Sensor Mobile Enablement (SME): a Light-Weight Standard for Opportunistic Sensing Services

Valerio Arnaboldi*, Marco Conti*, Franca Delmastro*, Giovanni Minutiello* and Laura Ricci†
*IIT Institute - National Research Council of Italy - via G. Moruzzi, 1 - 56124 Pisa, Italy
Email: firstname.lastname@iit.cnr.it
†Department of Computer Science, University of Pisa - Largo Bruno Pontecorvo 3 - 56127 Pisa, Italy
Email: ricci@di.unipi.it

***Abstract*—The proliferation of smartphones as complex sensing systems represents today the basis to further stimulate the active participation of mobile users in opportunistic sensing services. However, single sensing devices (either independent network components or integrated in more powerful devices) generally present different capabilities and implement proprietary standards. This highlights the necessity of defining a common standard for sensing data encoding in order to guarantee the interoperability of heterogeneous devices and personal mobile systems. In this paper we present *Sensor Mobile Enablement (SME)*, a light-weight standard for efficiently identifying, coding and decoding heterogeneous sensing information on mobile devices. After a detailed analysis of SME features and advantages, we present its performances derived from real experiments on Android smartphones. Results highlight that SME does not heavily impact on mobile system's performances while efficiently supporting opportunistic sensing services.**

I. Introduction

Opportunistic and participatory sensing represent today one of the hottest research topic in pervasive and ubiquitous computing due to the widespread diffusion of personal mobile devices that make the user a sort of human mobile sensor, exploring the surrounding environment. The research trend started with the notion of *participatory sensing* [1], directly requiring the user interaction to exploit (mainly) the smartphone's audio/video signals as sensing information in order to witness particular events. This information was then stored on centralized servers to be shared with other interested users. Then, research evolved into *phone-embedded sensing* services [2], in which the major issue was how to use phone sensors to correctly infer the activity of a user, his/her social interactions and his/her personal behavior in the daily life. In both cases the smartphone represented a complex sensing instrument designed to interact with the single user, collect data and upload it for remote sharing with others.

Opportunistic sensing enlarges this vision by allowing the cooperation of multiple personal mobile devices through resource and data sharing mechanisms based on opportunistic communications [3], not necessarily requiring the explicit interaction with the user. It belongs to a wider research field called *opportunistic computing* [4], in which all the smartphone's resources (i.e., computation, storage, available connectivity in addition to embedded sensors) can be requested for a cooperative use by remote devices in order to improve features of mobile services and applications. In this scenario the users' mobility is fundamental both as source of information (as smartphone carrier) and as generator of devices' communication opportunities (due to users and devices encounters). However, it is also one of the major causes of the high dynamism that characterizes mobile systems together with the heterogeneity of involved devices. In order to overcome the limitation of devices' heterogeneity, with particular attention to both phone-embedded and external sensors, in this paper we propose the specification and implementation of light-weight standard models and related software tools for identifying, encoding and processing heterogeneous sensing information on mobile devices. The idea arises from the technical limitations of using the existing reference standard for sensor web services (i.e., OCG Sensor Web Enablement (SWE) [5]) on mobile devices, and from the need to guarantee the interoperability of existing infrastructured solutions with the innovative approach of participatory and opportunistic sensing services. The proposed standard, called *Sensor Mobile Enablement* (SME) presents multiple advantages: (i) it provides the definition of general data structures and XML codification of all information related to phone-embedded sensors to support the development of opportunistic and participatory sensing services (not included in SWE); (ii) it guarantees efficient processing of XML files on mobile devices (related to encoding information of both phone-embedded sensors and SWE data); (iii) it guarantees interoperability of opportunistic sensing services with standard SWE services. In this work we present an implementation of SME for Android OS (version 4.1.1) as Java library. However, several implementations can be provided for different mobile operating systems.

After a deep analysis of adopting SME standard in mobile sensing applications, we decided to support it also in our middleware platform CAMEO [6], [7], designed for the efficient development of context- and social-aware applications over opportunistic networks. In fact, the efficient management of sensing data represents a common requirement for several mobile applications (e.g., personal and environmental monitoring, mobile social networks) in which developers tend to exploit all the available information to improve users' experience and applications' performances. CAMEO support of SME standard provides additional sensing features to context-aware mobile applications and the possibility to interact with both external sensors and remote sensor web services adopting SWE standard.

The paper is organized as follows: Section II presents

978-1-4244-9529-0/13/$31.00 ©2013IEEE

related work in this field and technical motivations of our new solution. Section III presents the main functionalities of SME standard and the software library for android OS and Section IV highlights the advantages of its use in CAMEO. Section V shows performances analysis of SME management through real experiments on Android smartphones.

II. Background and Motivations

Due to the current proliferation of heterogeneous sensing devices, generally characterized by proprietary standards for data encoding and transmission, the Open Geospatial Consortium (OCG) decided to define and implement SWE framework [5] as a set of interoperability interfaces and metadata encodings (based on XML schemas) that enable real time integration of sensing information into a server web infrastructure. SWE defines three standards for encoding, respectively, sensors' descriptions (Sensor-ML), observations and measurements (O&M) and transducers (TML). In addition, it provides four web service interfaces for accessing related information. Generally, the most used is Sensor Observations Service (SOS) interface that enables client applications to access observations and sensor system information stored on remote web servers. SWE relies on the general assumption that each sensor is a web-connected device and all sensed data must be remotely managed through a web service. To this aim, SWE defines sensors' discovery procedures, processing and correlations of sensor observations in a completely centralized way. SWE is currently becoming the reference standard for remote sensing services and for the emerging paradigm of Internet of Things. In fact, OCG formed a new standard working group called *Sensor Web for IoT* [8] aimed at developing new standards based on Web of Things protocols while also leveraging the existing SWE standard. However, to consider the Web as the only reference scenario does not reflect the actual distributed nature of current mobile systems, especially involving smartphones as complex sensing devices.

Today smartphones are able to produce several sensing information, both as raw sensing data derived from phone-embedded sensors, and application level sensing events, derived from the correlation of multiple sensing data originated by local and/or remote nodes (e.g., wireless proximity information, crowded places). Through opportunistic communications, mobile devices can also share their own resources (e.g., processing and sensing capabilities) extending the use of local sensors also to remote nodes. The personal mobile device of each user is thus able to generate and share useful sensing information not necessarily passing through a web service. However, to maintain the interoperability of opportunistic sensing services with external sensors implementing SWE or with centralized SOS servers, mobile devices must be able to efficiently elaborate SWE data.

Currently, mobile operating systems do not support the efficient elaboration of large XML files, like those defined by SWE, due to limited hardware and software resources. For these reasons, SWE standard cannot be used as-they-are on mobile devices and this is currently representing an open issue in this research area. Limited works on this topic has been presented in literature and they highlight the extreme inefficiency of managing SWE standard on mobile devices. Specifically, [9] revealed that processing SOS XML files on mobile devices can be 30 to 90 times slower than their elaboration on a desktop PC. To overcome this limitation, the same authors proposed to use different file formats to encode SWE messages on mobile devices [10] trying to reduce the overhead of XML processing and their impact on the network usage. Even though some formats (e.g., JSON [11], EXI and EXI with compression [12]) are able to slightly improve the system performances, they introduce an additional overhead due to the local conversion of SWE data into the new format and vice versa. In this paper we show that, in addition to SWE performances limitations, technical limitations prevent SWE implementation on mobile devices, especially on Android OS.

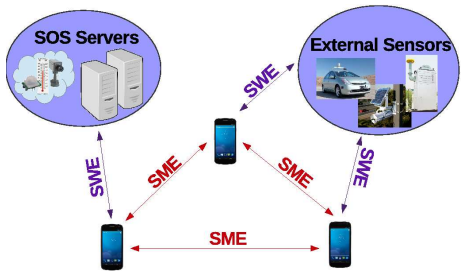

Fig. 1. Smart city scenario involving heterogeneous sensing devices.

A. Technical limitations of SWE on Android

Implementing SWE on mobile devices presents two main issues: (i) the lack of efficient software tools for managing XML files on mobile OS, and (ii) the large size and complexity of SWE XML schemas (most of them including several dependencies). As far as managing and processing XML files, the first approach we investigated is XML file parsing by using standard APIs like DOM (tree-based) [13] and SAX (event-based) [14]. In both cases, applications' developers have to know a priori the specific XML file structure to implement serialization and deserialization procedures, manually selecting relevant tags and values (e.g., sensor descriptions and measurements fields). As an alternative, there are some software tools able to automatically serialize and deserialize XML files into a predefined set of Java classes, reflecting the specific XML file structure (i.e, XML data binding). In this way, the application development is completely independent of the XML files' interpretation and a unique library can be provided for a specific set of XML files. Nevertheless, the high dimension and complexity of SWE XML schemas make really difficult the application of this procedure. In fact, even desktop tools like JAXB [15], XMLBeans [16] and XBinder [17], are not able to provide a complete support for the automatic generation of SWE classes (as witnessed by [18]). Specifically we tested XBinder, which is the only tool able to generate Java classes for Android, and we found that it is able to support only Sensor-ML schemas.

In order to overcome these limitations, we implemented SME as a Java library able to support XML data binding of a light-weight standard compliant with SWE, customized for opportunistic and participatory sensing services. In the

following sections we present SME functionalities both to support independent mobile sensing applications and to provide additional sensing capabilities to CAMEO middleware.

III. Sensor Mobile Enablement (SME)

In order to completely understand SME potentialities in mobile environments, let us consider a *smart city* reference scenario as that depicted in Figure 1. We can envision to have several heterogeneous sensing devices spread around the city, either located at fixed positions (e.g., traffic lights, panels) or in movement (e.g., through vehicular networks, as bus carrying air pollution sensors), thousands of users with their smartphones, equipped with additional embedded sensors, and remote sensor data repositories. In this scenario, smartphones and mobile users can dynamically aggregate in Mobile Social Networks [4], making users able to generate and share useful sensing information through opportunistic communications. At the same time, single smartphones can interact with external sensing devices (e.g., single sensors) and remote sensor web services (e.g., SOS server collecting air pollution data of the city). To allow the interaction of all these heterogeneous sensing systems it is necessary to define a common standard suitable for mobile devices' management and elaboration.

SME defines standard models, compliant with SWE, to encode information related to both phone-embedded sensors and application-level sensing events. In this way it allows to implement opportunistic sensing services and their integration with external sensing components. As far as SME Android implementation, it provides to mobile application developers a set of primitives for efficient XML data binding of both SME and SWE XML files and for the interaction with SOS servers (and the efficient elaboration of related data). In order to reduce XML processing and transmission overhead on mobile devices, SME data structures are based on a subset of SWE schemas, appropriately selected for mobile environments. Specifically, SME mainly refers to Sensor-ML and O&M as relevant standards for describing sensors, their possible operations and related measurements, and SOS interfaces to support mobile access to sensor web services. Starting from sensors' descriptions and observations derived from real SOS servers, we defined a subset of SWE XML tags relevant for mobile sensing services and related XML schemas. SWE schemas reduction does not affect the interoperability of the SME with the original standard. In fact, experimental studies on a sample of SOS servers in [19] witnessed that most of them actually use less than 30% of SWE functionalities. Moreover, SME defines data structures for encoding information related to phone-embedded sensors, which are not included in original SWE. In fact, currently the only support to the development of mobile applications involving phone-embedded sensors is the implementation of operating system APIs, which allow developers to recover a predefined set of information (e.g., sensor type, vendor) and to register to the sensing event specifying the desired sampling frequency. SME defines ex-novo XML schemas for all phone-embedded sensors derived from the interaction with Android APIs, and implements a set of Java classes for data binding operations. These schemas contain the minimum set of tags to be compliant with SWE standard.

As far as the interaction of the mobile device with external SWE sensors and repositories, SME implementation presents a two-fold functionality. On the one hand, it implements an efficient data binding procedure of complete SWE XML files derived from download operations. In fact, to reduce XML processing overhead additional fields not defined in SME classes are discarded. On the other hand, SME converts local objects into SME XML files, compliant with SWE, allowing mobile device upload operations to remote servers.

In this way, SME is able to overcome technical limitations of using SWE on mobile devices and to efficiently support the development of opportunistic and participatory sensing services, maintaining their interoperability with SWE standard services.

IV. SME and CAMEO

CAMEO[1] [6], [7] is a mobile middleware platform that provides to mobile applications' developers a set of functionalities for the collection, sharing and elaboration of context information over opportunistic networks, in addition to efficient networking protocols in case of intermittent connectivity. Context information includes data characterizing the user (e.g., personal and social information, like habits, interests, mobility patterns and interactions), his/her mobile device(s) (e.g., information related to internal resources: sensors, capacity, available memory), running applications (e.g. produced and shared contents), and the surrounding environment (e.g., data collected from external sources). CAMEO is in charge of (i) collecting and managing context information of the local node (i.e., the ensemble of the user and his/her mobile devices), (ii) locally defining the social context of the node, derived from the contexts' exchange among neighbors, and its historical profile. CAMEO implements a beaconing procedure that periodically disseminates local context information among 1-hop neighbors, and it provides to upper-layer applications a set of functionalities to access context information and to require data correlations to optimize their features. In this scenario, phone-embedded sensors represent additional sources of information and sharable resources on the opportunistic network. In fact, nodes running CAMEO can ask their neighbors for already available sensors' measurements or for specific sensing operations (e.g., a node not having pressure sensor can ask to one of its neighbors to measure the related data, even if the remote node is not interested in it). To this aim CAMEO, by interacting with SME library, extends the context of the local node by introducing the notion of *sensing context*. In fact, CAMEO directly recovers sensors' information through operating system APIs and exploits SME data structures to initialize the sensing context. Sensing context includes main information related to the available local sensors (i.e., sensor descriptions including both hardware and software capabilities) and their measurements (i.e., observations) and it is disseminated on the network through the beaconing procedure.

[1]Context-Aware MiddlEware for Opportunistic Mobile Social Networks

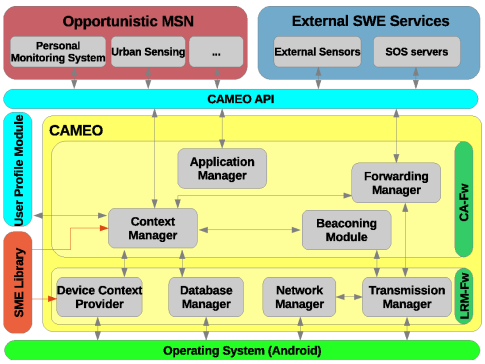

Fig. 2. CAMEO architecture.

In this way, nodes running CAMEO have a complete view of available contents and resources among their neighbors[2].

SME definition of general data structures for descriptions and observations of embedded sensors has been designed to support both existing and possible new integrated devices, supporting thus heterogeneous sensing components even inside the same mobile device. In addition, SME implementation allows CAMEO to interact with external sensing components based on SWE standard, like independent sensors or SOS services. In this way, CAMEO can support both standard upload/download operations to/from SWE services and export local node sensing capabilities to external entities. This allows to include complex sensing systems, like smartphones or tablets, into the emerging paradigm of Sensor Web for IoT [8].

Figure 2 shows the architecture of CAMEO and the interactions between the software library implementing SME (i.e., SME library) and CAMEO internal modules. Specifically, at low level the `Device Context Provider` is in charge of interacting with Android APIs for embedded sensors management. It directly inherits SME data structures to create Java objects that represent sensors' descriptions and observations according to Android information. Java objects are then passed to the `Context Manager`, the core of the middleware, which manages and elaborates all context information. The same objects are also included in CAMEO messages in case of data exchange over the opportunistic network, among nodes running CAMEO. This allows CAMEO to further optimize sensor data exchange over the opportunistic network simply transmitting serialized Java objects instead of SME XML files. On the other hand, Java objects are converted in SME XML files if they must be transmitted to independent nodes.

Referring to CAMEO interactions with external SWE services, CAMEO extends the API provided to upper-layer applications to include SWE standard interfaces (e.g., SOS GetCapabilities, GetDescriptions, GetObservations). When a CAMEO node receives SWE XML files from the network, the `Context Manager` exploits SME data binding procedure to filter relevant information and deserialize the file. Instead, in case CAMEO and/or upper-layer applications need to export local information as SME XML files, the `Context Manager` uses SME deserialization procedures. Even in this case, CAMEO reduces the data transmission towards SWE services, since SME XML files are light-weight codifications with respect to SWE standard.

Therefore, the introduction of SME as reference standard for mobile sensing applications largely extends CAMEO functionalities, both in terms of additional sensing opportunities in mobile networks and interactions with remote standard solutions. In addition, the use of CAMEO as middleware platform to support multiple context-aware mobile applications presents the fundamental advantage of local sharing of multi-dimensional context information among different applications running on the same node. In this way, independent mobile applications can access and correlate heterogeneous context information, provided both by local and external services.

V. Experimental results and Performance evaluation

In order to evaluate real performances of using SME on mobile devices, we performed a set of experiments with Google Galaxy Nexus smartphones (HSPA+), equipped with Android 4.1.1 (Jelly Bean) and 1.2 GHz dual-core processor. Google Galaxy Nexus includes six hardware sensors and seven software sensors (as highlighted in Table I). Hardware sensors are physical components integrated into the device, measuring specific environmental properties. Software sensors represent the elaboration of data derived from one or more hardware sensors. Android manages hardware and software sensors without distinction. It defines `Sensor` objects as the abstraction of sensing operations, involving one or more hardware sensors. `Sensor` objects are characterized by: event type, event name, event description, measurement unit, resolution, sensor name, sensor vendor, maximum range, maximum sampling frequency, consumed power. SME sensor descriptions are defined as data structures reflecting SME adaptation of Sensor-ML standard and they are populated by `Sensor` object values. As far as data derived from sensors' measurements, Android generates `SensorEvent` objects according to the sampling frequency. A `SensorEvent` is characterized by: accuracy, sensor name, timestamp, and values. SME sensor observations are defined as data structures based on SME adaptation of O&M standard and they are populated by Android `SensorEvent` characteristics.

Through real experiments we evaluated SME performances in terms of:
- SME library size with respect to desktop implementations;
- XML processing overhead on mobile devices related to serialization and deserialization procedures of both SME and SWE files;
- impact of SME processing on battery consumption.

A. SME size and XML processing overhead

As a first result it is worth noting that SME library, implementing a light-weight version of SWE standard, highly reduces the size of current desktop implementations, like *JAXB for OGC* project [18]. Specifically, SME requires 38.9KB for

[2]Naturally, the successful implementation of opportunistic resources sharing mechanisms is subject to dynamic constraints of the involved nodes (i.e., local resources availability to host remote service requests).

#	Sensor Name	Type
1	Sharp GP2A Light Sensor	HW
2	Sharp GP2A Proximity Sensor	HW
3	Bosch BMP180 Pressure Sensor	HW
4	Invensense MPL Gyroscope	HW
5	Invensense MPL Accelerometer	HW
6	Invensense MPL Magnetic Field	HW
7	Invensense MPL Orientation	SW
8	Invensense MPL Rotation Vector	SW
9	Invensense MPL Linear Acceleration	SW
10	Invensense MPL Gravity	SW
11	Google Rotation Vector	SW
12	Google Gravity	SW
13	Google Linear Acceleration	SW
14	Google Orientation	SW
15	Google Corrected Gyroscope	SW

TABLE I
GOOGLE NEXUS EMBEDDED SENSORS.

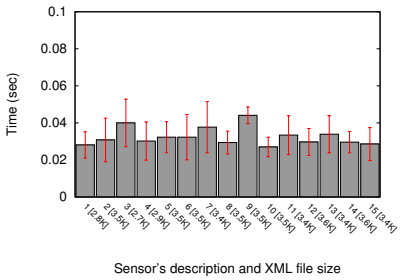

Fig. 3. Scenario 1: mean deserialization time and 95% c.i. over 10 tests. On the x axis we report sensor id and the related XML file size.

classes definitions and additional space for the use of a data binding tool (we used SIMPLE [20] that requires 384KB). On the other hand, the library provided by *JAXB for OCG* requires 3.43MB just for classes definitions, even not supporting the entire SWE standard.

In order to evaluate XML processing overhead introduced by SME standard and data binding procedures, we refer to the classification of XML files provided by W3C [21]. They define a reference parameter called *Content Density (CD)* representing the ratio between the values' size of XML attributes and elements (real content of the file measured in number of characters) and the total size of the XML file (content and XML overhead). XML files are classified into two main categories depending on CD value: *High CD* if CD $> 33\%$ and *Low CD* if CD $< 33\%$. *Low CD* files are additionally divided into *Large* files if size $>$ 100KB, *Small* files if 1KB $<$ size $<$ 100KB and *Tiny* files if size $<$ 1KB.

In our experiments we considered sensors' descriptions and observations derived from local embedded sensors and remote SOS servers. We envisioned three reference scenarios:
1) Two nodes encounters and exchange their own sensors' descriptions and observations encoded in SME standard.
2) The local node encounters up to 10 remote nodes and receives from them sensors' descriptions and observations encoded in SME standard.
3) The local node requests a SOS server for a single observation including an increasing number of values (from 0 to $50,000$ values related to independent sensors' measurements). The received observations are encoded in SWE standard. In a second step, the local node converts the elaborated data in SME standard in order to send it to other remote nodes.

As far as the first scenario, Figure 3 shows the time required by SME library to deserialize the description of each local sensor from SME XML files to Java objects. Sensors are identified by sequential number as in Table I. The first time the local node applies the deserialization procedure to a sensor's description, the processing time includes also the time necessary for loading SME library into memory. On average, the total time is equal to 0.618s with a 95% confidence interval equal to $(0.521\text{s}, 0.7150\text{s})$. Instead, once the library is already loaded, deserialization time for each embedded sensor description is in the range $[0.027\text{s}, 0.044\text{s}]$, as shown in Figure 3. XML file size of each local sensor's description ranges between 2.7KB and 3.5KB with CD between 9.8% and 30.1%. We compared these results with the time needed by SME library to deserialize a real SOS sensor's description. The size of the original SWE codification of the used SOS sensor's description is 10.5KB with CD equal to 23.81% (i.e., small file - Low CD). During SME deserialization procedure, the XML file is filtered, discarding not relevant tags and values. As a result, the size is reduced to $6.6KB$ and the CD increased to 43.48%, becoming thus a High CD XML file. In this case the deserialization time of the original description takes 0.085s on average, with a 95% confidence level (calculated on 10 experiments) in the range $(0.067\text{s}, 0.102\text{s})$. These results witness the advantages of using SME on mobile devices for efficient elaboration of SWE XML files. In fact, SME is able to reduce SWE codifications' size while increasing their content density. In addition, processing time is proportional to the size of resulting SME codification, independently of the original file size.

In the same scenario, we evaluated the processing overhead related to the management of an increasing number of observations exchanged between two nodes. Specifically, we measured both serialization and deserialization times of up to 15 sensors' observations measured on one of the involved nodes. Experimental results are shown in Figure 4. In both cases, the last value represents the worst case in which the local node sequentially processes all the sent/received observations. In case of deserialization, SME library requires at most 0.24s to complete the operation. Instead, in case of serialization it requires at most 1.14s. The difference between these processing times is mainly due to I/O operations of SME encoding. Specifically, read operations on mobile devices equipped with NAND memories are generally faster than write operations. In both cases, SME processing overhead does not heavily affect mobile device performances.

As far as the second scenario, Figure 5 shows experimental results of SME deserialization procedure applied to an increasing number of sensors' descriptions and observations generated by up to 10 remote nodes. Even in this case, SME library performances are really advantageous for mobile devices, requiring up to $1.782s$ and $3.662s$ to deserialize observations and descriptions respectively. The difference between these times is mainly due to the different size of single observations and descriptions of embedded sensors (on average, 1.9KB and 3.347KB respectively).

Results related to scenario 1 show that serialization and deserialization of a relatively high number of small XML files does not overload the mobile system processing capabilities, making SME library able to support mobile sensing

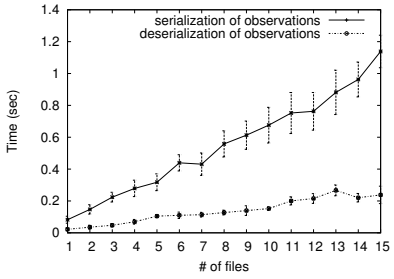

Fig. 4. Scenario 1: XML processing times of an increasing number of observations (up to 15).

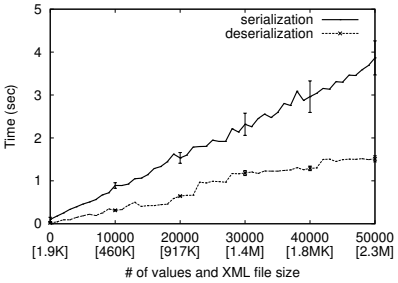

Fig. 6. Scenario 3: XML processing times of a single observation with an increasing number of values.

services that require medium/high frequency I/O operations on XML files. This is an important characteristic in opportunistic computing scenarios, in which limited contact times require efficient processing especially in case of real-time services.

As far as the third scenario, we measured serialization and deserialization times of a real sensor observation derived from a SOS server. We considered an observation containing an increasing number of values related to independent measurements (from 0 up to $50,000$ with steps of 1000). The observation is originally encoded in SWE O&M standard and each value corresponds to a sensing event represented by the following fields: a date, a number in double precision and a string describing the event. The XML file size of the considered observation ranges between 1.9KB (i.e, containing only SWE XML overhead and 0 values) and 2.3MB (i.e., containing $50,000$ values). The content density ranges between 18.4% and 99.9% respectively. Figure 6 highlights the efficiency of SME library to manage complex SWE observations, corresponding to large XML files, in few seconds. Also in this case the difference between serialization and deserialization times is mainly due to I/O operations.

B. Battery Consumption

As last set of experiments, we analyzed the impact of XML processing on the battery consumption of mobile devices. Firstly, we considered power consumption related to continuous execution of serialization and deserialization procedures. At the end of a 15 hours test, the battery level changed from 100% to 98%. Considering the reference scenario involving both a group of CAMEO nodes and external SWE services, we ran a second set of experiments involving the use of CAMEO WiFi communications. We performed 5 tests for one hour each, and the battery level reduced to 97% for all the 5 tests. To evaluate the impact of XML processing on battery consumption with respect to a power consuming service, we performed the previous experiments including also the use of GPS. In this case the battery level reduced to 88% for all the 5 tests. These results clearly show that XML processing does not significantly impact on battery consumption, while the main consumption is related to the use of WiFi, and GPS especially.

All the experimental results demonstrated the efficiency of SME introducing a limited overhead for XML processing on mobile devices.

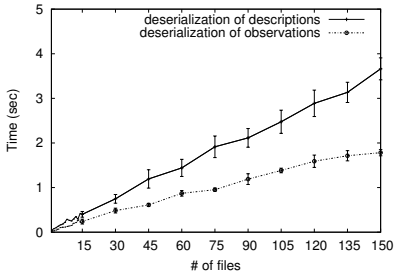

Fig. 5. Scenario 2: deserialization times of an increasing number of observations and descriptions (up to 150).

ACKNOWLEDGMENT

This work was partially funded by the European Commission under the SCAMPI (FP7-FIRE 258414) project.

REFERENCES

[1] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich, "Mobiscopes for human spaces," *IEEE Pervasive Computing*, vol. 6, no. 2, pp. 20–29, 2007.

[2] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell, "A survey of mobile phone sensing," *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 140–150, 2010.

[3] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: data forwarding in disconnected mobile ad hoc networks," *Communications Magazine, IEEE*, vol. 44, no. 11, pp. 134–141, 2006.

[4] M. Conti and M. Kumar, "Opportunities in Opportunistic Computing," *Computer*, vol. 43, no. 1, pp. 42–50, 2010.

[5] "http://www.opengeospatial.org/projects/groups/sensorwebdwg."

[6] V. Arnaboldi, M. Conti, and F. Delmastro, "Implementation of CAMEO: a Context-Aware MiddlEware for Opportunistic Mobile Social Networks," in *IEEE WOWMOM 2012*, 2011.

[7] V. Arnaboldi, M. Conti, and F. Delmastro, "CAMEO: a novel Context-Aware MiddlEware for Opportunistic Mobile Social Networks," Tech. Rep., 2012. [Online]. Available: http://cnd.iit.cnr.it/fdelmastro/pub/techrep/cameo_tr.pdf

[8] "http://www.opengeospatial.org/projects/groups/sweiotswg."

[9] A. Tamayo, C. Granell, and J. Huerta, "Analysing Performance of XML Data Binding Solutions for SOS Applications," in *Proceedings of Workshop on Sensor Web Enablement*, 2011.

[10] A. Tamayo, C. Granell, and J. Huerta, "Using SWE Standards for Ubiquitous Environmental Sensing: A Performance Analysis," *Sensors*, vol. 12, no. 9, pp. 12 026–12 051, 2012.

[11] "http://www.json.org."

[12] "http://www.w3.org/XML/EXI/."

[13] "http://www.w3.org/DOM/."

[14] "http://sax.sourceforge.net/."

[15] "http://www.oracle.com/technetwork/articles/javase/index-140168.html."

[16] "http://xmlbeans.apache.org/."

[17] "http://www.obj-sys.com/xbinder.shtml."

[18] "http://www.ogcnetwork.net/jaxb4ogc."

[19] A. Tamayo, C. Granell, and J. Huerta, "Dealing with large schema sets in mobile SOS-based applications," *arXiv preprint arXiv:1110.0209*, 2011.

[20] "http://simple.sourceforge.net/."

[21] "http://www.w3.org/tr/exi-measurements/."