

Social Internet of Things for Domotics: a Knowledge-based Approach over LDP-CoAP

Michele Ruta^{a,*}, Floriano Scioscia^a, Giuseppe Loseto^a, Filippo Gramegna^a, Saverio Ieva^a, Agnese Pinto^a, and Eugenio Di Sciascio^a

^a *Politecnico di Bari, Dipartimento di Ingegneria Elettrica e dell'Informazione, via E. Orabona 4, I-70125, Bari, Italy*

E-mail: name.surname@poliba.it

Abstract. Ambient Intelligence aims at simplifying the interaction of a user with her surrounding context minimizing the effort needed to increase comfort and assistance. Nevertheless, especially in built and structured environments, current technologies and market solutions are often far to provide the required levels of automation, coordination and adaptivity of the ambient. This paper proposes a novel semantic-based framework complying the emerging Social Internet of Things paradigm. Infrastructured spaces can be intended as populated by device agents organized in social networks, interacting autonomously and sharing information, cooperating and orchestrating resources. A service-oriented architecture allows collaborative dissemination, discovery and composition of service/resource descriptions. The Semantic Web languages are adopted as linguistic layer and mobile-oriented implementations of non-monotonic inferences for semantic matchmaking are used to give decision capabilities to software agents. Finally, the Linked Data Platform (LDP) over the Constrained Application Protocol (CoAP) provides the knowledge organization and sharing infrastructure underpinning social object interactions. The framework has been implemented and tested in a home automation prototype integrating several communication protocols and off-the-shelf devices. Experiments advocate the effectiveness of the approach.

Keywords: Semantic Web of Things, Objects Social Networks, Building Automation, Linked Data Platform, Constrained Application Protocol

Eventually everything connects – people, ideas, objects. The quality of the connections is the key to quality per se.

Attributed to Charles Eames [5]

1. Introduction

In the *Ambient Intelligence* (AmI) vision, built environments interact with their inhabitants in a “unobtrusive, interconnected, adaptable, dynamic, embedded and intelligent” way [36]. Personal requirements and preferences are grasped, deciphered and formalized as well as the environment can adapt to them, and even anticipate people’s needs and behaviors. The AmI

idea leverages technological progress in the *Internet of Things* (IoT), where large numbers of everyday objects are augmented with communication and computation capabilities. People in their usual environments are increasingly surrounded by networks of micro-devices, endowed with embedded sensors for data capture as well as processing units for deriving context information. To create real cohesive AmI, such devices should communicate and coordinate autonomously, making decisions dynamically based on manifold factors, including the state of surroundings objects and places as well as user activities and profiles. While traditional human-computer interaction has been explicit and mediated by input peripherals, in AmI predominate implicit, effortless interaction paradigms, where relevant information about users’ goals and intentions is inferred automatically by analyzing their actions and

* Corresponding author. E-mail: michele.ruta@poliba.it.

context, through sensors integrated in the environment or in wearable stuff.

Current solutions for Home and Building Automation (HBA) are still far from the above levels of intelligence, automation and adaptivity. They grant limited flexibility, as devices are logically associated at the application level by means of static profiles, defined at system deployment stage. With most established HBA standards, changing the set of possible configurations or introducing new devices require the intervention of qualified practitioners. Recently, product manufacturers and system integrators have proposed more user-friendly “smart home” devices and platforms, leveraging the IoT [23]. Unfortunately, solutions are proprietary and centralized, and they still require manual configuration. This seemingly improved usability comes at the price of providing only very basic automation [24], typically using Event-Condition-Action (ECA) rules on simplistic threshold or on/off conditions.

Hence, significant technological advances are needed to fully accomplish the AmI vision. Flexible and meaningful relationships among devices in a given environment should be possible, established automatically to support articulate orchestration and coreography patterns. Recent research in the so-called *Social Internet of Things* [2] is starting to define models and architectures to reach this goal. Paradigms are often borrowed from Social Networking Services (SNS) for human users. If properly adapted to the peculiarities and requirements of Multi-Agent Systems (MAS), they can support powerful approaches. This is not enough, however, for true AmI: 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 high-level formalisms. Semantic Web technologies are natural candidates for such a role, as they provide interoperable languages and tools grounded on formal logic semantics [38]. Semantic Web standards enable knowledge modeling, assertion, organization, querying and inference in distributed systems, but technologies and tools require proper adaptation to work efficiently in resource-constrained environments like the IoT. The *Semantic Web of Things* [33] aims at the convergence of the Semantic Web and IoT visions, endowing environments with intelligence by means of semantic metadata dynamically produced by ubiquitous micro-devices to characterize sensor data, detected events and phenomena, objects, places and other relevant entities in a context. Due to the volatility

and unpredictability of mobile and IoT environments, device and service discovery are two major challenges in the SWoT. Achieving acceptable performance also requires attention, as Semantic Web tools, protocols and languages are typically too resource-consuming for current IoT devices. Application-level protocols and reasoning tools for the (Semantic) Web must be properly adjusted, tailoring their feature set to pervasive computing contexts.

This paper presents a possible approach for a Semantic Social Internet of Things grounded on Ambient Intelligence scenarios. According to the SWoT paradigm, standard technologies were adapted to provide a cohesive knowledge and service discovery architecture. The proposal leverages: i) the Linked Data Platform (LDP) [40] to annotate and organize information resources and ii) the Constrained Application Protocol (CoAP) [8] –a proposed IETF¹ standard RESTful protocol– for resource exchange in constrained environments, as it is more efficient than HTTP. Above this knowledge/service interoperability layer, a semantic-enhanced application level enables social networking among “agentified” things.

Borrowing core relationships and structure from popular SNSs, devices enable specific interaction patterns for information sharing and cooperative decentralized service/resource discovery. Such selective choreography is triggered autonomously, based on the kind of managed resources and other contextual factors; this capability enhances scalability in dense multi-agent environments. Resource discovery exploits semantic matchmaking between ontology-based annotations which describe requests and available resources. Non-standard, non-monotonic inferences [32] implemented in the *Mini-ME* mobile matcher and reasoner [37] allow supporting approximated matches, resource ranking and aggregation for covering complex requests. The framework also supports basic and legacy devices, which do not have computational power enough for on-board reasoning, by allowing them to select a more capable friend as inference facilitator.

The general framework outlined above has been focused on smart HBA, to provide AmI experiences in residential and workplace settings. It was implemented and evaluated in a real prototypical testbed, encompassing diverse device types, communicating across different wired and wireless HBA protocols. Experi-

¹ Internet Engineering Task Force, <https://www.ietf.org>

mental evidences are reported and assess framework feasibility and effectiveness.

The remainder of the paper is as follows. Section 2 discusses the state of the art, while the framework is described in detail in Section 3. Section 4 presents a case study to further clarify the proposal and its benefits. Experimental evaluations in Section 5 provide an assessment of both practicability and efficiency of the proposed approach, before conclusion.

2. State of the art: pervasive computing in the social networks epoch

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 [27], because SNSs provide both utility (*extrinsic* value) and gratification (*intrinsic* value). Their usefulness also grows as they connect more users, and particularly *complementary* ones [27], since opportunities increase for discovering interesting information and services.

A social evolution of pervasive computing [2] envisions objects acting as independent agents, capable of establishing relationships and using them to share information and services more effectively. This may allow to reap the above benefits in advanced IoT scenarios; actually, it is reasonable to expect them to be higher in large and heterogeneous networks, such as in HBA. An in-depth analysis of object social networks is in [41], which discussed key metrics about nodes and links by adapting from and expanding upon the social network analysis literature. Definitions were formalized in an ontology that objects can use to manage their policies, friends and reputation (main difference with the present paper is that the considered friendship model is asymmetrical). Further ontology proposals exist to formalize models of the social networking domain, *e.g.*, [4]. Particularly, in [2], things engage with one another in social networks independently from human SNSs and from user interactions. A relevant case

[31] included social object capabilities in control networks, aiming at distributed Web Ontology Language (OWL) Knowledge Base (KB) management and inference. When connecting to the network, every object proactively exchanged information with other devices in a handshake process. “Requester” devices, equipped with reasoning facilities, could then distribute queries automatically among “known” devices. Unfortunately, the adopted query language supported only very simple inferences, limiting the practical usefulness of the proposal. Another research direction has been focusing on the integration of the IoT into the social context of human users [4], either to improve adaptivity in Aml [22] or to monitor users and assist them in personalizing their SNS experience and interactions [29]. In [21] semantic-based situation detection and goal retrieval were used for matchmaking with task profiles to recommend activities to users, based on their current context. Unlike our approach, social interactions occurred only between devices and users; furthermore, adopted rule-based reasoning could not retrieve approximate matches when exact ones did not exist. A further effort to achieve social capabilities is *object blogging*, defined as an object’s capability of annotating and publishing its history and context on the Web and/or in a mobile ad-hoc network, supporting intelligent machine-to-machine interactions. Some proposed approaches required user intervention [13], while others aimed at autonomous self-description and decision-making [9].

Many of the above works combine social networks of pervasive objects with semantic technologies. Indeed, semantic-based approaches have wide adoption in pervasive MAS, and smart building automation is one of the most relevant areas [17,35]. Ontologies have been used in all stages of the lifecycle of HBA systems, including design and deployment, infrastructure description, data modeling and access, and device control [14,7]. In [16] an ontology-based building automation system delivered context-aware information in a customized way to different kinds of users, *e.g.*, upkeep and healthcare operators in a clinic. OWL device and user descriptions were matched through SPARQL queries and SWRL rules were used to implement temporal and extra-logical constraints, achieving capabilities similar to classical Complex Event Processing (CEP) systems. Nevertheless, the solution was affected from poor maintainability, because installing new devices required not only manual configuration, but also changes to the reference ontology. The proposed architecture in [7] included a reason-

ing module exploiting rule-based inferences. Unfortunately, the system state should fully match the rule head in order to trigger its body. Full matches seldom occur in realistic scenarios, whose entities are featured by detailed, heterogeneous and often contradictory information, unless one uses very basic rules. In our approach, non-monotonic inference services allow supporting approximate matches, which can yield “good enough” results whenever full matches are not available.

3. A social framework for smart linked objects

In what follows the proposed framework, architecture and technologies are described.

3.1. Knowledge-based architecture

The approach proposed here aims at object coordination in purposely infrastructured environments and particularly in domotics scenarios through interaction paradigms borrowed from social networks. The main goal is allowing devices (a.k.a. nodes) to gain wide agency and autonomy in sharing information and services, enabling them to distribute requests and obtain responses through fully decentralized peer-to-peer (P2P) interactions also assuming decisions.

Service-oriented architecture. Each node is a social object, which exposes an individual profile, describing its basic features (device type, location, hardware details) and the resources/services it can provide, e.g., its possible configurations and functional profiles. A node makes *posts* on its wall when its settings or capabilities change, and also when it produces new or updated information through context sensing and analysis. Posts are expressed as semantic annotations referred to ontologies in Web Ontology Language (OWL 2) [30], formally grounded on Description Logics (DLs) semantics. A decentralized service-oriented architecture (SOA) underlies the whole proposed social network model, where shared knowledge fragments about devices, functional profiles and context represent annotated service/resource advertisements.

Semantic matchmaking. Service/resource discovery conveys decision capabilities of nodes. As stated before, this collaborative process leverages semantic matchmaking, i.e., the overall process allowing the retrieval and ranking of the most relevant resources for a given request, where both resources and requests are satisfiable concept expressions w.r.t. a common ontol-

ogy \mathcal{T} in a DL \mathcal{L} . From a linguistic point of view, this paper refers to the OWL2 DLs subset corresponding to the \mathcal{ALN} (Attributive Language with unqualified Number restrictions) Description Logics, as it is supported by an embedded matchmaking and reasoning engine which provides the required inference services [37]. Standard reasoning services for matchmaking include *Subsumption* and *Satisfiability*. Given a request R and a resource S , subsumption verifies whether all features in R are included in S : its outcome is either *full match* or not. Satisfiability checks whether any constraint in R contradicts some specification in S , hence it divides resources in *compatible* (a.k.a. *potential matches*) and *incompatible* (a.k.a. *partial matches*) w.r.t. the request. The boolean *full or no-match* approach is inadequate for advanced scenarios, because full matches are rare and incompatibility is frequent when dealing with articulated concept expressions from heterogeneous sources.

In order to produce a finer resource ranking and a logic-based explanation of outcomes, the framework proposed here extends the basic subsumption/satisfiability setting exploiting the following non-standard inference services [37]:

- *Concept Abduction*: whenever R and S are compatible, but S does not imply R , Abduction allows to determine what should be hypothesized in S in order to completely satisfy R ;
- *Concept Contraction*: if request R and resource S are not compatible, Contraction determines which part of R is conflicting with S . If one retracts conflicting requirements in R , G (for *Give up*), a concept K (for *Keep*) is obtained, representing a contracted version of the original request, such that $K \sqcap S$ is satisfiable w.r.t. \mathcal{T} ;
- *Concept Covering*: pervasive computing scenarios often require relatively large number of resources to be aggregated in order to satisfy a complex request. To this aim, a further non-standard reasoning task based on the solution of *Concept Covering Problem* (CCoP) has been defined. It allows to: (i) satisfy features expressed in a request as much as possible, through the conjunction of one or more small instances of a KB –seen as elementary knowledge blocks– and (ii) provide explanation of the uncovered part of the request itself. Given a request R and a set of available resources $S = \{S_1, S_2, \dots, S_n\}$, all satisfiable in the reference ontology \mathcal{T} , *Concept Covering* aims to find a pair $\langle S_c, H \rangle$ where S_c includes concepts in S covering R w.r.t. \mathcal{T} and H is the residual part of R not covered by concepts in S_c .

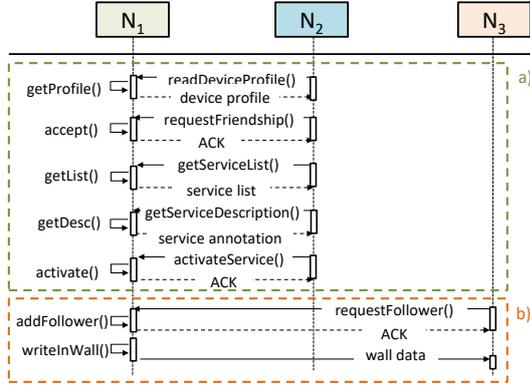


Fig. 1. Sequence diagram for device relationships: a) friend; b) follower

Social entities and relationships. The above inference services are used to regulate the interactions between social objects. Nodes are grouped in two families:

- *Smart*: devices able to perform reasoning tasks exploiting non-standard deductions;
- *Basic*: low-memory, low (or no)-computing power devices. They can only provide sensing/acting services, but don't perform autonomous reasoning.

A pair of objects can establish a social relationship following the basic interaction pictured in Figure 1. Two schemes are implemented:

- *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 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 behavior.

The decision of N_j to require the N_i 's friendship is based on information in N_i 's profile. For example, if two devices are located in the same room/area or share annotations referred to the same ontology, they are good candidates for friendship. On the other hand, N_k asks to become a follower of N_i if there is low utility in a strict interaction –*e.g.*, two devices are deeply different or belong to distinct owners or areas– but information produced by N_i can still be useful to N_k to characterize its own context.

Like in SNSs, in the framework proposed here object's wall is the main mean for sharing knowledge. Both *push* and *pull* models are supported, through the above relationships. In a nutshell, if a node wants to receive updates from another node automatically, it will ask to become a follower; if it wants to be able to access the other node's wall on demand, it will ask to become a friend (and in doing so it will also grant access to its own wall). Every agent will select either model –or even both– depending on its application requirements.

Collaborative adaptivity. When a node detects changes in internal or contextual conditions requiring adaptation, it writes a post on its own wall. A post P is modeled as a pair $\langle R, L \rangle$. R is the reconfiguration requested by the node; L is the *like* value, *i.e.*, the percentage of coverage w.r.t. R . The post triggers a collaborative service discovery process to reconfigure the environment, as exemplified in Figure 2. It consists of the following steps:

- 1) When a node N_i detects a reconfiguration is needed, it writes a post P_i on its own wall. Initially, L_i is set to 0.
- 2) If N_i is a basic device, go to step 3. Otherwise, N_i executes the Concept Covering task on the local set of services S . Upon completion, N_i adds a *comment* C_i to P_i as a pair $\langle U_i, T_i \rangle$, where U_i is the uncovered part of R_i and T_i tags the local selected services/resources. Moreover, the value of L_i is updated to the obtained score.
- 3) If R_i is not completely covered, N_i selects a friend N_j and writes a post $P_j = \langle R_j, L_j \rangle$ on its wall. Particularly, if N_i has executed step 2, R_j is set to the uncovered part U_i , otherwise R_j is equal to R_i and L_j is reset. Furthermore, N_j requests to be notified when a comment is added to P_j . N_j recursively executes the tasks 2) - 3).
- 4) When N_i receives the notification of P_j , it reads the comment from the friend's wall, which is appended to P_i to update the status of the request. Finally, N_i updates the *like* value according to the overall covering score.

Some remarks may be useful:

- The recursive discovery procedure can be applied in a depth search with no theoretical bounds. Agents can manage heuristics to decide the practical depth limit.
- The choice of friend(s) to call in the above step 3 also depends on heuristic preference criteria, such as the number and type of services exposed by the friend (known at friendship establishment time), network latency or friend's computational resources.

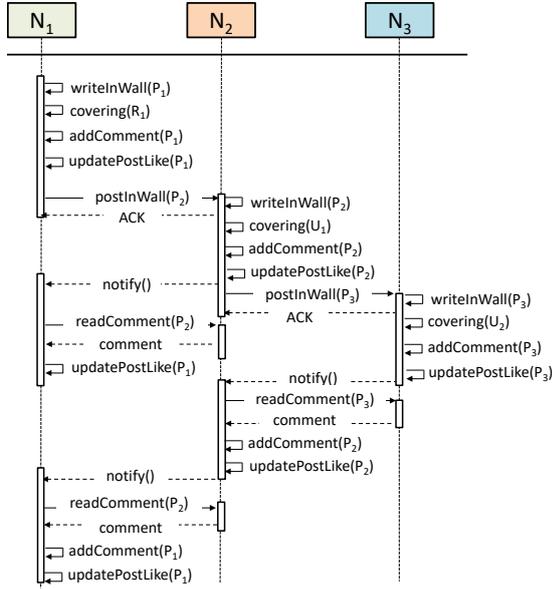


Fig. 2. Sequence diagram of distributed reconfiguration

– Main purpose of comments is to keep track of the progressive fulfillment of an adaptation request, exploiting tagging to avoid duplication of service/resource selection.

3.2. Interoperability layer: LDP-CoAP interface

At the application layer, the reference above architecture is implemented on a LDP-CoAP framework [28]. The *Linked Data Platform* W3C Recommendation [40] provides standard rules for accessing and managing Linked Data on the Web. Basically, it defines a set of communication patterns based on HTTP methods and headers for CRUD (Create, Read, Update, Delete) operations as well as different types of LDP *Resources* (LDPs): *RDF Source* (LDP-RS), whose status corresponds to an RDF graph and can be fully represented in an RDF syntax; *Non-RDF Source* (LDP-NR), not represented in RDF (e.g., a binary or text document without useful RDF annotation); *Basic* (LDP-BC), *Direct* (LDP-DC) and *Indirect* (LDP-IC) containers, defining collections of LDP resources according to specific membership patterns.

LDP specification only supports the HTTP protocol, which requires not negligible bandwidth, processing and memory resources for most IoT devices. LDP-CoAP variant, on the contrary, aimed to integrate LDP in resource-constrained devices and networks just leveraging CoAP [8], a compact counterpart of HTTP

conceived for machine-to-machine (M2M) communication. Some CoAP options are derived from HTTP header fields (e.g., content type, headers and proxy support), while some other ones have no analogous in HTTP. In any case, the HTTP-CoAP mapping, included in the LDP-CoAP framework, can be exploited to support all LDP features with CoAP.

In the present case, social devices communicate over the network through CoAP messages. Basically, each message is composed of: (i) a 32-bit *header*, containing the request method code or response status; (ii) an optional *token* value, used to associate replies to requests, (iii) a sequence of *option* fields (containing information such as resource URI and payload media type), (iv) the *payload* data. CoAP adopts the CoRE Link Format specification [39] for resource discovery. A client accesses the reserved `.well-known/core` URI on the server via GET to retrieve available resource entry points. Further GET requests will include `URI-query` options to retrieve only resources with given attributes. Standardized query attributes include resource type (`rt`), interface usage (`if`), content-type (`ct`), and MIME (Multipurpose Internet Mail Extension) `type` for a resource. Further non-reserved attributes can be freely used. CoAP also provides push notifications without polling [18], a useful feature when data have to be monitored over time (e.g., in case of *follower* relationship). CoAP also supports proxies, enabling Web applications (i.e., HTTP clients) to transparently access the resources hosted in devices based on CoAP.

Each device in the social network is modeled as an LDP-CoAP node exposing the resources reported in Table 1. The *profile* resource exposes main device features as an RDF-based annotation. An example is reported in Figure 3. In addition to well-known RDF vocabularies, a so-called *Semantic Web of Social Things* (SWST) ontology has been defined to model basic elements of a social device. In particular, each profile contains the following properties:

- type of device, according to the classification proposed by the *M3-lite* taxonomy [1], a lightweight version of the *Machine-to-Machine Measurement* (M3) ontology used to describe sensor measurements and observations;
- device name, using the `dcterms:title` property of the *DCMI Metadata Terms* vocabulary [15];
- supported ontologies (`dcterms:requires`) used as reference vocabularies to define the OWL-based annotations of the functionalities exposed by the device;
- location of the device (e.g., in a area/building/department/apartment),

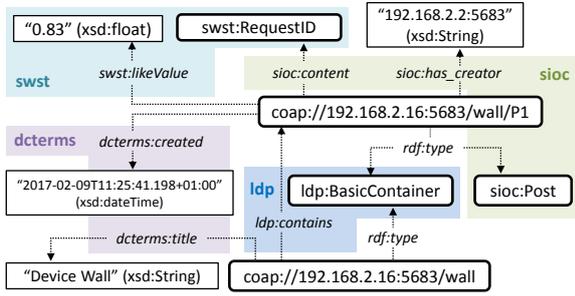


Fig. 5. Wall and posts

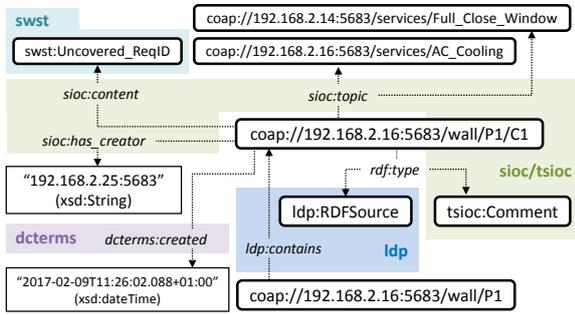


Fig. 6. Comment to a post

(*sioc:content*), as IRI of the individual representing the received OWL annotation; like value (*swst:likeValue*). Finally, a comment annotation (Figure 6) consists of: creation date; sender device; content of the comment, corresponding to the part of the post the friend device is not able to cover; tagged (*i.e.*, activated) services (*sioc:topic*), selected through the covering process.

According to the resource modeling described above, all social network interactions can be implemented as request/response messages over LDP-CoAP. In order to clarify the proposed approach, some reference examples are shown in Figure 7. RDF annotations are reported in Turtle syntax [11] but can be also retrieved in JSON-LD [26], setting the *Accept* header appropriately. As shown in Example 8 and in Table 1, OWL annotations are treated as LDP-NR resources, in order to support any OWL concrete syntax, not only RDF-based ones.

4. Case study: semantic web of (social) things for building automation

This section presents a case study, devoted to clarify the social and collaborative features of the proposed framework in terms of orchestration of smart devices

in a complex HBA context. Let us consider the example scenario, depicted in Figure 8. *Two apartments on the same floor of a building, H₁ and H₂, include a set of semantic-enabled devices forming a home social network.* In particular, H₁ is configured with an alarm system (AS), a rolling shutter controller (SC₁), an air conditioner (AC₁) and a dimmer lamp (L₁). A weather station (WS), a rolling shutter controller (SC₂), an air conditioner (AC₂) and a dimmer lamp (L₂) are installed in H₂ instead. The blue arrows in Figure 8 specify the existing friendship relations between the different devices. As said, when a friendship relation is established, each friend is able to directly read the wall of an object, write a post on its wall and use the services of the device. Within the two apartments, each object has sensing and/or actuating capabilities and exposes a set of features to its friends. *It's evening, there is no one in the apartment H₁ and the AS detects an intrusion in the house.* Immediately the AS writes a new post on its wall representing what it has sensed as an OWL annotation. A possible formalization of the post (reported in OWL2 Manchester syntax [20] for the sake of readability) w.r.t. the reference ontology follows:

```
AS_Request ≡ (detectsOutdoorLuminosity some)
and (detectsOutdoorLuminosity only
LowLuminosity) and (detectsIntrusion some)
and (detectsIntrusion only Intrusion)
and (detectsOccupancy some) and
(detectsOccupancy only (not Presence))
```

The AS starts a *Concept Covering* process using the post content as request, whereas available resources are represented by the functionalities exposed by all the devices directly involved into a friendship relation. The AS verifies if the services of the connected objects, SC₁ and L₁ in our example, were modified and (only in this case) retrieves updated annotations, listed in Figure 9 and 10 respectively. According to the semantic service descriptions, the matchmaking procedure highlights as the shutter should be *fully closed* and the dimmer lamp *turned on* to completely satisfy the request. This is due to the fact that the *Intrusion* concept was defined as more specific than *IntrusionForLamp* and *IntrusionForShutter* in the reference ontology, as shown in Figure 11. With a modestly expressive DL like \mathcal{ALN} , such a modeling pattern allows to activate, during the covering process, functionalities of different devices that are fired when the same event is detected. Therefore, the AS writes on its wall a comment to the post, containing only a tag for each service

Example 1. Read the profile of a social device

```
[REQ] GET coap://192.168.2.16:5683/profile      Accept: text/turtle
[RES] 2.05 Content      Content-Format (ct): text/turtle      ETag: W/'1234'
      <coap://192.168.2.16:5683/profile> a ldp:RDFSsource, m3-lite:AirConditioner ;
      dcterms:requires <http://sisinflab.poliba.it/swot/swst>, <http://purl.oclc.org/NET/UNIS/fiware/iot-lite> ;
      dcterms:title "Air Conditioner LR" ; iot-lite:relativeLocation swst:LivingRoom ;
      iot-lite:endpoint <coap://192.168.2.16:5683> ; swst:clientEndpoint <coap://192.168.2.16:61655> ;
      sor:friendOf <coap://192.168.2.25:5683> ; sioc:follows <coap://192.168.2.2:5683> .
```

Example 2. Send a friendship request

```
[REQ] POST coap://192.168.2.32:5683/friendship?title=AC      Accept: text/turtle
      ...payload (RDF device profile)...
[RES] 2.01 Created      Location-Path: coap://192.168.2.32:5683/friendship/AC
```

Example 3. Read the wall of a social device

```
[REQ] GET coap://192.168.2.16:5683/wall      Accept: text/plain
[RES] 2.05 Content      Content-Format (ct): text/turtle      ETag: W/'4567'
      <coap://192.168.2.16:5683/wall> a ldp:BasicContainer ; dcterms:title "Device Wall" ;
      ldp:contains <coap://192.168.2.16:5683/wall/P0>, <coap://192.168.2.16:5683/wall/P1> .
```

Example 4. Read a post on the wall

```
[REQ] GET coap://192.168.2.16:5683/wall/P0      Accept: text/turtle
[RES] 2.05 Content      Content-Format (ct): text/turtle      ETag: W/'a235'
      <coap://192.168.2.16:5683/wall/0> a ldp:BasicContainer, sioc:Post ;
      dcterms:created "2017-02-09T16:02:56.993+01:00"^^xsd:dateTime ; sioc:content swst:WS_Request ;
      sioc:has_creator <192.168.2.15:5683> ; swst:likeValue "87,15"^^xsd:double ;
      ldp:contains <coap://192.168.2.16:5683/wall/P0/C0>, <coap://192.168.2.16:5683/wall/P0/C1> .
```

Example 5. Write a post on the friend wall

```
[REQ] POST coap://192.168.2.16:5683/wall
      ...payload (OWL annotation)...
[RES] 2.01 Created      Location-Path: coap://192.168.2.16:5683/wall/P2
```

Example 6. Tag a device functionality on a comment

```
[REQ] PUT coap://192.168.2.2:5683/wall/P0/C1?ldp=patch
      If-Match: W/"a872"      Content-Format (ct): application/rdf-patch
      A <coap://192.168.2.2:5683/wall/P0/C1> sioc:topic <coap://192.168.2.16:5683/services/AirCondition_Cooling> .
[RES] 2.04 Changed
```

Example 7. Read the RDF-based description of a device functionality

```
[REQ] GET coap://192.168.2.16:5683/services/AirCondition_Cooling      Accept: text/turtle
[RES] 2.05 Content      Content-Format (ct): text/turtle      ETag: W/'bd72'
      <coap://192.168.2.16:5683/services/AirCondition_Cooling> a ldp:RDFSsource, iot-lite:Service ;
      dcterms:title "Air Condition Cooling" ; dcterms:modified "2017-02-09T16:02:48.698+01:00"^^xsd:dateTime ;
      iot-lite:interfaceDescription swst:AC_Cooling ; swst:activationValue "10"^^xsd:integer ; swst:hasState "off" .
```

Example 8. Read the OWL annotation of a device functionality

```
[REQ] GET coap://192.168.2.16:5683/services/AirCondition_Cooling/owl
[RES] 2.05 Content      Content-Format (ct): text/plain      ETag: W/'bd72'
      ...payload (OWL annotation)...
```

Example 9. Read LDP-CoAP headers to check (possible) modifications in the description of a device functionality

```
[REQ] GET coap://192.168.2.16:5683/services/AirCondition_Cooling?ldp=head
[RES] 2.03 Valid      Content-Format (ct): text/turtle      ETag: W/'bd72'
```

Example 10. Activate/deactivate a functionality

```
[REQ] PUT coap://192.168.2.16:5683/services/AirCondition_Cooling?ldp=patch
      If-Match: W/"bd72"      Content-Format (ct): application/rdf-patch
      D <coap://192.168.2.16:5683/services/AirCondition_Cooling> swst:hasState "off" .
      A <coap://192.168.2.16:5683/services/AirCondition_Cooling> swst:hasState "on" .
[RES] 2.04 Changed
```

Fig. 7. Examples of basic device interactions over LDP-CoAP

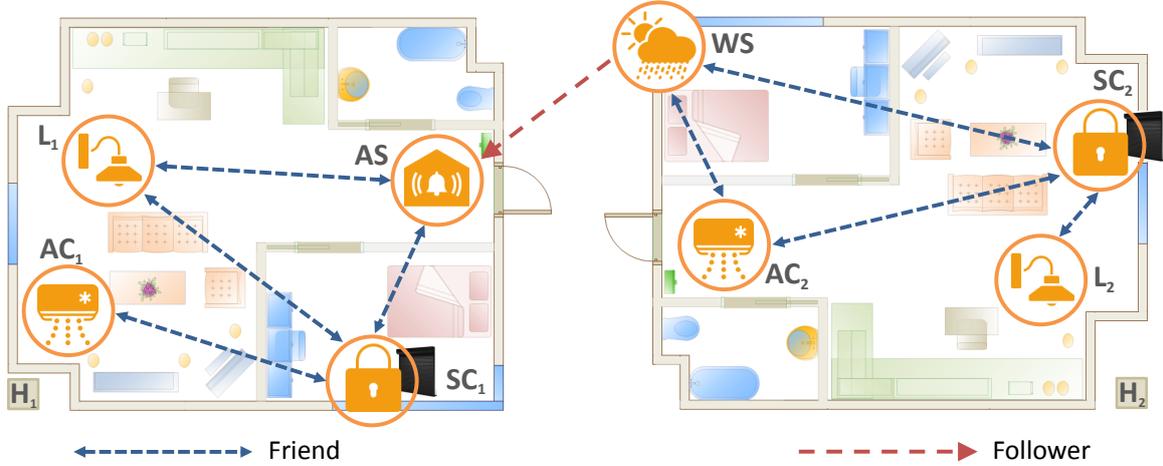


Fig. 8. Case study scenario

to activate, *i.e.*, *Full_Close* (SC_1) and *Lamp_On* (L_1). The uncovered part of the request is empty because the request was completely satisfied. Finally, according to the covering results, the like value of the post is automatically updated to 1 and no further operations are required.

Full_Close \equiv (detectsPrecipitation **some**) and (detectsPrecipitation **only** Rain) and (detectsWindSpeed **some**) and (detectsWindSpeed **only** StrongWind) and (detectsIntrusion **some**) and (detectsIntrusion **only** IntrusionForShutter) and (detectsOccupancy **some**) and (detectsOccupancy **not** Presence)

Half_Close \equiv (detectsPrecipitation **some**) and (detectsPrecipitation **only** (not Rain)) and (detectsWindSpeed **some**) and (detectsWindSpeed **only** ModerateWind)

Open \equiv (detectsPrecipitation **some**) and (detectsPrecipitation **only** (not Rain)) and (detectsWindSpeed **some**) and (detectsWindSpeed **only** LightBreeze) and (detectsOutdoorLuminosity **some**) and (detectsOutdoorLuminosity **only** HighLuminosity)

Fig. 9. Shutter controllers SC_1 and SC_2 service annotations

Simultaneously, devices within the apartment H_2 could exploit the knowledge shared by the home social network in H_1 to adapt their configuration according to the detected conditions. As shown in Figure 8, a *follower* relation exists between the weather station and the alarm system. As explained in Section 3.1, the WS follows (*i.e.*, continuously observes) the wall of the AS, so when the alarm system posts the intrusion

Lamp_On \equiv (detectsOutdoorLuminosity **some**) and (detectsOutdoorLuminosity **only** LowLuminosity) and (detectsIntrusion **some**) and (detectsIntrusion **only** IntrusionForLamp)

Lamp_Medium \equiv (detectsOutdoorLuminosity **some**) and (detectsOutdoorLuminosity **only** MediumLuminosity) and (detectsOccupancy **some**) and (detectsOccupancy **only** Presence)

Lamp_Off \equiv (detectsOutdoorLuminosity **some**) and (detectsOutdoorLuminosity **only** HighLuminosity) and (detectsOccupancy **some**) and (detectsOccupancy **only** (not Presence))

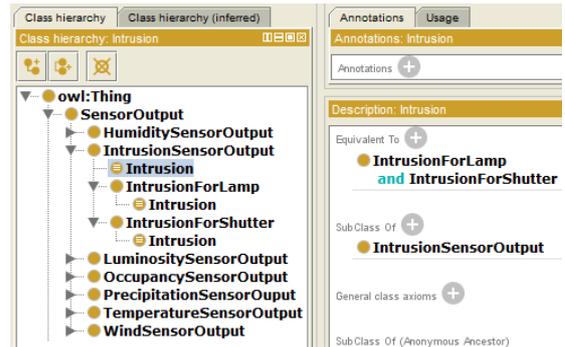
Fig. 10. Dimmer lamps L_1 and L_2 service annotations

Fig. 11. Intrusion ontology class modeling

message, it is immediately notified. The WS reads the annotation and shares it on its wall as a new post. This event triggers also in H_2 a *Concept Covering* process involving the services exposed by the friends of WS (SC_2 and AC_2), which are listed respectively in Figure 9 and Figure 12.

$AC_Cooling \equiv (\text{detectsTemperature } \mathbf{some}) \mathbf{and}$
 $(\text{detectsTemperature } \mathbf{only} \text{ HighTemperature}) \mathbf{and}$
 $(\text{detectsHumidity } \mathbf{some}) \mathbf{and} (\text{detectsHumidity}$
 $\mathbf{only} \text{ MediumHumidity})$

$AC_Heating \equiv (\text{detectsTemperature } \mathbf{some}) \mathbf{and}$
 $(\text{detectsTemperature } \mathbf{only} \text{ LowTemperature}) \mathbf{and}$
 $(\text{detectsHumidity } \mathbf{some}) \mathbf{and} (\text{detectsHumidity}$
 $\mathbf{only} \text{ LowHumidity})$

$AC_Dehumidification \equiv (\text{detectsTemperature } \mathbf{some})$
 $\mathbf{and} (\text{detectsTemperature } \mathbf{only}$
 $\text{MediumTemperature}) \mathbf{and} (\text{detectsHumidity } \mathbf{some})$
 $\mathbf{and} (\text{detectsHumidity } \mathbf{only} \text{ HighHumidity})$

Fig. 12. Air conditioners AC₁ and AC₂ service annotations

Only the *Full_Close* service, provided by SC₂, is selected to partially satisfy the request. WS comments its post including a tag to the shutter service and the uncovered part of the request as content. In this case, to further satisfy the post, the WS can forward the uncovered part to one of its friends. The WS selects the SC₂, since it provided the highest contribution to covering in the initial step, and posts on the wall of SC₂ the following OWL annotation of the uncovered part:

$Req_Uncovered \equiv (\text{detectsOutdoorLuminosity}$
 $\mathbf{only} \text{ LowLuminosity}) \mathbf{and} (\text{detectsIntrusion}$
 $\mathbf{only} \text{ IntrusionForLamp})$

Moreover, the WS starts to observe the post it just sent to the friend's wall. SC₂ in turn receives the message, starts a covering process involving the services exposed by L₂ (Figure 10) and selects the *Lamp_On* functionality that completely cover the remaining part of the initial request. SC₂ comments its post tagging the activated services and updates the like value with the percentage of covered features. The request is fully satisfied so the uncovered part is empty and no other posts are needed. Thanks to the observer pattern, the WS receives a notification about the post, reads the comment and understands that the initial request has been completely served. As a consequence, it updates the like value of the post on its wall, not forwarding further requests.

It is useful to point out how social capabilities allowed apartment H₂ to compensate for the lack of an alarm system, taking advantage of the sensing capabilities of the one in H₁ to appropriately configure and modify the status of its devices. This is just an obvious example of the benefits of the proposed semantic-based social framework in information and service/resource sharing in complex settings and heterogeneous networks. Furthermore, request and service descriptions in the case study were kept short for

easier understanding of the proposed framework, but the adopted inferences allow managing more detailed specifications with articulated constraints.

5. Evaluation

A prototypical testbed was developed following the proposed social framework described in Section 3. It implements a basic environment (similar to the one described in the above case study) consisting of a subset of home areas, *i.e.*, a hall door, a living room, a kitchen, and a small outdoor space. An IEEE 802.11 network was exploited as a fast backbone including 3 *smart* nodes, each implementing a single social device. Moreover, a KNX sub-network was connected to the main area by means of an additional smart node, acting as *gateway* toward the social network, allowing KNX-based devices to interact with the other social objects in a transparent way. As detailed in Table 2, KNX installation consisted of 8 off-the-shelf devices, produced by Gewiss Inc.³, connected through a twisted pair bus in a hierarchical network. Each device (except the KNX router, used only for the communication over IP) corresponds to one or more LDP-CoAP *endpoints*, exposed by the gateway node, representing the social objects in Table 2. In this way, all devices in the home can interact through the proposed LDP-CoAP interface independently from the specific HBA protocol.

Table 2
List of KNX devices

Product ID	Description	Social Device
GW90707	KNP/IP router	—
GW90740	Switch actuator 4 channels	Air conditioner (2 ch.) + Garden watering system (2 ch.)
GW90740	Switch actuator 4 channels	n.4 Simple on/off lamp
GW12782	Push button 4 channels	n.4 Basic on/off button
GW90800	Weather Station with GPS	Weather Station
GW90746	Dimmer actuator	Dimming lamp
GW10948	Burglar alarm system interface	Alarm system
GW90754	Roller shutter actuator	Shutter controller

According to Figure 13, a Java-based management software was implemented to run on each home device. The main Java package *it.poliba.sisinflab.swst*

³<http://www.gewiss.com>

was partitioned in the following sub-packages to separate developed classes in well-defined sections each providing the following specific functionality:

- *core*: contains the *HomeDevice* reference implementation. It extends the *CoAPLDP*Server provided by the LDP-CoAP library⁴ and exposes one or more *DeviceEndpoint* managing the resources described in Table 1. At network level, LDP-CoAP provides also a modified version of the *Californium* CoAP framework [25] supporting LDP features over CoAP;

- *resources*: several Java classes model the different device resources. LDP-CoAP *RDFSource*, *Non-RDFSource* and *BasicContainer* base classes were extended, providing common attributes and methods to save, retrieve and update the home data. All information is stored within an RDF repository based on the *RDF4J 2.1.3* library⁵;

- *rdf.vocabulary*: contains RDF ontology files mapped as Java classes to simplify creation and querying of RDF triples. Ontologies cited in Section 3.2 (e.g., SIOC, IoT-lite, M3-lite) were mapped through the *Sesame Vocabulary Builder*⁶ tool and included in the package;

- *owl*: provides basic functionalities to load the reference KB, to manage all generated OWL annotations through the *OWL-API 3.4.10* library [19] and to invoke the Mini-ME reasoner [37] implementing inference services;

- *knx*: an additional package implemented to support the communication over the *ISO/IEC 14543-3 EIB/KNX* protocol stack [34]. *Calimero-core* library⁷ was exploited for network management and to exchange data with KNX devices (e.g., read state values or send commands). An import utility was also implemented to parse data from an XML-based project file exported from ETS4⁸, the official software tool used to design and configure home installations based on KNX systems, and to model the same device features within the home social network.

Performance evaluation of the proposed approach was carried out developing each smart node on three reference platforms with different processing capabilities. In particular, the following embedded boards were used to implement the social devices:

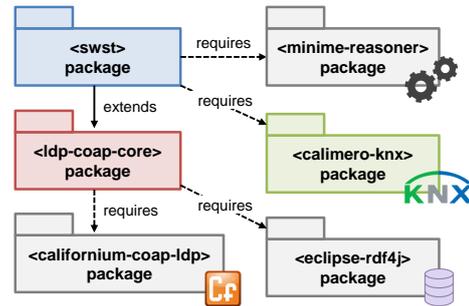


Fig. 13. Reference software modules

(a) *Raspberry Pi Model B*⁹, equipped with a single-core ARM11 CPU at 700 MHz, 512 MB RAM (shared with GPU), 8 GB storage memory on SD card, Raspbian Wheezy OS;

(b) *Intel Edison Kit*¹⁰ equipped with an Intel Quark x86 CPU at 400 MHz, 1 GB RAM, 4 GB eMMC flash storage and Yocto Poky Linux OS (32-bit kernel 3.10.98);

(c) *UDOO Quad*¹¹ equipped with quad-core ARM Cortex A9 at 1 GHz clock frequency, ARM Cortex M3 coprocessor, 1 GB DDR3 RAM, 32 GB storage memory on SD card, UDOObuntu 2.0 Minimal Edition OS. All platforms included a 32-bit Java 8 SE Runtime Environment (JRE, build 1.8.0-b121).

Experiments have been carried out performing the 10 reference tasks described in Figure 7, to identify and evaluate specific features characterizing their performance. Each test was repeated five times and average values were taken and reported in Figure 14. In particular, the *processing* time is defined as the time elapsed on the device receiving a request to process the message and send the related response, whereas the *communication* time represents the time needed to exchange request/response data (i.e., CoAP packets) over the home network between the sender and the receiver device. As expected, RaspberryPi required a longer time to process the requests due to the reduced computational capabilities. On the contrary, Intel Edison was the fastest platform in case of tasks only requiring simple I/O operations, thanks to the internal flash memory.

Concerning communication time, it should notice a significant variation, due to the different hardware adopted for connecting to the home network. In particular, RaspberryPi and UDOO were equipped with a

⁴<https://github.com/sisinflab-swot/ldp-coap-framework>

⁵<http://rdf4j.org/>

⁶<http://github.com/tkurz/sesame-vocab-builder>

⁷<http://github.com/calimero-project>

⁸<http://www.knx.org/knx-en/software/ets/about/>

⁹<http://www.raspberrypi.org/products/model-b/>

¹⁰<http://www.intel.com/content/www/us/en/do-it-yourself/edison.html>

¹¹<http://www.udoo.org/udoo-dual-and-quad>

Table 3
Request Dataset

ID	Size (Byte)	Tagged Services	Like Value	Processing Time (ms)		
				Rasp. Pi	Intel Edison	UDDO
R1	1381	1	28.57	89.2	20.8	12.8
R2	1967	2	61.54	122.2	30.8	14.2
R3	2468	3	73.68	132.0	37.6	17.8
R4	2902	4	79.17	140.2	47.0	18.0
R5	3309	5	82.76	146.4	64.4	23.2
R6	3651	6	84.85	154.6	78.8	29.8
R7	4202	7	87.18	160.8	81.4	35.4

Wi-Fi IEEE 802.11 USB Dongle, whereas Intel Edison exploited the on-board Wi-Fi transceiver.

Test results about the Concept Covering task performed on a single node are reported in Figure 15. Experiments were conducted exploiting a shared dataset of 7 requests (see Table 3) with growing size and restriction complexity. As a consequence, a different number of services was selected (from a set of 50 instances) and tagged after the reasoning step. It can be pointed out that like value was lower for simpler requests and increased for more complex ones. This is due to the fact that ontology and service annotations are modeled to fit articulate and specific descriptions, such as device operating requirements; generic requests result less significant, leading to a relative loss of the semantic-based score. For all platforms, the complexity of the request affected time only slightly, showing a similar trend. As expected, the processing time was longer for complex annotations, because a higher number of services was retrieved to satisfy the request. Moreover, memory usage values are shown in Figure 16. Framework requirements were low on all platforms, with a memory peak always under 18.5 MB for stack memory and 16.5 MB for heap memory, representing reasonable values for embedded systems.

Another relevant parameter of the social framework performance is the amount of data exchanged over the home network. In order to reduce the number of packets used to transmit each message over CoAP, the LDP-CoAP implementation described in [28] was extended to support the following encoding algorithms, aiming to reduce the size of resource descriptions: (i) *GZIP* and *BZIP2* (both included within the *Apache Commons Compress* library¹²), general-purpose and suitable for annotations described with RDF Turtle [11], JSON-LD [26] and all OWL syntaxes; (ii) *Binary JSON* (BSON)¹³, *Universal Binary JSON* (UBJSON)¹⁴ and *Message Pack* (MsgPack)¹⁵, specific for JSON-based annotations.

Selected algorithms were tested on three basic resources corresponding to the LDP-CoAP resource types available in the proposed framework: RDF Source (LDP-RS, e.g., device profiles and comments); Basic Container (LDP-BC, e.g., walls and posts); Non-RDF Source (LDP-NR, e.g., OWL annotations of posts, comments and services). LDP-RS and LDP-BC were described with RDF Turtle and JSON-LD to test both syntaxes supported by LDP-CoAP, whereas LDP-NR was described through the OWL 2 Manchester syntax. Figure 17 reports on the size of the reference annotations with and without compression. GZIP provided better results, achieving a compression ratio of about 48%. In this way, each device sends on average the half of CoAP packets (which must contain a maximum of 64B as payload), so reducing the overall communication latency. On the contrary, JSON-specific algorithms were not particularly useful for short messages, being designed to encode large documents.

Finally, benefits of the devised semantic social platform were assessed in a comparison w.r.t. the following IoT-oriented frameworks in the HBA market: KNX IoT¹⁶; IzoT Platform¹⁷, originally developed by Echelon Corporation for the Industrial IoT but also exploited for building applications; Dog Gateway¹⁸ [7]; Eclipse SmartHome¹⁹. Table 4 highlights as only the proposed approach combines fitness for resource-constrained environments (by using CoAP and a P2P architecture), expressiveness of device modeling (by exploiting RDF and OWL2) and support for both exact and approximated matches, with formally grounded service composition.

6. Conclusion and Future Work

This paper introduced a novel semantic-based framework for Social Internet of Things, particularly useful for home and building automation but inherently general-purpose. The proposal adopted a decentralized service-oriented architecture to manage, pub-

¹²<http://commons.apache.org/proper/commons-compress/>

¹³<http://bsonspec.org/>

¹⁴<http://ubjson.org/>

¹⁵<http://msgpack.org/>

¹⁶<http://www.knx.org/knx-en/Landing-Pages/KNX-IoT>

¹⁷<http://www.echelon.com/izot-platform>

¹⁸<http://dog-gateway.github.io/>

¹⁹<http://www.eclipse.org/smarthome/index.html>

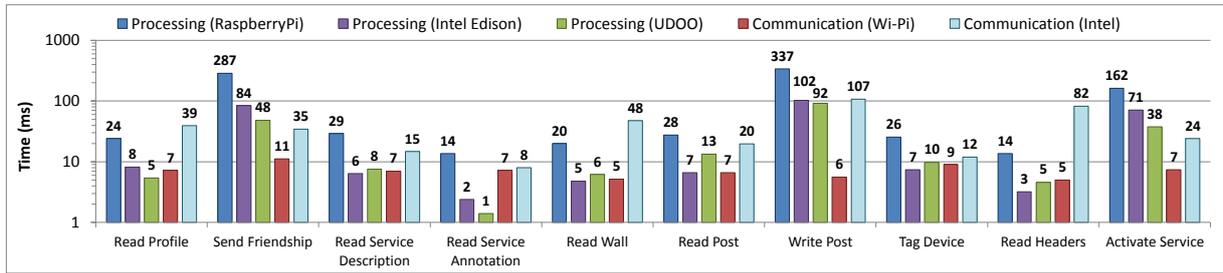


Fig. 14. Processing and communication time for the social tasks

Table 4
Comparison with current IoT-oriented frameworks for HBA

Features	Proposed Approach	KNX IoT	IzoT Platform	Dog Gateway	Eclipse SmartHome
Home Area Network reference protocol	multi-protocol over CoAP	EIB/KNX	LonTalk	multi-protocol over HTTP	multi-protocol over HTTP
Network architecture	full P2P	centralized	centralized	centralized	centralized
Network/devices configuration	autonomous social agents configuration	via ETS software	via LonBuilder software or XML configuration files	XML configuration files	Domain Specific Language (DSL) configuration files
Add/remove devices	agents self-configuration	edit network/devices configuration	edit network/devices configuration	edit network/devices configuration	edit network/devices configuration
Multi-protocol communication	smart node acting as gateway	KNX gateway	IzoT gateway	Dog gateway	node acting as gateway
Device binding	dynamic, based on friendship relationships	static, defined during network configuration	static, defined during network configuration	dynamic, based on device profile	static, defined during device configuration
Scenarios configuration	dynamic, exploiting non-standard inferences	static, defined during network configuration	static, defined during network configuration	dynamic, exploiting rule-based reasoning	static, based on an ECA rule engine
Service composition	yes, through distributed covering	no	no	no	no
Message data format	RDF Turtle, JSON-LD, OWL 2	proprietary, according to KNX spec.	proprietary, XML-based according to LonTalk spec.	OWL 2	XML
Standardised framework interface	LDP-CoAP RESTful interface	KNX IoT Web Services	HTTP RESTful API	WebSocket and HTTP RESTful API	HTTP RESTful API

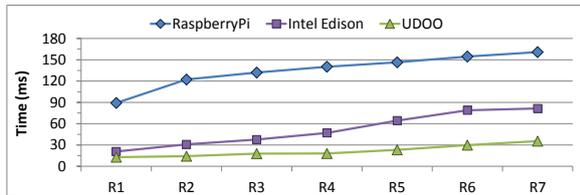


Fig. 15. Processing time for the Concept Covering task

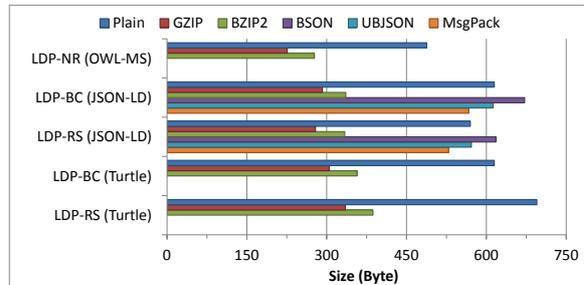


Fig. 17. Size of encoded payload messages

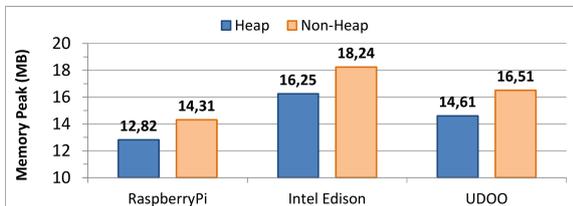


Fig. 16. Memory usage

lish, discover and compose semantically annotated service/resource descriptions. It adopted LDP-CoAP to join the benefits of efficient RESTful machine-

to-machine communication and structured Linked Data organization. Non-standard, non-monotonic inferences enabled semantic matchmaking for service/resource discovery with support for approximate matches, logic-based ranking and composition via request covering. The framework was developed on a multi-protocol HBA testbed with single-board computers and embedded home devices, exhibiting effectiveness in AmI scenarios.

Future work includes a wider testbed implementation and experimentation, to validate scalability of the proposal in very large object networks. Moreover, heuristics governing decisions about the creation and removal of friend/follower relationships will be explored, including behaviors based on agents' past experience, possibly by means of machine learning techniques. A similar approach can be adopted to endow social objects with proactive adaptivity to environmental modifications. Finally, further object interaction schemes will be investigated according to the *Linked Data Notifications* protocol [10].

References

- [1] R. Agarwal, D. G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas, and V. Issarny. Unified IoT Ontology to Enable Interoperability and Federation of Testbeds. In *3rd IEEE World Forum on Internet of Things*, 2016.
- [2] L. Atzori, A. Iera, and G. Morabito. From "Smart Objects" to "Social Objects": The next evolutionary step of the Internet of Things. *Communications Magazine, IEEE*, 52(1):97–105, 2014.
- [3] M. Bermudez-Edo, T. Elsaleh, P. Barnaghi, and K. Taylor. IoT-Lite: A Lightweight Semantic Model for the Internet of Things. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, pages 90–97. IEEE, 2016.
- [4] G. Biamino. A semantic model for socially aware objects. *Advances in Internet of Things*, 2(3):47–55, 2012.
- [5] G. Bizios. *Architecture Reading Lists and Course Outlines*. Eno River Press, 1998.
- [6] U. Bojars, J. G. Breslin, and S. Decker. Porting social media contributions with SIOC. In *Recent Trends and Developments in Social Software*, pages 116–122. Springer, 2010.
- [7] D. Bonino, E. Castellina, and F. Corno. The DOG gateway: enabling ontology-based intelligent domotic environments. *IEEE Transactions on Consumer Electronics*, 54(4):1656–1664, 2008.
- [8] C. Bormann, A. Castellani, and Z. Shelby. CoAP: An Application Protocol for Billions of Tiny Internet Nodes. *IEEE Internet Computing*, 16(2):62–67, 2012.
- [9] E. Bove. Object (b)logging: semantically rich context mining and annotation in pervasive environments. In *Advances in Sensors and Interfaces (IWASI), 2015 6th IEEE International Workshop on*, pages 210–215. IEEE, 2015.
- [10] S. Capadisli, A. Guy, C. Lange, S. Auer, A. Samba, and T. Berners-Lee. Linked Data Notifications: a resource-centric communication protocol. In *14th Extended Semantic Web Conference (ESWC2017)*, 2017. to appear.
- [11] G. Carothers and E. Prud'hommeaux. RDF 1.1 Turtle (Terse RDF Triple Language). W3C Recommendation, W3C, Feb. 2014. <http://www.w3.org/TR/turtle/>.
- [12] M. Compton, P. Barnaghi, L. Bermudez, R. Garcia-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al. The SSN Ontology of the W3C Semantic Sensor Network Incubator Group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17, 2012.
- [13] D. N. Crowley, E. Curry, and J. G. Breslin. Leveraging Social Media and IoT to Bootstrap Smart Environments. In *Big Data and Internet of Things: A Roadmap for Smart Environments*, pages 379–399. Springer, 2014.
- [14] H. Dibowski and K. Kabitzsch. Ontology-based device descriptions and device repository for building automation devices. *EURASIP Journal on Embedded Systems*, 2011(1):623461, 2011.
- [15] Dublin Core Metadata Initiative and others. Dublin core metadata element set, version 1.1, 2012.
- [16] Y. Evchina, J. Puttonen, A. Dvoryanchikova, and J. L. M. Lastra. Context-aware knowledge-based middleware for selective information delivery in data-intensive monitoring systems. *Engineering Applications of Artificial Intelligence*, 43:111–126, 2015.
- [17] S. N. Han, G. M. Lee, and N. Crespi. Semantic context-aware service composition for building automation system. *IEEE Transactions on Industrial Informatics*, 10(1):752–761, 2014.
- [18] K. Hartke. Observing Resources in the Constrained Application Protocol (CoAP). RFC 7641 (Proposed Standard), Sept. 2015.
- [19] M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web*, 2(1):11–21, 2011.
- [20] M. Horridge and P. F. Patel-Schneider. OWL 2 Web Ontology Language Manchester Syntax (Second Edition). W3C Working Group Note, W3C, Dec. 2012. <https://www.w3.org/TR/owl2-manchester-syntax/>.
- [21] D. Hussein, S. N. Han, G. M. Lee, N. Crespi, and E. Bertin. Towards a dynamic discovery of smart services in the social internet of things. *Computers & Electrical Engineering*, 2016.
- [22] B. Jadhav and S. Patil. Wireless Home Monitoring using Social Internet of Things (SIoT). In *Automatic Control and Dynamic Optimization Techniques (ICADOT), International Conference on*, pages 925–929. IEEE, 2016.
- [23] W. Kastner, M. Jung, and L. Krammer. Future Trends in Smart Homes and Buildings. In R. Zurawski, editor, *Industrial Communication Technology Handbook*, chapter 59, pages 1483–1502. CRC Press/Taylor & Francis, 2nd edition, 2014.
- [24] W. Kastner, L. Krammer, and A. Fernbach. State of the Art in Smart Homes and Buildings. In R. Zurawski, editor, *Industrial Communication Technology Handbook*, chapter 55, pages 1415–1434. CRC Press/Taylor & Francis, 2nd edition, 2014.
- [25] M. Kovatsch, M. Lanter, and Z. Shelby. Californium: Scalable cloud services for the Internet of Things with CoAP. In *Internet of Things, 2014 Int. Conf. on the*, pages 1–6. IEEE, 2014.
- [26] M. Lanthaler, M. Sporny, and G. Kellogg. JSON-LD 1.0. W3C Recommendation, W3C, Jan. 2014. <http://www.w3.org/TR/json-ld/>.
- [27] K.-Y. Lin and H.-P. 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.
- [28] G. Loseto, S. Ieva, F. Gramegna, M. Ruta, F. Scioscia, and E. 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 International Semantic Web Conference, Proceedings, Part II*, volume 9982 of *Lecture Notes in Computer Science*, pages 131–139,

- Cham, oct 2016. Springer International Publishing.
- [29] E. Papadopoulou, S. Gallacher, N. K. Taylor, M. H. Williams, F. R. Blackmun, I. S. Ibrahim, M. Y. Lim, I. Mimitsoudis, P. Skillen, and S. Whyte. Combining pervasive computing with social networking for a student environment. In *Proceedings of the Twelfth Australasian Symposium on Parallel and Distributed Computing-Volume 152*, pages 11–19. Australian Computer Society, Inc., 2014.
- [30] B. Parsia, S. Rudolph, M. Krötzsch, P. Patel-Schneider, and P. Hitzler. OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, W3C, Dec. 2012. <http://www.w3.org/TR/owl2-primer>.
- [31] B. Ramis Ferrer, S. Iarovy, L. Gonzalez, A. Lobov, and J. L. Martinez Lastra. Management of distributed knowledge encapsulated in embedded devices. *International Journal of Production Research*, pages 1–18, 2015.
- [32] M. Ruta, E. Di Sciascio, and F. Scioscia. Concept abduction and contraction in semantic-based P2P environments. *Web Intelligence and Agent Systems*, 9(3):179–207, 2011.
- [33] M. Ruta, F. Scioscia, and E. Di Sciascio. Enabling the Semantic Web of Things: Framework and Architecture. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, pages 345–347. IEEE, 2012.
- [34] M. Ruta, F. Scioscia, G. Loseto, and E. Di Sciascio. KNX, a worldwide standard protocol for home and building automation: state of the art and perspectives. In R. Zurawski, editor, *Industrial Communication Technology Handbook*, chapter 58, pages 1463–1481. CRC Press/Taylor & Francis, 2nd edition, aug 2014.
- [35] M. Ruta, F. Scioscia, G. Loseto, and E. Di Sciascio. Semantic-based resource discovery and orchestration in home and building automation: a multi-agent approach. *IEEE Transactions on Industrial Informatics*, 10(1):730–741, 2014.
- [36] F. Sadri. Ambient intelligence: A survey. *ACM Computing Surveys (CSUR)*, 43(4):36, 2011.
- [37] F. Scioscia, M. Ruta, G. Loseto, F. Gramegna, S. Ieva, A. Pinto, and E. Di Sciascio. A mobile matchmaker for the Ubiquitous Semantic Web. *International Journal on Semantic Web and Information Systems*, 10(4):77–100, dec 2014.
- [38] N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE intelligent systems*, 21(3):96–101, 2006.
- [39] Z. Shelby. Constrained RESTful Environments (CoRE) Link Format. RFC 6690 (Proposed Standard), Aug. 2012.
- [40] S. Speicher, J. Arwe, and A. Malhotra. Linked Data Platform 1.0. W3C Recommendation, W3C, Feb. 2015. <https://www.w3.org/TR/ldp/>.
- [41] O. Voutyras, P. Bourellos, S. Gogouvitis, D. Kyriazis, and T. 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.