Semantic-enhanced EPCglobal Radio-Frequency IDentification

Michele Ruta[1], Tommaso Di Noia[1], Floriano Scioscia[1], Eugenio Di Sciascio[1]

SisInfLab, Politecnico di Bari, Bari, Italy
{m.ruta,t.dinoia,f.scioscia,disciascio}@poliba.it

Abstract. We propose to enhance EPCglobal RFIDs enriching them with semantic capabilities. Memory organization of tags and the data exchange protocol are exploited and extended to the purpose. By design, the proposed enhancements do not alter the basic behavior of protocol and tag memory organization and are thus fully backward compatible. In order to store annotated descriptions, a compression algorithm for DIG syntax has also been devised. We report here results in comparison with other XML-based compression tools and simulations for enhanced tags reading and decompression.

1 Introduction

Radio-Frequency IDentification (RFID) is a promising infrastructure-less technology interconnecting via radio two main components: **(1)** a transponder carrying data (*tag*) located on the object to be identified; **(2)** an interrogator (*reader*) able to receive the transmitted data. Traditional RFID applications have been focused on supply chain management and asset tracking [12]. Nevertheless, at the state of the art, tags with higher memory capacity and on-board sensors disclose new scenarios and enable further applications. Currently, RFID technology is merely used as a link between physical objects and a "virtual counterpart" [9] in the digital world. Tags only store an identification code, which is used as a key to retrieve relevant properties of the object from an information server, through a networked infrastructure. Two main issues restrain an overall exploitation of the standard capabilities. First of all, the original identification mechanism only enables a rudimentary string matching, providing exclusively "yes/no" replies. Furthermore, RFID-based technology usually relies on stable support infrastructure and fixed database servers.

We propose an extension of EPCglobal RFID standard [11] supporting logic-based formalisms for knowledge representation and enabling advanced services. Semantic-based annotations are stored on RFID tags, exploiting machine understandable ontological languages originally conceived for the Semantic Web effort. Noteworthily, protocols to read/write tags are preserved in the proposed extension, maintaining the original code-based access, thus keeping a backward compatibility with basic applications practically without any modification.

According to W3C recommendations for mobile applications [7], our approach copes with limited storage and computational capabilities of mobile and

embedded devices, and with reduced bandwidth provided by wireless links. Issues related to the verbosity of semantic annotation languages cannot be neglected. Compression techniques become essential to enable storage and transmission of semantically annotated information in mobile contexts. Hence, in order to make our approach sustainable in reality, we devised and exploited a novel efficient XML compression algorithm, specifically targeted for DIG 1.1 [1] document instances.

2 Motivation

The main idea of our approach is that a semantic-based extension of current RFID technology supporting formalisms for knowledge representation, allows semantically rich and unambiguous information to follow an object in each step of its life cycle. Products then auto-expose their description to whatever RFID-enabled computing environment they are dipped in. This favors decentralized approaches for context-aware applications in pervasive computing environments, based on less expensive and more manageable mobile ad-hoc networks. Product and process information can be queried, updated and integrated during manufacturing, quality control, packaging and supply chain management, thus allowing full traceability up to sales, and intelligent and de-localized querying of product data. Semantic-enhanced RFID object discovery can be leveraged also for sales and post-sale services, by assisting customers in using the products they purchased more effectively.

Beyond manufacturing and commerce, other application areas can benefit from adding accurate semantic-based object description to traditional RFID identification and tracking capabilities. For example, in tourism settings such as museums or archaeological sites, visitors could perform interactive knowledge discovery by approaching tagged items with an RFID-enabled mobile device and querying the system for further resources of interest. In the healthcare sector, relevant information can be embedded within RFID tags attached to patient accessory (*e.g.*, wristband) and to drug packages. Since no further infrastructure is needed, support can be provided for patient diagnosis and therapy at the hospital as well as for follow-up at home.

3 Proposed Enhancements

3.1 EPCglobal RFID standards

In our framework we refer to RFID transponders conforming to the EPC (Electronic Product Code) standard for class I - second generation UHF tags [11]. We assume the reader be familiar with basics of this technology.

The practical feasibility of a proposal for advanced usage of RFID technologies must take into account some important constraints. First of all the severe bandwidth and memory limitations of current RFID systems, in order to meet cost requirements for large-scale adoption. Due to technological advances and

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

PARAMETER	Target	Action	MemBank	Pointer	Length	Mask
VALUE	100_2	000_2	01_2	00010101_2	00000010_2	11_2
DESCRIPTION	SL flag	set (if match)	EPC bank	initial address	bit to be compared	bit mask

growing demand, passive RFID tags with greater memory amounts are expected to be available [2]. Nevertheless, XML-based ontological languages like OWL (http://www.w3.org/TR/owl-features/) and DIG (http://dl.kr.org/dig/) are far too verbose for a direct storage on RFID tags. A further goal is to preserve the original EPCglobal RFID technology standards as much as possible, in order to ensure compatibility and smooth coexistence of new semantic-based object discovery applications and legacy identification and tracking ones.

In order to enable the outlined enhancements, RFID tags and the air interface protocol must provide read/write capabilities for semantically annotated product descriptions w.r.t. a reference ontology, along with additional data-oriented attributes. Neither new commands nor modification to existing ones have been introduced. Moreover, a mechanism is clearly required to distinguish semantic enabled tags from standard ones, so that semantic based applications can exploit the new features without interfering with legacy applications. In order to accomplish that, we extend the memory organization of tags compliant with the above referenced standard. We exploit two bits in the EPC tag memory area currently reserved for future purposes. The first one –at 15_{hex} address– is used to indicate whether the tag has a user memory (bit set) or not (bit reset). The second one –at 16_{hex} address– is set to mark semantic enabled tags. In this way, a reader can easily distinguish semantic based tags by means of a SELECT command with parameter values as in Table 1. Values for the triple $\langle MemBank, Pointer, Length\rangle$ identify the two-bit memory area starting at 15_{hex} address in the EPC memory bank. The reader commands each tag in range to compare those two bits with bit mask 11_2. The match outcome will be positive for semantic enabled tags only. The *Target* and *Action* parameter values mean that in case of positive match the tag must set its *SL* flag and clear it otherwise. The following inventory step will skip tags having SL flag cleared, thus allowing a reader to identify only semantic enabled tags. Protocol commands belonging to the inventory step have not been described, because they are used in the standard fashion.

The EPC standard requires the content of TID memory up to $1F_{hex}$ bit is fixed. TID bank can be extended to store optional information, generally consisting of tag serial number or manufacturer data. Hence we use the TID memory area starting from 100000_2 address to store a 128-bit *Ontology Universally Unique Identifier* (OUUID) marking the ontology w.r.t. the description contained within the tag is expressed [10]. In order to retrieve the OUUID stored within a tag, a reader will exploit a READ command by adopting parameter values as in Table 2. *MemBank* parameter identifies the TID memory bank and the *WordPtr* value specifies that the reading must start from the third 16-bit memory word, *i.e.*, from 20_{hex} address. Finally, the *WordCount* parameter indicates that 128 bits (eight 16-bit words) have to be read.

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

PARAMETER	MemBank	WordPtr	WordCount
VALUE	10_2	00000010_2	00001000_2
DESCRIPTION	TID memory bank	starting address	read up to 8 words (128 bit)

Table 3. READ command able to extract the semantically annotated description from the User memory bank

PARAMETER	MemBank	WordPtr	WordCount
VALUE	11_2	00000000_2	00000000_2
DESCRIPTION	User memory bank	starting address	read up to the end

Contextual parameters (whose meaning may depend on the specific application) are stored within the *User memory bank* of the tag. There, we also store the semantically annotated description of the product the tag is clung to (compressed with the algorithm described later on). An RFID reader can perform extraction and storing of a description from/on a tag by means of one or more READ or WRITE commands, respectively. Both commands are obviously compliant with the RFID air interface protocol. Table 3 reports parameter values of the READ command for extracting the full contents of the User memory, comprising both contextual parameters and the compressed annotation.

The EPCglobal standard also provides a support infrastructure for RFID applications by means of the so called *Object Naming Service* (ONS) [3]. In our approach the ONS mechanism is considered as a supplementary system able to grant the *ontology support*. If a reader does not manage the ontology the description within the tag refers to, we may retrieve it exploiting the ONS service. The *EPCglobal Network Protocol Parameter Registry* maintains all the registered service suffixes (*ws* for a Web service, *epcis* for a EPCglobal Information Service (providing authoritative information about objects associated with an EPC code), *html* for a Web Page of the manufacturer). We hypothesize to register the new *dig* suffix to indicate a service able to retrieve ontologies with a specified OUUID value.

In case of EPC derived from the GS1 standard[1], we assume that the pair of fields used for ONS requests –and referred to the manufacturer and to the merchandise class of the good– will correspond to a specific ontology. In fact that pair identifies exactly the product category. Two goods with the same values for that field parameters will be surely homogeneous or even equal. Nevertheless the vice versa is not verified.

3.2 Compression algorithm

A compression algorithm specifically targeted to the packing of standard DIG 1.1 format has been devised in our framework. The general approach, however, is easily adaptable to any other ontological language based on XML, such as OWL. Each DIG document instance conforms to DIG XML Schema, which comprises

[1] GS1 (originally EAN.UCC) is the international organization that introduced the bar code identification of products and services.

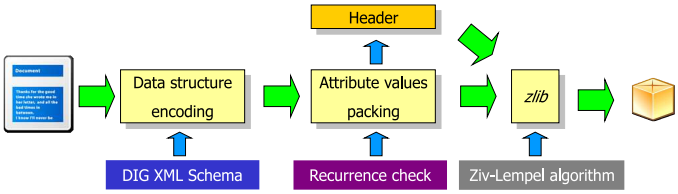

Fig. 1. Structure of the proposed DIG compression tool

at most 40 different tags. In a DIG document, no value is set inside any tag; only tag attributes can be specified, within a well defined finite set of types.

We propose a simple DIG compression solution particularly suitable for pervasive applications, whose structure is shown in Figure 1. Three fundamental phases can be identified: **(1)** *data structures packing*; **(2)** *attribute values packing*; **(3)** *zlib packing*. We exploit the peculiarity of the DIG format having few, well defined and limited tag elements.

(1) *Data-structures packing.* The proposed compression algorithm is based on two fundamental principles. First of all, pure data have to be divided from data structures; furthermore data and data structures have to be separately encoded in order to obtain a higher compression rate. Data structures are basically XML elements with possible related attributes, whereas data simply are attribute values. As noted above, data-structures in DIG syntax are fixed and well defined by DIG XML Schema, whereas data are different from document to document. XML elements are coded by associating an unambiguous 8-bit code to each structure in a static fashion. Consider that DIG files adopt either `ISO 8859-1` or `UTF-8` character encodings, which use 1 byte for each character (special characters requiring more than 1 byte in `UTF-8` do not belong to the DIG symbol set): so an early size saving is achieved. The association between XML structures and the corresponding code is fixed and invariable. This is a further advantage because, unlike general purpose XML compressors, it is unnecessary to include a header containing the decoding table within the compressed file.

(2) *Attribute-values packing.* Most recurrent words are identified in the previously distinguished data section. They will be coded with a 16-bit sequence. A header for the compressed file is thus built, containing correspondences between each text string and the related 16-bit code. It is dynamically created and exclusively belongs to a specific DIG document instance. The provided header will be exploited in the decompression phase.

Assigned codes differ by their second byte, because the first octet is adopted as padding in order to distinguish the attribute value coding from regular ASCII characters. This second compression stage allows to obtain a further size saving, particularly in ontologies with very frequent concepts and roles. On the other hand, the use of the header could compromise compression performances for short files, as the space consumption for the header itself reduces savings obtained with compression. Hence the encoding of all the string values of a DIG file without any a-priori distinction has to be definitely avoided.

A correct compression procedure should properly take into account both the length of an attribute string and its number of occurrences within the file. The minimum length of strings to encode can be trivially calculated by comparing the size consumption needed to store *string–code* correspondences and the saving obtained with the encoding: in the proposed approach only text attributes with a length of at least three characters will be encoded.

Furthermore we must evaluate the number of occurrences of each attribute i (from now on $nr\_occurences_i$). We set an optimal minimum value we call $nr\_occurences\_min$ and we will encode only i attribute values where results $nr\_occurences_i > nr\_occurences\_min$. We have performed statistical evaluations trying the compression of 72 sample DIG documents and evaluating obtained compression rates varying $nr\_occurences\_min$. Results show that the best compression rates are produced by $nr\_occurences\_min$ values within the range [2–8] with an average of 4.03 and a standard deviation in the range [0–0.3]. Thus we set $nr\_occurences\_min = 4$, so only attribute strings with at least three characters and recurring at least four times will be encoded.

(3) *zlib packing.* Finally *zlib* library based on the *Ziv-Lempel* compression algorithm [13] is exploited to apply an eventual third compression level, optimizing the overall result. Ziv-Lempel algorithm does not perform particularly well when compressing a partially coded input (it is difficult to find more occurrences of the same character sequence). The use of *zlib*, however, resulted useful in our approach specially for large files, where it produces the compression of words excluded by previous compression steps and of the file header.

4 Evaluation

A generic evaluation of the proposed approach has been carried out taking into account two different aspects. First of all, performances of the compression/decompression algorithms have been investigated and furthermore reading and decompression times of software simulated semantic-enhanced RFID tags were evaluated, in order to provide an initial assessment of the impact that our approach may have on RFID systems performances.

Regarding the compression and decompression performances three fundamental parameters have been estimated: **(1)** *compression rate*, **(2)** *turnaround time*, **(3)** *memory exploitation.* Two tools were developed in C language implementing our compression and decompression algorithms. They were named *DIG Compressor* and *DIG Decompressor*, respectively. Currently, Windows and Linux platforms are supported, leveraging the freely available *zlib* compression library. Tests for compression rate and running time were performed using: (1) a PC equipped with an Intel Pentium 4 CPU (3.06 GHz clock frequency), 512 MB RAM at 266 MHz and Windows XP operating system; (2) a PC equipped with a Pentium M CPU (2.00 GHz clock frequency) and 1 GB RAM at 533 MHz, running Gentoo GNU/Linux with 2.6.19 kernel version and *Valgrind* [6] profiling toolkit.

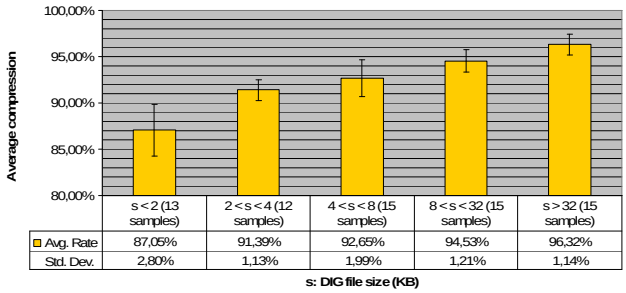

Fig. 2. Obtained compression rates

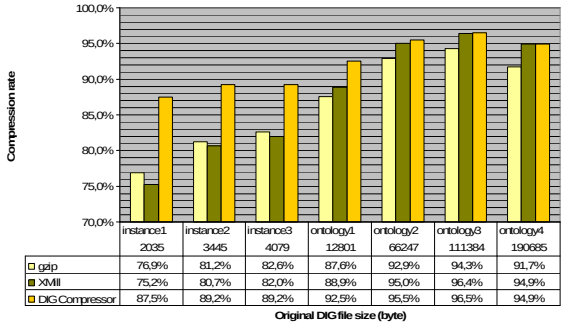

Fig. 3. Performance comparison on a representative sample of DIG documents – **Compression rate**

Firstly, compression rates achieved by the proposed algorithm were considered. We carried out tests with 70 DIG documents of various size. Our aim was to evaluate compression rates for both smaller instance descriptions and larger ontologies. Figure 2 shows average compression rates and standard deviations for different size ranges of DIG input data. Overall average compression rate is $92.58 \pm 3.58\%$. As expected, higher compression rates were achieved for larger documents. Even for very short DIG files (less than 2 kB), however, average compression rate is $87.05 \pm 2.80\%$, which is surely satisfactory for our purposes.

A comparative evaluation was carried out using as benchmarks the general purpose XML compressor *XMill* [5] and *gzip* (http://www.gzip.org/) generic compressor. Testing the compression rate, the proposed system allowed to obtain smallest resulting files, as shown in Figure 3. For each DIG file, the original size in bytes is reported. It should be noticed our algorithm performed significantly better for small DIG documents. This result is very encouraging, since in our mobile scenarios we usually deal with small XML documents representing the annotations of available resources.

In order to evaluate the turnaround time, each test was run 10 times consecutively, and the average of the last 8 runs was taken. Results are presented in Figure 4. It can be noted that DIG Compressor has higher turnaround times

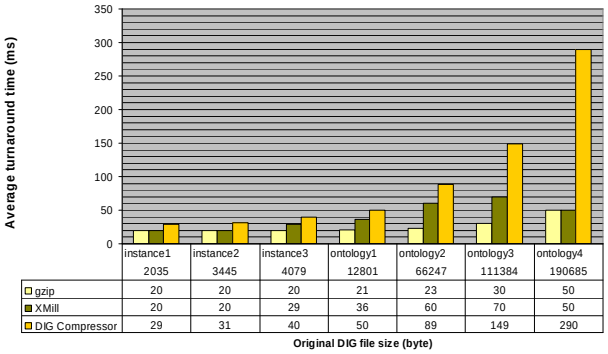

Fig. 4. Performance comparison on a representative sample of DIG documents – **Turnaround time**

than other tools, though absolute values are still quite acceptable. Such a result suggests we need further optimizations for execution speed.

Finally, memory usage analysis was performed using *Massif* tool of *Valgrind* debugging and profiling toolkit. Massif measures stack and heap memory profile throughout the life of a process. For our comparison, only the memory occupancy peak was considered. DIG Compressor memory usage is only slightly higher than the one of *gzip*, with high correlation ($r = 0.96$) between the two value sets. This result could be expected, since our algorithm relies on Ziv-Lempel compression in its last phase. On the contrary, *XMill* showed a more erratic behavior. Outcomes can be reputed as encouraging because memory-efficient implementations of *zlib* library are currently available for all major mobile platforms.

A thorough experimental evaluation of all aspects of framework performance requires its complete implementation into a testbed with real semantic-enabled RFID devices. That would only be possible through partnership agreements with device manufacturers/integrators, that we are currently pursuing. At this stage, a prototypical semantic-enhanced RFID infrastructure has been simulated by extending IBM WebSphere RFID Tracking Kit middleware solution for RFID applications. RFID simulations and tests have been performed on that testbed, which is deployed on a laptop PC equipped with Pentium M processor (2.00 GHz clock frequency), 1 GB RAM at 533 MHz and Microsoft Windows XP operating system. Compressed semantic annotations of 40 different products were used. Their average size is 266 ± 104 B (range 91-440 B). Simulated RFID data access from each tagged item was repeated 100 times, recording the sum of reading and decompression time. For each item the mean value was then considered.

Results are reported in Figure 5. Average access time is 2.02 ± 0.36 ms, corresponding to a theoretical tag read rate of approximately 500 tags/s. Since tests were run on a software-simulated RFID platform, exact numerical values are not significant as their order of magnitude. The latter can be sensibly compared to performance of RFID systems compliant with EPCglobal standards for Class 1 Generation 2 UHF RFID systems.

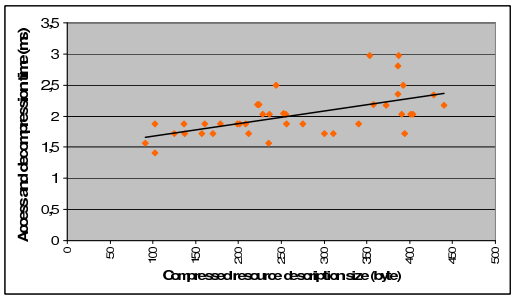

Fig. 5. Simulated RFID tag reading and decompression time for 40 resource descriptions. Regression line is plotted.

It is known that RFID system performance in the field highly depends on the particular application, environmental conditions (electromagnetic noise, RFID reader density) and local regulations affecting the available bandwidth. Early simulations and tests carried out by independent research laboratories estimated reading rates in a range of 7-100 tags/s in typical conditions [8, 4]. Our simulation results are quite above these with such data. Hence, a very preliminary evidence that adoption of compressed semantic annotations on RFID tags does not impair performances in the field w.r.t. traditional ones is so provided. The latter, in turn, will not suffer any direct performance degradation from the newly introduced features, as they will read the EPC only. Finally, the access time showed a moderate positive correlation ($r = 0.60$) with annotation size. This may suggest that structure of a DIG annotation (*i.e.*, exploited logic constructors and frequency of attribute names) has also an impact over the decompression.

5 Conclusions

Our approach can support a range of use cases, involving different stakeholders in each step of a product life cycle. During product manufacturing and distribution, a wide-area support network interconnecting commercial partners is not strictly needed. This is a significant innovation w.r.t. common RFID supply chain management solutions. By means of their semantic-enabled RFID tags, products are always accompanied by structured and rich description of their characteristics, endowed with unambiguous and machine-understandable semantics. Beyond improved traceability, a semantic-based approach may provide unique value-added capabilities. In particular, query flexibility and expressiveness are much greater than both keyword-based information retrieval and standard service/resource discovery protocols, which support code-based exact matches only. Non-exact match types are prevalent in real scenarios, involving a large number of resources by many different sources. Semantic-based techniques can support non-exact matches and ranking, further providing means for results explanation. This enables an effective query refinement process and can strengthen user trust in the discovery facility.

Bibliography

[1] S. Bechhofer, R. Möller, and P. Crowther. The DIG Description Logic Interface. In *Proceedings of the 16th International Workshop on Description Logics (DL'03)*, volume 81 of *CEUR Workshop Proceedings*, September 2003.

[2] R. Bridgelall. Enabling Mobile Commerce Through Pervasive Communications with Ubiquitous RF Tags. *IEEE Wireless Communications and Networking, (WCNC)*, 3:2041–2046, March 2003.

[3] EPCglobal Ratified Specification. Object Naming Service (ONS - ver. 1.0). http://www.epcglobalinc.org, October, 4, 2005.

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

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

[6] N. Nethercote and J. Seward. Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation. In *Conference on Programming Language Design and Implementation - PLDI 07*. ACM SIGPLAN, June 2007.

[7] J. Rabin and C. McCathieNevile. Mobile Web Best Practices 1.0. *W3C Proposed Recommendation*, 2006.

[8] K. Ramakrishnan and D. Deavours. Performance benchmarks for passive UHF RFID tags. In *Proceedings of the 13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems*, 2006.

[9] K. Römer, T. Schoch, F. Mattern, and T. Dübendorfer. Smart Identification Frameworks for Ubiquitous Computing Applications. *Wireless Networks*, 10(6):689–700, November 2004.

[10] M. Ruta, T. Di Noia, E. Di Sciascio, and F. Donini. Semantic-Enhanced Bluetooth Discovery Protocol for M-Commerce Applications. *International Journal of Web and Grid Services*, 2(4):424–452, 2006.

[11] K. Traub, G. Allgair, H. Barthel, L. Bustein, J. Garrett, B. Hogan, B. Rodrigues, S. Sarma, J. Schmidt, C. Schramek, R. Stewart, and K. Suen. EPCglobal Architecture Framework. Technical report, EPCglobal, July 2005.

[12] R. Weinstein. Rfid: A technical overview and its application to the enterprise. *IT Professional*, 07(3):27–33, 2005.

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