Internet of Agents. The description targets the agent's profile, object, model, service, social connection, and context.

Like Pico-Valencia and Holgado-Terriza, Mzahm et al. raise the same limitations that result from the challenging task of embedding advanced hardware and software into things [27]. As a solution, Mzahm et al. propose Agents of Things (AoT) through a 6-layer architecture. The layers are business, application, reasoning, middleware, network, and perception. Particularly, the reasoning layer consists of agents that make decisions on behalf of things. Although some (even many) could argue that it is challenging to embed advanced hardware and software into things [27], Taivalsaari and Mikkonen mention that "*hardware advances and the availability of powerful but inexpensive integrated chips will make it possible to embed connectivity and fully edged virtual machines and dynamic language run-times everywhere*" [39]. A result of these advances is that everyday things will become connected and programmable dynamically.

Looking at IoT from a social perspective is discussed in the work of Atzori et al. [5] and [25]. They consider things as intelligent objects and suggest that models designed for studying social networks of humans can be extended to social networks of objects.

Such a network could be built upon specific relations such as parental (similar objects built in the same period by the same manufacturer), co-location (objects in the same venue), co-work (objects participating in the same scenario), ownership (objects having the same user), and social (when objects come into contact sporadically or continuously). Atzori et al. note the paradigm shift from human-object interaction to object-object interaction.

According to Ortiz et al. [30], many challenges and open issues still undermine the blend of social computing with IoT. These include defining a social thing architecture, addressing interoperability of things, considering new business models, discovering things, managing energy consumption of things, handling security, privacy, and trust of things A social thing architecture would consist of actors (smart things and humans), an intelligent system to manage actors' interactions, an interface for actors to engage in interaction, and the Internet as a means for supporting interaction.

The afore-mentioned works offer a concise overview of IoT limitations that are making things passive. Things should no longer remain "silent". Instead, they should seek the necessary support that would make them active and responsive to changes in users' and peers' (cyber-physical) surroundings. Developing social connections between things could help identify this support along with ensuring a fruitful use of these connections through norms, for example.

### *2.2. Norms and commitments in IoT*

There is a consensus in the R&D community that norms help regulate interactions in an open environment like the Web. Norms are largely adopted in many ICT domains like service computing [36], service-level-agreement analysis [31], correct protocol generation from contracts [28], e-business system modeling [45], etc. In this part of the paper, we provide a brief overview of norm and commitment use in the context of IoT. Brief because of the limited number of relevant references, to the best of our knowledge, that we could identify despite the growing interest in IoT, in general, and thing agentification, in particular.

Alkhabbas et al. suggest that responding to an emergent configuration could happen through commitments. These latter support form a dynamic set of things that

would temporarily cooperate to achieve some user goal [2]. This cooperation consists of analyzing a user goal's achievability given capabilities of things located within some specified geographical boundaries, supporting the negotiation with the available things so, that, a commitment-based emergent configuration is formed, and, finally, running and continuously monitoring this configuration to make adaptation if needed. During negotiation, some things are requested to commit towards other peers to perform some actions when some conditions hold. At run-time, things can release peers from commitments initially established. In another work, Dastani et al. propose a formal framework for modeling, analyzing, and comparing different commitments' lifecycles in virtual organizations with focus on business models based on cloud computing and IoT [11]. This framework models, first, a lifecycle as a set of interaction norms (defined as rules) and second, regulation policies as sanctions to apply when commitments are violated. The framework, also, tracks lifecycles through monitoring (to check the conformance of agents' interactions to norms) and enforcement (that regulates these interactions). Last but not least, the framework formally checks dynamic properties of commitments' lifecycles like interference between 2 distinct commitments and redundancy of interaction norms. To wrap-up this discussion, we note the limited use of norms and commitments in IoT, which is confining things into a passive, instead of proactive, role as described earlier.

### *2.3. Case study*

According to the International Centre for Missing & Exploited Children[4], the number of missing children is alarming, for instance, 20,000 in Australia, 45,000 in Canada, and 460,000 in the United States. Resolving missing-child cases is complex due to their sensitivity, unicity, and multiplicity of intervenants such as parents, witnesses, law-enforcement agencies, and social services [7, 29].

From an IoT perspective, surveillance cameras could help answer questions like when and where was the last time the child was seen, with whom was the child the last time (s)he was seen, and what was the child wearing or holding? We advocate for

[4] `globalmissingkids.org/awareness/missing-children-statistics`.

collaborative cameras in order to speed up the case handling and cover wider search areas. To this end, the initially-selected cameras by the law enforcements, for example, would reach out to other cameras based on certain business and social agreements. Business agreements are between municipalities that own the cameras and law enforcement agencies, for example. And, social agreements are between cameras with respect to their respective similar/alternative/joint capabilities (e.g., zoom and storage).

Giving cameras a certain "freedom" to act such as contacting peers could lead to situations that violate agreements. Cameras can exceed the authorized zoom-level and capture people's faces without their approvals. Thus, monitoring the satisfaction of all agreements will permit to detect violations so, that, actions are taken with respect to what happened and who to hold responsible. In this paper, we provide means for setting-up such proactive things by generating norms and commitments based on business and social agreements and defining things' operations using norms and commitments. We, also, develop control mechanisms to ensure the consistency between commitments in a multi-thing ecosystem and monitoring mechanisms to verify their compliance with norms.

## 3. Norm-based agentification of things

This section discusses thing agentification by categorizing norms, identifying potential social relations between things, setting-up an agentification chronology, defining norms and their commitments, and, finally, developing guidelines for cross-commitment consistency and compliance with norms.

### *3.1. How to categorize norms?*

Agentification is to make things proactive and responsive to users' and peers' requests in an autonomous and independent way and, hence, capable of executing operations, as they see fit[5]. Agents of different types and different capabilities (e.g., monitoring, communicating, and reasoning) would be required acting on behalf of things. To

[5]For Pérez Hernández and Reiff-Marganiec, autonomous things are expected to sense, actuate, and reason [32].

ensure that the outcomes of things' operations are beneficial to the whole IoT ecosystem such as satisfying users' requests and rewarding collaborating peers, we adopt norms to identify necessary operations and specialize norms into business and social. The objective of this specialization is to capture the particular business and social aspects of the under-development IoT applications.

First, business norms capture User-*2*-Thing ($U2T$) interactions and are strictly application dependent. The nature of under-development IoT applications (e.g., missing children) limits the choices of defining business norms and commitments. Business norms and commitments in the missing-child application (e.g., screening camera records *requires*[6] approval and approval *should* be secured prior to screening any camera record, respectively) are different from those in a loan application (e.g., maintaining minimum solvency ratio of 1.5 is *required* and solvency-ratio calculation *should* include all incomes and debts, respectively).

Second, social norms capture Thing-*2*-Thing ($T2T$) interactions and are strictly application independent. These norms reflect interactions between things that arise during the implementation of business/social commitments. From a thing perspective, we define the progress of a $T2T$ interaction along 3 stages: before, during, and after:

1. Before-interaction stage: example of social norm is to *invite* things (e.g., cameras) to participate in certain operations (e.g., supporting child search) while a corresponding social commitment is to contact all things without excluding any (i.e., *check* with all cameras in a targeted search area). Another example of social norm is to *require* that things make interactions transparent while a corresponding social commitment is to use appropriate means (e.g., public electronic bulletin-board *should* be used) to guarantee this transparency.
2. During-interaction stage: example of social norm is to *prohibit* things from exchanging sensitive details publicly (e.g., child's family name) while a corresponding social commitment is to anonymize these details (e.g., nickname instead of family name *should* be used). Another example of social norm is to

[6]We use particular verbs, formatted in *italic*, to ensure the consistent definition of norms and commitments.

*require* that things exchange up-to-date data (e.g., child's latest location) while a corresponding social commitment is that data should continually be streamed to all relevant parties (e.g., reserved communication channels *should* be used).

3. After-interaction stage: example of social norm is to *require* that things identify all pending operations (e.g., suspension of broadcasting a child's photo) following the completion of an interaction while a corresponding social commitment is to finalize all these operations (e.g., photo broadcast *should* be resumed after any suspension). Another example of social norm is to *prohibit* things from opening a new communication channel with a third party that was deemed unnecessary for an interaction (e.g., camera's resolution below a threshold) while a corresponding social commitment is to verify the relevancy of this party before (e.g., camera's capabilities are to be *checked*).

### *3.2. How to socialize things?*

In the context of thing agentification, $T2T$ interactions shed the light on a certain form of socialization among things. In line with the multiple thing-socialization works [4, 5, 25, 30, 47], we propose 3 specific social relations between things upon which networks of things are developed[7] (Fig. 1). These relations allow things to look at peers from 3 perspectives: recommendation exemplified with *complementarity* relation, opposition exemplified with *antagonism* relation, and exclusion exemplified with *competition* relation (below $t$ stands for thing).

1. Complementarity relation is the concurrent participation of things in joint operations. A social norm for this relation is that a recommended thing is *required* to fulfill its duties while a corresponding commitment is not to *make* the recommending thing *feel* disappointed. We propose Equation 1 to assess the complementarity level between 2 things where $acceptedRec(t_i, t_j)$ is the number of times that the IoT application engineer accepts $t_i$'s recommendations for $t_j$ and $madeRec(t_i, t_j)$ is the total number of times $t_i$ recommended $t_j$ (including those

[7] Our social relations are in line with Atzori et al.'s 5 relations (though the social dimension in Atzori et al.'s work is not stressed-out).

recommendations that are declined). A highest value of $w_{compl(t_i,t_j)}$ reflects the positive opinion of $t_i$ about $t_j$.

$$w_{compl(t_i,t_j)} = \frac{acceptedRec(t_i, t_j)}{madeRec(t_i, t_j)} \tag{1}$$

2. Antagonism relation is the "sensitivity" (or "friction") among things when both participate in joint operations. A social norm for this relation is that opponent peers are *required* to fairly participate in joint operations while a corresponding commitment is not to *make* the other peer *feel* frustrated. We propose Equation 2 to assess the antagonism level between 2 things where $jointOp(t_i, t_j)$ is the number of times $t_i$ and $t_j$ participated together in joint operations and $inJointOp(t_i \mid \neg t_j)$ is the total number of times $t_i$ took part in some operations without $t_j$ and *vice versa*. A highest value of $w_{antag(t_i,t_j)}$ reflects a strong co-presence between $t_i$ and $t_j$.

$$w_{antag(t_i,t_j)} = \frac{jointOp(t_i, t_j)}{inJointOp(t_i \mid \neg t_j) + inJointOp(t_j \mid \neg t_i)} \tag{2}$$

3. Competition relation is the exclusion among things as only one will participate in an operation. A social norm for this relation is that an excluded thing is *required* not to undermine the selected thing's operation while a corresponding commitment is that the excluded thing is expected to *make* all necessary details available to the selected thing, should this one request them. We propose Equation 3 to assess the competition level between 2 things where $selectedOp(t_i)$ is the total number of times $t_i$ was selected over $\mathsf{t}_j$ to take part in some operations and $possibleOp(t_i, t_j)$ is the number of times $t_i$ and $t_j$ were both considered for selection to participate in common operations. A highest value of $w_{compe(t_i,t_j)}$ reflects the appropriateness of $t_i$ over $t_j$ for satisfying operations' requirements.

$$w_{compe(t_i,t_j)} = \frac{selectedOp(t_i)}{possibleOp(t_i, t_j)} \tag{3}$$

### *3.3. How to agentify things?*

Norms have an impact on shaping thing agentification. Indeed, they provide the necessary guidelines to things about how to act when processing requests of users

($U2T$ interactions) and/or involving peers in this processing ($T2T$ interactions). Due to the diversity of users' requests, agentification defines norms at the conceptual level and maps norms onto commitments at the operational level. Simply put, commitments implement norms.

Our proposed chronology of thing agentification is depicted in Fig. 1. Agent-Thing correspondence could take different forms including one-to-one (one agent per camera), one-to-many (one agent for all cameras related to the same missing child case), and many-to-many (many agents for all cameras in the same neighborhood). It all starts when some application engineers define business and social norms as per the formalism of Section 3.4. The engineers heavily rely on the under-development IoT application's characteristics (e.g., missing children) to define business norms. We recall that social norms are application independent (Section 3.1), although, engineers can rely on social relations between things to define additional social norms. When a user initiates an interaction with a thing, the social/business norms associated with this interaction are loaded from the respective repositories. Then, the norms are mapped onto social/business commitments in preparation for their execution by things. During execution, commitment violations could happen and, thus, need to be detected (Section 3.7).

### *3.4. How to formalize norms?*

To formalize $\mathcal{B}$usiness $\mathcal{N}$orms ($\mathcal{N}_{\mathcal{B}}$) and $\mathcal{S}$ocial $\mathcal{N}$orms ($\mathcal{N}_{\mathcal{S}}$), we adopt Vázquez-Salceda et al.'s formal language whose core components are 3 *deontologic concepts* ($OBLIGED$ (ought to), $PERMITTED$ (may), and $FORBIDDEN$ (ought not to)) and 2 *temporal operators* ($BEFORE$ and $AFTER$) [41]. In this language, norms can (*i*) refer to an abstract state/action that an agent role should take over/perform, (*ii*) be conditional, (*iii*) include a deadline, and/or (*iv*) be norms concerning other norms (i.e., meta-norms like obligation to enforce norms).

**Definition 1** (*a*). norm $\mathcal{N}$ is expressed as: $C(a, r)$ where:

- $C \in \{OBLIGED, PERMITTED, FORBIDDEN\}$,
- $a$ represents the entity that is expected to bind the norm (i.e., thing's owner),

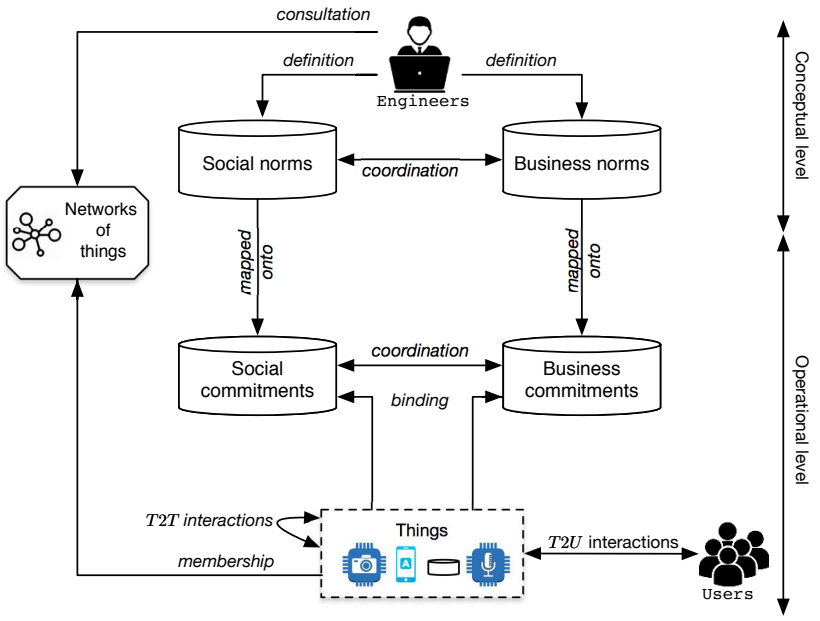



Figure 1: Components supporting thing agentification

- and, $r$ represents ($i$) the abstract state that $a$ should take ON or ($ii$) the abstract action that $a$ should perform[8]. In case of an abstract state, $a$ should also report its state to the ecosystem's authority.

As per Section 3.3, $U2T$ and $T2T$ interactions exist and, hence, will be regulated through business and social norms. We begin with social norms and define them along **B**efore (***B***), **D**uring (***D***), and **A**fter (***A***) stages:

- **Before stage (*B*)**. Some social norms include, but are not limited to:
  - $\boldsymbol{B}$-$\mathcal{N}_{S_1}$: Inviting peers (*peerInvitation*) to participate in a certain operation should indicate its purpose[9] ($p$) (e.g., operation name and requirements) and deadline ($d$) of receiving responses from peers to this invitation. This norm is represented as $OBLIGED(host, \underline{do}\ peerInvitation(p, d, pe))$ where

[8] In case of an abstract action, $r$ is preceded by do.

[9] For the sake of simplicity, purpose is unique and thus, used for identification purposes.

*host* is the thing issuing the invitation, *peerInvitation* is the abstract action, and *pe* are peers acting as potential respondents. $\boldsymbol{B}$-$\mathcal{N}_{S_1}$ can be applied when a thing is looking for collaborating peers.

- $\boldsymbol{B}$-$\mathcal{N}_{S_2}$: A thing ($t$) in the ecosystem reports its status to the ecosystem's authority (*auth*), represented as $PERMITTED(t, currentStatus(busy \otimes available, auth))$ where *currentStatus* is the thing's abstract state (e.g., *busy*).
- $\boldsymbol{B}$-$\mathcal{N}_{S_3}$. A recommended thing (*sollicited*) should report its decision of participating in some joint operation ($op$) to the ecosystem's authority (*auth*), represented as $OBLIGED(sollicited, decision(op, yes \otimes no, auth))$ where *decision* is the thing's abstract state (e.g., *yes*).

- **During stage ($\boldsymbol{D}$)**. Some social norms include, but are not limited to:
  - $\boldsymbol{D}$-$\mathcal{N}_{S_4}$. Any classified/sensitive detail ($d$) should not be disclosed (*detailDisclosure*), represented as $FORBIDDEN(detailDisclosure(d, yes))$ where *detailDisclosure* is the abstract state (e.g., *yes* for not-disclosed).
  - $\boldsymbol{D}$-$\mathcal{N}_{S_5}$. Any detail ($d$) exchanged during interaction (*detailFrehsnessLevel*) should have a certain freshness level ($fl$), represented as $OBLIGED(detailFrehsnessLevel(d, fl))$ where *detailFrehsnessLevel* is the abstract state.

- **After stage ($\boldsymbol{A}$)**. Some social norms include, but are not limited to:
  - $\boldsymbol{A}$-$\mathcal{N}_{S_6}$. An audit of a thing ($t$) participating in an operation (*opAudit*) should indicate the status of all operations ($op$), represented as $PERMITTED(auditor, opAudit(t, op(complete \otimes pending)))$ where *auditor* is the thing carrying the audit, $t$ is the thing being audited, and *opAudit* is the abstract state (e.g., *complete*).

After presenting some social norms, we now define some business norms using the missing child case-study:

- **Before stage ($\boldsymbol{B}$)**. Some business norms include, but are not limited to
  - $\boldsymbol{B}$-$\mathcal{N}_{\mathcal{B}_1}$. Releasing a tape (*tapeRelease*) to local law enforcement agencies (*lea*) should be backed by some reasons ($r$) beforehand from these

agencies, represented as $OBLIGED(t, \underline{do}\ tapeRelease(ta, lea, r))$ where $t$ is the thing managing the tape *ta* and *tapeRelease* is the abstract action.

- **During stage (*D*)**. Some business norms include, but are not limited to
  - ***D***-$\mathcal{N}_{\mathcal{B}_2}$. Any photo ($p$) about a search area (*areaZoom*) should have a certain granularity level ($gl$), represented as $OBLIGED(areaZoom(p, a, gl))$ where *areaZoom* is the abstract state and $a$ is the area being zoomed.

*3.5. How to map norms onto commitments?*

To structure commitments with respect to social and business norms, we adopt Fornara and Colombetti's formalism [14]: **Stage**-$C_j^{\mathcal{S}_i}|C_j^{\mathcal{B}_i}(debtor,\ creditor,\ content[|condition])$ where $C_j^{\mathcal{S}_i}|C_j^{\mathcal{B}_i}$ is a social|business commitment associated with a social|business norm $\mathcal{N}_{\mathcal{S}_i}|\mathcal{N}_{\mathcal{B}_i}$ at a certain stage of the interaction (i.e., ***B***, ***D***, and ***A***), *debtor* is the thing that makes the commitment towards either a peer or user known as *creditor*, *content* is some action(s) that *debtor* will execute, [ ] means optional, and *condition* validity authorizes *debtor* to execute the action(s). Note that a norm can be mapped onto at least one commitment. For the sake of illustration, we define 4 social commitments (***B***-$C_1^{\mathcal{S}_1}$, ***B***-$C_1^{\mathcal{S}_2}$, ***D***-$C_1^{\mathcal{S}_4}$, and ***A***-$C_1^{\mathcal{S}_6}$) and 1 business commitment (***B***-$C_1^{\mathcal{B}_1}$).

1. ***B***-$C_1^{\mathcal{S}_1}(host,\ auth,\ contactAll(p, d, pe))$ is a *host*'s social commitment towards the ecosystem's authority (*auth*) to contact all peers (*pe*), *contactAll* is the concrete action associated with the abstract action *peerInvitation*, $p$ is the purpose referring to an operation that *host* would like to see some peers participate in, and $d$ is the deadline of receiving responses from *pe*.
2. ***B***-$C_1^{\mathcal{S}_2}(host,\ auth,\ report(busy \otimes available))$ is a *host*'s social commitment towards the ecosystem's authority (*auth*) to report its status, *report* is the concrete action associated with the abstract state *currentStatus*, and $busy \otimes available$ is the possible status.
3. ***D***-$C_1^{\mathcal{S}_4}(host,\ auth,\ hide(d, yes))$ is a *host*'s social commitment towards the ecosystem's authority (*auth*) to not disclose any classified detail ($d$), *hide* is the concrete action associated with the abstract state *detailDisclosure*, and *yes* confirms the no-disclosure.

4. $\boldsymbol{A}$-$C_1^{\mathcal{S}_6}(auditor,\ auth,\ reportAll(t, op(pending \otimes complete)))$ is an *auditor*'s social commitment towards the ecosystem's authority (*auth*) to report the status of all operations (*op*) per thing (*t*), *reportAll* is the concrete action associated with the abstract state *opAudit*, and $pending \otimes complete$ is the possible status.
5. $\boldsymbol{B}$-$C_1^{\mathcal{B}_1}(host,\ auth,\ handOver(ta, lea, r)|valid(r))$ is a *host*'s business commitment towards the ecosystem's authority (*auth*) to hand tapes (*ta*) over to the law enforcement agencies (*lea*), *handOver* is the concrete action associated with the abstract action *releaseTape*, and *valid* is a function that *auth* uses to check at run-time whether reasons (*r*) back the release approval to *lea*.

### *3.6. How to check the consistency of commitments?*

Because a norm can be mapped onto many commitments (Section 3.5) and that commitments of different norms are defined independently from each other, conflicts of types intra-norm and inter-norm could arise. Reasons of conflicts could be, for instance inaccurate execution chronology between commitments' actions linked to the same norm and, also, incompatible conditions between commitments linked to separate norms. If not addressed, conflicts could lead to an unstable ecosystem of things (e.g., deadlock) and, even, to jeopardize its existence (e.g., dismantlement). To achieve commitment consistency, we proceed as follows using some examples related to the missing child case-study.

- Intra-norm conflict: A camera commits to the ecosystem's authority to contact some peers (i.e., certain cameras) so, that, they participate in some joint operation ($\boldsymbol{B}$-$C_1^{\mathcal{S}_1}$). But, the camera adds a peer to this operation without waiting for its final response. For the sake of time, the requesting camera assumed a positive response based on past cases (i.e., the peer should positively reply). The reason of conflict is inappropriate execution chronology between *add peer* ($add(pe)$) and *wait for peer's response* ($wait(pe, resp)$) actions. The commitment having $add(pe)$ as an action ($\boldsymbol{B}$-$C_2^{\mathcal{S}_1}$) should be conditioned by the response reported in the commitment having $wait(pe, resp)$ as an action ($\boldsymbol{B}$-$C_3^{\mathcal{S}_1}$). As a solution, we revisit this free-condition commitment's definition so, that, a new condition ($valid(resp)$) is attached to the commitment.

- Inter-norm conflict: A camera commits to the ecosystem's authority to report both its decision of participating in some operation ($\boldsymbol{B}$-$C_1^{S_3}$) and the status of all its operations ($\boldsymbol{A}$-$C_1^{S_7}$). However its rejection decision of participation does not match the status (e.g., *pending*) of certain operations. The reason of conflict is incompatibility between the respective parameter values (i.e., *no* for decision and *pending* for status) of *report* and *reportAll* actions. The commitment having $reportAll(op(pe, pending \otimes complete))$ as an action should be conditioned by the acceptance decision reported in the commitment having $report(op(yes \otimes no))$ as an action. As a solution, we revisit the commitment's definition so, that, $reportAll(op(pe, pending \otimes complete))$ is changed to $reportAll(pe(yes), op(pending \otimes complete))$.

To achieve commitment consistency, we develop guidelines that consider execution dependencies and data exchanges between commitments' constituents (i.e., actions) and between commitments as well.

- Guidelines for intra-norm conflicts. Commitments that belong to the same norm could start and/or finish the execution of their actions depending on events like same-start-with-another-action and same-finish-with-another-action. We specialize action dependencies between two actions ($a_1$,$a_2$) into 4 types: start-to-start (i.e., $a_1$ must start before $a_2$ can start), start-to-finish, finish-to-start, and finish-to-finish. For instance, there exists a finish-to-start dependency between *wait* and *add* actions reported in $\boldsymbol{B}$-$C_2^{S_1}$ and $\boldsymbol{B}$-$C_3^{S_1}$, respectively. These dependencies will be enforced at run-time through functions (e.g., *valid*()), which should guarantee the consistent progress of actions at run-time.

- Guidelines for inter-norm conflicts. Commitments (*Com*) that are linked to different norms could be synchronized by ensuring proper data exchange at run-time. For each commitment, we define the following (null or non-null) parameters: (input, output). We check consistency if $output(Com_i) \cap input(Com_{(i+1)}) \neq \phi$. For instance, $output(\boldsymbol{B}\text{-}C_1^{S_3}) = \{op(yes \otimes no)\}$ and $input(\boldsymbol{A}\text{-}C_1^{S_7}) = \{op(pe, pending \otimes complete)\}$.

*3.7. How do things comply with norms?*

In our thing-agentification approach, we action (business/social) norms through (business/social) commitments. To ensure full compliance with norms at run-time, monitoring commitments is a must and should permit to detect satisfaction or violation of these commitments [11, 37]. In fact, monitoring checks if a commitment's *debtor* has either executed[10] (i.e., satisfaction) or not (i.e., violation[11]) the actions included in the commitment's content.

Because of the social aspects that we inject into our agentification approach, we consider that the exclusivity of either satisfaction or violation does not permit to capture social aspects upon which ongoing and complex interactions between entities, such as *debtor* and *creditor*, are built (Fig. 2). Indeed, these interactions require more than "yes" and "no" that characterize business interactions (i.e., either approval or rejection) that are usually governed by strict contracts [20]. To address the exclusivity limitation, we propose a $3^{rd}$ option that would "sit" between a commitment's (complete) satisfaction and (complete) violation, which is partial satisfaction/violation. The revised semantics of complete *versus* partial is discussed below using the missing child case-study and depicted with Fig. 3:

- Complete satisfaction of commitment: the *debtor* has successfully executed all actions (after satisfying all conditions, if any) and the outcomes of this execution meet the *creditor*'s expectations; e.g., the tapes handed over to the law enforcement agencies are relevant since they record the areas of interest from 9am to 11am.

- Complete violation of commitment: either the *debtor* has not executed any action or has executed actions despite the non-satisfaction of conditions, if any; e.g., either no tapes have been handed over to the law enforcement agencies or the tapes have been handed over despite the no-eligibility of the recipients (e.g., local medias).

[10] For the sake of simplicity, we assume that all executions are successful.

[11] Violation could lead to investigation/argumentation but this is outside this work's scope.

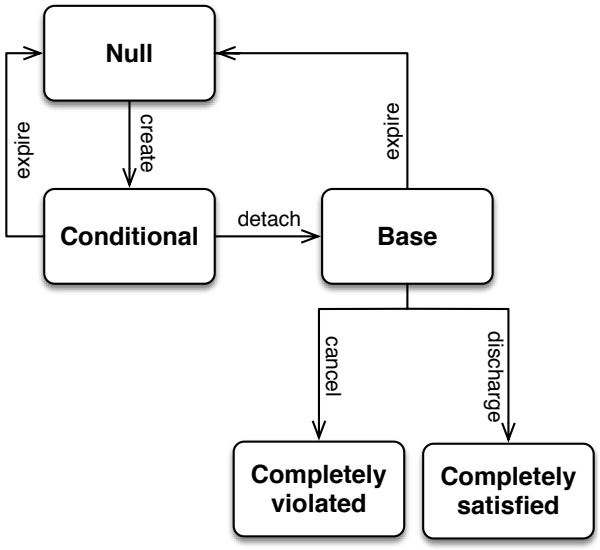



Figure 2: Business commitment's lifecycle (as per [37])

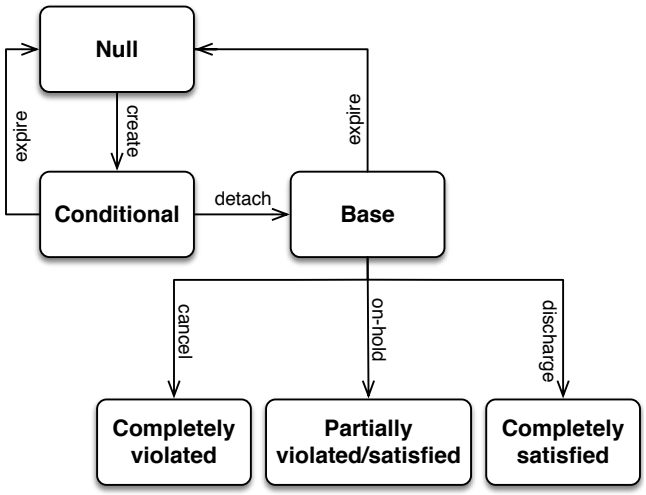



Figure 3: Social commitment's lifecycle

- Partial violation/satisfaction of commitment: the *debtor* has successfully executed all actions (after satisfying all conditions, if any) but the outcomes of this execution do not meet the *creditor*'s expectations; e.g., "fresh" data has been shared with the requestors, but upon receipt, the data has become obsolete.

We, hereafter, illustrate the monitoring of certain business and social commitments presented earlier.

- $\boldsymbol{B}$-$C_1^{\mathcal{B}_1}(host,\ auth,\ handOver(ta, lea, r)|valid(r))$. Monitoring this business commitment means that the ecosystem's authority (*auth*) verifies the reasons ($r$) that led the camera's owner (*host*) to hand over the tapes (*ta*) to the law enforcement agencies (*lea*). If the reasons are valid, the ecosystem's authority raises a complete satisfaction of the commitment. Otherwise, it raises a complete violation.

- $\boldsymbol{D}$-$C_1^{\mathcal{S}_3}(host,\ auth,\ hide(d, yes))$. Monitoring this social commitment means that the ecosystem's authority (*auth*) verifies that the camera's owner (*host*) has hidden every classified detail ($d$) (e.g., child's name). If this is the case, the ecosystem's authority raises a complete satisfaction of the commitment. Otherwise, it raises either a partial violation if $d$ has been hidden but $d$ is not classified or a complete violation if $d$ has not been hidden but $d$ is classified.

## 4. Thing-agentification testbed

This section discusses the testbed for implementing, experimenting, and evaluating our thing-agentification approach. Java, Eclipse Mosquitto broker [1], and Message Queuing Telemetry Transport (MQTT) publish/subscribe protocol [6] are among the technologies implementing the testbed.

### *4.1. Architecture*

Fig. 4 is the architecture of the testbed that comprises a set of Java-based in-house developed components such as definition manager and monitor manager, and uses Mosquitto in support of exchanging MQTT messages (e.g., streaming request) between certain components. Requests that messages convey are expected to comply with the

defined norms and corresponding commitments; otherwise, violations are flagged (Section 4.2). In the testbed, 2 relational databases are developed. The first stores details about cameras (e.g., identifier, communication protocol, capabilities, and physical location), and the second stores details about instantiated norms and commitments used during monitoring.

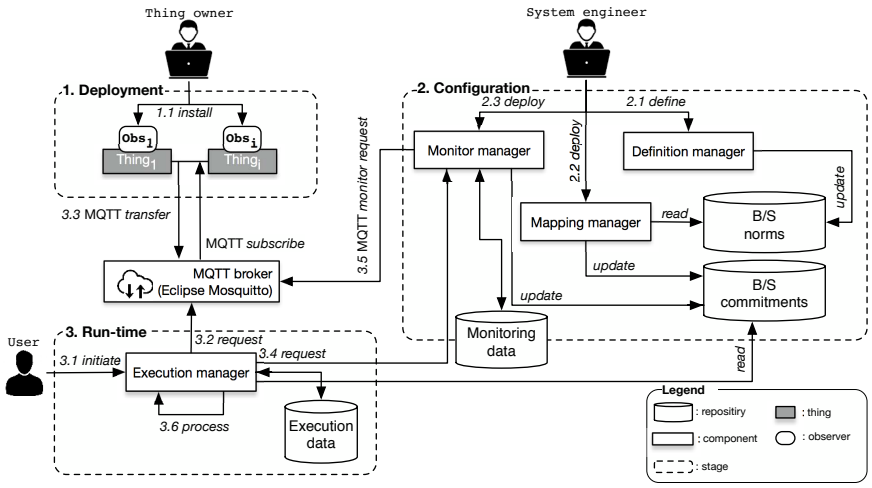



Figure 4: Testbed's architecture for thing agentification

The chronology of setting up and operating the testbed goes through 3 stages.

1. *Deployment* stage is about owners who install and initialize their things (cameras in the following) with necessary details (1.1 in Fig. 4) along with assigning observers to these cameras (one-2-one). Observers report on the actions that cameras execute such as recording scenes, zooming-in/out, and rotating. These actions reflect the progress of $T2U$ and $T2T$ interactions and permit to establish the lifecycles of cameras' commitments (Fig. 2 and Fig. 3).
2. *Configuration* stage targets the definition manager, mapping manager, monitor manager, and B/S norms/commitments repositories. The definition manager assists the testbed engineer define (2.1) the necessary $\mathcal{N}_{B_j}$ and $\mathcal{N}_{S_j}$ and stores them in the B/S norms repository (update operation). The mapping manager, also, assists the engineer deploy the corresponding commitments (2.2) for each norm (read

operation) prior to storing these commitments in the B/S commitments repository (update operation). To track the completion process of things' commitments (so, that, potential violations are detected), the engineer deploys the monitor manager (2.3) whose role becomes evident during the *run-time* stage.

3. *Run-time* stage is about users who have requests like streaming cameras' contents. First, a user initiates the execution manager (3.1), that on behalf of the user, requests cameras to execute specific operations (3.2-3.3). In conjunction with sending users' requests, the execution manager requests from the monitor manager (3.4) to "keep-an-eye" on the cameras' commitments (3.5) based on the observers' reports so, that, the monitor manager updates the B/S commitments according to these commitments' statuses (Fig. 3, update operation). Finally, the execution manager analyzes these statuses (3.6) by consulting the B/S commitments repository (read operation). In the case of any commitment violation (illustrated below), the execution manager informs the user through a dedicated dashboard.

### *4.2. Experiments*

We considered 3 real IP cameras that we connect through an "IP Webcam" Android app installed on 3 Lenovo Yoga tablets. The cameras are placed in different parts of the city of Novi Pazar in Serbia, stream 24/7 live video content, and offer a secure access to authenticated users via an in-house Java-based camera control panel. We use the panel to direct 2 requests to cameras: rotation and streaming. Rotation requests originate from users such as Police and other things such as another camera (when extending a search area). Contrarily, streaming requests originate from users, only, such as Police and social services. In conjunction with the cameras, we deployed the testbed on a 64bit Intel Core 2.50GHz CPU, 6GB RAM laptop.

2 types of violation-related experiments were performed: detection efficiency and detection duration. The experiments cover the following commitments:

- $\boldsymbol{D}\text{-}C_1^{S_4}(host,\ auth,\ hide(d, yes))$ is a *host*'s social commitment towards the ecosystem's authority (*auth*) to not disclose any classified detail (*d*). The detail is about a person's face.

- $\boldsymbol{B}$-$C_1^{\mathcal{S}_2}(host,\ auth,\ report(busy \otimes available))$: A thing ($t$) in the ecosystem reports its status to the ecosystem's authority (*auth*).

- $\boldsymbol{B}$-$C_1^{\mathcal{B}_2}(host; auth; zoom(p; a; gl))$ gl)) is a *host*'s business commitment toward the ecosystem's authority (*auth*) to zoom a search area ($a$) resulting in a photo ($p$) with a certain granularity level ($gl$). For experiment needs, we replaced "zoom" with "rotate" that would hold details about the sender (e.g., cameraID: 10000), receiver (e.g., cameraID: 0003), operation name (e.g., rotate), and degree of agreed rotation (e.g., 45).

1. Detection efficiency: To assess the testbed's capacity of detecting commitment violations, we considered 2 cases.
   (a) The first case targets $\boldsymbol{D}$-$C_1^{\mathcal{S}_4}$ and uses 5 sets of experiments, each set consisting of samples of 10, 20, 30, 40, and 50 video streaming sessions captured from midnight to midday and extracted from the monitor manager's database. Each session is formed in response to a user's request like Police and singles out a specific camera. Upon releasing the video content to the user, the execution manager processes the streamed video using an in-house face recognition module developed in Python and Open Source Computer Vision Library (OpenCV, `opencv.org`). When a face is detected in the streamed video, the execution manager notifies the monitor manager regardless of who has requested the streaming. Details about a user's session (userID, cameraID, cameraView, and session starting and ending time) are collected by the control panel and submitted to the monitor manager for analysis. This one checks the collected details with regard to the active commitments and report possible violations that would be displayed in real-time on the control panel. During analysis, the monitor manager decides on 3 cases per session (Fig. 5):
      i. Completely violated (commonly known as True/Positive): the alert of violation is confirmed in the sense that the user (e.g., social services) is not authorized to consult parts of the video content due to the presence of persons' faces in this video.

ii. Completely satisfied (commonly known as True/Negative): the alert of violation is not confirmed in the sense that the user (e.g., Police) is authorized to consult the whole video content with or without persons' faces.

iii. Partially violated/satisfied: the alert of violation is simultaneously confirmed (social services are not allowed to check some parts of a video content) and not confirmed (A Police officer has temporarily been assigned to social services and hence, is allowed to check all the content).

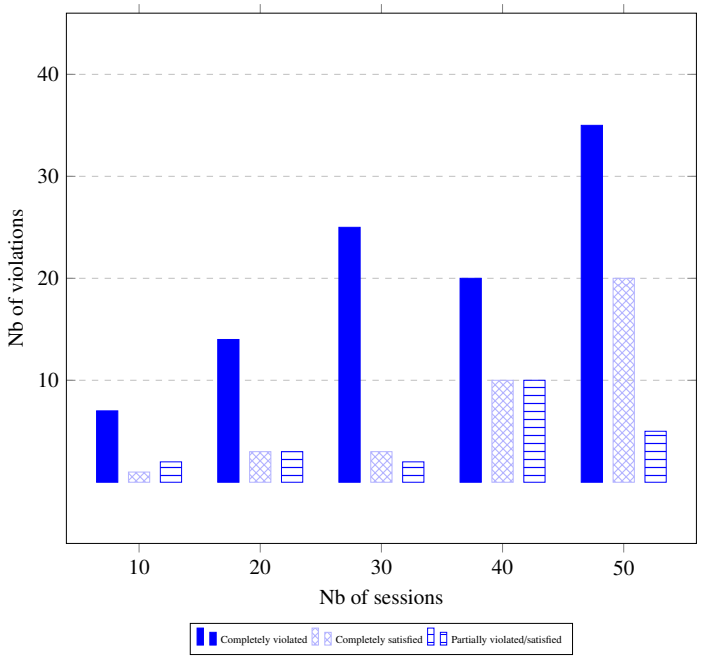



Figure 5: Testbed's effectiveness for violation detection of $\boldsymbol{D}$-$C_1^{S_4}$

As per Fig. 5, the monitor manager detects violations related to $\boldsymbol{D}$-$C_1^{S_4}$. In the second set of 20 video streaming sessions (one session is started by one authenticated user who selects one camera and a desired camera view) of a specific camera, the results show 14 completely violated, 3 completely satisfied, and 3 partially violated/satisfied commitments. It should be noted that the reported violations numbers, such as 7, 14, 25, and 20 would vary from one experiment to another depending on the streamed content used

during the experiments and the part of the day during which the streaming sessions are formed (i.e, morning, afternoon, and evening).

(b) The second case is about reporting violations in connection with different commitments. We considered 20 different user requests for concurrent rotation and streaming. These requests concern 3 cameras and are sent from the control panel using a dedicated user interface (Fig. 6). The execution manager receives the requests and dispatches them to a particular camera. We note that the monitor manager tracks the different requests and reports violations related to independent and combined commitments (Fig. 7). For instance, the monitor manager detects out of 20 requests to Camera 001, 10 true/positive violations of $\boldsymbol{D}$-$C_1^{\mathcal{S}_4}$, 18 of $\boldsymbol{B}$-$C_1^{\mathcal{S}_2}$, 13 of $\boldsymbol{B}$-$C_1^{\mathcal{B}_2}$, 25 violations of $\boldsymbol{D}$-$C_1^{\mathcal{S}_4}$ + $\boldsymbol{B}$-$C_1^{\mathcal{S}_2}$, and 37 violations of $\boldsymbol{D}$-$C_1^{\mathcal{S}_4}$ +$\boldsymbol{B}$-$C_1^{\mathcal{B}_2}$. These violations could be due to unauthorized rotation degree, not sending status to the system's authority, and/or unauthorized users watching faces on the streamed video.

2. Detection duration: To assess the testbed's time performance for detecting commitment violations, we computed the time of processing live video content and the time of reasoning over the commitments that are bound to a specific requester. We considered social services' streaming requests sent to Camera 001, Camera 002, and Camera 003 with different commitments (i.e, one to three of these commitments $\boldsymbol{D}$-$C_1^{\mathcal{S}_4}$, $\boldsymbol{B}$-$C_1^{\mathcal{S}_2}$ and $\boldsymbol{B}$-$C_1^{\mathcal{B}_2}$), then we recorded the time that the monitor manager took to process the enabled commitments so, that, potential violations are detected (Fig. 8). We noticed that the detection time for completely violated commitments depends on the streamed content, especially for the violation detection of $\boldsymbol{D}$-$C_1^{\mathcal{S}_4}$. The reasoning over the agreed commitments does not affect much the violation detection time, since the active commitments are checked instantly by the monitor manager.

### *4.3. Scalability analysis*

To assess the scalability of our system so, that, more streamed content is analyzed in an acceptable time frame, we simulated 100 IP-cameras in conjunction with the 3 reals

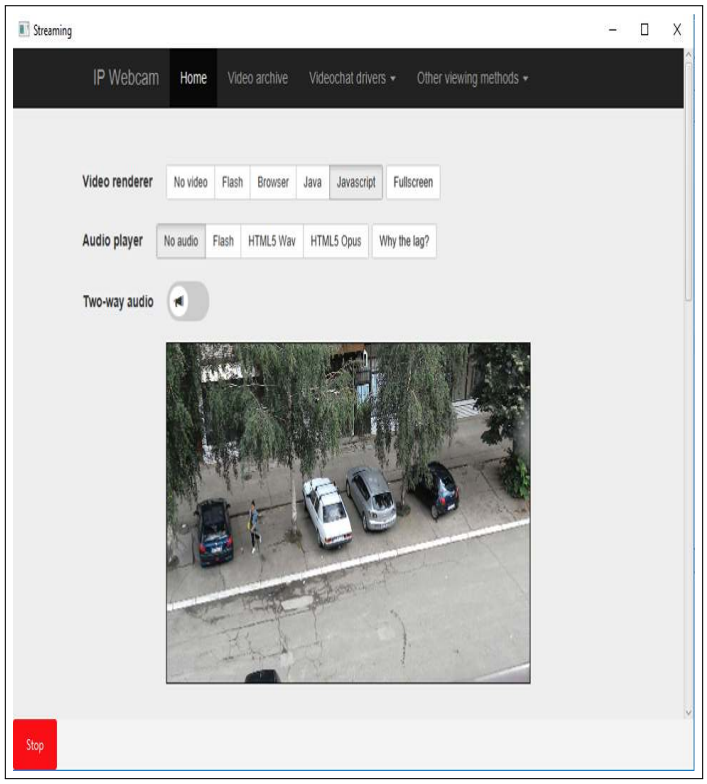



Figure 6: Live video streaming of a street in Novi Pazar

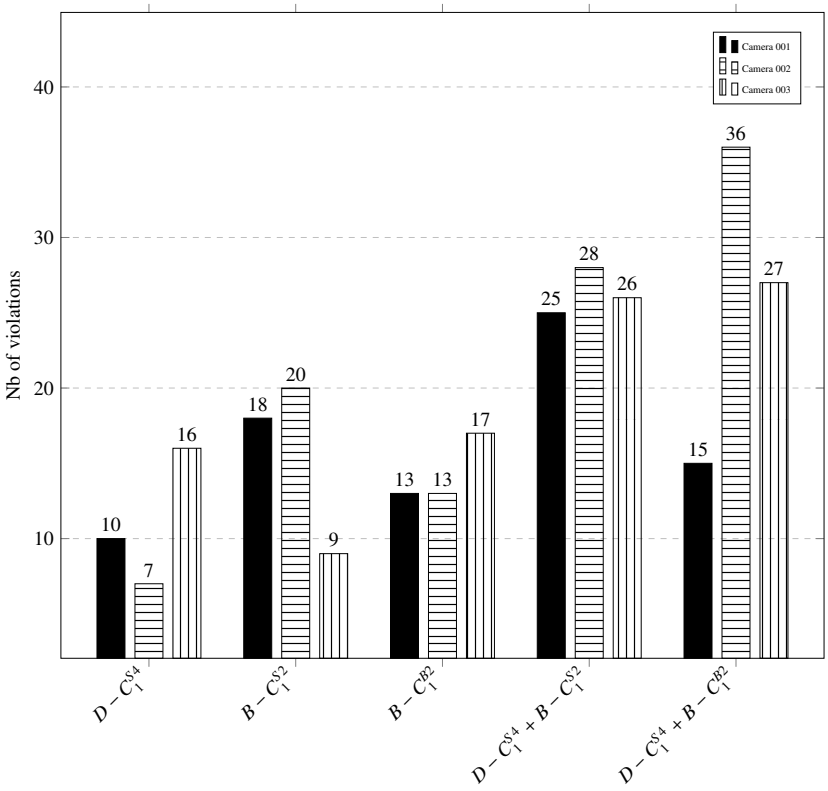



Figure 7: Testbed's effectiveness for violation detection of different commitment types

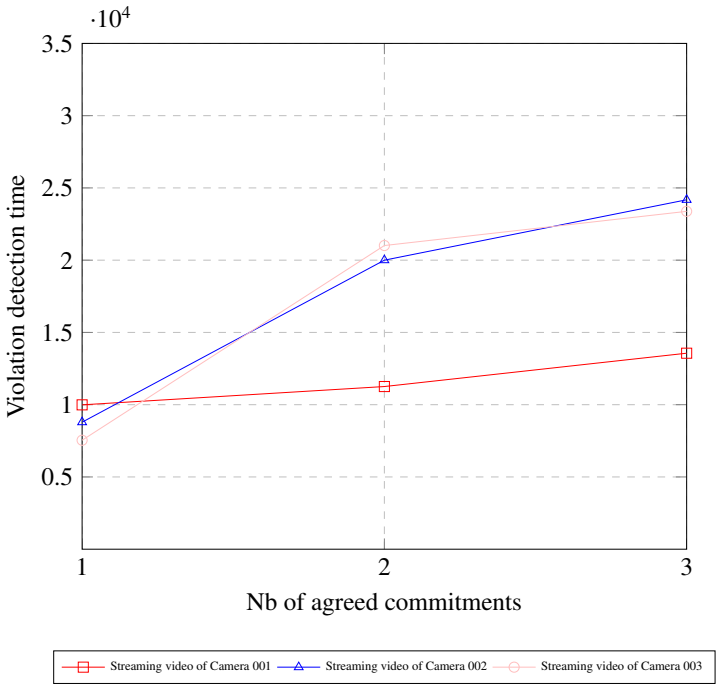



Figure 8: System performance evaluation based on detection time

that we used in the first experiment. As per Fig. 4, the IP cameras' observers analyze their respective streamed contents. We deployed the observers on 100 cloud instances based in Germany (*Hetzner CX11: 1vCPU, 2GB RAM, 20GB disk space costing 0.00347 euro per hour and running on Linux Ubuntu*). More cloud instances could be added to host more observers, should this be deemed necessary. During the scalability experiment, 6 sessions that lasted 24 hours each have been considered with different number of received messages (Table 1).

Table 1: Details about sessions' received messages

| | Session # | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| Nb of received messages | 2132110 | 3083143 | 5751902 | 6234271 | 8187840 | 9396200 |

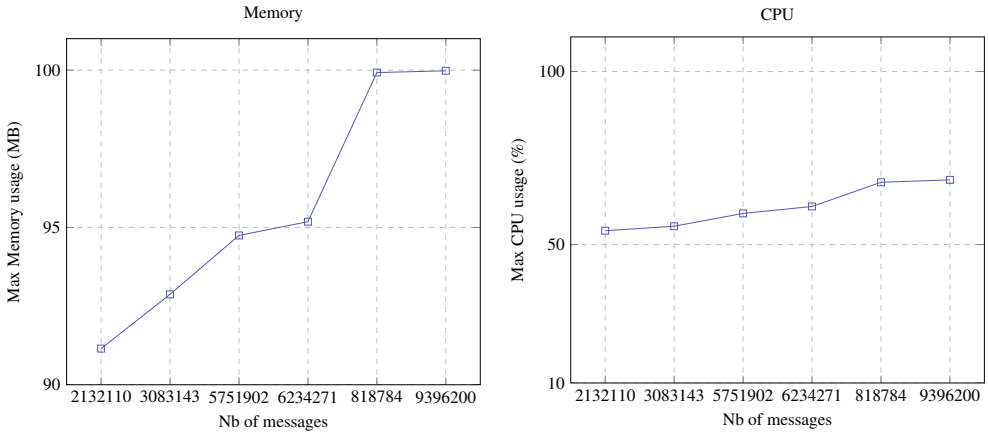
During the different sessions, we tracked the performance of the monitor manager using VisualVM [40]. It is a profiling tool that assesses a system's memory and CPU usage with minimum impact on its performance. Our objective is to evaluate the monitor manager capability of handling a growing amount of commitment-related violation/non-violation messages and storing them in a MongoDB database.

The results of the experiment in terms of performance metrics are related to CPU and memory usage. An example of these metrics for session 8 is reported in Table 2. First, VisualVM automatically collects these metrics over a session period. Then, it considers the maximum value of each metric from the collected values. Regarding memory usage, it peaks up to 99.97 MB when the number of messages hits 9396200 messages (Fig. 9a), while the CPU usage hits 68.7% (Fig. 9b), which demonstrates that the monitor manager performance remains "acceptable" (low memory and CPU usage under high load). This experiment also demonstrates that CPU usage increases when initializing the system, decreases afterwards, and remains steady during the session, which proves that the CPU usage is independent from the number of observers. The graphs in Fig. 10 represent the trend of memory and CPU usage for session 8 over different monitoring periods.

Based on the presented metrics, we conclude that the monitor manager is not sub-

Table 2: Extract of the monitor-manager performance during session 8

| Monitoring time (min) | Max memory usage (megabytes) | Max CPU usage (%) |
|---|---|---|
| 3 | 61.81 | 69.0 |
| 10 | 94.17 | 64.2 |
| 30 | 96.95 | 68.2 |
| 60 | 98.96 | 68.0 |
| 120 | 98.96 | 68.0 |
| 240 | 98.97 | 68.0 |

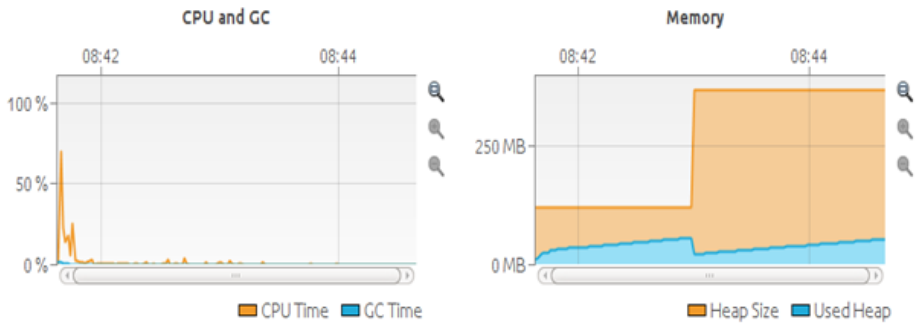



(a) Memory usage

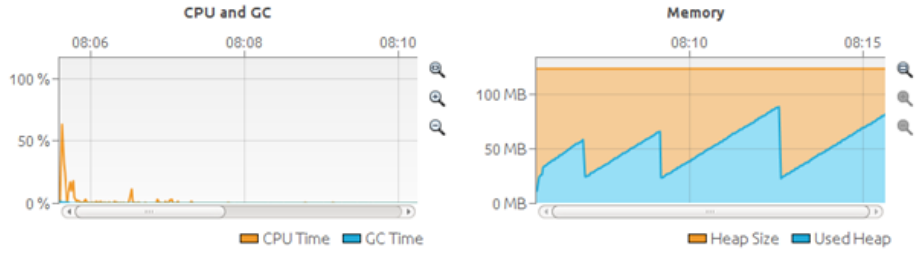
(b) CPU usage

Figure 9: Monitor manager performance

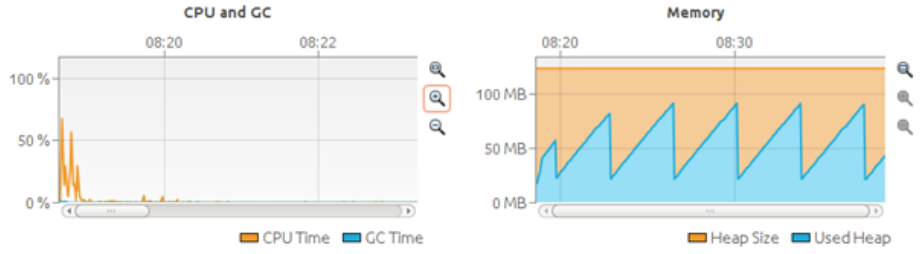
ject to severe stress when the number of cameras increases by adding more cloud instances hosting the observers that are in charge of analyzing streamed content. Hardware upgrade of the monitor manager is not necessary when increasing the number of observers. The burden of streamed content analysis is not on the monitor manager but on observers. Thanks to the cloud's load balancer, the observers perform the necessary analysis.



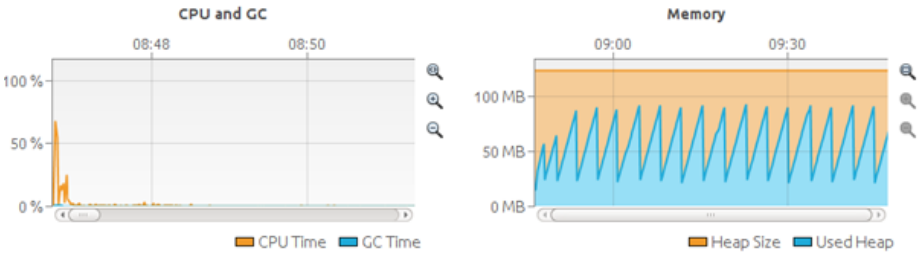

(a) 3 minutes of monitoring




(b) 10 minutes of monitoring




(c) 30 minutes of monitoring






(d) 60 minutes of monitoring

Figure 10: Trend of memory and CPU usage for session 8

## 5. Conclusion

This paper addressed the particular issue of things' limited responsiveness (being passive) to events impacting users' and peers' surroundings. This limitedness confines things into a data supplier role and, hence, prevents them from handling complex business scenarios like missing child. Both norms at the conceptual level and commitments at the operational level define the necessary capabilities that should empower things and, hence, make them proactive. Norms, specialized into business and social, regulate the operations of things in terms of obligations, permissions, and prohibitions whereas commitments, also specialized into business and social, ensure thing compliance with these norms at run-time. This compliance could be jeopardized because of absence of consistency among norms during their definitions and/or norm violation. Either way requires commitment monitoring so, that, things avoid sanctions due to non-compliance. A testbed and case study demonstrating the role of norms/commitments in empowering things have been implemented using Bevywise simulator and Mosquitto broker. In term of future work, we would like to expand the testbed by including more real things, examine further the partial violation/satisfcation of social commitments, and last but not least discuss the impact of trust on thing selection [26].

## References


[1] An Open Source MQTT v3.1 Broker. `https://mosquitto.org/`, 2018. (Accessed on 01/30/2018).

[2] F. Alkhabbas, M. Ayyad, R-C. Mihailescu, and P. Davidsson. A Commitment-based approach to realize Emergent Configurations in the Internet of Things. In *International Workshop on Engineering IoT Systems: Architectures, Services, Applications, and Platforms (IoT-ASAP'2017), co-located with International conference on software architecture (ICSA' 2017)*, Gothenburg, Sweden, May 2017.

[3] D. Androcec, B. Tomaš, and T. Kišasondi. Interoperability and Lightweight Security for Simple IoT Devices. In *Proceedings of the Information Systems Security Conference (ISS'2017) held in conjunction with the 40$^{th}$ Jubilee International Conven-*

*tion on Information and Communication Technology, Electronics, and Microelectronics (MIPRO'2017)*, Opatija, Croatia, May 2017.

[4] L. Atzori, A. Iera, and G. Morabito. SIoT: Giving a Social Structure to the Internet of Things. *IEEE Communications Letters*, 15(11), November 2011.

[5] L. Atzori, A. Iera, G. Morabito, and M. Nitti. The Social Internet of Things (SIoT) - When Social Networks Meet the Internet of Things: Concept, Architecture and Network Characterization. *Computer Networks*, 56(16), 2012.

[6] A. Banks and R. Gupta. OASIS Message Queuing Telemetry Transport (MQTT) TC — OASIS. `https://www.oasis-open.org/committees/tc\_home.php?wg\_abbrev=mqtt`, 2014. (Accessed on 01/30/2018).

[7] J. Brokenshire. Missing Children and Adults: A Cross-Government Strategy. *Home Office*, (ISBN: 978-1-84987-614-8), 2011.

[8] A. Bröring, S. Schmid, C.K. Schindhelm, K. Khelil, S. Käbisch, D. Kramer, J. Le Phuoc, D.and Mitic, D. Anicic, and E. Teniente. Enabling IoT Ecosystems through Platform Interoperability. *IEEE Software*, 34(1), Jan-Feb 2017.

[9] J. Chen, J. Ma, N. Zhong, Y. Yao, J. Liu, R. Huang, W. Li, Z. Huang, Y. Gao, and J. Cao. WaaS: Wisdom as a Service. *IEEE Intelligent Systems*, 29(6), 2014.

[10] F. Chesani, P. Mello, M. Montali, and P. Torroni. Representing and Monitoring Social Commitments Using the Event Calculus. *Autonomous Agents and Multi-Agent Systems*, 27(1), 2013.

[11] M. Dastani, L. van der Torre, and N. Yorke-Smith. Commitments and interaction norms in organisations. *Autonomous Agents and Multi-Agent Systems*, 31(2), 2017.

[12] M. Dastani, L. van der Torre, and N. Yorke-Smith. Agent-Oriented Cooperative Smart Objects: From IoT System Design to Implementation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, (Fortino, G. and Russo, W. and Savaglio, C. and Shen, W. and Zhou, M.), 2017 (forthcoming).

[13] DZone. The Internet of Things, Application, Protocls, and Best Practices. Technical report, https://dzone.com/guides/iot-applications-protocols-and-best-practices, 2017 (visited in May 2017).

[14] N. Fornara and M. Colombetti. Operational Specification of a Commitment-based Agent Communication Language. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'2002)*, Bologna, Italy, July 2002.

[15] H. Green. The Internet of Things in the Cognitive Era: Realizing the Future and Full Potential of Connected Devices. `www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=WWW12366USEN`, December 2015.

[16] IBM. Munich Genius of Things: Connected Workplaces and Buildings, February 2018.

[17] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Terziyan. Smart Semantic Middleware for the Internet of Things. In *Proceedings of the 5th International Conference on Informatics in Control, Automation and Robotics (ICINCO'2008)*, Funchal, Madeira, Portugal, 2008.

[18] A. Kott and D.S. Alberts. How Do You Command an Army of Intelligent Things? *IEEE Computer*, 50(12), 2017.

[19] J. Kwan, Y. Gangat, D. Payet, and R. Courdier. An Agentified Use of the Internet of Things. In *Proceedinmgs of the 2016 IEEE International Conference on Internet of Things (iThings'2016) and IEEE Green Computing and Communications (GreenCom'2016) and IEEE Cyber, Physical and Social Computing (CPSCom'2016) and IEEE Smart Data (SmartData'2016)*, Chengdu, China, December 2016.

[20] M. Kyas, C. Prisacariu, and G. Schneider. *Run-Time Monitoring of Electronic Contracts*. Springer Berlin Heidelberg, 2008.

[21] T. Leppänen and J. Riekki. A Lightweight Agent-based Architecture for the Internet of Things. In *Proceedings of the WEICE Workshop on Smart Sensing, Wireless Communications, and Human Probes*, Wuxi, China, March 2013.

[22] Z. Maamar, M. Sellami, N. Faci, E. Ugljanin, and Q.Z. Sheng. Storytelling Integration of the Internet of Things into Business Processes. In *Proceedings of the Business Process Management Forum (BPM Forum'2018) held in conjunction with the 16th International Conference on Business Process Management (BPM'2018)*, Sydney, NSW, Australia, 2018.

[23] P. Maes. Pattie Maes on Software Agents: Humanizing the Global Computer. *IEEE Internet Computing*, 4(1), July 1997.

[24] A. Martínez-Ballesté, P.A. Pérez-Martínez, and A. Solanas. The Pursuit of Citizens' Privacy: a Privacy-Aware Smart City is Possible. *IEEE Communications Magazine*, 51(6), 2013.

[25] L. Militano, M. Nitti, L. Atzori, and A. Iera. Enhancing the Navigability in a Social Network of Smart Objects: A Shapley-Value based Approach. *Computer Networks*, 103, 2016.

[26] D. Minoli and B. Occhiogrosso. Blockchain mechanisms for IoT security. *Internet of Things*, 1-2, 2018.

[27] A. M. Mzahm, M. S. Ahmad, and A. Y. C. Tang. Agents of Things (AoT): An intelligent operational concept of the Internet of Things (IoT). In *Proceedings of the 13th International Conference on Intellient Systems Design and Applications (ISDA'2013)*, Bangi, Malaysia, 2013.

[28] N. C. Narendra. Generating Correct Protocols from Contracts: A Commitment-based Approach. In *Proceedings of the 2008 IEEE Congress on Services - Part I (SERVICES I 2008)*, Honolulu, Hawaii, USA, 2008.

[29] College of Policing. Major investigation and public protection: Missing person investigations, https://www.app.college.police.uk/app-content/major-investigation-and-public-protection/missing-persons/missing-person-investigations/, 2016, (visited February 2018).

[30] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi. The Cluster Between Internet of Things and Social Networks: Review and Research Challenges. *IEEE Internet of Things Journal*, 1(3), June 2014.

[31] A. Paschke and M. Bichler. Knowledge Representation Concepts for Automated SLA Management. *Decision Support Systems*, 46(1), 2008.

[32] M. E. Pérez Hernández and S. Reiff-Marganiec. Towards a Software Framework for the Autonomous Internet of Things. In *Proceedings of the 4th IEEE International Conference on Future Internet of Things and Cloud (FiCloud'2016)*, Vienna, Austria, August 2016.

[33] P. Pico-Valencia and J. A. Holgado-Terriza. Semantic Agent Contracts for Internet of Agents. In *Proceedings of the 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW'2016)*, Omaha, NE, USA, 2016.

[34] C. Savaglio, G. Fortino, M. Ganzha, M. Paprzycki, C. Badica, and M. Ivanovic. Agent-Based Computing in the Internet of Things: A Survey. In *Proceedings of the 11th International Symposium on Intelligent Distributed Computing (IDC'2017)*, Belgrade, Serbia, October'2017.

[35] Z. Shen, H. Yu, L. Yu, C. Miao, Y. Chen, and V.R. Lesser. Dynamic Generation of Internet of Things Organizational Structures Through Evolutionary Computing. *IEEE Internet of Things Journal*, 5(2), 2018.

[36] M. N. Singh, A. K. Chopra, and N. Desai. Commitment-Based Service-Oriented Architecture. *Computer*, 42(11), November 2009.

[37] M. P. Singh. An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts. *Artificial Intelligence and Law*, 7(1), 1999.

[38] L. Slowey. IoT for the OAP: New Technology for an Ageing Population, 2016.

[39] A. Taivalsaari and T. Mikkonen. A Roadmap to the Programmable World: Software Challenges in the IoT Era. *IEEE Software*, 34(1), 2017.

[40] VisualVM Team. VisualVM: All-in-One Java Troubleshooting Tool. `https://visualvm.github.io/`, 2018. (Accessed on 02/05/2019).

[41] J. Vázquez-Salceda, H. Aldewereld, and F. Dignum. Norms in multiagent systems: from theory to practice. *Computer Systems Science and Engineering*, 20, 2005.

[42] M. Weiser. The Computer for the $21^{st}$ Century. *Newsletter ACM SIGMOBILE Mobile Computing and Communications Review*, 3(3), 199.

[43] Q. Wu, G. Ding, Y. Xu, S. Feg, Z. Du, J. Wang, and K. Long. Cognitive Internet of Things: A New Paradigm Beyond Connection. *IEEE Internet of Things Journal*, 1(2), April 2014.

[44] Y. X. Blind Review. In *Proceedings of the IEEE 12th International Conference on Research Challenges in Information Science (RCIS'2018)*, Rennes, France, May 2018.

[45] L. Xu, M. A. Jeusfeld, and P. W. P. J. Grefen. Detection tests for identifying violators of multi-party contracts. *ACM SIGecom Exchanges*, 5(3), 2005.

[46] F. Zambonelli. Key Abstractions for IoT-Oriented Software Engineering. *IEEE Software*, 34(1), January-February 2017.

[47] C. Zhang, C. Cheng, and Y. Ji. Architecture Design for Social Web of Things. In *Proceedings of the 1st International Workshop on Context Discovery and Data Mining (ContextDD'2012)*, Beijing, China, 2012.