RFID-enhanced Ubiquitous Knowledge Bases: framework and approach

Michele Ruta, Eugenio Di Sciascio, Floriano Scioscia

Politecnico di Bari, Via Re David 200, Bari, BA I-70125, ITALY

Phone/Fax: +39 080 596 3515

Email: {m.ruta,disciascio,f.scioscia}@poliba.it

Abstract As opposed to classical paradigms, in pervasive computing the user simultaneously interacts with several micro-devices such as RFID tags "integrated" into the environment, extracting data from them. Current approaches based on centralized control and information storage are utterly impractical. Here a ubiquitous Knowledge Base (u-KB) is defined as a distributed knowledge base whose individual facts are disseminated on RFID-tagged objects within the environment, without centralized coordination. The proposed framework includes: (i) components and operations specification of a u-KB, (ii) a distributed application-layer protocol for dissemination and discovery of knowledge embedded within RFIDs dipped in a MANET and (iii) a homomorphic algorithm for compressing on-tag data. The feasibility of the approach has been confirmed by a large-scale simulation campaign.

1 Introduction

In pervasive computing both information and computation are embedded into a given environment, that is into everyday objects and/or actions. The user simultaneously interacts with many micro-devices during ordinary activities, even not necessarily being aware of interaction details. Due to the volatility and unpredictability of such scenarios, centralized and fixed control is practically unfeasible.

Considering Radio Frequency IDentification (RFID) technology, nowadays tags with higher memory availability disclose the concrete possibility of building cooperative environments where autonomous objects can be discovered, interrogated and inventoried without any backend infrastructure. A proper dissemination protocol can allow to exactly locate suitable descriptions directly on tags attached to objects thanks to a capillary diffusion of tag/reader basic information within the environment. Such a vision allows to build an environment where tagged objects and readers make a self-contained retrieval architecture where external data links are not compulsory. The approach aims at the so-called Internet of Things and goes beyond common infrastructure-based RFID applications, because it allows to users simply equipped with a personal device to perform an advanced object discovery.

We define a *ubiquitous Knowledge Base* (*u-KB*) as a distributed knowledge base whose individuals (assertional knowledge) are disseminated on the objects within a given context, lacking the need for centralized coordination. The framework presented here specify both elements and operations of a u-KB, as well as a distributed protocol for dissemination and discovery of knowledge embedded within RFID tags dipped in a Mobile Ad-hoc Network (MANET). RFID readers are considered as cluster-heads with respect to tags in radio visibility. They are also able to automatically build up multi-hop inter-reader wireless communication when placed in the same area (Ramanathan and Redi, 2002) exploiting IEEE 802.11 (IEEE 802.11 1999), along with IP and UDP, for network infrastructure.

As formalism for resource annotation we use ontology languages based on Description Logics (DL) and originally conceived for the Semantic Web effort, particularly DIG (Description Logics Implementation Group) (Bechhofer et al. 2003),

which is a more compact equivalent of OWL-DL Web Ontology Language (McGuinness and van Harmelen 2004). These languages were defined to provide a standard for semantically rich, formal and unambiguous description of Web resources through annotated metadata. According to the Internet of Things vision, we aim to adopt them to describe objects and phenomena of the physical world, linking them to the digital world through semantic-enhanced RFID. From this standpoint, a relevant technical aspect is information compression, because XML-based formats adopted in the Semantic Web (such as RDF, OWL and DIG) are too verbose to allow efficient data storage and management in mobility. Compression techniques become essential in order to enable memorization and transmission of semantically annotated information on tiny mobile devices such as RFID tags or wireless sensors. Moreover, 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 energy consumption for handhelds.

The main contributions of the chapter are:

- to give an innovative KB (knowledge base) framework for pervasive, infrastructure-less environments preserving the distinction between factual and terminological knowledge and also adhering to the classical Tell/Ask paradigm;
- to propose some slight (backward compatible) modifications to current state-of-the-art RFID protocol in order to enable semantic enhancements;
- to integrate a compression algorithm completing the framework and able to store within a tag a semantically rich description of the object the RFID is attached to (using less than 2 kB of tag memory);
- to provide a case study clarifying and motivating the approach, possibly evidencing benefits it offers.

The remaining of the chapter is organized as follows: in the next Section motivation and possible application scenarios are presented. In Section 2 a survey on related work is provided. In Section 3 significant theoretical aspects of the framework are described. The reference architecture and protocol are outlined in Sections 4 and 5 respectively. The case study in Section 6 clarifies framework implementation in detail (including compression issues). Experimental methods and results are presented and commented upon in Section 7. Finally, conclusion closes the paper.

2 Motivation

In current applications (see (Baader at al., 2002) §1.5 for a survey), Knowledge Representation Systems (KRSs) play a role which is similar to Database Manage-

ment Systems. Both are used as central repositories where explicit domain knowledge is inserted with the aim of extracting the implicit one by means of inference procedures. Hence, in traditional KRSs, a KB is seen as a fixed entity immediately available, either in local storage or via a high-throughput network link. This approach is effective only as long as large computing resources and a dependable infrastructure are granted.

A different setting is needed to adapt KR tools and technologies to mobile and ubiquitous computing applications. They are characterized by:

- user and device *mobility*, which makes connections volatile and the (un)availability of resources unpredictable;
- dependency on *context*, which means that applications must adapt the way they support user tasks not only according to the availability of both nearby resources, but also to information characterizing environment and user;
- severe *resource limitations* of mobile devices affecting processing, storage, link bandwidth and power consumption.

Hence, knowledge-based systems designed for wired networks are not simply adaptable to wireless ones, due to intrinsic differences and performance issues.

The goal of a pervasive knowledge-based system is to embed semantically rich and easily accessible information into the physical world. This requires sharable vocabularies, otherwise the disagreement about the meaning of descriptions would impair system interactions (Vasudevan 2004). Logic-based languages for the Semantic Web endowed with formal semantics can provide such levels of interoperability. Advantages of semantic-aware approaches in RFID applications are widely acknowledged (Weinstein 2005). Basic services the technology offers could be significantly improved by introducing a semantically rich object description and discovery capabilities.

Let us consider the products lifecycle: manufacturing and quality control can exploit accurate descriptions of raw materials, components and processes; supply chain management could benefit from improved item tracking; verification of multi-factor service level agreements between commercial partners can be automated. Furthermore, sale depots could be easily inventoried also providing ubiquitous commerce (u-commerce) (Watson et al. 2002) capabilities – like those introduced in (Venkataramani and Iyer, 2007) and (Ruta et al. 2006) – without expensive investments in infrastructure. Finally, smart post-sale services can be provided to purchasers, by integrating knowledge discovery capabilities into home and office appliances (Ruta et al. 2007). In addition, asset management is greatly improved in those scenarios where retrieval should be based on relevant object properties and purposes, rather than mere identification codes.

In healthcare applications, equipment, drugs and patients can be thoroughly and formally described and tracked, not only to ensure that appropriate treatments are given, but also to provide decision support in therapy assignment. The pervasive-

ness of infrastructure helps to reduce costs and break barriers between patient management in the hospital and at home.

Likewise, in museums, libraries and archaeological sites, semantic-based content fruition can be granted either to local visitors or remote users connected through the Internet. In both cases the internal RFID infrastructure will be leveraged.

A knowledge-based approach can be integrated with other monitoring and sensing technologies beyond RFID. Wireless semantic sensor networks (Ni et al. 2005) are an emerging yet challenging technology. Semantic-based sensory data dissemination and query processing are needed to enable advanced solutions for *e.g.* environmental monitoring, precision agriculture, green supply chains and disaster recovery.

3 Related work

Pervasive computing requires a decentralized and collaborative coordination between autonomous mobile hosts. Therefore, pervasive knowledge-based systems have to achieve high degrees of autonomic capability, providing transparent access to knowledge sources that may be present in a given area. This is achieved by exploiting a relatively large number of heterogeneous micro devices – for example RFID tags or wireless sensors – each conveying a small amount of useful information.

Middleware infrastructures for ubiquitous computing which can be found in literature usually rely on centralized nodes for management and discovery of information (Chakraborty et al. 2006, Pallapa and Das 2007, Vazquez and Lopez-de-Ipina 2007, Toninelli et al. 2008). Particularly, RFID technology currently links physical objects with their “virtual counterpart” (Römer et al., 2004) in the digital world. Tags trivially store an identification code, which is used to retrieve object properties from an information server, through a networked infrastructure.

The biggest obstacle toward decentralized approaches is seen in the too high cost of RFID tags with sufficient memory. Notwithstanding, the growing demand of RFID solutions allow to expect that passive RFID tags with higher memory capacity will be available at low cost in the next few years (Ayoub et al. 2009).

In (De et al. 2004) a pervasive architecture is presented for tracking mobile objects in real-time for supply chain and B2B transaction management. A global and persistent IT infrastructure is needed in order to interconnect RFID systems within partner organizations through the Internet. These requirements make the approach unsuitable for mobile B2C and C2C applications.

More recent research works (Venkataramani and Iyer, 2007) (Ruta et al., 2007) proposed approaches for the integration of semantic-enhanced EPCglobal RFID into MANETs. Semantic annotations could be put into RFIDs attached to objects so that tagged goods stored a semantically rich description featuring the product the tag is clung to. In this way, objects equipped with RFID tags described them-

selves toward the “rest of the world” in a self-contained fashion. Nevertheless, a fixed central component was still needed for reasoning over a Knowledge Base (KB). This led to expensive information duplication within the environment: semantic annotations were simultaneously placed on tags and within the KB; moreover, the reasoning engine was a single point of failure. Such issues can be mitigated if a more distributed approach is followed, as the one proposed here.

For what concerns compression, lossless algorithms can be categorized into three families: static, semi-adaptive, and adaptive (Howard and Vitter 1991). *Static* compression uses either fixed statistics or no statistics. *Semi-adaptive* compression works in two steps: the input data is scanned once for statistics gathering and again for encoding. In *adaptive* compression, instead, statistics are calculated and dynamically updated along with compression.

gzip is one of the most popular universal compression tool, based on a variant of the *LZ77* adaptive algorithm (Ziv and Lempel 1977).

Huffman encoding (Huffman 1952) and *arithmetic encoding* (Witten et al. 1987) are two fundamental semi-adaptive compression techniques. The former uses a variable-length code table (the *huffman tree*), derived from evaluating the frequency of each symbol. In the latter, instead, the whole message is represented by a real value between 0 and 1. In this class of algorithms and tools, the *PAQ* family (Mahoney 2005) currently has the best compression rates, but its computing and memory requirements are far beyond capabilities of mobile devices.

Higher compression rates can be achieved by algorithms specifically designed for XML encoding. *XMill* (Liefke and Suciu 2000) is an efficient schema-aware XML compressor. It splits XML content into different *containers*, which are separately compressed and sequentially stored in the output file (hence XMill is non-homomorphic). XMill works better than generic compressors for medium and large XML documents, while for small files (up to 20 kB approximately) it is penalized by the small size of containers. In *DPDT* (Harrusi et al. 2006) the XML document is encoded using *Partial Prediction Matching* (PPM), an adaptive technique. Reported compression rates are higher than XMill, though PPM-based compressors are generally slower.

XPRESS (Min et al. 2006) adopts *reverse arithmetic encoding*, a semi-adaptive algorithm. Furthermore, a *type inference* module detects data types of XML attribute values and a specialized encoding method is applied for each type (numbers, dates and so on). Experimental results showed XPRESS achieves high query performance for compressed XML data and updates can be directly performed on compressed XML. Compression rates, though, are lower than other methods.

4 Theoretical framework

Fig. 1 depicts the proposed framework for knowledge dissemination and discovery in a pervasive context. The u-KB layer provides common access to information embedded into semantic-enhanced EPCglobal (Traub et al. 2005) RFID tags popu-

lating a smart environment. UDP is used for data exchange in IEEE 802.11 mobile ad-hoc networks. Further identification and sensing systems can be interfaced to the general framework, through a semantic support micro-layer added to standard protocols. Applications can use knowledge provided by a u-KB, by means of DL-based reasoning services exploiting the semantic discovery protocol featuring the u-KB layer. They can be performed either by hosts in the local MANET or by a remote entity through a gateway exposing a high-level interface (*e.g.* Web Sevices of RPC[1] or REST[2] type) and translating remote methods into operations on the u-KB.

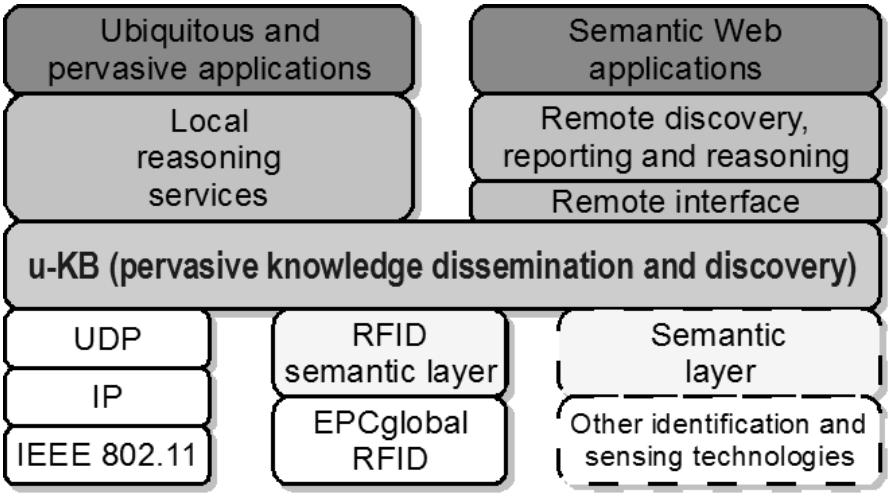


Fig. 1. Architecture of the proposed approach

This chapter focuses on definition and experimentation of the u-KB layer and the RFID semantic layer in Fig. 1. Parallel research effort is being spent into the adaptation of reasoning procedures to resource-constrained mobile devices.

4.1 KB components and operations

Given a reference problem in a specified domain, a DL *Knowledge Base* models it in two components (Baader et al. 2002):

- a **TBox** (terminological box), containing intensional knowledge, *i.e.* general knowledge about the domain. It has the form of an ontology – also called terminology – describing concepts and their relationships;

[1] Remote Procedure Call
[2] REpresentational State Transfer

- an **ABox** (assertion box), containing *extensional* knowledge, which is specific to the particular problem. It consists of an assertions set concerning the domain individuals.

It is useful to recall that intensional knowledge is usually thought not to change whereas extensional knowledge is usually contingent, or dependent on a set of circumstances, and therefore subject to change (Baader et al. 2002, p. 17).

Current KRSs are characterized in terms of what *functions* they provide to applications, instead of system data structures and allowed low-level operations (Levesque 1984). As a consequence, design separates functionality from implementation, leading to both better predictability of system behavior and greater ease of use and integration of a KRS within larger application solutions. Two basic functions were identified for KB management:

- **Tell:** build the TBox and the ABox by means of explicit terminological axioms and assertions about individuals;

- **Ask:** extract (implicit) knowledge. Queries are answered by the system through inference procedures, determining if the query meaning is implied by the information that has been told.

This paradigm has led to detailed and formal interface specifications for Knowledge Representation Systems – such as *KRSS* (Patel-Schneider and Swartout 1993) and, more recently, *DIG* (Bechhofer et al. 2003) – implemented by most KRSs.

The capability to selectively remove information from a KB is also desirable. Current systems allow to *Un-Tell* (*i.e.* retract) only information that has been previously told (see (Brachman et al. 1991) for a discussion). Experience has shown that, for the vast majority of applications, the TBox seldom or never changes after an initial knowledge acquisition phase (see (Baader et al. 2002, ch. 8) for a review).

4.2 u-KB components and operations

In our approach we preserve differences between TBox and ABox.

The **TBox** is expressed by means of an ontology document, which can be owned by one or more mobile hosts. For what reported above, it can be reasonably assumed that ontologies defined before object annotation and u-KB deployment seldom change during normal system activity. Nevertheless, in order to improve the system flexibility, the TBox management could be performed also following paradigms and approaches devised in peer-to-peer protocols such as BitTorrent (Cohen 2008). That is the ontology document file can be entirely available on a single host or it can be fragmented in one or more chunks scattered within the MANET. Notice that since several object classes, described with respect to differ-

ent ontologies, can co-exist within the same physical space, multiple u-KBs can actually populate a given environment sharing the system infrastructure. Ontology Universally Unique Identifiers (OUUIDs) are adopted to unambiguously mark ontologies and to associate each individual to its reference ontology (Ruta et al. 2006).

The **ABox** is deployed within a smart context, as KB individuals are physically tied to micro devices disseminated in the field. In RFID-based scenarios, each individual consists in semantically annotated metadata describing object/product features, stored within the RFID transponder. Each annotation refers to an ontology providing the intensional knowledge. In detail, each individual is characterized by:

- a globally unique item identifier (the EPC – Electronic Product Code – in the case of RFID tags);
- the OUUID;
- a set of contextual attributes, which allow to extend logic-based reasoning services with application-specific information (*e.g.* price is a typical parameter in mobile commerce, whereas content duration is more useful in mobile learning);
- semantic annotation, stored as a compressed document fragment in the DL-based language DIG.

In our u-KB approach, we adhere to the fundamental *Tell/Ask paradigm*. These operations, however, are implemented in a novel way, coping with the peculiarities of pervasive computing scenarios. Next Section explains in detail the data structures and protocol devised to build a u-KB system.

Tell/Un-Tell operations are *hidden* from users, *i.e.* no explicit knowledge declaration/retraction is needed. Intuitively, a u-KB is built with knowledge fragments carried by individual micro devices that populate the environment in a given instant and by the ontology they refer to. The system allows an autonomic and adaptive knowledge base maintenance, by means of a data alignment protocol between cache memories of multiple mobile devices. Each node advertises the individuals detected in its proximity (*e.g.* via RFID), other hosts store advertisements in their cache and forward them to hosts at one-hop distance. The protocol tackles the issues of moving tagged objects entering/leaving the environment by keeping track of the freshness of advertised individuals through sequence numbers, so that updates will be automatically propagated. Furthermore, each individual also has a limited time-to-live, so that it will be automatically removed (un-told) from the u-KB if not renewed. Similar mechanisms are employed to disseminate ontology chunks within the network, aiming at a trade-off between fault tolerance and generated overhead. The system aims at a steady state where every host is aware of both ontologies and individuals in the environment. To reduce storage requirements and network load, semantic annotations are not included in advertisements; when needed they are provided *on-demand* instead.

Ask operations require a preliminary resource discovery step. The requester specifies the ontology identifier and a range for each attribute it is interested in. The system then returns IP addresses of hosts owning the ontology chunks and all the individuals that meet the specified criteria. *On-demand* provisioning of KB individuals reduces transmissions for data alignment and prevents propagation issues in case of description update or individual removal. Additionally, filtering avoids unnecessary data transfers. Then the requester can fetch ontology chunks and filtered individuals by their respective providers, so that it can reconstruct a local subset of the whole KB, containing only the TBox and annotations which are actually needed. Finally, it is able to submit any Ask-type request to a local or remote reasoning engine.

5 Architecture details

In what follows, details of the proposed architecture will be provided. Particularly, an exhaustive description of the infrastructure underling the framework is presented along with dissemination and discovery protocols specification.

5.1 Layered infrastructure

Nodes participating to an ad-hoc network are moving and their inclusion within the MANET does not require configuration procedures. Each node could act as requester/provider or play the role of a router forwarding the traffic coming from nearby hosts.

The proposed framework presents a two-level infrastructure, as sketched in Fig. 2. RFID is exploited at the *field layer* (interconnecting tags dipped in the environment and readers in range), whereas the *discovery layer* is related to inter-reader ad-hoc communication. Tags in the field and readers are interconnected via the semantic-enhanced EPCglobal RFID protocol data exchange (Ruta et al., 2007), whereas the data propagation among readers is performed following a data dissemination paradigm in 802.11 (IEEE 802.11, 1999) described in what follows. Resources are autonomously exposed via the enhanced RFID and, in the same way, readers should be able to perform a discovery thanks to the preliminary propagation of data each cluster-head has seen in its range. Note that really pervasive environments do not necessarily provide stable e dependable communication links, on the contrary more probably and realistically they are infrastructure-less. Hence, an autonomous and decentralized approach to the object discovery should anyway ensure the availability of required information also when a connection with a central remote server is missing.

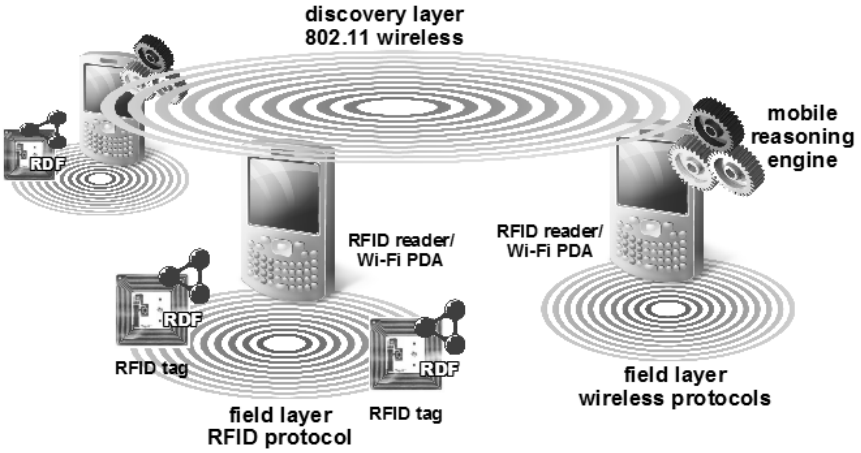


Fig. 2. Field and Discovery layers in the proposed framework architecture

Thus, the resource discovery is based on three stages:

1. the extraction of the object parameters (for carrying object features from field layer to discovery one);

2. the resource data dissemination (to make the overall nodes fully aware about the "network content");

3. the extraction of resource annotations (for carrying semantic-based descriptions from field level to the discovery one) for the further matchmaking. This phase is performed *on-demand* when a request is addressed in unicast to a selected node.

Each reader covers a central role in the whole architecture as it advertises contextual parameters referred to tags in its radio range (at the field layer) and during further phases, it will receive requests from nearby nodes (via 802.11 at the discovery layer). In case, it will extract semantic annotations from tags in range (by exploiting RFID EPCglobal) and replies to the requester. A reader maintains a cache containing the advertisements which will be matched against requests.

The proposed approach is fully decentralized: address and main characteristics of each resource/tag are autonomously advertised by related readers using small-sized messages throughout the network composed by the other readers.

A large use of broadcasting mechanism to advertise object features could result as inefficient in terms of bandwidth and power consumption (both precious resources in pervasive environments). So in the proposed approach, only resource parameters are advertised in broadcast in order to unambiguously identify both the location and the category of a resource/tag. Given them, if a node explicitly is interested in a resource, it will download in unicast the semantic annotation it is as-

sociated to. In this way, advertisement flooding (due to the broadcasting mechanism) is reduced without sacrificing the correct fruition of a resource.

5.2 Semantic-based dissemination and discovery protocol

Each resource in the MANET is labeled by means of the triple *{SOURCE_ADDRESS, OUUID, EPC}*, that is the IP address of the RFID reader which has "seen" the resource, the ontology the resource is associated with and the Electronic Product Code, respectively.

To perform the data dissemination, resource providers periodically send *advertisement PDUs* (Protocol Data Units) traveling a given number of hops (MAX_ADV_DIAMETER) at most. As shown in Fig. 3, for each managed resource, a reader advertises the reference OUUID and some context-aware parameter (*i.e.* the resource Time-To-Live (TTL)). An early selection allows to choose only semantically compatible annotations (OUUID matching) with suitable values for contextual attributes.

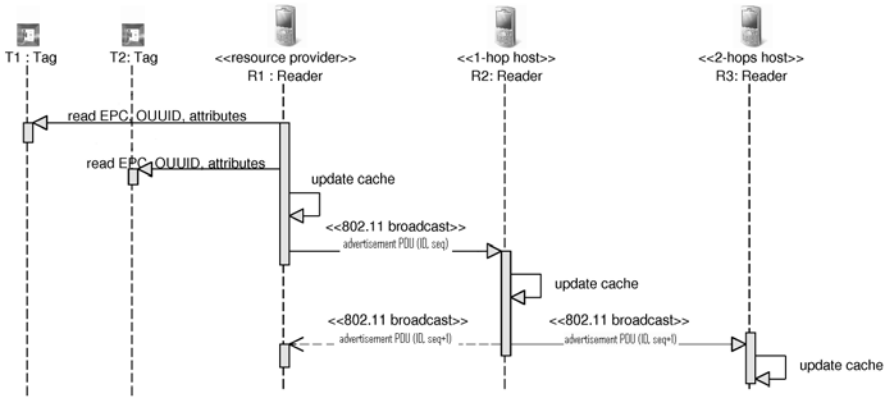


Fig. 3. Tag data dissemination phase

During their travel, advertisements are forwarded using MAC broadcasts and they are stored in the cache memories of nodes they go through. Particularly, Fig. 4 shows a cache entry. In what follows, the content of each field is reported:

- **source address**: address of resource provider;
- **size**: annotation size (in byte);
- **OUUID**: ontology identifier;
- **lifetime**: remaining time to live of a resource/tag.;

- **timestamp**: last reference to the entry (read/write). That is, when a new resource is stored or when an existing one is invoked, the field is updated;
- **traveled hops**: distance (hops number) between provider and cache holder;
- **sequence number**: reference to the last resource provider;
- **EPC**: Electronic Product Code of a resource/tag;
- **resource description**: semantic annotation of a resource. It has a variable length, but in some case it could be a pointer to a compressed DIG file.

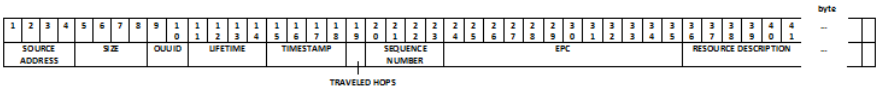


Fig. 4. The structure of a cache record

An entry is added to the cache table whenever the node receives either an advertisement or a cache content frame. The corresponding entry is updated if the PDU carries a sequence number higher than the stored one or if the route the packet suggests is shorter than the previously stored one.

Before starting whatever reasoning task, the requester has to properly recompose the TBox. To this aim techniques of hybrid peer-to-peer sharing are exploited. The goal is to avoid collecting the whole TBox before reasoning, but only the actually needed portions. For the sake of conciseness, further details about reasoning are not provided here, the interested reader can refer to (Di Noia et al., 2007).

Fig. 5 shows as resource discovery phase happens. When starting a matchmaking, a node attempts to cover the request by using resource descriptions stored within its own cache. If a requester has no stored descriptions or if managed resources do not satisfy the request, the node sends a *solicit PDU* with a maximum travel diameter (MAX_REQ_DIAMETER) to get new resource locators.

A node receiving a solicit replies (in unicast) with a *cache content PDU*, providing cache entries whose parameters match the ones contained within the solicit frame. On the contrary, it will reply with a “no matches” message. Note that, during their travel, replies to a request and solicit PDUs are used to update the cache memory of forwarding nodes.

Finally, if some semantic annotation is missing, it can be retrieved by sending a *request PDU* in unicast to the node “owning” the resource(s). The latter will reply with a *resource PDU*, containing the annotation(s).

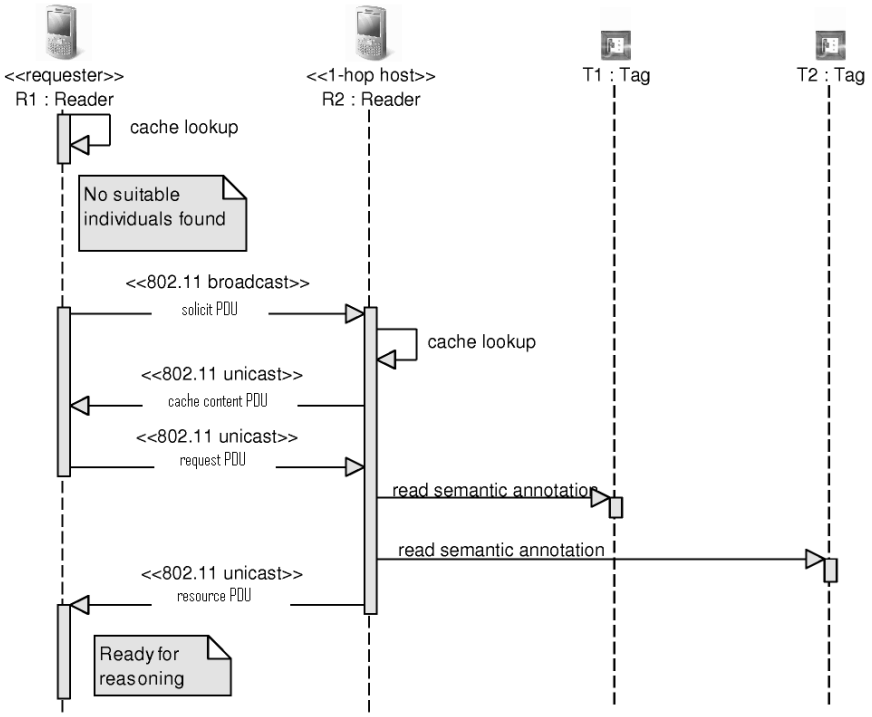


Fig. 5. Retrieval of advertised semantic annotations from tags

After receiving required information from hosts in its search range, the requester performs the matchmaking again. If the result is still unsatisfactory, the node could try to forward a new solicit by increasing the search diameter and previous steps can be repeated progressively expanding the diameter, until a maximum extension is reached.

6 Case study

The proposed framework has been specifically studied in pervasive computing environments where a wide range of objects/products are endowed with RFID transponders complying with the EPCglobal standard for class I - second generation UHF tags (EPCglobal Inc. 2005a). Mobile RFID readers equipped with IEEE 802.11 wireless connectivity create and manage the u-KB. Tagged objects are KB individuals in our system.

Previous works (Ruta et al. 2007) proposed enhancements to the EPCglobal standard which recur in our definition of u-KB individuals, as explained above. In the following subsections, main features of semantic-enhanced RFID protocol are

recalled, and interactions between RFID technology and proposed framework are outlined, respectively.

6.1 RFID-based u-KB

Tag memory is organized in four logical banks (EPCglobal 2008):

1. *Reserved*, storing optional kill and access passwords;

2. *Electronic Product Code (EPC);*

3. *Tag IDentification (TID)*, storing tag manufacturer and model identification codes;

4. *User*, optionally present for custom application data.

Contents of TID memory up to $1F_h$ bit are invariable. For tags having class identifier value $E2_h$ stored in the first byte of the TID bank, optional information could be stored in additional TID memory from 20_h address. There we store:

- a 16 bit word for *optional protocol features*, stored starting from 20_h address most significant bit first: currently only the most significant bit is used to indicate whether the tag is semantic-enabled or not; other bits are reserved for future uses;
- the OUUID of the ontology referred by the semantic annotation in the tag.

Finally, contextual attributes are stored within the User memory bank, along with the semantically annotated object description expressed in DIG. Due to verbosity of DIG language, the annotation is compressed using the encoding algorithm proposed in (Scioscia and Ruta 2009), since it allows direct queries over the compressed format.

EPCglobal UHF RFID air interface protocol consists in three steps:

1. *Preselection* by a reader of a subset of the tag population currently in range, by means of *Select* command;

2. *Inventory* loop, where the reader detects one tag in range for each iteration and reads its EPC code;

3. *Access* to the tag memory content.

In our approach, original commands are exploited in novel ways, while keeping full backward compatibility. *Select* command is used to preselect semantic-enabled tags. The inventory step is performed in the standard way. *Read* and *Write* commands in the access step can be used to extract and update the OUUID, the contextual attributes and the semantic annotation of the tagged object.

6.2 Interaction with EPCglobal RFID technology

1. Dissemination. After each advertisement period DEFAULT_RTIME, a reader scans RFID tags in its range. Only semantic-enabled ones are preselected, by means of a *Select* command whose parameters are shown in Tab. 1.

Values for the (MemBank, Pointer, Length) triple identify the bit at 20_h address in the EPC memory bank, which is compared with Mask value 1_2.

Since semantic-enabled tags are identified by having the bit set, Target and Action parameters have the effect to set the SL tag status flag only for them and to clear it for the other ones.

Table 1. SELECT command able to detect only semantic-enabled tags

Parameter	Value	Description
Target	100_2	SL flag
Action	000_2	if bits match then set target flag, else reset
MemBank	10_2	TID memory bank
Pointer	00100000_2	start address
Length	00000001_2	number of bits to compare
Mask	1_2	bit mask

The subsequent inventory step skips tags having SL flag cleared.

EPC codes of semantic-based tags are then individually scanned and TTL of corresponding cache table entries are refreshed.

If the reader detects a new EPC code not in the cache memory, it will read its OUUID and contextual attributes, exploiting two *Read* commands, as shown in Tab. 2 and Tab. 3, respectively[3].

Table 2. READ command able to extract OUUID from the TID memory bank

Parameter	Value	Description
MemBank	10_2	TID memory bank
WordPtr	00000011_2	starting address (4^{th} memory word)
WordCount	00000010_2	read 2 words (32 bits)

Table 3. READ command to extract contextual attributes from User memory bank

Parameter	Value	Description
MemBank	11_2	User memory bank
WordPtr	00000000_2	starting address (1^{st} memory word)
WordCount	00001000_2	read 8 words (16 bytes)

[3] *Read* command allows to read one or more 16-bit memory words from any of the four tag memory banks. `MemBank` parameter identifies the memory bank (as in *Select* command). `WordPtr` and `WordCount` are the starting address and the number of memory words to be read, respectively; if `WordCount` is 0, all the memory words will be read up to the end of the selected bank.

Data extracted from the RFID tag will be stored in a new cache entry with a fresh sequence number. Conversely, if the EPC code is not detected for an existing local entry in the cache table, the reader will wait for the TTL to expire before removing the entry. This prevents the well-known issue of "RFID event flickering[4]" (Römer et al. 2004) from causing incorrect removal/addition of u-KB individuals. At the end of the loop, the cache table is fully updated and the reader can issue an *advertisement* PDU to notify individuals to neighbor hosts.

2. Discovery. When a reader receives a *request* PDU, it starts an RFID scan of semantic-enabled tags only, as seen above. During inventory, for each detected EPC among those listed in the PDU payload, it reads the compressed semantic annotation stored in the User memory bank of the tag, with a *Read* command as in Tab. 4. Finally, it replies to the requester.

In addition to cache tables described in the previous section, the reader may have an optional cache for the most recently used semantic annotations in order to reduce RFID accesses. That may improve response latency and battery life.

Table 4. READ command to extract the compressed semantic annotation from User

Parameter	Value	Description
MemBank	11_2	User memory bank
WordPtr	00000000_2	starting address
WordCount	00000000_2	read up the end

3. Implication of RFID data locality. In common scenarios, the content of a particular RFID tag can be relevant to a user only if she is in the same environment as the tagged object is. For instance, in a large shopping mall, buyers are implicitly interested in products of the department they are currently in. The proposed data dissemination protocol exploits this locality property and delivers advertisements only to readers within a maximum hop distance from the advertiser (which is supposed to be physically close to the RFID transponder). The value of this operational parameter can be adjusted to balance network load and physical extension of a u-KB, according to specific application requirements. In the above shopping example, a different u-KB will be built in each store department by IEEE 802.11-enabled RFID readers placed on shelves. Two departments far from each other should not and will not share information. Nevertheless, each department may have a gateway node allowing interactions between the local u-KB system and a back-end warehouse information system.

[4] Due to collisions, a tag might not be detected in every consecutive scan. This phenomenon can trigger spurious leave-enter event pairs.

6.3 Encoding of semantic annotations

In spite of RFID tags integrate memories with an increasing capacity (nowadays RFIDs with at least 4kB memory already have wide diffusion (M Ayoub et al. 2009)), the compression of semantic annotations is a relevant technical issue for our proposal. XML-based languages adopted in the Semantic Web are too verbose to allow efficient data management in mobile environments. Encoding techniques become essential in order to enable proper storage and transmission of annotations on tiny devices such as RFID tags. Moreover, compression reduces communication delays, implies an efficient usage of bandwidth and improves battery performances.

When evaluating approaches to compression, besides compression ratio and required processing/memory resources, *efficiency of queries* on compressed data must be taken into account. Recent research has been increasingly focused on compression schemes for XML documents allowing to directly query encoded annotations (Min et al. 2006), which would eliminate the need for decompression before query evaluation. To this aim, we designed an approach for compression of XML-based semantic annotations able to support queries directly on compressed data (Scioscia and Ruta 2009). Reverse Arithmetic Encoding (RAE), a semi-adaptive and homomorphic technique (Min et al. 2006) was adapted to that purpose. *Semi-adaptive* compression works in two steps: the input data is scanned once for statistics collection and again for encoding. *Homomorphic* compression (Tolani and Haritsa 2002) preserves the structure of the original XML data. On the contrary, with non-homomorphic algorithms the document structure is not recognizable after compression. Homomorphism is the feature allowing to avoid decompression before detecting document pieces which satisfy given query conditions. The approach was implemented into a tool called *COX* (Compressor for Ontological XML-based languages) and another one, *DECOX* (DECompressor) which can be used for documents in RDF, DIG and OWL formats.

COX distinguishes data structures and data (particularly, for data structures - XML tags and attributes- RAE is used). It adopts two different compression solutions. Both techniques require a two-stage process. In the first step the XML document is parsed and statistics are gathered. In the second step compression is performed and output is written.

1. First step. DIG and OWL languages only contain tags, attributes and attribute values (excluding other syntax elements such as comments and processing instructions). COX deals with tag and attribute names in the same way, only distinguishing the latter by means of a "@" prefix added to the name. Therefore, from now on with the term "tag" we will refer either to tags or attributes.

Let us consider a very short DIG document instance, describing a refrigerated cow milk delivery with respect to a reference ontology for the food products domain (not reported here):

```
<tells xmlns="http://dl.kr.org/dig/2003/02/lang">
  <defindividual name="milk_delivery_M2"/>
```

```
    <instanceof>
      <individual name="milk_delivery_M2"/>
      <and>
        <catom name="cow_milk"/>
        <all>
          <ratom name="processed_with"/>
          <catom name="refrigeration"/>
        </all>
      </and>
    </instanceof>
  </tells>
```

In the first step the document is parsed and the syntactic tree in Fig. 6 is built in memory.

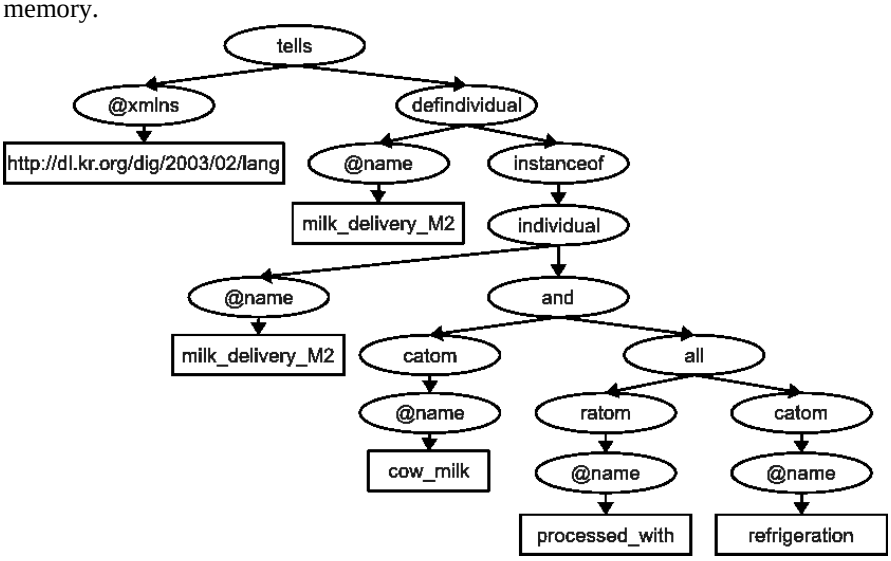


Fig. 6 COX tree model for the example DIG document**.**

After parsing, according to the arithmetic encoding scheme (Witten et al. 1987), an adjusted frequency of each tag name is calculated as ratio between the number of occurrences of a tag and the total document tags. Then the interval $[d,D) = [1.0+2^{-7}, 2.0-2^{-15})$ is split into disjoint sub-intervals (each associated to a single tag) using the sub-interval calculation function:

$$a_i = kf_i + \frac{1-k}{n}$$

where a_i is the length of the i-th sub-interval, f_i is the adjusted frequency of the i-th tag, and n is the number of different tags. The weight k is defined as:

$$k = 1 - \sqrt{\sigma}$$

where σ is the standard deviation of absolute frequencies of all document tags. The formula is a linear combination of two components: the first one is propor-

tional to tag frequency, whereas the second one is fixed for all tags. When the frequency distribution of tags in a document is regular ($\sigma \to 0$ and $k \to 1$), the proportional term dominates. Conversely, for irregular frequency distributions (the most common case) σ increases and k decreases, so that the second component of the formula increases and even very rare tags receive a not-too-small sub-interval. This prevents errors in decompression for tags with a very low frequency (in the order of 10^{-4} or lower).

The minimum value of the sub-interval assigned to each tag is computed with the formula:

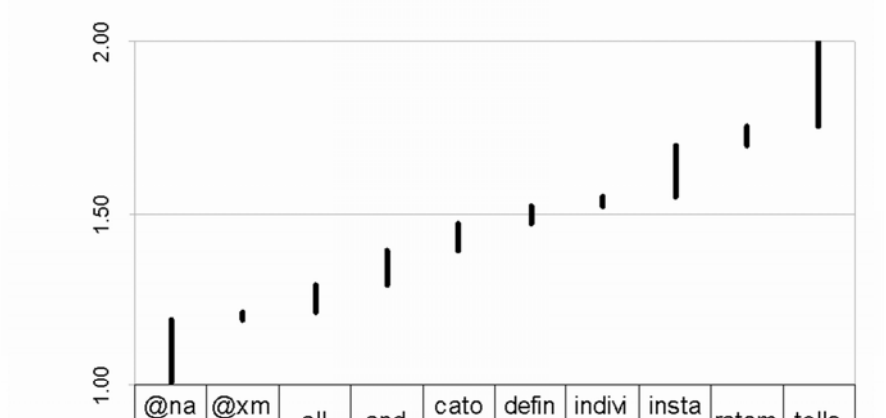
$$min_{i+1} = min_i + a_i(D-d), \qquad i = 1, \cdots, n-1$$

where n is total number of tags and $min_1 = d$. Hence, values referred to opening tags fall in the interval $[d,D)$.

Values of sub-interval limits for the above example document are reported in Fig. 7.

The interval $[1.0,d)$ is reserved to encode closing tags. It can be verified that every possible value strictly falls between 1.0 and 2.0. By exploiting 32-bit floating point representation, the first byte will always be 01111111_2, so it can be truncated without loss of information (Min et al. 2007).

Concluding the first step, a header is written at the beginning of the output file. It contains a sequence of records composed by: 1 byte for both tag name length and tag name itself; 3 bytes (after truncation) for the encoding of tag sub-interval minimum value.



	@name	@xmlns	all	and	catom	definidividu	individual	instanceof	ratom	tells
max_i	1.187	1.212	1.289	1.392	1.469	1.520	1.546	1.700	1.751	1.999
min_i	1.007	1.187	1.212	1.289	1.392	1.469	1.520	1.546	1.700	1.751

Fig. 7. Computed intervals associated to tags in the example document**.**

Attribute values are interpreted as ASCII strings; most recurrent words are identified and encoded with 16-bit sequences. The statistic collection for their fre-

quencies is performed concurrently with the document structure analysis. Since a simple code substitution is applied, homomorphism is obeyed also in this case.

Note that although a significant size saving can be obtained in ontologies with recurrent elements, on the other hand the use of a header could lower compression performance for short documents. As a consequence, a heuristic rule was adopted: only attributes above both a length threshold and a frequency threshold are encoded. A header is thus built, containing correspondences between each attribute string and the related code.

2. Second step. The body of the output file is produced. Opening and closing tags, attributes and attribute values are encoded in the order as they appear in the input document. An *opening tag T* (or the beginning of an attribute) is encoded with RAE. We follow the same approach of XPRESS (Min et al. 2007) to encode values referred to tags or sequences of tags in a path. The applied algorithm is reported in what follows:

input *T* tag to be encoded, *subint* associative array of sub-intervals assigned to tags
output $n \in [d, D)$ real number that encodes the tag path from *T* up to the root tag
1. (*min*, *max*) := *subint*[*T*]
2. **while** has_parent_tag(*T*)
3. *T* := get_parent_tag(*T*)$
4. (*begin*, *end*) := *subint*[*T*]$
5. *min* := *min* + (*max* - *min*) * *begin*
6. *max* := *min* + (*max* - *min*) * *end*
7. **end while**
8. *n* := *min*

Basically, the interval associated to the tag path from *T* to the root (hence the *reverse* adjective), is computed by progressively narrowing the starting interval according to the size of sub-intervals assigned in the first step. In this way, every value in the final numeric interval unambiguously identifies the sequence of tags from *T* up to the root. Given a 32-bit floating point representation of the output value, to encode a tag our actual implementation only takes the two central bytes. In addition to the first one (which is fixed), the last byte (which is the least significant mantissa byte) is truncated to save space. We verified this intentional loss of precision does not prevent a correct tag reconstruction in decompression phase.

With reference to the previous DIG document excerpt, the following Fig. 8 clarifies the compression algorithm behavior, showing how compression procedure progressively calculates the intervals associated to each document portion. The figure illustrates the processing of the `name` attribute under the `defindividual` tag, at line 2 of the document. The path from the leaf to the root element is built as `@name – defindividual – tells`. Then the algorithm loop starts: firstly, in iteration (a) the sub-interval corresponding to `@name` is taken (as from Fig. 7). In the next iteration (b), `defindividual` is considered and the interval computed in (a) is reduced in a proportional way to the sub-interval of `defindividual` (from Fig. 7) with respect to the base interval [*d*,*D*). Similarly, in step (c) tells is considered and the interval computed in (b) is reduced proportion-

ally to the sub-interval of `tells` (again from Fig. 7) with respect to the base interval. Once the root element of the document has been reached, the loop terminates: the numerical interval that identifies unambiguously the above path is [1.09897,1.10126). The lower bound is finally taken to encode the path.

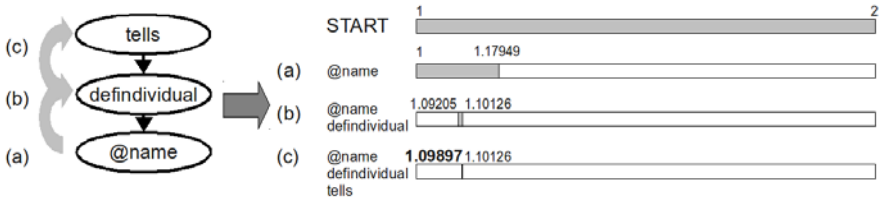


Fig. 8. Encoding procedure example**.**

A *closing tag* or the end of an attribute are encoded by the two central bytes of the 32-bit floating point representation of 1.0070 and 1.00444, respectively. These values were chosen because truncating does not cause loss of precision.

Finally, an *attribute value* is processed as follows: if it was encoded in the first step, it is replaced by its 1-byte code followed by the delimiter byte $\texttt{fe}_h$, otherwise the ASCII string is copied to output, followed by the delimiter $\texttt{ff}_h$.

7 Experiments

Preliminary tests about read/write time from/to semantic-enhanced tags have been performed in (Di Noia et al. 2008) in order to provide an early evaluation of the impact that EPCglobal protocol evolution may have on RFID system performance. They are used as initial evidence that adoption of compressed semantic resource annotations on RFID tags does not impair semantic-based RFID applications with respect to most common ones.

Furthermore, the u-KB overall framework proposed here has been verified and tested using *ns-2* network simulator in a proper simulation campaign. Performance evaluation includes the following metrics (Ruta et al. 2009).

1. *Network load*, assessed by means of the total number of packets generated at discovery layer. Results show that traffic has higher correlation with the number of resource providers rather than requesters (clients). This happens because advertisement frames are regularly sent in a proactive way by providers, even if there are no requests. On the other hand, solicit, cache content and request packets are produced *on-demand* by client hosts and their neighbors.

2. *Hit ratio*, *i.e.* percentage of successful resource retrieval in a variety of scenarios. Obtained hit ratios are very high, with values above 90% in all tests and above 95% in more than half the tests. As a general consideration, this clearly indicates the effectiveness of the proposed approach.

3. *Duration* of service discovery. For a given number of resource providers, the service time decreases as requesters increase. This happens because, when a solicit PDU is answered by cache content PDUs, intermediate nodes cache related resource records. This reduces latency in replying to later requests. Overall values are still slightly high with respect to typical needs of pervasive scenarios. This result may be influenced by the fact that current implementation is not specifically optimized for execution speed.

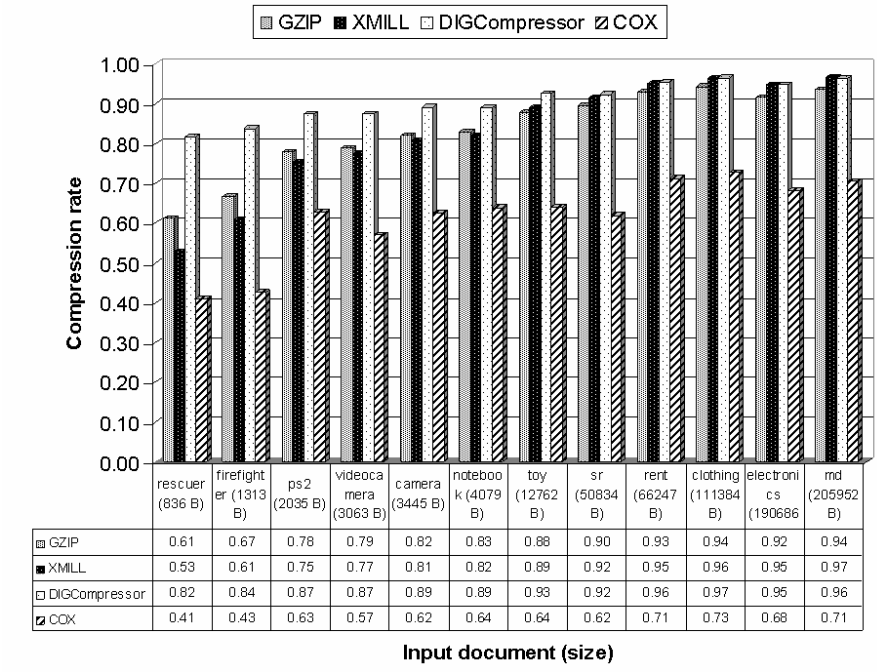
A second experimental campaign was performed to assess benefits and drawbacks of different compression approaches for semantic annotations. The proposed COX tool was compared to: (i) *gzip*[5], a widely-adopted general-purpose compressor; (ii) *XMill* (Liefke and Suciu 2000), an XML-specific compression tool and (iii) our own previous tool *DIGCompressor* (Ruta et al. 2007), which was optimized for high compression rates and execution speed.

Performance comparison has been carried out estimating three basic parameters: (1) *compression rate*, defined as (1 - $size_{compressed}$ / $size_{uncompressed}$); (2) *turnaround time*, that is the overall duration of the compression process; (3) *memory usage* by the compressor process.

Tests were performed using a laptop PC equipped with an Intel P8400 Core 2 Duo CPU (2.27 GHz clock frequency), 3 GB RAM at 800 MHz and Ubuntu 9.04 GNU/Linux operating system with 2.6.27 kernel version and *Valgrind* (Nethercote and Seward 2007) 3.4.1 profiling toolkit. To obtain a complete picture of performance, 12 DIG documents of different size were used, 6 semantic annotations of individual objects/products and 6 domain ontologies, which are significantly longer.

Fig. 9 shows the resulting compression rate. For each DIG file, the original size in byte is reported. COX and DIGCompressor were respectively the worst and the best performer. Compression rate of COX is somewhat sacrificed in order to allow general-purpose compression with support for direct queries over the encoded format.

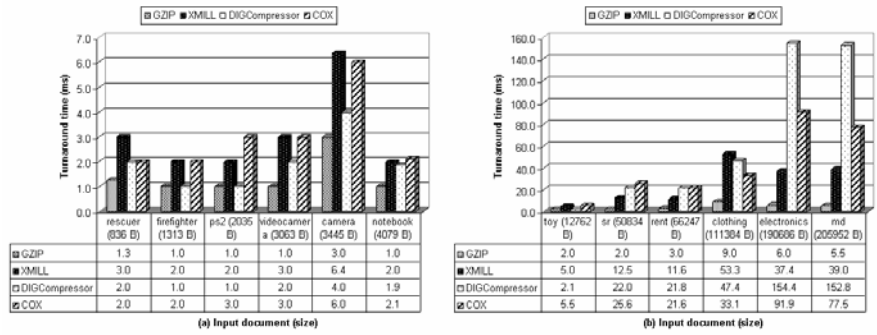
[5] GZIP compression utility: http://www.gzip.org/



	rescuer (836 B)	firefighter (1313 B)	ps2 (2035 B)	videocamera (3063 B)	camera (3445 B)	notebook (4079 B)	toy (12762 B)	sr (50834 B)	rent (66247 B)	clothing (111384 B)	electronics (190686	md (205952 B)
GZIP	0.61	0.67	0.78	0.79	0.82	0.83	0.88	0.90	0.93	0.94	0.92	0.94
XMILL	0.53	0.61	0.75	0.77	0.81	0.82	0.89	0.92	0.95	0.96	0.95	0.97
DIGCompressor	0.82	0.84	0.87	0.87	0.89	0.89	0.93	0.92	0.96	0.97	0.95	0.96
COX	0.41	0.43	0.63	0.57	0.62	0.64	0.64	0.62	0.71	0.73	0.68	0.71

Fig. 9. Performance comparison – compression rate.

For turnaround time, each test was run 10 times consecutively, and the average of the last 8 runs was taken. Results for annotations are reported in Fig. 10-a. gzip is the most optimized tool. XMill and COX suffer from the intrinsic complexity of their algorithms, but absolute values are still acceptable. For ontology documents, (see Fig. 10-b), the scenario changed. COX provides acceptable performance, whereas DIGCompressor has significantly higher turnaround times for input longer than 100 kB.

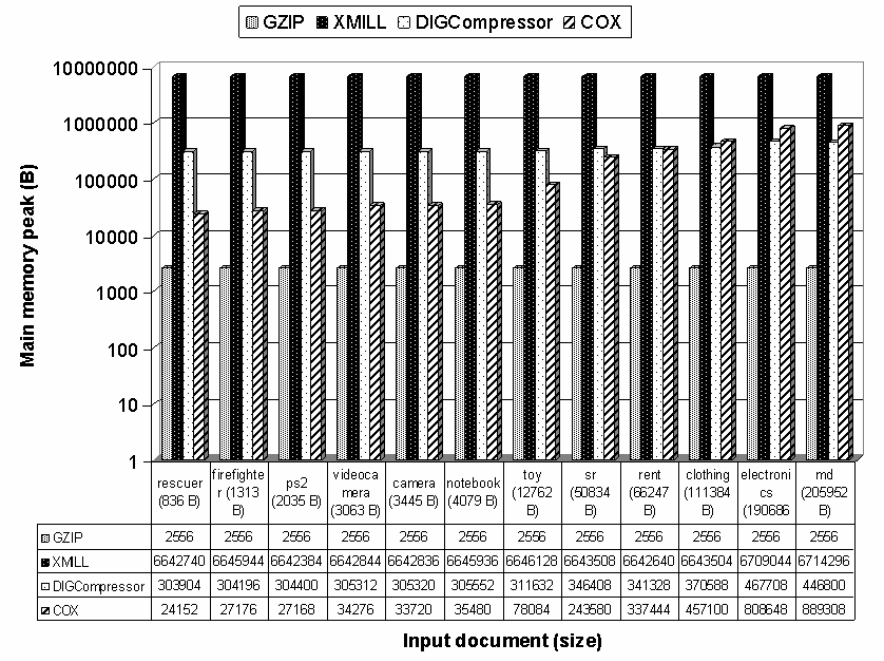


	rescuer (836 B)	firefighter (1313 B)	ps2 (2035 B)	videocamera (3063 B)	camera (3445 B)	notebook (4079 B)
GZIP	1.3	1.0	1.0	1.0	3.0	1.0
XMILL	3.0	2.0	2.0	3.0	6.4	2.0
DIGCompressor	2.0	1.0	1.0	2.0	4.0	1.9
COX	2.0	2.0	3.0	3.0	6.0	2.1

	toy (12762 B)	sr (50834 B)	rent (66247 B)	clothing (111384 B)	electronics (190686 B)	md (205952 B)
GZIP	2.0	2.0	3.0	9.0	6.0	5.5
XMILL	5.0	12.5	11.6	53.3	37.4	39.0
DIGCompressor	2.1	22.0	21.8	47.4	154.4	152.8
COX	5.5	25.6	21.6	33.1	91.9	77.5

Fig. 10. Performance comparison – turnaround time: (a) instance annotations; (b) ontologies.

In conclusion, memory usage analysis was performed using *Massif* tool of Valgrind toolkit. For our comparison, only the memory peak was considered. Results are shown in Fig. 11. XMill and gzip, due to their specific structure, use an almost constant memory amounts regardless of the input size. DIGCompressor uses about 300 kB for small annotations, growing up to 460 kB for bigger ontologies. COX is much more efficient (about 35 kB) for small documents, but its memory peak exceeds DIGCompressor for the largest inputs.



	rescuer (836 B)	firefighter (1313 B)	ps2 (2035 B)	videocamera (3063 B)	camera (3445 B)	notebook (4079 B)	toy (12762 B)	sr (50834 B)	rent (66247 B)	clothing (111384 B)	electronics (190686	md (205952 B)
GZIP	2556	2556	2556	2556	2556	2556	2556	2556	2556	2556	2556	2556
XMILL	6642740	6645944	6642384	6642844	6642836	6645936	6646128	6643508	6642640	6643504	6709044	6714296
DIGCompressor	303904	304196	304400	305312	305320	305552	311632	346408	341328	370588	467708	446800
COX	24152	27176	27168	34276	33720	35480	78084	243580	337444	457100	808648	889308

Fig. 11. Performance comparison – main memory peak.

8 Conclusion

This chapter presented an approach to carry out an advanced matchmaking using semantic metadata stored in RFID tags without requiring unique and fixed knowledge bases. An advanced resource discovery framework is supported by a knowledge dissemination protocol, so allowing an *on-demand* retrieval of suitable descriptions directly from tags located on the objects. An analytical model has been developed for a preliminary assessment of resource requirements. An experimental validation of the proposed approach –including dissemination, discovery and RFID data compression evaluations– has been performed in order to test its feasibility.

References

Baader F, Calvanese D, Mc Guinness D, Nardi D, Patel-Schneider P (2002) The Description Logic Handbook. Cambridge University Press.

Bechhofer S, Möller R, Crowther P (2003) The DIG Description Logic Interface. Proceedings of the 16th International Workshop on Description Logics (DL'03). Volume 81 of CEUR Workshop Proceedings.

Brachman R, McGuinness D, Patel-Schneider P, Resnick L, Borgida A (1991) Living with CLASSIC: When and how to use a KL-ONE-like language. Principles of Semantic Networks, 401–456, Morgan Kaufmann Publishers.

Chakraborty D, Joshi A, Yesha Y, Finin T (2006) Toward distributed service discovery in pervasive computing environments, IEEE Transactions on Mobile Computing, 5(2), 97-112.

Cohen B (2008) The BitTorrent Protocol Specification, version 11031. http://www.bittorrent.org/beps/bep_0003.html. Accessed 20 November 2009.

Di Noia T, Di Sciascio E, Donini F M (2007) Semantic matchmaking as non-monotonic reasoning: A description logic approach, Journal of Artificial Intelligence Research (JAIR), Volume 29, 269–307.

Di Noia T, Di Sciascio E, Donini F M, Ruta M, Scioscia F, Tinelli E (2008) Semantic-based Bluetooth-RFID interaction for advanced resource discovery in pervasive contexts**,** International Journal on Semantic Web and Information Systems, 4(1), 50–74.

EPCglobal Inc (2005) Object Naming Service (ONS} - ver. 1.0 . EPCglobal Ratified specification.

EPCglobal Inc (2008) EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz-960 MHz Version 1.2.0 EPCglobal Ratified specification.

Harrusi S, Averbuch A, Yehudai A (2006). XML syntax conscious compression. In Data Compression Conference (DCC2006), pages 10-19.

Howard P, Vitter J (1991) Analysis of arithmetic coding for data compression. In Data Compression Conference (DCC'91), 3–12.

Huffman D (1952). A method for the Construction of Minimum Redundancy Codes. In Proceedings of the IRE, volume 40, pages 1098–1101.

IEEE 802.11 (1999) Information Technology Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. ANSI/IEEE Std. 802.11, ISO/IEC 8802-11. First edn.

Kawakita Y, Mistugi J (2006) Anti-collision performance of Gen2 air protocol in random error communication link. Proceedings of the International Symposium on Applications and the Internet Workshops - SAINT 2006, 68–71.

Levesque H (1984) Foundations of a Functional Approach to Knowledge Representation. Artificial Intelligence, 23, 155–212.

Liefke H, Suciu D (2000). Xmill: an efficient compressor for xml data. SIGMOD Rec., 29(2):153–164.

M Ayoub K, Manoj S, Brahmanandha P R (2009) A Survey of RFID Tags, International Journal of Recent Trends in Engineering, 1(4), 68-71.

MacGregor R (1991) The Evolving Technology of Classification-Based Knowledge Representation Systems. Principles of Semantic Networks: Explorations in the Representation of Knowledge, 385-400, Morgan Kaufmann Publishers.

Mahoney M (2005). Adaptive Weighing of Context Models for Lossless Data Compression. Technical report, Florida Tech. Technical Report CS-2005-16.

McGuinness DL, van Harmelen F (2004) OWL Web Ontology Language, W3C Recommendation. http://www.w3.org/TR/owl-features/. Accessed 20 November 2009.

Min J, Park M, Chung C (2006). A compressor for effective archiving, retrieval, and updating of XML documents. ACM Transactions on Internet Technology (TOIT), 6(3):223–258.

Nethercote N, Seward J. Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation. In Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation - PLDI 07, pages 89–100. ACM Press New York, NY, USA.

Network Simulator (1995). The Network Simulator – ns-2. http://www.isi.edu/nsnam/ns/. Accessed 20 November 2009.

Ni LM, Zhu Y, Ma J, Li M, Luo Q, Liu Y et al. (2005) Semantic Sensor Net: An Extensible Framework. Lecture Notes in Computer Science, 3619, 1144-1153, Springer Berlin / Heidelberg.

Pallapa G, Das S (2007) Resource Discovery in Ubiquitous Health Care. 21st International Conference on Advanced Information Networking and Applications Workshops, AINAW'07, vol. 2.

Patel-Schneider P, Swartout B (1993) Description-Logic Knowledge Representation System Specification, KRSS Group of the ARPA Knowledge Sharing Effort.

Ramanathan R, Redi J (2002) A brief overview of ad hoc networks: Challenges and directions. IEEE Communications Magazine 40, 20–22.

Römer K, Schoch T, Mattern F, Dübendorfer T (2004). Smart Identification Frameworks for Ubiquitous Computing Applications. Wireless Networks, 10, 689–700.

Ruta M, Di Noia T, Di Sciascio E, Donini FM (2006) Semantic-Enhanced Bluetooth Discovery Protocol for M-Commerce Applications. International Journal of Web and Grid Services, 2, 424–452.

Ruta M, Di Noia T, Di Sciascio E, Scioscia F, Piscitelli G (2007) If objects could talk: A novel resource discovery approach for pervasive environments. International Journal of Internet and Protocol Technology (IJIPT), Special issue on RFID: Technologies, Applications, and Trends, 2, 199–217.

Ruta M, Scioscia F, Di Noia T, Di Sciascio E, Piscitelli G (2009). Ubiquitous knowledge-based framework for RFID semantic discovery in smart u-Commerce environments. Proceedings of the 11th International Conference on Electronic Commerce, page 9-18, ACM.

Scioscia F, Ruta M (2009) Building a Semantic Web of Things: issues and perspectives in information compression. Semantic Web Information Management (SWIM'09). In Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC 2009), 589-594.

Tolani P, Haritsa J (2002). XGRIND: A Query-friendly XML Compressor. In Proceedings of the 18th International Conference on Data Engineering (ICDE.02), page 225-234, IEEE.

Toninelli A, Corradi A, Montanari R (2008) Semantic-based discovery to support mobile context-aware service access, Computer Communications, Elsevier.

Traub K, Allgair G, Barthel H, Bustein L, Garrett J, et al. (2005) EPCglobal Architecture Framework. EPCglobal Ratified specification.

Vasudevan V (2004) Ensembleware: contextual service provisioning in an ubiquitous services world. International Symposium on Applications and the Internet 2004, 10.

Vazquez JI, Lopez-de-Ipina D (2007) mRDP: An HTTP-based lightweight semantic discovery protocol, Computer Networks 51(16), 4529-4542, Elsevier.

Venkataramani G, Iyer P (2007) Semantics-aware RFID Middleware for Personalized Web Services to Retail Customers. International Workshop on Service-Oriented Engineering and Optimization, Goa (India), December 2007.

Watson R, Pitt L, Berthon P, Zinkhan G (2002) U-Commerce: Expanding the Universe of Marketing. Journal of the Academy of Marketing Science, 30, 333–347.

Weinstein R (2005) RFID: A technical overview and its application to the enterprise. IT Professional, 7, 27–33.

Witten I, Neal I, Cleary J (1987). Arithmetic coding for data compression. Communications of the ACM, 30(6):520–540.

Ziv J, Lempel A (1977). A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 23(3):337–343.

Index