An efficient data compression algorithm for semantic-based ubiquitous computing applications

Michele Ruta
Politecnico di Bari
via Re David 200
I-70125 Bari, Italy
m.ruta@poliba.it

Tommaso Di Noia
Politecnico di Bari
via Re David 200
I-70125 Bari, Italy
t.dinoia@poliba.it

Eugenio Di Sciascio
Politecnico di Bari
via Re David 200
I-70125 Bari, Italy
disciascio@poliba.it

Floriano Scioscia
Politecnico di Bari
via Re David 200
I-70125 Bari, Italy
f.scioscia@poliba.it

Abstract

We present an efficient compression algorithm specifically devised and implemented to reduce size of document instances expressed in various ontological languages, and specifically targeted at DIG 1.1 syntax. Motivation derives from a widespread exploitation of Knowledge Representation formalisms in innovative ubiquitous computing contexts. In carried out tests, the compressor we present showed better compression rates than currently available general-purpose XML compression tools.

1 Introduction

Technological innovation in mobile computing and communication is steadily growing. Nowadays a lot of personal mobile devices (such as smartphones and PDAs) are endowed with high processing power, extensible software platforms and support for multiple wireless connectivity. Basically, network technologies for ubiquitous computing can be divided into *infrastructure-based* and *ad-hoc*. Mobile Ad-hoc NETworks (MANETs) are characterized by shorter communication range, higher connection volatility, lower data rates and energy consumption w.r.t. infrastructure-based ones [18].

As devices become more powerful, the request for "smarter", more advanced applications and services arises. In particular, context-awareness and customization according to user preferences are highly desired features in ubiquitous computing scenarios. *Context* consists in any implicit or explicit information which can be used by a system to automatically characterize the status of a user within the environment [5]. Emerging ubiquitous applications aim to disseminate knowledge into the environment surrounding people. Such information should be automatically extracted and processed by mobile devices, in order to better support the current activity of a user and satisfy her needs.

Several technologies are emerging as possible means to fill the gap between physical and digital world (for example *Radio Frequency IDentification* (RFID) and *Wireless sensor networks* (WSNs)).

RFID systems allow an object identification by means of electronic transponders (*tags*) attached to items, which are detected by reader devices and identified through a unique Electronic Product Code (EPC). Worldwide industry leaders and research centers, under the guide of EPCglobal consortium [6], are currently engaged in a joint effort to develop a set of technological standards for a global platform aiming at the identification and tracking of commercial products during their entire lifecycle (manufacturing, distribution and sales).

WSNs, on the other hand, allow to monitor environmental parameters of interest. Latest WSN generations support both queries and automatic alerts when events defined by application criteria occur [7]. Both these technologies are characterized by relatively large numbers of unobtrusive, inexpensive and disposable micro-devices, disseminated in a given environment. Due to cost and power constraints, data transfer bandwidths and on-board memory and processing capabilities of individual nodes are still extremely limited.

In mobile environments, users search and use services/resources exploiting basic wireless connectivity and elementary discovery protocols. Current discovery paradigms are based on a "one to one" matching of resource attributes which results largely unsuitable for advanced applications. Ideas and technologies borrowed from the Semantic Web vision may allow to overcome these limitations. Each available resource in the semantic-enabled Web vision is annotated, using RDF [14], w.r.t. an OWL ontology [19]. Both RDF and OWL are defined through XML Schema [20]. Furthermore there is a close relationship between the OWL-DL subset of OWL and Description Logics (DLs) [1] semantics, which allows the use of DL-based rea-

0-7695-2993-3/07 $25.00 © 2007 IEEE
DOI 10.1109/UBICOMM.2007.33

soners in order to infer new information from the one stated in the semantically annotated descriptions. The communication with state-of-the-art DL reasoners can be performed via an HTTP interface developed by the Description logics Implementation Group (DIG) [2].

A known drawback of XML formats (such as DIG) is their verbosity. Usually it is not a concern for Internet-based applications (because link bandwidth and host storage capacity are enough for most practical purposes), but surely reduces efficiency of data storage and communication in mobile environments. Adaptation of ideas and techniques from the Semantic Web vision to ubiquitous scenarios requires to cope with the limited storage and computational capabilities of mobile and embedded devices and with reduced bandwidth provided by wireless links. From this point of view, the first issue is in downscaling the large amount of transiting data. Compression techniques become essential in order to enable storage and transmission of semantically annotated information on tiny mobile devices such as RFID tags or wireless sensors. Moreover, the benefits of compression will apply to the whole ubiquitous computing environment, as decreasing data size means shorter communication delays, efficient usage of bandwidth and reduced battery drain for mobile devices in a MANET.

In this paper, we present a novel efficient XML compression algorithm. It is currently targeted at DIG 1.1 document instances: structural elements of the XML Schema for DIG formalism have been encoded in a compact way. In our tests, better compression rates have been achieved w.r.t. currently available general-purpose and XML-specific compressors.

The remaining of the paper is organized as follows. Section 2 illustrates motivation of the approach. In Section 3 we comment on relevant related work. The proposed compression algorithm is described in Section 4. Section 5 reports a performance evaluation. Conclusions and future work are outlined in Section 6.

2 Framework and Motivation

In [15] a framework for resource discovery in Bluetooth-based MANETs was proposed, exploiting ontology-based Knowledge Representation techniques. A –backward compatile– micro-layer of semantic capabilities was integrated into Bluetooth Service Discovery Protocol (SDP). This approach allows to exploit semantic-enhanced discovery techniques to support approximate matches. A similar vision can be projected onto objects populating the world surrounding us, as devised in [17]. By extending the above-mentioned technologies to support storage and transmission of descriptions expressed in suitable Knowledge Representation (KR) formalisms, everyday objects could automatically expose relevant information about themselves to

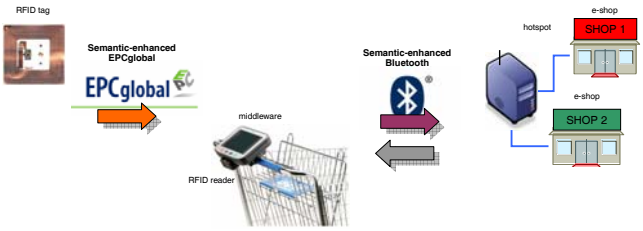

Figure 1. Semantic-based object discovery framework

nearby mobile devices. In this manner a fully integrated context representation could be enabled.

In a companion paper presented at this same venue [16] the above vision has been expanded in order to enable object discovery capabilities through semantic-enhanced RFID for m-commerce purposes. Backward-compatible extension to EPCglobal specifications for tag standards and data exchange protocol has been devised and designed. Furthermore, a prototypical system has been implemented, oriented to ubiquitous commerce scenarios. Its basic structure is depicted in Figure 1 and briefly summarized in what follows. An object can expose its semantic annotation, stored in the RFID tag it is associated with. In a mobile marketplace context, when a user picks an item, the choice is detected and considered as an implicit request to interact with the system. A RFID reader scanning the description of the selected good thus enables a discovery phase to find further resources, either similar or to be combined with the selected one. Semantic based Bluetooth SDP and non standard inference services are exploited to discover and return the best matching resources within the marketplace.

During the implementation and evaluation of the above framework, we noticed DIG syntax is more compact than OWL [17], but experimental results and technological evidence show it is still too verbose for a significant application in ubiquitous computing. If we want to really induce an unobtrusive decentralization of ontology-based mobile systems for making them really pervasive, annotations have to be compacted and unpacked in an efficient and reliable fashion. In this way, metadata about smart objects (*i.e.*, wireless sensors, tagged goods, embedded micro-devices) can carry a description about provided features and managed functionalities and no concentrated knowledge/data bases are required. In those cases a strategy to compress semantic annotations is definitely required.

Also ontologies expressed in DIG syntax can benefit from a compact encoding. It decreases both data occupancy as well as data transfer –and thus time and power consumption– in communication sessions between a reasoner and a client agent. This can be particularly useful for mobile scenarios, but also for applications originally de-

vised for wired networks.

3 Related Work

gzip [8] (along with its library version **zlib** [22]) is a very popular universal compression tool. It is based on a variant of the LZ77 Ziv-Lempel algorithm [21]. It basically searches for string sequences recurring at least one time within the file to be compressed. Furthermore it substitutes the second occurrence with the pair: distance (in byte) from the first exemplar of the sequence and length of the sequence itself.

XMill [12] is an efficient general purpose XML compressor. Its approach is based upon the separation of XML content into different *containers*, which are stored sequentially in the output file. The first container is always used for encoding document structure. Another one lists XML tag and attribute names and is compressed using *zlib*. For each XML tag type in the source document, one more container is created. Each of these data containers is compressed by a specialized module. XMill provides optimized compression modules for several basic data types (such as text, integers and dates), as well as an API to add custom modules for application-specific data types. XMill performances are better than generic compressors for medium and large XML documents. For small files (up to 20 KB approximately), however, tests showed lower compression rates than generic algorithms such as *gzip*. Since each container is compressed separately, in those cases efficiency of compression algorithms is penalized by the small size of containers. For our intended applications, it is important to achieve high compression rates even for short documents as DIG annotations stored within tagged objects, sensors or micro-devices. A different approach w.r.t. the one of XMill is therefore required. **Flash XML Compressor** is another example of XML-specific encoding tool: its compression rates, however, are much lower than XMill and practically inadequate to the purposes stated in Section 2.

In [10] a syntactic approach is pursued. A DTD or XML Schema is treated as a *Dictionary Grammar*, which is a variant of a Context Free Grammar. A parser generator was developed, that builds a parser for the DTD or XML Schema of the document to be compressed. The XML document is then parsed and symbols are encoded using *partial prediction matching* (PPM), an adaptive coding technique [3]. Authors showed that grammar-driven PPM is more efficient than other PPM-based XML encoding techniques. Compression rates are higher than XMill, though PPM-based compressors are generally slower. A further advantage of that approach is in allowing the compression of XML data streams. That feature can be useful in network applications. Unfortunately we could not compare proposed algorithm with that tool, since it is not publicly available. The XML corpus used in [10] for benchmarking does not contain documents in DIG formalism.

4 Algorithm details

The proposed compression algorithm is specifically oriented to the packing of the novel standard DIG 1.1 format [2]. Each XML file has a specific DTD. In particular a DIG format can together contain 40 tags at most. A DIG document is an XML document exposing specific characteristics. That is, no value is set for any tag; the value of tag attributes is within a well defined finite set of values. An example of DIG syntax is reported in Figure 2a.

A basic distinction among various encoding techniques is in *fixed length* and *variable length* algorithms [9]. In the first case, having a specified alphabet, a fixed bit number is used to code each symbol: in particular we need $n = log_2 k$ bits, if alphabet has k symbols. A DIG file is coded by means of `ISO 8859-1` or `UTF-8` encoding. In particular each allowed character can be associated to 1 byte (special characters needing more than 1 byte in `UTF-8` do not belong to the symbol set of a DIG). Hence, in order to obtain a good compression rate, we must recur to a variable length coding algorithm: in this case the most efficient algorithm is the Huffman one [11, 4]. It requires to have a list containing the correspondences between each symbol and the corresponding bit sequence. The list obviously varies according to the document. Although Huffman algorithm could appear a good choice to compress an ontology in DIG syntax, it does not work well with short semantically annotated DIG descriptions as the ones referred to resource metadata annotation. Basically a resource description is usually few hundred bytes long and then the Huffman compression is sometimes inadequate because a description could be smaller than the decode list itself.

We propose a different and simple DIG compression solution particularly suitable for pervasive applications whose structure is shown in Figure 3. 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. A typical DIG file is mostly composed of empty XML 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 more effective compression rate. Data structures are basically XML elements with possible related attributes whereas data simply are attribute values. Recall that data structures in DIG syntax are fixed and well defined by means of the DIG XML Schema, whereas data are different from document to document. XML elements

```
<?xml version="1.0" encoding="UTF-8"?>
<tells xmlns="http://dl.kr.org/dig/2003/02/lang">
  <definidividual name="*dual_display_phone"/>
  <instanceof>
    <indidividual name="*dual_display_phone"/>
    <and>
      <catom name="mobile_phone"/>
      <atleast num="2">
        <ratom name="has_display"/>
        <top/>
      </atleast>
      <atmost num="2">
        <ratom name="has_display"/>
        <top/>
      </atmost>
      <all>
        <ratom name="has_display"/>
        <catom name="color_display"/>
      </all>
      <some>
        <ratom name="has_display"/>
        <catom name="QCIF_LCD_display"/>
      </some>
      <some>
        <ratom name="has_display"/>
        <catom name="QVGA_LCD_display"/>
      </some>
    </and>
  </instanceof>
</tells>
```

(a) Original document (791 B)

```
be cb dual_display_phone
b5 c6 mobile_phone
c1 2 cc has_display
a8 c2 2 cc has_display
c4 a9 b6 cc has_display
c6 color_display
a7 c0 cc has_display
c6 QCIF_LCD_display
b2 c0 cc has_display
c6 QVGA_LCD_display
b2 a6 b0 b3
```

(b) Stage 1 result (178 B)

```
has_display ff fe
be cb dual_display_phone
b5 c6 mobile_phone
c1 2 cc 01
a8 c2 2 cc 01
c4 a9 b6 cc 01
c6 color_display
a7 c0 cc 01
c6 QCIF_LCD_display
b2 c0 cc 01
c6 QVGA_LCD_display
b2 a6 b0 b3
```

(c) Stage 2 result (143 B)

Figure 2. Example of DIG document at different stages of the proposed algorithm

are coded by associating an unambiguous 8-bit code to each structure in a static fashion. Consider that DIG files adopt an encoding which exploits 1 byte for each character: so an early size saving is performed. The association between XML structures and corresponding code is fixed and invariable. This is a further benefit because it is not necessary to integrate within the compressed file a header which contains the decoding table as in the general purpose XML compressors. Figure 2b shows the output of this compression step for the example DIG document (for a better readability, byte encoding data structures are reported in hexadecimal notation and whitespace has been added by hand)

(2) *Attribute-values packing*. In order to pack the attribute values, in the proposed approach a further phase is introduced. Most recurrent words are identified in the previously distinguished data section. They will be encoded with a 16-bit sequence. This second compression stage allows to obtain a further size saving –especially in ontologies– for concepts and roles particularly recurrent. The second packing phase needs to build and maintain a header of the compressed file 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 steps. Notice that assigned codes differ between them for the second byte because the first octet is adopted as padding in order to distinguish the attribute value coding from the ASCII one. The use of this header could compromise compression performances for short files: recall that the size consumption for the header reduces saving 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. Care has to be paid in choosing attribute-value strings to encode. A correct compression procedure should properly take into account two variables. First of all the length of an attribute string and furthermore its number of occurrences within the file. The minimum length of strings to encode can be trivially established by comparing the size consumption needed to store correspondences *string–code* 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, in order to establish what attribute values (among remaining ones) have to be coded, we must evaluate the number of occurrences of each attribute i (from now on $nr\_occurences_i$). We fix a minimum optimum value $nr\_occurences\_min$ and we will encode only i attribute values where $nr\_occurences_i > nr\_occurences\_min$. We have performed statistical evaluations trying the compression of 72 sample ontologies and evaluating obtained compression rate varying $nr\_occurences\_min$. The best compression rates were 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]. In the proposed approach we set $nr\_occurences\_min = 4$, so we will encode only attribute strings with at least three characters recurring at least four times. In the example document, `has_display` is the only attrbitue value encoded in the file header, as shown in Figure 2c ($\texttt{ff}_\texttt{h}$ delimits header entries, while $\texttt{fe}_\texttt{h}$ marks the end of header).

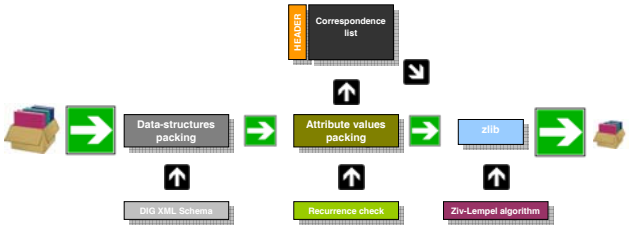

Figure 3. Structure of the proposed DIG compression tool

(3) *zlib packing*. The third and final compression step exploits the *zlib* library. Words not encoded will be successively coded by means of the Huffman algorithm [11]. Although the *zlib* algorithm does not work particularly well when it has to compress a partially encoded input (it is difficult to find more occurrences of the same character se-

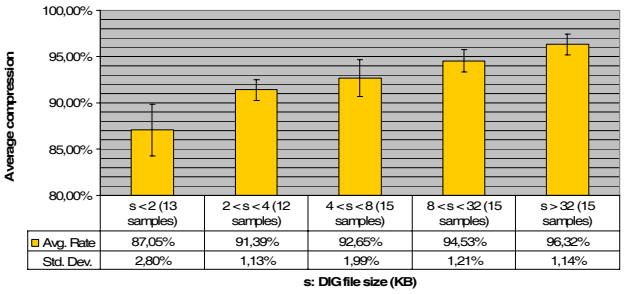

Figure 4. Obtained compression rates

quence), the use of *zlib* in our approach resulted however useful especially for large files, where it produces the compression of words excluded by the previous steps and of the file header.

5 Performance Evaluation

Performance evaluation of the proposed algorithm ha been carried out estimating three fundamental parameter **(1)** *compression rate*, **(2)** *turnaround time*, **(3)** *memory u age*. Two tools were developed in C language implemen ing our compression and decompression algorithms. The were named *DIG Compressor* and *DIG Decompressor*, re spectively. Currently, Windows and Linux platforms ar supported, leveraging the freely available *zlib* compressio library. Tests for compression rate and running time wer performed using: (1) a PC equipped with a Intel Pentiu 4 CPU (3.06 GHz clock frequency), 512 MB RAM at 26 MHz and Windows XP operating system; (2) a PC runnin Gentoo GNU/Linux with 2.6.19 kernel version and *Valgrin* [13] profiling toolkit. This second PC was equipped wit a Pentium M CPU (2.00 GHz clock frequency) and 1 G RAM at 533 MHz.

Firstly, compression rates achieved by the proposed algorithm were considered, in order to assess whether our approach is adequate to the purposes outlined above. The framework was tested with 70 DIG documents of various size. Our aim was to evaluate compression rates for both smaller instance descriptions and larger ontologies. Figure 4 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 the general purpose XML compressor *XMill* [12] and *gzip* [8] generic compressor as benchmarks. Testing the compression rate, the proposed system allowed to obtain smallest

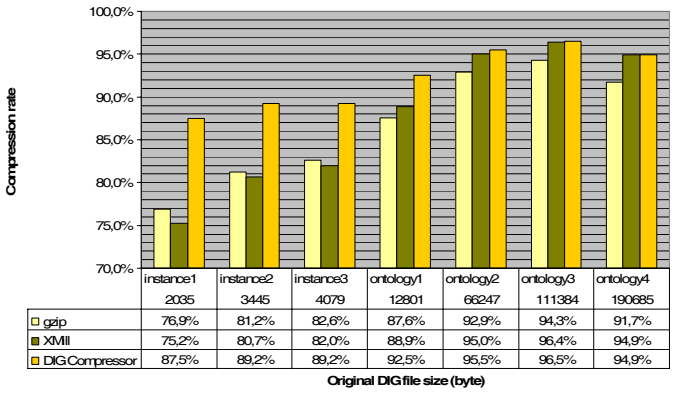

Figure 5. Performance comparison – Compression rate

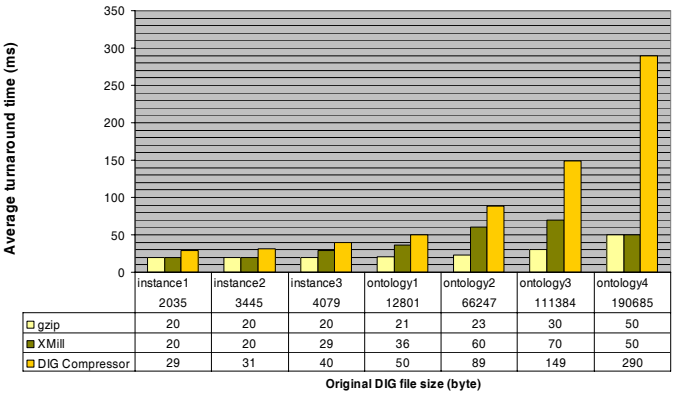

Figure 6. Performance comparison – Turnaround time

resulting files, as shown in Figure 5. For each DIG file, original size (in bytes) is reported. Our algorithm performed significantly better for small DIG documents. This result is very encouraging, since in our mobile scenarios we usually deal with short XML annotations of available resources.

In order to evaluate turnaround time, each test was run 10 times consecutively, and the average of the last 8 runs was taken. Results are presented in Figure 6. It can be noted that DIG Compressor has higher turnaround times than other tools, though absolute values are still quite acceptable. Such an outcome suggests we should put further work into optimizing our implementation 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. Results are reported in Table 5. DIG Compressor memory usage is only slightly higher than the one of *gzip*, with high correlation ($r = 0.96$)

Table 1. Performance comparison – Memory usage peak (kB)

DIG document	Original size (B)	gzip	XMill	DigCompressor
Playstation_3_Slim.dig	2035	220	2700	290
Kodak_P880_camera.dig	3445	200	4500	250
Asus_A3FP_Notebook.dig	4079	200	6500	250
toy_ontology.dig	12801	200	4000	240
rent_ontology.dig	66247	200	6500	250
clothing_ontology.dig	111384	202	4500	250
electronic_products_ontology.dig	190685	210	4000	260

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.

6 Conclusion and Future Work

We have devised, implemented and tested a compression algorithm for XML documents in DIG syntax. The study is motivated by the need for better adapting Semantic Web technologies to pervasive environments. Our DIG compressor tool showed high efficiency in terms of compression rates and computational requirements. DIG data compression can speed up communications among nodes both for wired and wireless networks. Implemented tool results suitable for mobile applications, since it achieves high compression rates even for short DIG annotations of resource instance descriptions. With respect to *XMill* XML compressor and *gzip* generic compressor, our tool showed the highest compression rate in all performed tests. Processing times were comparable for documents up to 80 kB. Future work comprises porting the compression tool to mobile platforms. Java-enabled phones and Linux-based wireless PDAs are the main targets. Evaluation of time and memory requirements on mobile devices is important to assess the feasibility of our approach. A further goal is to extend our solution to a broader range of XML documents. This may be achieved by dynamically adapting our algorithm to the DTD or XML Schema of the document to be compressed. Moreover, by applying the same procedure when decompressing, the original document could be restored with no information loss. It would be particularly useful to efficiently encode other widespread DL-based formalisms such as OWL, which are even more verbose than DIG. A preliminary investigation is currently being performed in this direction.

References

[1] F. Baader, I. Horrocks, and U. Sattler. Description Logics as Ontology Languages for the Semantic Web. *Lecture Notes in Artificial Intelligence*, Springer, 2003.

[2] 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.

[3] J. Cleary and I. Witten. Data Compression Using Adaptive Coding and Partial String Matching. *IEEE Transactions on Communications*, 32(4):396–402, April 1984.

[4] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., New York, 1991.

[5] A. Dey. *Providing Architectural Support for Building Context-Aware Applications*. PhD thesis, Georgia Institute of Technology, 2000.

[6] EPCglobal. http://www.epcglobalinc.org.

[7] D. Graanin, W. Eltoweissy, A. Wadaa, and L. DaSilva. A service-centric model for wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(6):1159–1166, June 2005.

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

[9] R. Hamming. *Coding and Information Theory*. Prentice Hall, 1986.

[10] S. Harrusi, A. Averbuch, and A. Yehudai. XML syntax conscious compression. In *Data Compression Conference (DCC2006)*, March 2006.

[11] D. Huffman. A method for the Construction of Minimum Redundancy Codes. In *IRE*, pages 1098–1101, September 1952.

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

[13] 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.

[14] RDF Primer-W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-primer/.

[15] 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.

[16] M. Ruta, T. Di Noia, E. Di Sciascio, and F. Scioscia. Integrating Radio Frequency Object Discovery and Bluetooth for Semantic-based M–commerce. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2007)*, 2007. To appear.

[17] M. Ruta, T. D. Noia, E. D. Sciascio, and F. Scioscia. If Objects Could Talk: Semantic-enhanced Radio-Frequency IDentification. In *The First International Workshop on RFID Technology Concepts, Applications, Challenges (IWRT 2007)*, pages 25–34. INSTICC Press, 2007.

[18] C. Toh. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. *IEEE Communications Magazine*, 39(6):138–147, June 2001.

[19] W3C Recommendation. OWL Web Ontology Language. www.w3.org/TR/owl-features/, 2004.

[20] XML Schema. http://www.w3.org/XML/Schema.

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

[22] ZLIB Library. http://www.zlib.net.