Object (B)logging: a Semantic-Based Self-Description for Cyber-Physical Systems

Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING
Ph.D. Program
SSD: ING-INF/05

Final Dissertation

Object (B)logging: a Semantic-Based Self-Description for Cyber-Physical Systems

---

by
Giovanna Capurso :

Supervisor:

Prof. Michele Ruta

Coordinator of Ph.D. Program:
Prof. Alfredo Grieco

Course n°32, 01/11/2016-31/10/2019

Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING
Ph.D. Program
SSD: ING-INF/05

Final Dissertation

Object (B)logging: a Semantic-Based Self-Description for Cyber-Physical Systems

---

by

Giovanna Capurso :

_____

Referees:

Prof. Giancarlo Fortino

Prof. Andrea Omicini

Supervisor:

Prof. Michele Ruta

_____

Coordinator of Ph.D Program:

Prof. Alfredo Grieco

_____

Course n°32, 01/11/2016-31/10/2019

Per aspera ad astra

To my family
that believes in me
even more than I do

Contents

List of Figures

List of Tables

Abstract

Pervasive computing deals with heterogeneous mobile agents attached to ubiquitous micro-devices. In such scenarios, what one agent knows about the environment is based on perception components it uses or has access to, and it can be significantly different from another agent's knowledge. Furthermore, transient conditions and uncertainty affect perceptions and communication, aggravating the need to cope with the lack of complete and reliable information. Current solutions in the Internet of Things (IoT) are mostly based on centralized data collection and analysis and on top-down agent orchestration, with obvious limitations in latency, connection availability and data confidentiality. This thesis proposes a novel distributed knowledge-based framework named *object (b)logging* to tackle the above issues. The approach is conceived as a general-purpose evolution of the IoT, able to associate semantic annotations to real-world objects and events as well as to trigger complex objects choreography through advanced resource discovery. It envisions several smart entities organized in social networks, interacting autonomously and sharing information, cooperating and orchestrating resources through a published *micro-blog*. Ontology-referred context annotations produced and shared by individual smart objects in mobile ad-hoc networks are merged by means of novel Concept Fusion and enhanced Concept Integration reasoning services in Description Logics, specifically devised for context-aware multi-agent systems and tailored to resource-constrained devices. Management of incomplete information, reconciliation of inconsistencies in context descriptions, quick adaptation to changes and robustness against spurious or inaccurate information allow to progressively enrich a node's core knowledge in a private *micro-log*. Then it becomes able to identify on-the-fly the task(s) needed to change its own configuration or the environment state and automatically infer what useful capabilities it can provide to or needs from other entities in order to enact them, in a decentralized and collaborative fashion. A novel semantic-enhanced blockchain infrastructure underlies the dissemination, discovery and selection of services and resources. These tasks have been revisited as smart contracts with opportunistic and distributed execution, exploiting validation by consensus. The introduced paradigm ideally applies to pervasive cyber-physical systems, where several mobile heterogeneous micro-devices cooperate to connote and modify appropriately the environment they are dipped in, as demonstrated by relevant case studies and extensive experimental evaluations.

Chapter 1

Introduction

The *Internet of Things* (IoT) [8] vision is increasingly enabled by the miniaturization of microelectronic devices, enabling the deployment of relatively large numbers of heterogeneous micro-components capable of storing and exchanging not-negligible amounts of information. A significant limitation of current IoT lies in a limited compatibility of devices and software stacks from different manufacturers. This forces the design of single-purpose object networks and solutions, impairing a wider usefulness of the empoloyed technologies. The interoperability in –and relevance of– the IoT could be enhanced by embedding semantically rich and easily accessible information into the physical world. This vision has been called *Semantic Web of Things* (SWoT) [108] as the convergence of the Semantic Web [16] and the Internet of Things paradigms. The SWoT improves intelligence of embedded objects and autonomic information management in pervasive contexts, in order to better support user activities and provide general-purpose innovative services.

In pervasive computing, information is scattered in the form of atoms which deeply permeate the context [30]. Heterogeneous data streams are continuously retrieved and locally processed by mobile ad-hoc networks of *smart objects* dipped in the environment, in order to detect events of interest in observed areas and achieve objects cooperation. This gives rise to distributed cyber-physical systems, able to make decisions and interventions on the environment according to the detected context and events in a fully automated fashion. A smart object [9] is an intelligent agent running on a device equipped with embedded sensors, actuators, communication ports as well as (usually constrained) computation, storage and energy resources. For effective interaction, each smart object should describe itself and the context where it operates toward a variety of external devices and IoT applications. In order to improve flexibility and interoperability, Semantic Web standard

technologies [16] can be adopted for rich and unambiguous semantic-based information exchange. The essential benefit for smart entities concerns the integration of *Knowledge Representation and Reasoning* (KRR) capabilities into objects to automatically extract and process implicit useful information starting from explicit event and context detection.

This work proposes object (b)logging, a novel semantic-based framework for high-level knowledge discovery, integration and sharing within smart object networks in the Semantic Web of Things. Borrowing core relationships and structure from popular Social Networking Services (SNSs), devices enable specific interaction patterns for information dissemination and cooperative decentralized service/resource discovery. This selective choreography is triggered autonomously, based on the kind of managed resources and other contextual factors; this capability enhances interoperability across heterogeneous platforms and scalability in dense multi-agent environments.

Each object equipped with an embedded reasoning micro-engine can perform automated inference procedures to derive previously implicit knowledge out of information gathered from the environment or other smart entities. Detected and received information must be integrated in a coherent view by autonomous agents in order to recognize and annotate the context they are in. A novel knowledge fusion inference services is needed in order to merge different perspectives of heterogeneous entities over situations, suitable to robust distributed context monitoring even in the presence of incomplete and/or inaccurate information. In this way, starting from a basic descriptive core, smart objects continuously enrich their knowledge according to detected events and phenomena and offer advanced and semantically unambiguous descriptions of their state and context perspective toward the rest of the world in a self-contained fashion. Environment, process, subject and any other entities of interest in a cyber-physical system are expressed in Web Ontology Language (OWL 2) annotations [93]. According to the monitored and detected context, an agent must be able to take decisions on-the-fly, dynamically adapting its behavior and acting on the surrounding environment in order to suitably modify it by exploiting reaction and enforcement capabilities. This is achieved by means of available actuators encapsulated as resources/services, provided either by the agent itself or by other entities. These features are enabled by an advanced resource discovery facility supporting non-exact matches and providing a ranked list of discovered resources or services, suitable to accomplish the needed interventions. Computational limitations and resource volatility featuring mobile and pervasive computing contexts must be considered as well.

For this purpose, the proposal includes a framework redesigning semantic resource discovery thanks to an underlying blockchain infrastructure, ensur-

ing robustness, scalability and trust management. Basic resource/service registration, discovery, selection and finalization operations have been revisited as *smart contracts* in order to comply with an opportunistic and distributed execution leveraging validation by consensus. In this way, blockchain-based discovery acquires logic-based outcome explanation capabilities, obtained through non-standard inference for matchmaking among request and resources.

The overall approach results in a general-purpose, cross-domain semantic-based context detection, knowledge discovery and sharing facility among pervasive smart devices, supporting intelligent machine-to-machine interactions

Several prototypes were implemented and tested in simulation campaigns, basically devoted to assess feasibility, correctness and sustainability of the proposed solutions. Performance evaluation was carried out with reference to selected case studies, highlighting strengths and weaknesses of the approach.

The remainder of this dissertation is organized as in what follows.

Chapter 2 recalls the technological background for the research and provides a survey of related work.

Chapter 3 provides in detail functional and architectural description of the proposed object (b)logging framework and the novel knowledge fusion algorithms. An illustrative case study is presented to allow a better understanding of the proposal, along with performance evaluation experiments.

Chapter 4 describes the knowledge sharing and discovery framework underpinning social object interactions, along with a brief case study. Details about experimental results are provided.

Chapter 5 focuses on the semantic-enhanced blockchain infrastructure. The implemented software prototype and the experimental evaluation results are presented to assess the feasibility of the approach.

Chapter 6 concludes the dissertation by summarizing the main contributions and outlining open research perspectives.

Chapter 2

Background

This chapter presents an overview of the state of the art trends in the ubiquitous and pervasive computing: in these contexts, several heterogeneous mobile micro-devices embedded in the environment collect, process and exchange information cooperatively and autonomously, in order to adapt their behavior. Afterwards, a discussion of the languages and tools for Knowledge Representation and Reasoning in the Semantic Web is provided. Its vision and foundational technologies are described, in order to provide a theoretical and technical background for the subsequent chapters. Limitations of the state of the art are finally analyzed, in order to outline the problems tackled in the research work for this thesis.

2.1 The Social Internet of Things

Modern *Cyber-Physical Systems* (CPS) are smart complex systems engineered through a deep integration of embedded information processing subsystems and physical sub-systems. The convergence of the cyber and the physical worlds in this paradigm leverages technological progress in the Internet of Things, where everyday objects are augmented with communication and computation capabilities. In a CPS, a large number of sensors, actuators, and control devices is connected by a network to acquire, process, calculate, and analyze context information and to apply the results to the physical environment. Managing complexity calls for *smart objects* capable of communicating and coordinating autonomously, making decisions dynamically based on manifold factors, including the state of surrounding objects and places, as well as user activities and profiles if human-computer interaction is required. Flexible and meaningful relationships among smart devices in a

given environment should be established automatically to support articulate orchestration and coreography patterns.

Recent research in the so-called *Social Internet of Things* (SIoT) [9] is exploring models and architectures to reach this goal. Paradigms are often borrowed from Social Networking Services for human users. If properly adapted to the peculiarities and requirements of Multi-Agent Systems (MAS), they can support powerful approaches. SIoT offers several benefits and interesting perspectives for the IoT/CPS. The adoption of a *social model* for object information interchange gives structure (to some extent) to the intrinsically unpredictable interaction in the IoT/CPS, and therefore it gives an added value in terms of interoperability, autonomicity, versatility and coordination. This is not enough, however, for really cohesive CPSs: versatile cooperation, organization and integration can be achieved only if connected things can represent, discover and share information and services described in an articulate way by means of machine-understandable formalisms. Semantic Web technologies are candidates for such a role, as they can grant interoperability grounded on formal logic semantics [125].

2.1.1 Smart objects

The *Internet of Things* term refers to a loosely coupled, decentralized, and dynamic system in which large numbers of everyday objects are globally interconnected and endowed with smartness, becoming active participants in business, logistics, information, and social processes [149]. Such *things* can be commonly defined as smart objects (SOs) and, if supported by an "anywhere, anytime, and anything connection" [8], they represent the fundamental building blocks for the IoT [68], as well as for the derived CPS paradigms. In fact, smart objects are able to provide highly pervasive cyber-physical capabilities and services to both humans and machines thanks to their communication, sensing, actuation, embedded processing, and even reasoning abilities.

In literature, a *smart object* [141] is defined as an intelligent agent running on a device equipped with embedded sensors, actuators, communication ports as well as (usually constrained) computation, storage and energy resources. Smart objects are able to gather data streams for internal and external parameters, to adapt themselves to the environment and/or to act in order to modify it.

Basically, four kinds of information are managed, as shown in Figure 2.1:

- *Context Information*: a description of the environment where the object operates, according to data collected through built-in sensors;

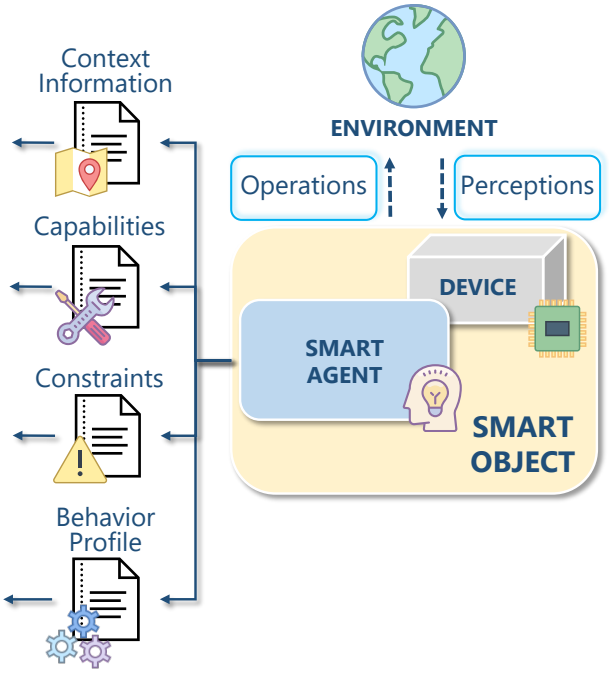


Figure 2.1: Smart object descriptions

- *Capabilities*: smart object sensing and actuation features, respectively called perception and operation capabilities;
- *Constraints*: limits and features, imposed by itself or by other entities, modelling the desired state of the environment. They influence the smart object's behavior so that it is in charge of driving adaptation honoring them;
- *Adaptation/Behavior Profiles*: context-aware reactive patterns in the smart object, declared using a rule-like form comprising two sections:
  - *Preconditions*: existing requirements about the present state of the environment that must be met for the profile to be activated;
  - *Postconditions*: desired patterns in the future state of the environment that must be accomplished by the agent in order to honor the profile. Smart objects should decide how to meet the postcondition requirements, depending on their internal logic and built-in actuators.

The approach explored in this work adopts the previous definitions and devises them in a novel way, as detailed in Chapter 3. Particularly, informa-

tion is expressed in a semantically rich and structured formalism grounded on the *Attributive Language with unqualified Number restrictions* ($\mathcal{ALN}$) Description Logic, which is a subset of the *Web Ontology Language* (OWL) 2 [93] standard adopted for modelling ontologies and annotating resources in the Semantic Web. A background on knowledge representation languages and technologies is provided in Section 2.2. In the proposed object (b)logging paradigm, each smart object is also equipped with a reasoning engine performing automated inference procedures to derive implicit knowledge out of semantic-based information gathered from the environment. In this way, an object becomes able to identify on-the-fly the task(s) needed to change its own configuration or to act on the environment, also exposing information possibly useful to nearby objects in a *micro-blog*, which acts as a medium for social interactions.

2.1.2 State of the art

In latest years, social networking services have changed personal interaction habits and relationships management on a global scale. Members of SNSs create personal profiles with basic information about themselves; connect with other users in either bidirectional (*e.g.*, *friendship*, *group*) or unidirectional (*e.g.*, *follower*) relationships; post text and/or multimedia items on their *wall* (*i.e.*, log) for sharing with their contacts; flag (*tag*) some contacts to associate them and draw their attention to a certain element; respond to content published by other users with comments and reactions (*e.g.*, *like*). SNS adopters generally manifest an intention to continue using them [78], because SNSs provide both utility (*extrinsic* value) and gratification (*intrinsic* value). Their usefulness also grows as they connect more users, and particularly *complementary* ones [78], since opportunities increase for discovering unforeseen information and services.

In the Social Internet of Things [9], objects engage with one another independently from direct user interactions and human SNS. Things act as autonomous agents, building networks of social relationships and exploiting them to share information and services more effectively. Research on SIoT is very active and covers a wide range of related topics, as surveys in [90] and [129] show.

Several classifications of things relationship types exist in literature. In [11] and subsequent works, *parental* (same manufacturer), *co-location* (same environment), *co-work* (cooperation), *co-ownership* (same owner) and *social* (sporadic or frequent contact) were identified as basic object relationships. The analysis in [142] was inspired by literature on human SNS, instead, and

resulted in an ontology allowing objects to manage their policies, friends and reputation. The ontology model developed in [72] included social relationships among events, people and objects in IoT environments.

The SIoT aims at extended device collaboration. Service-oriented architectures are one of the dominant paradigms, where intelligent service/resource discovery is a core capability. Most SIoT approaches rank services/resources w.r.t. a request/profile specification based on a combination of (i) object social connectivity, (ii) object mobility patterns, and (iii) preference similarity. This work exploits (i) and (iii), while (ii) is taken into account implicitly through dynamic relationship set-up and tear-down. In [75] a cosine similarity was computed combining metrics (ii) and (iii), improving results w.r.t. each single method, although preferences and resources were described by means of a simplistic set of keywords, lacking formal meaning.

Semantics-based approaches improve both context-awareness and discovery capabilities in multi-agent systems. In [98] distributed OWL *Knowledge Base* (KB) management supported machine-to-machine social interactions in control networks. Every connected object proactively discovered other nodes in the network; requesters were then able to distribute queries automatically among known devices equipped with reasoning engines. Unfortunately, supported inferences were very basic, limiting the applicability of the proposal. In [52] semantics-based situation detection and goal retrieval were exploited for matchmaking with task profiles for recommending activities to users, based on their current context. Nevertheless, social interactions occurred only between devices and users; furthermore, adopted rule-based reasoning could not retrieve approximate matches when exact ones did not exist. Likewise, [51] focused on activities of daily living in smart homes, generating task-oriented recommendations for user's situational goals. In [77] Near Field Communication (NFC) technology mediated social interactions between everyday objects and agents running on mobile devices. Ontology-based representations supported context-awareness, enabling both reactive and proactive agents behaviors, exploiting rule-based reasoning for planning toward situation-dependent goals. Though interesting, the approach does not appear general enough to support a wide range of SIoT scenarios.

Recent SIoT proposals increasingly adopt cloud-assisted architectures, where social objects are *virtualized*, *i.e.*, functionalities are mapped to software agents managed by Platform as a Service (PaaS) infrastructures. Relevant examples include *Paraimpu* [95], *Lysis* [42], *Smartbuddy* [94] and the platforms in [51], [145] and [70]. Notably, the *ASSIST* framework [57] exploited semantics-based rules to implement social network primitives. While granting higher scalability, availability and robustness, cloud-assisted approaches can exhibit non-negligible drawbacks, including longer delays, higher

network load and greater energy usage for pervasive devices [39]. As such, it seems appropriate essentially for scenarios and applications where a dependable global networking infrastructure is available. Although SIoT research trends take this kind of connectivity more and more for granted, many IoT technologies in the present and near future cannot provide it [10]. The approach explored in this thesis, instead, is based on *Edge Computing* models [128] and is compatible with mobile ad-hoc networks (MANETs), not requiring a stable connectivity infrastructure. This makes it suitable for challenging scenarios such as disaster recovery or environmental monitoring and surveillance.

Among challenging scenarios, the SIoT paradigm is being increasingly applied to Vehicular Ad-hoc NETworks (VANETs) and urban mobility [129], like the case study proposed in Section 4.2. The fast motion of nodes (*i.e.*, vehicles) leads to highly volatile network topologies as well as strict computation and communication latency constraints for safety-related services. Furthermore, situation awareness in such environments requires information fusion from multiple heterogeneous sources. Social interaction paradigms in VANETs –often denominated Social Internet of Vehicles (SIoV) [3]– have been demonstrated to be effective in improving information discovery [111] and resource allocation [148].

Trust management is one of the most relevant issues in the SIoT [146]. As discussed in Chapter 4, this work focuses on application-level trust, particularly in social objects relationships. Trust metrics can be retrieved from friends (recommendation), reliable intermediaries (reputation) or direct knowledge [32]. In [85] both a *subjective* and an *objective* (*i.e.*, shared) model were introduced, exploiting techniques mutuated from peer-to-peer networks. The collaborative Cognitive Radio framework in [86] leveraged the SIoT relationship types described above in [11] to improve both the reach of channel status queries and the trustworthiness of information through weights depending on the degree of friendship. A refinement was proposed in [26], where trust was defined as a function of reputation on past transactions, relationship type and energy status. Reputation combined direct and friend recommendation scores, while the relationship type between two devices was derived from a correlation of trustworthiness w.r.t. mutual friends. Similarly, in [56] a *honesty* model was composed out of spatial (object proximity), temporal (frequency and duration of interactions), relationship and credibility (cooperativeness and penalties) metrics. The credibility aspect is the closest to the proposal in this work. A similar approach was adopted in [25], albeit in that case honesty referred to the absence of malicious behaviors: the main goal was, in fact, to devise a robust trust model against attacks to service discovery and management. For this purpose, a trust propagation and aggregation

scheme was introduced, conceptually similar to the objective trust model in [85]. The above works, however, aimed only at improving transaction success and did not consider dynamic social relationships. That issue was tackled in [92], exploiting Bayesian belief propagation networks.

As a final remark, all the above works focused either on service discovery and composition or on dynamic trust management, disregarding or trivially simulating the other aspect.

2.2 Knowledge Representation and Reasoning in pervasive contexts

In ubiquitous and pervasive contexts, intelligence is integrated into objects and physical locations by means of a relatively large number of heterogeneous micro-devices. Each conveys a small amount of useful data, which should be processed and exchanged by smart objects in order to enable adaptive context-aware behaviors in a range of applications. Agents must be able to integrate *detected* and *received* information in a coherent view by fusing partial and redundant information, and by retracting consequences when new conflicting knowledge becomes available. Furthermore, high degrees of autonomic capability are required in order to trigger actions or make interventions on the environment according to the detected context without human interaction. Knowledge Representation and Reasoning (KRR) techniques and technologies allow information modelling based on formal and rigorous interpretation of its meaning (semantics). This enables not only greater interoperability across different HW/SW platforms, but also reasoning tasks *i.e.*, inferences, to derive new implicit insight from information explicitly asserted in a Knowledge Base. Among the many available KRR languages and tools, those born from the Semantic Web initiative enjoy global adoption and high optimization of algorithms and implementations. In the following subsections, Description Logics (DLs) family and non-monotoning reasoning tasks relevant for this work are recalled in detail.

2.2.1 The Semantic Web of Things

In latest years, the Semantic Web of Things [108] vision is merging the Semantic Web and the Internet of Things. Its goal is to exploit *semantic* annotations conveyed by heterogeneous micro-components to embed intelligence into real-world objects, locations and events. Large numbers of smart

devices interact autonomously to achieve context awareness and provide high-level services to users, via decision support and task automation. In order to enable such vision, approaches have to deal with pervasive computing constraints, *i.e.*, unpredictability, dependence on context, severe resource limitations.

This work is grounded on the general framework for the SWoT named *ubiquitous Knowledge Base (u-KB)* [108]. Basically it can be envisioned as an evolution of classic KB paradigms, also providing information storage, communication and processing capabilities to bridge the gap between physical and digital worlds. As sketched in Figure 2.2 in terms of functional architecture, a u-KB is defined as a distributed knowledge base whose individuals (assertional knowledge) are physically tied to micro-devices on the objects disseminated in an environment, without centralized coordination. Each annotation refers to an ontology providing the conceptual model for the particular domain. Current identification and sensing technologies are extended by a semantic adaptation micro-layer for application-level protocols, defining data formats and storage schemes to host annotated fragments, while keeping backward compatibility with standard technologies and applications. Furthermore, specialized compression algorithms [105][121] are defined for ontology languages, to properly store and transmit annotations on tiny embedded mobile devices such as RFID tags or wireless sensors. The benefits of compression apply to the whole ubiquitous computing environment, as decreasing data size means shorter communication delays, efficient usage of bandwidth and reduced battery drain for mobile devices.

Novel application models are possible, enabling a direct interaction with objects by-passing centralized information repositories: such approaches refer to really pervasive environments where agents –human or not– can directly dialog with objects without intermediaries. The u-KB model is supported by an advanced matchmaking where inference tasks are distributed among mobile computing devices which provide minimal computational capabilities. Such an approach enables objects to describe themselves toward the rest of the world in a self-contained fashion. The u-KB layer provides common access to information embedded into semantic-enhanced micro-devices populating a smart pervasive environment, while information processing and reasoning tasks can be performed either by local hosts (so enabling semantic-enhanced ubiquitous applications) or by a remote entity through a gateway exposing a high-level interface (so integrating the local environment with the WWW). Reuse of Semantic Web standards is the key to simplify the integration of u-KBs in larger information infrastructures. Notice that, the u-KB approach is more focused on allowing semantic-based dynamic resource dissemination and discovery in mobile ad-hoc contexts, without necessarily linking to the

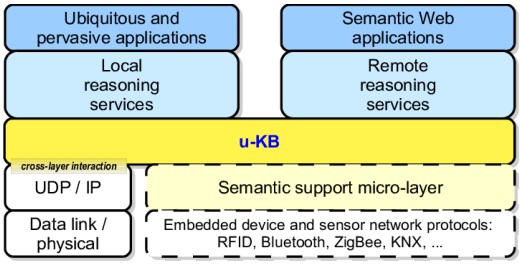


Figure 2.2: Semantic Web Of Things framework

Internet to provide useful services (even though connection can be exploited when available), so being more inclined to really pervasive computing.

The proposed object (b)logging model is a u-KB specialization, where smart objects dynamically build a u-KB as a private micro-log and disseminate information in their mobile ad-hoc network as a public micro-blog.

2.2.2 Description Logics

Description Logics (DLs) are a family of Knowledge Representation (KR) languages in a fragment of First Order Logic (FOL) [19, 35]. DLs allow to represent knowledge by means of:

- *concepts a.k.a. classes*, representing sets of objects;
- *roles a.k.a. properties*, representing relationships between pairs of concepts;
- *individuals*, *i.e.*, named instances of classes.

These elements can be combined via *constructors* to create DL expressions, whose formal semantics is specified by means of an *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ associating each term to a subset of the universe of discourse (the *domain* Δ). Concept *conjunction* is interpreted as set intersection: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$. Concept *disjunction* is interpreted as set union: $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$. The connector $\neg$, if present, is interpreted as *complement*.

An *ontology* (*a.k.a.* terminology, terminological box, TBox) is composed by two types of *assertions*: *inclusion*, which allows to define *is-a* relationships between classes; *equivalence*, which allows to give a name to a particular concept expression.

DLs are distinguished by the constructors they provide. The *Attributive Language*($\mathcal{AL}$) has been introduced in [118] as a minimal language that is of practical interest. Constructs of $\mathcal{AL}$ are reported in what follows:

- $\top$, *universal concept*. All the objects in the domain.
- $\bot$, *bottom concept*. The empty set.
- A, *atomic concepts*. All the objects belonging to the set A.
- $\neg A$, *atomic negation*. All the objects not belonging to the set A.
- $C \sqcap D$, *intersection*. The objects belonging to both C and D.
- $\forall R.C$, *universal restriction*. All the objects participating in the relation R whose range are all the objects belonging to C.
- $\exists R$, *unqualified existential restriction*. There exists at least one object participating in the relation R.

In order to increase the expressiveness, the $\mathcal{AL}$ language can be extended with additional constructs such as:

$\mathcal{N}$: $(\geq nR)$[1], $(\leq nR)$, $(= nR)$[2], *unqualified number restrictions*. Respectively the minimum, the maximum and the exact number of objects participating in the relation R.

$\mathcal{U}$: $C \sqcup D$, *concept union*. The objects belonging to C or D.

$\mathcal{E}$: $\exists R.C$, *full existential quantification*. Existential restrictions that have fillers other than $\top$.

$\mathcal{C}$: *complex concept negation*. Negation of concepts that are are comprised of other concepts.

$\mathcal{I}$: R^{-}, *inverse roles*. Refer to the inverse of a binary relation.

The formula $\mathcal{AL}[\mathcal{N}][\mathcal{U}][\mathcal{E}][\mathcal{C}][\mathcal{I}]$ is used to indicate the $\mathcal{AL}$ language extensions.

This work adopts the *Attributive Language with unqualified Number restrictions* ($\mathcal{ALN}$) DL as reference. It provides adequate expressiveness while keeping polynomial complexity, both for standard and non-standard inferences. Table 2.1 summarizes syntax and semantics of constructors and assertions in $\mathcal{ALN}$.

[1]Notice that $\exists R$ is equivalent to $(\geq 1R)$

[2]Notice that $(= nR)$ is a shortcut for $(\geq nR) \sqcap (\leq nR)$

Table 2.1: Syntax and semantics of $\mathcal{ALN}$

Name	Syntax	Semantics
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
Universal quantification	$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow \ d_2 \in C^{\mathcal{I}}\}$
Number restrictions	$\geq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geq n\}$
	$\leq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leq n\}$
Inclusion	$A \sqsubseteq D$	$A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Equivalence	$A \equiv D$	$A^{\mathcal{I}} = D^{\mathcal{I}}$

Web Ontology Language (OWL)

Web Ontology Language (OWL) [93] is a W3C Recommendation extending RDF Schema for creating ontologies and expressing metadata on resources in the Semantic Web. OWL uses the Internationalized Resource Identifier (IRI) to for unique resource naming and is based on the Resource Description Framework (RDF) [120] knowledge model. OWL branches into three different levels of complexity, from the least to the most expressive:

- *OWL Lite*: allows very basic taxonomy and constraints definition;
- *OWL DL*: enables a fairly wide expressiveness while retaining computational tractability. The name indicates a direct correspondence between OWL and Description Logics;
- *OWL Full*: provides the highest level of flexibility and expressiveness, sacrificing computational tractability.

OWL Lite is a subset of *OWL DL*, which is in turn a subset of *OWL Full*. The subset of *OWL DL* elements allowing to express the $\mathcal{ALN}$DL is presented in Table 2.2 in *RDF/XML* syntax [119].

In pervasive scenarios featured by volatile nodes interacting in an opportunistic fashion in order to achieve a common goal, *OWL Full* is not suitable because of its intractability, while suitable fragments of *OWL DL* provide a good trade-off between expressiveness and complexity. In this work the domain of interest was modeled using the latest specification of OWL language, *i.e.*, OWL 2 [93]. It supports a variety of syntaxes to read, store and exchange knowledge conceptualization among applications. OWL 2 includes three sublanguages, named profiles: (i) *OWL 2 EL*, a fragment that has

Table 2.2: Correspondence between OWL RDF/XML and DL syntax

OWL RDF/XML syntax	**DL syntax**
<owl:Thing>	$\top$
<owl:Nothing>	$\bot$
<owl:Class rdf:ID="C">	C
<owl:ObjectProperty rdf:ID="R">	R
<rdfs:subClassOf>	$\sqsubseteq$
<owl:equivalentClass>	$\equiv$
<owl:disjointWith>	$\neg$
<owl:intersectionOf>	$\sqcap$
<owl:allValuesFrom>	$\forall$
<owl:someValuesFrom>	$\exists$
<owl:maxCardinality>	$\leq$
<owl:minCardinality>	$\geq$
<owl:cardinality>	$=$

polynomial time reasoning complexity; (ii) *OWL 2 QL*, designed to enable easier access and query to data stored in databases and (iii) *OWL 2 RL*, a rule subset of OWL 2. Each profile applies for specific use cases and offers a different trade-off between expressiveness and reasoning efficiency. Unlike early OWL sublanguages, OWL 2 profiles are mutually unrelated.

2.2.3 Reasoning services

Polynomial-complexity *structural* reasoning algorithms [122] for $\mathcal{ALN}$ concept descriptions can be exploited in order to enable a distributed information aggregation and discovery in agent networks and MASs.

To this aim, when an $\mathcal{ALN}$ KB is loaded, it is preprocessed performing the *unfolding* and *Conjunctive Normal Form* (CNF) *normalization* [106]. Particularly, given a TBox $\mathcal{T}$ and a concept C, the unfolding procedure recursively expands references to axioms in $\mathcal{T}$ within the concept expression itself. In this way, $\mathcal{T}$ is not needed any more when executing subsequent inferences. The CNF translation in then obtained by applying a set of predefined substitutions. Any concept expression C in CNF can be expressed as:

$$C \equiv C_{CN} \sqcap C_{\leq} \sqcap C_{\geq} \sqcap C_{\forall}$$

where

- C_{CN}: conjunction of (possibly negated) concept names;
- $C_{\leq}$: conjunction of maximum cardinality restrictions, at most one per role;
- $C_{\geq}$: conjunction of minimum cardinality restrictions, at most one per role;
- $C_{\forall}$: conjunction of universal restrictions, at most one per role; fillers are recursively in CNF.

Normalization preserves semantic equivalence with respect to models induced by the TBox; furthermore, CNF is unique (up to commutativity of conjunction operator) [34]. The normal form of an unsatisfiable concept is simply $\bot$.

The following definition adapted from [103] will be used in the rest of the work:

Definition 1 ***(Concept Components)*** *Let C be an unfolded and CNF-normalized concept expression satisfiable w.r.t. a TBox $\mathcal{T}$ in a DL $\mathcal{L}$, formalized as $C \equiv C_{CN} \sqcap C_{\leq} \sqcap C_{\geq} \sqcap C_{\forall}$: then*
– each concept name or negated concept name in C_{CN} is a Concept Component of C;
– each maximum cardinality restriction in $C_{\leq}$ is a Concept Component of C;
– each minimum cardinality restriction in $C_{\geq}$ is a Concept Component of C;
– $\forall R.D_k$ is a Concept Component of C for each $\forall R.D$ in $C_{\forall}$ and for each D_k Concept Component of D.

Given a DL ontology $\mathcal{T}$ and S, D two satisfiable concepts in $\mathcal{T}$, the *satisfiability* and *subsumption* standard inference services provided by DL-based systems [14] can be formalized as follows:

- *Subsumption*: checks if S is more specific than D w.r.t. the ontology $\mathcal{T}$, *i.e.*, $\mathcal{T} \models S \sqsubseteq D$.
- *Satisfiability*: verifies if the conjunction of S and D is satisfiable w.r.t. the ontology $\mathcal{T}$, *i.e.*, $\mathcal{T} \not\models S \sqcap D \sqsubseteq \bot$.

In real-world application scenarios featuring articulated and –often– conflicting descriptions, standard inference services like *Subsumption* and *Satisfiability* are often inadequate, as they provide just a boolean answer. An

outcome explanation is required in more advanced settings, dealing with heterogeneous information from several independent sources. Non-monotonic reasoning tasks originally defined for belief revision are needed. The following non-standard inferences are particularly relevant for the purposes of this work:

- *Concept Contraction* [106]: if $\mathcal{T} \models S \sqcap D \sqsubseteq \bot$, *i.e.*, S and D are not compatible with each other, Concept Contraction (CC) is able to determine a pair of concepts $\langle G, K \rangle$ such that $\mathcal{T} \models D \equiv G \sqcap K$, and $K \sqcap D$ is satisfiable in $\mathcal{T}$. Then K is called a contraction of D according to S and $\mathcal{T}$. G (for *Give up*) is the explanation about what in D is incompatible with S and must be retracted to obtain an expression K (for *Keep*) such that $K \sqcap S$ is satisfiable in $\mathcal{T}$. Hence, Concept Contraction Problem (CCP) amounts to an extension of a (in)*Satisfiability* one.

- *Concept Abduction* [106]: if $\mathcal{T} \models S \sqcap D \not\sqsubseteq \bot$ and $\mathcal{T} \models S \not\sqsubseteq D$ then Concept Abduction (CA) finds a concept H (for *Hypothesis*) such that $\mathcal{T} \models S \sqcap H \sqsubseteq D$. Basically, H represents what is in D but is missing from S. In particular, solving a Concept Abduction Problem (CAP) provides an explanation when *Subsumption* does not hold.

- *Concept Difference* [136]: if $\mathcal{T} \models S \sqsubseteq D$, then the difference $S - D$ is a concept E such that $S \equiv D \sqcap E$.

- *Bonus* [28]: extracts a concept B from S which denotes something that S provides even though not specified in D. It is basically computed via Concept Abduction between contracted versions of D and S. Bonus can also be seen as an extension of Concept Difference, which remains valid if $\mathcal{T} \models S \not\sqsubseteq D$ and even if $\mathcal{T} \models S \sqcap D \sqsubseteq \bot$.

- *Concept Covering* [97]: Given D and a set of semantic descriptions $R = \{S_1, S_2, ..., S_k\}$, where D and $S_1, S_2, ..., S_k$ are satisfiable in $\mathcal{T}$, the Concept Covering Problem (CCoP) aims to find a pair $\langle R_C, H \rangle$ where R_C contains concepts in R (partially) covering D with respect to $\mathcal{T}$ and H is the (possible) part of D not covered by concepts in R_C.

For both Concept Abduction and Concept Contraction , minimality criteria are defined –since one usually wants to hypothesize or give up as little as possible– which induce numerical distance (*penalty*) functions, based on thbe norm of expressions H and G respectively. These penalties can be combined through a *utility function* such as 2.1, in order to evaluate goodness of match approximation and are studied and applied in *semantic matchmaking*

problems [28]. Semantic matchmaking is basically the process of finding best matches of a *request* (named D in the above definitions) among available *resource ads* (named S above), where both the request and the ads are semantically annotated w.r.t. a reference ontology [28]. In such scenarios, the result is a list of ads ranked by semantic proximity, computed as the inverse of the *semantic distance* p:

$$p(D,S) = \frac{w \cdot penalty_{(c)} + (1-w) \cdot penalty_{(a)}}{penalty_{(a),max}} \tag{2.1}$$

where 2.1 $penalty_{(c)}$ is the penalty calculated by Concept Contraction between D and S, while $penalty_{(a)}$ is the penalty value of the Concept Abduction procedure between D and the consistent part K of S. The value of p is normalized w.r.t. the maximum possible semantic distance $penalty_{(a),max}$ which is the one between D and the most generic concept $\top$, and depends only on axioms in the reference domain ontology [106]. The parameter w ranges from 0 to 1 and determines the relative weight of explicitly conflicting elements w.r.t. unspecified ones.

The principles underpinning semantic matchmaking are:

- *Open World Assumption.* The absence of a characteristic in a description should not be interpreted as a constraint of absence, but as unknown or irrelevant information.

- *Non-symmetric evaluation.* A matchmaking system may give different evaluations to the match between a resource S and a request D, depending on whether it is trying to match S with D, or D with S.

Five different match classes (categories) [73] reported in Table 2.3 are originated by matchmaking process. The most desired match is obviously the exact one, but from the viewpoint of a requester full match is equally acceptable. However, potential and partial matches are the most common in real complex scenarios. By means of Concept Contraction and Concept Abduction it is possible to move from a partial match to a full one by exploiting a query refining process:

$$\textbf{partial} \rightarrow \textbf{potential} \rightarrow \textbf{full}$$

2.2.4 State of the art

Pervasive and mobile computing have peculiarities such as highly volatile connectivity, unexpected disconnections, location-dependency, reduced energy,

Table 2.3: Match classes

Match class	**Description**	**Semantics**
Exact	S ia semantically equivalent to D. All the requirements expressed in D are in S and S does not expose any additional feature w.r.t. D.	$\mathcal{T} \models D \equiv S$
Full	S is more specific than D. All the requirements expressed in D are provided by S and S exposes further characteristics both not required by D and not in conflict with the ones in D.	$\mathcal{T} \models S \sqsubseteq D$
Plug-in	D is more specific than S. All the characteristics expressed in S are requested by D and D exposes also other requirements both not exposed by S and not in conflict with characteristics in S.	$\mathcal{T} \models D \sqsubseteq S$
Potential	D is compatible with S. Nothing in D is logically in conflict with anything in S and vice-versa.	$\mathcal{T} \not\models S \sqcap D \sqsubseteq \bot$
Partial	D is not compatible with S. At least one requirement in D is logically in conflict with some characteristic in S.	$\mathcal{T} \models S \sqcap D \sqsubseteq \bot$

storage and computational availability among others, making them different from traditional wired and dependable computing contexts. Mobile agents endowed with quick decision support, query answering and stream reasoning capabilities are required [38, 74] supporting user activities and providing general-purpose innovative services. In these contexts, reasoning engines are exploited as decisional and organizational systems: specific non-standard inference services may be more suitable than standard ones [123]. Furthermore, mobile and embedded devices are basically resource-constrained, so they can run properly only optimized inference engines, while common reasoners generally impose non-trivial hardware and software constraints.

Due to architectural constraints and computational complexity of Description Logics reasoning, the majority of early mobile inference engines provided only rule processing for entailment materialization in a KB. Proposals include *3APL-M* [65], *COROR* [135], *MiRE4OWL* [63], *Delta-Reasoner* [83] and the system in [124], that results unsuitable to support applications requiring non-standard inference tasks and extensive reasoning over ontologies [83].

More expressive languages could be used by adapting tableaux algorithms –whose variants are implemented in reasoners running on PCs– to mobile computing platforms, but an efficient implementation of reasoning services is still an open problem. Several techniques [49] allow to increase expressiveness or decrease running time at the expense of main memory usage, which is precisely the most constrained resource in mobile systems.

Pocket KRHyper [130] was the first reasoning engine specifically designed for mobile devices. It supported the the $\mathcal{ALCHIR}+$ DL and was built as a Java ME (Micro Edition) library. Pocket KRHyper was exploited in a DL-based matchmaking framework between user profiles and descriptions of mobile resources/ services [64]. However, frequent "out of memory" errors strongly limited the size and complexity of manageable logic expressions. To overcome these constraints, tableaux optimizations to reduce memory consumption were introduced in [133] and implemented in *mTableaux*, a modified version of Java Standard Edition (SE) *Pellet* reasoner [131]. Comparative performance tests were performed on a PC, showing faster turnaround times than both unmodified Pellet and *Racer* [44] reasoner. Nevertheless, the Java SE technology is not expressly tailored to the current generation of handheld devices. In fact, other relevant inference engines cannot run on common mobile platforms, since they rely on Java class libraries incompatible with most widespread mobile OS (*e.g.*, Android). In [147] four Semantic Web reasoners were successfully ported to the Android platform (Pellet, *CB* [58], *Hermit* [126] and *JFact*, a Java port of *Fact++* [138]), albeit with significant rewriting or restructuring effort in some cases. Similarly, in [59] the *ELK* reasoner

was optimized and evaluated on Android. Nevertheless, all ported systems were designed mainly for batch jobs over large ontologies and/or expressive languages. This makes mobile device usage less suitable due to computation and memory constraints.

Furthermore, the above reasoners only support standard inference services such as satisfiability and subsumption, which are not enough for pervasive scenarios. Non-monotonic reasoning tasks are needed in order to enable the creation of software agents able to provide quick decision support and/or on-the-fly organization in such intrinsically unpredictable environments.

In latest years, the bad worst-case complexity of OWL language stimulated a different approach to implement reasoning tools. It was based on simplifying both the underlying logic languages and admitted KB axioms, so that structural algorithms could be adopted, while maintaining expressiveness enough for broad application areas. In [12], the basic $\mathcal{EL}$ DL was extended to $\mathcal{EL}^{++}$, a language deemed suitable for various applications, characterized by very large ontologies with moderate expressiveness. A structural classification algorithm was also devised, which allowed high-performance $\mathcal{EL}^{++}$ ontology classifiers such as *CEL* [13], *Snorocket* [71] and ELK [60]. OWL 2 profiles definition complies with this perspective, focusing on language subsets of practical interest for important application areas rather than on fragments with significant theoretical properties. The μOR reasoner [4] for Ambient Intelligence distinguishes itself for adopts a resolution algorithm on the *OWL-Lite*$^{-}$ language, while *LOnt* [69] works on *DL Lite*, a subset of OWL-Lite. The mobile OWL 2 RL engine in [139] exploits an optimization of the classic *RETE* algorithm for rule systems. In a parallel effort motivated by similar principles [104], an early approach was proposed to adapt non-standard logic-based inferences to pervasive computing contexts. By limiting expressiveness to the $\mathcal{AL}$ language, acyclic, structural algorithms were adopted reducing standard (*e.g.*, subsumption) and non-standard (*e.g.*, abduction and contraction) inference tasks to set-based operations [34]. KB management and reasoning were then executed through a data storage layer, based on a mobile RDBMS (Relational DBMS). Such an approach was further investigated in [101] and [106], by increasing the expressiveness to $\mathcal{ALN}$ DL and allowing larger ontologies and more complex descriptions, through the adoption of both mobile OODBMS (Object-Oriented DBMS) and performance optimized data structures. Finally, in [102] expressiveness was extended to $\mathcal{ALN}$(D) DL with fuzzy operators.

The current implementation of *Mini-ME - the Mini Matchmaking Engine* [122] provided a standards-compliant implementation of most common inferences (both standard and non-standard) for Android and Java Standard Edition (SE). More recently, a Mini-ME Swift reenginering was released for

iOS devices [109]. In this work a next-generation Mini-ME reasoning engine currently being developed in C has been integrated.

2.3 Distributed transactional systems

IoT suffers from unpredictability of node and resource availability, due to the volatility of actors and appliances. This makes trust and coordination management difficult and these limits are inevitably inherited by the SWoT: their burden is particularly evident when reliable and trustworthy applications are needed. Distributed transactional systems and specifically *blockchain* technology are interesting from this perspective. By integrating blockchain with SWoT approaches interesting possibilities can rise for large-scale distributed trustless systems. In what follows some relevant background and related state-of-the-art works are surveyed.

2.3.1 Blockchain basics

Blockchain is a data structure and protocol for *trustless* distributed transactional systems. In traditional distributed databases, a trusted intermediary is needed to guarantee irreversibility (*i.e.*, no committed transaction can be reverted or altered) and preventing censorship (*i.e.*, all valid transactions are committed). Blockchain systems avoid intermediaries by approving transactions through a distributed *consensus* protocol, which guarantees no single node or small group of colluding nodes can force the addition, removal or modification of data. The minimum percentage of colluding nodes among participating peers needed to subvert a blockchain depends on the particular consensus mechanism. Transactions approved in a given time period –again, the window size depends on the particular blockchain system– are grouped in *blocks*. As depicted in Figure 2.3, for each block a hash is appended, computed not only on the contents of the block, but also on the hash of the previous block, thus forming a chain of blocks. This prevents tampering even with old blocks without consensus among the nodes.

Building on previous theoretical results on *Proof-of-Work* consensus algorithms [143], blockchain technology was born with *Bitcoin*, an open source platform for electronic currency. Bitcoin uses blockchain as a ledger to store currency transfer transactions. After the success of Bitcoin, many other blockchain-based electronic currency platforms have been introduced. At the same time, it has been realized that the underlying blockchain technology is an inherently general-purpose distributed database, enabling trustless

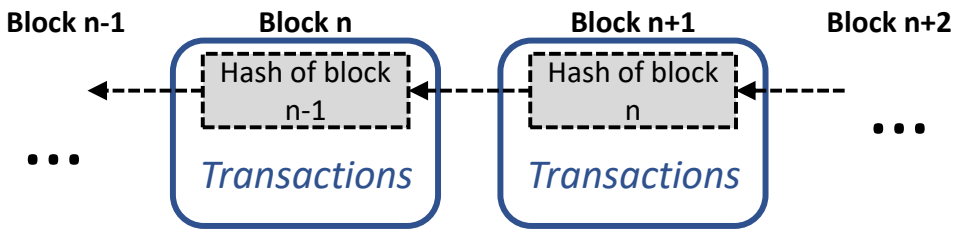


Figure 2.3: Blockchain structure

collaboration of *Decentralized Autonomous Organizations* (DAOs). This feature enables practical implementations of the *Smart Contract* (SC) idea [134], *i.e.*, programs encoding and enforcing cooperative processes among two or more parties. Originally, SCs required a trusted mediator, restraining a large development of the approach. Indeed, consensus about SCs in a blockchain is reached through a parallel execution in the network, effectively making every SC-enabled blockchain a general-purpose application platform based on a distributed virtual machine (VM). Many proposals have emerged, including proprietary platforms (*Ethereum*[3] is perhaps the most popular) and standardization efforts, such as the *Hyperledger*[4] initiative guided by the Linux Foundation. SC-based blockchains are being experienced in several financial and industry sectors.

Several types of blockchain systems exist, based on the following key design decisions:

- *Access policy.* A blockchain network is *permissionless* (*a.k.a. public*) if any node can join the consensus protocol –even anonymously– at any time, or *permissioned* (*a.k.a. private*) if a white-list of admitted nodes exists and nodes are uniquely identified. This choice has a deep impact on the blockchain design: permission-less chains usually have to reward participants for their computational effort, *e.g.*, Bitcoin allows nodes to generate (*mine*) and keep new currency for the validation of transaction blocks. Permissioned chains are instead adopted in more controlled collaboration contexts, where access itself is a reward, as it enables selling and buying services or resources. *Consortium* blockchains follow an intermediate model, where all nodes can use the blockchain but only a particular subset is able to validate transactions through consensus.

- *Consensus algorithm.* Permission-less systems require stricter consensus methods, such as Proof-of-Work, which guarantee data security

[3]Ethereum Project: `https://www.ethereum.org/`
[4]Hyperledger: `https://www.hyperledger.org/`

Table 2.4: Typical blockchain features for e-currency and IoT solutions

Feature	**E-currency**	**IoT**
Access policy	Permission-less	Permissioned
Consensus algorithm	Proof-of-Work	BFT-like
Transaction model	UTXO	Account-based
Smart contracts	No	Yes
Programmable	No	Yes
General purpose	No	Yes
Transaction latency	Lower	Higher

unless a large portion of nodes is colluding to subvert the blockchain. Permissioned systems –where each node is identifiable and accountable– may relax consensus constraints in order to reduce the computational load, by selecting simpler algorithms; Byzantine Fault Tolerance (BFT) variants [143] are often adopted.

- *Transaction model.* In blockchain systems, *assets* can be registered and exchanged. At any time, each node typically owns some assets in a certain quantity. In the *unspent transaction outputs* (UTXO) model, a transfer from A to B is modeled as *consuming* (*i.e.*, deleting) records for A's spent assets and *producing* (*i.e.*, adding) new ones for B's received assets. In the *account-based* model, instead, every node has an account reporting all its assets, which is updated by transactions. The UTXO model is similar to a bank statement of account; it allows simpler reconstruction of current state from a transaction log and is typically adopted by e-currency systems. The account-based model is more general, but it can make transaction processing slower; nevertheless, it is the only practical choice for general-purpose SC-based blockchains.

- *Smart contract language.* Blockchains can adopt any formalism for SC specification and execution, such as procedural (imperative) languages or logical (declarative) languages or automata [53]. Industry proposals mostly adopt computationally complete programming languages, either existing or created for the purpose (*e.g.*, Ethereum's *Solidity*).

Table 2.4 summarizes blockchain features in typical e-currency and IoT solutions. A wider discussion of blockchain technology is *e.g.*, in [27].

2.3.2 State of the art

The transparent trustless peer-to-peer models enabled by blockchain technologies are emerging as a viable path for the current and future expansion of reliable Internet of Things networks and application [96]. Several blockchain-based proposals, with various levels of maturity, already exist in smart manufacturing, smart energy, smart home, smart city and information-intensive marketplaces [91, 99]. Emerging distributed file systems, billing services and other blockchain-based tools can be leveraged as an application-agnostic machine-to-machine middleware layer for running IoT resource/service marketplaces with minimal or no human intervention [27].

Due to its standards-based approach and economic relevance, *Industry 4.0* (I4.0) is a prominent blockchain use case [37]. Asset tracking and supply chain are among the most popular applications, due to the widely recognized benefits of trustless DAO collaboration [67, 81, 82] and the easy fit of blockchain solutions in existing industry standards for information-sharing distributed infrastructures, most notably the Electronic Product Code Information Services (EPCIS) [61]. The simplest approaches rely on transactional ledgers for asset transfer, which grant high troughput with low costs [27]. Blockchain networks based on SCs enable more flexible systems, allowing any application logic to be implemented and embedded in the blockchain [27], and also supporting discoverable, composable and verifiable multi-step business processes in multi-party service-oriented architectures (SOA) [87]. The Reference Architecture Model Industry 4.0 (RAMI 4.0) [54] specification outlines a SOA for facilitating cross-organizational interoperability and cooperation along the full lifecycle of objects and processes in industrial cyber-physical systems. It can exploit existing results on service discovery and composition, like those demonstrated by case studies in [17, 113].

Unfortunately, the benefits of smart contracts come at a not-negligible cost in terms of concurrent execution of transactions and, consequently, system throughput [27]. This occurs because in the general case, before executing a smart contract, a node cannot know what computing resources it will need –even possibly including other smart contracts– and what its effects will be on the system state. That makes it impossible to run all transactions in a block in parallel. In Bitcoin-style asset transfer blockchains, on the contrary, transaction have a fixed inherent semantics: conditions for transaction dependencies and ordering are simpler and known in advance, leading to the possibility to execute (usually most of) the transactions in a block concurrently and thus achieve higher throughput and scalability. This is a significant barrier to advanced blockchain applications in the IoT, which require freely specifiable business logic and low-latency high-throughput trans-

actions at the same time. Research on blockchain scalability is very active, mainly by optimizing performance of consensus protocols [143] and by introducting parallelism in a blockchain through *sidechains* and/or *sharding* [29]. Basically, the use of sidechains will transform the chain structure in a direct acyclic graph. On the other hand, sharding is a parallelization technique borrowed from Database Management Systems, consisting in splitting data elements (*e.g.*, rows in relational databases) horizontally across node subsets in a cluster. Research results, however, are not mature enough [144] for building efficient, robust, large-scale IoT-oriented blockchains. In this respect, the approach proposed in this work aimed to mitigate scalability issues by providing a semantic-enabled SOA above contract-based blockchains: while enabling general-purpose service/resource discovery and retrieval, this layer has a finite set of smart contracts as primitives, which do not have recursive calls.

Many opportunities exist for exploiting KRR technologies in blockchains. In [37] a prototypical ontology was proposed to annotate transactions with Linked Data [46] in order to make contents of the blockchain easier to explore for humans through semantic-enabled user agents. The ontology-based smart contract design of a proof-of-concept blockchain system in [62] enabled traceability in supply chains. Besides acting as design guidelines, logical languages can be used to specify formally and execute smart contracts. Several logical frameworks have been applied to the problem of SC specification and execution. The work in [53] used *defeasible reasoning*, a well-known approach to formalize legal regulations and contracts. On the other hand, [50] endorsed the usage of *Linear Temporal Logic* (LTL), which is implemented in a large number of model checking systems. This allows formal verification that the behavior of a SC satisfies specific conditions. The need for formal verification of SCs in business-oriented blockchains was also highlighted in [87]. In [113] the first semantic-based discovery approach for blockchain systems was proposed. However, that approach presents some limitations mainly due to the adopted blockchain framework Hyperledger *Iroha*[5]. Analogously, in [137] a blockchain-based framework uses smart contracts to mediate robot coalition formation, where both robot sensors/actuators and environmental parameters are exposed as resources annotated w.r.t. an ontology; implementation details and experimental results are not available yet.

The proposal detailed in Chapter 5 improves the previous work [113] by exploiting the more IoT-oriented blockchain substratum Hyperledger *Sawtooth*[6] blockchain framework. Sawtooth design is particularly suitable for

[5]Hyperledger Iroha: `https://www.hyperledger.org/projects/iroha`
[6]Hyperledger Sawtooth: `https://www.hyperledger.org/projects/sawtooth`

IoT scenarios, due to the high decoupling between components, which allows to distribute request management, transaction processing and validation on different kinds of devices. Furthermore, Sawtooth adopts the *Proof of Elapsed Time* (PoET) consensus protocol, which requires lower computational and energy resources than PoW and most BFT-like protocols [91].

2.4 Current issues and limitations

Basically, operations of context-aware systems (CAS) are arranged according a so-called *context information life cycle*, defining where information is produced and where it is consumed [1]. As Figure 2.4 shows, it comprises four stages:

- *Context acquisition*: perception components (*e.g.*, embedded sensors, network interfaces) collect low-level data from the cyber-physical environment: data is possibly preprocessed and/or stored to make it available to further processing;
- *Context modeling*: this step concerns multi-sensor fusion and semantic-based annotation of perceived data w.r.t. reference domain ontologies, in order to obtain a formal model of the perceived context;
- *Context reasoning*: inference procedures to derive implicit knowledge from available annotations are executed at this stage to achieve a better understanding of the context and enable decision-making.
- *Context distribution*: at this stage, context knowledge is shared; different dissemination models exist, which can be basically classified in *pull*-based (*e.g.*, *request/response*) and *push*-based (*e.g.*, *publish/subscribe*).

The research line investigated in this thesis concerns the integration of intelligence in micro-devices deployed in pervasive environments and wirelessly interconnected within unstructured settings in MANETs. Accordingly to the CAS general framework, cognitive *smart objects* should have the ability to:

- automatically extract and process environmental data;
- generate semantically rich and compact descriptions about themselves and the context they operate in;
- reach a steady state with a coherent situation awareness by integrating high-level self-detected and received information in a micro-*log*;

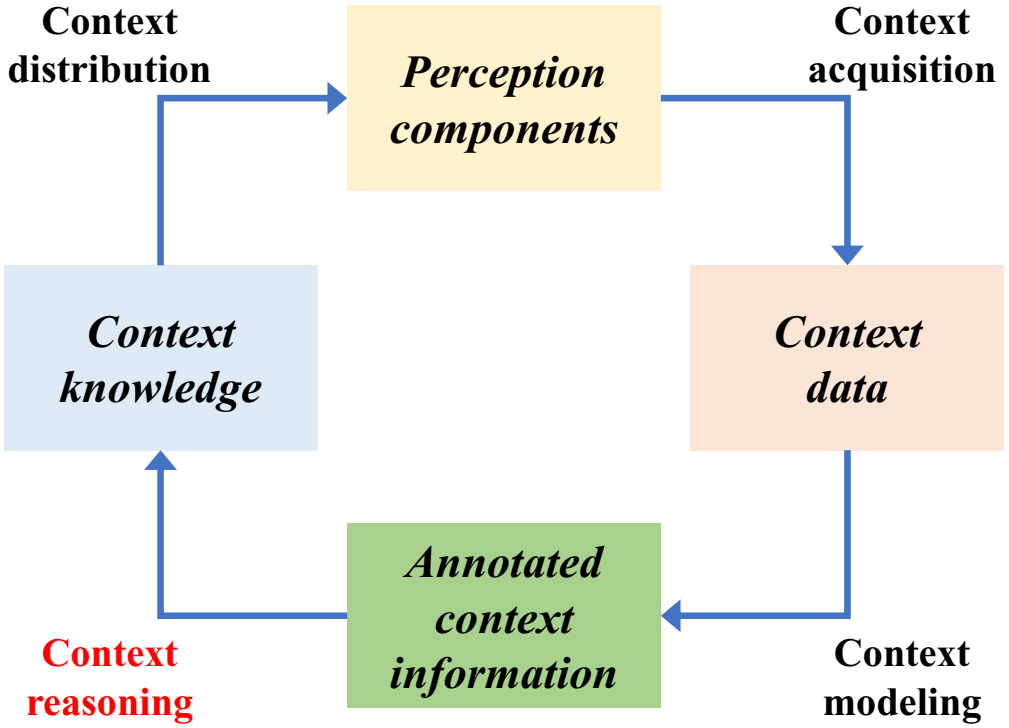


Figure 2.4: Context information life cycle [1]

- make the gained knowledge easily accessible by other network nodes through publishing it to a micro-*blog*.

Objects behave like autonomous *social agents*, interacting and coordinating automatically as new information about the environment is available. Pervasive scenarios deal with heterogeneous mobile micro-devices: what one agent knows about the environment, based on perception components it uses or has access to, can be significantly different from another agent's knowledge. Furthermore, uncertain conditions governing sensor-originated data, such as unreliable sensor technology or frequent wireless communication disconnections, require agents to cope with imposed incomplete or incoherent information. Agents evidently hold a partial view of the environment and wish to obtain a more reliable and complete context *snapshot*, by also considering other agents' perspectives. Enabling such *fusion* capability in a smart object calls for an ability to merge ontology-referred self-generated and received context annotations by dealing with resolution of inconsistencies and resilience against spurious information. By doing so, a smart object becomes able to identify on-the-fly the task(s) needed to change its own configuration

or the environment state and automatically infer what useful capabilities it can provide or it needs from other entities in order to act accordingly, in a decentralized and collaborative fashion. The history and the inferred context is published in a mobile ad-hoc network –and on the Web, if Internet connection is available– in order to foster intelligent cooperation.

A basic distinction in literature exists between High-Level Information Fusion (HLIF) and Low-Level Information Fusion (LLIF) [18]. Both types deal with error reconciliation and information enrichment, but at two fundamentally distinct levels and with different tools and techniques. In detail, the above *Context modeling* step exploits LLIF, while *Context reasoning* is strictly related to HLIF.

LLIF deals with numerical data collected from physical phenomena and it is typical of multi-sensor fusion for identification, tracking and classification applications. In this area, *aggregation functions* [21, 66, 23] provide a way to fuse measurements of individual devices and to reduce the size of exchanged data. Similarly, in [23] a cooperative surveillance MAS allows coalitions of agents with overlapping fields of view. After receiving data converted in a common reference system, a local fusion agent can track moving objects and also identify inconsistencies within the coalition. *Distributed particle filtering* [47] is one of the most popular and versatile techniques for environment state estimation, especially suited to large-scale, nonlinear systems. The subfamily of *Aggregation Chain* methods has two similarities with the approach proposed here: (i) only estimated probability distributions are exchanged, not raw measurements and (ii) probability estimations are merged iteratively, in a chain. Nevertheless, the purely statistical nature of the method makes it quite different from semantic-based ones, since managed data have no high-level formal meaning.

Conversely, HLIF processes abstract symbolic information and it is generally adopted for knowledge refinement and situation awareness.

The *fusion* framework proposed in Chapter 3 refers specifically to HLIF: according to the classification of knowledge fusion patterns in [132], it can be described as *"fusing knowledge of knowledge workers and knowledge accumulated in repositories"*, since the semantic annotations produced by independent pervasive smart objects is aggregated and reconciled with respect to clashes detected, leveraging a core of domain knowledge encoded in a reference ontology. Therefore sensor data fusion is beyond the scope of the thesis, supposing smart agents are equipped with on-board sensing and data mining capabilities for LLIF to produce logic-based annotated descriptions of their perceptions, like *e.g.*, in [110, 116].

Semantic-based information fusion relies on operators and inference services on formulas expressed in a logical language. Epistemic logic was adopted

for peer-to-peer (P2P) information integration in [20], which however focused on schema mappings for queries on a heterogeneous P2P system as a single entity, rather than distributed intelligence. The belief merging framework in [24] has theoretical similarities with the approach proposed in this work. In particular, both works exploit the so-called *weak disagreement* –*i.e.*, information neither confirmed nor contradicted by different agents– in order to enrich knowledge toward a better overall truth approximation. However, [24] focused on majority agreement among several peers, whereas the work proposed here aims to eliminate explicit disagreements rather than to reach a majority agreement. For this reason, it considers all arguments as having equal importance, which is typical of logical *arbitration* [76], unlike methods in the area of *belief revision* [41], where latest arguments are considered as more important. An example of the latter type is in [112], which is nonetheless a decentralized iterative knowledge fusion approach like the one proposed here.

Data management and analytics in the IoT are surveyed in [2]. In latest years the cloud-based computation centralization trends have been shifting toward more distributed paradigms like *Fog Computing* [33] and *Edge Computing* [128], which move storage and processing closer to data sources. The proposed approach is oriented to these architectures, as they enable lower processing latency, higher scalability and better privacy, particularly in very large application scenarios. In these contexts, *semantic data aggregation* often refers to approximate techniques or lossy compression to summarize data from multiple sources in the IoT and wireless sensor network literature. Some works, however, exploit semantic-based knowledge representation formalisms to improve not only queries, but also data aggregation [117, 5].

Relevant applications of semantic-based information fusion include ambient intelligence [88], home and building automation [114] wireless sensor networks [31] and vehicular networks [112]. For these scenarios, other popular approaches are based on Bayesian [100] or Dempster-Shafer (*a.k.a.* belief function) theory. Unfortunately, they cannot be integrated easily with semantically explicit formalisms for context modeling. A relevant attempt is found in [43] for situation awareness in vehicular networks, adopting Multi-Entity Bayesian Networks, which extend entities in a Bayesian network with semantic relationships, in order to improve probabilistic event evaluation.

The HLIF proposal of this work was devised starting from the knowledge-fusion approach in [112]. The designed *Concept Integration* (CI) inference service aims to merge perspectives of different independent entities into a single context observation. The individual knowledge is the result of a local CI process on detected and received semantic annotations about the current environment state. It is structured as a 4-tuple of annotations, each providing

a particular point of view on the context:

- ***C*** *(Confirmed)*: elements detected in both the local and other agents' observations;
- ***X*** *(Clash)*: local observation, not consistent with observations of others;
- ***M*** *(My)*: elements detected in the local observation, but not by other agents;
- ***E*** *(External)*: elements observed by other agents, but not locally.

This approach is suitable for MAS like vehicular networks, featured by mostly homogeneous mobile agents with relatively ample computational resources. The goal of this work is to provide a more simplified knowledge management approach, which can be both general-purpose architecture-wise and oriented toward large-scale IoT scenarios and/or resource-constrained devices performance-wise.

Semantics-based approaches enable interoperability between heterogeneous devices, allowing high-level situation description, assessment and inference. Nevertheless, trust and reliability issues remain basically unsolved. Decentralized and dynamic IoT infrastructures suffer from the unpredictability of nodes ensuing from the volatility of actors and appliances. Scalability is a further problematic aspect to be faced on when trust and coordination among smart objects are needed. From these standpoints, blockchain technology could incorporate SWoT approaches providing an interesting potential.

Chapter 3

Object (b)logging: framework, technology, implementation

This chapter describes in detail the paradigms, architectures and methods defined and developed for the object (b)logging vision. Each task of the proposed framework is analyzed in order to provide a clear picture of the whole approach designed for high-level descriptions of the environment and the object itself, knowledge discovery, fusion and sharing in distributed scenarios populated by smart objects. In order to assess the benefits of the proposed framework, a software prototype has been implemented and performance evaluations were carried out with reference to a case study in the smart agriculture field.

3.1 Architecture

The object (b)logging vision relies on both ideas and technologies of distributed knowledge-based systems [108] discussed in Section 2.2.1, whose individuals (*a.k.a.* assertional knowledge) are physically tied to objects in a given environment, not requiring centralized coordination. In the above SWoT vision, object (b)logging provides means to make an object capable of sensing the environment and detecting events, describing itself and what surrounds it in a fully automatic way, finally exposing its descriptions to the outside world and acting appropriately in cooperation with other objects deployed in the environment.

The paradigm is based on the design, development and optimization of knowledge representation and reasoning techniques in pervasive contexts in order to build and exchange a structured and formal representation of an

environment, process, subject or any other entity of interest in a cyber-physical system. The overall goal is to enable the development of autonomous smart objects complying with the definition in Section 2.1.1: they are capable to produce and annotate high-level descriptions about themselves and the environment they are located in –*i.e.*, *Context Information*– as in a micro-*log*). Each annotation refers to an ontology providing the conceptualization for the particular domain. According to this vision, smart objects are able to continuously enrich an initial logical *descriptive core*, which models ground knowledge about their own features and capabilities as well as about concepts and relationships with general validity in the domain. Annotations evolve during the object's lifetime and are exposed toward external devices and applications in a self-contained fashion like in a micro-*blog*.

In detail, the proposed approach envisions an object log as the integrated and comprehensive view of all the information a smart object has learned first-hand or received from peers. Exploiting the log, a smart object is able to identify on the fly the available *Capabilities* according to the task(s) needed to change its own configuration or to act on the environment in order to reach the desired state *i.e.*, *Constraints*. More formally, a log is composed by the semantic annotations referred to a domain ontology progressively enriched during the object's lifetime. It is based on the logical descriptive core and is used for intelligent interpretation of retrieved information. The *Concept Fusion* approach described in Section 3.2 is exploited in order to dynamically update the log by merging partial perspectives of different nodes into a single high-level situation description.

Objects are equipped with a reasoning engine performing automated inference procedures to derive implicit knowledge, which is used to trigger actions, take decisions and make interventions on the environment by leveraging semantic-based matchmaking between available effectors (*i.e.*, services in what follows) and required goals. Each actuation capability is envisioned in term of semantically annotated *preconditions* to take into account for the activation and *effects* produced. A service is considered *available* only if it could be correctly activated according to the environment and device state, *i.e.*, its preconditions are satisfied. The smart object's KB encapsulates also its behavior in the form of *trigger* conditions, which require specific *actions* on the environment, as per the classical *Behaviour Profile* interpretation outlined in Section 2.1.1.

The object (b)logging activity is a continuous process, composed of the main steps depicted in Figure 3.1 in reference to the precision agriculture case study scenario:

1. When a smart object completes data gathering at the end of each time

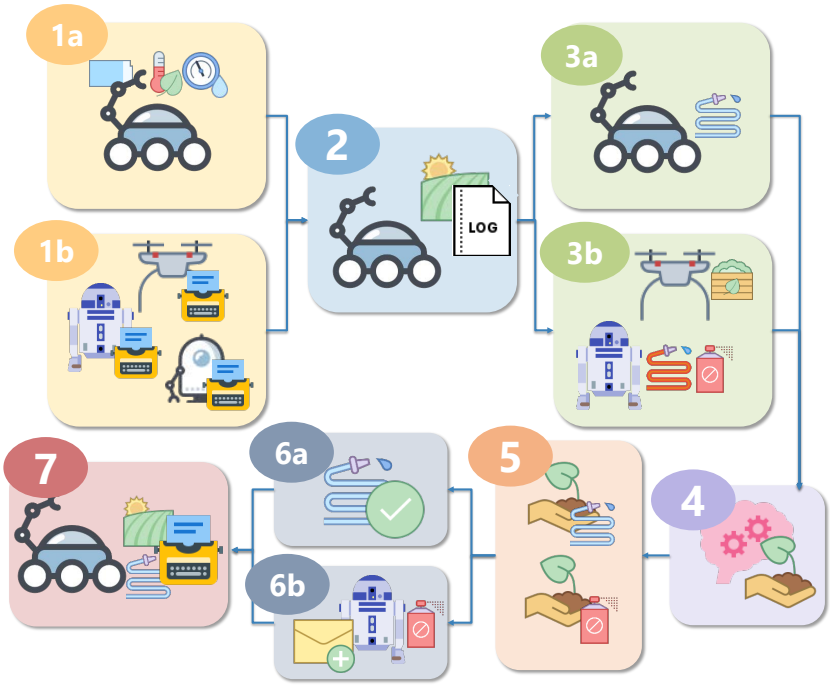


Figure 3.1: Architecture of the object (b)logging paradigm

window, (1a) it generates a semantic annotation, representing fresh detected information about the context it is in and its internal state while (1b) incoming "blogged" packets from other entities are cached;

2. Each agent fuses locally gathered knowledge with the one collected from nearby nodes in the last time window, eliminating inconsistencies. A semantic-based coherent view is obtained, which is reported in the log along with the internal status of the device;

3. The smart object infers and caches available services provided (3a) by itself –according to their activation preconditions and the detected state and context– and (3b) by others as stated in shared blogs;

4. According to the detected event and conditions, decisions about the needed intervention on the environment are taken.

5. The best available cached services are selected;

6. Each agent (6a) activates the selected services directly provided, and (6b) asks for the activation of the ones exposed by peers;

7. Machine-understandable knowledge and provided capabilities are shared through blog.

Further details about each step are provided in Section 3.3. The key aspect of object (b)logging is sharing the information possibly useful to nearby objects through the *blog*. Autonomous objects cooperation is achieved: all actors become able to exchange information, discover new services, start new acquaintances, connect to external services, share knowledge, exploit other objects' capabilities opportunistically and collaborate toward a common goal in a decentralized way.

It can be noticed this form of cooperation requires preliminary ontology agreement among participating agents. Since the problem has been tackled in literature [107], it is deemed as outside the scope of the thesis work.

3.2 Knowledge fusion in pervasive contexts

Considering n collected observations $C_1, C_2, \ldots C_n, \quad n \geq 1$ satisfiable in $\mathcal{ALN}$ w.r.t. acyclic TBox $\mathcal{T}$, the basic idea behind the newly proposed *Concept Fusion* inference service is to eliminate every concept component in any C_i which clashes with another concept component in a C_j. In an adaptation of logical arbitration to DLs this is accomplished by adopting the same principles as non-standard, non-monotonic *Concept Contraction* inference service in Section 2.2.3, but implemented in an innovative and more efficient way. The conjunction of the remaining concept components is taken as the result. The approach is grounded on the above *Open World Assumption* (OWA), which is well suited for pervasive computing multi-agent CASs featured by volatile nodes and incomplete information. Furthermore the Concept Component definition recalled in 2.2.3 is exploited. *Concept Fusion* inference service is formally defined as follows:

Definition 2 ***(Concept Fusion Problem)*** *Let $\mathcal{L}$ be a DL, $\mathcal{T}$ a set of axioms in $\mathcal{L}$, and $C_1, C_2, \ldots C_n$ n concept descriptions (with $n \geq 2$) satisfiable in $\mathcal{L}$. A* ***Concept Fusion Problem*** *(CFP) for given $\langle \mathcal{L}, \mathcal{T}, (C_1, C_2, \ldots C_n) \rangle$ is finding a concept $F \in \mathcal{L}$ such that $F \equiv \bigsqcap_{i=1}^{n} K_i$ where $\mathcal{T} \models F \not\sqsubseteq \bot$ and $\forall i = 1, \ldots, n : \mathcal{T} \models C_i \equiv G_i \sqcap K_i$. F is called a fusion of $C_1, C_2, \ldots C_n$ according to $\mathcal{T}$.*

The CFP can be deemed as an extension of the Concept Contraction Problem in Section 2.2.3 to n concept descriptions; G_i and K_i represent the Give-up

and Keep parts of C_i, respectively. Informally, Concept Fusion drops every concept component in any C_i which clashes with another concept component in a C_j; the conjunction of the remaining concept components is taken as the result F. Several remarks apply to CFP as well as to CCP.

Remark 1. Intuitively, each G_i represents "why" C_i is not compatible with the conjunction of the other $n-1$ concept descriptions, thus providing an explanation of the unsatisfiability of the conjunction of all collected knowledge.

Remark 2. The definition rules out having any unsatisfiable C_i, because that would lead to counterintuitive results [34].

Remark 3. In general, several solutions exist to a CFP. The trivial solution $\langle G_1, G_2, \ldots, G_n, K_1, K_2, \ldots K_n\rangle = \langle C_1, C_2, \ldots, C_n, \top_1, \top_2, \ldots \top_n\rangle$ is always possible, corresponding to giving up everything. Conversely, if $\bigsqcap_{i=1}^{n} C_i$ is satisfiable in $\mathcal{T}$, the most desirable solution has $\langle G_1, G_2, \ldots, G_n, K_1, K_2, \ldots K_n\rangle = \langle \top_1, \top_2, \ldots \top_n, C_1, C_2, \ldots, C_n\rangle$, *i.e.*, keep everything.

Remark 4. Since one wants to merge as much knowledge as possible, a maximality criterion for selecting the best solution to a CFP must be defined. For DLs admitting a CNF, a conjunction-minimality criterion can be adopted on G_i as defined in [34].

Nevertheless, there are also significant differences between CFP and CCP.

Remark 5. The order of concept descriptions C_i is basically irrelevant in a CFP, *i.e.*, a permutation of C_i would yield the corresponding permutation of G_i and K_i concepts and the solution F would not change due to the commutativity of conjunction. Conversely, CCP is asymmetrical and not commutative, as it retracts a concept from one of the two input descriptions w.r.t. the other; switching the inputs would change the outcome. This is why CFP is akin to logical arbitration while CCP belongs to belief revision algorithms.

Remark 6. CCP is defined only for incompatible (*i.e.*, clashing) concept descriptions, because otherwise it would yield counterintuitive results. On the other hand, a set of n descriptions all compatible with each other is supported by the definition and the implementation of CFP, yielding F consisting in the conjunction of all descriptions.

Furthermore, Concept Fusion is different from *Concept Integration* [112] –introduced in Section 2.4– in two aspects.

Remark 7. While the repeated application of Concept Integration allows merging knowledge with *reconciliation* of inconsistencies [112], Concept Fusion just grants the *elimination* of inconsistencies. This is a simplified approach, aimed specifically at general-purpose large-scale deployments of very resource-constrained IoT nodes.

Remark 8. Differently from Concept Integration, a straightforward applica-

Require: DL $\mathcal{L} = \mathcal{ALN}$, acyclic TBox $\mathcal{T}$, $C_1, \ldots, C_n$ $n \geq 1$ concept descriptions in $\mathcal{L}$ satisfiable w.r.t. $\mathcal{T}$
Ensure : F concept description satisfiable in $\mathcal{L}$ w.r.t. $\mathcal{T}$, $\mathcal{G}$ cache of discarded concept components

$F := \top$;
$\mathcal{G} := (\mathcal{G}_{CN}, \mathcal{G}_{\leq}, \mathcal{G}_{\geq}, \mathcal{G}_{\forall})$;
$\mathcal{G}_{CN} := \emptyset; \quad \mathcal{G}_{\leq} := \emptyset; \quad \mathcal{G}_{\geq} := \emptyset; \quad \mathcal{G}_{\forall} := \emptyset$;
$\langle F, \mathcal{G} \rangle := merge((C_1, \ldots, C_n), F, \mathcal{G})$;
return $\langle F, \mathcal{G} \rangle$

Algorithm 1: $\langle F, \mathcal{G} \rangle = fusion(\mathcal{L}, \mathcal{T}, (C_1, \ldots, C_n))$

Require: $C_1, \ldots, C_n$, F $(n \geq 1)$ concept descriptions in $\mathcal{ALN}$satisfiable w.r.t. $\mathcal{T}$, and $\mathcal{G} = (\mathcal{G}_{CN}, \mathcal{G}_{\leq}, \mathcal{G}_{\geq}, \mathcal{G}_{\forall})$ 4-tuple of sets of concept components in $\mathcal{ALN}$
Ensure : F updated concept description, $\mathcal{G}$ updated cache of discarded concept components

for $i := 1$ *to* n **do**
  $processConceptNames(C_{i,CN}, F, \mathcal{G})$;
  $processLessThanRestrictions(C_{i,\leq}, F, \mathcal{G})$;
  $processLessGreaterRestrictions(C_{i,\geq}, F, \mathcal{G})$;
end
$F := normalize(F)$;
for $i := 1$ *to* n **do**
  $processUniversalRestrictions(C_{i,\forall}, F, \mathcal{G})$;
end

Algorithm 2: $merge((C_1, \ldots, C_n), F, \mathcal{G})$

tion of Concept Fusion in a multi-agent CAS regards information produced by the local and remote agents in the same way. In realistic scenarios –where a more fine-grained information categorization is needed– a hybrid approach is possible: the object (b)logging overall framework proposed in this work applies Concept Fusion only to semantic annotations received from remote agents, and finally an enhanced version of Concept Integration is used to merge outside knowledge with endogenous one preserving the different viewpoints.

Within the scope of the $\mathcal{ALN}$ DL, the Concept Fusion inference service is implemented by the recursive Algorithm 1 and its subroutines. Throughout computation, F contains the current state of the fusion of concept descriptions $C_1, C_2, \ldots C_n$. Additionally, $\mathcal{G}$ is a cache collecting all given-up concept

Require: $C_{i,CN}$ conjunction of (possibly negated) concept names in TBox $\mathcal{T}$,
F concept description satisfiable in $\mathcal{ALN}$ w.r.t. $\mathcal{T}$, and
$\mathcal{G} = \{\mathcal{G}_{\mathcal{CN}},\ \mathcal{G}_{\leq},\ \mathcal{G}_{\geq},\ \mathcal{G}_{\forall}\}$ 4-tuple of sets of concept components in $\mathcal{ALN}$
Ensure : F and $\mathcal{G}$ are properly updated

```
1 foreach (possibly negated) concept name CN in C_{i,CN} do
2     if there exists ¬CN in F then
3         remove ¬CN from F ;
4         G_CN := G_CN ∪ {CN, ¬CN} ;
5     else if {CN} ∉ G_CN then
6         F := F ⊓ CN ;
7     end
8 end
```

Algorithm 3: $processConceptNames(C_{i,CN},\ F,\ \mathcal{G})$

Require: $C_{i,\leq}$ set of maximum cardinality role restrictions in TBox $\mathcal{T}$,
F concept description satisfiable in $\mathcal{ALN}$ w.r.t. $\mathcal{T}$, and
$\mathcal{G} = \{\mathcal{G}_{\mathcal{CN}},\ \mathcal{G}_{\leq},\ \mathcal{G}_{\geq},\ \mathcal{G}_{\forall}\}$ 4-tuple of sets of concept components in $\mathcal{ALN}$
Ensure : F and $\mathcal{G}$ are properly updated

```
1  foreach ≤ x R in C_{i,≤} do
2      clash := false ;
3      foreach ≥ y R in F with y ≥ x + 1 do
4          clash := true ;
5          remove ≥ y R from F ;
6          addGreaterThanRestriction(≥ y R, G) ;
7      end
8      if clash = true or ({≥ y R} ∈ G_≥ and y ≥ x + 1) then
9          addLessThanRestriction(≤ x R, G) ;
10     else
11         F := F ⊓ ≤ x R ;
12     end
13 end
```

Algorithm 4: $processLessThanRestrictions(C_{i,\leq},\ F,\ \mathcal{G})$

Require: $C_{i,\leq}$ set of maximum cardinality role restrictions in TBox $\mathcal{T}$,
F concept description satisfiable in $\mathcal{ALN}$ w.r.t. $\mathcal{T}$, and
$\mathcal{G} = \{\mathcal{G}_{\mathcal{CN}}, \mathcal{G}_{\leq}, \mathcal{G}_{\geq}, \mathcal{G}_{\forall}\}$ 4-tuple of sets of concept components in $\mathcal{ALN}$
Ensure : F and $\mathcal{G}$ are properly updated

```
1  foreach ≥ x R in C_{i,≥} do
2  |  clash := false ;
3  |  foreach ≤ y R in F with y ≤ x − 1 do
4  |  |  clash := true ;
5  |  |  remove ≤ y R from F ;
6  |  |  addLessThanRestriction(≤ y R, G) ;
7  |  end
8  |  if clash = true or ({≤ y R} ∈ G_≤ and y ≤ x − 1) then
9  |  |  addGreaterThanRestriction(≥ x R, G) ;
10 |  else
11 |  |  F := F ⊓ ≥ x R ;
12 |  end
13 end
```

Algorithm 5: $processGreaterThanRestrictions(C_{i,\geq},\ F,\ \mathcal{G})$

components: it is necessary to track them in the main loop of the algorithm, in order to avoid inserting in F concept components which have been already detected as clashing in previous iterations. It is important to note that $\mathcal{G}$ is not a concept description but just a 4-tuple of sets of concept components, where conflicting elements can coexist (*e.g.*, $\mathcal{G}$ can contain both A and $\neg A$). On the other hand, $\mathcal{G}$ does not contain duplicates by construction. After F and $\mathcal{G}$ are initialized in lines 1-3, they are progressively updated in the *merge* subprocedure in Algorithm 2 for each C_i. Finally, $\mathcal{G}$ is returned alongside F to allow keeping track of discarded concepts. This may be useful for exploiting the cache in further inferences if Concept Fusion is used as a service in larger knowledge management frameworks.

The detection of semantic inconsistencies exploits basic concept clash checks, in the same way as Concept Contraction $\mathcal{ALN}$ implementations [122], but rearranged for greater efficiency when managing multiple concept descriptions. The algorithm further maintains concept components in F and $\mathcal{G}$ normalized in CNF to avoid redundancies. The recursive *merge* procedure in Algorithm 2 performs knowledge fusion. For each C_i, $i = 1, \ldots, n$, lines 1-5 compute the fusion of (possibly negated) atomic concepts, less-than ($\leq$) and greater-than ($\geq$) number restrictions, by calling the subroutines explained

hereafter.

The *processConceptNames* procedure works on $C_{i,CN}$ as outlined in Algorithm 3: for each (possibly negated) concept name CN, the algorithm checks whether its negation is in F: in that case a clash occurs and both CN and $\neg CN$ must be added to the cache $\mathcal{G}_{CN}$ of discarded concept components (lines 2-4); otherwise, the absence of CN from $\mathcal{G}_{CN}$ must be checked, and in that case CN can be added to $\mathcal{F}_{CN}$ (lines 5-7).

The *processLessThanRestrictions* and *processGreaterThanRestrictions* procedures, specified in Algorithms 4 and 5 respectively, work in perfectly dual fashion, therefore just the former is explained here. If $C_{i,\leq}$ contains a less-than restriction on a role R ($\leq x\ R$, line 1), a clash occurs with every $\geq$ number restriction on R already in F with a disjoint number interval (lines 3-4): in that case, both number restrictions must be moved into $\mathcal{G}$ (lines 5-9). If a clash is found with an element of $\mathcal{G}_{\geq}$, then $\leq x\ R$ is added to $\mathcal{G}$ (lines 8-9). Only if no clash occurs, $\leq x\ R$ can be safely added to F (lines 10-11). Subprocedures *addLessThanRestriction* and *addGreaterThanRestriction*, reported in the Appendix A, are called to insert number restrictions into $\mathcal{G}$ in an optimized fashion: they work like CNF normalization, by keeping no more than one number restriction per type per role (the minimum and maximum value for $\leq$ and $\geq$ restrictions, respectively). This minimizes the number of checks on $\mathcal{G}_{\leq}$ and $\mathcal{G}_{\geq}$ in further iterations of the main loop in Algorithm 2. Furthermore, it must be noticed that at this stage F must not be normalized in CNF yet, in order to store all compatible number restrictions (not just one per type per role), in order to discard all and only the ones that are found clashing with other concept components: this approach ensures processing order independence of C_i concept descriptions. F is normalized just once (Algorithm 2, line 6), before processing universal restrictions; the classical CNF normalization algorithm is in [106].

Lines 7-9 in Algorithm 2 loop over universal restrictions, which must be processed after number restrictions in order to take into account the interplay between the two types of constructors in the detection of clashes. The *processUniversalRestriction* subroutine, outlined in Algorithm 6, first checks whether the partially merged F contains $\leq 0\ R$ and $\forall R.\bot$ for a given role R: this is typically a consequence of the CNF normalization of an unsatisfiable role filler, and in this case any further universal restriction on R found in the C_i descriptions becomes irrelevant (lines 2-3). Otherwise, if either F or $\mathcal{G}$ contain a minimum cardinality restriction with value at least 1 on R, a recursive *merge* is computed on the fillers of the universal restriction in C_i and the possibly pre-existent one in F, taking into account previously discarded concepts in $\mathcal{G}$: lines 4-10 do this if a universal restriction already exists in F (updating F and $\mathcal{G}$ accordingly in lines 11-14), otherwise lines 15-17 check

Require: $C_{i,\leq}$ set of maximum cardinality role restrictions in TBox $\mathcal{T}$,
F concept description satisfiable in $\mathcal{ALN}$ w.r.t. $\mathcal{T}$, and
$\mathcal{G} = \{\mathcal{G}_{\mathcal{CN}}, \mathcal{G}_{\leq}, \mathcal{G}_{\geq}, \mathcal{G}_{\forall}\}$ 4-tuple of sets of concept components in $\mathcal{ALN}$

Ensure : F and $\mathcal{G}$ are properly updated

foreach $\forall R.D$ *in* $C_{i,\forall}$ **do**
  if *there exists* $\leq 0\ R \sqcap \forall R.\bot$ *in* F **then**
    /* nothing to be done */
  else if *there exists* $\forall R.E_1$ *in* F ***and*** *((there exists* $\geq x\ R$ *in* F ***and*** $x \geq 1$*)* ***or*** *(*$\{\geq y\ R\} \in \mathcal{G}_{\geq}$ ***and*** $y \geq 1$*))* **then**
    /* merge D into E_1, inserting clashing concept components in $\mathcal{G}_{\forall}$ */
    if $\{\forall R.E_2\} \in \mathcal{G}_{\forall}$ **then**
      $\langle F',\ \mathcal{G}' \rangle := merge((D),\ E_1, \{E_{2,CN},\ E_{2,\leq},\ E_{2,\geq}, E_{2,\forall}\})$;
    else
      $\langle F',\ \mathcal{G}' \rangle := merge((D),\ E_1,\ \{\emptyset,\ \emptyset,\ \emptyset,\ \emptyset\})$;
    end
    remove $\forall R.E_1$ from F ;
    $F := F \sqcap \forall R.F'$;
    /* $\mathcal{G}'$ ''flattened'' as conjunctive expression becomes the filler of R */
    $addUniversalRestriction(\forall R.(\bigsqcap_{components} \mathcal{G}'),\ \mathcal{G})$;
  else if $\{\forall R.E_2\} \in \mathcal{G}_{\forall}$ ***and*** *((there exists* $\geq x\ R$ *in* F ***and*** $x \geq 1$*)* ***or*** *(*$\{\geq y\ R\} \in \mathcal{G}_{\geq}$ ***and*** $y \geq 1$*))* **then**
    /* take D, inserting clashing concept components in $\mathcal{G}_{\forall}$ */
    $\langle F',\ \mathcal{G}' \rangle := merge((D),\ \top, \{E_{2,CN},\ E_{2,\leq},\ E_{2,\geq}, E_{2,\forall}\})$;
    $F := F \sqcap \forall R.F'$;
    $addUniversalRestriction(\forall R.(\bigsqcap_{components} \mathcal{G}'),\ \mathcal{G})$;
  else
    if *no* $\forall R.E_1$ *exists in* F ***or*** $D \sqcap E_1$ *is satisfiable w.r.t.* $\mathcal{T}$ **then**
      $F := F \sqcap \forall R.D$;
    else
      remove $\forall R.E_1$ from F ;
      if *there exists* $\leq x\ R$ *in* F **then**
        remove $\leq x\ R$ from F ;
      end
      $F := F \sqcap \forall R.\bot \sqcap \leq 0\ R$;
    end
  end
end

Algorithm 6: $processUniversalRestrictions(C_{i,\forall},\ F,\ \mathcal{G})$

the new universal restriction just against the cache of discarded concepts (updating F and $\mathcal{G}$ in lines 18-19 as needed). The *addUniversalRestriction* support subroutine, reported in the Appendix A, is adopted as an optimization to merge the given-up fillers of universal restrictions recursively, in order to minimize the number of concept components in $\mathcal{G}_\forall$ and consequently the amount of checks when processing further C_i descriptions. Conversely, if no minimum cardinality restriction with value at least 1 is found, then if a pre-existent universal restriction in F does not exist or is compatible with the new one, we just add the latter to F (lines 21-22); otherwise (lines 23-28) the outcome of clash is just $\forall R.\bot$ and this implies a $\leq 0R$ must also be added to F to keep the result of fusion in CNF [106]. Notice that $\leq 0R$ can be added directly in this case, due to the non-existence of clashing greater-than number restrictions.

3.3 Autonomous decision-making

The proposed object (b)logging approach refers to a swarm of independent smart and mobile entities exchanging –"blogged"– semantically annotated *packets*. Each packet contains the sender's current knowledge of the environment and context, as well as a description of its provided actuation capabilities, expressed in OWL 2 language w.r.t. a shared reference ontology.

Whenever an agent N completes a data gathering and annotation round, it generates a semantic annotation CI_N representing fresh detected context information along with S_N describing its internal state. A 1-hop broadcast data dissemination protocol enables agents to receive, store, augment and forward knowledge, increasing accuracy of situation awareness progressively. In the same time window (*broadcast period* (BP) in what follows), incoming packets from other entities $B_i = \langle P_i, AvlCap_i \rangle \quad i = 1, \ldots, n$, are cached. P_i is a $\langle C, M, E \rangle$ 3-tuple of annotations, each providing a point of view on the context of the i-th node, according to the definitions introduced in [112] and detailed in Section 2.4. $AvlCap_i$ lists all the s_i currently available services supplied by i-th node $AvlCap_{i,1}, \ldots, AvlCap_{i,s_i}$. Each available j-th $\quad j = 1, \ldots, s_i$ service is further structured as a tuple $\langle IP_i, effCap_{i,j}, timestamp, TTL \rangle \quad i = 1, \ldots, N$. Moreover, s_i consists in (i) the address of the service owner, (ii) the semantic description of the effects provided by the actuator, (iii) the current timestamp, (iv) the *time to live* (TTL) *i.e.*, the time before considering the cached service expired/no longer available. Different services could have different TTL values. Smart objects continuously enrich their initial logical descriptive core, which models ground knowledge about their own features and capabilities as well as about concepts

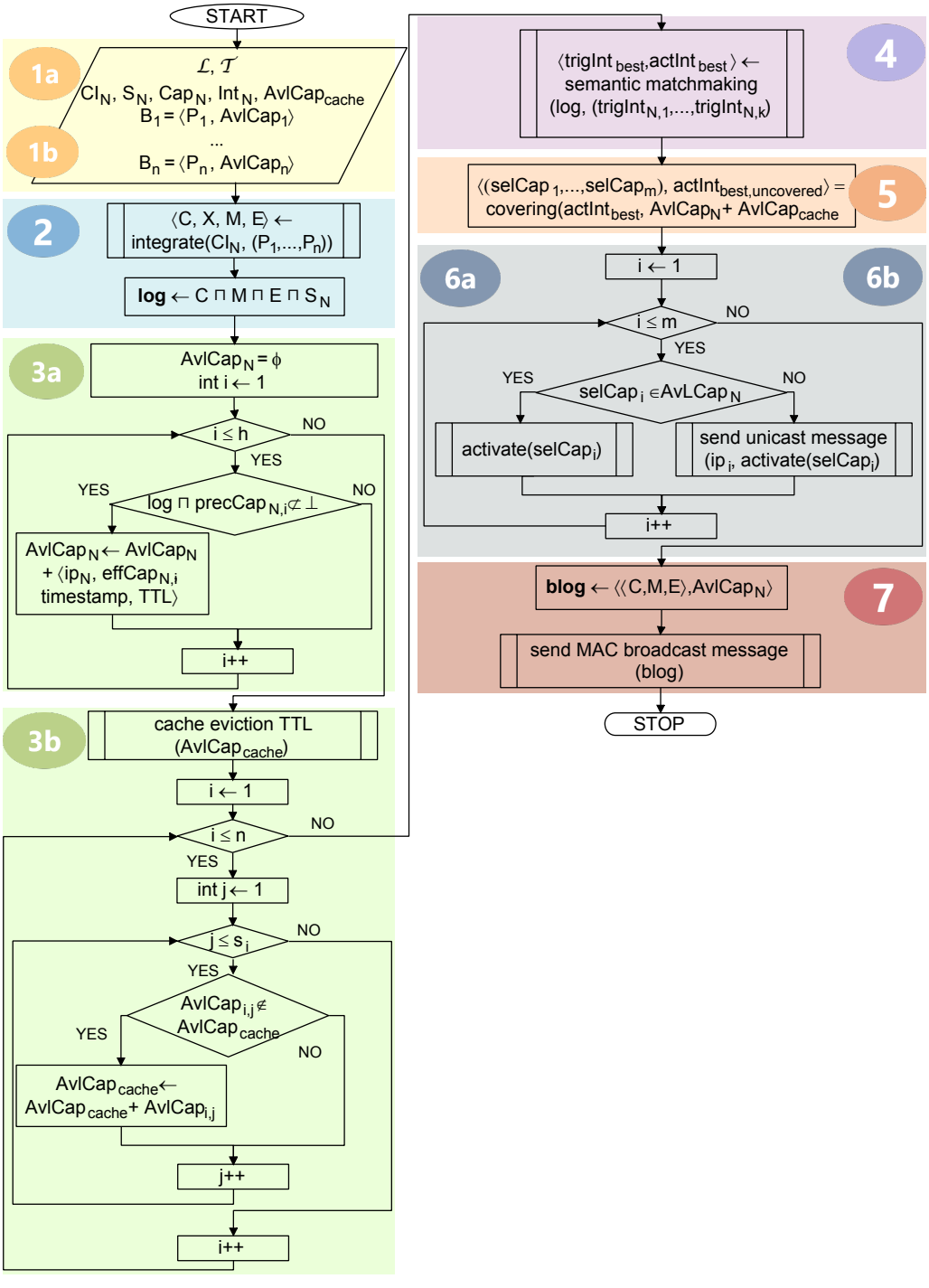


Figure 3.2: Object (b)logging workflow

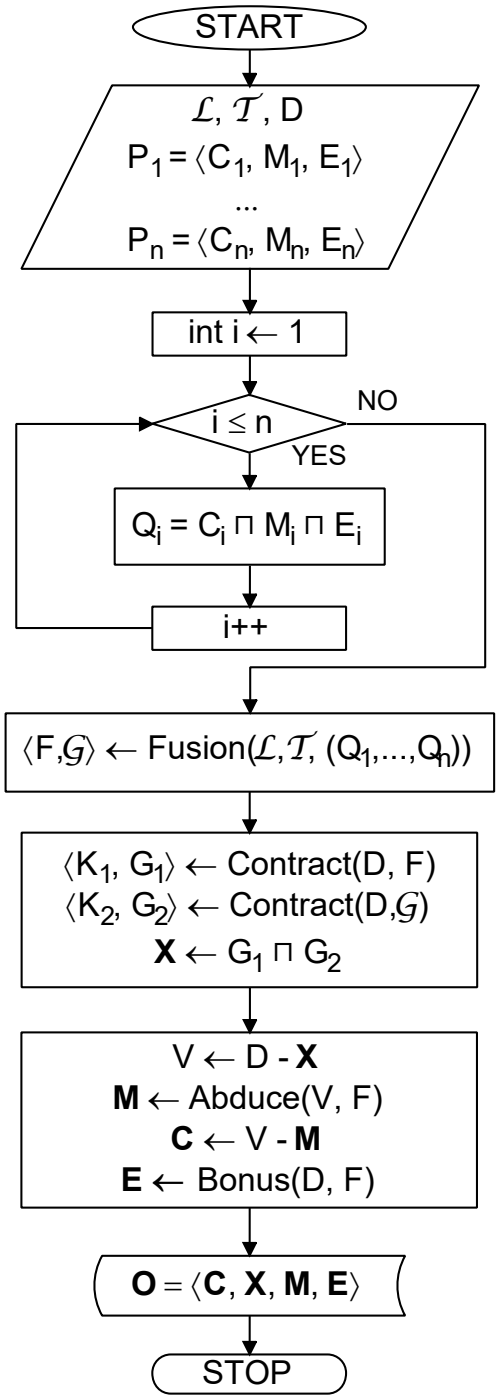


Figure 3.3: $\langle C, X, M, E \rangle = integrate(\mathcal{L}, \mathcal{T}, \mathrm{N}, (P_1, \ldots, P_n))$

and relationships with general validity in the domain. Particularly, a generic agent N keeps the set of equipped effectors $Cap_N = (Cap_{N,1}, \ldots, Cap_{N,h})$, that according to the environment and device state could be *available i.e.*, active or inactive. In what follows, preconditions and effects are denoted as a pair of annotations $Cap_{N,i} = \langle precCap_{N,i}, effCap_{N,i} \rangle \quad i = 1, \ldots, h$. The basic descriptive core also allows the smart object to take autonomous decision and behave in the appropriate way depending on the detected situations. Each node (*i.e.*, agent) is endowed with a set of $Int_N = (Int_{N,1}, \ldots, Int_{N,k})$ reactive behaviors, each modeled as a pair $Int_{N,i} = \langle trigInt_{N,i}, actInt_{N,i} \rangle \quad i = 1, \ldots, k$; trigger conditions $trigInt_{N,i}$ represent existing requirements about the environment state that must be met in order to require the corresponding actions $actInt_{N,i}$.

The object (b)logging workflow in Figure 3.2 shows the tasks required to implement the high-level steps of Figure 3.1. At the end of each broadcast period, a new processing and (b)logging round starts, articulated as follows for each generic node N in the scenario:

1. N is able to summarize the information gathered via its sensing interfaces into a semantically annotated description of the environment CI_N and internal state S_N. The logical descriptive core is initialized with: the domain conceptualization $\mathcal{T}$, the set of device built-in effectors Cap_N and the desired context-aware reactivity Int_N. $AvlCap_{cache}$ consists in the cached available services of other smart objects retrieved in the previous (b)logging rounds. Meanwhile, incoming packets $B_i \quad i = 1, \ldots, n$ collected during the current broadcast period are stored until the end of the round.

2. The modified version of the Concept Integration algorithm [112] (CI-v2) showed in Figure 3.3 is exploited in order to integrate the cached knowledge referring to the current time window $P_i \quad i = 1, \ldots, n$ along with the self-detected CI_N. The depicted approach leverages the ontology-based HLIF presented in Section 3.2 in order to get a new annotation F summarizing the knowledge in *received* observations. Robustness against spurious or inaccurate information is granted. The new X field is simply the conjunction of the incompatible fragments of the self-observed D with F and the collected given-up concept components $\mathcal{G}$, computed by Concept Contraction $\dot{\text{T}}$he new M field is determined by solving a Concept Abduction Problem between the semantically consistent fragments V and F obtained from D and G, respectively; this represents what only the node N has detected. To determine what information is common to the self-detected and the received observations, it is needed to subtract M from V via Concept Difference. Finally, the

Bonus of E w.r.t. F is computed in order to compute what other agents have seen but N has not. The *log* definition is grounded on the above Concept Integration outcome and is envisioned as the conjunction of the field C, M and E and the internal status S_N. The X part is discarded, coherently to how conflicting fragments in P_i are treated, due to the fact that it is dangerous to take a decision based on low-confidence information.

3. This task concerns the detection of the currently available services and is made up by two sub-tasks. The first one (3a) aims to find the services which can be activated on request among the h self-provided ones. For each i-th $(i = 1, \ldots, h)$) service, a Satisfiability check is run between its precondition $precCap_i$ and the log (*i.e.*, the conjunction of inferred context and internal status): if confirmed, the effect $effCap_i$ is added to the $AvlCap_N$ list along with the owner's address, the current timestamp and the TTL value. The second subtask (3b) updates the cached services $AvlCap_{cache}$ which can be provided by other agents. A cache eviction policy is necessary in order to discard expired services as probably no longer available from the owner, based on their generation timestamp and TTL value. Then, for each i-th $(i = 1, \ldots, n)$ received packet, the s_i listed services are checked: the j-th $(j = 1, \ldots, s_i)$ element is added only if not already cached in previous rounds.

4. Semantic matchmaking is leveraged to give decision capabilities to smart objects. It produces k ranked semantic similarity measures by comparing each trigger condition $trigInt_{N,i}$ $(i = 1, \ldots, k)$ of smart object behaviors with the log annotation. The needed interventions $\langle trigInt_{best}, actInt_{best} \rangle$ are determined as the ones with the lowest distance: a semantic relevance threshold could be set in order to discard results with a score below this value, as deemed irrelevant for decision-making.

5. The N agent reaches the final opportunistic decisions and take actions accordingly, by computing the Concept Covering task on the local set of available actuation capabilities $AvlCap_N$ and the other cached services $AvlCap_{cache}$. The list of selected services $selCap_i$ $(i = 1, \ldots, m)$ and an explanation of which part of the required intervention is not satisfied by the composite resource set $actInt_{best,uncovered}$ are returned.

6. According to the previous outcome, each selected service $selCap_i$ $i = 1, \ldots, m$ owned by the agent N (*i.e.*, in the $AvlCap^N$ set) is directly activated. Furthermore, unicast activation requests are sent to the

owners' addresses ip_i along with the specification of the desired services, in order to take advantage of distributed cooperation.

7. The $\langle C, M, E \rangle$ compact, high-level logical descriptions about the environment computed in task 2, enriched with additional information available from other peers and the available services $AvlCap_N$ compose the *blog*, which is published toward external devices and systems through wireless ad-hoc links. Specifically, MAC broadcasts messages are sent to one-hop neighbors. The X part of the Concept Integration output is not shared due to the fact that in the object (b)logging framework it is not useful to others agents, so it would be an unjustified network overload. Other multi-agent CAS approaches could consider also the conflicting fragments and X could be included in the shared knowledge.

The main purpose of the proposed framework is to achieve distributed collective cooperation. Smart agents execute locally the above steps in order to detect the relevant features of the current situation and trigger actions or make interventions accordingly, starting from integrated and comprehensive high-level knowledge inferred by considering additional information available from other entities at run time.

3.4 Case study

An illustrative example is discussed to highlight the capabilities of the proposed object (b)logging framework in dynamic environments, also demonstrating the induced benefits. It is extracted from a case study focused on the autonomous interaction of devices in *precision agriculture*. Agricultural processes can be automated to reduce effort and improve efficiency of resource management. In the reference scenario, several products –characterized by a set of features– are farmed in different fields, managed by a team of heterogeneous robots with different perception and actuation capabilities. The robots are agents able to (i) produce and share useful and comprehensive knowledge for finding out the needed interventions in the field, (ii) formulate plans to reach the mission goals and (iii) act accordingly. Annotations of field context and robots states, required interventions and provided capabilities refer to the *ONTAgri* [6] ontology, which has been selected and extended. The TBox is organized in two main parts, as shown in Figure 3.4: (i) agriculture concepts like soil characteristics, crop stages and service descriptions, (ii) system concepts such as sensors and effectors. In the case study, crop-specific actions or services are described by means of context-aware features,

using conjunctive expressions related to measured soil parameters, crop type, growth stage and weather conditions. The actuation capabilities, *e.g.*, irrigation and fertilization, are expressed as subclasses of the *Service* class, each having further subclasses. In order to relate services to actuator devices, their descriptions are annotated as a conjunctive expression of the actuation capabilities (usually independent from the specific crop type) needed to fulfill the service.

Let us consider the following example:

Heterogeneous farmer robots are employed in a wheat field in order to carry out a cyclical process of observation, data analysis, sharing and decision of actions to be performed. At the end of its last cycle, a rover robot N computes the semantic-based annotations CI_N and S_N from the collected raw data in the field and about itself respectively. Furthermore N also cached B_1, B_2, B_3 incoming packets shared by a nearby drone, another rover and an hybrid robots respectively.

Figure 3.5 depicts the self-detected descriptions[1] CI_N and S_N annotated in OWL 2 Manchester syntax [48]. Notice that the $AvlCap_{cache}$ cached in the previous (b)logging rounds is currently empty.

Figure 3.6 and Figure 3.7 show a part of the logical descriptive core of N, modelling ground knowledge about its features and capabilities, as well as decision-making criteria and self-adaptation strategies. For brevity, only few intervention samples among those modelled in the knowledge base are reported, along with their trigger preconditions and required actions.

Figure 3.8 reports the B_i $(i = 1, 2, 3)$ cached packets, retrieved from heterogeneous neighborhood. Multiple points of view over the context are provided thanks to the different sensing equipment owned by the various kinds of farming robots. For example drones are equipped with cameras used for recording field images and inferring the crop type and growth stage, while rovers are able to extract information about soil parameters.

Rover N merges personal and neighbourhood context viewpoints into a consistent and richer representation of the current situation. This output and its internal status are stored as a log.

The Concept Fusion inference service in Section 3.2 is exploited in order to merge the received and cached knowledge referring to the current time window, in order to get a new comprehensive annotation F and the detected

[1]For the sake of compactness and readability, all the reported concept expressions are simplified w.r.t. the ones actually used for the case study. In particular, for each universal restriction of the form `role only concept`, the reader should assume a corresponding `role some owl:Thing` existential restriction is present in conjunction.

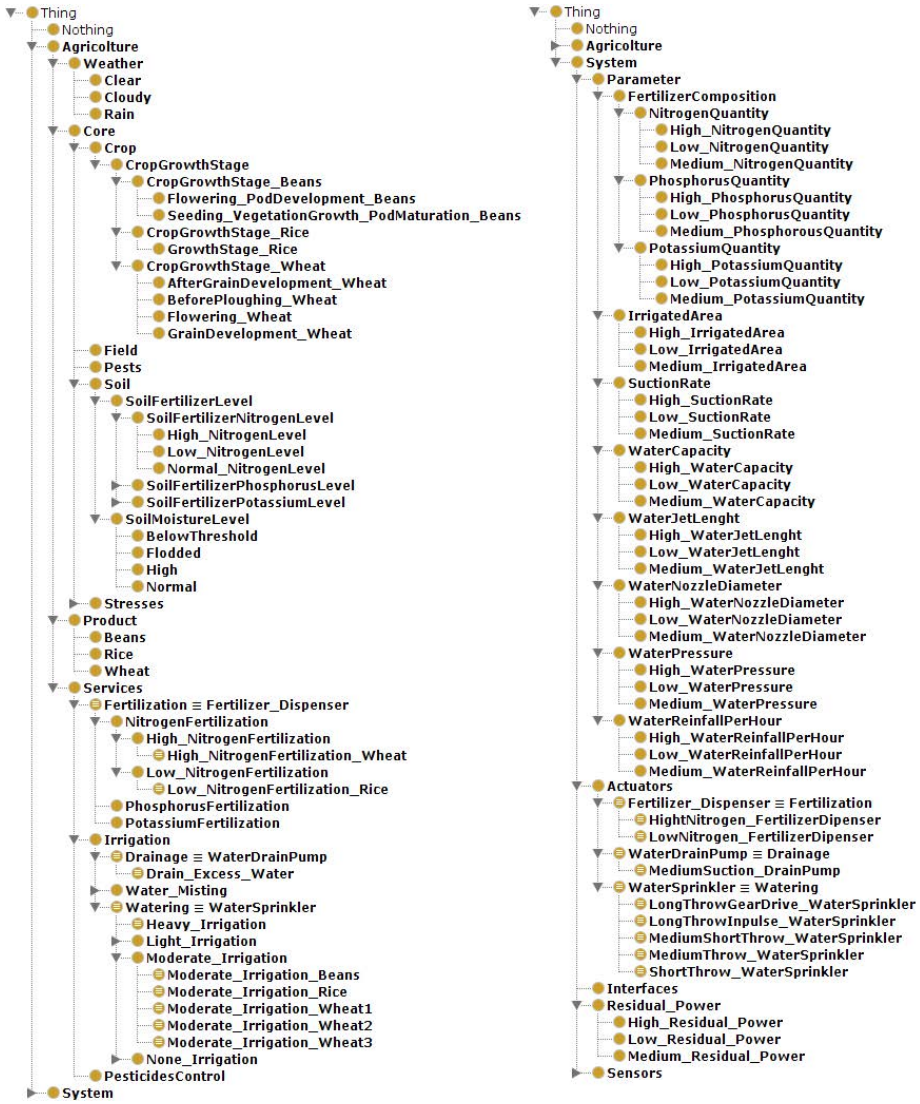


Figure 3.4: Precision agriculture domain ontology

CI_N EquivalentTo: (hasSoilMoistureLevel only BelowThreshold) and (hasSoilNitrogenFertilizerLevel only Low_NitrogenLevel) and (hasSoilPotassiumFertilizerLevel only Normal_PotassiumLevel) and (hasSoilPhosphorusFertilizerLevel only High_PhosphorusLevel)

S_N EquivalentTo: hasResidualPower only Low_Residual_Power

$AvlCap_{cached}$ = $\emptyset$

Figure 3.5: Rover N self-description

Cap_N = ($\langle$ MediumThrow_WaterSprinkler-Prec, MediumThrow_WaterSprinkler $\rangle$)

MediumThrow_WaterSprinkler-Prec EquivalentTo: hasResidualPower only (not (Low_Residual_Power))

MediumThrow_WaterSprinkler EquivalentTo: (hasWaterCapacity only Medium_WaterCapacity) and (hasIrrigatedArea only Medium_IrrigatedArea) and (hasWaterJetLenght only Medium_WaterJetLenght) and (hasWaterNozzleDiameter only Medium_WaterNozzleDiameter) and (hasWaterPressure only Medium_WaterPressure) and (hasWaterRainfallPerHour only Medium_WaterReinfallPerHour)

Figure 3.6: Subset of rover N's services

conflicting elements $\mathcal{G}$. The approach enables management of incomplete information and elimination of inconsistencies. Figure 3.9 shows the Concept Fusion output: in particular conflicting information about the phosphorus level in soil is reabsorbed. Subsequently, the CI-v2 algorithm presented in Section 3.3 is exploited in order to merge N's personal viewpoint about the context CI_N and the external one, in a 4-tuple of annotations. Only the non-conflicting fields C, M, E are joined together with rover residual power level S_N in the log, while the –potentially inaccurate– phosphorus level detection is discarded.

Rover N identifies all farming services required in its area and detects the most suitable actuators among those available for each needed service. N residual power level is insufficient to turn on its water sprinkler, which cannot be used to act on the environment directly. However, N could take advantages of actuators available from other robots cached in $AvlCap_{cache}$ and request their activation in a fully autonomous and collaborative fashion. The acquired knowledge is shared like a blog with nearby farmer robots.

As depicted in Figure 3.10, the Satisfability check between N's actuator precondition $MediumThrow\_WaterSprinkler - Prec$ and the registered log is unsuccessful: $AvlCap_N$ remains empty. Instead $AvlCap_{cached}$ is initialized with the services in B_i $(i = 1, 2, 3)$ shared by nearby robots. Semantic

$Int_{N,fertilization}$ = (⟨ High_NitrogenFertilization_Wheat-Trig, High_NitrogenFertilization_Wheat-Act ⟩, ⟨ Low_NitrogenFertilization_Rice-Trig, Low_NitrogenFertilization_Rice-Act ⟩, ...)

High_NitrogenFertilization_Wheat-Trig EquivalentTo: (hasCropStage only GrainDevelopment_Wheat) and (hasCropType only Wheat) and (hasSoilNitrogenFertilizerLevel only Low_NitrogenLevel)

High_NitrogenFertilization_Wheat-Act EquivalentTo: (hasNitrogenQuantity only High_NitrogenQuantity) and (hasPhosphorusQuantity only Low_PhosphorusQuantity) and (hasPotassiumQuantity only Low_PotassiumQuantity)

Low_NitrogenFertilization_Rice-Trig EquivalentTo: (hasCropStage only GrowthStage_Rice) and (hasCropType only Rice) and (hasSoilNitrogenFertilizerLevel only High_NitrogenLevel)

Low_NitrogenFertilization_Rice-Act EquivalentTo: (hasNitrogenQuantity only Low_NitrogenQuantity) and (hasPhosphorusQuantity only Medium_PhosphorusQuantity) and (hasPotassiumQuantity only Low_PotassiumQuantity)

$Int_{N_irrigation}$ = (⟨ Light_Irrigation_Wheat-Trig, Light_Irrigation_Wheat-Act ⟩, ⟨ Moderate_Irrigation_Wheat-Trig, Moderate_Irrigation_Wheat-Act ⟩, ...)

Light_Irrigation_Wheat-Trig EquivalentTo: (hasCropStage only Flowering_Wheat) and (hasCropType only Wheat) and (hasSoilMoistureLevel only Normal) and (hasWeatherCondition only (not (Rain)))

Light_Irrigation_Wheat-Act EquivalentTo: Watering and (hasWaterCapacity only Low_WaterCapacity) and (hasWaterNozzleDiameter only Low_WaterNozzleDiameter) and (hasWaterRainfallPerHour only Low_WaterReinfallPerHour)

Moderate_Irrigation_Wheat-Trig EquivalentTo: (hasCropStage only GrainDevelopment_Wheat) and (hasCropType only Wheat) and (hasSoilMoistureLevel only BelowThreshold) and (hasWeatherCondition only (not (Rain)))

Moderate_Irrigation_Wheat-Act EquivalentTo: Watering and (hasWaterCapacity only Medium_WaterCapacity) and (hasWaterNozzleDiameter only Medium_WaterNozzleDiameter) and (hasWaterRainfallPerHour only Medium_WaterReinfallPerHour)

Figure 3.7: Subset of rover N's intervention KB

matchmaking process, as described in Section 2.2.3, is carried out to detect possibly needed irrigation and fertilization actions on the field, by considering the trigger conditions annotated in the KB and the log description. $High\_NitrogenFertilization\_Wheat$ and $Moderate\_Irrigation\_Wheat$ are deemed as necessary in order to improve the wheat crop status. Now N is

B_1 = $\langle\langle C_1, M_1, E_1\rangle$, ShortThrow_WaterSprinkler $\rangle$

Q_1 = $C_1 \sqcap M_1 \sqcap E_1$ EquivalentTo: (hasWeatherCondition only Clear) and (hasCropType only Wheat) and (hasCropStage only GrainDevelopment_Wheat)

ShortThrow_WaterSprinkler EquivalentTo: (hasWaterCapacity only Low_WaterCapacity) and (hasIrrigatedArea only Low_IrrigatedArea) and (hasWaterJetLenght only Low_WaterJetLenght) and (hasWaterNozzleDiameter only Low_WaterNozzleDiameter) and (hasWaterPressure only Medium_WaterPressure) and (hasWaterRainfallPerHour only Low_WaterReinfallPerHour)

B_2 = $\langle\langle C_2, M_2, E_2\rangle$, MediumShortThrow_WaterSprinkler $\rangle$

Q_2 = $C_2 \sqcap M_2 \sqcap E_2$ EquivalentTo: (hasWeatherCondition only Cloudy) and (hasCropType only Wheat) and (hasSoilNitrogenFertilizerLevel only Low_NitrogenLevel) and (hasSoilPotassiumFertilizerLevel only Normal_PotassiumLevel) and (hasSoilPhosphorusFertilizerLevel only Low_PhosphorusLevel)

MediumShortThrow_WaterSprinkler EquivalentTo: (hasWaterCapacity only Low_WaterCapacity) and (hasIrrigatedArea only Low_IrrigatedArea) and (hasWaterJetLenght only Low_WaterJetLenght) and (hasWaterNozzleDiameter only Medium_WaterNozzleDiameter) and (hasWaterPressure only Medium_WaterPressure) and (hasWaterRainfallPerHour only Medium_WaterReinfallPerHour)

B_3 = $\langle\langle C_3, M_3, E_3\rangle$, HighNitrogen_FertilizerDispenser $\rangle$

Q_3 = $C_3 \sqcap M_3 \sqcap E_3$ EquivalentTo: (hasSoilNitrogenFertilizerLevel only Low_NitrogenLevel) and (hasSoilPotassiumFertilizerLevel only Normal_PotassiumLevel) and (hasSoilPhosphorusFertilizerLevel only High_PhosphorusLevel)

HighNitrogen_FertilizerDispenser EquivalentTo: (hasNitrogenQuantity only High_Nitrogen) and (hasPhosphorusQuantity only Low_Phosphorus) and (hasPotassiumQuantity only Low_Potassium)

Figure 3.8: Rover N's retrieved packets

$\langle F, \mathcal{G} \rangle$ = $fusion(\mathcal{L}, \mathcal{T}, (Q_1, Q_2, Q_3))$ = ⟨ (hasWeatherCondition only (Clear and Cloudy)) and (hasCropType only Wheat) and (hasCropStage only GrainDevelopment_Wheat) and (hasSoilNitrogenFertilizerLevel only Low_NitrogenLevel) and (hasSoilPotassiumFertilizerLevel only Normal_PotassiumLevel) ,
{ (hasSoilPhosphorusFertilizerLevel only Low_PhosphorusLevel), (hasSoilPhosphorusFertilizerLevel only High_PhosphorusLevel) } ⟩

$\langle C, X, M, E \rangle$ = $integrate(\mathcal{L}, \mathcal{T}, CI_N, \langle F, \mathcal{G} \rangle)$ = ⟨ (hasSoilNitrogenFertilizerLevel only Low_NitrogenLevel) and (hasSoilPotassiumFertilizerLevel only Normal_PotassiumLevel),
(hasSoilPhosphorusFertilizerLevel only High_PhosphorusLevel),
(hasSoilMoistureLevel only BelowThreshold),
(hasWeatherCondition only (Clear and Cloudy)) and (hasCropType only Wheat) and (hasCropStage only GrainDevelopment_Wheat) ⟩

log = $C \sqcap M \sqcap E \sqcap S_N$ EquivalentTo: (hasSoilNitrogenFertilizerLevel only Low_NitrogenLevel) and (hasSoilPotassiumFertilizerLevel only Normal_PotassiumLevel)and (hasSoilMoistureLevel only BelowThreshold) and (hasWeatherCondition only (Clear and Cloudy)) and (hasCropType only Wheat) and (hasCropStage only GrainDevelopment_Wheat) and (hasResidualPower only Low_Residual_Power)

Figure 3.9: Rover N's log derived by means of Concept Fusion and Integration

able to identify the set of available actuators best covering the requested interventions –along with the possibly uncovered part of the request– by means of the greedy Concept Covering process described in Section 2.2.3. As shown in Figure 3.10, selected services are $HighNitrogen\_FertilizerDispenser$ and $MediumShortThrow\_WaterSprinkler$; their activation is requested via unicast messages to service owner robots. It can be noticed that the selection of $MediumShortThrow\_WaterSprinkler$ is a suboptimal choice, as the best service $MediumThrow\_WaterSprinkler$ is currently unavailable. The adoption of non-standard inference services enables support for approximate matches, resource ranking and formal explanation of outcomes: the reported uncovered part is an example. The conjunction of deemed reliable knowledge C, M, E and the (empty) $AvlCap_N$ make up the blog, which is shared towards the external world through 1-hop broadcast messages.

The example was intentionally kept simple for explanatory purposes. In real scenarios, a smart object could expose more services and more complex descriptions about context, capabilities and decision-making criteria. Nevertheless, the basic interaction sequence and mechanisms are the same.

$AvlCap_N$ = ∅

$AvlCap_{cached}$ = (ShortThrow_WaterSprinkler, MediumShortThrow_WaterSprinkler, HighNitrogen_FertilizerDispenser)

covering(High_NitrogenFertilization_Wheat-Act, $AvlCap_N \sqcap AvlCap_{cached}$) = (HighNitrogen_FertilizerDispenser, ⊤)

covering(Moderate_Irrigation_Wheat-Act, $AvlCap_N \sqcap AvlCap_{cached}$) = (MediumShortThrow_WaterSprinkler, hasWaterCapacity only Medium_WaterCapacity)

blog = ⟨ $C \sqcap M \sqcap E$, $AvlCap_N$ ⟩ = ⟨ (hasSoilNitrogenFertilizerLevel only Low_NitrogenLevel) and (hasSoilPotassiumFertilizerLevel only Normal_PotassiumLevel)and (hasSoilMoistureLevel only BelowThreshold) and (hasWeatherCondition only (Clear and Cloudy)) and (hasCropType only Wheat) and (hasCropStage only GrainDevelopment_Wheat), ∅ ⟩

Figure 3.10: Rover N Concept Covering task(s) and blog

3.5 Experiments

This section provides a detailed description about prototypical implementations of the proposed framework and an assessment of its feasibility through experimental results. Tests aimed to assess intra- and inter-node performance. Intra-node experiments are focused on evaluating efficiency and scalability of the Concept Fusion inference service on a real resource-constrained computing platform. Inter-node tests refer to the evaluation of the overall devised semantic-enabled object (b)logging framework, by considering a plethora of heterogeneous simulated nodes interacting autonomously and sharing information, cooperating and orchestrating services in a MANET.

3.5.1 Concept Fusion

A complete implementation of the Concept Fusion inference service defined in Section 3.2 has been developed in C language, integrating a prototypical C port of the *Mini-ME* (*Mini Matchmaking Engine*) semantic matchmaker and reasoner [122] under development at the Information Systems Laboratory of the Polytechnic University of Bari. Early experiments assessing efficiency and scalability on a real resource-constrained computing platform are reported hereafter. Performance evaluation has been carried out on a Raspberry Pi Model B single-board computer, equipped with a single-core ARM11 CPU at 700 MHz, 512 MB RAM (shared with GPU), 32 GB storage memory on SD card, Raspbian Stretch OS[2]. Turnaround time and peak memory usage

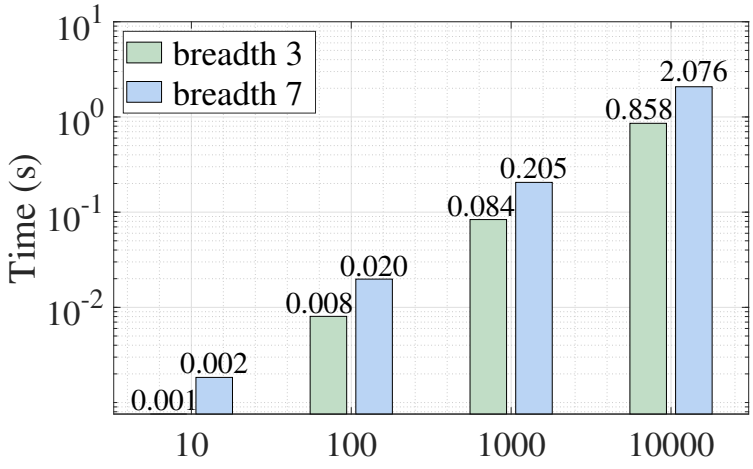
[2] http://www.raspbian.org/

of Concept Fusion have been considered: reported results are the average of ten repeated runs. Memory peak values represent the maximum resident set size (MRSS) for the process during its lifetime, extracted via GNU *time* command (version 1.7).

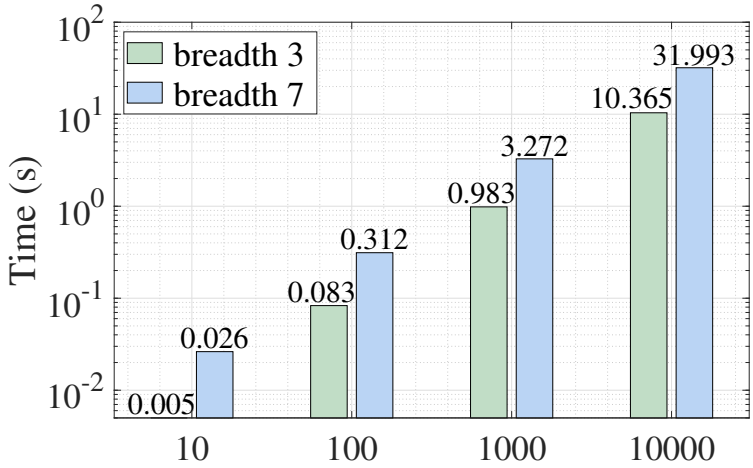
Experiments have taken into account (i) an increasing number of resources undergoing Concept Fusion and (ii) a growing complexity of individual resources, in terms of the nesting level of property restrictions (*depth*) and the number of conjuncts in nested properties (*breadth*). The semantic annotations have been randomly generated by configuring the following parameters: resources = $\{10, 100, 1000, 10000\}$, depth = $\{1, 2\}$, breadth = $\{3, 7\}$, for a total of 16 tests. For each universal restriction $\forall R.D$ in generated expressions, an existential restriction has been associated in order to simulate the detection of an attribute ($\exists R$) with a specific value ($\forall R.D$) by an agent's sensor in a realistic collaborative sensing scenario. Results are reported hereafter.

Time. Figure 3.11(a) and Figure 3.11(b) show the Concept Fusion inference service provides acceptable performance, also with the worst stress tests involving 10000 resources. As expected, Figure 3.12(a) demonstrates that in all the experiments time increases significantly with higher resources number. Furthermore, processing times tend to rise at higher resource complexity, as the Concept Fusion task has to work on a larger number of concept components, as evidenced in Figure 3.12(b). The fusion time among 7-breadth resource dominates that with 3-breadth resources in both 1-depth and 2-depth experiments. Obviously 2-depth resources fusion takes more time than 1-depth resources fusion, for the same reasons. Growth exhibits linear trends in both analyzed perspectives, suggesting adequate scalability of the proposed approach and demonstrating sustainability for the target scenarios.

Memory. Max RSS memory usage is shown in Figure 3.13(a) and Figure 3.13(b). Also in this case, the number and the complexity of involved resources influenced memory consumption, as demonstrated in Figure3.14(a) and 3.14(b) respectively. Memory consumption can be deemed as low in the 10-, 100-, and 1000-resource scenarios, while dealing with 10000 2-depth resources, it reached 186 MB of memory peak mainly due to large number and complexity of resources loaded in memory. It should be noted that the prototypical Concept Fusion implementation keeps in memory all the input resources along with support data structures, and discards them only at the end of fusion process. Memory usage improvements could be introduced by unloading resources immediately after they are processed. In this way the gradual memory freeing of resource annotations will compensate for the growth of Concept Fusion data structures.

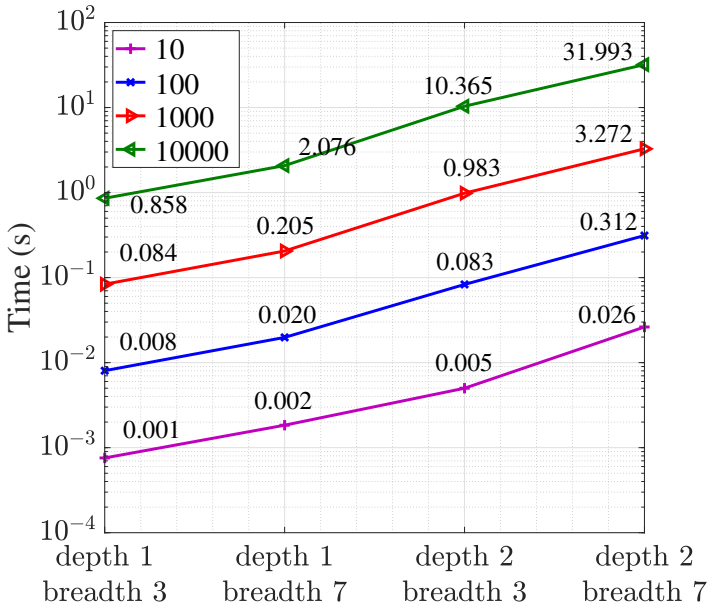


(a) depth = 1

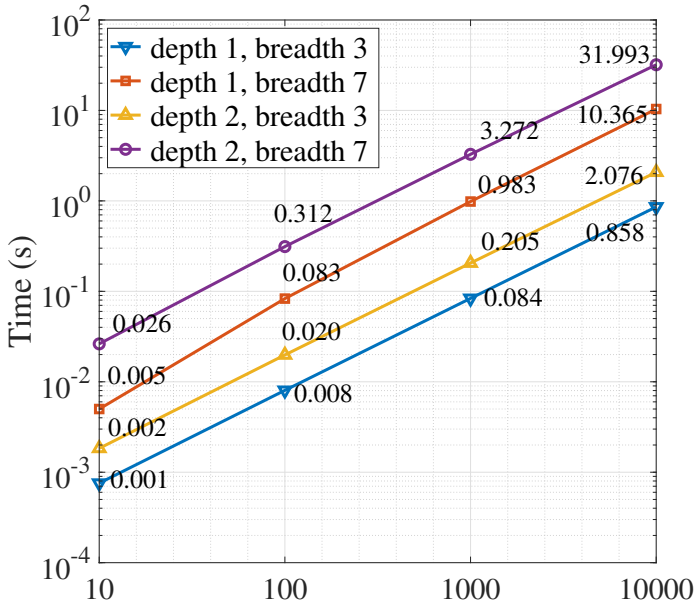


(b) depth = 2

Figure 3.11: Concept Fusion turnaround time

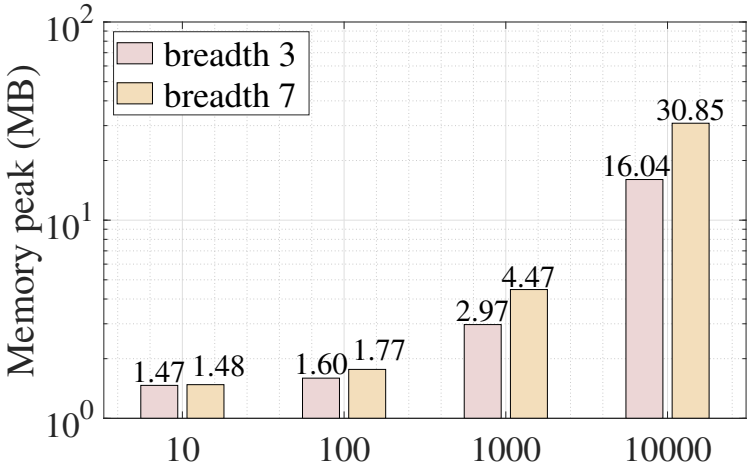


(a) Growing resources number

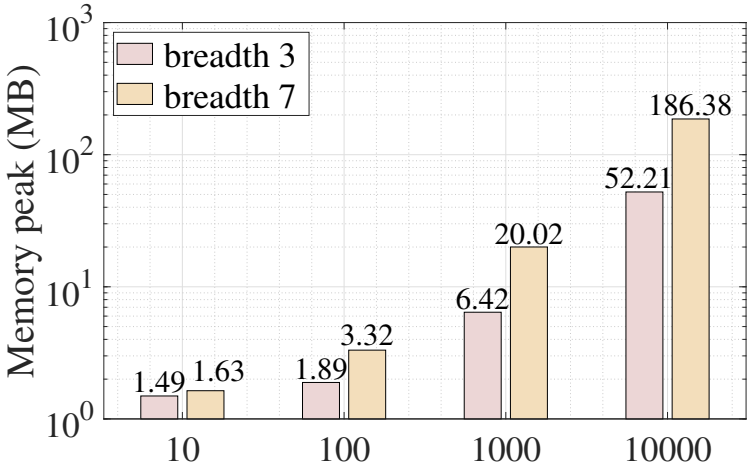


(b) Increasingly complex configurations

Figure 3.12: Time performance comparison

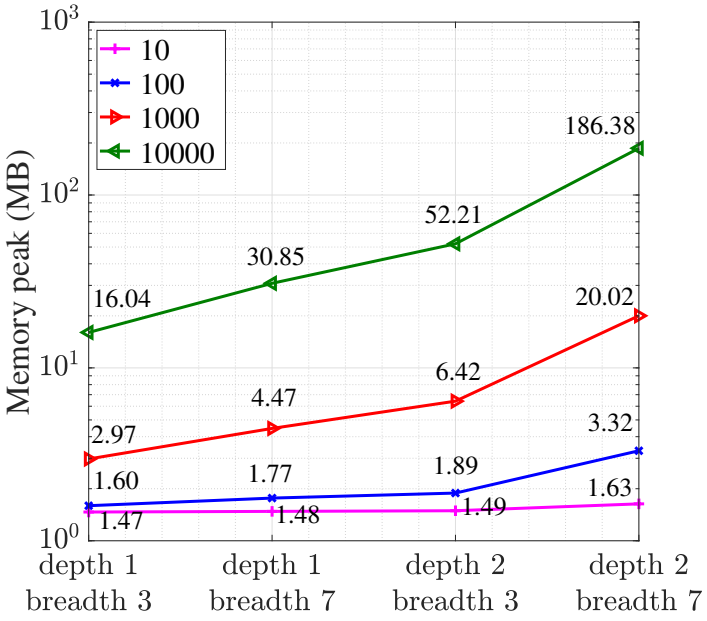


(a) depth = 1

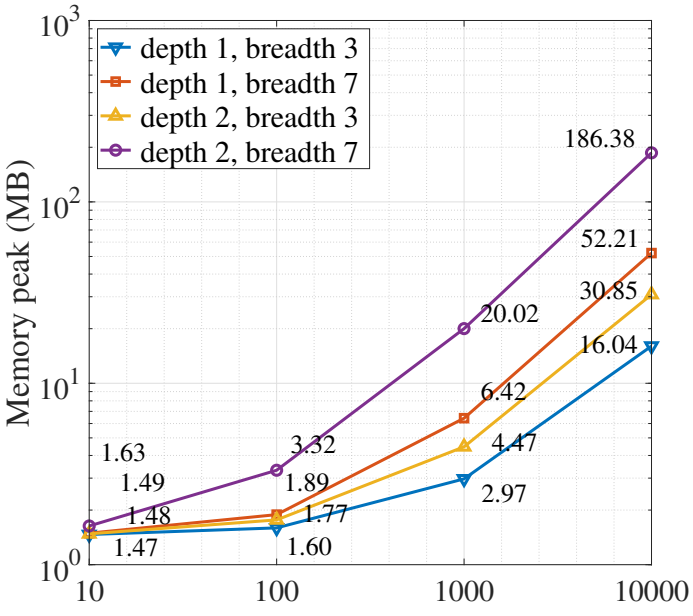


(b) depth = 2

Figure 3.13: Concept Fusion peak memory usage



(a) Growing resources number



(b) Increasingly complex configurations

Figure 3.14: Memory peak performance comparison

Globally, experimental outcomes show the approach is computationally sustainable on computing devices for IoT. Time performance is well suited to real-time context detection and action planning, while memory load needs some improvements in order to fit properly the strict constraints of pervasive objects.

3.5.2 Logging and blogging

Implementation and performance evaluation of the object (b)logging framework detailed in Section 3.3 has been carried out exploiting the *Omnet++* network simulator [140] with the *INETMANET*[3] framework [7]. OMNeT++ is an event-based network simulator featuring a collection of frameworks to support specific types of networks. The INETMANET fork of the *INET*[4] framework 3.x branch provides important additional components to simulate mobile ad-hoc network families such as IEEE 802.15.4 (ZigBee), including models for PHY and MAC layers, battery, and application protocols.

The developed testbed simulates a MANET for cooperative monitoring of fields in precision agriculture, where each agent runs on a robot with different sensing and actuation capabilities. The three types of robot agents defined in Section 3.4 have been modelled and are able to infer and share concept expressions referred to the ontology *e.g.*, soil parameters, crop type, growth stage, weather conditions, as well as available services *e.g.*, fertilization, irrigation. Specifically, *Type 0* hybrid robots are capable of sensing all the above features, *Type 1* rover robots detect only the soil parameters, and *Type 2* drone robots gather only aerial data about crop type, growth stage and weather. Multiple kinds of irrigation services with different characteristics in water capacity, pressure, jet length, irrigated area and nozzle diameter are randomly assigned to robot types.

The reference farmland scenario is described in Table 3.1: checkerboard arrangement of 9 fields has been adopted, each having peculiar crop features and weather characteristics. The north-east map zones are characterized by clear weather conditions while in south-east there are clouds. Furthermore two fields on the west side of the map have a different beans growth stage and almost all map zones have a low nitrogen level except the two on the south side. Pre-defined annotations pertaining to the different map zones simulated the environmental conditions extraction. The different actions in Table 3.2 are needed in each field according to the modelled situations.

The proposed framework has been developed by extending the INET

[3]INETMANET: `https://github.com/aarizaq/inetmanet-3.x`
[4]INET framework for OMNeT++:`https://inet.omnetpp.org/`

Table 3.1: Scenario environment description

x/y (m)	**y ≤ 200**	**200 < y ≤ 400**	**400 < y ≤ 600**
x ≤ 200	Clear, Beans, Flowering Pod Development, Below Threshold, Low Nitrogen, Normal Potassium, Normal Phosphorus	Clear, Beans, Flowering Pod Development, Below Threshold, Low Nitrogen, Normal Potassium, Normal Phosphorus	Cloudy, Beans, Flowering Pod Development, Below Threshold, Low Nitrogen, Normal Potassium, Normal Phosphorus
200 < x ≤ 400	Clear, Beans, Flowering Pod Development, Below Threshold, Low Nitrogen, Normal Potassium, Normal Phosphorus	Cloudy, Beans, Flowering Pod Development, Below Threshold, Low Nitrogen, Normal Potassium, Normal Phosphorus	Cloudy, Beans, Seeding Vegetation Growth Pod Maturation, Below Threshold, Low Nitrogen, Normal Potassium, Normal Phosphorus
400 < x ≤ 600	Cloudy, Beans, Flowering Pod Development, Below Threshold, Low Nitrogen, Normal Potassium, Normal Phosphorus	Cloudy, Beans, Flowering Pod Development, Below Threshold, Normal Nitrogen, Normal Potassium, Normal Phosphorus	Cloudy, Beans, Seeding Vegetation Growth Pod Maturation, Below Threshold, Normal Nitrogen, Normal Potassium, Normal Phosphorus

Table 3.2: Actions required on scenario environment

x/y (m)	**y ≤ 200**	**200 < y ≤ 400**	**400 < y ≤ 600**
x ≤ 200	Moderate Irrigation Beans	Moderate Irrigation Beans	Moderate Irrigation Beans
200 < x ≤ 400	Moderate Irrigation Beans	Moderate Irrigation Beans	Light Irrigation Beans
400 < x ≤ 600	Moderate Irrigation Beans	Moderate Irrigation Beans	Light Irrigation Beans

UDPBasicBurst[5] modules. Dynamic On-demand MANET routing protocol[6] (DYMO) has been adopted. Non-standard inference services provided by the C port of Mini-ME mobile reasoner [122] enhanced with the Concept Fusion and CI-v2 algorithms (Section 3.2) are invoked to implement all the object (b)logging workflow tasks. The reasoner is integrated directly within the agent software module. *Google Protocol Buffers*[7] is used to serialize the high-level structured data to a platform-independent binary format.

In order to obtain a quantitative analysis of the performance, the metrics in Table 3.3 have been measured and evaluated. They can be grouped as follows:

- *Blogging load*: network load generated by blog messages is measured as bytes and as packets. Specifically, out of the total incoming bandwidth usage, the *useful* bandwidth is extracted, corresponding to messages which do not have an expired TTL and are not *out of order*: in INETMANET each packet has a progressive ID and a node will mark an incoming packet as out-of-order if it has received another packet from the same sender with higher ID value. Outgoing bandwidth usage is also measured and the *bandwidth gain* is computed as the ratio between outbound and inbound data usage, representing the capability of the proposed fusion approach to summarize knowledge.

- *Activation load*: network usage due to activation messages exchanged among nodes is measured, focusing on overall incoming, useful incoming and outgoing data.

- *Reasoning time*: specifically time required for the CI-v2 algorithm and the overall object (b)logging workflow are measured. In order to represent a realistic performance profile of resource-constrained agents, in the simulation the reasoning time is delayed by an order of magnitude. This proportional factor has been estimated by comparing the reasoning time taken by a single node in the Omnet++ simulation environment and a Raspberry Pi Model B single-board computer on the same set of resources.

- *Fusion performance:* this set of metrics concerns the quality the results of Concept Fusion for context detection. Annotations computed by each node are compared with the ground truth on the state of the simulation area the agent is located in at the end of the BP.

[5]UDPBasicBurst documentation: `https://doc.omnetpp.org/inet/api-current/neddoc/inet.applications.udpapp.UdpBasicBurst.html`

[6]DYMO: `https://tools.ietf.org/html/draft-ietf-manet-dymo-26`

[7]Google Protocol Buffers: `https://github.com/google/protobuf`

- *Decision performance:* the quality of service activation decision obtained from semantic matchmaking is analyzed w.r.t. the theoretically optimal decision in for the simulation area configuration the agent is located in at the end of the BP.

Simulation parameters reported in Table 3.4 determine the different operating conditions in the various scenarios. A total of 96 experiments, as summarized in Table 3.5 and Table 3.6, have been carried out with 3 runs and the mean values have been considered. Furthermore, the following settings have been adopted: (i) simulation time fixed to 300 s, (ii) weights for Concept Abduction an Concept Contraction penalties in semantic matchmaking set to 0.3 and 0.7 respectively, in order to penalize more the explicitly conflicting features, and (iii) minimum semantic relevance for decision-making set to 0.8. Performance evaluation has been executed on a *VMware vSphere* virtual machine running on a server blade[8]. In what follows the reported results are referred to the weighted average of the values collected over the simulation time. For each set of metrics, experimental analysis are provided.

Blogging Load

The blogging load results in long BP scenarios, *i.e.*, experiments from #1 to #48, are reported in Figure 3.15(a) and 3.16(a), considering respectively the byte size and the number of packets. Basically, it is possible to observe that the adoption of IEEE 802.15.4 protocol with IEEE 802.11g is the optimal choice if compared with the usage with IEEE 802.11a. More total data (in bytes and packets) are collected in these scenarios, as well as useful (*i.e.*, not expired, not out-of-order) information. As expected, blogging load is affected by the number of involved nodes. In 2000 nodes scenarios the above values are higher than 500 and 1000 nodes scenarios, reaching over 12000 B and 40 packets. It can be further noticed that performance in experiments with both IEEE 802.11g and IEEE 802.11a protocols drastically worsens in high-density scenarios, due to a growing number of collisions. Furthermore, slight differences exist between high and low detection probability experiments. In the latter case, scenarios must cope with an increased number of conflicting (and therefore discarded) fragments, and consequently less shared information. Concerning blog output, nodes broadcast a single packet of approximately 300 bytes. In heterogeneous nodes scenarios this size is slightly lower due to limited sensing capabilities per node and consequently decreased

[8]Intel Xeon E5-2650 v3 CPU –8 cores/16 threads at 2.30 GHz–, 96 GB of RAM and Ubuntu 16.04 (64bit) operating system.

Table 3.3: Evaluated metrics

Performance	**Name**		**Description**
Blogging Load	TotalBlogInput (B)	Bytes per BP	Total broadcast input byte bandwidth
	UsefulBlogInput (B)		Not out-of-order, not expired broadcast input byte bandwidth
	BlogOutput (B)		Output broadcast (*i.e.*, shared) byte bandwidth
	BandwidthByteGain		Ratio between the broadcast input and the broadcast output byte bandwidth
	TotalBlogInputPackets	Packets per BP	Total broadcast input packets number
	UsefulBlogInputPackets		Not out-of-order, not expired, broadcast input packets number
	BlogOutputPackets		Output broadcast packets number
	BandwidthPacketsGain		Ratio between the broadcast input and the broadcast output packets number
Activation Load	TotalActivationInput (B)	Bytes per BP	Total unicast (*i.e.*, activation requests of provided services) input byte bandwidth
	UsefulActivationInput (B)		Not out-of-order, not expired unicast input byte bandwidth
	ActivationOutput (B)		Output unicast (*i.e.*, sent) byte bandwidth
	TotalActivationInputPackets	Packets per BP	Total unicast input packets number
	UsefulActivationInputPackets		Not out-of-order, not expired unicast input packets number
	ActivationOutputPacket		Output unicast packets number
Reasoning Time	IntegrationTime (ms)		Turnaround time to compute the CI-v2 algorithm
	TotalReasoningTime (ms)		Turnaround time to accomplish the object (b)logging workflow
Fusion Performance	DetectedKnowledgeCovering%		% of self-detected information w.r.t. the real map zone description
	DetectedConflictingKnowledge%		% of detected information in conflict w.r.t. the real map zone description
	ReceivedKnowledgeCovering (C,M,E)%		% of information in C-M-E parts of incoming packets w.r.t. the real map zone description
	ConflictingInfoPerPackets (C,M,E)%		% of information in C-M-E parts per incoming packet conflicting w.r.t. the real map zone description
	IntegrationKnowledgeCovering (C,X,M,E)%		% of integrated (CI-v2) information in C-X-M-E parts w.r.t. the real map zone description
	IntegrationConflictingKnowledge (C,X,M,E)%		% of integrated (CI-v2) information in C-X-M-E parts conflicting w.r.t. the real map zone description
Decision Performance	DecisionCovering%		% of selected action description w.r.t. the needed one
	WrongDecision%		% of selected action description conflicting w.r.t. the needed one

Table 3.4: Simulation configuration parameters

Parameter	Description	Value	Configuration Name
Broadcast Period (BP)	Time window in which on-board data gathering and retrieved packets caching are carried out	1 sec	Short BP
		10 sec	Long BP
Node Number	Number of involved nodes	500	Low-Density Scenario
		1000	Medium-Density Scenario
		2000	High-Density Scenario
Node Types	Percentage of involved nodes per type affecting sensing and actuation capabilities	100% Type 0	Homogeneous Nodes
		40% Type 0 30% Type 1 30% Type 2	Heterogeneous Nodes
Mobility	INETMANET mobility models adopted to describe the node motion patterns	80% Stationary Mobility 10% Mass Mobility 10% Gauss-Markov Mobility	Mostly Static Mobility
		20% Stationary Mobility 40% Mass Mobility 40% Gauss-Markov Mobility	Mostly Dynamic Mobility
Communication Protocols and Multiradio	Percentage of involved nodes adopting a certain communication protocols	30% 802.11a(54 Mbps) 30% 802.11g(2 Mbps) 40% multiradio 802.11a(54 Mbps)+802.11g(2 Mbps)	802.11a+802.11g
		30% 802.15.4, 250 Kbps 30% 802.11g, 2 Mbps 40% multiradio 802.15.4 (ZigBee), 250 Kbps + 802.11g, 2 Mbps	802.15.4+802.11g
Detection Probability	Probability to detect context annotations with a certain accuracy degree compared to the real description in Table 3.1	Correct: 0.7 Erroneous and conflicting: 0.1 Erroneous: 0.1 Missing: 0.1	High Detection Probability
		Correct: 0.4 Erroneous and conflicting: 0.3 Erroneous: 0.15 Missing: 0.15	Low Detection Probability

Table 3.5: Experiment configurations Pt.1 (#1 - #48)

# Experiment	BP	Node Number	Node Types	Mobility	Protocols	Detection Probability
1	LongBP	Low-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
2	LongBP	Low-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
3	LongBP	Low-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
4	LongBP	Low-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
5	LongBP	Low-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
6	LongBP	Low-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
7	LongBP	Low-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
8	LongBP	Low-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
9	LongBP	Low-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
10	LongBP	Low-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
11	LongBP	Low-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
12	LongBP	Low-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
13	LongBP	Low-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
14	LongBP	Low-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
15	LongBP	Low-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
16	LongBP	Low-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
17	LongBP	Low-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
18	LongBP	Medium-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
19	LongBP	Medium-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
20	LongBP	Medium-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
21	LongBP	Medium-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
22	LongBP	Medium-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
23	LongBP	Medium-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
24	LongBP	Medium-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
25	LongBP	Medium-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
26	LongBP	Medium-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
27	LongBP	Medium-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
28	LongBP	Medium-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
29	LongBP	Medium-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
30	LongBP	Medium-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
31	LongBP	Medium-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
32	LongBP	Medium-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
33	LongBP	Medium-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
34	LongBP	High-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
35	LongBP	High-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
36	LongBP	High-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
37	LongBP	High-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
38	LongBP	High-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
39	LongBP	High-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
40	LongBP	High-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
41	LongBP	High-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
42	LongBP	High-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
43	LongBP	High-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
44	LongBP	High-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
45	LongBP	High-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
46	LongBP	High-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
47	LongBP	High-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
48	LongBP	High-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability

Table 3.6: Experiment configurations Pt.2 (#49 - #96)

# Experiment	BP	Node Number	Node Types	Mobility	Protocols	Detection Probability
49	ShortBP	Low-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
50	ShortBP	Low-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
51	ShortBP	Low-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
52	ShortBP	Low-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
53	ShortBP	Low-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
54	ShortBP	Low-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
55	ShortBP	Low-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
56	ShortBP	Low-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
57	ShortBP	Low-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
58	ShortBP	Low-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
59	ShortBP	Low-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
60	ShortBP	Low-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
61	ShortBP	Low-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
62	ShortBP	Low-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
63	ShortBP	Low-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
64	ShortBP	Low-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
65	ShortBP	Medium-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
66	ShortBP	Medium-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
67	ShortBP	Medium-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
68	ShortBP	Medium-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
69	ShortBP	Medium-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
70	ShortBP	Medium-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
71	ShortBP	Medium-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
72	ShortBP	Medium-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
73	ShortBP	Medium-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
74	ShortBP	Medium-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
75	ShortBP	Medium-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
76	ShortBP	Medium-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
77	ShortBP	Medium-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
78	ShortBP	Medium-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
79	ShortBP	Medium-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
80	ShortBP	Medium-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
81	ShortBP	High-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
82	ShortBP	High-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
83	ShortBP	High-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
84	ShortBP	High-DensityScenario	HomogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
85	ShortBP	High-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
86	ShortBP	High-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
87	ShortBP	High-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
88	ShortBP	High-DensityScenario	HomogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability
89	ShortBP	High-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	HighDetectionProbability
90	ShortBP	High-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.11a+802.11g	LowDetectionProbability
91	ShortBP	High-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	HighDetectionProbability
92	ShortBP	High-DensityScenario	HeterogeneousNode	MostlyStaticMobility	802.15.4+802.11g	LowDetectionProbability
93	ShortBP	High-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	HighDetectionProbability
94	ShortBP	High-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.11a+802.11g	LowDetectionProbability
95	ShortBP	High-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	HighDetectionProbability
96	ShortBP	High-DensityScenario	HeterogeneousNode	MostlyDynamicMobility	802.15.4+802.11g	LowDetectionProbability

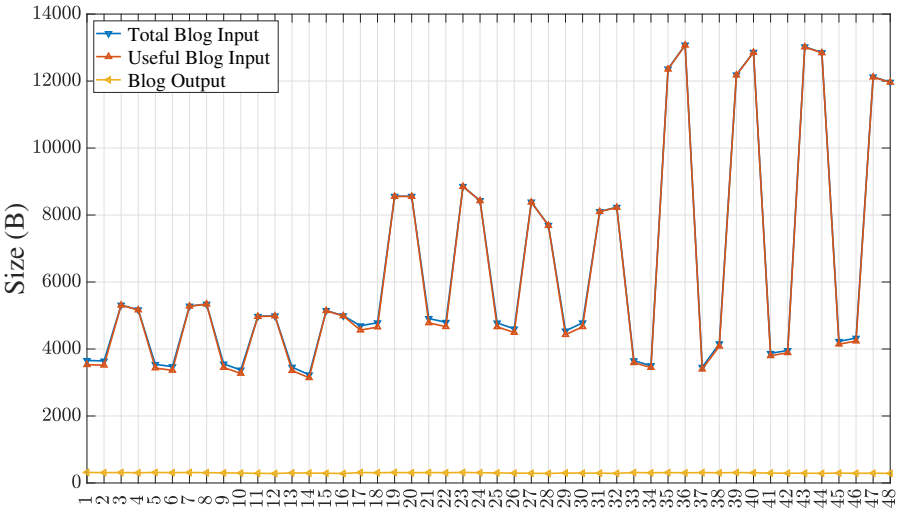
size of shared information. This can also explains why in these experiments the input bytes are less than homogeneous nodes scenarios.

Similar trends can be observed in Figure 3.15(b) and 3.16(b), showing results of experiments from #49 to #96 with short BP. It is important to recognize that in these experiments, size and number of incoming information are quite similar, regardless of the node density. Incoming packets per broadcast period are always below 18 and total size below 6500 B, obviously lower than those collected over a longer BP.

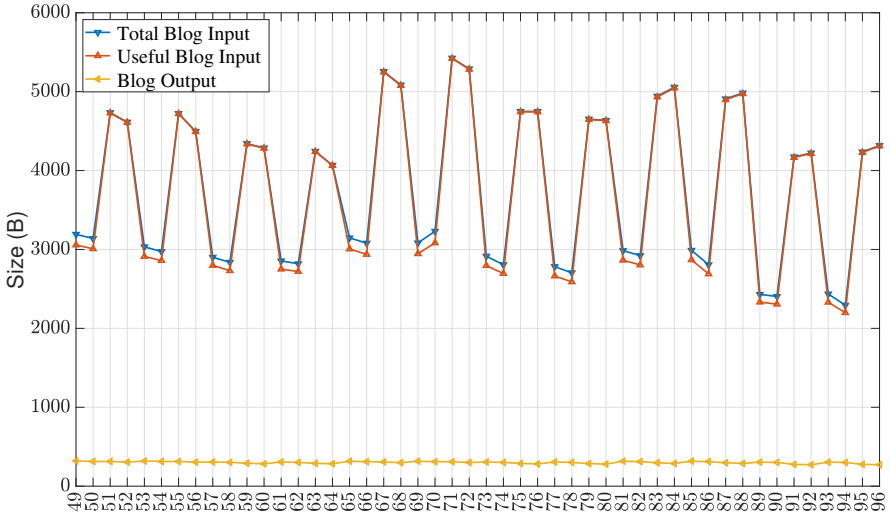
Bandwidth gains are plotted in Figure 3.17(a) and 3.17(b) and confirm what highlighted above. The gain is between 3.5 and 7 thanks to the Concept Fusion algorithm ensuring the nodes output significantly shorter than the input data in all scenarios. Heterogeneous node scenarios are characterized by lower gain values because in these experiments less input information is managed, for the reasons explained above. The same conclusion can be reached for low detection probability experiments, as well as those adopting 802.11a and 802.11g protocols together.

Activation Load

Activation Load input packets are strictly related to the provided actuators. Obviously nodes without actuation capabilities can only send unicast messages for service activation, without receiving any. The weighted average values are reported by byte and packet perspective in Figure 3.18(a) and Figure 3.19(a) respectively for long BP, and in Figure 3.18(b) and Figure 3.19(b) for short BP. Results show Activation Load is closely correlated with the detection probability configuration: with low detection probability, reaching decisions is harder for agents and fewer service requests are sent and received. This is presumably the reason why the incoming bytes and packets are lower than in other experiments. It can be observed that among all scenarios the worst performance occurs when 802.11a and 802.11g protocols are combined and low detection probability is set. Furthermore, the total incoming requests decrease for scenarios with larger numbers of nodes. This is related to an increased number of available services in the environment, and so a decreased likelihood to be selected as provider. There are no substantial differences in number and total size of managed packets between long and short BP scenarios. In this last case, experiments report higher values of incoming bytes and packets, mainly due to an increased packets sending rate. The output activation load follows a similar behaviour showing that a node on average sends 0.6 activation requests corresponding to 15 bytes.

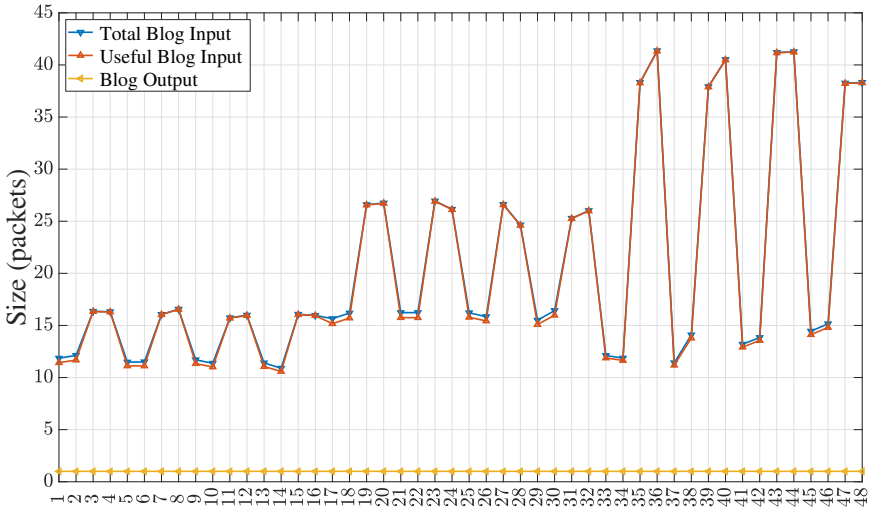


(a) Experiments from #1 to #48

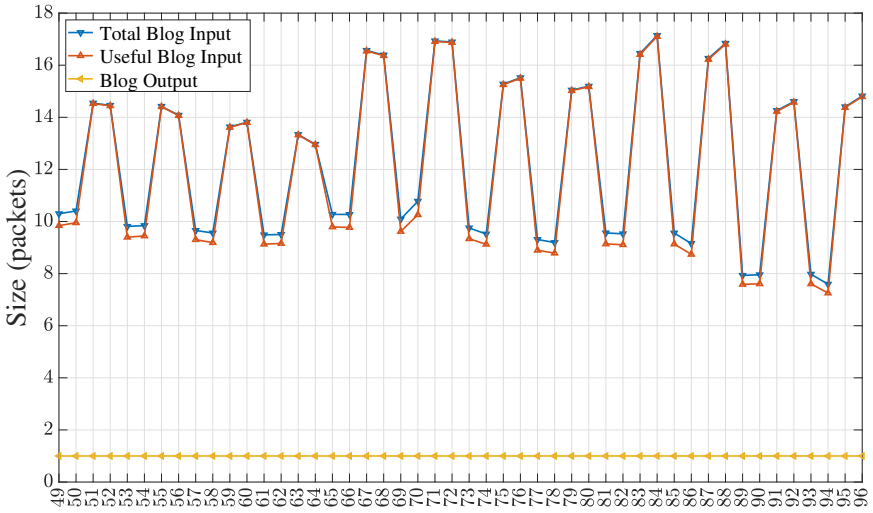


(b) Experiments from #49 to #96

Figure 3.15: Blogging Load (bytes)

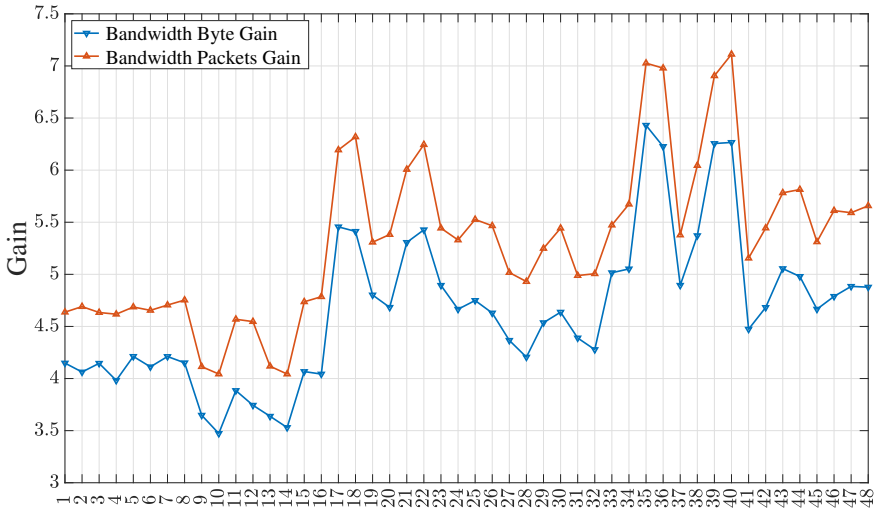


(a) Experiments from #1 to #48

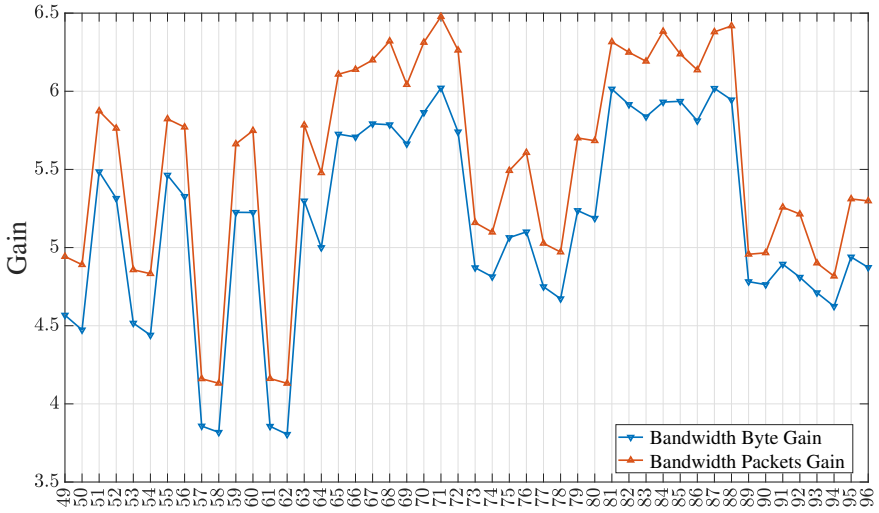


(b) Experiments from #49 to #96

Figure 3.16: Blogging Load (packets)

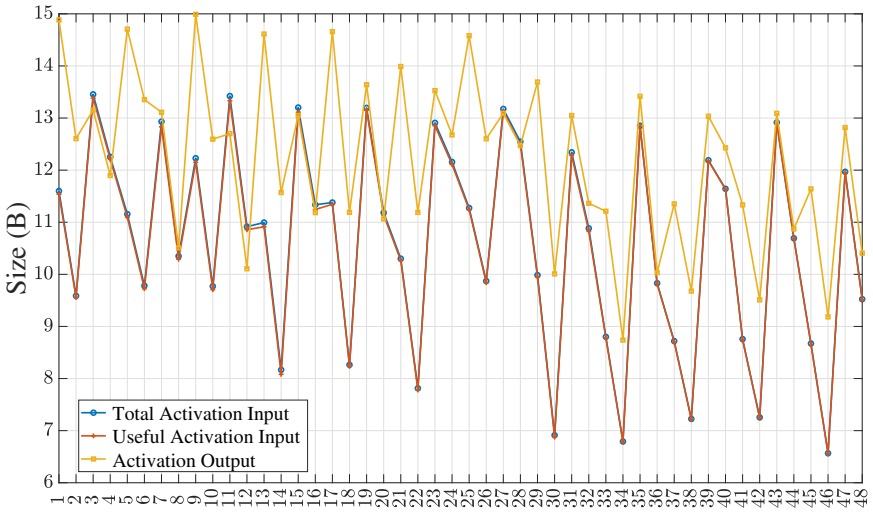


(a) Experiments from #1 to #48

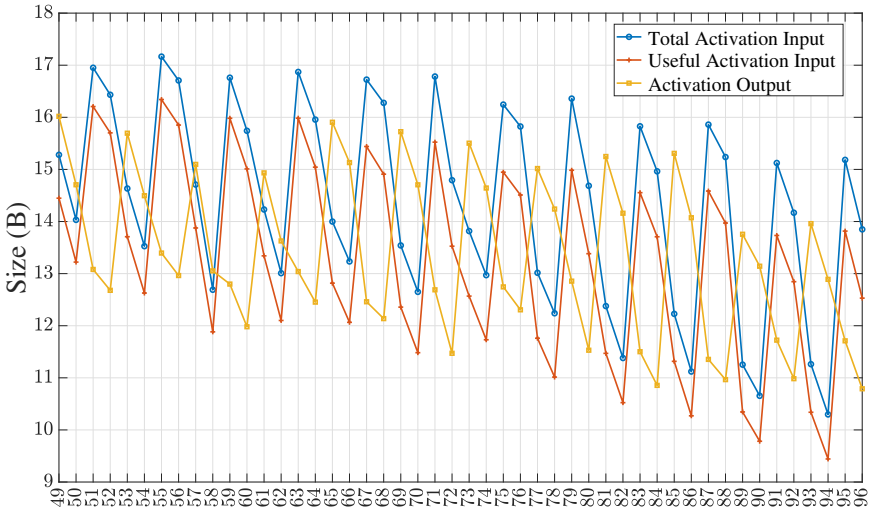


(b) Experiments from #49 to #96

Figure 3.17: Bandwidth gain

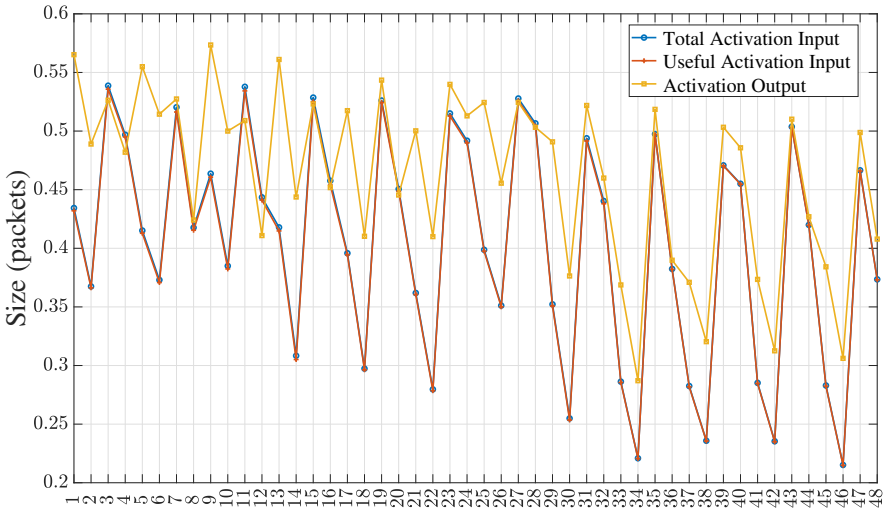


(a) Experiments from #1 to #48

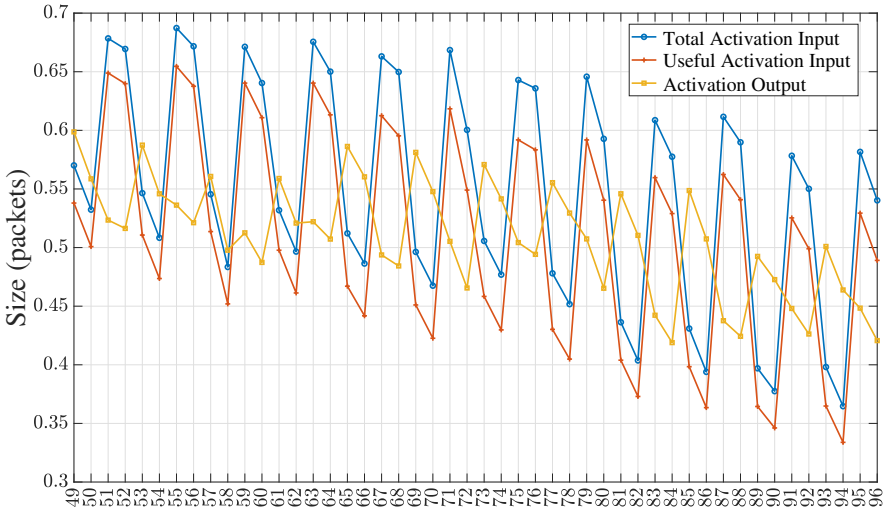


(b) Experiments from #49 to #96

Figure 3.18: Activation Load (bytes)



(a) Experiments from #1 to #48



(b) Experiments from #49 to #96

Figure 3.19: Activation Load (packets)

Reasoning Time

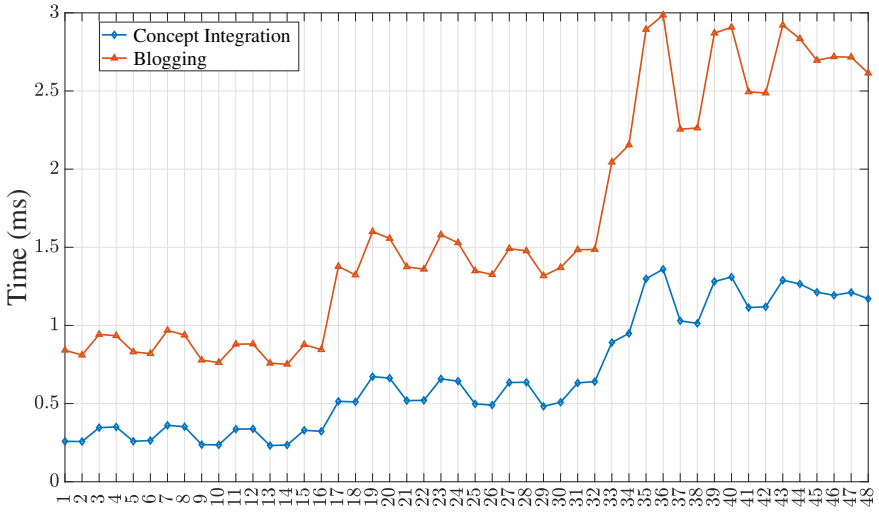
As expected, Figure 3.20(a) and Figure 3.20(b) show that reasoning times are closely related to number and size of collected packets, as well as to node density in the environment. In long BP scenarios, both CI-v2 and overall blogging time are affected by the adoption of IEEE 802.15.4 or 802.11a protocols. Indeed, in the latter case times are lower due to the fact that a lower number of packets is managed. It is also possible to observe that heterogeneous node scenarios involve less information and so time is slightly lower. A similar behaviour can be observed in short BP scenarios, where times are lower in all experiments because time windows in which each node can collect packets are shorter, therefore less information is cached. CI-v2 time can be deemed as low, being always below 1.5 ms and 0.7 ms in the worst case respectively for long and short BP. Analogously, the overall blogging time requires always less than 3 ms and 2 ms respectively demonstrating the sustainability of the approach.

Fusion Performance

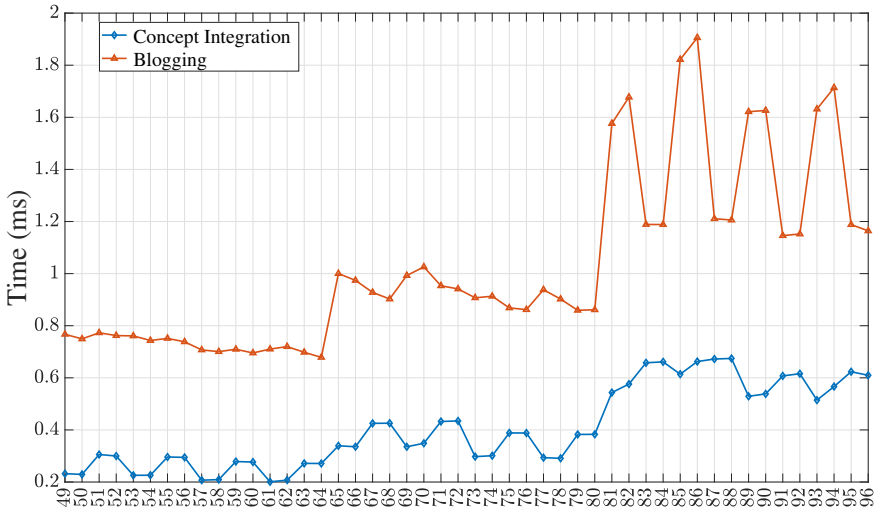
In order to evaluate the Concept Fusion performance, an overview of the detected and received knowledge covering w.r.t. the real environment description is provided in Figure 3.21(a) and Figure 3.21(b). As expected, in all scenarios with low detection probability, the detected knowledge covering is lower, while the conflicting percentage is higher. Furthermore, heterogeneous nodes gather only a subset of the context features: in those most realistic scenarios the related knowledge covering is limited.

It can be noticed the overall detected knowledge is mainly kept in the C and/or E fields of the cached packets, according to the similarity of the involved nodes. However, a non-negligible percentage is associated to the M description. C and E also store the highest conflicting knowledge percentages, which further grows in the C field of heterogeneous node experiments due to a slight increased number of erroneous fusion. These results are encouraging, because they confirm the effectiveness of the integration approach. Moreover, in short BP experiments there is less knowledge in the M field than in the long BP ones, and the percentages of covering of C and E are better. The shared conflicting knowledge is also lower, suggesting a better fusion effectiveness in those scenarios.

Figure 3.22(a) and Figure 3.22(b) show the Concept Integration v2 outcomes. As depicted, the log *i.e.*, the conjunctions of C, M, and E, covers over 90% of ground truth knowledge. Values are closely related to the detection probability settings. It can be observed that the performance is good



(a) Experiments from #1 to #48



(b) Experiments from #49 to #96

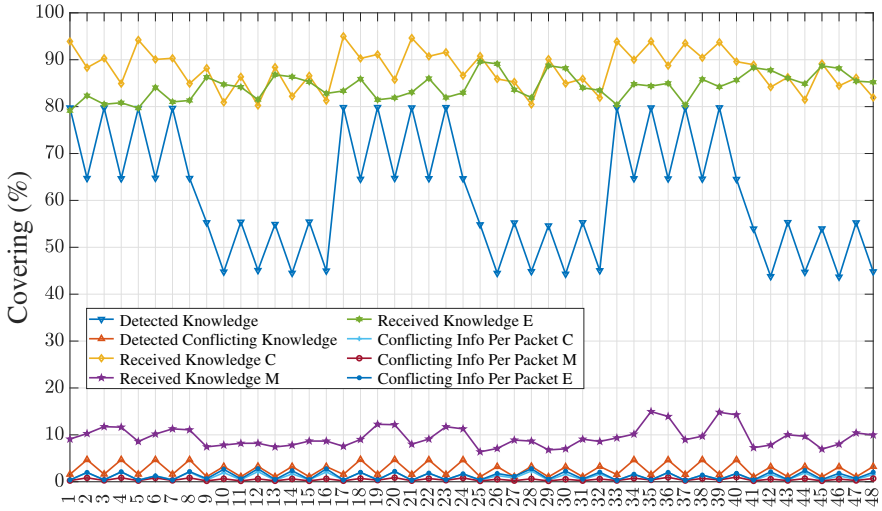
Figure 3.20: Reasoning Time

also in experiments involving heterogeneous nodes, and it further improves in short BP scenarios, confirming the capability of the proposed approach to merge several seemingly dissimilar perspectives. The integrated knowledge is mainly associated to the C field, while E takes greater relevance in heterogeneous node experiments, compensating for the decreased percentage of confirmed observations. A small knowledge percentage is stored in M field. Furthermore, it can be observed that the integrated knowledge could be erroneously interpreted as incorrect and stored in the X field, but related percentage can be deemed as very low and it further improves in short BP experiments.

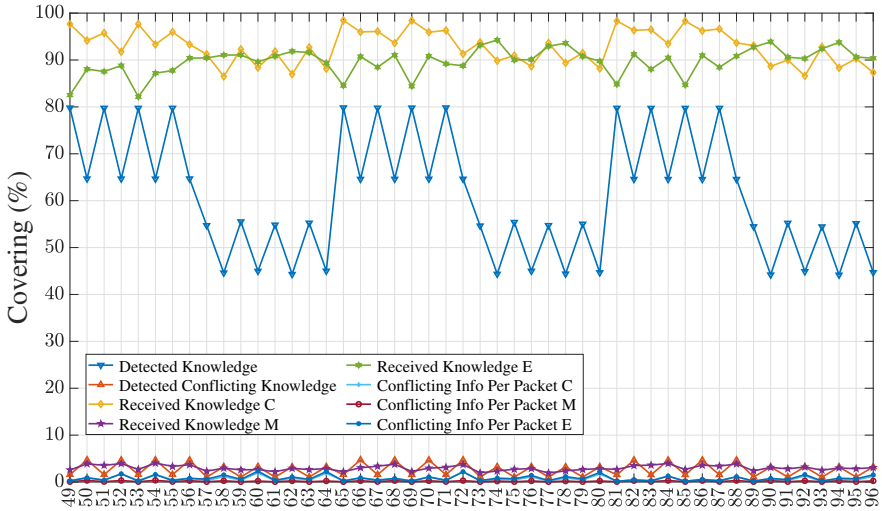
Conversely, the integrated conflicting knowledge covering results is plotted in Figure 3.23(a) and Figure 3.23(b). As expected, the discarded field X reports the biggest conflicting knowledge percentage. Unfortunately, in experiments involving heterogeneous nodes, less information is collected and detecting conflicting observations is quite difficult. For this reason, E field keeps a non-negligible percentage of incorrect knowledge, though always below 2% in long BP tests and 1.5% in short BP ones. Non-zero quotas are associated also to C and M. It can be noticed that performance improves with larger node density scenarios, thanks to an increased amount of managed information allowing to correctly integrate the knowledge. Furthermore, short BP experiments demonstrate a better resilience against conflicting information with bigger percentage of conflicting knowledge properly discarded in the X field. The short time window grants quick adaptation to changes and robustness against spurious or inaccurate information by managing updated and comprehensive knowledge in fewer cached packets.

Decision Performance

Figure 3.24(a) and Figure 3.24(b) depict the decision performance in long and short BP tests, respectively. The best results have been reached in the latter case, with a covering percentage w.r.t. the description of the needed action always over 90%. Even in the worst cases in long BP, it is never below 80%. Furthermore, decisions in contrast with the required action are nearly 0 in all experiments. Nodes avoid to reach –potential wrong– resolution after matchmaking if all the KB interventions have semantic relevance scores lower than the given threshold. This demonstrates the capability of each node to enable adaptive context-aware behavior triggering intervention and/or making the right decision according to the inferred knowledge of the environment.

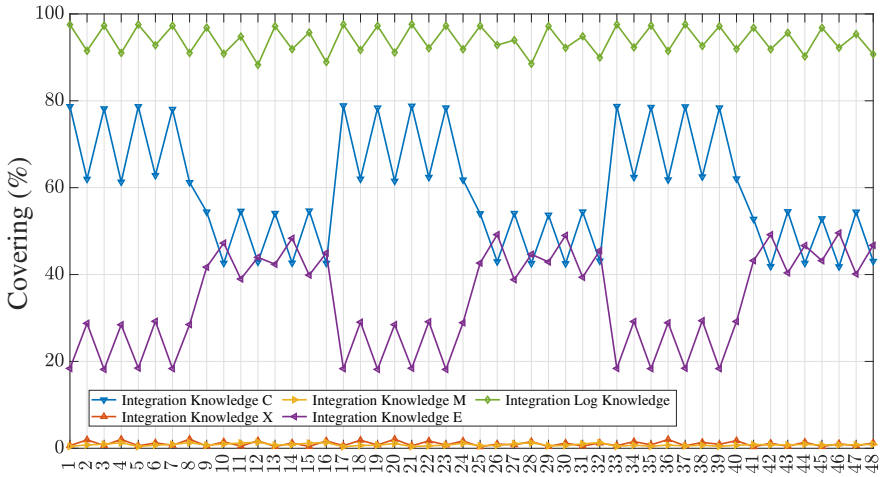


(a) Experiments from #1 to #48

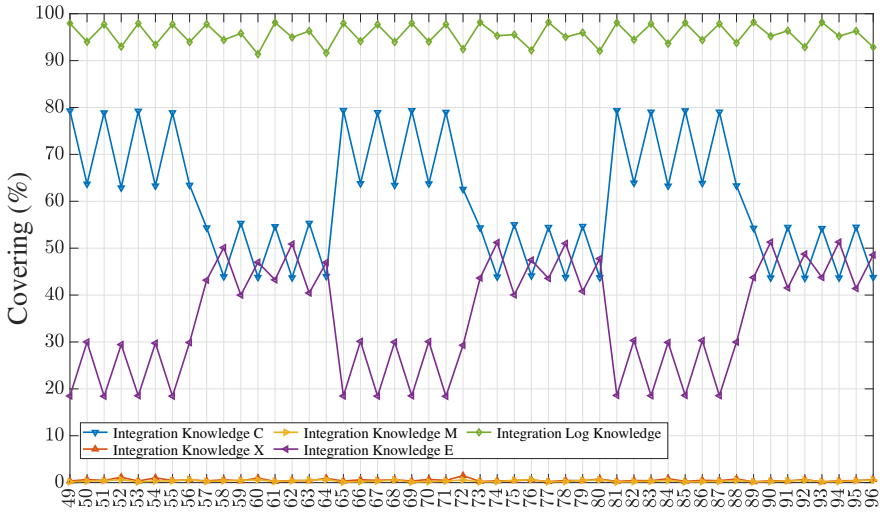


(b) Experiments from #49 to #96

Figure 3.21: Integration Performance - Input Knowledge Covering

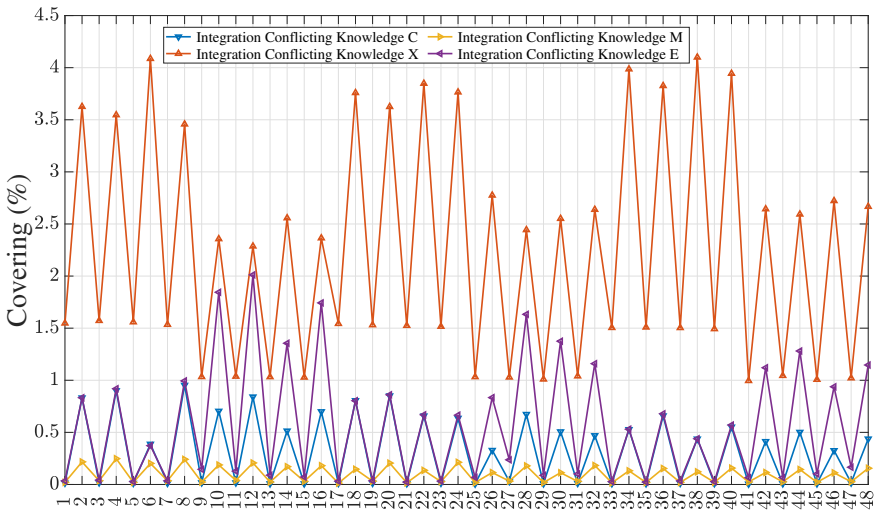


(a) Experiments from #1 to #48

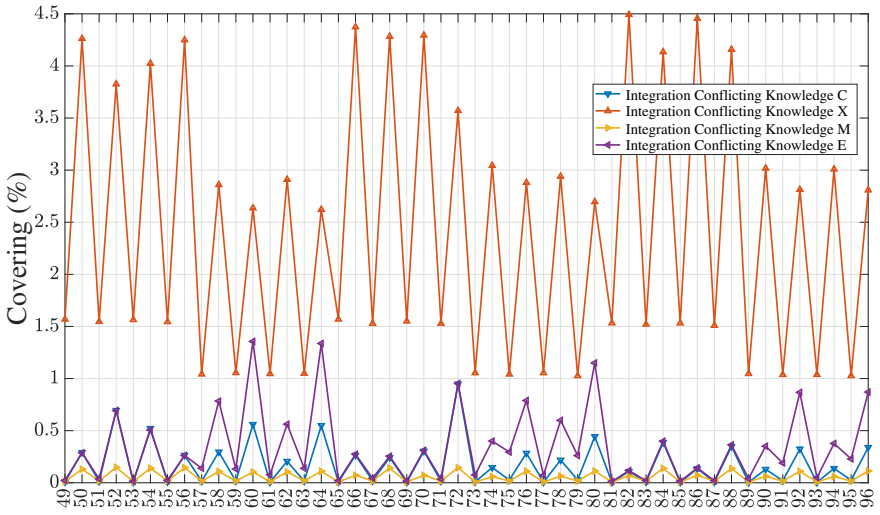


(b) Experiments from #49 to #96

Figure 3.22: Integration Performance - Integrated Knowledge Covering

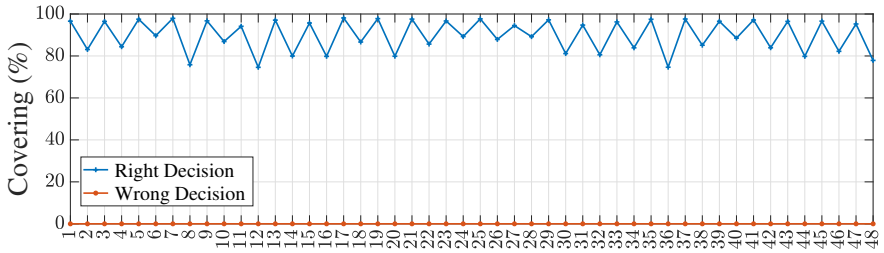


(a) Experiments from #1 to #48

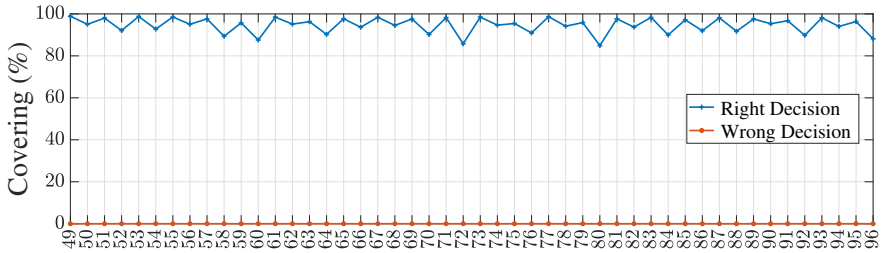


(b) Experiments from #49 to #96

Figure 3.23: Integration Performance - Integrated Conflicting Knowledge Covering



(a) Experiments from #1 to #48



(b) Experiments from #49 to #96

Figure 3.24: Reasoning Time

Chapter 4

Social capabilities for smart objects

This chapter outlines the proposal for social things intelligence and knowledge-based cooperation. A service-oriented architecture is defined, where every device can expose and request *services* (*i.e.*, functionalities). Both device and service profiles are semantically annotated w.r.t. ontologies. According to the high-level SIoT architecture outlined in [114], each device manages both social relationships with peers on the network and context information obtained through data mining on embedded sensor readings. When a device detects a context change which requires a reconfiguration, it issues a request on the social network. Then a collaborative discovery process starts, exploiting semantic matchmaking to retrieve the best available services from the network for covering the request.

4.1 Social awareness in the Semantic Web of Things

The envisioned approach aims at object self-organization in complex contexts through interaction paradigms borrowed from *social networks*. Both interaction protocol and object self-awareness give devices extensive agency and autonomy capabilities, enabling them to share information, publish requests, and receive responses in a controlled interchange.

A decentralized service-oriented architecture underlies the whole proposed social network model, where shared knowledge fragments about devices, functional profiles and context represent annotated service/resource advertise-

ments. They are described exploiting both several well-known RDFS vocabularies and an additional ontology, named *Semantic Web of Social Things* (SWST)[1], developed within this work to model basic elements of a social network. The inference services depicted in Section 2.2.3 are used to regulate the interactions between the nodes (*i.e.*, things). Among them, we can distinguish (i) *Smart* devices, able to execute inference tasks; (ii) *Basic* devices, endowed with low memory and processing capabilities, only providing sensing/acting services without performing autonomous reasoning.

The way people interact and cultivate relationships through *Social Network Services* was adapted to *social objects* as follows:

1. A social object posts on its *wall* an individual profile, describing its basic characteristics (device type, location, hardware details) and the resources/services it is able to provide, *e.g.*, different functional profiles and configurations it can take.

2. A pair of social objects can establish two basic kinds of social relationships:

   - *Friend*, a bidirectional relationship where nodes N_i and N_j can exchange both information and services. In particular, a device N_j sends a *friendship* request; since the receiver N_i accepts it, they became able to: (i) read and write on each other's wall; (ii) request the friend's service descriptions; (iii) activate or deactivate the friend's services. A basic node, when becoming friend with a smart node, can also select it as semantic facilitator *i.e.*, reasoning supporter.
   - *Follower*, a unidirectional relationship where a node N_k is interested only in receiving the updates published by N_i on its wall. In other words, if N_k sends a *follower* request to N_i, N_k becomes an observer of N_i's behavior. This kind of relationship is useful in case of the information produced by N_i can be useful to N_k to characterize its own context but there is low utility in a direct interaction, *e.g.*, the two devices are deeply different.

   This mechanism grants effortless system configuration and high scalability since enabled interactions are neither decided *a priori* nor by a centralized entity.

[1]Semantic Web of Social Things (SWST) Ontology: `http://sisinflab.poliba.it/swottools/onto/swst/`

3. Each social object posts updates on its *wall* when its settings or capabilities change, as well as information produced through context sensing and analysis. A *post* on a friend's wall is interpreted as a request, instead, triggering a reconfiguration of the friend device in order to best adapt to the new situation.

4. *Tagging* is used to ask a device to activate or deactivate a specific service.

5. A *like* is used as confirmation to a post at the end of the reorganization process

At the application layer, the proposed framework is implemented through the *Linked Data Notifications* (LDN) protocol [22], on top of the *Linked Data Platform* (LDP) [80] mapping [79] for the *Constrained Application Protocol* (CoAP) [127], a lightweight HTTP-like protocol for the Web of Things. Each agent in the social network is modeled as an LDP-CoAP node exposing the interface detailed in Table 4.1.

A notable aspect of the framework is that the social network is completely distributed. No central platform is needed to mediate interactions. Every object stores locally: the list of its friends, each characterized by identity and addressing information; its own wall; a cache of posts on friends' walls. Likes are stored along with the post they refer to.

Cooperative service/resource discovery

The adopted service discovery and composition is grounded on semantic matchmaking, exploiting the non-standard inference services recalled in Section 2.2.3. Whenever a device requires new services or changing current settings, its request is formalized as a DL annotation R. The proposed framework does not constrain *when* a device should issue a request, neither *how* to derive a proper request annotation. As outlined in Section 2.1.1, smart objects are expected to execute data mining and inference on perceptions collected from on-board or nearby sensors in order to characterize the environment and detect relevant events and changes. When the request is formulated, a cooperative multi-hop discovery process starts, using Concept Covering to satisfy it as much as possible through aggregation of available services/resources associated to the same reference ontology as the request.

Service discovery is collaborative and recursive, exploiting service descriptions shared between friends and the possibility to write posts and comments on each other's wall as the example reported in Figure 4.1 clarifies. Three devices are considered: D_1 is the requester and D_2 is friend to both D_1 and

Table 4.1: LDP-CoAP interface of a social device

Resource URI	Resource Type	LDP-CoAP Method	Description
/profile	LDP-RS	GET	Returns the device profile
/friendship	LDN Inbox	GET	Returns the list of friend devices
		POST	Receives a friendship request
/friendship/<device-name>	LDP-RS	GET	Returns the profile of a specific friend device
/services	LDP-BC	GET	Returns the list of the services exposed by the device
/services/<service-name>	LDP-RS	GET	Returns the RDF-based description of a specific service
		PATCH	Updates the status of a service according to the received command
		HEAD	Returns the LDP-CoAP headers (*e.g.*, *Etag* value) to check the presence of updates
/services/<service-name>/owl	LDP-NR	GET	Returns the OWL annotation related to a specific service
/wall	LDN Inbox	GET	Returns the device wall, containing the list of published posts
		POST	Publishes on the wall a post received from a friend device
/wall/<post-id>	LDN Inbox	GET	Returns the detailed description of a specific post
		POST	Publishes on the wall a comment related to a post
/wall/<post-id>/owl	LDP-NR	GET	Returns the content of a post as OWL annotation
/wall/<post-id>/<comment-id>	LDP-RS	GET	Returns the description of a comment
		PATCH	Tag a device service on a comment
/wall/<post-id>/<comment-id>/owl	LDP-NR	GET	Returns the content of a comment as OWL annotation

D_3. When D_1 generates its request R_1, it will post it on its wall (post P_1). A local covering process starts, where D_1 exploits its embedded matchmaking engine to cover R_1 as much as possible with the services it can access: this set is formed by D_1's own services and the ones belonging to D_1's friends, which were read via LDP-CoAP at friendship establishment: the CoAP Observe option [45] is adopted to notify the device whenever a friend's service set changes. The covering result consists in a set of selected services and possibly the uncovered part U_1 of the request. This information is appended to P_1 as comment C_1 on D_1's wall; furthermore, the covering degree –in $[0, 1]$ range, with 0 denoting "no cover" and 1 "full cover"– is associated to P_1 as the *like* value L_1.

If $L_1 < 1$ (hence, there is some uncovered part U_1), then D_1 picks some friends to help find further services. Selection criteria include the number of friend's services referring to the same ontology of the request and historical performance data concerning the number and relevance of retrieved services, as well as communication latency and bandwidth. In the reference figure, D_1 picks D_2 and POSTs a new post P_2 on D_2's wall, attaching U_1 as the new request. D_2 computes semantic matchmaking using its friends' service descriptions, while excluding its own services as they have already been considered by D_1. The new covering process yields (possibly) further selected services, a new (smaller) uncovered part U_2 and an updated (higher) like value. D_2 adds this result as a comment to P_2. Complying with LDN, this update is notified to D_1 through the Observe CoAP pattern[2]. Then D_1 GETs the content of the comment updates the like value associated to its own original request on its own wall. Supposing the request has not been fully covered yet, D_2 repeats the process recursively, writing a new post P_3 on its friend D_3's wall (which is not a friend of D_1). D_3 executes matchmaking with cached service descriptions of its friends (except D_2). The outcome leads to a comment and updated like value on P_3: they are notified to D_2, which GETs the content of the comment and reports it as a new comment C_2 on post P_2 in its own wall. This in turn activates notification for D_1, which GETs the comment. Finally, it updates its original post P_1 with the updated set of selected services, uncovered part and like value. The process can go on until (a) the request is fully covered, or at least reaches a minimum threshold l_{min} of like (*i.e.*, satisfaction) value, or (b) a maximum distance d_{max} from the request source is reached in the social graph. Both l_{min} and d_{max} are

[2]Actually, a node writing a post on a friend's wall is automatically added to the list of CoAP observers for the post, so that it is notified when a comment is added. This is an extension of the standard CoAP Observe pattern, which is normally available for GET requests only; otherwise, after a POST on a friend's wall, a further GET request would be needed to start observing.

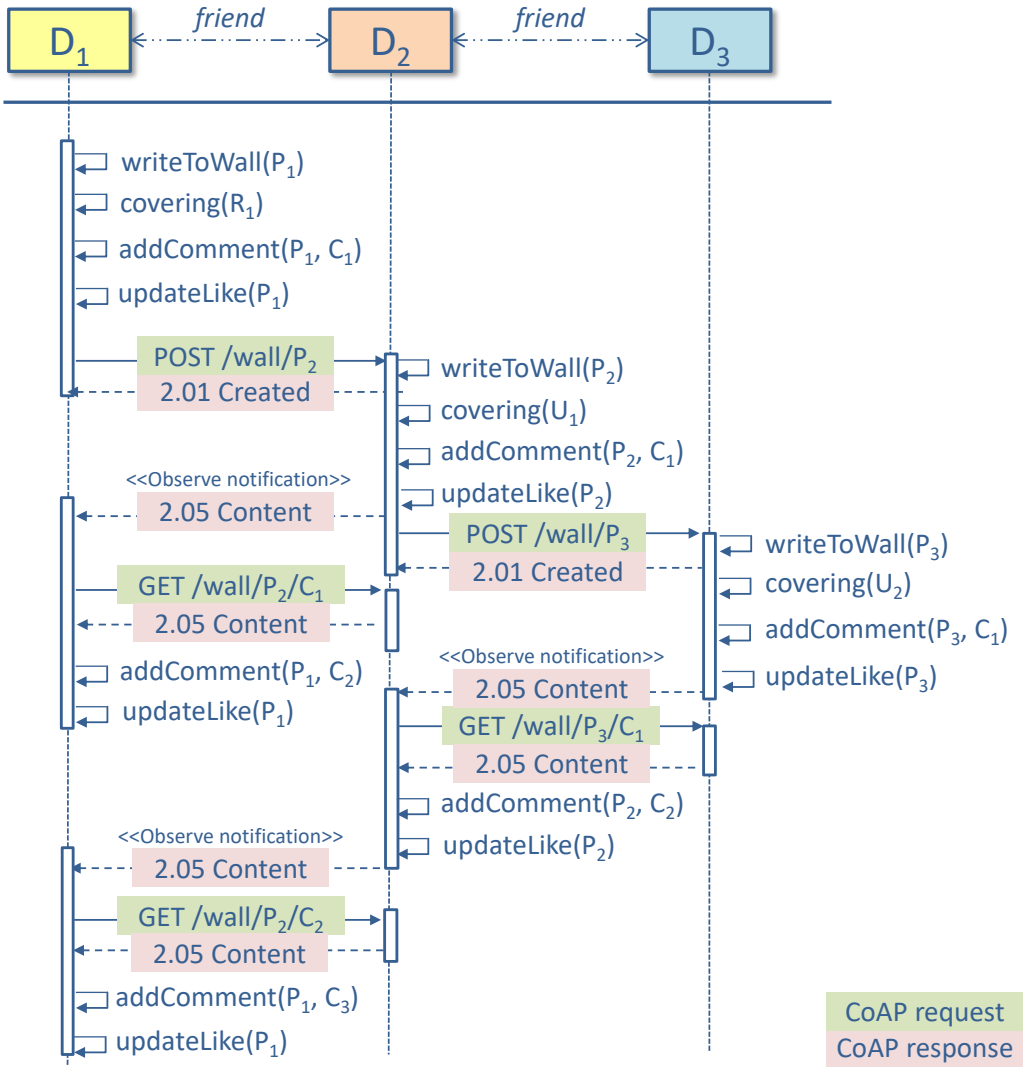


Figure 4.1: Distributed service discovery

tunable parameters.

Selected services are activated at the end of each discovery step by *tagging* them. A discovery session produces, for each participating device D, exactly one wall post, with a number of comments equal to the devices D has directly or indirectly involved in the matchmaking process. Devices can exploit them as a log to adapt their relationships dynamically, calculating metrics of usefulness for each friend.

Managing dynamic social relationships among objects

This work extends the framework described in [115] to endow social objects with proactive adaptivity to environmental modifications in order to improve self-organization through social relationships refinement. Based on past experience (*i.e.*, previous interactions over the social network), each object is able to adapt its relationships dynamically according to context changes, in order to improve not only the processing on the single node but also the overall network performance. Removing unnecessary relationships aims to reduce network traffic *e.g.*, less messages will be forwarded during a service discovery process. Conversely, new meaningful interactions can increase the network effectiveness and reduce the relative computational load per node: a request could be satisfied with lower processing time and activating the minimum set of services. The following models are proposed to suggest and remove friendship relationships based on the experience of each social object.

Suggestion. After every time period T, each smart device queries its wall to retrieve the list of devices that sent a comment to or were tagged in a post. For each such device D, the *Suggest* function is computed as follows:

$$Suggest(D) = [w \cdot C(D) + (1 - w) \cdot U(D)] \cdot \frac{X - 1}{X}$$

with

$$C(D) = \sum_{i}^{N_c(D)} X^{(t_i - T)} \qquad U(D) = \sum_{j}^{N_u(D)} X^{(t_j - T)}$$

The following symbols are used in the above formulae: (i) $N_c(D)$, number of comments created by D; (ii) $N_u(D)$, number of useful services provided by D, *i.e.*, D's services tagged as result of Concept Covering; (iii) w (between 0 and 1), used to weigh the contribution of $U(D)$, considering a device as direct provider of useful services, and $C(D)$, representing the ability of D to contribute to service discovery by acting as a "bridge" toward other useful devices: (iv) T, timestamp at which all functions are evaluated; (v) $t_{i/j} \leq T$, timestamp at which a comment/post was created; (vi) $X > 1$, decay rate

coefficient, weighing the past history of a device: the higher the value, the lower the relevance of older contributions. Finally, division by $\frac{X}{X-1}$ normalizes $S(D)$ w.r.t. the theoretical convergence value of the geometric series with ratio $\frac{1}{X}$, implying that $0 \leq S(D) \leq 1$. According to the proposed model, a device will be suggested as a good candidate friend if it either provides services frequently or it is a friend of many useful devices. New friendship requests will be sent to devices with an S score higher than a reference threshold (*TH-suggest*). Considering the example in Figure 4.2(a), if device D_1 frequently activates services of D_5 through intermediate friends D_2 and D_4, it will learn and ask for D_5's direct friendship as in Figure 4.2(b).

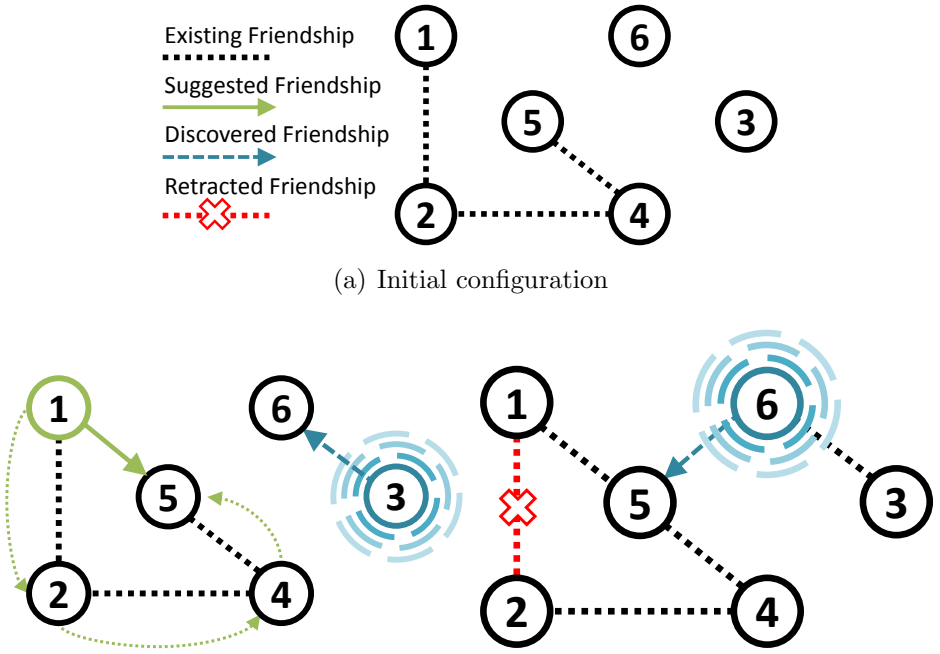
Retraction. Along with the suggestion of new friends, a *retraction* score $R(D)$ is calculated for each friend device D to identify worthless relationships:

$$R(D) = S(D) - P(D)$$

$$P(D) = [\sum_{k}^{N_i(D)} X^{(t_k - T)} \cdot P_k(D)] \cdot \frac{X-1}{X}$$

where $P(D)$ exploits past results of local Concept Covering processes. In detail, $N_i(D)$ is the number of D's services, detected as incompatible w.r.t. the requests received in the period T; $t_k \leq T$ is the timestamp at which the covering process was performed and the incompatibility was identified; $P_k(D) \in [0, 1]$ is the penalty score measured by Concept Contraction. For each friend device, if $R(D)$ (which is always in $[-1, 1]$) is less than a *TH-retract* threshold, the friendship will be removed: the meaning is that D often provides conflicting services (wasting processing and communication resources) and/or it does not have a useful impact in satisfying received requests. The example in Figure 4.2(c) shows device D_1 breaking its friendship with D_2.

Satisfaction. In loosely connected networks, social devices could be without any friends, *e.g.*, due to a "cold start" configuration (as shown in Figure 4.2(a)) or frequent friendships retractions. In the protocol described above isolated objects are unable to both establish new friendship relations and cooperate. In order to overcome this problem, a *satisfaction* indicator, based on likes, identifies and solves object deadlocks. Periodically, each device D evaluates its satisfaction degree w.r.t. received or generated requests



(a) Initial configuration

(b) Suggested friendships

(c) Retracted friendship

Figure 4.2: Sample network with loosely connected nodes

by means of the formula:

$$Sat(D) = [\sum_{h}^{N_p(D)} X^{(t_h - T)} \cdot like_h(D)] \cdot \frac{X-1}{X}$$

where $N_p(D)$ is the number of posts (*i.e.*, requests) on D's wall and $t_h \leq T$ represents the timestamp at which the $like_h(D)$ value related to the post was calculated. Also in this case, $Sat(D)$ is compared with a reference threshold (*TH-satisfaction*) and devices with a lower score will perform a further device discovery process, as during the device initialization. In Figure 4.2(b), isolated device D_3 finds D_6 and they establish friendship; then in Figure 4.2(c) D_6, still having a $Sat(D_6) <$ *TH-satisfaction*, links to D_5 too.

4.2 Illustrative example

A case study regarding power management of *plug-in electric taxis* (PETs) in a *smart grid* is discussed to highlight capabilities of the proposed framework in dynamic environments. Let us consider the following example:

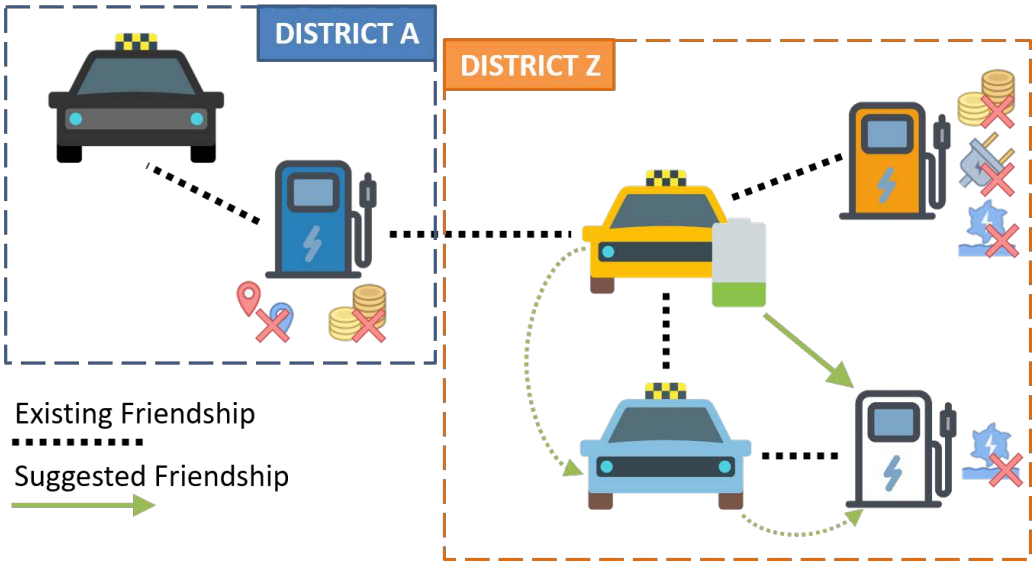


Figure 4.3: Social smart grid scenario

A 2017 BMW i3 plug-in electric taxi usually serves district A *of a large city. It has friendship relations with vehicles of its PET fleet company and some of the charging stations in the smart grid.*

Figure 4.3 depicts the envisioned scenario: each taxi and charging point is a social object, which exposes toward its friends an individual profile, describing its basic features and the services it can provide.

Figure 4.4 depicts a possible formalization of electric vehicle power management aspects (reported in OWL 2 Manchester syntax for the sake of readability) w.r.t. the reference domain ontology. A relevant excerpt of the main concept taxonomy defined for the case study is in Figure 4.5; in addition to hierarchical relationships, concepts and role expressions are exploited to build the complex definitions in the following figures.

```
BMW_i3_2017 SubClassOf: ElectricVehicle and (hasACCompatiblePlug only
VDE-AR-E2623-2-2Type2Plug) and (hasDCCompatiblePlug only CCSCOMBO2Plug)
and (hasEVSEProfile only ((hasOutputRate only (daW max 74)) and
(hasCurrent only (Ampere max 32)) and (hasVoltage only (Volt max 230))))
and (hasACCompatiblePlug some) and (hasDCCompatiblePlug some)

ElectricVehicle SubClassOf: (hasEVSEProfile some) and (hasEVSEProfile only
((hasOutputRate some) and (hasVoltage some) and (hasCurrent some)))
```

Figure 4.4: Electric taxi profile semantic description

Notice *2017 BMW i3* electric car model adopts a standard *VDE-AR-E*

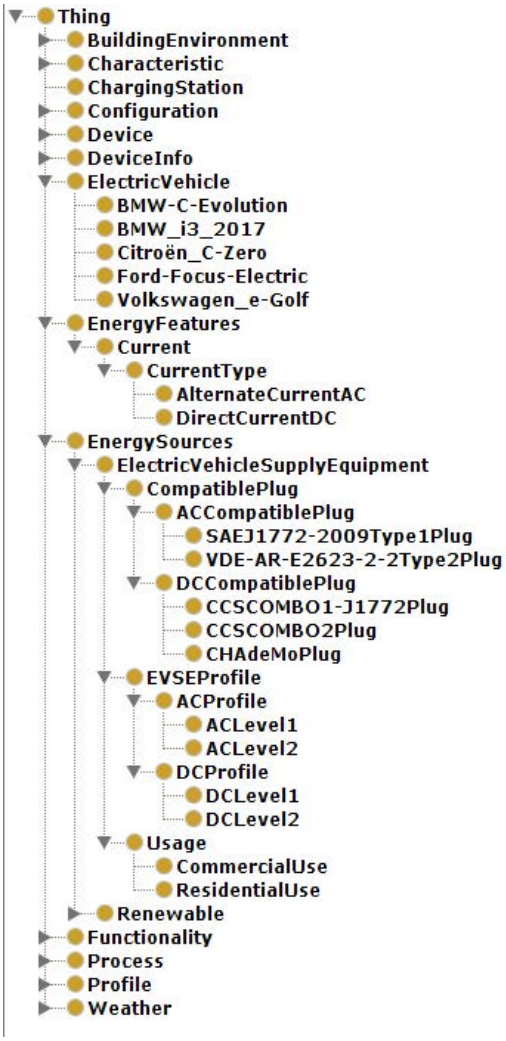


Figure 4.5: Smart grid domain ontology

2623-2-2 (Type 2) *a.k.a. Mennekes* connector and receptacle for AC charging, incompatible with *SAEJ1772* (Type 1). Also Fast DC charging currently has many standard connectors. *CHAdeMO*, *CCS Combo1* or *CCS Combo2* receptacles can be found on different chargers and are incompatible with each other; i3 supports CCS Combo2 plug only. Figure 4.6 shows modeled charging profiles, according to existing solutions for electric vehicle supply equipment (EVSE).

```
EVSEProfile SubClassOf: (hasCurrent some) and (hasCurrentType some) and
(hasOutputRate some) and (hasVoltage some) and (isSuggestedForUse some)
and ElectricVehicleSupplyEquipment

ACProfile SubClassOf: EVSEProfile and (hasCurrentType only
AlternateCurrentAC)

ACLevel1 SubClassOf: ACProfile and (hasCurrent only (Ampere max 20)) and
(hasOutputRate only (daW max 17)) and (hasVoltage only (Volt max 120)) and
(isSuggestedForUse only ResidentialUse)

ACLevel2 SubClassOf: ACProfile and (hasCurrent only (Ampere max 100)) and
(hasOutputRate only (daW max 192)) and (hasVoltage only (Volt max 240))
and (isSuggestedForUse only CommercialUse)

DCProfile SubClassOf: EVSEProfile and (hasCurrentType only
DirectCurrentDC)

DCLevel1 SubClassOf: DCProfile and (hasCurrent only (Ampere max 80)) and
(hasOutputRate only (daW max 400)) and (hasVoltage only (Volt max 500))
and (isSuggestedForUse only CommercialUse)

DCLevel2 SubClassOf: DCProfile and (hasCurrent only (Ampere max 200)) and
(hasOutputRate only (daW max 1000)) and (hasVoltage only (Volt max 500))
and (isSuggestedForUse only CommercialUse)
```

Figure 4.6: EVSE charging profile classes in the ontology

A customer requests to be taken to district Z, *seldom frequented by the taxi. After leaving the customer, PET informs taxi driver that battery level is below 50% and recharge is needed.*

The taxi formulates the semantic-based recharge request depicted in Figure 4.7, corresponding to a charging rate of at least 25 km per charging hour, minimum available energy of 20 kWh at charging station and maximum distance of 15 km between taxi and charging station and 30 km between charging station and dispatcher, and maximum price of 50 cents per kWh. This is used in a Concept Covering task (Section 2.2.3) on the locally cached set of services, *i.e.*, functionalities exposed by all direct friends of the taxi. Unfortunately those service instances have some conflicting features, as shown in Figure 4.8, and cannot be selected. In detail, *ChargingStationdistrictA* is

too far from the current taxi position (*hasDistance*), because it is located in district A, and it is too much expensive (*hasFeePerkWh*). In *ChargingStationdistrictZ*, DC charger equipment is *CHAdeMO* (*hasDCCompatiblePlug*) which is not suitable to charge *i3* taxi. Furthermore the price per kWh and the distance between charging station and dispatcher exceed the desired one. According to Section 4.1, taxi periodically computes *Retract R(D)* scores for each friend. If *ChargingStationdistrictZ* service of device D still remains incompatible with taxi needs, D's score will progressively decrease until friendship breaks up when it becomes less than the given *TH-retract* threshold.

```
TaxiRechargeRequest EquivalentTo: BMWi32017 and (hasChargingRatePerHour
some) and (hasAvailablePower some) and (hasDispatcherDistance some) and
(hasDistance some) and (hasFeePerkWh some) and (hasChargingRatePerHour
only (km min 25)) and (hasAvailablePower only (kWh min 20)) and
(hasDispatcherDistance only (km max 30)) and (hasDistance only (km max
15)) and (hasFeePerkWh only (cents max 50))
```

Figure 4.7: Semantic annotation of taxi request

```
ChargingStationdistrictA EquivalentTo: ChargingStation and
(hasACCompatiblePlug only VDE-AR-E2623-2-2Type2Plug) and
(hasChargingRatePerHour only (km exactly 45)) and (hasAvailablePower
only (kWh exactly 500)) and (hasDispatcherDistance only (km exactly 25))
and (hasDistance only (km exactly 70)) and (hasEVSEProfile only ACLevel2)
and (hasFeePerkWh only (cents exactly 55))

ChargingStationdistrictZ EquivalentTo: ChargingStation and
(hasDCCompatiblePlug only CHAdeMoPlug) and (hasChargingRatePerHour only
(km exactly 200)) and (hasAvailablePower only (kWh exactly 100)) and
(hasDispatcherDistance only (km exactly 45)) and (hasDistance only (km
exactly 5)) and (hasEVSEProfile only DCLevel1) and (hasFeePerkWh only
(cents exactly 60))

ChargingStation SubClassOf: (hasChargingRatePerHour some) and
(hasAvailablePower some) and (hasDispatcherDistance some) and
(hasFeePerkWh some) and (hasEVSEProfile some)
```

Figure 4.8: Semantic annotations of friends' services

The request cannot be satisfied directly. So taxi fleet agents autonomously cooperate in order to share useful information about electricity suppliers among their friends and suggest the services they offer.

ChargingStationdistrictZNew service, summarized in Figure 4.9, is selected by a friend and tagged (*i.e.*, suggested). Notice that service description has a dispatching distance higher than the desired maximum: non-exact

match is tackled by means of non-standard reasoning services for semantic matchmaking recalled in Section 2.2.3. Furthermore, the experience factor *Suggest* $S(D)$ described in Section 4.1 is exploited to refine social relationships. Let us suppose that taxi become a frequent visitor of district Z and here it often needs charging. If $ChargingStationdistrictZNew$ service of device D_{new} is frequently tagged, $S(D_{new})$ value will increase during the social object's lifetime. If this score becomes higher than *TH-suggest* threshold, a new friendship relation is established, so that the taxi will be able to directly retrieve and use the services of that charging station.

```
ChargingStationdistrictZNew EquivalentTo: ChargingStation and
(hasDCCompatiblePlug only CCSCombo2Plug) and (hasChargingRatePerHour
only (km max 150)) and (hasAvailablePower only (kWh exactly 300)) and
(hasDispatcherDistance only (km exactly 40)) and (hasDistance only (km
exactly 6)) and (hasEVSEProfile only DCLevel1) and (hasFeePerkWh only
(cents exactly 45))
```

Figure 4.9: Semantic description of selected service

This is just a small example of the benefits of the proposed semantic-based framework, capable of autonomic and proactive configuration through interaction paradigms mutuated from social networks. Furthermore, semantic annotations in the case study have been simplified for the sake of understandability. In real smart grid scenarios, more complex requests and service descriptions are expected, and non-standard inference services will be even more useful for service ranking and composition to satisfy requests.

4.3 Experimental evaluation

Performance evaluation of the dynamic social relationship framework has been carried out testing social objects with 18 different network configurations, generated by combining the parameters reported in Table 4.2. A preliminary phase has been performed to empirically validate values (the same for all devices) of parameters used by the *experience* functions described in Section 4.1: *decay rate* $X = 2$, ensuring an adequate and not excessive decay for older elements of the time series; $w = 0.3$, to favor devices exposing tagged (*i.e.*, selected) services over objects only sending comments; *TH-suggest* = 0.5, *TH-retract* = -0.1 and *TH-satisfaction* = 0.2, aiming to remove friendships only if really needed (in volatile contexts devices should be able to exploit availability opportunistically) and to suggest new relationships appropriately, while limiting the network overhead. *Maximum hop*

Table 4.2: Network configuration parameters

Parameter	Value	Description
Generation algorithm	RND	*Randomly* generated nodes with mean number of friends set to 1. Loosely connected network with frequent isolated objects.
	BA	Nodes generated exploiting the preferential attachment rule of the *Barabási-Albert* model [15]. Each node has 2 friends on average. Hierarchical topology similar to a sensor network with several edge nodes and one or more sinks.
	DM	Nodes generated using the *Dorogovtsev-Mendes* algorithm [36]. Strongly connected network with nodes having 4 friends on average.
Nodes	10	Small-size network
	100	Medium-size network
	1000	Large-size network
Request rate	20	20% of nodes identifies a new event and posts a message in each period. Network with *few* device requests.
	70	70% of nodes identifies a new event and posts a message in each period. Network with *frequent* device requests.

count of cooperative service discovery process –detailed in Section 4.1– has been set to 2 for all devices.

For each configuration, tests have been repeated 20 times (corresponding to 20 random initial topologies) monitoring the network evolution during 10 periods of 60 seconds. Experiments have been conducted exploiting three sets of 10, 100 and 1000 device descriptions, respectively. Every device exposes 5 services, each associated to an OWL annotation generated as the logical conjunction of 10 OWL classes (also including negated classes) randomly selected from a total of 168 classes in the reference ontology. After each period, the following output values have been computed to identify specific behaviors characterizing the social network and in particular to evaluate the capability to adapt social relationships dynamically according to context changes: (i) number of friends per node; (ii) like value received by a post; (iii) number of forwarded messages (*i.e.*, how many devices are contacted to satisfy a request); (iv) number of activated services per request; (v) size of payload data exchanged on the network to satisfy a request (including uncovered part of requests forwarded to friends and comments retrieved from the wall of contacted objects). Obtained results have been averaged in order to filter out the bias deriving from conditions of single runs. Devices have been initialized at the beginning of each test whereas requests have been generated at random instants, with uniform probability distribution within the simulation time.

Small-size networks. Figure 4.10 shows test results for scenarios with 10 nodes. For loosely connected networks (RND), like values are very low (below 0.35) in the first period, as reported in Figure 4.10(b), due to the few friendship relationships (Figure 4.10(a)). For each request, a device starts a local covering process with few available services, so it is only partially

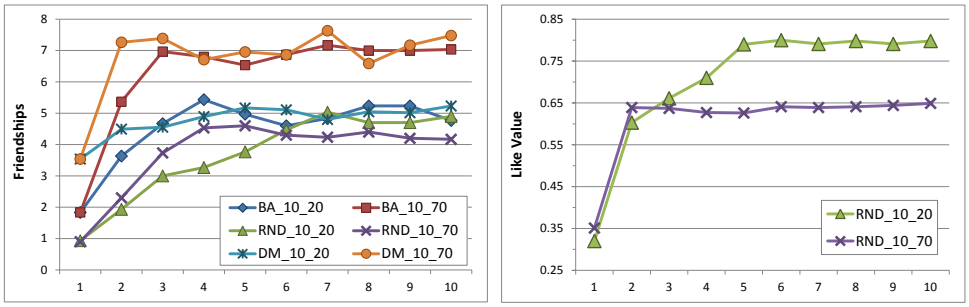
satisfied through a minimal set of useful functionalities (Figure 4.10(d)) and a large amount of data (corresponding to the uncovered part of the request) is forwarded to friends (Figure 4.10(f)). Thanks to the *Satisfaction* and *Suggestion* indexes described in Section 4.1, *unsatisfied* devices (*e.g.*, isolated nodes) promptly look for useful friendships. In the second period, it is possible to notice both a considerable increase of the like value and a reduction of the data exchanged on the network. Having more friends, each device is able to locally satisfy a request in a better way and to forward a smaller uncovered part. A similar trend can be also identified for BA and DM network configurations; in these cases, however, the network starts with a high degree of connectivity in the first period generating higher like values –as shown in Figure 4.10(c)– which increase more slowly w.r.t. the previous scenario. Nevertheless, semantic-based experience factors allow satisfactory like values, reducing at the same time the network traffic. In fact, introducing more useful friendships and removing useless ones allows social things to select the most suitable services at each hop of the cooperative covering process, converging faster to the final result. Moreover, the request rate mainly affects the behavior of social networks with few connections: device relationships are modified more frequently, leading to lower like values and less efficient cooperative covering. Few messages are forwarded to friends, as reported in Figure 4.10(e), due to the more variable social network configuration. On the contrary, social devices are able to establish stable relationships after about 4 periods for all other configurations.

Medium-size networks. Also in scenarios with 100 nodes (results detailed in Figure 4.11) loosely connected networks take quickly advantage of dynamic relationship adaptivity, increasing average like values (Figure 4.11(b)) and reducing transmitted data (Figure 4.11(f)), but also BA and DM scenarios improve their performance over the periods. The presence of many devices facilitates the discovery of additional useful friends, as reported in Figure 4.11(a). As a consequence, higher like scores have been obtained: all network parameters tend to stable values in about three (friendships and exchanged data) to five (activated services and forwarded messages) periods.

Large-size networks. Finally, results for scenarios with 1000 nodes are reported in Figure 4.12. Large networks follow similar trends highlighted for small and medium configurations. Main differences compared to medium scenarios concern: (i) fewer friends per device (Figure 4.12(a)), as friendships are removed more frequently in large networks, so the number of friends grows more slowly; (ii) fewer activated services per request (Figure 4.12(d)), as when nodes reach the stable configuration, the covering process is able to select few services which largely satisfy a request, rather than many services

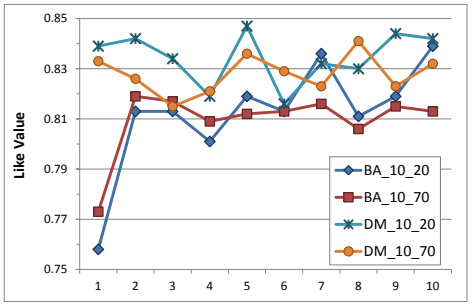
each covering a small part.

Thanks to these optimized relationships, devices save storage space (activated functionalities are tagged and saved on the device wall) and reduce the network overhead required to contact and activate remote services. This is a clear benefit of the approach, envisioning good scalability: even on large networks, nodes establish a balanced number of high-quality friends and share their resources without overloading the network.

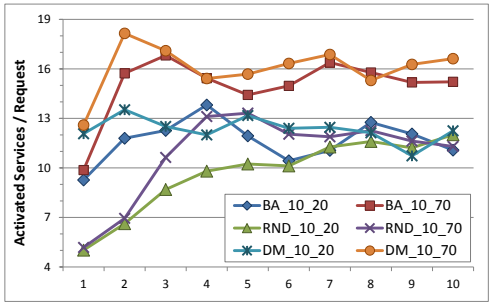


(a) Friendships

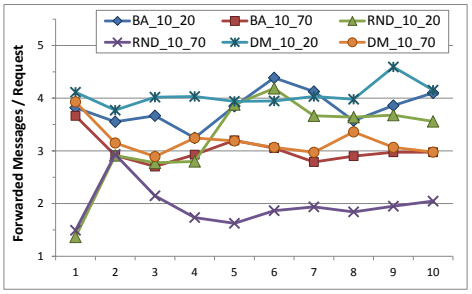
(b) Like value (RND)



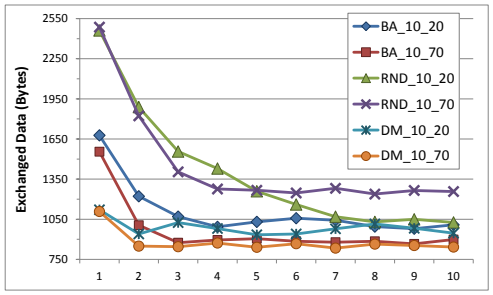
(c) Like value (BA, DM)



(d) Activated services

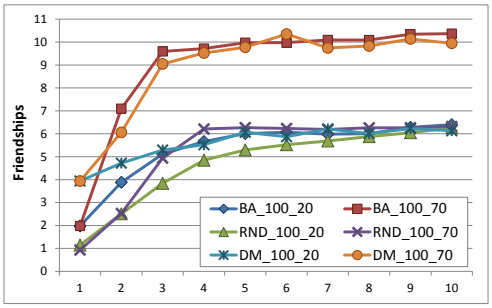


(e) Forwarded messages

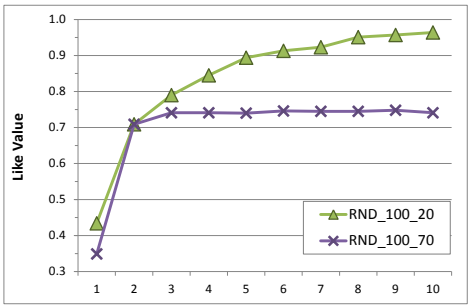


(f) Exchanged data

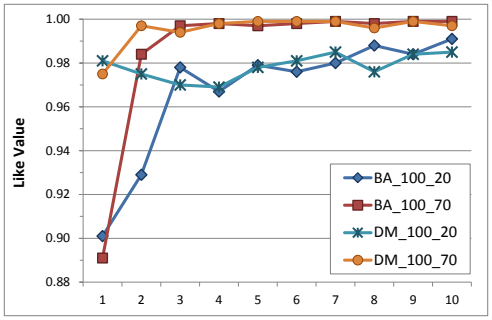
Figure 4.10: Test results for small-size networks. Legend denotes values of parameters for each configuration.



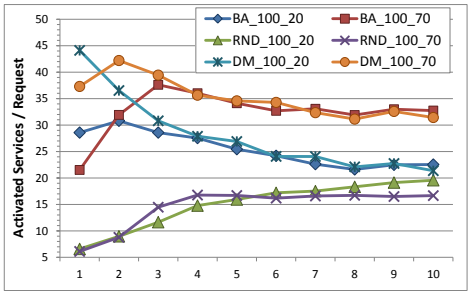
(a) Friendships



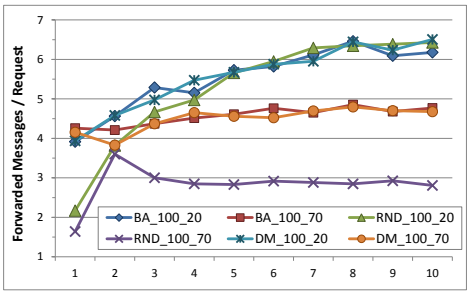
(b) Like value (RND)



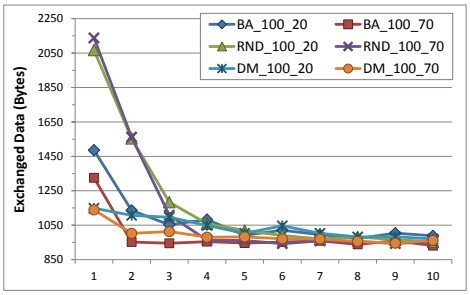
(c) Like value (BA, DM)



(d) Activated services

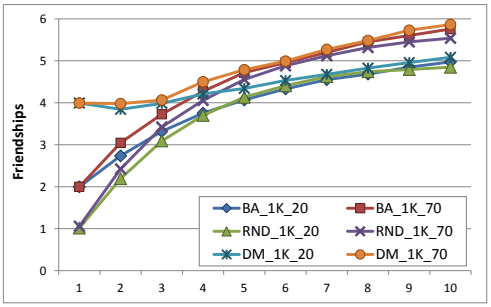


(e) Forwarded messages

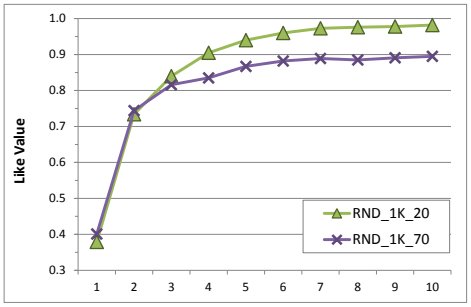


(f) Exchanged data

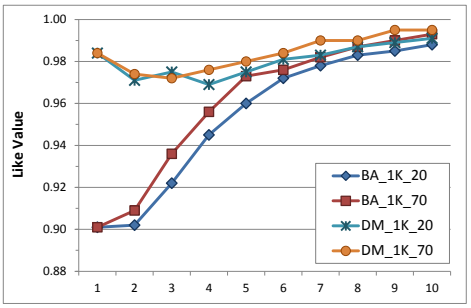
Figure 4.11: Test results for medium-size networks. Legend denotes values of parameters for each configuration.



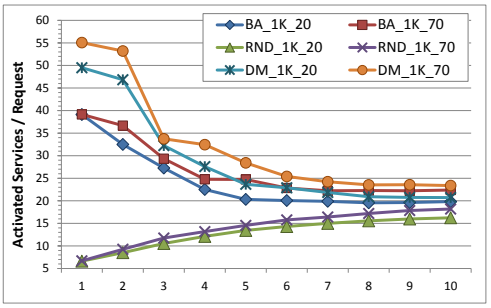
(a) Friendships



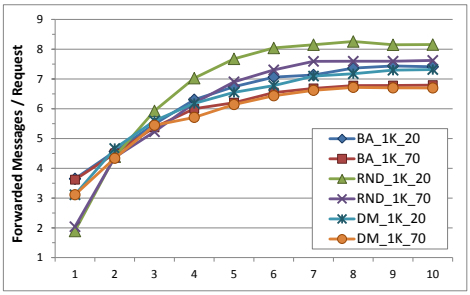
(b) Like value (RND)



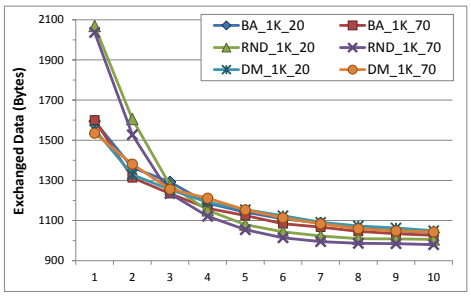
(c) Like value (BA, DM)



(d) Activated services



(e) Forwarded messages



(f) Exchanged data

Figure 4.12: Test results for large-size networks. Legend denotes values of parameters for each configuration.

Chapter 5

Decisional capabilities for smart objects

This chapter provides a detailed description of the proposed semantic-enabled blockchain framework for SWoT contexts. The *Hyperledger Sawtooth*[1] framework [89] has been extended with a fully decentralized service-oriented architecture layer. Registration, discovery and selection of annotated services/resources are implemented as smart contracts executed by nodes, allowing distributed execution and trust. While semantic matchmaking enables relevant resource retrieval with logic-based ranking and explanation features, blockchain provides reliable transaction storage. Experimental tests on a cluster of virtual nodes provide early insight on effectiveness, performance and scalability.

5.1 Proposed framework

The Semantic Web of Things improves the Internet of Things power by increasing resource representation capabilities through knowledge management and reasoning technologies adapted from the Semantic Web. This promotes information interoperability and decision autonomy. Nevertheless, trust and reliability issues remain basically unsolved. Large-scale, decentralized and dynamic infrastructures suffer from unpredictable volatility of nodes, which compromises resource availability. Trust and coordination are still difficult. Blockchain is increasingly used as a transactional data storage solution for distributed ledgers. It enables trustless collaboration by enforcing smart

[1]Hyperledger Sawtooth: `https://www.hyperledger.org/projects/sawtooth`

contracts and prevents data tampering by validating transactions through consensus protocols.

This work introduces *SeeSaw* (SEmantic-Enhanced SAWtooth) semantics-enabled SOA for trustless collaboration in pervasive computing, particularly aimed at advanced Industrial IoT (IIoT) scenarios. It is based on the *Sawtooth* project [89] developed within the *Hyperledger* initiative by the Linux Foundation and players in the information technology industry. The proposal aims to overcome the limitations of the previous approach in [113] by adopting a more IoT-oriented blockchain substratum articulated in several independent components with well-defined responsibilities. Enhanced requests flexibility and improvement in scalability and computational resource usage are also desired.

Figure 5.1 shows the overall architecture. Key elements are as follows:

- *Producer (P)*: exposes resources by registering them on the blockchain as assets.
- *Consumer (C)*: requests resources through a semantic-based process comprising Discovery, Explanation (optionally) and Selection transactions, through *smart contracts* detailed as in what follows.
- *Web Interface (WI)*: acts as gateway to the blockchain, collecting requests from Producers and Consumers and forwarding them to a Validator. A WI can communicate with multiple C and P instances; in the current implementation the *WebSocket* protocol [40] is adopted, which is suitable for resource-constrained IoT contexts for its bandwidth and computational efficiency [84]. Further point-to-point or mesh protocols for IoT could be implemented. On the other hand, a WI communicates with exactly one Validator by means of *ZeroMQ*[2] distributed messaging protocol.
- *Transaction Processor (TP)*: executes transactions at the edge of the network by implementing smart contracts. Sawtooth supports Transaction Processors written in Python, C++, Go, Java, JavaScript or Rust; C++ has been chosen in the SeeSaw implementation for efficiency reasons. Each TP communicates with one Validator through ZeroMQ.
- *Validator*: has access to the *radix Merkle tree* data structure of the blockchain. It receives transaction requests from a WI and dispatches them, balancing the load among all its connected TPs, according to

[2]ZeroMQ: `http://zeromq.org/`

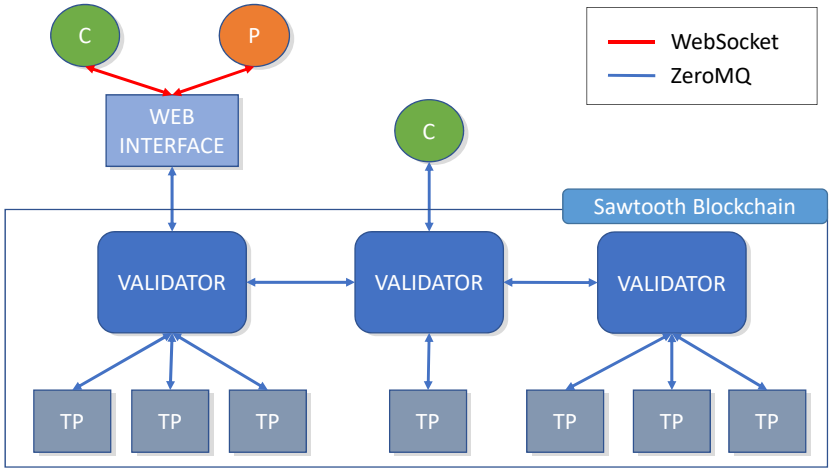


Figure 5.1: Framework architecture

the master/workers model. If TP load is full, a transaction is returned as invalid: WI typically implements a backoff mechanism for waiting before resubmission. Validators form a peer-to-peer network, communicating through ZeroMQ for the replication of smart contract execution and the consensus protocol. Sawtooth uses PoET as consensus protocol: a Validator is elected as leader through a lottery-like mechanism and is allowed to add a new block of transactions to the chain. Sawtooth Validators are implemented in Python and use a gossip protocol to allow new peers to join dynamically an existing network.

All exchanged messages are serialized in the Protocol Buffers format. Transactions can be processed and validated individually or in *batches*, depending on request parameters. A batch establishes a sequential dependency relationship among transactions: if one fails, the subsequent ones are not processed and the whole batch is invalidated.

Depending on target scenarios, several configurations are possible, based on the following considerations.

- Producers and Consumers have minimal computational requirements, being mainly aimed at nodes deployed in the field (*e.g.*, mobile devices or embedded in cyber-physical system). A device will run both components if it needs to provide as well as request services.
- WI nodes can be either dedicated devices at the edge of the network or integrated into Validators, as shown in Figure 5.1.

- Due to blockchain storage and consensus, Validators require the largest amounts of mass memory and bandwidth. For this reason, they are the most suitable to be hosted on premises at the core of the organization's network.

- Conversely, TPs do not need large storage or bandwidth, as they process one transaction at a time. They need relatively fast processing capabilities, but with proper smart contract optimization even low-cost single-board computers like *Raspberry Pi* may be a suitable platform. *Fog computing* devices at the edge of the network are good candidates for the TP role, or even mobile devices with moderate computational and energy resources. For increasing transaction throughput, in fact, scaling out (*i.e.*, increasing the number of TPs) may be more beneficial than scaling up (*i.e.*, using more powerful devices), as studies suggest [55].

Knowledge representation in smart contracts for complex service-oriented architectures

The proposed approach defines a semantic resource/service discovery layer. The IoT-oriented blockchain thus becomes a SOA, supporting basic tasks of resource registration, discovery, and selection implemented as SCs, with distributed execution and consensus-based validation. Discovery is based on semantic matchmaking of descriptions of a request and a set of resources, annotated as DL concept expressions in OWL 2 w.r.t. a shared ontology. For each request-resource annotation pair, a $0 - 100$ *semantic relevance score* is computed from a combination of penalties induced by Concept Contraction and Concept Abduction, as recalled in Section 2.2.3. This enables a formally founded relevance ranking of all available resources on the blockchain that are described w.r.t. the same ontology as the request. The adopted inference services also return a logical *explanation* of discovery outcomes. Transactions are recorded on the blockchain for robustness, traceability and accountability purposes. SOA primitives and corresponding SCs are reported as in what follows.

Registration. Several resource domains can co-exist in the same blockchain. Each domain is associated to a different ontology, which provides the reference conceptual vocabulary to annotate resources. Every ontology is identified by a unique Uniform Resource Identifier (URI), as per OWL specifications. Each node can own resource instances, characterized by:

- a URI identifying the resource unambiguously;

- a semantic annotation in OWL language, modeling high-level descriptive information of resource features;
- the URI of the reference ontology;
- a set of data-oriented attributes stored as a key-value pair, allowing to integrate and extend logic-based inferences with application-specific and context-aware information processing.

In order to make a resource available for discovery and usage, the owner registers it as an asset on the blockchain storage. Ontologies are registered in the same way. Through this SC a blockchain-backed u-KB is thus obtained.

Resource discovery. IoT applications vary widely in functional and service level agreement (SLA) requirements. Some use cases need quick-response resource discovery and best-effort recall is tolerated; this is typical of pervasive computing contexts. Other applications require an exhaustive search space exploration to guarantee that the best possible resources are found. In order to cope with the widest range of scenarios, SeeSaw includes two discovery modes, named *fast* and *full*. Supposing n annotated resources are associated to an ontology, the whole set (*i.e.*, the ABox of the u-KB) is divided in *pieces* of size p: each of the first $k = \lfloor n/p \rfloor$ pieces will contain exactly p resources and the last piece the remaining $n - kp$ ones. Any discovery request will generate up to $k + 1$ SC transactions, one for each piece. A transaction yields a *hit* if at least one of the resources in the piece has a semantic relevance score higher than a given threshold, a *miss* otherwise. Hit resources are returned to the requester. In full discovery, all $k + 1$ transactions are submitted simultaneously and the receiving Validator will take care of load balancing among Transaction Processors; the requester will receive all resources above semantic relevance threshold. Furthermore, both hit and miss transactions are committed to the blockchain for traceability purposes. Conversely, fast discovery submits *clusters* of at most $c \leq k$ transactions at a time; cluster size is a system configuration parameter. Furthermore, fast discovery is limited by an overall *timeout*[3]. As soon as a cluster returns a hit or when the timeout expires, remaining clusters are not submitted. Moreover, in fast discovery miss transactions are invalidated and not committed to the chain, in order to reduce consensus stress of Validators. The adoption of clusters aims at a trade-off between a completely serial and parallel piece processing: the former minimizes blockchain load but may increase length

[3]In the current implementation, cluster size and timeout length are static system-wide parameters. Studying dynamic adaptation strategies w.r.t. past performance and current network status is an aspect of interest, but it is left for future work.

and variability of hit latency, potentially incurring in more frequent timeouts; the latter ensures that a hit is found if it exists in the chain, but places a heavier computational burden and incurs in higher turnaround time.

Parameters of the discovery SC are as follows:

- mode: `fast` or `full`;
- URI of the reference ontology: this determines the resource domain as well as the vocabulary used to express both the request and the resources to be retrieved;
- semantic annotation of the request in OWL language, specifying desired resource features and constraints;
- maximum acceptable value for the i^{th} data-oriented attribute $a_{i_{max}}$ (*e.g.*, the maximum price the requester is willing to pay). Resources with at least one value higher than this threshold will be skipped from matchmaking (thus reducing computational overhead);
- minimum semantic relevance threshold s_{min}, as a floating-point number in the $[0, 1]$ interval, with a value of 1 corresponding to a full match and 0 to a complete mismatch (both rare situations in realistic scenarios); after matchmaking, resources with a relevance score below this threshold will not be returned, as deemed irrelevant to the requester.

Explanation. This SC is used to request a justification of the matchmaking outcome for a specific resource among received results. This may be useful for request revision and refinement [106] as well as *post-hoc* audit of the discovery process. Explanation reinforces the overall trust in the blockchain framework not only at data management level, but also at application level. Parameters of the SC are (i) the semantic annotation of the request and (ii) the URI of the discovered resource. SC result consists in the semantic affinity score $0 \leq s_i \leq 1$ and concept expressions of G and K from Concept Contraction and of H from Concept Abduction.

Resource selection. After receiving all results exploiting full discovery, or a subset with fast discovery, the requester can select the best discovered resource with this SC. The complete registered resource representation is retrieved from the blockchain and returned. The proposal does not constrain resource fruition in any way, leaving application-specific details –such as interface endpoint or payment method– to resource annotations themselves.

5.2 Experiments

In order to obtain a quantitative performance analysis, small, medium and large scale chain scenarios have been considered, respectively with 100, 200 and 400 nodes. Nodes are split in three sets: *Clients (20%)*, acting as workload generators, *e.g.*, Providers of annotated resources registered in the blockchain and Consumers requesting resources; *Validators (20%)*; *Transaction Processors (60%)*. The following experimental parameters have been set: (i) duration of 300 s, after a preliminary resource registration phase (excluded from the analysis as it is very lightweight); (ii) 6 randomly-generated annotations per Producer; (iii) each Consumer sends a new randomly-generated request every 8 s; (iv) the minimum threshold of semantic affinity is 0.85. Each scenario has been executed in all the combinations of the following parameters: (i) discovery timeout (T) set to 2, 6 or 10 s; (ii) piece size (P), described in Section 5.1, set to 10, 30 or 50 resources; (iii) either *fast* or *full* discovery. For each combination, the average of two runs has been considered.

The experimental campaign has leveraged the *Docker* platform to deploy the testbed, by performing the following steps: (i) each prototype component has been compiled as a Docker image; (ii) each node has been executed as a container instance of the corresponding compiled image; (iii) a Docker *Swarm mode* cluster has been deployed on 10 *VMware vSphere* virtual machines running on two server blades (Intel Xeon E5-2650 v3 CPU –8 cores/16 threads at 2.30 GHz– and 96 GB of RAM per blade, Ubuntu 16.04 64bit operating system), with an overlay network configured to allow communications among nodes; (iv) the execution of experiments has been managed via the Docker API SDK[4]. The following performance metrics have been measured: (i) mean turnaround time and standard deviation for fulfilling requests, also split into individual SCs of Discovery, Explanation and Selection; (ii) mean hit ratio, *i.e.*, the percentage of requests which retrieve at least one resource satisfying semantic relevance constraint within the given timeout; (iii) mean and standard deviation of memory and CPU load for Transaction Processor and Validator nodes. No RAM and CPU constraints have been imposed in the configuration of Docker to avoid affecting other performance metrics.

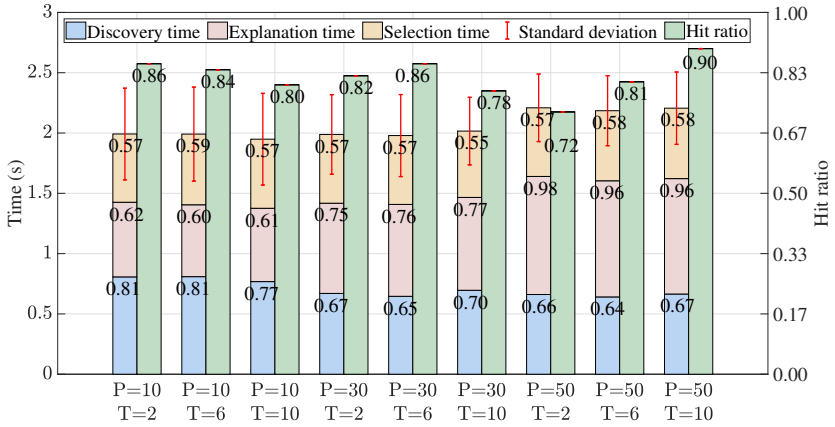
Fast discovery. Mean turnaround time can be deemed as very low in all scenarios, as depicted in Figure 5.2. For $T = 2s$, the mean discovery time of requests which retrieved results (hits) was low, while the others reached timeout. Therefore the overall hit ratio and the standard deviation of turnaround were lower. For higher timeout values, the hit ratio increased but turnaround time had higher variability, possibly due to load fluctuations. As expected,

[4]A simple docker client for the JVM: `https://github.com/spotify/docker-client`

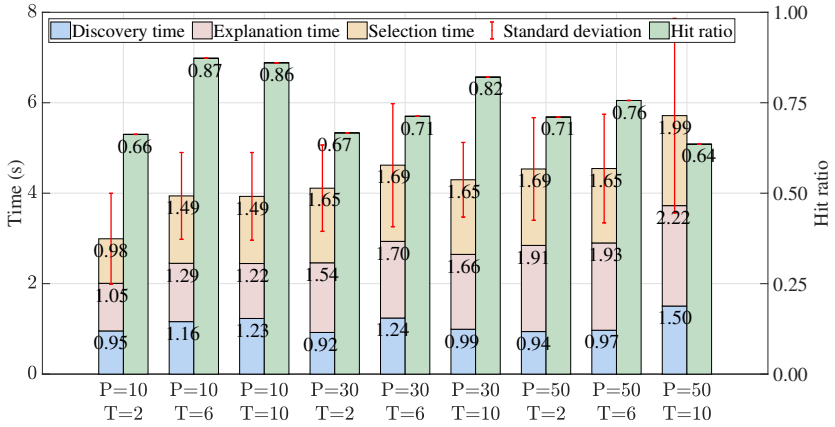
hit ratio tends to decrease for higher number of nodes, but the trend appears as regular and predictable. Significantly, in all the experiments the time of discovery process was similar to the ones of explanation and selection phases. This suggests that consensus is the longest task, while transaction processing time is relatively short: this implies that the performance impact of the proposed semantic SOA layer is low. As shown in Figure 5.3, the average CPU and RAM usage per Validator increased for larger scenarios. Moreover, the value of P slightly influenced the overall load. CPU and memory usage for TPs are reported in Figure 5.4: the CPU load is higher for larger scenarios, but on average it is below 2%; memory consumption is also very low.

Full discovery. Figure 5.5(a) shows average turnaround time is closely related to the number of nodes and registered resources. The best results were obtained in the 100 nodes scenario. Turnaround time decreases when P is larger, because of the lower number of committed transactions. Moreover, the standard deviation is higher when P is small, because discovery requests generate more blocks which need to be committed by Validators, stressing the consensus protocol. Figure 5.5(b) reports the average CPU and RAM usage per Validator: also in this case higher values and higher variance are found for larger scenarios, with practically no influence of Piece size. Figure 5.5(c) shows average values for Transaction Processors, denoting similar trends.

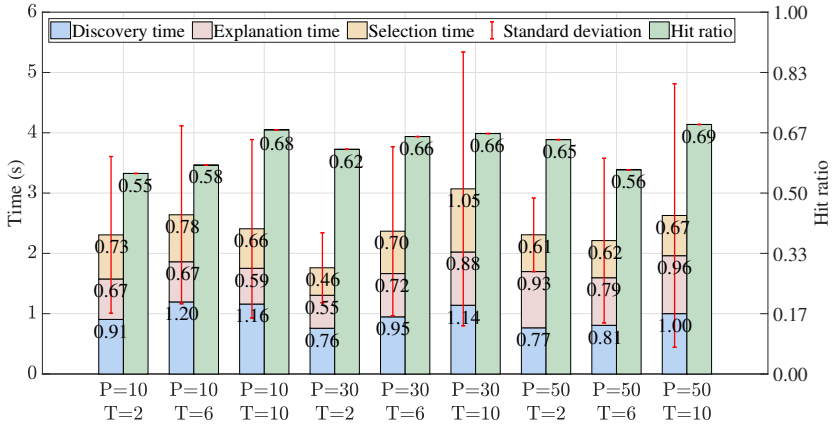
The experimental results support the feasibility of the proposed approach. Semantic matchmaking does not introduce significant overhead. Fast discovery has stable times, best-effort hit ratio and graceful degradation at larger scales, while full discovery guarantees the best semantic matchmaking results with a linear turnaround time increase w.r.t. network scale. Overall, the See-Saw architecture appears as amenable to MEC, where semantic blockchain load can be sustained by Validators associated with relatively large numbers of lightweight mobile and embedded devices, acting as semantic Transaction Processors at the network edge and in the field.



(a) 100 nodes

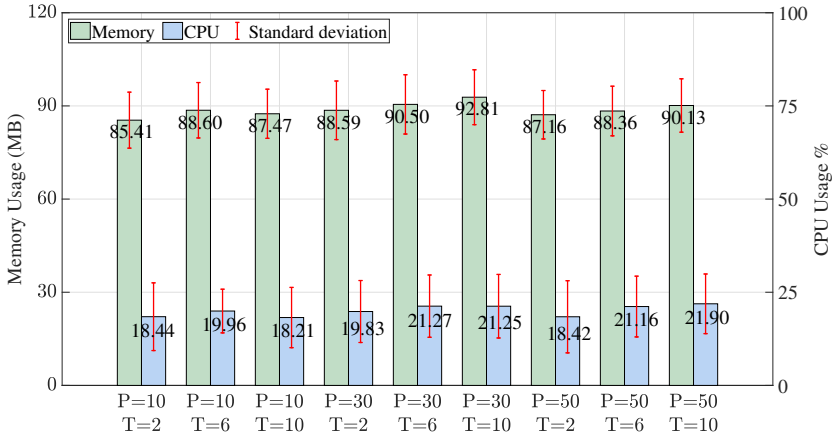


(b) 200 nodes

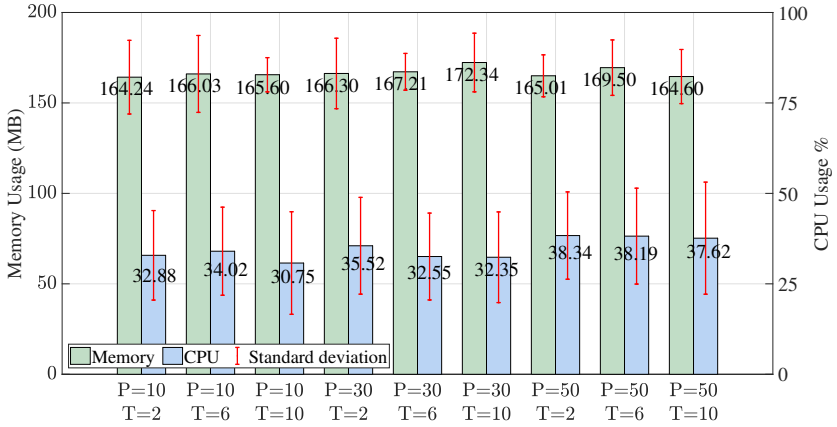


(c) 400 nodes

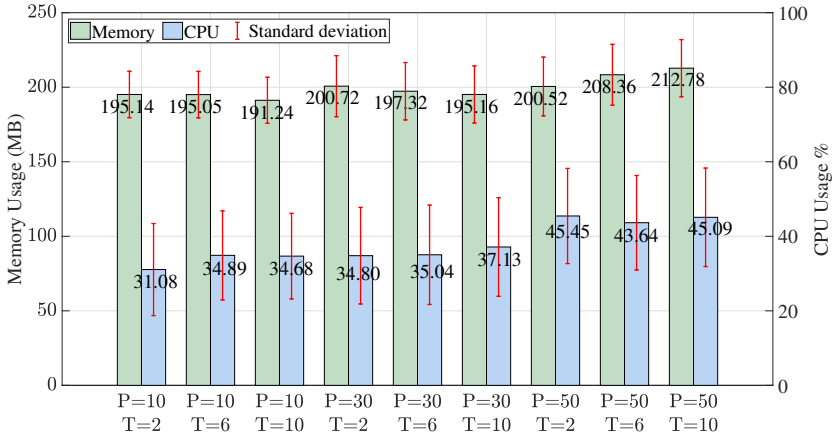
Figure 5.2: Fast discovery turnaround time and hit ratio; P: piece size; T: timeout (s)



(a) 100 nodes

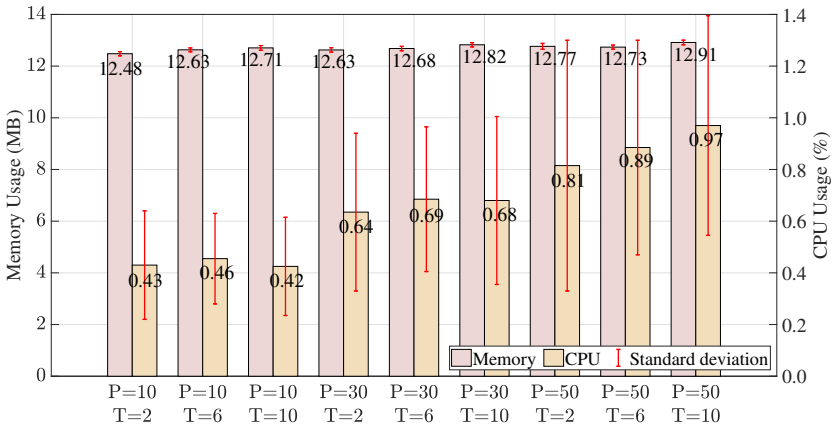


(b) 200 nodes

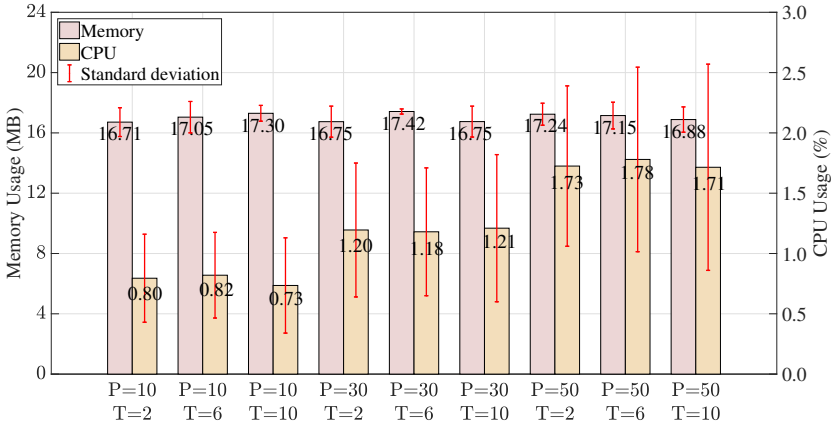


(c) 400 nodes

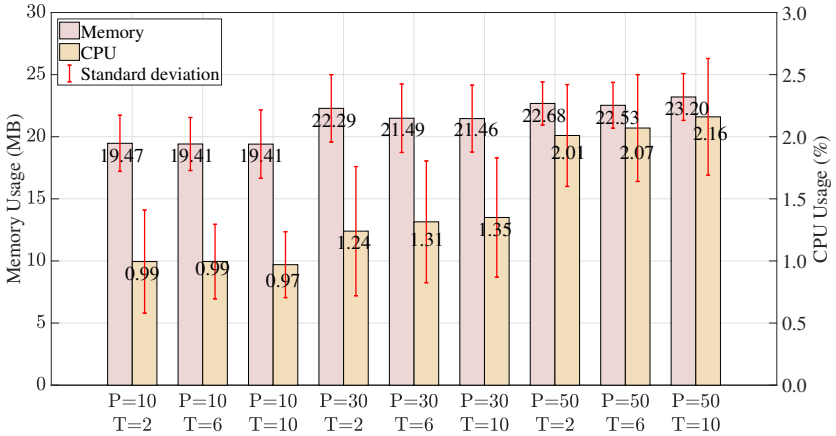
Figure 5.3: Fast discovery Validator memory and CPU usage; P: piece size; T: timeout (s)



(a) 100 nodes

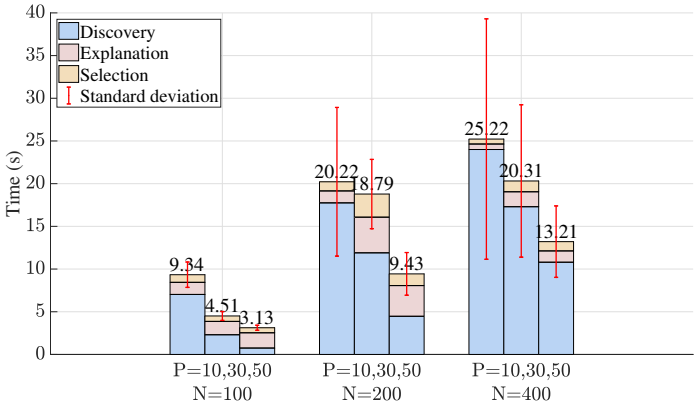


(b) 200 nodes

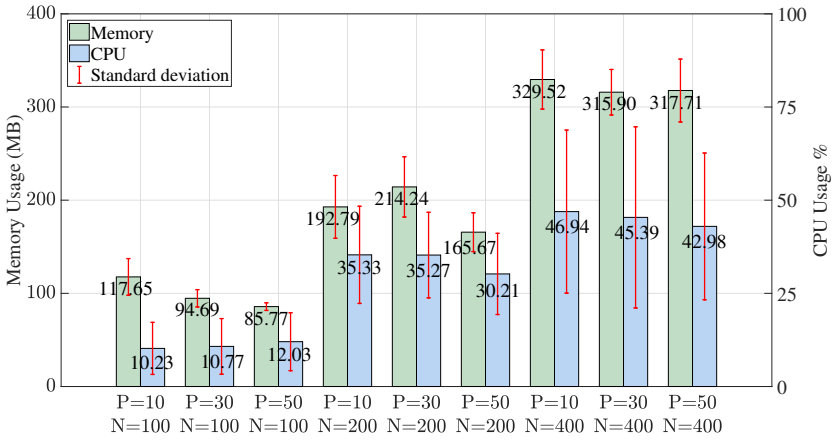


(c) 400 nodes

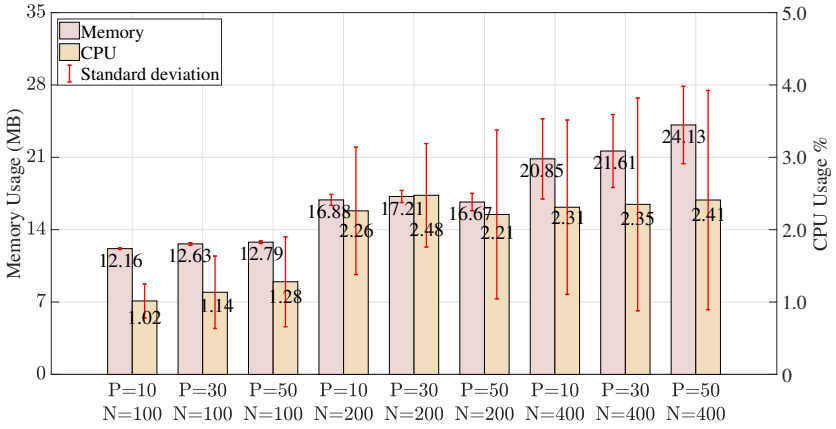
Figure 5.4: Fast discovery Transaction Processor memory and CPU usage; P: piece size; T: timeout (s)



(a) Turnaround time



(b) Validator memory and CPU usage



(c) Transaction Processor memory and CPU usage

Figure 5.5: Full discovery performance; P: piece size; N: nodes

Chapter 6

Conclusions and perspectives

This thesis has introduced the *object (b)logging* paradigm, defining a novel semantics-based framework for pervasive Cyber-Physical Systems. The main goal is to enable smart objects to drive complex behaviors through swarm intelligence, dealing with the intrinsically unpredictable nature of mobile and pervasive scenarios. Devices act as social agents, capable of configuration, coordination, and orchestration starting from the inferred high-level context description. Interaction patterns inspired to Social Networking Services (SNSs) enable objects to establish cooperation relationships, share information, issue requests, and update their status and settings, in a fully dynamic and decentralized fashion. Each object is equipped with an embedded reasoning micro-engine through which it is able to perform automated inference in order to derive previously implicit knowledge out of information gathered from the environment and/or other smart entities. Agents must integrate detected and received information in a coherent view to recognize and annotate in a *micro-log* the context they are in and their status.

A novel Concept Fusion inference service for ubiquitous MAS has been developed in order to merge heterogeneous information sources, suitable for robust distributed context monitoring even in the presence of incomplete and/or inaccurate information. Early experiments devoted to assessing efficiency and scalability on a realistic resource-constrained computing platform have been carried out on a Raspberry Pi Model B. A novel general-purpose variant of Concept Integration [111] algorithm has been also introduced. It applies Concept Fusion only to observation shared by remote agents, and finally merges this aggregated knowledge with the endogenous one, preserving the different viewpoints. In this way smart objects continuously enrich their basic descriptive core according to detected events and phenomena, and expose advanced and semantically unambiguous descriptions of their context

and capabilities toward the rest of the world in a *micro-blog*. Via blog entries ("posts"), objects can interact at the application layer with humans and other entities in order to create an efficient service-oriented social network. where all nodes are able to cooperate toward a common goal.

The object (b)logging vision has been implemented and tested via simulations built upon the INETMANET framework, embedding the Mini-ME semantic matchmaker. Experimental evaluation has been carried out via Omnet++ simulation in order to assess effectiveness, correctness and feasibility of the proposal with reference to a possible exploitation on real large-scale resource-constrained scenarios.

An extension of the framework described in [115] has been proposed to endow social objects with proactive adaptivity to environmental modifications in order to improve self-organization through social relationships refinement. Based on past experience (*i.e.*, previous interactions over the social network), each object is able to adapt its relationships dynamically according to context changes, in order to improve not only the processing on the single node but also the overall network performance. Removing unnecessary relationships with devices often providing conflicting services and/or not useful in satisfying requests aims to reduce network traffic, *i.e.*, less messages will be forwarded during a service discovery process. Conversely, the suggestion of new meaningful interactions with objects providing services frequently or in relationship with many useful devices can increase the network effectiveness and reduce the relative computational load per node: a request could be satisfied with lower processing time and activating the minimum set of services. Benefits of the proposal have been demonstrated in a case study on the power management of electric vehicles in a Smart Grid and experimental evaluations have been carried out.

Finally, in order to guarantee trust and transaction traceability in IoT contexts, a semantic enhancement layer has been added on top of the Hyperledger Sawtooth blockchain framework. Sawtooth design is particularly suitable for IoT scenarios, due to the high decoupling between components, which allows to distribute request management, transaction processing and validation on different kinds of devices. Results of early performance evaluations have been provided, particularly targeted toward scenarios including large numbers of resource-constrained nodes, and support the feasibility and sustainability of the approach.

The overall proposed solution results as a general-purpose, cross-domain semantic-based framework for context detection, knowledge discovery and sharing among pervasive smart devices. The approach provides the means to harness the flow of semantically annotated detected and received updates, enabling context-aware adaptive behaviors in several application areas, which

include urban search and rescue, power management in smart grid, personal assistance, home automation, smart agriculture and many more.

Future work directions concern further performance optimization and thorough comparison with state-of-the-art approaches. Several perspectives are open for more advanced (b)logging work-flow, where an agent shares in its blog also the uncovered part of the requested action as Constraint. When nearby entities receive this packet they may try their best in order to satisfy the request, discovering suitable services among their own actuation capabilities and their peers'. Furthermore the expansion of the set of semantic Smart Contracts with advanced SOA facilities like service composition, substitution and negotiation, as well as with the support for ontology partitioning and on-the-fly reconstruction to achieve a full blockchain-backed u-KB realization is foreseeable. Finally, a wider experimental campaign involving implementation on real devices in order to perform further effectiveness evaluation should be carried out.

Bibliography

[1] Jose Aguilar, Marxjhony Jerez, and Taniana Rodríguez. CAMeOnto: Context awareness meta ontology modeling. *Applied Computing and Informatics*, 14(2):202–213, 2018.

[2] Ejaz Ahmed, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Imran Khan, Abdelmuttlib Ibrahim Abdalla Ahmed, Muhammad Imran, and Athanasios V Vasilakos. The role of big data analytics in Internet of Things. *Computer Networks*, 129:459–471, 2017.

[3] Kazi Masudul Alam, Mukesh Saini, and Abdulmotaleb El Saddik. Toward social internet of vehicles: Concept, architecture, and applications. *IEEE access*, 3:343–357, 2015.

[4] Safdar Ali and Stephan Kiefer. μOR–A micro OWL DL reasoner for ambient intelligent devices. In *4th International Conference on Advances in Grid and Pervasive Computing*, pages 305–316, Berlin, Germany, 2009. Springer.

[5] L. Andrade, M. Serrano, and C. Prazeres. The Data Interplay for the Fog of Things: A Transition to Edge Computing with IoT. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7, May 2018.

[6] Shaikh ZA Aqeel-ur Rehman and Zubair Ahmed Shaikh. Ontagri: scalable service oriented agriculture ontology for precision farming. In *2011 international conference on agricultural and biosystems engineering vols*, pages 1–2, 2011.

[7] Alfonso Ariza and Vincenzo Inzillo. INETMANET framework. In *Recent Advances in Network Simulation*, pages 107–138. Springer, 2019.

[8] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A Survey. *Computer networks*, 54(15):2787–2805, 2010.

[9] Luigi Atzori, Antonio Iera, and Giacomo Morabito. From "smart objects" to "social objects": The next evolutionary step of the internet of things. *Communications Magazine, IEEE*, 52(1):97–105, 2014.

[10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm. *Ad Hoc Networks*, 56:122–140, 2017.

[11] Luigi Atzori, Antonio Iera, Giacomo Morabito, and Michele Nitti. The Social Internet of Things (SIoT) – When Social Networks Meet the Internet of Things: Concept, Architecture and Network Characterization. *Computer networks*, 56(16):3594–3608, 2012.

[12] F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In *Int. Joint Conf. on Artificial Intelligence*, volume 19, page 364. Lawrence Erlbaum Associates LTD, 2005.

[13] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL – a polynomial-time reasoner for life science ontologies. *Automated Reasoning*, pages 287–291, 2006.

[14] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.

[15] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[16] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, 284(5):28–37, 2001.

[17] Nicola Bicocchi, Giacomo Cabri, Federica Mandreoli, and Massimo Mecella. Dealing with data and software interoperability issues in digital factories. In *Proceedings of the 25th ISPE Inc. International Conference on Transdisciplinary Engineering*, volume 7, page 13. IOS Press, 2018.

[18] Erik P Blasch, Dale A Lambert, Pierre Valin, Mieczyslaw M Kokar, James Llinas, Subrata Das, Chee Chong, and Elisa Shahbazian. High level information fusion (HLIF): Survey of models, issues, and grand challenges. *IEEE Aerospace and Electronic Systems Magazine*, 27(9):4–20, 2012.

[19] Alexander Borgida. Description logics in data management. *IEEE transactions on knowledge and data engineering*, 7(5):671–682, 1995.

[20] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Logical foundations of peer-to-peer data integration. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 241–251. ACM, 2004.

[21] Tomasa Calvo and Gleb Beliakov. Aggregation functions based on penalties. *Fuzzy sets and Systems*, 161(10):1420–1436, 2010.

[22] Sarven Capadisli, Amy Guy, Christoph Lange, Sören Auer, Andrei Sambra, and Tim Berners-Lee. Linked Data Notifications: a resource-centric communication protocol. In *European Semantic Web Conference*, pages 537–553. Springer, 2017.

[23] Federico Castanedo, Jesús García, Miguel A Patricio, and José M Molina. Data fusion to improve trajectory tracking in a Cooperative Surveillance Multi-Agent Architecture. *Information Fusion*, 11(3):243–255, 2010.

[24] Gustavo Cevolani. Truth approximation, belief merging, and peer disagreement. *Synthese*, 191(11):2383–2401, 2014.

[25] Ray Chen, Fenye Bao, and Jia Guo. Trust-based Service Management for Social Internet of Things Systems. *IEEE Transactions on Dependable and Secure Computing*, 13(6):684–696, 2016.

[26] Zhikui Chen, Ruochuan Ling, Chung-Ming Huang, and Xu Zhu. A scheme of access service recommendation for the Social Internet of Things. *International Journal of Communication Systems*, 29(4):694–706, 2016.

[27] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, 4:2292–2303, 2016.

[28] Simona Colucci, Tommaso Di Noia, Agnese Pinto, Azzurra Ragone, Michele Ruta, and Eufemia Tinelli. A Non-Monotonic Approach to Semantic Matchmaking and Request Refinement in E-Marketplaces. *Int. Jour. of Electronic Commerce*, 12(2):127–154, 2007.

[29] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

[30] Li Da Xu, Wu He, and Shancang Li. Internet of Things in industries: a survey. *Industrial Informatics, IEEE Transactions on*, 10(4):2233–2243, 2014.

[31] Orhan Dagdeviren, Ilker Korkmaz, Fatih Tekbacak, Kayhan Erciyes, et al. A survey of agent technologies for wireless sensor networks. *IETE Tech Rev*, 28:168–184, 2011.

[32] Elizabeth M Daly and Mads Haahr. Social network analysis for information flow in disconnected delay-tolerant MANETs. *IEEE Transactions on Mobile Computing*, 8(5):606–621, 2009.

[33] Amir Vahid Dastjerdi and Rajkumar Buyya. Fog Computing: Helping the Internet of Things Realize its Potential. *Computer*, 49(8):112–116, 2016.

[34] Tommaso Di Noia, Eugenio Di Sciascio, and Francesco M. Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Jour. of Artificial Intelligence Research (JAIR)*, 29:269–307, 2007.

[35] Francesco M Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. *Principles of Knowledge representation*, 1:191–236, 1996.

[36] Sergey N Dorogovtsev and Jose FF Mendes. Evolution of networks. *Advances in physics*, 51(4):1079–1187, 2002.

[37] Matthew English, Sören Auer, and John Domingue. Block Chain Technologies & The Semantic Web: A Framework for Symbiotic Development. In *Computer Science Conference for University of Bonn Students*, pages 47–61, 2016.

[38] Timofey Ermilov, Ali Khalili, and Sören Auer. Ubiquitous semantic applications: a systematic literature review. *International Journal on Semantic Web and Information Systems*, 10(1):66–99, 2014.

[39] Ivan Farris, Roberto Girau, Michele Nitti, Luigi Atzori, Roberto Bruschi, Antonio Iera, and Giacomo Morabito. Taking the SIoT down from the cloud: Integrating the Social Internet of Things in the INPUT architecture. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 35–39. IEEE, 2015.

[40] I. Fette and A. Melnikov. The WebSocket Protocol. RFC 6455, IETF, December 2011.

[41] Peter Gärdenfors. *Belief revision*, volume 29. Cambridge University Press, 2003.

[42] Roberto Girau, Salvatore Martis, and Luigi Atzori. Lysis: a platform for IoT distributed applications over socially connected objects. *IEEE Internet of Things Journal*, 4(1):40–51, 2017.

[43] Keyvan Golestan, Bahador Khaleghi, Fakhri Karray, and Mohamed S Kamel. Attention assist: A high-level information fusion framework for situation and threat assessment in vehicular ad hoc networks. *IEEE Transactions on Intelligent Transportation Systems*, 17(5):1271–1285, 2016.

[44] V. Haarslev and R. Müller. Racer system description. *Automated Reasoning*, pages 701–705, 2001.

[45] Klaus Hartke. Observing Resources in the Constrained Application Protocol (CoAP). RFC 7641, September 2015.

[46] Tom Heath and Christian Bizer. *Linked data: Evolving the Web into a Global Data Space*. Synthesis lectures on the semantic web: theory and technology. Morgan & Claypool Publishers, 2011.

[47] Ondrej Hlinka, Franz Hlawatsch, and Petar M Djuric. Distributed particle filtering in agent networks: A survey, classification, and comparison. *Signal Processing Magazine, IEEE*, 30(1):61–81, 2013.

[48] Matthew Horridge and Peter Patel-Schneider. OWL 2 Web Ontology Language Manchester Syntax (Second Edition). W3C note, W3C, December 2012. http://www.w3.org/TR/owl2-manchester-syntax/.

[49] I. Horrocks and PF Patel-Schneider. Optimizing description logic subsumption. *Jour. of Logic and Computation*, 9(3):267–293, 1999.

[50] Richard Hull, Vishal S Batra, Yi-Min Chen, Alin Deutsch, Fenno F Terry Heath III, and Victor Vianu. Towards a shared ledger business collaboration language based on data-aware processes. In *International Conference on Service-Oriented Computing*, pages 18–36. Springer, 2016.

[51] Dina Hussein, Son N Han, Gyu Myoung Lee, and Noel Crespi. Social cloud-based cognitive reasoning for task-oriented recommendation. *IEEE Cloud Computing*, 2(6):10–19, 2015.

[52] Dina Hussein, Soochang Park, Son N Han, and Noel Crespi. Dynamic social structure of things: a contextual approach in CPSS. *IEEE Internet Computing*, 19(3):12–20, 2015.

[53] Florian Idelberger, Guido Governatori, Régis Riveret, and Giovanni Sartor. Evaluation of logic-based smart contracts for blockchain systems. In *International Symposium on Rules and Rule Markup Languages for the Semantic Web*, pages 167–183. Springer, 2016.

[54] IEC PAS 63088. *Smart manufacturing – Reference architecture model industry 4.0 (RAMI4.0).* IEC – International Electrotechnical Commission, 2017.

[55] Fatemeh Jalali, Kerry Hinton, Robert Ayre, Tansu Alpcan, and Rodney S Tucker. Fog computing may help to save energy in cloud computing. *IEEE Journal on Selected Areas in Communications*, 34(5):1728–1739, 2016.

[56] Upul Jayasinghe, Hyun-Woo Lee, and Gyu Myoung Lee. A Computational Model to Evaluate Honesty in Social Internet of Things. In *Proceedings of the Symposium on Applied Computing*, pages 1830–1835. ACM, 2017.

[57] Panagiotis Kasnesis, Lazaros Toumanidis, Dimitris Kogias, Charalampos Z Patrikakis, and Iakovos S Venieris. ASSIST: An agent-based SIoT simulator. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, pages 353–358. IEEE, 2016.

[58] Yevgeny Kazakov. Consequence-driven reasoning for horn shiq ontologies. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[59] Yevgeny Kazakov and Pavel Klinov. Experimenting with elk reasoner on android. In *Proc. of 2nd International Workshop on OWL Reasoner Evaluation (ORE'13), Ulm (Germany)*, pages 68–74.

[60] Yevgeny Kazakov, Markus Krötzsch, and František Simančík. The incredible elk. *Journal of automated reasoning*, 53(1):1–61, 2014.

[61] Andrew Kennedy, Michele Southall, Gena Morgan, Ken Traub, et al. EPC Information Services (EPCIS) Specification. Technical report, GS1, May 2014. Available: `http://www.gs1.org/epcis/epcis/1-1`.

[62] Henry M Kim and Marek Laskowski. Toward an ontology-driven blockchain design for supply-chain provenance. *Intelligent Systems in Accounting, Finance and Management*, 25(1):18–27, 2018.

[63] T. Kim, I. Park, S.J. Hyun, and D. Lee. MiRE4OWL: Mobile Rule Engine for OWL. In *Computer Software and Applications Conf. Workshops (COMPSACW), 2010 IEEE 34th Annual*, pages 317–322. IEEE, 2010.

[64] Thomas Kleemann and Alex Sinner. User Profiles and Matchmaking on Mobile Phones. In Oscar Bartenstein, editor, *Proc. of 16th Int. Conf. on Applications of Declarative Programming and Knowledge Management INAP2005, Fukuoka*, 2005.

[65] F. Koch. 3APL-M platform for deliberative agents in mobile devices. In *Proc. of the fourth international joint conference on Autonomous agents and multiagent systems*, page 154. ACM, 2005.

[66] Sébastien Konieczny and Ramón Pino Pérez. Logic based merging. *Journal of Philosophical Logic*, 40(2):239–270, 2011.

[67] Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital Supply Chain Transformation toward Blockchain Integration. In *Proc. of 50th Hawaii International Conference on System Sciences*, pages 4182–4191, 2017.

[68] Gerd Kortuem, Fahim Kawsar, Vasughi Sundramoorthy, Daniel Fitton, et al. Smart Objects as Building Blocks for the Internet of Things. *IEEE Internet Computing*, 14(1):44–51, 2009.

[69] Michal Koziuk, Jaroslaw Domaszewicz, Radoslaw Olgierd Schoeneich, Marcin Jablonowski, and Piotr Boetzel. Mobile context-addressable messaging with DL-Lite domain model. In *European Conference on*

Smart Sensing and Context, pages 168–181, Berlin, Germany, 2008. Springer.

[70] SK Lakshmanaprabu, K Shankar, Ashish Khanna, Deepak Gupta, Joel JPC Rodrigues, Plácido R Pinheiro, and Victor Hugo C De Albuquerque. Effective Features to Classify Big Data Using Social Internet of Things. *IEEE Access*, 6:24196–24204, 2018.

[71] M.J. Lawley and C. Bousquet. Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In *Proc. 6th Australasian Ontology Workshop (IAOA'10). Conf.s in Research and Practice in Information Technology*, volume 122, pages 45–49, 2010.

[72] Ali Li, Xiaozhen Ye, and Huansheng Ning. Thing Relation Modeling in the Internet of Things. *IEEE Access*, 5:17117–17125, 2017.

[73] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. *Int. Jour. of Electronic Commerce*, 8(4):39–60, 2004.

[74] Yuan-Fang Li, Jeff Z Pan, Manfred Hauswirth, and Hai Nguyen. The Ubiquitous Semantic Web: Promises, Progress and Challenges. In *Web-Based Services: Concepts, Methodologies, Tools, and Applications*, pages 272–289. IGI Global, Hershey, PA, USA, 2016.

[75] Zhiyuan Li, Rulong Chen, Lu Liu, and Geyong Min. Dynamic resource discovery based on preference and movement pattern similarity for large-scale social Internet of Things. *IEEE Internet of Things Journal*, 3(4):581–589, 2016.

[76] Paolo Liberatore and Marco Schaerf. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, 10(1):76–90, 1998.

[77] Chih-Hao Lin, Pin-Han Ho, and Hong-Chuan Lin. Framework for NFC-based intelligent agents: a context-awareness enabler for Social Internet of Things. *International Journal of Distributed Sensor Networks*, 10(2), 2014.

[78] Kuan-Yu Lin and Hsi-Peng Lu. Why people use social networking sites: An empirical study integrating network externalities and motivation theory. *Computers in human behavior*, 27(3):1152–1161, 2011.

[79] Giuseppe Loseto, Saverio Ieva, Filippo Gramegna, Michele Ruta, Floriano Scioscia, and Eugenio Di Sciascio. Linked Data (in low-resource) Platforms: a mapping for Constrained Application Protocol. In P. Groth and et al., editor, *The Semantic Web - ISWC 2016: 15th Int. Semantic Web Conference, Proc., Part II*, volume 9982 of *Lecture Notes in Computer Science*, pages 131–139, Cham, oct 2016. Springer International Publishing.

[80] Ashok Malhotra, John Arwe, and Steve Speicher. Linked Data Platform 1.0. W3C Recommendation, W3C, February 2015. Available: `http://www.w3.org/TR/ldp/`.

[81] Juri Mattila, Timo Seppälä, and Jan Holmström. Product-centric Information Management: A Case Study of a Shared Platform with Blockchain Technology. In *Berkeley Roundtable on the International Economy*, 2016.

[82] Matthias Mettler. Blockchain technology in healthcare: The revolution starts here. In *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pages 1–3. IEEE, 2016.

[83] B. Motik, I. Horrocks, and S.M. Kim. Delta-Reasoner: a Semantic Web Reasoner for an Intelligent Mobile Platform. In *Twentyfirst Int. World Wide Web Conf. (WWW 2012)*. ACM, 2012. To appear.

[84] Dae-Hyeok Mun, Minh Le Dinh, and Young-Woo Kwon. An assessment of Internet of Things protocols for resource-constrained applications. In *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, volume 1, pages 555–560. IEEE, 2016.

[85] Michele Nitti, Roberto Girau, and Luigi Atzori. Trustworthiness Management in the Social Internet of Things. *IEEE Transactions on knowledge and data engineering*, 26(5):1253–1266, 2014.

[86] Michele Nitti, Maurizio Murroni, Mauro Fadda, and Luigi Atzori. Exploiting Social Internet of Things Features in Cognitive Radio. *IEEE Access*, 4:9204–9212, 2016.

[87] Alex Norta. Creation of smart-contracting collaborations for decentralized autonomous organizations. In *International Conference on Business Informatics Research*, pages 3–17. Springer, 2015.

[88] Andrei Olaru, Adina Magda Florea, and Amal El Fallah Seghrouchni. A context-aware multi-agent system as a middleware for ambient intelligence. *Mobile Networks and Applications*, 18(3):429–443, 2013.

[89] Kelly Olson, Mic Bowman, James Mitchell, Shawn Amundson, Dan Middleton, and Cian Montgomery. Sawtooth: An Introduction. Technical report, The Linux Foundation, 01 2018.

[90] Antonio M Ortiz, Dina Hussein, Soochang Park, Son N Han, and Noel Crespi. The Cluster Between Internet of Things and Social Networks: Review and Research Challenges. *IEEE Internet of Things Journal*, 1(3):206–215, 2014.

[91] Alfonso Panarello, Nachiket Tapas, Giovanni Merlino, Francesco Longo, and Antonio Puliafito. Blockchain and IoT Integration: A Systematic Survey. *Sensors*, 18(8):2575, 2018.

[92] Mrutyunjaya Panda and Ajith Abraham. Development of a reliable trust management model in social internet of things. *International Journal of Trust Management in Computing and Communications*, 2(3):229–258, 2014.

[93] Bijan Parsia, Sebastian Rudolph, Markus Krötzsch, Peter Patel-Schneider, and Pascal Hitzler. OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, W3C, December 2012. http://www.w3.org/TR/owl2-primer.

[94] Anand Paul, Awais Ahmad, M Mazhar Rathore, and Sohail Jabbar. Smartbuddy: Defining Human Nehaviors Using Big Data Analytics in Social Internet of Things. *IEEE Wireless Communications*, 23(5):68–74, 2016.

[95] Antonio Pintus, Davide Carboni, and Andrea Piras. Paraimpu: a Platform for a Social Web of Things. In *Proceedings of the 21st International Conference on World Wide Web*, pages 401–404. ACM, 2012.

[96] Veena Pureswaran and Paul Brody. Device democracy: Saving the future of the Internet of Things. Technical report, IBM Institute for Business Value, September 2014. Available: `http://www-935.ibm.com/services/us/gbs/thoughtleadership/internetofthings`.

[97] Azzurra Ragone, Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, Simona Colucci, and Francesco Colasuonno. Fully automated web services discovery and composition through concept covering and

concept abduction. *International Journal of Web Services Research (JWSR)*, 4(3):85–112, 2007.

[98] Borja Ramis Ferrer, Sergii Iarovyi, Luis Gonzalez, Andrei Lobov, and Jose L Martinez Lastra. Management of distributed knowledge encapsulated in embedded devices. *International Journal of Production Research*, pages 1–18, 2015.

[99] Ana Reyna, Cristian Martín, Jaime Chen, Enrique Soler, and Manuel Díaz. On blockchain and its integration with IoT. Challenges and opportunities. *Future Generation Computer Systems*, 88:179–190, 2018.

[100] Mohsen Rohani, Denis Gingras, Vincent Vigneron, and Dominique Gruyer. A new decentralized Bayesian approach for cooperative vehicle localization based on fusion of GPS and VANET based inter-vehicle distance measurement. *IEEE Intelligent Transportation Systems Magazine*, 7(2):85–95, 2015.

[101] M. Ruta, F. Scioscia, T. Di Noia, and E. Di Sciascio. Reasoning in Pervasive Environments: an Implementation of Concept Abduction with Mobile OODBMS. In *2009 IEEE/WIC/ACM Int. Conf. on Web Intelligence*, pages 145–148. IEEE, 2009.

[102] M. Ruta, F. Scioscia, and E. Di Sciascio. Mobile Semantic-based Matchmaking: a fuzzy DL approach. *The Semantic Web: Research and Applications*, pages 16–30, 2010.

[103] Michele Ruta, Simona Colucci, Floriano Scioscia, Eugenio Di Sciascio, and Francesco M Donini. Finding commonalities in RFID semantic streams. *The 5th International Workshop on RFID Technology Concepts, Applications, Challenges (IWRT 2011), Procedia Computer Science*, 5:857–864, 2011.

[104] Michele Ruta, Tommaso Di Noia, Eugenio Di Sciascio, Giacomo Piscitelli, and Floriano Scioscia. A semantic-based mobile registry for dynamic RFID-based logistics support. In *ICEC '08: Proc. of the 10th Int. Conf. on Electronic commerce*, pages 1–9, New York, USA, 2008. ACM.

[105] Michele Ruta, Tommaso Di Noia, Eugenio Di Sciascio, and Floriano Scioscia. An efficient data compression algorithm for semantic-based ubiquitous computing applications. In *Int. Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM07)*, pages 177–182. IARIA, IEEE Comp. Society, 2007.

[106] Michele Ruta, Eugenio Di Sciascio, and Floriano Scioscia. Concept abduction and contraction in semantic-based P2P environments. *Web Intelligence and Agent Systems*, 9(3):179–207, 2011.

[107] Michele Ruta, Floriano Scioscia, Eliana Bove, Annarita Cinquepalmi, and Eugenio Di Sciascio. A knowledge-based approach for resource discovery and allotment in swarm middleware. In *SET-222 Specialists' Meeting on Swarm Centric Solution for Intelligent Sensor Networks*, pages 16.1–16.12, 2016.

[108] Michele Ruta, Floriano Scioscia, and Eugenio Di Sciascio. Enabling the semantic web of things: framework and architecture. *Sixth IEEE International Conference on Semantic Computing (ICSC 2012)*, pages 345–347, sep 2012. doi: 10.1109/ICSC.2012.42.

[109] Michele Ruta, Floriano Scioscia, Filippo Gramegna, Ivano Bilenchi, and Eugenio Di Sciascio. Mini-me swift: the first mobile owl reasoner for ios. In *European Semantic Web Conference*, pages 298–313. Springer, 2019.

[110] Michele Ruta, Floriano Scioscia, Filippo Gramegna, and Eugenio Di Sciascio. A mobile knowledge-based system for on-board diagnostics and car driving assistance. In *UBICOMM 2010, The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 91–96. Citeseer, 2010.

[111] Michele Ruta, Floriano Scioscia, Filippo Gramegna, Saverio Ieva, Eugenio Di Sciascio, and Raffaello Perez De Vera. A Knowledge Fusion Approach for Context Awareness in Vehicular Networks. *IEEE Internet of Things Journal*, 5(4):2407–2419, 2018.

[112] Michele Ruta, Floriano Scioscia, Filippo Gramegna, Saverio Ieva, Eugenio Di Sciascio, and Raffaello Perez De Vera. A knowledge fusion approach for context awareness in vehicular networks. *IEEE Internet of Things Journal*, 5(4):2407–2419, 2018.

[113] Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, and Eugenio Di Sciascio. Semantic blockchain to improve scalability in the Internet of Things. *Open Journal of Internet Of Things (OJIOT)*, 3(1):46–61, 2017.

[114] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, and Eugenio Di Sciascio. A semantic-enabled social network of devices for building au-

tomation. *IEEE Transactions on Industrial Informatics*, 13(6):3379–3388, Dec 2017.

[115] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. Social Internet of Things for domotics: A knowledge-based approach over LDP-CoAP. *Semantic Web*, 9:781–802, 2018.

[116] Michele Ruta, Floriano Scioscia, Agnese Pinto, Filippo Gramegna, Saverio Ieva, Giuseppe Loseto, and Eugenio Di Sciascio. Coap-based collaborative sensor networks in the semantic web of things. *Journal of Ambient Intelligence and Humanized Computing*, 10(7):2545–2562, 2019.

[117] F Sandu, C Costache, and T Balan. Semantic data aggregation in heterogeneous learning environments. In *2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pages 409–412. IEEE, 2015.

[118] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial intelligence*, 48(1):1–26, 1991.

[119] Guus Schreiber and Fabien Gandon. RDF 1.1 XML syntax. W3C recommendation, W3C, February 2014. http://www.w3.org/TR/rdf-syntax-grammar/.

[120] Schreiber, Guus and Raimond, Yves. RDF 1.1 Primer. W3C Recommendation, W3C, June 2012. https://www.w3.org/TR/rdf11-primer/.

[121] Floriano Scioscia and Michele Ruta. Building a Semantic Web of Things: issues and perspectives in information compression. In *Semantic Web Information Management (SWIM'09). In Proc. of the 3rd IEEE Int. Conf. on Semantic Computing (ICSC 2009)*, pages 589–594. IEEE Computer Society, 2009.

[122] Floriano Scioscia, Michele Ruta, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. A mobile matchmaker for the ubiquitous semantic web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(4):77–100, 2014.

[123] Floriano Scioscia, Michele Ruta, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva, Agnese Pinto, and Eugenio Di Sciascio. Mini-ME matchmaker and reasoner for the Semantic Web of Things. In *Innovations,*

Developments, and Applications of Semantic Web and Information Systems, pages 262–294. IGI Global, Hershey, PA, USA, 2018.

[124] Christian Seitz and René Schönfelder. Rule-based OWL reasoning for specific embedded devices. In *International Semantic Web Conference*, pages 237–252, Berlin, Germany, 2011. Springer.

[125] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The Semantic Web revisited. *IEEE intelligent systems*, 21(3):96–101, 2006.

[126] R. Shearer, B. Motik, and I. Horrocks. Hermit: A highly-efficient owl reasoner. In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008)*, pages 26–27, 2008.

[127] Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, June 2014.

[128] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.

[129] Ayesha Siddiqa, Munam Ali Shah, Hasan Ali Khattak, Adnan Akhunzada, Ihsan Ali, Zaidi Bin Razak, and Abdullah Gani. Social Internet of Vehicles: Complexity, Adaptivity, Issues and Beyond. *IEEE Access*, 6, 2018.

[130] A. Sinner and T. Kleemann. KRHyper - In Your Pocket. In *Proc. of 20th Int. Conf. on Automated Deduction (CADE-20)*, pages 452–457, Tallinn, Estonia, July 2005.

[131] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[132] Alexander Smirnov and Tatiana Levashova. Knowledge Fusion Patterns: A Survey. *Information Fusion*, 52:31–40, 2019.

[133] L. Steller and S. Krishnaswamy. Pervasive Service Discovery: mTableaux Mobile Reasoning. In *Int. Conf. on Semantic Systems (I-Semantics). Graz, Austria*, 2008.

[134] Nick Szabo. Formalizing and securing relationships on public networks. *First Monday*, 2(9), 1997.

[135] W. Tai, J. Keeney, and D. O'Sullivan. COROR: a composable rule-entailment owl reasoner for resource-constrained devices. *Rule-Based Reasoning, Programming, and Applications*, pages 212–226, 2011.

[136] Gunnar Teege. Making the difference: A subtraction operation for description logics. In *Principles of Knowledge Representation and Reasoning*, pages 540–550. Elsevier, 1994.

[137] Nikolay Teslya and Alexander Smirnov. Blockchain-based framework for ontology-oriented robots' coalition formation in cyberphysical systems. In *MATEC Web of Conferences*, volume 161, page 03018. EDP Sciences, 2018.

[138] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. *Automated Reasoning*, pages 292–297, 2006.

[139] William Van Woensel and Syed Sibte Raza Abidi. Optimizing Semantic Reasoning on Memory-Constrained Platforms Using the RETE Algorithm. In *Extended Semantic Web Conference (ESWC)*, pages 682–696. Springer, 2018.

[140] András Varga and Rudolf Hornig. An overview of the OMNeT++ simulation environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 60. ICST, 2008.

[141] Juan Ignacio Vazquez, Diego López De Ipiña, and Iñigo Sedano. Soam: an environment adaptation model for the pervasive semantic web. In *Computational Science and Its Applications-ICCSA 2006*, pages 108–117. Springer, 2006.

[142] Orfefs Voutyras, Panagiotis Bourelos, Spyridon Gogouvitis, Dimosthenis Kyriazis, and Theodora Varvarigou. Social monitoring and social analysis in Internet of Things virtual networks. In *Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on*, pages 244–251. IEEE, 2015.

[143] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[144] Huaiqing Wang, Kun Chen, and Dongming Xu. A maturity model for blockchain adoption. *Financial Innovation*, 2(12), 2016.

[145] Jin Wang, Jiayi Cao, Bin Li, Sungyoung Lee, and R Simon Sherratt. Bio-inspired ant colony optimization based clustering algorithm with mobile sinks for applications in consumer home automation networks. *IEEE Transactions on Consumer Electronics*, 61(4):438–444, 2015.

[146] Zheng Yan, Peng Zhang, and Athanasios V Vasilakos. A survey on trust management for Internet of Things. *Journal of network and computer applications*, 42:120–134, 2014.

[147] Roberto Yus, Carlos Bobed, Guillermo Esteban, Fernando Bobillo, and Eduardo Mena. Android goes semantic: Dl reasoners on smartphones. In *Proc. of 2nd International Workshop on OWL Reasoner Evaluation (ORE'13), Ulm (Germany)*, pages 46–52, 2013.

[148] Long Zhang, Zhen Zhao, Qiwu Wu, Hui Zhao, Haitao Xu, and Xiaobo Wu. Energy-aware dynamic resource allocation in uav assisted mobile edge computing over social internet of vehicles. *IEEE Access*, 6:56700–56715, 2018.

[149] Mengchu Zhou, Giancarlo Fortino, Weiming Shen, Jin Mitsugi, James Jobin, and Rahul Bhattacharyya. Guest Editorial: Special Section on Advances and Applications of Internet of Things for Smart Automated Systems. *IEEE Transactions on Automation Science and Engineering*, 13(3):1225–1229, 2016.

Appendix A

Concept Fusion support subroutines

Require: $\leq x\ R$ less than restriction in TBox $\mathcal{T}$,
$\mathcal{G} = \{\mathcal{G}_{\mathcal{CN}},\ \mathcal{G}_{\leq},\ \mathcal{G}_{\geq},\ \mathcal{G}_{\forall}\}$ 4-tuple of sets of concept components in $\mathcal{ALN}$ w.r.t. $\mathcal{T}$
Ensure : G is properly updated

if $\{\leq z\ R\} \in \mathcal{G}_{\leq}$ ***and*** $z > x$ **then**
  $\mathcal{G}_{\leq} := (\mathcal{G}_{\leq} \setminus \{\leq z\ R\}) \cup \{\leq x\ R\}$;
else if $\{\leq z\ R\} \notin \mathcal{G}_{\leq}$ **then**
  $\mathcal{G}_{\leq} := \mathcal{G}_{\leq} \cup \{\leq x\ R\}$;
end

Algorithm 7: *addLessThanRestriction*$(\leq x\ R,\ \mathcal{G})$

Require: $\geq z\ R$ greater than restriction in TBox $\mathcal{T}$,
$\mathcal{G} = \{\mathcal{G}_{\mathcal{CN}},\ \mathcal{G}_{\leq},\ \mathcal{G}_{\geq},\ \mathcal{G}_{\forall}\}$ 4-tuple of sets of concept components in $\mathcal{ALN}$ w.r.t. $\mathcal{T}$
Ensure : G is properly updated

if $\{\geq z\ R\} \in \mathcal{G}_{\geq}$ ***and*** $z < x$ **then**
  $\mathcal{G}_{\geq} := (\mathcal{G}_{\geq} \setminus \{\geq z\ R\}) \cup \{\geq x\ R\}$;
else if $\{\geq z\ R\} \notin \mathcal{G}_{\geq}$ **then**
  $\mathcal{G}_{\geq} := \mathcal{G}_{\geq} \cup \{\geq x\ R\}$;
end

Algorithm 8: *addGreaterThanRestriction*$(\geq z\ R,\ \mathcal{G})$

Require: $\forall R.B$ universal restriction with B satisfiable in $\mathcal{T}$,
$\mathcal{G} = \{\mathcal{G}_{\mathcal{CN}},\ \mathcal{G}_{\leq},\ \mathcal{G}_{\geq},\ \mathcal{G}_{\forall}\}$ set of sets of concepts in $\mathcal{ALN}$ w.r.t. $\mathcal{T}$

Ensure : G is properly updated

```
1  if {∀R.E} ∈ G_∀ then
2      foreach (possibly negated) CN in B do
3          if CN is not in E_CN then
4              E_CN := E_CN ∪ {CN} ;
5          end
6      end
7      foreach {≤ x R} ∈ B_≤ do
8          addLessThanRestriction(≤ x R, E) ;
9      end
10     foreach {≥ xR} ∈ B_≥ do
11         addGreaterThanRestriction(≥ x R, E) ;
12     end
13     foreach {∀R.D} ∈ B_∀ do
14         addUniversalRestriction(∀R.D, E) ;
15     end
16 else
17     G_∀ := G_∀ ∪ {∀R.B} ;
18 end
```

Algorithm 9: *addUniversalRestriction*$(\forall R.B,\ \mathcal{G})$

List of publications

Journal Articles

1. Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso and Eugenio Di Sciascio. *Semantic blockchain to improve scalability in the internet of things*, In Open Journal of Internet Of Things (OJIOT), RonPub, volume 3, pp. 46-61, 2017

Peer-Reviewed Conference Papers

1. Floriano Scioscia, Agnese Pinto, Filippo Gramegna, Giovanna Capurso, Danilo De Filippis, Raffaello Perez de Vera, Eugenio Di Sciascio. *Supporting Environmental Analysis and Requalification of Taranto Sea: an Integrated ICT Platform* In The Tenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2016), 2016

2. Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso and Eugenio Di Sciascio. *Supply chain object discovery with semantic-enhanced blockchain*, In Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, pp. 60, 2017

3. Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, Giuseppe Loseto, Filippo Gramegna, Agnese Pinto and Eugenio Di Sciascio. *Semantic-enhanced blockchain technology for smart cities and communities*, In 3rd Italian conference on ICT for Smart Cities & Communities (ICiTies 2017), 2017

4. Giovanna Capurso, Michele Ruta, Floriano Scioscia and Eugenio Di Sciascio. *Object (B)logging: Semantic Self-Description for Cyber-Physical Systems* In RuleML+ RR (Supplement), 2017

5. Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, Agnese Pinto and Eugenio Di Sciascio. *A Blockchain Infrastructure for*

the Semantic Web of Things, In 26th Italian Symposium on Advanced Database Systems (SEBD 2018), 2018

6. (In Italian) Michele Ruta, Floriano Scioscia, Giuseppe Loseto, Giovanna Capurso, Agnese Pinto and Eugenio Di Sciascio. *Semantic Web of Things: dalla rappresentazione degli oggetti alla decisione automatica* In Convegno Nazionale CINI sull'Intelligenza Artificiale (Ital-IA 2019), 2019

7. Michele Ruta, Floriano Scioscia, Saverio Ieva, Giovanna Capurso, Eugenio Di Sciascio. *Semantic matchmaking as a way for attitude discovery*, In 8th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI 2019), 2019

8. Giovanna Capurso, Michele Ruta, Eugenio Di Sciascio. *Object (B)logging: a Decentralized Cognitive Paradigm for the Industrial Internet of Things*, In 2019 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2019), 2019

Others

1. Giovanna Capurso. *Integrating machines and artificial intelligence for predictive maintenance in advanced production systems* In 1st workshop of Poliba Phd Student Research (PHDAYS 2017), 2017

Academic activities

Attended Seminars/Courses

1. Doctoral school courses (ScuDo) offered by Polytecnic University of Bari:
    - Applications of MATLAB (academic year 2016/2017)
    - Elements of Probability for Engineering Sciences (academic year 2016/2017)
    - Theory and applications of stochastic processes (academic year 2016/2017)
    - Middleware and architecture for Industry 4.0 (academic year 2016/2017)
    - How and why to build an ontology to model a scenario (academic year 2016/2017)
    - How to write a technical paper and to present it effectively to an educated audience (academic year 2017/2018)
2. Doctoral school courses (ScuDo) offered by University of Bari "Aldo Moro":
    - Python Programming (academic year 2017/2018)
3. *13th Reasoning Web Summer School (RW 2017) "Semantic Interoperability on the Web"* held at Birkbeck, University of London in London, UK
4. *14th LASER Summer School on Software Engineering (LASER 2018) "Software Technology for Blockchains, Bitcoin and Distributed Trust Systems"* held at Elba Island, Italy
5. PhD Seminar *Machine Learning and Condition-based Monitoring, Neural Networks and State Estimation* taught by Prof. Dr. Anselm Haselhoff from Computer Science Institute, University of Applied Sciences Hochschule Ruhr West (Bottrop, Germany)

6. *29th Summer PhD School of Information Engineering (SSIE 2019) "AI and Machine Learning for ICT Applications"* held at Brixen, Italy

7. *14th EDBT Summer School "Extracting Hidden Knowledge from Heterogeneous Massive Data"* held at Lyon, France

Attended Conferences/Workshops

1. Attended *International Joint Conference on Rules and Reasoning (RuleML+RR 2017)* held at London, UK and presented the poster paper *Object (B)logging: Semantic Self-Description for Cyber-Physical Systems*

2. Attended *International Workshop on Very Large Internet of Things (VLIoT at VLDB 2017* held at Munich, Germany and presented the paper *Semantic Blockchain to Improve Scalability in the Internet of Things*

3. Attended *Italian Conference on ICT for Smart Cities & Communities (ICiTies 2017)* held at Bari, Italy and presented the paper *Semantic-enhanced blockchain technology for smart cities and communities*

4. Attended *The 15th ACM Conference on Embedded Networked Sensor Systems (SenSys 2017)* held at Delft, The Netherlands and presented the paper *Supply Chain Object Discovery with Semantic-enhanced Blockchain*

5. Attended *1st workshop of Poliba Phd Student Research (PHDAYS 2017)* held at Bari, Italy and joined the challenge *PhD Student Research Competition (SRC)* by presenting the paper *Integrating machines and artificial intelligence for predictive maintenance in advanced production systems*

6. Attended *26th Italian Symposium in Advanced Database Systems (SEBD 2018)* held at Castellaneta Marina (Taranto), Italy and presented the paper *A Blockchain Infrastructure for the Semantic Web of Things*

7. Session Chair at *2nd International Conference on Computational Biology and Bioinformatics (ICCBB 2018)* held at Bari, Italy

8. Attended *2019 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2019)* held at Bari, Italy and presented the paper *Object (B)logging: a Decentralized Cognitive Paradigm for the Industrial Internet of Things*

Teaching Experience

1. Taught lectures for the *Operating Systems* courses (Bachelor's Degree in *Computer and Automation Engineering* and Master's Degree in *Telecommunications Engineering* of Polytechnic University of Bari). They usually cover Linux operating system and shell bash, Android operating system and Robot Operating System (ROS)

2. Taught lectures for the *Embedded and Certified Software* (Master's Degree in *Aerospace Engineering* of University of Salento) course. They usually cover Android operating system and Robot Operating System (ROS)